

Apache Doris 2.0 Documentation (English)

Archived documentation

2026-11-24 · 915 documents

Source: e0fed2a66fcd

Contents

1. What's Apache Doris
2. Quick Start
3. Manual Deployment
4. Deploying with Docker
5. Cluster Environment Requirements
6. Deploying Doris Operator
7. Configuring Doris Cluster
8. Deploying Doris Cluster
9. Accessing Doris Cluster
10. Upgrading Apache Doris cluster deployed by Doris Operator
11. Used by Root users
12. Service expansion and contraction
13. How to enter the container when the service crashes
14. Deploying on AWS
15. Connecting to Database
16. Data Types
17. Table Model Overview
18. Duplicate Key Model
19. Unique Key Model
20. Aggregate Model
21. Usage Notes
22. Hybrid Storage
23. Basic concepts
24. Manual partitioning
25. Dynamic partition
26. Manual bucketing
27. Auto bucket
28. Common Issues
29. Schema Change
30. Tiered Storage
31. Index Overview
32. Sort Key and Prefix Index
33. Inverted Index
34. BloomFilter Index
35. N-Gram BloomFilter Index
36. Best Practices
37. Loading Overview
38. Stream Load

39. Broker Load
40. Routine Load
41. Insert Into
42. MySQL Load
43. Loading Data in JSON Format
44. Migrating Data from Other OLAP
45. Loading Transaction and Atomicity
46. Converting Source Data
47. Minimum Number of Replicas Loading
48. Loading Strict Mode
49. Updating Overview
50. Updating Data with UPDATE Command
51. Updating Data on Unique Key Model
52. Updating Data on Aggregate Key Model
53. Concurrency Control for Updates in the Primary Key Model
54. Deleting Data with DELETE Command
55. Batch Deletion
56. Deleting Data with TRUNCATE Command
57. Replacing Atomic Table
58. Temporary Partition
59. Exporting Overview
60. Exporting Query Result
61. Using MySQL Dump
62. MySQL Compatibility
63. Select Manual
64. Complex Type
65. Subquery
66. Common Table Expressions
67. Column to Row (Lateral View)
68. Window Function
69. Encrption and Masking Function
70. Variable
71. SQL Mode
72. Nereids-the Brand New Planner
73. Statistics
74. Pipeline Execution Engine
75. Query Caches Overview
76. SQL Cache
77. Partition Cache
78. View
79. Materialized View
80. Doris Join Optimization Principle

81. Bucket Shuffle Join
82. Colocation Join
83. Runtime Filter
84. Join Hint using Document
85. Orthogonal BITMAP Calculation
86. Approximate Deduplication Using HLL
87. High-Concurrency Point Query
88. TOPN Query Optimization
89. Statistics of query execution
90. Query Analysis
91. Import Analysis
92. Java UDF
93. Remote User Defined Function Service
94. Lakehouse Overview
95. Hive Catalog
96. Hudi Catalog
97. Iceberg Catalog
98. Paimon Catalog
99. Cloud Service Authentication
100. JDBC Catalog
101. MySQL
102. PostgreSQL
103. Oracle
104. SQL Server
105. ClickHouse
106. SAP HANA
107. OceanBase
108. Elasticsearch
109. MaxCompute
110. Querying Files on S3/HDFS
111. File Caches
112. Elastic Compute Node
113. External Table Statistics
114. Cluster Upgrade
115. Elastic scaling
116. Load Balancing
117. Time Zone
118. FQDN
119. Data Backup
120. Data Restore
121. CCR (Cross Cluster Replication)
122. Recover from Recycle Bin

123. Data Recovery
124. Workload Group
125. Resource Group
126. SQL Interception
127. Kill Query
128. Authority Management
129. MySQL Client Certificate
130. FE SSL Certificate
131. LDAP
132. Authority Management based on Apache Ranger
133. Memory Tracker
134. Memory Limit Exceeded Analysis
135. BE OOM Analysis
136. Monitor Metrics
137. Monitoring and alarming
138. Disk Capacity Management
139. Data Replica Management
140. Tablet metadata management tool
141. Tablet Restore Tool
142. Tablet Local Debug
143. Metadata Operations and Maintenance
144. Automated Service Startup
145. Debug Log
146. Config Dir
147. FE Configuration
148. BE Configuration
149. User Property
150. Audit Log Plugin
151. Config Action
152. HA Action
153. Hardware Info Action
154. Help Action
155. Log Action
156. Login Action
157. Logout Action
158. Query Profile Action
159. Session Action
160. System Action
161. Colocate Meta Action
162. Meta Action
163. Cluster Action
164. Node Action

165. Query Profile Action
166. Backends Action
167. Bootstrap Action
168. Cancel Load Action
169. Check Decommission Action
170. Check Storage Type Action
171. Connection Action
172. Extra Basepath Action
173. Fe Version Info Action
174. Get DDL Statement Action
175. Get Load Info Action
176. Get Load State
177. Get FE log file
178. Get Small File Action
179. Health Action
180. Meta Info Action
181. Meta Replay State Action
182. Metrics Action
183. Profile Action
184. Query Schema Action
185. Query Stats Action
186. Row Count Action
187. Set Config Action
188. Show Data Action
189. Show Meta Info Action
190. Show Proc Action
191. Show Runtime Info Action
192. Show Table Data Action
193. Statement Execution Action
194. Table Query Plan Action
195. Table Row Count Action
196. Table Schema Action
197. Upload Action
198. Import Action
199. Meta Info Action
200. Debug Point
201. Statistic Action
202. Check Stub Cache
203. Reset Stub Cache
204. View Compaction Status
205. Manually Trigger Compaction
206. View Meta

207. Make Snapshot
208. Check All Tablet Segment Lost
209. Config of BE
210. Metrics
211. View Tablet Distribution
212. Migration Tablet
213. View Tablet Info
214. Checksum
215. Download Log about Load Error
216. Pad Rowset
217. Be Version Info
218. Check Alive
219. Reload Tablet
220. Restore Tablet
221. Developing Plugins
222. Small File Manager
223. Compaction
224. Using rowsets
225. Star Schema Benchmark
226. TPC-H Benchmark
227. TPC-DS Benchmark
228. Spark Doris Connector
229. Flink Doris Connector
230. Doris Kafka Connector
231. Doris Streamloader
232. Apache Superset
233. FineBI
234. Power BI
235. Tableau
236. QuickSight
237. Quick BI
238. Smartbi
239. CloudDM
240. DBeaver
241. DataGrip
242. Logstash
243. Filebeat
244. FluentBit
245. BladePipe
246. DataX Doriswriter
247. DBT Doris Adapter
248. Seatunnel Doris Sink

249. Kettle Doris Plugin
250. Kyuubi
251. AutoMQ Load
252. Hive Bitmap UDF
253. Install Error
254. Data Operation Error
255. SQL Error
256. Data Lakehouse FAQ
257. BI FAQ
258. Data Integrity FAQ
259. Overview
260. BOOLEAN
261. TINYINT
262. SMALLINT
263. INT
264. BIGINT
265. LARGEINT
266. FLOAT
267. DOUBLE
268. DECIMAL
269. DATE
270. TIME
271. DATETIME
272. CHAR
273. VARCHAR
274. STRING
275. ARRAY
276. MAP
277. STRUCT
278. JSON
279. HLL (HyperLogLog)
280. BITMAP
281. QUANTILE_STATE
282. AGG_STATE
283. ARRAY
284. ARRAY_MAX
285. ARRAY_MIN
286. ARRAY_MAP
287. ARRAY_FILTER
288. ARRAY_AVG
289. ARRAY_SUM
290. ARRAY_SIZE

291. ARRAY_REMOVE
292. ARRAY_SLICE
293. ARRAY_SORT
294. ARRAY_REVERSE_SORT
295. ARRAY_SORTBY
296. ARRAY_POSITION
297. ARRAY_CONTAINS
298. ARRAY_EXCEPT
299. ARRAY_PRODUCT
300. ARRAY_INTERSECT
301. ARRAY_RANGE
302. ARRAY_DISTINCT
303. ARRAY_DIFFERENCE
304. ARRAY_UNION
305. ARRAY_JOIN
306. ARRAY_WITH_CONSTANT
307. ARRAY_ENUMERATE
308. ARRAY_ENUMERATE_UNIQ
309. ARRAY_POPBACK
310. ARRAY_POPFRONT
311. ARRAY_PUSHFRONT
312. ARRAY_PUSHBACK
313. ARRAY_COMPACT
314. ARRAY_CONCAT
315. ARRAY_ZIP
316. ARRAY_SHUFFLE
317. ARRAY_CUM_SUM
318. ARRAY_EXISTS
319. ARRAY_FIRST_INDEX
320. ARRAY_LAST_INDEX
321. ARRAY_FIRST
322. ARRAY_LAST
323. ARRAYS_OVERLAP
324. ARRAY_COUNT
325. ARRAY_APPLY
326. COUNTEQUAL
327. ELEMENT_AT
328. CONVERT_TZ
329. CURDATE,CURRENT_DATE
330. CURTIME,CURRENT_TIME
331. CURRENT_TIMESTAMP
332. LOCALTIME,LOCALTIMESTAMP

333. NOW
334. YEAR
335. QUARTER
336. MONTH
337. DAY
338. DAYOFYEAR
339. DAYOFMONTH
340. DAYOFWEEK
341. WEEK
342. WEEKDAY
343. WEEKOFYEAR
344. YEARWEEK
345. DAYNAME
346. MONTHNAME
347. HOUR
348. MINUTE
349. SECOND
350. FROM_DAYS
351. LAST_DAY
352. TO_MONDAY
353. FROM_SECOND
354. FROM_UNIXTIME
355. UNIX_TIMESTAMP
356. UTC_TIMESTAMP
357. TO_DATE
358. TO_DAYS
359. TIME_TO_SEC
360. SEC_TO_TIME
361. EXTRACT
362. MAKEDATE
363. STR_TO_DATE
364. TIME_ROUND
365. TIMEDIFF
366. TIMESTAMPADD
367. TIMESTAMPDIFF
368. DATE_ADD
369. DATE_SUB
370. DATE_TRUNC
371. DATE_FORMAT
372. DATEDIFF
373. SECOND_FLOOR
374. MINUTE_FLOOR

375. HOUR_FLOOR
376. DAY_FLOOR
377. MONTH_FLOOR
378. YEAR_FLOOR
379. DATE_CEIL
380. SECOND_CEIL
381. MINUTE_CEIL
382. HOUR_CEIL
383. DAY_CEIL
384. MONTH_CEIL
385. YEAR_CEIL
386. MICROSECOND
387. MICROSECONDS_ADD
388. MICROSECONDS_DIFF
389. MICROSECONDS_SUB
390. MILLISECONDS_ADD
391. MILLISECONDS_DIFF
392. MILLISECONDS_SUB
393. MINUTES_ADD
394. MINUTES_DIFF
395. MINUTES_SUB
396. SECONDS_ADD
397. SECONDS_DIFF
398. SECONDS_SUB
399. HOURS_ADD
400. HOURS_DIFF
401. HOURS_SUB
402. DAYS_ADD
403. DAYS_DIFF
404. DAYS_SUB
405. WEEKS_ADD
406. WEEKS_DIFF
407. WEEKS_SUB
408. MONTHS_ADD
409. MONTHS_DIFF
410. MONTHS_SUB
411. YEARS_ADD
412. YEARS_DIFF
413. YEARS_SUB
414. ST_X
415. ST_Y
416. ST_CIRCLE

417. ST_DISTANCE_SPHERE
418. ST_ANGLE
419. ST_AZIMUTH
420. ST_ANGLE_SPHERE
421. ST_AREA_SQUARE_METERS
422. ST_AREA_SQUARE_KM
423. ST_POINT
424. ST_POLYGON
425. ST_ASTEXT
426. ST_CONTAINS
427. ST_GEOMETRYFROMTEXT
428. ST_LINEFROMTEXT
429. ST_ASBINARY
430. ST_GEOMETRYFROMWKB
431. TO_BASE64
432. FROM_BASE64
433. ASCII
434. LENGTH
435. BIT_LENGTH
436. CHAR_LENGTH
437. LPAD
438. RPAD
439. LOWER
440. LCASE
441. UPPER
442. UCASE
443. INITCAP
444. REPEAT
445. REVERSE
446. CHAR
447. CONCAT
448. CONCAT_WS
449. SUBSTR
450. SUBSTRING
451. SUB_REPLACE
452. APPEND_TRAILING_CHAR_IF_ABSENT
453. ENDS_WITH
454. STARTS_WITH
455. TRIM
456. LTRIM
457. RTRIM
458. NULL_OR_EMPTY

459. NOT_NULL_OR_EMPTY
460. HEX
461. UNHEX
462. ELT
463. INSTR
464. LOCATE
465. FIELD
466. FIND_IN_SET
467. REPLACE
468. LEFT
469. RIGHT
470. STRLEFT
471. STRRIGHT
472. SPLIT_PART
473. SPLIT_BY_STRING
474. SUBSTRING_INDEX
475. MONEY_FORMAT
476. PARSE_URL
477. CONVERT_TO
478. EXTRACT_URL_PARAMETER
479. UUID
480. SPACE
481. ESQUERY
482. MULTI_SEARCH_ALL_POSITIONS
483. MULTI_MATCH_ANY
484. MASK
485. MASK_FIRST_N
486. MASK_LAST_N
487. LIKE
488. NOT LIKE
489. REGEXP
490. REGEXP_EXTRACT
491. REGEXP_EXTRACT_ALL
492. REGEXP_REPLACE
493. REGEXP_REPLACE_ONE
494. STRUCT
495. NAMED_STRUCT
496. STRUCT_ELEMENT
497. STATE
498. MERGE
499. UNION
500. COLLECT_SET

501. MIN
502. STDDEV_SAMP
503. AVG
504. AVG_WEIGHTED
505. PERCENTILE
506. PERCENTILE_ARRAY
507. HLL_UNION_AGG
508. TOPN
509. TOPN_ARRAY
510. TOPN_WEIGHTED
511. COUNT
512. SUM
513. MAX_BY
514. BITMAP_UNION
515. GROUP_BITMAP_XOR
516. GROUP_BIT_AND
517. GROUP_BIT_OR
518. GROUP_BIT_XOR
519. PERCENTILE_APPROX
520. STDDEV,STDDEV_POP
521. GROUP_CONCAT
522. COLLECT_LIST
523. MIN_BY
524. MAX
525. ANY_VALUE
526. VARIANCE_SAMP,VARIANCE_SAMP
527. APPROX_COUNT_DISTINCT
528. VARIANCE,VAR_POP,VARIANCE_POP
529. RETENTION
530. SEQUENCE_MATCH
531. SEQUENCE-COUNT
532. GROUPING
533. GROUPING_ID
534. COUNT_BY_ENUM
535. HISTOGRAM
536. MAP_AGG
537. BITMAP_AGG
538. TO_BITMAP
539. BITMAP_HASH
540. BITMAP_FROM_STRING
541. BITMAP_TO_STRING
542. BITMAP_TO_ARRAY

543. BITMAP_FROM_ARRAY
544. BITMAP_EMPTY
545. BITMAP_OR
546. BITMAP_AND
547. BITMAP_UNION
548. BITMAP_XOR
549. BITMAP_NOT
550. BITMAP_AND_NOT,BITMAP_ANDNOT
551. BITMAP_SUBSET_LIMIT
552. BITMAP_SUBSET_IN_RANGE
553. SUB_BITMAP
554. BITMAP_COUNT
555. BITMAP_AND_COUNT
556. BITMAP_AND_NOT_COUNT,BITMAP_ANDNOT_COUNT
557. BITMAP_XOR_COUNT
558. BITMAP_OR_COUNT
559. BITMAP_CONTAINS
560. BITMAP_HAS_ALL
561. BITMAP_HAS_ANY
562. BITMAP_MAX
563. BITMAP_MIN
564. INTERSECT_COUNT
565. BITMAP_INTERSECT
566. BITMAP_HASH64
567. BITMAP_FROM_BASE64
568. BITMAP_TO_BASE64
569. BITMAP_REMOVE
570. BITAND
571. BITOR
572. BITXOR
573. BITNOT
574. CASE
575. COALESCE
576. IF
577. IFNULL
578. NVL
579. NULLIF
580. JSON_PARSE
581. JSON_EXTRACT
582. JSON_EXISTS_PATH
583. JSON_TYPE
584. JSON_ARRAY

585. JSON_OBJECT
586. JSON_QUOTE
587. JSON_UNQUOTE
588. JSON_VALID
589. JSON_CONTAINS
590. JSON_LENGTH
591. GET_JSON_DOUBLE
592. GET_JSON_INT
593. GET_JSON_BIGINT
594. GET_JSON_STRING
595. JSON_INSERT
596. JSON_REPLACE
597. JSON_SET
598. MAP
599. MAP_KEYS
600. MAP_VALUES
601. MAP_CONTAINS_KEY
602. MAP_CONTAINS_VALUE
603. MURMUR_HASH3_32
604. MURMUR_HASH3_64
605. HLL_CARDINALITY
606. HLL_EMPTY
607. HLL_HASH
608. CONV
609. BIN
610. SIN
611. COS
612. TAN
613. ASIN
614. ACOS
615. ATAN
616. E
617. PI
618. EXP
619. LOG
620. LOG2
621. LOG10
622. LN
623. CEIL
624. FLOOR
625. PMOD
626. ROUND

627. ROUND_BANKERS
628. TRUNCATE
629. ABS
630. SQRT
631. CBRT
632. POW
633. DEGREES
634. RADIANS
635. SIGN
636. POSITIVE
637. NEGATIVE
638. GREATEST
639. LEAST
640. RANDOM
641. MOD
642. RUNNING_DIFFERENCE
643. UUID_NUMERIC
644. AES_ENCRYPT
645. AES_DECRYPT
646. MD5
647. MD5SUM
648. SM4_ENCRYPT
649. SM4_DECRYPT
650. SM3
651. SM3SUM
652. SHA
653. SHA2
654. EXPLODE_JSON_ARRAY
655. EXPLODE
656. EXPLODE_SPLIT
657. EXPLODE_BITMAP
658. NUMBERS
659. EXPLODE_NUMBERS
660. EXPLODE_NUMBERS_OUTER
661. S3
662. HDFS
663. LOCAL
664. ICEBERG_META
665. BACKENDS
666. FRONTENDS
667. FRONTENDS_DISKS
668. CATALOGS

669. WORKLOAD_GROUPS
670. WINDOW FUNCTIONS OVERVIEW
671. WINDOW_FUNCTION_SUM
672. WINDOW_FUNCTION_AVG
673. WINDOW_FUNCTION_MAX
674. WINDOW_FUNCTION_MIN
675. WINDOW_FUNCTION_COUNT
676. WINDOW_FUNCTION_RANK
677. WINDOW_FUNCTION_DENSE_RANK
678. WINDOW_FUNCTION_FIRST_VALUE
679. WINDOW_FUNCTION_LAST_VALUE
680. WINDOW_FUNCTION_LEAD
681. WINDOW_FUNCTION_LAG
682. WINDOW_FUNCTION_ROW_NUMBER
683. WINDOW_FUNCTION_NTILE
684. WINDOW_FUNCTION_WINDOW_FUNNEL
685. IPV4_NUM_TO_STRING
686. INET_NTOA
687. COSINE_DISTANCE
688. INNER_PRODUCT
689. L1_DISTANCE
690. L2_DISTANCE
691. CAST
692. DIGITAL_MASKING
693. WIDTH_BUCKET
694. ALTER-SYSTEM-ADD-FOLLOWER
695. ALTER-SYSTEM-ADD-OBSERVER
696. ALTER-SYSTEM-ADD-BACKEND
697. ALTER-SYSTEM-ADD-BROKER
698. ALTER-SYSTEM-MODIFY-BACKEND
699. ALTER-SYSTEM-DECOMMISSION-BACKEND
700. ALTER-SYSTEM-DROP-FOLLOWER
701. ALTER-SYSTEM-DROP-OBSERVER
702. ALTER-SYSTEM-DROP-BACKEND
703. ALTER-SYSTEM-DROP-BROKER
704. CANCEL-ALTER-SYSTEM
705. CREATE-ROLE
706. CREATE-USER
707. ALTER-USER
708. SET-PASSWORD
709. SET-PROPERTY
710. LDAP

711. GRANT
712. REVOKE
713. DROP-ROLE
714. DROP-USER
715. ADMIN-SHOW-CONFIG
716. ADMIN-SET-CONFIG
717. SET-VARIABLE
718. INSTALL-PLUGIN
719. UNINSTALL-PLUGIN
720. ADMIN SET REPLICA STATUS
721. SHOW REPLICA DISTRIBUTION
722. SHOW REPLICA STATUS
723. ADMIN-REPAIR-TABLE
724. ADMIN-CANCEL-REPAIR
725. ADMIN CHECK TABLET
726. ADMIN DIAGNOSE TABLET
727. ADMIN COPY TABLET
728. SHOW TABLET STORAGE FORMAT
729. ADMIN CLEAN TRASH
730. RECOVER
731. KILL
732. ADMIN-REBALANCE-DISK
733. ADMIN-CANCEL-REBALANCE-DISK
734. ENABLE FEATURE
735. CREATE-CATALOG
736. CREATE-DATABASE
737. CREATE-TABLE
738. CREATE-TABLE-LIKE
739. CREATE-TABLE-AS-SELECT
740. CREATE-EXTERNAL-TABLE
741. CREATE-INDEX
742. CREATE-VIEW
743. CREATE-MATERIALIZED-VIEW
744. CREATE-FUNCTION
745. CREATE-FILE
746. CREATE-POLICY
747. CREATE-ENCRYPT-KEY
748. CREATE-RESOURCE
749. CREATE-WORKLOAD-GORUP
750. CREATE-SQL-BLOCK-RULE
751. ALTER-CATALOG
752. ALTER-DATABASE

753. ALTER-TABLE-COLUMN
754. ALTER-TABLE-PARTITION
755. ALTER-TABLE-ROLLUP
756. ALTER-TABLE-RENAME
757. ALTER-TABLE-REPLACE
758. ALTER-TABLE-PROPERTY
759. ALTER-TABLE-COMMENT
760. CANCEL-ALTER-TABLE
761. ALTER-VIEW
762. ALTER-POLICY
763. ALTER-RESOURCE
764. ALTER-COLOCATE-GROUP
765. ALTER-WORKLOAD-GROUP
766. ALTER-SQL-BLOCK-RULE
767. DROP-CATALOG
768. DROP-DATABASE
769. DROP-TABLE
770. DROP-INDEX
771. DROP-MATERIALIZED-VIEW
772. DROP-FUNCTION
773. DROP-FILE
774. DROP-POLICY
775. DROP-ENCRYPT-KEY
776. DROP-RESOURCE
777. DROP-WORKLOAD-GROUP
778. DROP-DATABASE
779. TRUNCATE-TABLE
780. CREATE-REPOSITORY
781. DROP-REPOSITORY
782. BACKUP
783. CANCEL-BACKUP
784. RESTORE
785. CANCEL-RESTORE
786. STREAM-LOAD
787. MULTI-LOAD
788. BROKER-LOAD
789. MYSQL-LOAD
790. CREATE-ROUTINE-LOAD
791. ALTER-ROUTINE-LOAD
792. PAUSE-ROUTINE-LOAD
793. RESUME-ROUTINE-LOAD
794. STOP-ROUTINE-LOAD

795. CREATE-SYNC-JOB
796. PAUSE SYNC JOB
797. RESUME-SYNC-JOB
798. STOP-SYNC-JOB
799. CANCEL-LOAD
800. CLEAN-LABEL
801. INSERT
802. SELECT
803. DELETE
804. UPDATE
805. EXPORT
806. CANCEL-EXPORT
807. INSERT-OVERWRITE
808. ANALYZE
809. OUTFILE
810. SHOW ALTER TABLE MATERIALIZED VIEW
811. SHOW-ALTER
812. SHOW-ANALYZE
813. SHOW-BACKUP
814. SHOW-BACKENDS
815. SHOW-BROKER
816. SHOW-CATALOGS
817. SHOW-CREATE-TABLE
818. SHOW-CHARSET
819. SHOW-CREATE-CATALOG
820. SHOW-CREATE-DATABASE
821. SHOW-CREATE-MATERIALIZED-VIEW
822. SHOW-CREATE-LOAD
823. SHOW-CREATE-REPOSITORY
824. SHOW-CREATE-ROUTINE-LOAD
825. SHOW-CREATE-FUNCTION
826. SHOW-COLUMNS
827. SHOW-COLUMN-STATS
828. SHOW-COLLATION
829. SHOW-DATABASES
830. SHOW DATA SKEW
831. SHOW-DATABASE-ID
832. SHOW-DYNAMIC-PARTITION
833. SHOW-DELETE
834. SHOW-DATA
835. SHOW-ENGINES
836. SHOW-EVENTS

837. SHOW-EXPORT
838. SHOW-ENCRYPT-KEY
839. SHOW-FUNCTIONS
840. SHOW-TYPECAST
841. SHOW-FILE
842. SHOW-GRANTS
843. SHOW-LAST-INSERT
844. SHOW-LOAD-PROFILE
845. SHOW-LOAD-WARNINGS
846. SHOW-INDEX
847. SHOW PARTITION ID
848. SHOW-SNAPSHOT
849. SHOW-SQL-BLOCK-RULE
850. SHOW-ROUTINE-LOAD
851. SHOW-SYNC-JOB
852. SHOW-WHITE-LIST
853. SHOW-WARNING
854. SHOW TABLET
855. SHOW TABLETS BELONG
856. SHOW-VARIABLES
857. SHOW-PLUGINS
858. SHOW-ROLES
859. SHOW-PROCEDURE
860. SHOW-ROUTINE-LOAD-TASK
861. SHOW-PROC
862. SHOW-TABLE-STATS
863. SHOW-TABLE-STATUS
864. SHOW-REPOSITORIES
865. SHOW-QUERY-PROFILE
866. SHOW-OPEN-TABLES
867. SHOW-TABLETS
868. SHOW-LOAD
869. SHOW-TABLES
870. SHOW-RESOURCES
871. SHOW-workload-GROUPS
872. SHOW-PARTITIONS
873. SHOW-FRONTENDS
874. SHOW FRONTENDS DISKS
875. SHOW-RESTORE
876. SHOW-PROPERTY
877. SHOW-TRIGGERS
878. SHOW-PROCESSLIST

879. SHOW TRASH
880. SHOW-VIEW
881. SHOW-TRANSACTION
882. SHOW-STREAM-LOAD
883. SHOW-STATUS
884. SHOW-TABLE-ID
885. SHOW-SMALL-FILES
886. SHOW-POLICY
887. SHOW-STORAGE-POLICY-USING
888. SHOW-CATALOG-RECYCLE-BIN
889. SHOW-QUERY-STATS
890. IN
891. HELP
892. USE
893. DESCRIBE
894. SWITCH
895. REFRESH
896. SYNC
897. CLEAN-QUERY-STATS
898. FE Configuration
899. CDC Load
900. Ecosystem Overview
901. Deploy Doris on Helm Chart
902. Access the Doris cluster
903. Deploy Doris on Kubernetes
904. Persistent Volume and ConfigMap
905. Building log analysis platform
906. Get Profile
907. Overview
908. SECOND_TIMESTAMP
909. CLEAN-PROFILE
910. SHOW-CONVERT-LIGHT-SCHEMA-CHANGE-PROCESS
911. SHOW-DATA-TYPES
912. SHOW-JOB-TASK
913. SHOW-MIGRATIONS
914. REFRESH-LDAP
915. Bitmap Index

What's Apache Doris

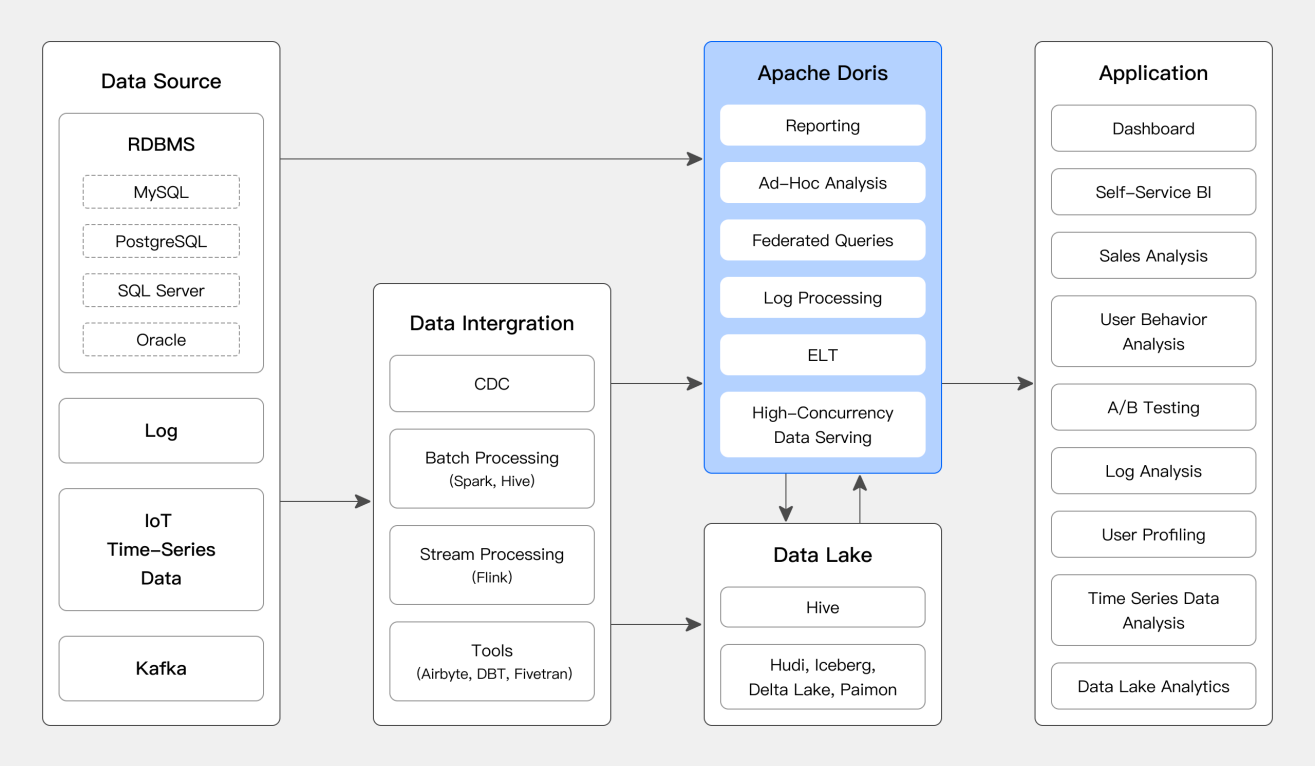
Apache Doris is an MPP-based real-time data warehouse known for its high query speed. For queries on large datasets, it returns results in sub-seconds. It supports both high-concurrent point queries and high-throughput complex analysis. It can be used for report analysis, ad-hoc queries, unified data warehouse, and data lake query acceleration. Based on Apache Doris, users can build applications for user behavior analysis, A/B testing platform, log analysis, user profile analysis, and e-commerce order analysis.

Apache Doris, formerly known as Palo, was initially created to support Baidu's ad reporting business. It was officially open-sourced in 2017 and donated by Baidu to the Apache Software Foundation in July 2018, where it was operated by members of the incubator project management committee under the guidance of Apache mentors. In June 2022, Apache Doris graduated from the Apache incubator as a Top-Level Project. By 2024, the Apache Doris community has gathered more than 600 contributors from hundreds of companies in different industries, with over 120 monthly active contributors.

Apache Doris has a wide user base. It has been used in production environments of over 5000 companies worldwide, including giants such as TikTok, Baidu, Tencent, and NetEase. It is also widely used across industries from finance, retailing, and telecommunications to energy, manufacturing, medical care, etc.

Usage Scenarios

The figure below shows what Apache Doris can do in a data pipeline. Data sources, after integration and processing, are ingested into the Apache Doris real-time data warehouse and offline data lakehouses such as Hive, Iceberg, and Hudi. Apache Doris can be used for the following purposes:



apache-doris-usage-scenarios-pipeline

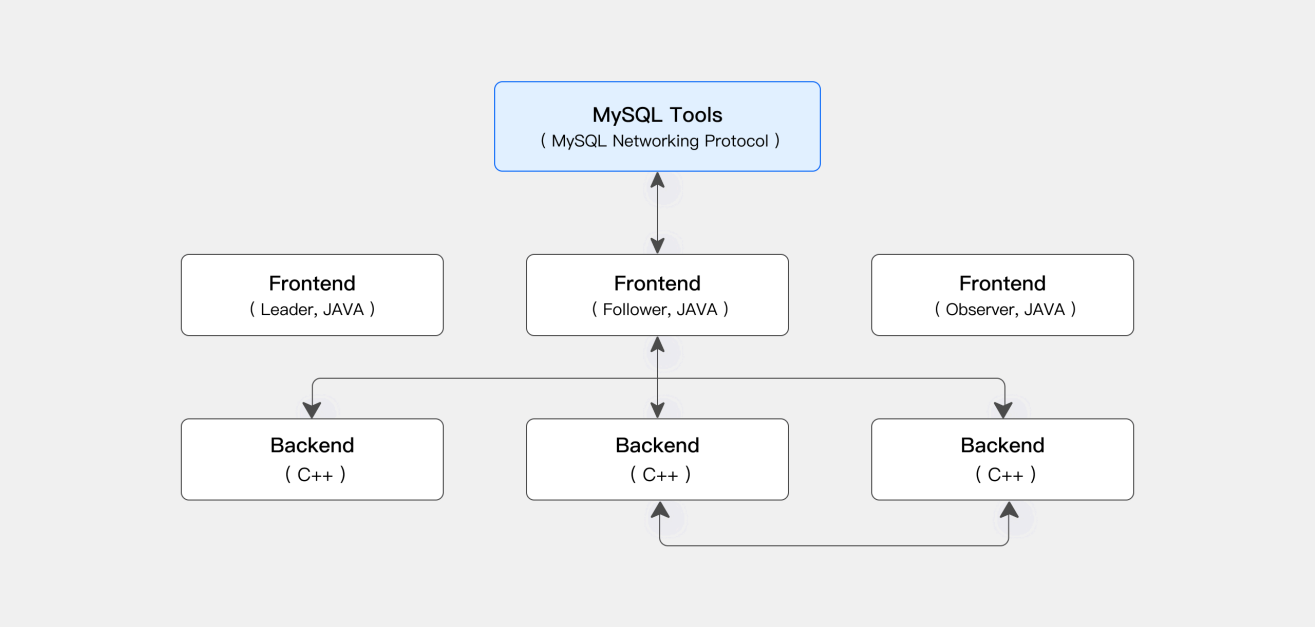
- **Report analysis**
    - Real-time dashboards
    - Reports for internal analysts and managers
    - Customer-facing reports: such as site analysis for website owners and advertising reports for advertisers. Such cases typically require high concurrency (thousands of QPS) and low query latency (measured in milliseconds). For example, the e-commerce giant JD.com uses Apache Doris for ad reporting. It ingests 10 billion rows of data per day and achieves over 10,000 QPS and P99 latency of 150ms.
- **Ad-hoc query**: analyst-facing self-service analytics with irregular query patterns and high throughput requirements. For example, Xiaomi builds a Growth Analytics platform based on Apache Doris. Handling 10,000s of SQL queries every day, it delivers an average query latency of 10 seconds and a P95 latency of 30 seconds.
- **Data Lakehouse**: Apache Doris allows federated queries on external tables in offline data lakehouses such as Hive, Hudi, and Iceberg and achieves outstanding query performance by avoiding data copying.
- **Log analysis**: Apache Doris supports inverted index and full-text search since version 2.0. Relying on its highly efficient query and storage engines, Apache Doris enables 10 times higher cost-effectiveness than common log analytic solutions.
- **Unified data warehouse**: Apache Doris can work as a unified data processing platform for various analytic workloads, saving users from handling complicated data components and tech stacks. For example, Haidilao, a world-renowned chain restaurant, replaces its old architecture consisting of Spark, Hive, Kudu, HBase, and Phoenix with Apache Doris.

Technical overview

Apache Doris has a simple and neat architecture with only two types of processes.

- **Frontend (FE)**: user request processing, query parsing and planning, metadata management, and node management
- **Backend (BE)**: data storage and query execution

Both frontend and backend processes are scalable, supporting up to hundreds of machines and tens of petabytes of storage capacity in a single cluster. Both types of processes guarantee high service availability and high data reliability through consistency protocols. This highly integrated architecture design greatly reduces the operation and maintenance costs of a distributed system.



Technical overview

Interface

Apache Doris adopts the MySQL protocol, supports standard SQL, and is highly compatible with MySQL syntax. Users can access Apache Doris through various client tools and seamlessly integrate it with BI tools, including but not limited to Smartbi, DataEase, FineBI, Tableau, Power BI, and Apache Superset. Apache Doris can work as the data source for any BI tools that support the MySQL protocol.

Storage engine

Apache Doris has a columnar **storage engine**, which encodes, compresses, and reads data by column. This enables a very high data compression ratio and largely reduces unnecessary data scanning, thus making more efficient use of IO and CPU resources.

Apache Doris supports various **index** structures to minimize data scans:

- Sorted Compound Key Index: Users can specify three columns at most to form a compound sort key. This can effectively prune data to better support highly concurrent reporting scenarios.
- Min/Max Index: This enables effective data filtering in equivalence and range queries of numeric types.
- BloomFilter Index: This is very effective in equivalence filtering and pruning of high-cardinality columns.
- Inverted Index: This enables fast searching for any field.

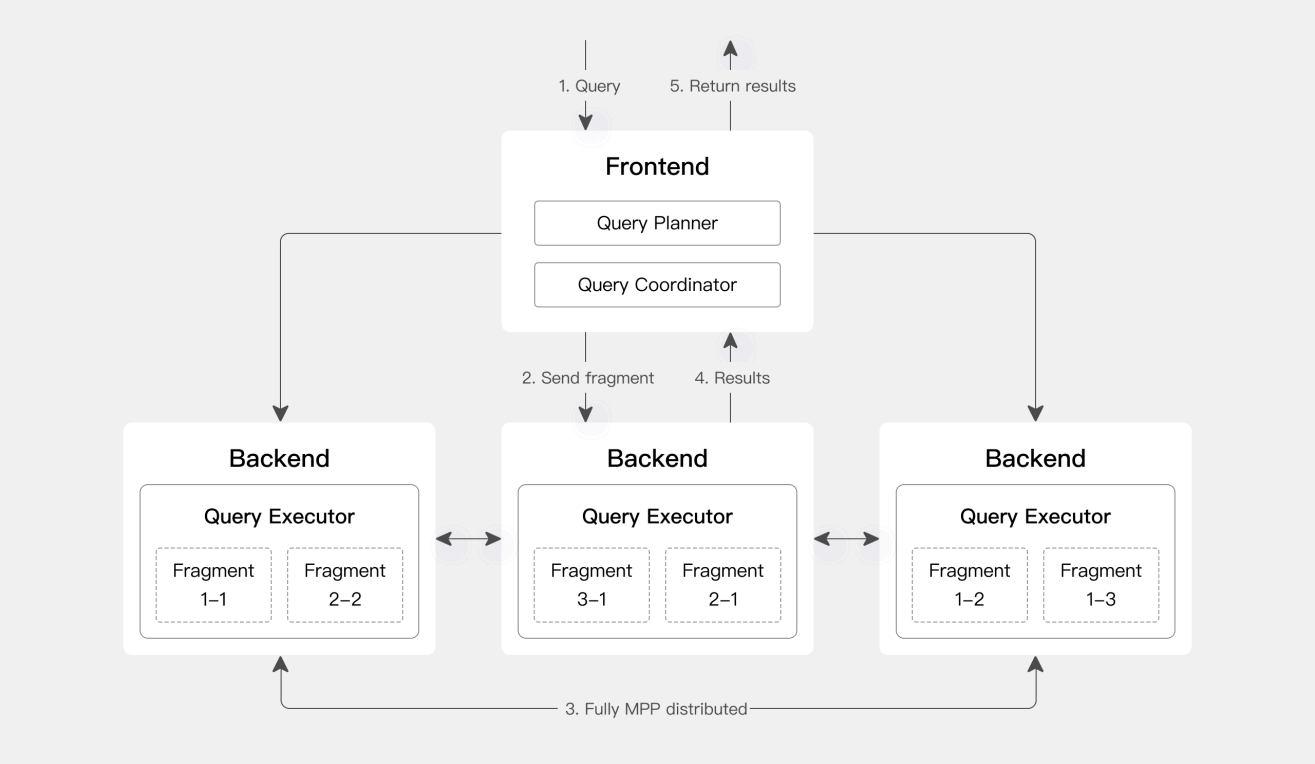
Apache Doris supports a variety of **data models** and has optimized them for different scenarios:

- Aggregate Key Model: merges the value columns with the same keys and improves performance by pre-aggregation
- Unique Key Model: ensures uniqueness of keys and overwrites data with the same key to achieve row-level data updates
- Duplicate Key Model: stores data as it is without aggregation, capable of detailed storage of fact tables

Apache Doris also supports **strongly consistent materialized views**. Materialized views are automatically selected and updated within the system without manual efforts, which reduces maintenance costs for users.

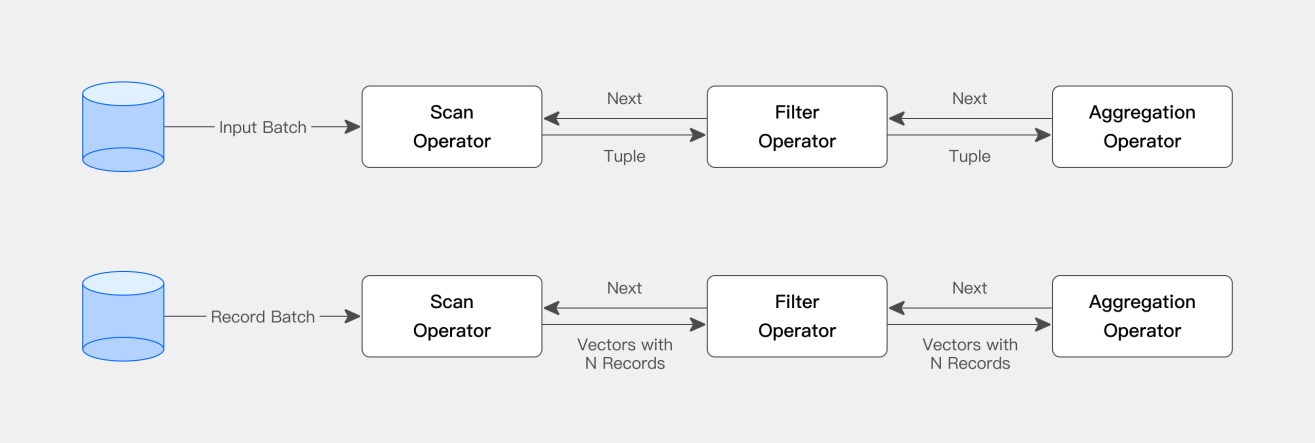
Query engine

Apache Doris has an MPP-based query engine for parallel execution between and within nodes. It supports distributed shuffle join for large tables to better handle complicated queries.



Query engine

The query engine of Apache Doris is fully vectorized, with all memory structures laid out in a columnar format. This can largely reduce virtual function calls, increase cache hit rates, and make efficient use of SIMD instructions. Apache Doris delivers a 5~10 times higher performance in wide table aggregation scenarios than non-vectorized engines.



Query engine

Apache Doris uses **adaptive query execution** technology to dynamically adjust the execution plan based on runtime statistics. For example, it can generate a runtime filter and push it to the probe side. Specifically, it pushes the filters to the lowest-level scan node on the probe side, which largely reduces the data amount to be

processed and increases join performance. The runtime filter of Apache Doris supports In/Min/Max/Bloom Filter.

The query **optimizer** of Apache Doris is a combination of CBO and RBO. RBO supports constant folding, subquery rewriting, and predicate pushdown while CBO supports join reorder. The Apache Doris CBO is under continuous optimization for more accurate statistics collection and inference as well as a more accurate cost model.

Quick Start

Quick Start

This guide is about how to download the latest stable version of Apache Doris, install it on a single node, and get it running, including steps for creating a database, data tables, importing data, and performing queries.

Environment requirements

- A mainstream Linux x86-64 environment. CentOS 7.1 or Ubuntu 16.04 or later versions are recommended. See the “Install and Deploy” section of the doc for guides on more environments.
- Install Java 8 runtime environment. (If you are not an Oracle JDK commercial license user, we suggest using the free Oracle JDK 8u202. Download now.)
- It is recommended to create a new user for Apache Doris on Linux (avoid using the root user to prevent accidental operations on the operating system).

Download binary package

Download the Apache Doris installation package from doris.apache.org and proceed with the following steps.

```
# Download the binary installation package of Apache Doris
server1:~ doris$ wget https://download.velodb.io/apache-doris-2.0.3-bin-x64.tar.gz

# Extract the installation package
server1:~ doris$ tar zxf apache-doris-2.0.3-bin-x64.tar.gz

# Rename the directory to apache-doris for simplicity
server1:~ doris$ mv apache-doris-2.0.3-bin-x64 apache-doris
```

Install Apache Doris

Configure FE

Go to the `apache-doris/fe/conf/fe.conf` file for FE configuration. Below are some key configurations to pay attention to. Add JAVA_HOME manually and point it to your JDK8 runtime environment. For other configurations, you can go with the default values for a quick single-machine experience.

```
# Add JAVA_HOME and point it to your JDK8 runtime environment. Suppose your JDK8 is at /home/doris/jdk8, set it as follows:
JAVA_HOME=/home/doris/jdk8

# The CIDR network segment of FE listening IP is empty by default. When started, Apache Doris will automatically select an available
network segment. If you need to specify a segment, you can set priority_networks=192.168.0.0/24, for example.
# priority_networks =

# By default, FE metadata is stored in the doris-meta directory under DORIS_HOME. It is created already. You can change it to your
specified path.
# meta_dir = ${DORIS_HOME}/doris-meta
```

Start FE

Run the following command under apache-doris/fe to start FE.

```
# Start FE in the background to ensure that the process continues running even after exiting the terminal.
server1:apache-doris/fe doris$ ./bin/start_fe.sh --daemon
```

Configure BE

Go to the `apache-doris/be/conf/be.conf` file for BE configuration. Below are some key configurations to pay attention to. Add JAVA_HOME manually and point it to your JDK8 runtime environment. For other configurations, you can go with the default values for a quick single-machine experience.

```
# Add JAVA_HOME and point it to your JDK8 runtime environment. Suppose your JDK8 is at /home/doris/jdk8, set it as follows:
JAVA_HOME=/home/doris/jdk8

# The CIDR network segment of BE listening IP is empty by default. When started, Doris will automatically select an available
network segment. If you need to specify a segment, you can set priority_networks=192.168.0.0/24, for example.
# priority_networks =

# By default, BE data is stored in the storage directory under DORIS_HOME. It is created already. You can change it to your
specified path.
# storage_root_path = ${DORIS_HOME}/storage
```

Start BE

Run the following command under apache-doris/be to start BE.

```
# Start BE in the background to ensure that the process continues running even after exiting the terminal.
server1:apache-doris/be doris$ ./bin/start_be.sh --daemon
```

Connect to Doris FE

Download the portable MySQL client to connect to Doris FE.

Unpack the client, find the `mysql` command-line tool in the `bin/` directory. Then execute the following command to connect to Apache Doris.

```
mysql -uroot -P9030 -h127.0.0.1
```

Note:

- The root user here is the built-in super admin user of Apache Doris. See Authentication and Authorization for more information.
- -P: This specifies the query port that is connected to. The default port is 9030. It corresponds to the `query_port`setting in fe.conf.
- -h: This specifies the IP address of the FE that is connected to. If your client and FE are installed on the same node, you can use 127.0.0.1.

Add BE nodes to cluster

An example SQL to execute in the MySQL client to add BE nodes to the cluster:

```
ALTER SYSTEM ADD BACKEND "be_host_ip:heartbeat_service_port";
```

Note:

1. be_host_ip: the IP address of the BE node to be added
2. heartbeat_service_port: the heartbeat reporting port of the BE node to be added, which can be found in `be.conf`under `heartbeat_service_port`, set as `9050` by default
3. You can use the “show backends” statement to view the newly added BE nodes.

Modify passwords for root and admin

Example SQLs to execute in the MySQL client to set new passwords for root and admin users:

```
mysql> SET PASSWORD FOR 'root' = PASSWORD('doris-root-password');
Query OK, 0 rows affected (0.01 sec)

mysql> SET PASSWORD FOR 'admin' = PASSWORD('doris-admin-password');
Query OK, 0 rows affected (0.00 sec)
```

> **Difference between root and admin users**
>
> The root and admin users are two default accounts that are automatically created after Doris installation. The root user has superuser privileges for the entire cluster and can perform various management operations, such as adding or removing nodes. The admin user does not have administrative privileges but is a superuser within the cluster, possessing all permissions except those related to cluster management. It is recommended to use the root privileges only when necessary for cluster administration and maintenance.

Create database and table

Connect to Apache Doris

Use admin account to connect to Apache Doris FE.

```
mysql -uadmin -P9030 -h127.0.0.1
```

> **If the MySQL client connecting to 127.0.0.1 is on the same machine as FE, no password will be required.**

Create database and table

```
create database demo;

use demo;
create table mytable
(
    k1 TINYINT,
    k2 DECIMAL(10, 2) DEFAULT "10.05",
    k3 CHAR(10) COMMENT "string column",
    k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
)
COMMENT "my first table"
DISTRIBUTED BY HASH(k1) BUCKETS 1
PROPERTIES ('replication_num' = '1');
```

Ingest data

Save the following example data to the local “data.csv” file:

```
1,0.14,a1,20
2,1.04,b2,21
3,3.14,c3,22
4,4.35,d4,23
```

Load the data from “data.csv” into the newly created table using the Stream Load method.

```
curl  --location-trusted -u admin:admin_password -T data.csv -H "column_separator:,"
http://127.0.0.1:8030/api/demo/mytable/_stream_load
```

- -T data.csv: data file name
- -u admin:admin_password: admin account and password
- 127.0.0.1:8030: IP and http_port of FE

Once it is executed successfully, a message like the following will be returned:

```
{
    "TxnId": 30,
    "Label": "a56d2861-303a-4b50-9907-238fea904363",
    "Comment": "",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 4,
    "NumberLoadedRows": 4,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 52,
    "LoadTimeMs": 206,
    "BeginTxnTimeMs": 13,
    "StreamLoadPutTimeMs": 141,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 7,
    "CommitAndPublishTimeMs": 42
}
```

- `NumberLoadedRows`: the number of rows that have been loaded
- `NumberTotalRows`: the total number of rows to be loaded
- `Status`: “Success” means data has been loaded successfully.

Query data

Execute the following SQL in the MySQL client to query the loaded data:

```
mysql> select * from mytable;
+------+------+------+------+
| k1   | k2   | k3   | k4   |
+------+------+------+------+
|    1 | 0.14 | a1   |   20 |
|    2 | 1.04 | b2   |   21 |
|    3 | 3.14 | c3   |   22 |
|    4 | 4.35 | d4   |   23 |
+------+------+------+------+
4 rows in set (0.01 sec)
```

Stop Apache Doris

Stop FE

Execute the following command under apache-doris/fe to stop FE.

```
server1:apache-doris/fe doris$ ./bin/stop_fe.sh
```

Stop BE

Execute the following command under apache-doris/be to stop BE.

```
server1:apache-doris/be doris$ ./bin/stop_be.sh
```

Manual Deployment

Manually deploying a Doris cluster typically involves four steps:

1. Check hardware and software environment: check the hardware resources available to the user and ensure compatibility with the operating system
2. Check operating system: review the operating system parameters and configurations
3. Plan the cluster: plan the FE and BE nodes of the cluster and estimate the required resources
4. Deploy cluster: deploy the cluster based on the planning and configuration
5. Verify: log in to the cluster and check if everything is right

1. Check hardware and software environment

Check hardware

CPU

When installing Doris, it is recommended to use machines that support the AVX2 instruction set to leverage its vectorization capabilities for query acceleration.

Run the following command. If a result is returned, that means the machine supports the AVX2 instruction set.

```
cat /proc/cpuinfo | grep avx2
```

If AVX2 instruction set is not supported, you can deploy Doris using the “noavx2” installation package.

Memory

Doris does not impose strict memory limits. In a production environment, it is generally recommended to have the number of GB of memory at least four times the number of CPU cores (e.g., a machine with 16 cores should have at least 64GB of memory). With more memory space comes better performance.

Storage

When deploying Doris, data can be stored on SSDs, HDDs, or object storage.

It is recommended to use SSD as the data storage in the following scenarios:

- High-concurrency point queries with large data volumes.
- High-frequency data update with large data volumes.

File system

Doris supports both ext4 and xfs file systems.

Network Card

During the computation process, Doris shards and distributes data to different instances for parallel processing, which incurs network overheads. To optimize Doris performance and minimize network overheads, it is strongly

recommended to deploy Doris using 10 Gigabit Ethernet (10 GbE) or faster network cards.

Recommended servers

Doris can run on servers with x86-64 architecture or ARM64 architecture.

Development and testing environment

Module	CPU	Memory	Disk	Network	Instance (minimum)
Frontend	8+ Cores	8+ GB	SSD or SATA, 10+ GB	1 or 10 GbE	1
Backend	8+ Cores	16+ GB	SSD or SATA, 50+ GB	1 or 10 GbE	1

Description:

- In a validation testing environment, you can deploy the FE and BE on the same server.
- It is generally recommended to deploy only **one BE instance on a single machine**. Meanwhile, you can only deploy **one FE on one machine**.
- If you need three replicas of data, you will need at least three machines and deploy one BE instance on each of them, instead of deploying three BE instances on one single machine.
- **The clocks of multiple servers hosting FEs must be synchronized, allowing a maximum clock deviation of 5 seconds.**
- In a testing environment, you can also perform tests with just one BE. In the actual production environment, the number of BE instances directly affects the overall query latency.

Production environment

Module	CPU	Memory	Disk	Network	Instance (minimum)
Frontend	16+ Cores	64+ GB	SSD or RAID, 100+ GB	10 GbE	1
Backend	16+ Cores	64+ GB	SSD or SATA, 100+ GB	10 GbE	3

Description:

- In a production environment, if the FE and BE are co-located, be mindful of resource contention issues. It is recommended to store metadata and data on separate disks.
- BE nodes can be configured with multiple disks for storage. You can bind multiple HDDs or SSDs to a single BE instance.
- The performance of the cluster is dependent on the resources of the BE nodes. The more BE nodes there are, the better the performance of Doris. Typically, Doris can fully leverage its performance potential with 10 to 100 machines.

Hard disk space calculation

In a Doris cluster, FE is mainly responsible for metadata storage, including metadata edit log and image. The disk space of BE mainly stores data, which needs to be calculated according to business needs.

Component	Disk space description
FE	Generally, the size of metadata ranges from several hundred MB to several GB. The recommended disk space is not to be less than 100GB.
BE	Doris uses LZ4 compression by default, with a compression ratio of 0.3~0.5.Disk space needs to be calculated based on the total data volume * 3 (3 data replicas)There is a need to reserve 40% disk space for background data compaction and temporary data storage.
Broker	If you want to deploy a Broker, you can usually deploy the Broker node on the same machine as the FE /BE nodes.

Java version

All Doris processes depend on Java.

Before version 2.1 (inclusive), please use Java 8, recommended version: `openjdk-8u352-b08-linux-x64`.

After version 3.0 (inclusive), please use Java 17, recommended version: `jdk-17.0.10_linux-x64_bin.tar.gz`.

2. Check operating system

Disable swap partition

When deploying Doris, it is recommended to disable the swap partition. The swap partition is used by the kernel to move some memory data to a configured swap space when it detects memory pressure. However, since the kernel's strategy may not fully understand the behavior of the application, it can significantly impact the performance of the database. Therefore, it is advisable to disable it.

You can **temporarily disable** it using the following command. It will be enabled again upon the next machine boot.

```
swapoff -a
```

To **permanently disable** it, use the Linux root account to comment out the swap partition in /etc/fstab, and then reboot.

```
# /etc/fstab
# <file system>        <dir>         <type>    <options>             <dump> <pass>
tmpfs                  /tmp          tmpfs     nodev,nosuid          0      0
/dev/sda1              /             ext4      defaults,noatime      0      1
# /dev/sda2            none          swap      defaults              0      0
/dev/sda3              /home         ext4      defaults,noatime      0      2
```

> **It is not recommended to disable swap by setting vm.swappiness = 0 because this parameter has different semantics in different Linux kernel versions. In many cases, this will not completely disable swap.**

Check and disable system firewall

If you find inaccessible ports,try disabling the firewall to confirm whether it is the cause. If it is, open the corresponding ports for communication based on the configured ports for Doris components.

```
sudo systemctl stop firewalld.service
sudo systemctl disable firewalld.service
```

Configure NTP service

Doris metadata requires a time precision of less than 5000ms, so clock synchronization is required for all machines in the cluster. This prevents metadata inconsistency from causing service abnormalities.

Typically, you can ensure clock synchronization across nodes by NTP service configuration.

```
sudo systemctl start ntpd.service
sudo systemctl enable ntpd.service
```

Set the maximum number of open file handles

Doris relies on a large number of files to manage table data. Thus, the system's limit on the number of files opened by programs needs to be increased.

```
vi /etc/security/limits.conf
* soft nofile 1000000
* hard nofile 1000000
```

> **For the changes to take effect, the current user needs to log out of the current session and log back in.**

Modify the number of virtual memory areas

Set the number of virtual memory areas to at least 2000000.

```
sysctl -w vm.max_map_count=2000000
```

Disable transparent huge pages

It is recommended to disable transparent huge pages when deploying Doris.

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

3. Plan the cluster

Plan the ports

Doris instances communicate directly over the network, requiring the following ports for normal operation. Administrators can adjust Doris ports according to their environment:

Instance	Port	Default Port	Communication Direction	Description
BE	be_port	9060	FE –> BE	thrift server port on BE, receiving requests from FE
BE	webserver_port	8040	BE <–> BE	http server port on BE
BE	heartbeat_service_port	9050	FE –> BE	heartbeat service port (thrift) on BE, receiving heartbeats from FE
BE	brpc_port	8060	FE <–> BE，BE <–> BE	brpc port on BE, used for communication between BEs
FE	http_port	8030	FE <–> FE，Client <–> FE	http server port on FE
FE	rpc_port	9020	BE –> FE，FE <–> FE	thrift server port on FE, configuration of each FE should be consistent
FE	query_port	9030	Client <–> FE	MySQL server port on FE
FE	edit_log_port	9010	FE <–> FE	port on FE for bdbje communication
Broker	broker_ipc_port	8000	FE –> Broker，BE –> Broker	thrift server on Broker, receiving requests

Plan the nodes

Plan the FE nodes

FE nodes are primarily responsible for handling user requests, query parsing and planning, metadata management, and node management.

For a production cluster, it is generally recommended to deploy at least 3 FE nodes for high availability. In Doris, there are two roles of FE nodes:

- Follower nodes participate in elections. If the Master node fails, an available Follower node will be selected as the new Master.
- Observer nodes only synchronize metadata from the Leader node and do not participate in elections. They can be scaled to provide scalable metadata read services.

Typically, it is advised to deploy 3 Follower nodes. In high-concurrency scenarios, you can scale the Observer nodes..

Plan the BE nodes

BE nodes are responsible for data storage and computation. In a production environment, it is common to have 3 data replicas, so it is recommended to deploy at least 3 BE nodes.

BE nodes can be scaled, too. By increasing the number of BE nodes, you can improve query performance and concurrency.

4. Deploy cluster

Deploy FE Master node

Create metadata path

Generally, FE metadata is no larger than 10GB. It is recommended to store FE metadata on a different disk than the BE node data.

After extracting the installation package, there is usually a default doris-meta directory included. It is advisable to create a separate metadata directory and create a symbolic link from that directory to doris-meta. **In a production environment, it is strongly recommended to specify a separate directory instead of placing it within the Doris installation directory, preferably on a dedicated disk (preferably SSD). For testing and development environments, you can simply use the default configurations.**

```
## Select a disk that is separated from the BE data storage and create the FE metadata directory.
mkdir -p <doris_meta_created>

## Create a symbolic link to the FE metadata directory.
rm -rf <doris_meta_original>
ln -s <doris_meta_created> <doris_meta_original>
```

Modify FE configuration file

The FE configuration file is in the “conf” directory under the FE deployment path. Before starting the FE node, you need to modify the `conf/fe.conf` file.

Here are some necessary modifications:

1. Modify FE metadata directory: In the configuration file, `meta_dir` specifies the location where metadata is stored. By default, `meta_dir` is under the FE installation path. If you have created a symbolic link to the FE metadata directory, you don’t need to configure `meta_dir`.
2. Bind Cluster IP: In an environment with multiple network interfaces, you need to explicitly configure the “priority_networks”.

```
priority_networks = 10.1.3.0/24
```

This is a CIDR representation that specifies the IP used by the FE. In environments using FQDN, you can ignore this.

3. Adjust FE Memory

4. In “fe.conf”, the default maximum Java heap memory is set to 8GB. It is recommended to increase it to 16GB or higher for production environments. You can adjust the maximum Java heap memory by specifying the `-Xmx` in the `JAVA_OPTS` parameter.

5. ```
   JAVA_OPTS="-Xmx16384m -XX:+UseMembar -XX:SurvivorRatio=8 -XX:MaxTenuringThreshold=7 -XX:+PrintGCDateStamps -
   XX:+PrintGCDetails -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSClassUnloadingEnabled -XX:-CMSParallelRemarkEnabled -
   ```
   ```

```
XX:CMSInitiatingOccupancyFraction=80 -XX:SoftRefLRUPolicyMSPerMB=0 -Xloggc:$DORIS_HOME/log/fe.gc.log.$DATE"
```

6. Modify the case sensitivity parameter `lower_case_table_names` By default, Doris is case-sensitive for table names. If you require case-insensitive table names, you need to set this during cluster initialization. Note that once the cluster initialization is completed, the table name case sensitivity cannot be changed. Please refer to the variable documentation for more details on the `lower_case_table_names` setting.

Start FE process

Start FE process by executing the following command

```
bin/start_fe.sh --daemon
```

After FE process is started, it runs in the background. The log files are stored by default in the `log/` directory. In the case of startup failure, you can check the error messages in `log/fe.log` or `log/fe.out`.

Check FE status

To check the FE startup status, you can connect to the Doris cluster using a MySQL client. The initial user is `root` with an empty password.

```
mysql -uroot -P<fe_query_port> -h<fe_ip_address>
```

After connecting to the Doris cluster, you can use the `show frontends` command to check the status of the FE nodes. It is important to verify the following:

- If “Alive” is true, the node is alive.
- If “Join” is true, the node has joined the cluster, but it may not be currently active within the cluster (it may be disconnected).
- If “IsMaster” is true, that means the node is the Master node.

Deploy FE cluster (optional)

In a production cluster, it is recommended to deploy at least three Follower nodes. After deploying the FE Master node, you need to deploy two additional FE Follower nodes.

Create the metadata directory

Refer to the deployment of the FE Master node and create the “doris-meta” directory.

Modify FE Follower node configuration

Refer to the deployment of the FE Master node and modify the FE configuration file. In most cases, you can directly copy the configuration file from the FE Master node.

Register the New FE Follower node in the Doris cluster

Before starting the new FE node, you need to register it in the FE cluster.

```
## Connect to any FE node alive
mysql -uroot -P<fe_query_port> -h<fe_ip_address>
```

```
## Register FE Follower node
## fe_edit_log_port can be viewed in fe.conf, set to 9010 by default
## In MySQL Client, execute ALTER SYSTEM statement
ALTER SYSTEM ADD FOLLOWER "<fe_ip_address>:<fe_edit_log_port>"
```

If you need to add an Observer node, you can use the `ADD OBSERVER` command.

```
## Register FE observer node, execute ALTER SYSTEM in MySQL Client.
ALTER SYSTEM ADD OBSERVER "<fe_ip_address>:<fe_edit_log_port>"
```

> **1. It is recommended to have an odd number of FE Follower nodes (including the Master node), with a suggested number of three to achieve high availability.**
>
> 2. When FE is deployed in a high availability mode (1 Master node, 2 Follower nodes), it is advisable to increase the scalability of the FE read services by adding an Observer FE.
> 3. Typically, one FE node can handle around 10-20 BE nodes. It is recommended to keep the total number of FE nodes below 10.

Start FE Follower node

Execute the following command to start FE Follower node and synchronize metadata automatically.

```
bin/start_fe.sh --helper <helper_fe_ip>:<fe_edit_log_port> --daemon
```

`helper_fe_ip` represents any node alive in the current FE cluster. The `--heper`parameter is only applied during the initial FE startup to synchronize metadata, and it is not necessary to specify it in subsequent FE restart operations.

Check Follower node status

To check the status of a Follower node, follow the same method of checking the status of the FE Master node. After registering the FE Follower node, use the `show frontends` command to view the FE node status. Unlike the Master node, the `IsMaster` status should be false for the Follower node.

Deploy BE

Create directory

BE processes are responsible for data computation and storage. The data directories are by default under `be/storage` . In a production environment, it is common to use dedicated disks to store data, separating the BE data from the BE deployment files. BE supports distributing data across multiple disks to better utilize the I/O capabilities of the disks.

```
## Create BE data storage directories on each data disk
mkdir -p <be_storage_root_path>
```

Modify BE configuration

The configuration file for BE is in the “conf” directory under the BE deployment path. Before starting the BE node, you need to modify the `conf/be.conf` file.

1. Configure Java environment Starting from version 1.2, Doris supports Java UDF (User-Defined Function), and BE relies on the Java environment. You need to configure the `JAVA_HOME` environment variable in the operating system beforehand or specify the Java environment variable in the BE configuration file.

```
## Modify Java environment variable in be/conf/be.conf
JAVA_HOME = <your-java-home-path>
```

2. Configure BE storage path

To modify the BE storage path, you can modify the “storage_root_path” parameter. Separate multiple paths using a semicolon ; (**Do not add a semicolon after the last directory.**)

Tiered Storage for hot and cold data

Doris supports tiered storage for hot and cold data, where cold data is stored on HDDs or object storage, and hot data is stored on SSDs.

You can differentiate the directories for storing hot and cold data within a node by using HDD (for cold data directory) or SSD (for hot data directory) paths. If you don’t require the hot/cold mechanism within the BE node, you only need to configure the paths without specifying the medium type. You also don’t need to modify the default storage medium configuration in FE.

When using the tiered storage feature, you need to specify the `medium` in the `storage_root_path` parameter.

```
## In storage_root_path, use medium to specify the disk type
## /home/disk1/doris,medium:HDD: this directory stores cold data;
## /home/disk2/doris,medium:SSD: this directory stores hot data;
storage_root_path=/home/disk1/doris,medium:HDD;/home/disk2/doris,medium:SSD
```

> **1. When specifying the storage type for the storage path, at least one path should have the storage type set as HDD.**
>
> 2. If the storage type for a storage path is not explicitly declared, it defaults to HDD.
> 3. Specifying the storage type as HDD or SSD is to differentiate the storage types for the paths. It is independent of the physical storage medium. For example, you can designate a directory on an HDD disk as SSD.
> 4. The keywords for storage types, HDD and SSD, must be capitalized.

6. Bind cluster IP

In a multi-network interface environment, it is necessary to explicitly configure the “priority_networks”. However, in an environment where FQDN is configured, the “priority_networks” can be ignored.

```
priority_networks = 10.1.3.0/24
```

Register BE node in Doris

Before starting a new BE node, you need to register the new BE node in the FE cluster.

```
## Connect to any FE node alive
mysql -uroot -P<fe_query_port> -h<fe_ip_address>

## Register BE node
## be_heartbeat_service_port can be checked from be.conf, set to 9050 by default
## In MySQL Client, execute ALTER SYSTEM statement
ALTER SYSTEM ADD BACKEND "<be_ip_address>:<be_heartbeat_service_port>"
```

Start BE process

Start BE process by executing the following command:

```
bin/start_be.sh --daemon
```

After BE process is started, it runs in the background. The log files are stored by default in the `log/` directory. In the case of startup failure, you can check the error messages in `log/be.log` or `log/be.out`.

Check BE status

After connecting to the Doris cluster, you can use the `show backends` command to check the status of the BE nodes.

```
## Connect to Doris cluster
mysql -uroot -P<fe_query_port> -h<fe_ip_address>

## Check BE status, execute show command in MySQL Client
show backends;
```

Normally, you should verify the following:

- If “Alive” is true, the node is alive.
- “TabletNum” represents the number of tablets (shards) on that node. When a new node is added, data balancing occurs, and the TabletNum gradually approaches the average.

Validate cluster configuration

Log in to the database

Log in to the Doris cluster using a MySQL Client.

```
## Connect to Doris cluster
mysql -uroot -P<fe_query_port> -h<fe_ip_address>
```

Check the version

Check the versions using the `show frontends` and `show backends` commands.

```
## Check the versions of all FE instances in MySQL Client
show frontends \G

## Check the versions of all BE instances in MySQL Client
show backends \G
```

Modify cluster password

After creating a Doris cluster, the system automatically creates a `root` user with an empty default password. It is recommended to set a new password for the `root` user after creating the cluster.

```
## To confirm that the current user is root, you can check the current user in the MySQL client.
select user();
+-----------------------+
| user()                |
+-----------------------+
| 'root'@'192.168.88.30' |
+-----------------------+

## To modify the password for the root user, execute the "set password" command in the MySQL client.
SET PASSWORD = PASSWORD('doris_new_passwd');
```

Create a test table and insert data

In the newly created cluster, you can create a table and insert data into it to verify if the cluster works properly.

```
## To create a test database, execute the "create database" statement in MySQL Client.
create database testdb;

## To create a test table, execute the "create table" statement in MySQL Client.
CREATE TABLE testdb.table_hash
(
    k1 TINYINT,
    k2 DECIMAL(10, 2) DEFAULT "10.5",
    k3 VARCHAR(10) COMMENT "string column",
    k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
)
COMMENT "my first table"
DISTRIBUTED BY HASH(k1) BUCKETS 32;
```

Doris is compatible with the MySQL protocol, so you can insert data using the “insert” statement.

```
## To insert test data, execute the "insert into" statement in MySQL Client.
INSERT INTO testdb.table_hash VALUES
(1, 10.1, 'AAA', 10),
(2, 10.2, 'BBB', 20),
(3, 10.3, 'CCC', 30),
(4, 10.4, 'DDD', 40),
(5, 10.5, 'EEE', 50);

## To verify the data, execute the "select" statement in MySQL Client.
SELECT * from testdb.table_hash;
+------+-------+------+------+
| k1   | k2    | k3   | k4   |
+------+-------+------+------+
|    3 | 10.30 | CCC  |   30 |
|    4 | 10.40 | DDD  |   40 |
|    5 | 10.50 | EEE  |   50 |
```

```
|    1 | 10.10 | AAA  |   10 |
|    2 | 10.20 | BBB  |   20 |
+------+-------+------+------+
```

5. FAQ

What is priority_networks?

Doris processes listen to network segments represented in CIDR format for IP addresses. If the deployed machines only have one network segment, no configuration is needed. However, if there are two or more network segments, it is essential to configure them.

This parameter is mainly used to help the system select the correct network card IP as its listening IP. For example, if the required listening IP is 192.168.0.1, you can set `priority_networks=192.168.0.0/24`. The system will automatically scan all IPs on the machine, and only those matching the 192.168.0.0/24 network segment will be used as service listening addresses. You can also configure multiple CIDR network segments in this parameter, such as `priority_networks = 10.10.0.0/16; 192.168.0.0/24`.

> ****Why use priority_networks to configure listening address segments instead of directly setting the listening IP address in the configuration file?****
>
> The main reason is that Doris is a distributed cluster, and the same configuration file will be deployed on multiple nodes. To facilitate deployment, updates, and maintenance, it is desirable to keep all nodes' configuration files consistent. By configuring the listening address segment and then starting the system, the appropriate listening IP can be found based on this network segment, thus each machine can use a single value in this configuration.

Why do new BE nodes need to be manually added to the cluster?

After the BE node is started, you need to send a command to the FE through the MySQL protocol or the built-in web console to join the BE node to the cluster.

> ****How does the FE know which BE nodes constitute the cluster?****
>
> As a distributed database, Doris generally has many BE nodes. Doris adds BE nodes to the cluster by sending commands to the FE. This is different from the way how BE nodes know the addresses of FE nodes and then actively report connections. The way of manual adding and having the FE actively connect to BE nodes is beneficial to cluster management in many aspects. For example, it can determine which nodes constitute the cluster and it can proactively shut down a BE node that cannot be connected to.

How to quickly detect whether the FE has started successfully?

You can use the following command to check if the FE has started successfully.

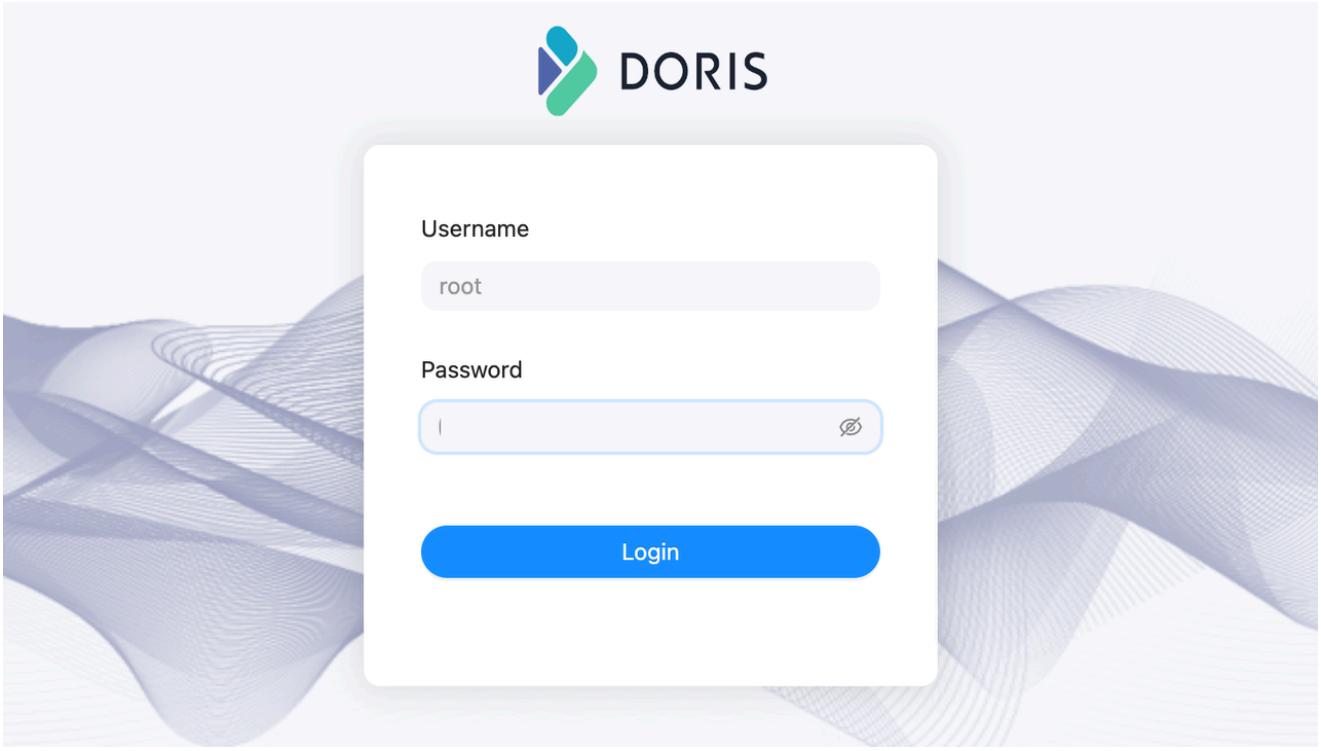
```
# Retry the following command, if it returns "msg":"success", the FE has started successfully.
server1:apache-doris/fe doris$ curl http://127.0.0.1:8030/api/bootstrap
{"msg":"success","code":0,"data":{"replayedJournalId":0,"queryPort":0,"rpcPort":0,"version":""},"count":0}
```

Does Doris provide built-in Web UI?

Doris FE has a built-in Web UI. Users do not need to install a MySQL client to perform tasks such as adding BE/FE nodes or running other SQL queries.

To access the Web UI, simply enter the URL in a web browser: http://fe_ip:fe_port, for example, `http://172.20.63.118:8030`. This will open the built-in Web console of Doris.

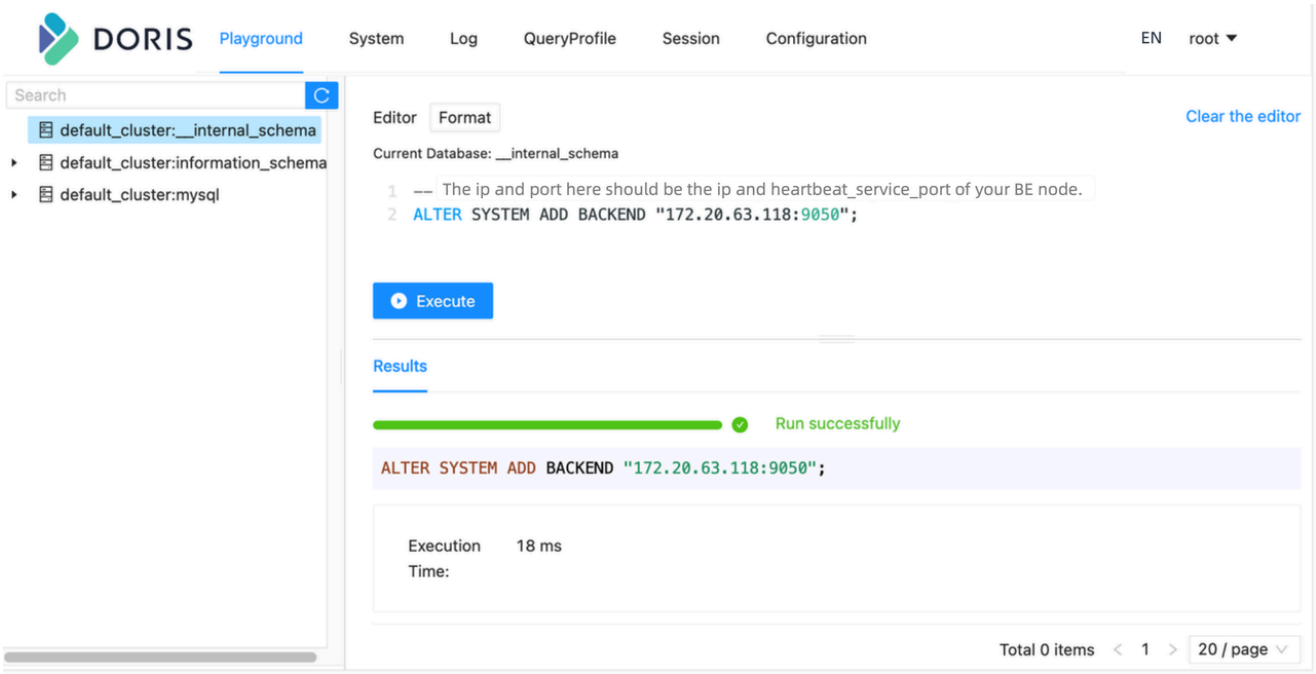
The built-in Web console is primarily intended for use by the root account of the cluster. By default, the root account password is empty after installation.



web-login-username-password

For example, you can execute the following command in the Playground to add a BE node.

```
ALTER SYSTEM ADD BACKEND "be_host_ip:heartbeat_service_port";
```



Doris-Web-UI-Playground-en

> **For successful execution of statements that are not related to specific databases/tables in the Playground, it is necessary to randomly select a database from the left-hand database panel. This limitation will be removed later.**

Why can't I change the root password via the Web UI?

The current built-in web console cannot execute SET type SQL statements. Therefore, in the web console, you cannot change the root password using statements like `SET PASSWORD FOR 'root' = PASSWORD('your_password')`. You can use MySQL Client to change the password for the root account.

Deploying with Docker

Deploy with Docker

This guide is about how to create a running image of Apache Doris using a Dockerfile. It allows for the quick pulling of an Apache Doris image to create and run a cluster, either with container orchestration tools or for a quick test.

Prerequisites

Overview

Prepare the build machine before creating the Docker image. The platform architecture of this machine determines the platform architecture for which the Docker image will be applicable. For example, if you use an X86_64 machine, you should download the X86_64 Doris binary program, because the resulting image will only run on X86_64 platforms. The same applies to ARM64 platforms.

Hardware

Recommended configuration: 4 cores, 16GB memory.

Software

Docker version: 20.10 or later.

Build image

Preparation

Dockerfile script writing

- Choose a Docker Hub-certified OpenJDK official image as the base parent image. The recommended base parent image is: openjdk:8u342-jdk (JDK 1.8 version).
- Use embed scripts for FE startup, multi-FE registration, FE status checks, BE startup, BE registration in FE, and BE status checks.
- Avoid using the `--daemon` option to start the application within Docker, as it may cause issues during deployment with orchestration tools like Kubernetes (K8s).

Methods to build

About the Dockerfile script used for compiling the Docker image, there are two ways to load the Apache Doris binary package:

- Use `wget` or `curl` commands to download the package during compilation and then complete the Docker Build process.
- Pre-download the binary package to the build directory and load it into the Docker Build process using the `ADD` or `COPY` command.

The former method produces a Docker image of smaller size, but if the build fails, the download operation may be repeated, leading to longer build times. The latter method is suitable for environments with unstable network conditions. Here, we will provide an example using the second method.

Build FE Image

1. Environment directory for building the FE image:

The build environment directory is as follows:

```
└─── docker-build                                  // Root directory
    └─── fe                                        // FE directory
        ├─── dockerfile                            // Dockerfile script
        └─── resource                              // Resource directory
            ├─── init_fe.sh                        // Startup and registration script
            └─── apache-doris-2.0.3-bin.tar.gz     // Binary package
```

2. Download binary package

Download the official binary package or the compiled binary package, and replace the apache-doris package in `./docker-build/fe/resource` with it.

3. Write Dockerfile

```
# Choose a base image
FROM openjdk:8u342-jdk

# Set environment variables
ENV JAVA_HOME="/usr/local/openjdk-8/"
ENV PATH="/opt/apache-doris/fe/bin:$PATH"

# Download the software into the Docker image
ADD ./resource/apache-doris-2.0.3-bin.tar.gz /opt/

RUN apt-get update && \
    apt-get install -y default-mysql-client && \
    apt-get clean && \
    mkdir /opt/apache-doris && \
    cd /opt && \
    mv apache-doris-2.0.3-bin/fe /opt/apache-doris/

ADD ./resource/init_fe.sh /opt/apache-doris/fe/bin
RUN chmod 755 /opt/apache-doris/fe/bin/init_fe.sh

ENTRYPOINT ["/opt/apache-doris/fe/bin/init_fe.sh"]
```

- Rename the file as `Dockerfile` and save it to the `./docker-build/fe` directory.
- For the FE execution script init_fe.sh, refer to init_fe.sh in the Doris source code library.

4. Perform build

Note that `${tagName}` should be replaced with the tag you need, such as: `apache-doris:2.0.3-fe`.

```
cd ./docker-build/fe
docker build . -t ${fe-tagName}
```

Build BE image

1. Environment directory for building the BE image:

```
└─── docker-build                                  // Root directory
    └─── be                                        // BE directory
        ├─── dockerfile                            // Dockerfile script
        └─── resource                              // Resource directory
            ├─── init_be.sh                        // Startup and registration script
            └─── apache-doris-2.0.3-bin.tar.gz     // Binary package
```

2. Write Dockerfile

```
# Choose a base image
FROM openjdk:8u342-jdk

# Set environment variables
ENV JAVA_HOME="/usr/local/openjdk-8/"
ENV PATH="/opt/apache-doris/be/bin:$PATH"

# 下Download the software into the Docker image
ADD ./resource/apache-doris-2.0.3-bin.tar.gz /opt/

RUN apt-get update && \
    apt-get install -y default-mysql-client && \
    apt-get clean && \
    mkdir /opt/apache-doris && \
    cd /opt && \
    mv apache-doris-2.0.3-bin/be /opt/apache-doris/

ADD ./resource/init_be.sh /opt/apache-doris/be/bin
RUN chmod 755 /opt/apache-doris/be/bin/init_be.sh

ENTRYPOINT ["/opt/apache-doris/be/bin/init_be.sh"]
```

- Rename the file as `Dockerfile` and save it to the `./docker-build/be` directory.
- For the BE execution script init_be.sh, refer to init_be.sh.

3. Perform build

Note that `${tagName}` should be replaced with the tag you need, such as: `apache-doris:2.0.3-be`.

```
cd ./docker-build/be
docker build . -t ${be-tagName}
```

Push the image to DockerHub or a private repository

Log in to DockerHub

```
docker login
```

Upon successful login, a “Success” prompt will be displayed. After that, you can push the image.

```
docker push ${tagName}
```

Deploy Docker cluster

The following is a brief overview of how to quickly create a complete Doris testing cluster using the `docker run` or `docker-compose up` commands.

It is advisable to avoid containerized solutions for Doris deployment in production environments. Instead, when deploying Doris on Kubernetes (K8s), it is recommended to utilize the Doris Operator for deployment.

Prerequisite

Software

Software	Version
Docker	20.0 and later
docker-compose	20.1 and later

Hardware

Configuration	Hardware	Maximum Running Cluster Size
Minimum	2C 4G	1FE 1BE
Recommended	4C 16G	3FE 3BE

Execute the following command in the host machine:

```
sysctl -w vm.max_map_count=2000000
```

Docker Compose

The required image varies depending on the platform. The following takes the `X86_64` platform as an example.

Network mode

Doris Docker supports two network modes:

- The HOST mode is suitable for deploying across multiple nodes, with one FE and one BE per node.
- The subnet bridge mode is suitable for deploying multiple Doris processes on a single node (recommended). If you want to deploy across multiple nodes, additional component deployments are required (not recommended).

For demonstration purposes, this section will only show scripts written for the subnet bridge mode.

Interface description

Since `Apache Doris 2.0.3 Docker Image`, the interface list for each process image is as follows:

Process	Interface	Interface Definition	Interface Example
FE	BE	BROKER	FE_SERVERS
FE	FE_ID	FE node ID	1
BE	BE_ADDR	BE node informatioin	172.20.80.5:9050
BE	NODE_ROLE	BE node type	computation

Note that the above interfaces must be specified with relevant information; otherwise, the process will not start.

> The FE_SERVERS interface follows the rule:
> `FE_NAME:FE_HOST:FE_EDIT_LOG_PORT[,FE_NAME:FE_HOST:FE_EDIT_LOG_PORT]`
>
> The FE_ID interface should be an integer from `1` to `9`, where `1` represents the Master node.
>
> The BE_ADDR interface follows the rule: `BE_HOST:BE_HEARTBEAT_SERVICE_PORT`
>
> The NODE_ROLE interface should be `computation` or empty. When it is empty or any other value, it indicates a `mix` node.
>
> The BROKER_ADDR interface follows the rule: `BROKER_HOST:BROKER_IPC_PORT`

Script template

Docker Run command

1 FE & 1 BE command template

Note that you should replace `${INTERNAL_IP_OF_CURRENT_MACHINE}` with the internal IP of your current machine.

```
docker run -itd \
--name=fe \
--env FE_SERVERS="fe1:${INTERNAL_IP_OF_CURRENT_MACHINE}:9010" \
--env FE_ID=1 \
-p 8030:8030 \
-p 9030:9030 \
-v /data/fe/doris-meta:/opt/apache-doris/fe/doris-meta \
-v /data/fe/log:/opt/apache-doris/fe/log \
--net=host \
apache/doris:2.0.3-fe-x86_64

docker run -itd \
--name=be \
--env FE_SERVERS="fe1:${INTERNAL_IP_OF_CURRENT_MACHINE}:9010" \
--env BE_ADDR="${INTERNAL_IP_OF_CURRENT_MACHINE}:9050" \
-p 8040:8040 \
-v /data/be/storage:/opt/apache-doris/be/storage \
-v /data/be/log:/opt/apache-doris/be/log \
--net=host \
apache/doris:2.0.3-be-x86_64
```

Download the Docker Run command template for 3 FE & 3 BE from here if needed.

Docker Compose scripte

1 FE & 1 BE template

Note that you should replace `${INTERNAL_IP_OF_CURRENT_MACHINE}` with the internal IP of your current machine.

```
version: "3"
services:
  fe:
    image: apache/doris:2.0.3-fe-x86_64
    hostname: fe
    environment:
     - FE_SERVERS=fe1:${INTERNAL_IP_OF_CURRENT_MACHINE}:9010
     - FE_ID=1
    volumes:
     - /data/fe/doris-meta/:/opt/apache-doris/fe/doris-meta/
     - /data/fe/log/:/opt/apache-doris/fe/log/
    network_mode: host
  be:
    image: apache/doris:2.0.3-be-x86_64
    hostname: be
    environment:
     - FE_SERVERS=fe1:${INTERNAL_IP_OF_CURRENT_MACHINE}:9010
     - BE_ADDR=${INTERNAL_IP_OF_CURRENT_MACHINE}:9050
    volumes:
     - /data/be/storage/:/opt/apache-doris/be/storage/
     - /data/be/script/:/docker-entrypoint-initdb.d/
    depends_on:
      - fe
    network_mode: host
```

Download the Docker Compose command template for 3 FE & 3 BE from here if needed.

Deploy Doris Docker

Choose one of the following deployment methods:

1. Execute the `docker run` command to create the cluster.
2. Save the `docker-compose.yaml` script and execute the `docker-compose up -d`command in the same directory to create the cluster.

Cluster Environment Requirements

Software version requirements

Software	Version Requirements
Docker	>= 1.20
Kubernetes	>= 1.19
Doris	>= 2.0.0
Helm (optional)	>= 3.7

Operating system requirements

Firewall configuration

When deploying a Doris cluster on Kubernetes, it is recommended to turn off the firewall configuration:

```
systemctl stop firewalld
systemctl disable firewalld
```

If the firewall service cannot be turned off, you can open the FE and BE ports according to your plan:

> **Tip**
>
> If the firewall cannot be turned off, you need to open the firewall of the corresponding Doris port according to the Kubernetes mapping rules. For specific ports, please refer to Doris Cluster Port Planning.

Disable and close swap

When deploying Doris, it is recommended to turn off the swap partition.

The swap partition can be permanently shut down with the following command.

```
echo "vm.swappiness = 0">> /etc/sysctl.conf
swapoff -a && swapon -a
sysctl -p
```

Set the maximum number of open file handles in the system

```
vi /etc/security/limits.conf
* soft nofile 65536
* hard nofile 65536
```

Modify the number of virtual memory areas

Modify the virtual memory area to at least 2000000

```
sysctl -w vm.max_map_count=2000000
```

Close transparent large page

When deploying Doris, it is recommended to turn off transparent huge pages.

```
echo madvise > /sys/kernel/mm/transparent_hugepage/enabled
echo madvise > /sys/kernel/mm/transparent_hugepage/defrag
```

Deploying Doris Operator

Add Doris Cluster resource definition

Doris Operator extends Kubernetes with Custom Resource Definition (CRD). The CRD of Doris Cluster encapsulates the description of Doris objects, such as the description of FE or BE. For details, please refer to doris-operator-api. Before deploying Doris, you need to create the CRD of Doris Cluster.

Doris Cluster CRD can be deployed in a Kubernetes environment with the following command:

```
kubectl create -f https://raw.githubusercontent.com/apache/doris-
operator/master/config/crd/bases/doris.selectdb.com_dorisclusters.yaml
```

If there is no external network, download the CRD file to your local computer first:

```
wget https://raw.githubusercontent.com/apache/doris-operator/master/config/crd/bases/doris.selectdb.com_dorisclusters.yaml
kubectl create -f ./doris.selectdb.com_dorisclusters.yaml
```

The following is the expected output:

```
customresourcedefinition.apiextensions.k8s.io/dorisclusters.doris.selectdb.com created
```

After the Doris Cluster CRD is created, you can view the created CRD with the following command.

```
kubectl get crd | grep doris
```

The following is the expected output:

```
dorisclusters.doris.selectdb.com                    2024-02-22T16:23:13Z
```

Add Doris Operator

Solution 1: Quickly deploy Doris Operator

You can directly pull the Doris Operator template in the warehouse for quick deployment.

Doris Operator can be deployed in a Kubernetes cluster using the following command:

```
kubectl apply -f https://raw.githubusercontent.com/apache/doris-operator/master/config/operator/operator.yaml
```

The following is the expected output:

```
namespace/doris created
role.rbac.authorization.k8s.io/leader-election-role created
rolebinding.rbac.authorization.k8s.io/leader-election-rolebinding created
```

```
clusterrole.rbac.authorization.k8s.io/doris-operator created
clusterrolebinding.rbac.authorization.k8s.io/doris-operator-rolebinding created
serviceaccount/doris-operator created
deployment.apps/doris-operator created
```

Option 2: Customized deployment of Doris Operator

After creating the CRD, there are two ways to deploy Doris Operator on the Kubernetes cluster: online and offline deployment.

The minimum requirements for deploying operator services are specified in the operator.yaml file. In order to adapt to complex production environments, you can download the operator.yaml file and update the configuration as desired.

Install Doris Operator online

After modifying the operator.yaml file, you can deploy the Doris Operator service using the following command:

```
kubectl apply -f ./operator.yaml
```

The following is the expected output:

```
namespace/doris created
role.rbac.authorization.k8s.io/leader-election-role created
rolebinding.rbac.authorization.k8s.io/leader-election-rolebinding created
clusterrole.rbac.authorization.k8s.io/doris-operator created
clusterrolebinding.rbac.authorization.k8s.io/doris-operator-rolebinding created
serviceaccount/doris-operator created
deployment.apps/doris-operator created
```

Offline installation of Doris Operator

1. Download the image file required to run the operator

If the server is not connected to the external network, you need to download the corresponding operator image file first. Doris Operator uses the following images:

```
selectdb/doris.k8s-operator:latest
```

Run the following command on a server that can connect to the external network to download the image:

```
## download doris operator image
docker pull selectdb/doris.k8s-operator:latest
## save the doris operator image as a tar package
docker save -o doris.k8s-operator-latest.tar selectdb/doris.k8s-operator:latest
```

Place the packaged tar file into all Kubernetes nodes and run the following command to upload the image:

```
docker load -i doris.k8s-operator-latest.tar
```

2. Configure Doris Operator

After downloading the operator.yaml file, you can modify the template according to your production environment expectations.

Doris Operator is a stateless Deployment in the Kubernetes cluster. Items such as `limits`, `replica`, `label`, `namespace` and other items can be modified according to needs. If you need to specify a certain version of the doirs operator image, you can make the following modifications to the operator.yaml file after uploading the image:

```
...
containers:
  - command:
      - /dorisoperator
    args:
      - --leader-elect
    image: selectdb/doris.k8s-operator:v1.0.0
    name: dorisoperator
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop:
          - "ALL"
  ...
```

3. Install Doris Operator

After modifying the Doris Operator template, you can use the apply command to deploy the Operator:

```
kubectl apply -f ./operator.yaml
```

The following is the expected output:

```
namespace/doris created
role.rbac.authorization.k8s.io/leader-election-role created
rolebinding.rbac.authorization.k8s.io/leader-election-rolebinding created
clusterrole.rbac.authorization.k8s.io/doris-operator created
clusterrolebinding.rbac.authorization.k8s.io/doris-operator-rolebinding created
serviceaccount/doris-operator created
deployment.apps/doris-operator created
```

Option 3: Helm deploys Doris Operator

Helm Chart is an encapsulation of a series of YAML files that describe Kubernetes-related resources. When deploying an application through Helm, you can customize the application's metadata to facilitate application distribution. Chart is a Helm package in TAR format for deploying Kubernetes native applications. Helm Chart can simplify the process of deploying Doris clusters.

1. Add a deployment warehouse

Add warehouse online

Add the remote repository through the `repo add` command

```
helm repo add doris-repo https://charts.selectdb.com
```

Update the latest version of the chart through the `repo update` command

```
helm repo update doris-repo
```

2. Install Doris Operator

Doris Operator can be installed in the doris namespace using the default configuration through the `helm install` command.

```
helm install operator doris-repo/doris-operator
```

If you need to customize the assembly values.yaml, you can refer to the following command:

```
helm install -f values.yaml operator doris-repo/doris-operator
```

Check the deployment status of Pods through the `kubectl get pods` command. When the Doris Operator's Pod is in the Running state and all containers in the Pod are ready, the deployment is successful.

```
kubectl get pod --namespace doris
```

The return result is as follows:

```
NAME                              READY   STATUS    RESTARTS   AGE
doris-operator-866bd449bb-zl5mr   1/1     Running   0          18m
```

Add warehouse offline

If the server cannot connect to the external network, you need to download the chart resources of Doris Operator and Doris Cluster in advance.

1. Download offline chart resources

Download `doris-operator-{chart_version}.tgz` to install Doris Operator chart. If you need to download version 1.4.0 of Doris Operator, you can use the following command:

```
wget https://charts.selectdb.com/doris-operator-1.4.0.tgz
```

2. Install Doris Operator

Doris Operator can be installed through the `helm install` command.

```
helm install operator doris-operator-1.4.0.tgz
```

If you need to customize the assembly values.yaml, you can refer to the following command:

```
helm install -f values.yaml operator doris-operator-1.4.0.tgz
```

Check the deployment status of Pods through the `kubectl get pods` command. When the Doris Operator's Pod is in the Running state and all containers in the Pod are ready, the deployment is successful.

```
kubectl get pod --namespace doris
```

The return result is as follows:

```
NAME                              READY   STATUS    RESTARTS   AGE
doris-operator-866bd449bb-zl5mr   1/1     Running   0          18m
```

Check service status

After deploying the Operator service, you can check the service status through the following command.

```
kubectl get pod -n doris
```

The return result is as follows:

```
NAME                              READY   STATUS    RESTARTS   AGE
doris-operator-6f47594455-p5tp7   1/1     Running   0          11s
```

You need to ensure that the STATUS status is Running and the status of all containers in the pod is Ready.

Configuring Doris Cluster

Configuration data and persistent storage

In a Doris cluster, components including FE, BE, CN, and monitoring components all need to persist data to physical storage. Kubernetes provides Persistent Volumes the ability to persist data to physical storage. In a Kubernetes environment, there are two main types of Persistent Volumes:

- Local PV storage (Local Persistent Volumes): Local PV is where Kubernetes directly uses the local disk directory of the host to persistently store container data. Local PV provides smaller network latency and can provide better read and write capabilities when using high-performance hard drives such as SSDs. Since the local PV is bound to the host, when the host fails, the local PV undergoes fault drift.
- Network PV storage (Network Persistent Volumes): Network PV is a storage resource accessed through the network. The network PV can be accessed by any node in the cluster. When the host fails, the network PV can be mounted to other nodes and continued to be used.

StorageClass can be used to define the type and behavior of PV. StorageClass can decouple disk resources from containers to achieve data persistence and reliability. In Doris Operator, deploying Doris on Kubernetes can support local PV and network PV, and you can choose according to business needs.

> **Warning**
>
> It is recommended to persist data to storage at deployment time. If PersistentVolumeClaim is not configured during deployment, Doris Operator will use emptyDir mode by default to store metadata, data, and logs. When the pod is restarted, related data will be lost.

Persistence directory type

In Doris, the following directories are recommended for persistent storage:

- FE node: doris-meta, log
- BE node: storage, log
- CN node: storage, log
- Broker node: log

There are multiple log types in Doris, such as INFO log, OUT log, GC log and audit log. Doris Operator can output logs to the console and the specified directory at the same time. If the user's Kubernetes has complete log collection capabilities, Doris' INFO logs can be collected through console output. It is recommended that all Doris logs be persisted to the designated storage through PVC configuration, which will help locate and troubleshoot problems.

Data persistence to network PV

Doris Operator uses Kubernetes' default StorageClass to support FE and BE storage. In the CR of DorisCluster, the specified network PV can be configured by modifying the StorageClass to specify

persistentVolumeClaimSpec.storageClassName.

```
persistentVolumes:
     - mountPath: /opt/apache-doris/fe/doris-meta
       name: storage0
       persistentVolumeClaimSpec:
         # When use specific storageclass, the storageClassName should reConfig, example as annotation.
         storageClassName: ${your_storageclass}
         accessModes:
         - ReadWriteOnce
         resources:
           # notice: if the storage size is less than 5G, fe will not start normal.
           requests:
             storage: 100Gi
```

FE configuration persistent storage

When deploying a cluster, it is recommended to provide persistent storage for the doris-meta and log directories in FE. Doris-meta users store metadata, usually from a few hundred MB to dozens of GB. It is recommended to reserve 100GB. The log directory is used to store FE logs. It is generally recommended to reserve 50GB.

In the following example, FE uses StorageClass to mount metadata storage and log storage:

```
feSpec:
     persistentVolumes:
     - name: fe-meta
       mountPath: /opt/apache-doris/fe/doris-meta
       persistentVolumeClaimSpec:
         storageClassName: ${storageClassName}
         accessModes:
         - ReadWriteOnce
         resources:
           requests:
             Storage: 50Gi
     - name: fe-log
       mountPath: /opt/apache-doris/fe/log
       persistentVolumeClaimSpec:
         storageClassName: ${storageClassName}
         accessModes:
         - ReadWriteOnce
         resources:
           requests:
             storage: 100Gi
```

Among them, the name of StorageClass needs to be specified in ${storageClassName}. You can use the following command to view the StorageClass supported in the current Kubernetes cluster:

```
kubectl get sc
```

The return result is as follows:

```
NAME                      PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
openebs-hostpath          openebs.io/local               Delete          WaitForFirstConsumer   false                  212d
openebs-device            openebs.io/local               Delete          WaitForFirstConsumer   false                  212d
openebs-jiva-csi-default  jiva.csi.openebs.io            Delete          Immediate              true                   212d
local-storage             kubernetes.io/no-provisioner   Delete          WaitForFirstConsumer   false                  149d
```

```
microk8s-hostpath (default)   microk8s.io/hostpath          Delete          Immediate              false                  219d
doris-storage                 openebs.io/local              Delete          WaitForFirstConsumer   false                  54d
```

> **Tip**
>
> The default metadata path and log path can be modified by configuring ConfigMap: 1. The mounthPath configuration of fe-meta needs to be consistent with the meta_dir variable configuration path in ConfigMap. By default, metadata will be written to the /opt/apache-doris/fe/doris-meta directory; 2. The mounthPath configuration of fe-log needs to be consistent with the LOG_DIR variable path in ConfigMap. By default, log data will be written to the /opt/apache-doris/fe/log directory.

BE configuration persistent storage

When deploying a cluster, it is recommended that the storage and log directories in BE be used for persistent storage. Storage users store data, which needs to be measured based on the amount of business data. The log directory is used to store FE logs. It is generally recommended to reserve 50GB.

In the following example, BE uses StorageClass to mount the data storage and log storage:

```
beSpec:
   persistentVolumes:
   - mountPath: /opt/apache-doris/be/storage
     name: be-storage
     persistentVolumeClaimSpec:
       storageClassName: {storageClassName}
       accessModes:
         - ReadWriteOnce
       resources:
         requests:
           Storage: 1Ti
   - mountPath: /opt/apache-doris/be/log
     name: belog
     persistentVolumeClaimSpec:
       storageClassName: {storageClassName}
       accessModes:
       - ReadWriteOnce
       resources:
         requests:
           storage: 100Gi
```

Cluster deployment configuration

Cluster name

The cluster name can be configured by modifying metadata.name in DorisCluster Custom Resource.

Mirror version

When deploying a Doris cluster, you can specify the cluster version. When deploying a cluster, you should ensure that the versions of each component in the cluster are consistent. Configure the version of each component by modifying `spec.{feSpec|beSpec}.image`.

Cluster topology

Before deploying a Doris cluster, you need to plan the topology of the cluster based on your business. The number of nodes of each component can be configured by modifying spec.{feSpec|beSpec}.replicas. Based on the principle of high data availability of production nodes, Doris Operator stipulates that there are at least 3 nodes in the Kubernetes cluster in the cluster. At the same time, in order to ensure the availability of the cluster, it is recommended to deploy at least 3 FE and BE nodes.

Service configuration

Kubernetes provides different Serivce methods to expose Doris's external access interface, such as `ClusterIP`, `NodePort`, `LoadBalancer`, etc.

ClusterIP

A service of type ClusterIP will create a virtual IP inside the cluster. It can only be accessed within the Kubernetes cluster through ClusterIP and is not visible to the outside world. In Doris Custom Resource, the ClusterIP type Service is used by default.

NodePort

Can be exposed via NodePort when LoadBalancer is not available. NodePort exposes services through the node's IP and static port. A NodePort service can be accessed from outside the cluster by requesting `NodeIP + NodePort`.

```
...
feSpec:
   replicas: 3
   service:
     type: NodePort
...
beSpec:
   replicas: 3
   service:
     type: NodePort
...
```

Cluster parameter configuration

Doris uses `ConfigMap` in Kubernetes to decouple configuration files and services. All nodes of the Doris component use ConfigMap as unified configuration management in Kubernetes, and all nodes of the component are started with the same configuration information. Doris' system parameters are stored in ConfigMap using key-value pairs. When deploying a doris cluster, you need to deploy ConfigMap under the same namespace in advance.

In the CR of Doris Cluster, provide ConfigMapInfo definitions to mount configuration information for each component. ConfigMapInfo contains two variables:

- ConfigMapName represents the name of the ConfigMap you want to use
- ResolveKey represents the corresponding configuration file, select fe.conf for FE configuration, and be.conf for BE configuration.

FE ConfigMap

Definition FE ConfigMap

When using ConfigMap to define FE configuration, you need to first define and deliver ConfigMap to the Kubernetes cluster.

The following example defines a ConfigMap named fe-conf:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: fe-conf
  labels:
    app.kubernetes.io/component: fe
data:
  fe.conf: |
    CUR_DATE=`date +%Y%m%d-%H%M%S`

    # the output dir of stderr and stdout
    LOG_DIR = ${DORIS_HOME}/log

    JAVA_OPTS="-Djavax.security.auth.useSubjectCredsOnly=false -Xss4m -Xmx8192m -XX:+UseMembar -XX:SurvivorRatio=8 -
XX:MaxTenuringThreshold=7 -XX:+PrintGCDateStamps -XX:+PrintGCDetails -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -
XX:+CMSClassUnloadingEnabled -XX:-CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=80 -XX:SoftRefLRUPolicyMSPerMB=0 -
Xloggc:$DORIS_HOME/log/fe.gc.log.$CUR_DATE"

    # For jdk 9+, this JAVA_OPTS will be used as default JVM options
    JAVA_OPTS_FOR_JDK_9="-Djavax.security.auth.useSubjectCredsOnly=false -Xss4m -Xmx8192m -XX:SurvivorRatio=8 -
XX:MaxTenuringThreshold=7 -XX:+CMSClassUnloadingEnabled -XX:-CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=80 -
XX:SoftRefLRUPolicyMSPerMB=0 -Xlog:gc*:$DORIS_HOME/log/fe.gc.log.$CUR_DATE:time"

    # INFO, WARN, ERROR, FATAL
    sys_log_level = INFO

    # NORMAL, BRIEF, ASYNC
    sys_log_mode = NORMAL

    # Default dirs to put jdbc drivers,default value is ${DORIS_HOME}/jdbc_drivers
    # jdbc_drivers_dir = ${DORIS_HOME}/jdbc_drivers

    http_port = 8030
    rpc_port = 9020
    query_port = 9030
    edit_log_port = 9010
    enable_fqdn_mode = true
```

Among them, the name of FE ConfigMap is defined in metadata.name, and the database configuration in fe.conf is defined in data. Be sure to add `enable_fqdn_mode = true` to your self-configured `fe.conf`

> **Tip**
>
> Use the data field in ConfigMap to store key-value pairs. In the above FE ConfigMap: - fe.conf is the key in the key-value pair, using | means that newlines and indents in subsequent strings will be preserved - Subsequent configuration is the value in the key-value pair, which is the same as the configuration in the

> fe.conf file In the data field, due to the use of the | symbol to retain the subsequent string format, two spaces need to be maintained in subsequent configurations.

After defining the FE ConfigMap, you need to issue it through the `kubectl apply` command.

Using FE ConfigMap

If you need to use FE ConfigMap, you need to specify the defined ConfigMap through spec.feSpec.configMapInfo in the RC of Doris Cluster.

```
Kind: DorisCluster
metadata:
   name: doriscluster-sample-configmap
spec:
   feSpec:
     configMapInfo:
       configMapName: {feConfigMapName}
       resolveKey: fe.conf
...
```

Replace ${feConfigMapName} with fe-conf in the above example to use the FE ConfigMap defined in the above example. For FE ConfigMap, you need to keep the resolveKey field fixed to `fe.conf`.

BE ConfigMap

Definition BE ConfigMap

When using ConfigMap to define BE configuration, you need to first define and deliver ConfigMap to the Kubernetes cluster.

The following example defines a ConfigMap named be-conf:

```
apiVersion: v1
kind: ConfigMap
metadata:
   name: be-conf
   labels:
     app.kubernetes.io/component: be
data:
   be.conf: |
     CUR_DATE=`date +%Y%m%d-%H%M%S`

     PPROF_TMPDIR="$DORIS_HOME/log/"

     JAVA_OPTS="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xloggc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command=DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"

     # For jdk 9+, this JAVA_OPTS will be used as default JVM options
     JAVA_OPTS_FOR_JDK_9="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xlog:gc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command =DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"

     # since 1.2, the JAVA_HOME need to be set to run BE process.
     # JAVA_HOME=/path/to/jdk/

     # https://github.com/apache/doris/blob/master/docs/zh-CN/community/developer-guide/debug-tool.md#jemalloc-heap-profile
```

```
    # https://jemalloc.net/jemalloc.3.html

JEMALLOC_CONF="percpu_arena:percpu,background_thread:true,metadata_thp:auto,muzzy_decay_ms:15000,dirty_decay_ms:15000,oversize_thres
hold:0,lg_tcache_max:20,prof:false,lg_prof_interval:32,lg_prof_sample:19,prof_gd ump:false,prof_accum:false
,prof_leak:false,prof_final:false"
    JEMALLOC_PROF_PRFIX=""

    # INFO, WARNING, ERROR, FATAL
    sys_log_level = INFO

    # ports for admin, web, heartbeat service
    be_port = 9060
    webserver_port = 8040
    heartbeat_service_port = 9050
    brpc_port = 8060
```

Among them, the name of BE ConfigMap is defined in metadata.name, and the database configuration in be.conf is defined in data.

> **Tip**
>
> Use the data field in ConfigMap to store key-value pairs. In the above BE ConfigMap: - be.conf is the key in the key-value pair, using | means that newlines and indents in subsequent strings will be retained - Subsequent configuration is the value in the key-value pair, which is the same as the configuration in the be.conf file In the data field, due to the use of the | symbol to retain the subsequent string format, two spaces need to be maintained in subsequent configurations.

After defining BE ConfigMap, you need to issue it through the `kubectl apply` command.

Using BE ConfigMap

If you need to use BE ConfigMap, you need to specify the defined ConfigMap through spec.beSpec.configMapInfo in the RC of Doris Cluster.

```
Kind: DorisCluster
metadata:
   name: doriscluster-sample-configmap
spec:
   beSpec:
     configMapInfo:
       configMapName: {beConfigMapName}
       resolveKey: be.conf
...
```

Replace ${beConfigMapName} with be-conf in the above example to use the BE ConfigMap defined in the above example. For BE ConfigMap, you need to keep the resolveKey field fixed to `be.conf`.

Add external configuration files to the conf directory

When using the Catalog function to access external data sources, you need to add the relevant configuration files to the conf directory of the Doris node. For example, when accessing the hive catalog, you need to add core-site.xml, hdfs-site.xml and hive-site.xml The files are placed in the conf directories of FE and BE.

In the Kubernetes environment, the relevant configuration files of the catalog need to be loaded into Doris in the form of ConfigMap. The following example shows loading the core-site.xml file into BE:

```
apiVersion: v1
kind: ConfigMap
metadata:
   name: be-configmap
   labels:
     app.kubernetes.io/component: be
data:
   be.conf: |
     be_port = 9060
     webserver_port = 8040
     heartbeat_service_port = 9050
     brpc_port = 8060
   core-site.xml: |
     <?xml version="1.0" encoding="UTF-8"?>
     <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
     <configuration>
       <property>
       <name>hadoop.security.authentication</name>
         <value>kerberos</value>
       </property>
     </configuration>
     ...
```

Among them, the configured key-value pairs are stored in the data field. In the above example, the key-value pairs whose keys are be.conf and core-site.xml are stored.

In the data field, the following key-value structure mapping needs to be satisfied:

```
data:
  filename_1: |
    config_string
  filename_2: |
    config_string
  filename_3: |
    config_string
```

Configure multi-disk storage for BE

Doris supports mounting multiple PVs for BE. By configuring the BE parameter `storage_root_path`, you can specify BE to use multi-disk storage. In the Kubernetes environment, you can map pv in DorisCluster CR and configure the `storage_root_path` parameter for BE through ConfigMap.

Configure pv mapping for BE multi-disk storage

In the DorisCluster CR file, compared to the single-disk configuration, you need to add the descriptions of `configMapInfo` and `persistentVolumeClaimSpec`:

- The specified ConfigMap under the same namespace can be identified through `configMapInfo` configuration, and the resolveKey is fixed to be.conf
- Multiple pv mappings can be configured for the BE storage directory through `persistentVolumeClaimSpec`

In the following example, the pv mapping of two disks is configured for BE:

```
...
  beSpec:
    replicas: 3
    image: selectdb/doris.be-ubuntu:2.0.2
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    configMapInfo:
      configMapName: be-configmap
      resolveKey: be.conf
    persistentVolumes:
    - mountPath: /opt/apache-doris/be/storage1
      name: storage2
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
    - mountPath: /opt/apache-doris/be/storage2
      name: storage3
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
    - mountPath: /opt/apache-doris/be/log
      name: storage4
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
```

In the above example, the Doris cluster specifies multi-disk storage

- beSpec.persistentVolumes specifies multiple pvs in an array, mapping two data storage pvs in `/opt/apache-doris/be/storage{1,2}`
- beSpec.configMapInfo specifies that the ConfigMap named `be-configmap` needs to be mounted

Configure BE ConfigMap to specify the storage_root_path parameter

According to the BE ConfigMap name specified in DorisCluster CR, you need to create the corresponding ConfigMap and specify the storage_root_path parameter.

In the following example, the `storage_root_path` parameter is specified in the ConfigMap named `be-configmap` to use two disks:

```
apiVersion: v1
kind: ConfigMap
metadata:
   name: be-configmap
   labels:
     app.kubernetes.io/component: be
data:
   be.conf: |
     CUR_DATE=`date +%Y%m%d-%H%M%S`

     PPROF_TMPDIR="$DORIS_HOME/log/"

     JAVA_OPTS="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xloggc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command=DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"

     # For jdk 9+, this JAVA_OPTS will be used as default JVM options
     JAVA_OPTS_FOR_JDK_9="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xlog:gc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command =DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"

     # since 1.2, the JAVA_HOME need to be set to run BE process.
     # JAVA_HOME=/path/to/jdk/

     # https://github.com/apache/doris/blob/master/docs/zh-CN/community/developer-guide/debug-tool.md#jemalloc-heap-profile
     # https://jemalloc.net/jemalloc.3.html

JEMALLOC_CONF="percpu_arena:percpu,background_thread:true,metadata_thp:auto,muzzy_decay_ms:15000,dirty_decay_ms:15000,oversize_thres
hold:0,lg_tcache_max:20,prof:false,lg_prof_interval:32,lg_prof_sample:19,prof_gd ump:false,prof_accum:false
,prof_leak:false,prof_final:false"
     JEMALLOC_PROF_PRFIX=""

     # INFO, WARNING, ERROR, FATAL
     sys_log_level = INFO

     # ports for admin, web, heartbeat service
     be_port = 9060
     webserver_port = 8040
     heartbeat_service_port = 9050
     brpc_port = 8060

     storage_root_path = /opt/apache-doris/be/storage,medium:ssd;/opt/apache-doris/be/storage1,medium:ssd
```

> **Warning**
>
> When creating a BE ConfigMap, you need to pay attention to the following: 1. metadata.name needs to be the same as beSpec.configMapInfo.configMapName in DorisCluster CR, indicating that the cluster uses the specified ConfigMap; 2. The storage_root_path parameter in ConfigMap must correspond one-to-one with the persistentVolume data disk in DorisCluster CR.

Deploying Doris Cluster

After planning the cluster topology, you can deploy the Doris cluster in Kubernetes.

Deploy cluster

Deploy using Custom Resource files

Online Deployment

Deploying a cluster online requires the following steps:

1. Create namespace:

```
kubectl create namespace ${namespace}
```

2. Deploy Doris cluster

```
kubectl apply -f ./${cluster_sample}.yaml -n ${namespace}
```

Offline deployment

To deploy Doris cluster offline, you need to upload the docker image used by Doris cluster to all nodes on a machine with external network. Then use docker load to install the image on the server. Offline deployment requires the following steps:

1. Download the required image

Deploying Doris cluster requires the following images:

```
selectdb/doris.fe-ubuntu:2.0.2
selectdb/doris.be-ubuntu:2.0.2
```

Download the image locally and package it into a tar file

```
## download docker image
docker pull selectdb/doris.fe-ubuntu:2.0.2
docker pull selectdb/doris.be-ubuntu:2.0.2

## save docker image as a tar package
docker save -o doris.fe-ubuntu-v2.0.2.tar selectdb/doris.fe-ubuntu:2.0.2
docker save -o doris.be-ubuntu-v2.0.2.tar docker pull selectdb/doris.be-ubuntu:2.0.2
```

Upload the image tar package to the server and execute the docker load command:

```
## load docker image
docker load -i doris.fe-ubuntu-v2.0.2.tar
docker load -i doris.be-ubuntu-v2.0.2.tar
```

2. Create namespace:

```
kubectl create namespace ${namespace}
```

3. Deploy Doris cluster

```
kubectl apply -f ./${cluster_sample}.yaml -n ${namespace}
```

Deploy using Helm

Online Deployment

Before installation, you need to add a deployment warehouse. If it has been added, you can directly install Doris Cluster. Otherwise, please refer to the operation of **Add Deployment Warehouse** when adding Deploy Doris Operator

1. Install Doris Cluster

Install doriscluster, using the default configuration this deployment only deploys 3 FE and 3 BE components, using the default `storageClass` to implement PV dynamic provisioning.

```
helm install doriscluster doris-repo/doris
```

If you need to customize resources and cluster shapes, please customize the resource configuration according to the annotations of each resource configuration in values.yaml and execute The following command:

```
helm install -f values.yaml doriscluster doris-repo/doris
```

2. Verify doris cluster installation results

You can check the pod deployment status through the `kubectl get pods` command. When the Pod of `doriscluster` is in `Running` state and all containers in the Pod are ready, the deployment is successful.

```
kubectl get pod --namespace doris
```

The return result is as follows:

```
NAME                    READY   STATUS    RESTARTS   AGE
doriscluster-helm-fe-0   1/1     Running   0          1m39s
doriscluster-helm-fe-1   1/1     Running   0          1m39s
doriscluster-helm-fe-2   1/1     Running   0          1m39s
doriscluster-helm-be-0   1/1     Running   0          16s
doriscluster-helm-be-1   1/1     Running   0          16s
doriscluster-helm-be-2   1/1     Running   0          16s
```

Offline deployment

1. Download the Doris Cluster Chart resource

Download `doris-{chart_version}.tgz` to install Doris Cluster chart. If you need to download the 1.6.1 version of the Doris cluster, you can use the following command:

```
wget https://charts.selectdb.com/doris-1.6.1.tgz
```

2. Install Doris cluster

Doris cluster can be installed through the `helm install` command.

```
helm install doriscluster doris-1.4.0.tgz
```

If you need to customize the assembly values.yaml, you can refer to the following command:

```
helm install -f values.yaml doriscluster doris-1.4.0.tgz
```

3. Verify doris cluster installation results

You can check the pod deployment status through the `kubectl get pods` command. When the Pod of `doriscluster` is in `Running` state and all containers in the Pod are ready, the deployment is successful.

```
kubectl get pod --namespace doris
```

The return result is as follows:

```
NAME                     READY   STATUS    RESTARTS   AGE
doriscluster-helm-fe-0   1/1     Running   0          1m39s
doriscluster-helm-fe-1   1/1     Running   0          1m39s
doriscluster-helm-fe-2   1/1     Running   0          1m39s
doriscluster-helm-be-0   1/1     Running   0          16s
doriscluster-helm-be-1   1/1     Running   0          16s
doriscluster-helm-be-2   1/1     Running   0          16s
```

View cluster status

Check cluster status

After the cluster deployment resources are delivered, you can check the cluster status by running the following command.

```
kubectl get pods -n ${namespace}
```

The return result is as follows:

```
NAME                       READY   STATUS    RESTARTS   AGE
doriscluster-sample-fe-0   1/1     Running   0          20m
doriscluster-sample-be-0   1/1     Running   0          19m
```

When the `STATUS` of all pods is in the `Running` state, and all containers in the pods of all components are `READY`, it means that the entire cluster is deployed normally.

Check deployment resource status

Doris Operator will collect the status of cluster services and display them in the distributed resources. Doris Operator defines the abbreviation `dcr` for the `DorisCluster` type resource name, which can be replaced by the abbreviation when using the resource type to view the cluster status.

```
kubectl get dcr
```

The return result is as follows:

```
NAME                FESTATUS    BESTATUS    CNSTATUS   BROKERSTATUS
doriscluster-sample   available   available
```

When the `STATUS` of the configured related services is `available`, the cluster deployment is successful.

Accessing Doris Cluster

Access using ClusterIP mode

Doris provides ClusterIP access mode by default on Kubernetes. The ClusterIP access mode provides an internal IP address within a Kubernetes cluster through which services are exposed. With ClusterIP mode, access is only possible within the cluster.

1. Configure and use ClusterIP as the Service type

Doris provides ClusterIP access mode by default on Kubernetes. No modification is required to use ClusterIP access mode.

2. Get Service

After deploying the cluster, you can view the services exposed by Doris Operator through the following command:

```
kubectl -n doris get svc
```

The return result is as follows:

```
NAME                              TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)                               AGE
doriscluster-sample-be-internal   ClusterIP   None          <none>        9050/TCP                              9m
doriscluster-sample-be-service    ClusterIP   10.1.68.128   <none>        9060/TCP,8040/TCP,9050/TCP,8060/TCP   9m
doriscluster-sample-fe-internal   ClusterIP   None          <none>        9030/TCP                              14m
doriscluster-sample-fe-service    ClusterIP   10.1.118.16   <none>        8030/TCP,9020/TCP,9030/TCP,9010/TCP   14m
```

In the above results, FE and BE have two types of Services, with the suffixes internal and service respectively:

- Service services with the suffix internal can only be used for Doris internal communication, such as heartbeat, data exchange and other operations, and are not used externally.
- Service services with the service suffix can be provided to users

3. Access Doris inside the container

Use the following command to create a pod containing the mysql client in the current Kubernetes cluster:

```
kubectl run mysql-client --image=mysql:5.7 -it --rm --restart=Never --namespace=doris -- /bin/bash
```

In the container in the cluster, you can use the externally exposed service name with the suffix `service` to access the Doris cluster:

```
## Use service type pod name to access the Doris cluster
mysql -uroot -P9030 -hdoriscluster-sample-fe-service
```

Access using NodePort mode

If users need to access Doris outside the Kubernetes cluster, they can choose to use NodePort mode.

1. Plan NodePort mode port mapping

To use and maintain Doris cluster, users need to access the following ports:

port name	default port	port description
Query Port	9030	Used to access Doris cluster through MySQL protocol
HTTP Port	8030	http server port on FE, used to view FE information
Web Server Port	8040	http server port on BE, used to view BE information

There are two ways of port allocation using NodePort:

- Dynamic allocation: If port mapping is not explicitly set, Kubernetes will automatically allocate an unused port when creating a pod (the default range is 30000-32767);
- Static allocation: If port mapping is explicitly specified, Kubernetes will fixedly allocate the port when the port is not in use without conflict.

Kubernetes uses dynamic port allocation by default. If you need to plan the port in advance, you need to specify it explicitly in Custom Resource. In the following example, the Doris port is mapped:

```
...
spec:
  feSpec:
    replicas: 3
    service:
      type: NodePort
      servicePorts:
        - nodePort: 31001
          targetPort: 8030
        - nodePort: 31002
          targetPort: 8040
        - nodePort: 31003
          targetPort: 9030
...
  beSpec:
    replicas: 3
    service:
      type: NodePort
      servicePorts:
        - nodePort: 31005
          targetPort: 9060
        - nodePort: 31006
          targetPort: 8040
        - nodePort: 31007
          targetPort: 9050
        - nodePort: 31008
          targetPort: 8060
...
```

2. Configure using NodePort as Service type

Custom DorisCluster resource to use NodePort mode with BE, the specific changes are as follows:

```
...
spec:
   feSpec:
     replicas: 3
     service:
       type: NodePort
...
    beSpec:
     replicas: 3
     service:
       type: NodePort
...
```

3. Get Service

After deploying the cluster, you can view the services exposed by Doris Operator through the following command:

```
kubectl get service
```

The return result is as follows:

```
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 10.152.183.1 <none> 443/TCP 169d
doriscluster-sample-fe-internal ClusterIP None <none> 9030/TCP 2d
doriscluster-sample-fe-service NodePort 10.152.183.58 <none> 8030:31041/TCP,9020:30783/TCP,9030:31545/TCP,9010:31610/TCP 2d
doriscluster-sample-be-internal ClusterIP None <none> 9050/TCP 2d
doriscluster-sample-be-service NodePort 10.152.183.244 <none> 9060:30940/TCP,8040:32713/TCP,9050:30621/TCP,8060:30926/TCP 2d
```

Doris's Query Port defaults to 9030, which is mapped to local port 31545 locally. When accessing the Doris cluster, you need to obtain the corresponding IP address, which can be viewed with the following command:

```
kubectl get nodes -owide
```

The return result is as follows:

```
NAME   STATUS   ROLES          AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION          CONTAINER-
RUNTIME
r60    Ready    control-plane   14d   v1.28.2   192.168.88.60   <none>        CentOS Stream 8   4.18.0-294.el8.x86_64
containerd://1.6.22
r61    Ready    <none>          14d   v1.28.2   192.168.88.61   <none>        CentOS Stream 8   4.18.0-294.el8.x86_64
containerd://1.6.22
r62    Ready    <none>          14d   v1.28.2   192.168.88.62   <none>        CentOS Stream 8   4.18.0-294.el8.x86_64
containerd://1.6.22
r63    Ready    <none>          14d   v1.28.2   192.168.88.63   <none>        CentOS Stream 8   4.18.0-294.el8.x86_64
containerd://1.6.22
```

In NodePort mode, you can access services in the Kubernetes cluster based on the host IP and port mapping of any node. In this example, you can use any node IP, 192.168.88.61, 192.168.88.62, 192.168.88.63, to access the Doris service. For example, in the following example, the node node 192.168.88.62 and the mapped query port port 31545 are used to access the cluster:

```
mysql -h 192.168.88.62 -P 31545 -uroot
```

Stream Load ErrorURL Redirect

Stream Load is a synchronous import mode provided by Doris. It is an efficient way to import local files into Doris. In the case of physical or virtual machine deployment, directly use http to initiate an import data request to FE, and FE will redirect the request to the BE service through the 301 mechanism to execute the write request. On Kubernetes, FE and BE use Service as the method of service discovery. In situations where a proxy is used to mask the internal real address to provide service discovery, the address of the BE (the real address used for internal communication within the service) returned using FE 301 cannot be accessed. On Kubernetes, you need to use BE's Service address to import data.

As in the following example, Stream Load ErrorUrl returns the result `http://doriscluster-sample-be-2.doriscluster-sample-be-internal.doris.svc.cluster.local:8040/api/_load_error_log?file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e`

View ErrorURL inside the container

If you perform Stream Load inside Kubernetes, you can directly use the error address returned by Stream Load to obtain a detailed error report.

In the return result of the above example, you can directly obtain the return result through the curl command in the Pod in the same Kubernetes cluster:

```
curl http://doriscluster-sample-be-2.doriscluster-sample-be-internal.doris.svc.cluster.local:8040/api/_load_error_log?file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e
```

View ErrorURL outside the container

An error occurs when importing data using Stream Load from outside Kubernetes. The returned error address cannot be directly accessed outside Kubernetes to obtain a detailed error report. In the Kubernetes environment, you need to use a customized Service proxy to detect the pod where an error occurred. Configure the customized Service in an externally accessible mode and obtain a detailed error report by accessing the proxy Service.

The customized Service template is as follows:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.doris.service/role: debug
    app.kubernetes.io/component: be
  name: doriscluster-detail-error
spec:
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: webserver-port
      port: 8040
      protocol: TCP
      targetPort: 8040
```

```
  selector:
    app.kubernetes.io/component: be
    statefulset.kubernetes.io/pod-name: ${podName}
  sessionAffinity: None
  type: ${ServiceType}
```

in:

- ${podName} represents the third-level domain name of the pod where the error currently occurs. For example, in the above example, the pod name needs to be filled in doriscluster-sample-be-2
- ${ServiceType} is the deployed Service type, you can choose NodePort or LoadBalancer

> **Tip**
>
> Since the pod name returned by each stream load may be different, please delete the customized Service after obtaining the detailed error information of the stream load.

NodePort Mode

1. Deploy NodePort Service

Follow the service in the above example, replace ${podName} in CR with doriscluster-sample-be-2, and replace ${ServiceType} with NodePort. Use the kubectl apply command to create the service service in the same namespace of the doris cluster.

```
kubectl -n {namespace} apply -f strem_load_get_error.yaml
```

2. Build access commands

Use the following command to view the NodePort assigned by the above deployment Service:

```
kubectl get service -n doris doriscluster-detail-error
```

The return result is as follows:

```
NAME                           TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)           AGE
doriscluster-detail-error      NodePort    10.152.183.35    <none>        8040:31201/TCP    32s
```

The BE port accessed by `Stream Load` is 8040, and the host port (NodePort) corresponding to 8040 in the above Service is 31201.

Get the host address controlled by K8s:

```
kubectl get node -owide
```

The return result is as follows:

```
NAME            STATUS   ROLES    AGE    VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE                      KERNEL-VERSION
CONTAINER-RUNTIME
```

```
vm-10-8-centos   Ready    <none>   226d   v1.28.7   10.16.10.8    <none>        TencentOS Server 3.1 (Final)   5.4.119-19-0009.3
containerd://1.6.28
vm-10-7-centos   Ready    <none>   19d    v1.28.7   10.16.10.7    <none>        TencentOS Server 3.1 (Final)   5.4.119-19.0009.25
containerd://1.6.28
```

Use any of the above hosts `INTERNAL-IP` and get the host port to build an access address using NodePort mode to get error details. In `NodePort` mode, the address for obtaining error details is spliced into `Host IP:NodePort`, then the accessible address of the case is `10.16.10.8:31201`. Replace the access address in the returned error address information to obtain the accessible error information. Available addresses for details:

```
http://10.16.10.8:31201/api/_load_error_log?file=__shard_1/error_log_insert_stmt_af474190276a2e9c-
49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e
```

Use the above command to obtain detailed error information of Stream Load.

LoadBalancer Mode

1. Deploy and obtain error details Service

Assume that the error address of the Stream Load range is as follows:

```
http://doriscluster-sample-be-2.doriscluster-sample-be-internal.doris.svc.cluster.local:8040/api/_load_error_log?
file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e
```

The domain name address of the above address is `doriscluster-sample-be-2.doriscluster-sample-be-internal.doris.svc.cluster.local`. Among the domain names used by pods deployed by `Doris Operator` on `Kubernetes`, the third level The domain name is the name of the pod. Replace {podName} in the above template with the real `pod` name, replace {serviceType} with `LoadBalancer`, and save the changes to the newly created `stream_load_get_error.yaml` file. Use the following command to deploy service:

```
kubectl -n {namespace} apply -f strem_load_get_error.yaml
```

2. Build access commands

Use the following command to view the LoadBalaner address EXTERNAL-IP assigned by the above deployment Service. The following is a test instance in aws eks:

```
kubectl get service -n doris doriscluster-detail-error
```

The return result is as follows:

```
NAME                         TYPE          CLUSTER-IP       EXTERNAL-IP
PORT(S)          AGE
doriscluster-detail-error    LoadBalancer  172.20.183.136   ac4828493dgrftb884g67wg4tb68gyut-1137856348.us-east-1.elb.amazonaws.com
8040:32003/TCP    14s
```

The above Service obtains the LoadBalancer address assigned by the K8s cluster as `ac4828493dgrftb884g67wg4tb68gyut``-1137856348.us-east-1.elb.amazonaws.com`. In LoadBalancer mode, the port is still the listening port for deployment deployment. In `LoadBalancer` mode , the address to obtain error details is spliced into “EXTERNAL-IP:listener-port”. In the above example, the address to obtain error details is

ac4828493dgrftb884g67wg4tb68gyut-1137856348.us-east-1.elb.amazonaws.com:8040. The address to obtain detailed error information is as follows:

```
http://ac4828493dgrftb884g67wg4tb68gyut-1137856348.us-east-1.elb.amazonaws.com:8040/api/_load_error_log?
file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e
```

Upgrading Apache Doris cluster deployed by Doris Operator

This document describes how to use updates to upgrade an Apache Doris cluster based on a Doris Operator deployment.

Similar to conventionally deployed cluster upgrades, Doris clusters deployed by Doris Operator still require rolling upgrades from BE to FE nodes. Doris Operator is based on Kubernetes' Performing a Rolling Update provides rolling upgrade capabilities.

Things to note before upgrading

- It is recommended that the upgrade operation be performed during off-peak periods.
- During the rolling upgrade process, the connection to the closed node will fail, causing the request to fail. For this type of business, it is recommended to add retry capabilities to the client.
- Before upgrading, you can read the General Upgrade Manual to help you understand some principles and precautions during the upgrade. .
- The compatibility of data and metadata cannot be verified before upgrading. Therefore, cluster upgrade must avoid single copy of data and single FE FOLLOWER node in the cluster.
- Nodes will be restarted during the upgrade process, so unnecessary cluster balancing and replica repair logic may be triggered. Please shut it down first with the following command.

```
admin set frontend config("disable_balance" = "true");
admin set frontend config("disable_colocate_balance" = "true");
admin set frontend config("disable_tablet_scheduler" = "true");
```

- When upgrading Doris, please follow the principle of not upgrading across two or more key node versions. If you want to upgrade across multiple key node versions, upgrade to the latest key node version first, and then upgrade in sequence. If it is a non-key node version, You can ignore skipping. For details, please refer to Upgrade Version Instructions

Upgrade operation

The order of node types in the upgrade process is as follows. If a certain type of node does not exist, it will be skipped:

```
cn/be -> fe -> broker
```

It is recommended to modify the `image` of the corresponding cluster components in sequence and then apply the configuration. After the current type of component is fully upgraded and the status returns to normal, the rolling upgrade of the next type of node can be performed.

Upgrade BE

If you retain the cluster's crd (Doris Operator defines the abbreviation of `DorisCluster` type resource name) file, you can upgrade by modifying the configuration file and running the `kubectl apply` command.

1. Modify `spec.beSpec.image`

   Change `selectdb/doris.be-ubuntu:2.0.4` to `selectdb/doris.be-ubuntu:2.1.0`

   ```
   $ vim doriscluster-sample.yaml
   ```

2. Save the changes and apply the changes to be upgraded:

```
$ kubectl apply -f doriscluster-sample.yaml -n doris
```

It can also be modified directly through `kubectl edit dcr`.

1. Check the dcr list under namespace 'doris' to obtain the `cluster_name` that needs to be updated.

```
$ kubectl get dcr -n doris
NAME                 FESTATUS    BESTATUS    CNSTATUS
doriscluster-sample   available   available
```

2. Modify, save and take effect

```
$ kubectl edit dcr doriscluster-sample -n doris
```

```
After entering the text editor, you will find `spec.beSpec.image` and change `selectdb/doris.be-ubuntu:2.0.4` to `selectdb/doris.be-ubuntu:2.1.0`
```

3. View the upgrade process and results:

```
$ kubectl get pod -n doris
```

When all Pods are rebuilt and enter the Running state, the upgrade is complete.

Upgrade FE

If you retain the cluster's crd (Doris Operator defines the abbreviation of the `DorisCluster` type resource name) file, you can upgrade by modifying the configuration file and running the `kubectl apply` command.

1. Modify `spec.feSpec.image`

   Change `selectdb/doris.fe-ubuntu:2.0.4` to `selectdb/doris.fe-ubuntu:2.1.0`

   ```
   $ vim doriscluster-sample.yaml
   ```

2. Save the changes and apply the changes to be upgraded:

```
$ kubectl apply -f doriscluster-sample.yaml -n doris
```

It can also be modified directly through `kubectl edit dcr`.

1. Modify, save and take effect

```
$ kubectl edit dcr doriscluster-sample -n doris
```

```
After entering the text editor, you will find `spec.feSpec.image` and change `selectdb/doris.fe-ubuntu:2.0.4` to `selectdb/doris.fe-ubuntu:2.1.0`
```

2. View the upgrade process and results:

```
$ kubectl get pod -n doris
```

When all Pods are rebuilt and enter the Running state, the upgrade is complete.

After the upgrade is completed

Verify cluster node status

Access Doris through `mysql-client` through the method provided in the Access Doris Cluster document. Use SQL such as `show frontends` and `show backends` to view the version and status of each component.

```
mysql> show frontends\G;
*************************** 1. row ***************************
             Name: fe_13c132aa_3281_4f4f_97e8_655d01287425
             Host: doriscluster-sample-fe-0.doriscluster-sample-fe-internal.doris.svc.cluster.local
       EditLogPort: 9010
          HttpPort: 8030
         QueryPort: 9030
           RpcPort: 9020
ArrowFlightSqlPort: -1
             Role: FOLLOWER
         IsMaster: false
        ClusterId: 1779160761
             Join: true
            Alive: true
 ReplayedJournalId: 2422
    LastStartTime: 2024-02-19 06:38:47
    LastHeartbeat: 2024-02-19 09:31:33
         IsHelper: true
           ErrMsg:
          Version: doris-2.1.0
  CurrentConnected: Yes
*************************** 2. row ***************************
             Name: fe_f1a9d008_d110_4780_8e60_13d392faa54e
             Host: doriscluster-sample-fe-2.doriscluster-sample-fe-internal.doris.svc.cluster.local
       EditLogPort: 9010
          HttpPort: 8030
         QueryPort: 9030
           RpcPort: 9020
ArrowFlightSqlPort: -1
             Role: FOLLOWER
         IsMaster: true
```

```
         ClusterId: 1779160761
              Join: true
             Alive: true
 ReplayedJournalId: 2423
     LastStartTime: 2024-02-19 06:37:35
     LastHeartbeat: 2024-02-19 09:31:33
          IsHelper: true
            ErrMsg:
           Version: doris-2.1.0
  CurrentConnected: No
*************************** 3. row ***************************
              Name: fe_e42bf9da_006f_4302_b861_770d2c955a47
              Host: doriscluster-sample-fe-1.doriscluster-sample-fe-internal.doris.svc.cluster.local
       EditLogPort: 9010
          HttpPort: 8030
         QueryPort: 9030
           RpcPort: 9020
ArrowFlightSqlPort: -1
              Role: FOLLOWER
          IsMaster: false
         ClusterId: 1779160761
              Join: true
             Alive: true
 ReplayedJournalId: 2422
     LastStartTime: 2024-02-19 06:38:17
     LastHeartbeat: 2024-02-19 09:31:33
          IsHelper: true
            ErrMsg:
           Version: doris-2.1.0
  CurrentConnected: No
3 rows in set (0.02 sec)
```

If the `Alive` status of the FE node is true and the `Version` value is the new version, the FE node is upgraded successfully.

```
mysql> show backends\G;
*************************** 1. row ***************************
              BackendId: 10002
                   Host: doriscluster-sample-be-0.doriscluster-sample-be-internal.doris.svc.cluster.local
          HeartbeatPort: 9050
                 BePort: 9060
               HttpPort: 8040
               BrpcPort: 8060
     ArrowFlightSqlPort: -1
          LastStartTime: 2024-02-19 06:37:56
          LastHeartbeat: 2024-02-19 09:32:43
                  Alive: true
   SystemDecommissioned: false
             TabletNum: 14
       DataUsedCapacity: 0.000
     TrashUsedCapcacity: 0.000
          AvailCapacity: 12.719 GB
          TotalCapacity: 295.167 GB
                UsedPct: 95.69 %
         MaxDiskUsedPct: 95.69 %
     RemoteUsedCapacity: 0.000
                    Tag: {"location" : "default"}
                 ErrMsg:
                Version: doris-2.1.0
                 Status: {"lastSuccessReportTabletsTime":"2024-02-19
09:31:48","lastStreamLoadTime":-1,"isQueryDisabled":false,"isLoadDisabled":false}
HeartbeatFailureCounter: 0
               NodeRole: mix
```

```
*************************** 2. row ***************************
              BackendId: 10003
                   Host: doriscluster-sample-be-1.doriscluster-sample-be-internal.doris.svc.cluster.local
          HeartbeatPort: 9050
                 BePort: 9060
               HttpPort: 8040
               BrpcPort: 8060
     ArrowFlightSqlPort: -1
          LastStartTime: 2024-02-19 06:37:35
          LastHeartbeat: 2024-02-19 09:32:43
                  Alive: true
   SystemDecommissioned: false
             TabletNum: 8
       DataUsedCapacity: 0.000
     TrashUsedCapcacity: 0.000
          AvailCapacity: 12.719 GB
          TotalCapacity: 295.167 GB
                UsedPct: 95.69 %
         MaxDiskUsedPct: 95.69 %
     RemoteUsedCapacity: 0.000
                    Tag: {"location" : "default"}
                 ErrMsg:
                Version: doris-2.1.0
                 Status: {"lastSuccessReportTabletsTime":"2024-02-19
09:31:43","lastStreamLoadTime":-1,"isQueryDisabled":false,"isLoadDisabled":false}
HeartbeatFailureCounter: 0
               NodeRole: mix
*************************** 3. row ***************************
              BackendId: 11024
                   Host: doriscluster-sample-be-2.doriscluster-sample-be-internal.doris.svc.cluster.local
          HeartbeatPort: 9050
                 BePort: 9060
               HttpPort: 8040
               BrpcPort: 8060
     ArrowFlightSqlPort: -1
          LastStartTime: 2024-02-19 08:50:36
          LastHeartbeat: 2024-02-19 09:32:43
                  Alive: true
   SystemDecommissioned: false
             TabletNum: 0
       DataUsedCapacity: 0.000
     TrashUsedCapcacity: 0.000
          AvailCapacity: 12.719 GB
          TotalCapacity: 295.167 GB
                UsedPct: 95.69 %
         MaxDiskUsedPct: 95.69 %
     RemoteUsedCapacity: 0.000
                    Tag: {"location" : "default"}
                 ErrMsg:
                Version: doris-2.1.0
                 Status: {"lastSuccessReportTabletsTime":"2024-02-19
09:32:04","lastStreamLoadTime":-1,"isQueryDisabled":false,"isLoadDisabled":false}
HeartbeatFailureCounter: 0
               NodeRole: mix
3 rows in set (0.01 sec)
```

If the `Alive` status of the BE node is true and the `Version` value is the new version, the BE node is upgraded successfully.

Restore cluster replica synchronization and balancing

After confirming that the status of each node is correct, execute the following SQL to restore cluster balancing and replica repair:

```
admin set frontend config("disable_balance" = "false");
admin set frontend config("disable_colocate_balance" = "false");
admin set frontend config("disable_tablet_scheduler" = "false");
```

Used by Root users

Doris-Operator uses the root account without password mode when deploying and managing related service nodes. The username and password can only be reset after deployment.

Modify root account and password

1. Refer to the Privilege Management document, modify or create the corresponding password or account name, and give the account the permission to manage nodes in Doris.
2. An example of adding spec.adminUser.* to the configuration in the DorisCluster CRD file is as follows:

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  annotations:
    selectdb/doriscluster: doriscluster-sample
  labels:
    app.kubernetes.io/instance: doris-sample
  name: doris-sample
  namespace: doris
spec:
  adminUser:
    name: root
    password: root_pwd
```

3. Update the new account and password to the deployed DorisCluster, and issue them through Doris-Operator so that each node can sense and take effect. Reference command:

```
kubectl apply --namespace ${your_namespace} -f ${your_crd_yaml_file}
```

Precautions

- The cluster management account is root and has no password by default.
- The username and password can only be reset after successful deployment. During initial deployment, adding `adminUser` may cause startup exceptions.
- Modifying the user name and password is not a necessary operation. Only when the current cluster management user (default root) or password is modified in Doris, it needs to be issued through Doris-Operator.
- If you modify the user name `spec.adminUser.name`, you need to assign the new user the permission to manage Doris nodes.
- This operation restarts all nodes in sequence.

Service expansion and contraction

The expansion and contraction of Doris on K8S can be achieved by modifying the replicas field of the corresponding component of the DorisCluster resource. Modifications can be made by directly editing the corresponding resources, or by using commands.

Get DorisCluster resources

Use the command `kubectl --namespace {namespace} get doriscluster` to get the name of the deployed DorisCluster (referred to as dcr) resource. In this article, we use doris as the namespace.

```
$ kubectl --namespace doris get doriscluster
NAME                  FESTATUS    BESTATUS    CNSTATUS   BROKERSTATUS
doriscluster-sample   available   available
```

Expansion and contraction

All K8S operation and maintenance operations are performed by modifying the resources to the final state, and the Operator service automatically completes the operation and maintenance. For expansion and contraction operations, you can directly enter the edit mode to modify the replicas value of the corresponding spec through `kubectl --namespace {namespace} edit doriscluster {dcr_name}`. After saving and exiting, Doris-Operator completes the operation and maintenance. You can also use the following commands to implement different components. Expansion and contraction.

FE expansion

1. Check the current number of FE services

```
$ kubectl --namespace doris get pods -l "app.kubernetes.io/component=fe"
NAME                      READY   STATUS    RESTARTS       AGE
doriscluster-sample-fe-0   1/1     Running   0              10d
```

2. Expansion FE

```
$ kubectl --namespace doris patch doriscluster doriscluster-sample --type merge --patch '{"spec":{"feSpec":{"replicas":3}}}'
```

3. Check expansion results

```
$ kubectl --namespace doris get pods -l "app.kubernetes.io/component=fe"
NAME                      READY   STATUS    RESTARTS   AGE
doriscluster-sample-fe-2   1/1     Running   0          9m37s
doriscluster-sample-fe-1   1/1     Running   0          9m37s
doriscluster-sample-fe-0   1/1     Running   0          8m49s
```

BE expansion

1. Check the current number of BE services

```
$ kubectl --namespace doris get pods -l "app.kubernetes.io/component=be"
NAME                       READY   STATUS    RESTARTS      AGE
doriscluster-sample-be-0   1/1     Running   0             3d2h
```

2. Expansion BE

```
$ kubectl --namespace doris patch doriscluster doriscluster-sample --type merge --patch '{"spec":{"beSpec":{"replicas":3}}}'
```

3. Check expansion results

```
$ kubectl --namespace doris get pods -l "app.kubernetes.io/component=be"
NAME                       READY   STATUS    RESTARTS      AGE
doriscluster-sample-be-0   1/1     Running   0             3d2h
doriscluster-sample-be-2   1/1     Running   0             12m
doriscluster-sample-be-1   1/1     Running   0             12m
```

Service contraction

Regarding the issue of node shrinkage, Doris-Operator currently does not support the safe shutdown of nodes. Here, the purpose of reducing FE or BE can still be achieved by reducing the replicas attribute of the cluster component. Here, the node is directly stopped to achieve node shutdown. line, the current version of Doris-Operator fails to implement decommission and goes offline after safely transferring the copy. This may cause some problems and precautions as follows:

- If the BE node is rashly taken offline when there is a single copy of the table, there will definitely be data loss, so avoid this operation as much as possible.
- FE Follower nodes try to avoid being offline at will, which may cause metadata damage and affect services.
- FE Observer type nodes can be taken offline at will without risk.
- The CN node does not hold a copy of the data and can be offline at will. However, the remote data cache existing on the CN node will be lost, resulting in a certain performance regression in data query within a short period of time.

How to enter the container when the service crashes

In the k8s environment, the service will enter the `CrashLoopBackOff` state due to some unexpected things. You can view the pod status and pod_name under the specified namespace through the `kubectl get pod --namespace ${namespace}` command.

In this state, the cause of the service problem cannot be determined simply by using the describe and logs commands. When the service enters the `CrashLoopBackOff` state, there needs to be a mechanism that allows the pod deploying the service to enter the `running` state so that users can enter the container for debugging through exec.

doris-operator provides a `debug` running mode. In essence, the debug process occupies the active detection port of the corresponding node, bypasses the k8s active detection mechanism, and creates a smoothly running container environment to facilitate users to enter and locate problems.

The following describes how to enter debug mode for manual debugging when the service enters `CrashLoopBackOff`, and how to return to normal startup state after solving the problem.

Start Debug mode

When a pod of the service enters CrashLoopBackOff or cannot be started normally during normal operation, take the following steps to put the service into `debug` mode and manually start the service to find the problem.

1.Use the following command to add annotation to the pod with problems.

```
$ kubectl annotate pod ${pod_name} --namespace ${namespace} selectdb.com.doris/runmode=debug
```

When the service is restarted next time, the service will detect the annotation that identifies the `debug` mode startup, and will enter the `debug` mode to start, and the pod status will be `running`.

2.When the service enters `debug` mode, the pod of the service is displayed in a normal state. Users can enter the inside of the pod through the following command

```
$ kubectl --namespace ${namespace} exec -ti ${pod_name} bash
```

3. Manually start the service under `debug`. When the user enters the pod, manually execute the `start_xx.sh` script by modifying the port of the corresponding configuration file. The script directory is under `/opt/apache-doris/xx/bin`.

FE needs to modify `query_port`, BE needs to modify `heartbeat_service_port` The main purpose is to avoid misleading the flow by accessing the crashed node through service in `debug` mode.

Exit Debug mode

When the service locates the problem, it needs to exit the `debug` operation. At this time, you only need to delete the corresponding pod according to the following command, and the service will start in the normal mode.

```
$ kubectl delete pod ${pod_name} --namespace ${namespace}
```

Notes

After entering the pod, you need to modify the port information of the configuration file before you can manually start the corresponding Doris component.

- FE needs to modify the `query_port=9030` configuration with the default path: `/opt/apache-doris/fe/conf/fe.conf`.
- BE needs to modify the `heartbeat_service_port=9050` configuration with the default path: `/opt/apache-doris/be/conf/be.conf`.

Deploying on AWS

To facilitate a quick experience of Doris on AWS, we have provided a CloudFormation template (CFT) that allows for rapid cluster launch and operation. With this template, you can automatically configure AWS resources and launch a Doris cluster with minimal configuration required.

Alternatively, you can also purchase AWS resources independently and deploy the cluster manually using standard methods.

What’s AWS CloudFormation?

CloudFormation enables users to create a “stack of resources” in just one step. Resources refer to the items created by users, such as EC2 instances, VPCs, subnets, and more. A group of such resources is referred to as a stack. Users can write a template that easily allows them to create a resource stack according to their preferences in a single step. This is faster, more repeatable, and offers better consistency compared to manual creation and configuration. Additionally, templates can be placed into source code for version control, enabling their use for any purpose whenever needed.

What’s Doris on AWS CloudFormation?

Currently, Doris provides the Doris CloudFormation Template, which allows users to quickly create a cluster of the relevant Doris version on AWS by directly using this template, enabling them to experience the latest Doris features.

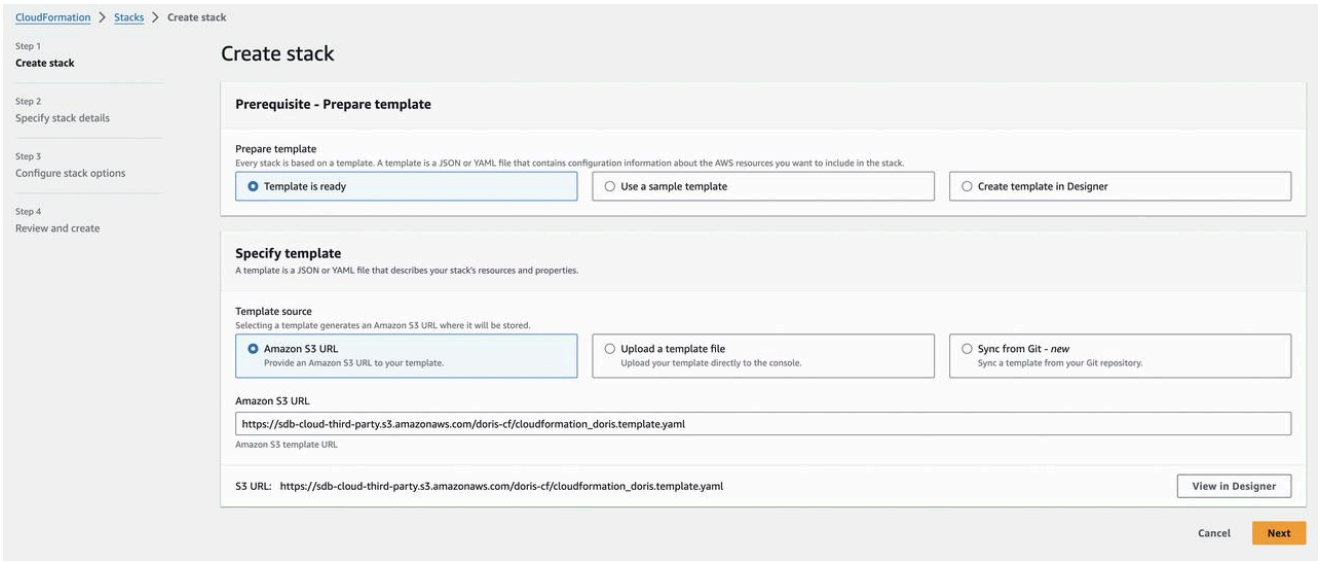
> ****Note: ** The template for building Doris clusters based on CloudFormation currently only supports the regions of us-east-1, us-west-1, and us-west-2. Doris on AWS CloudFormation is primarily intended for testing or experiencing purposes, and should not be used in production environments.**

Precautions for Use

- Determine the VPC and Subnet that will be deployed.
- Determine the key pair that will be used to log into the nodes.
- A VPC Endpoint Interface for S3 will be established during deployment.

Start Deployment

1. On the AWS console, navigate to CloudFormation and click on “Create stack”.

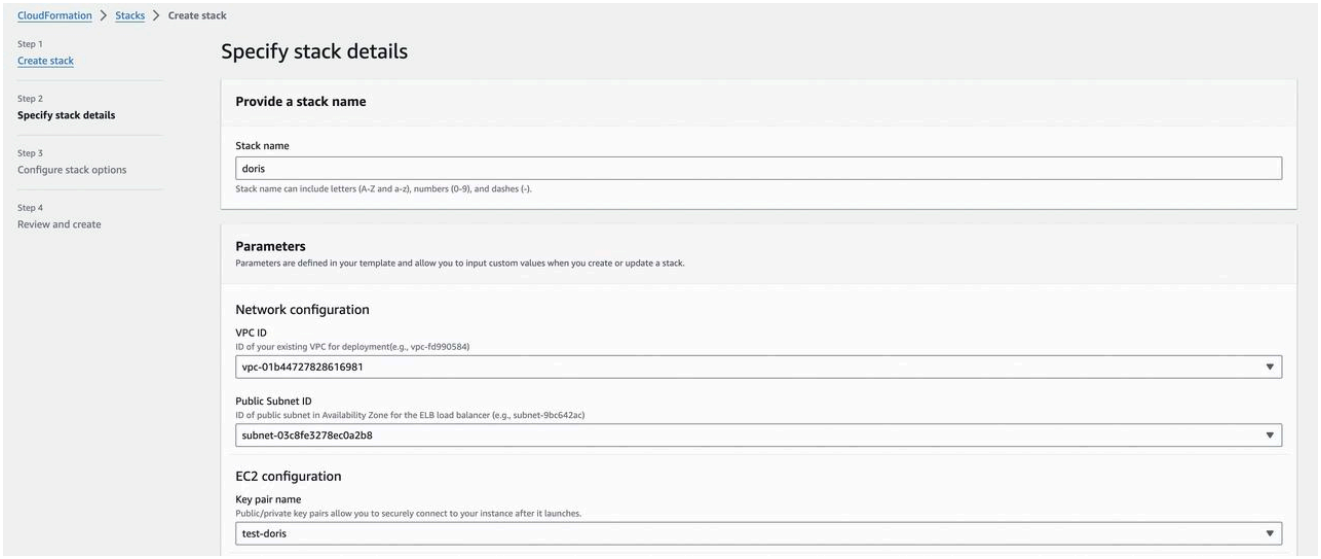


Start Deployment

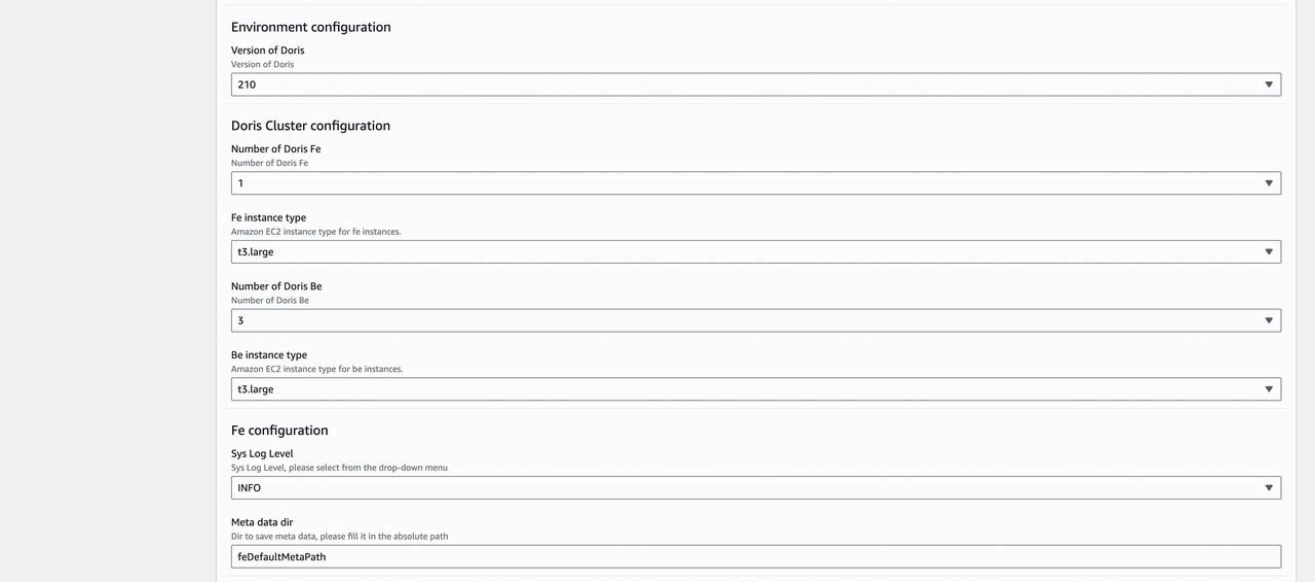
Select the “Amazon S3 URL Template source” option, and fill in the “Amazon S3 URL” field with the following template link:

https://sdb-cloud-third-party.s3.amazonaws.com/doris-cf/cloudformation_doris.template.yaml

2. Configure the specific parameters of the template



Configure the specific parameter



Configure the specific parameter



Configure the specific parameter

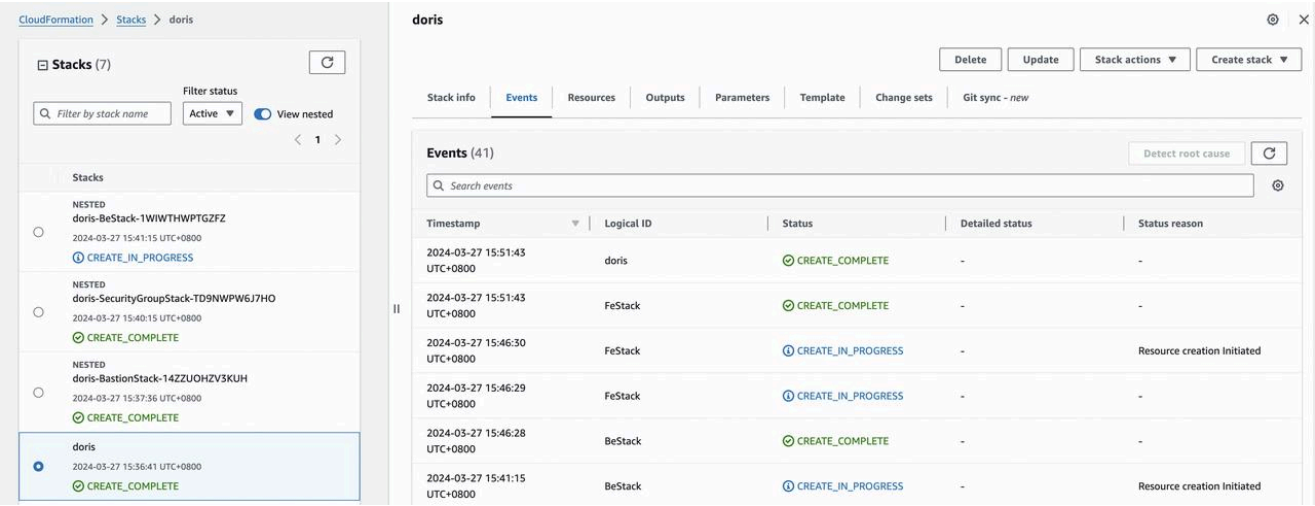
The main parameters are described as follows:

- **VPC ID**: The VPC where the deployment will be performed.
- **Subnet ID**: The subnet where the deployment will be deployed.
- **Key pair name**: The public/private key pairs used to connect to the deployed BE and FE nodes.
- **Version of Doris**: The version of Doris to be deployed, such as 2.1.0, 2.0.6, etc.
- **Number of Doris FE**: The number of FE nodes. The template defaults to selecting only 1 FE.
- **Fe instance type**: The node type of FE, and the default value can be used.
- **Number of Doris Be**: The number of BE nodes, which can be 1 or 3.
- **Be instance type**: The node type of BE, and the default value can be used.
- **Meta data dir**: The metadata directory of the FE node, and the default value can be used.
- **Sys log level:** Sets the level of system logs, and the default value of “info” can be used.
- **Volume type of Be nodes:** The volume type of EBS mounted on BE nodes. Each node is mounted with one disk by default. The default value can be used.

- **Volume size of Be nodes**: The size of EBS mounted on BE nodes, measured in GB. The default value can be used.

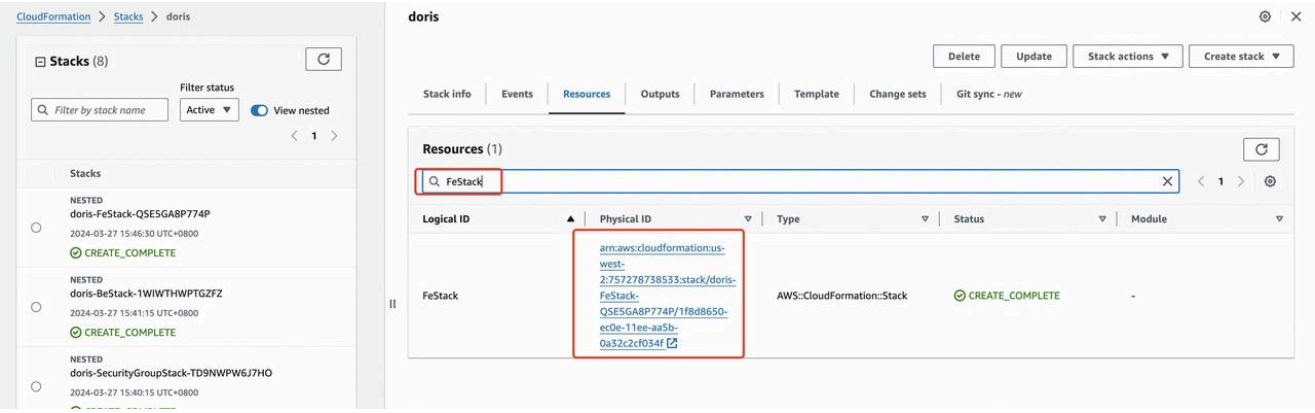
How to Connect to the Database

1. The display after successful deployment is as follows:

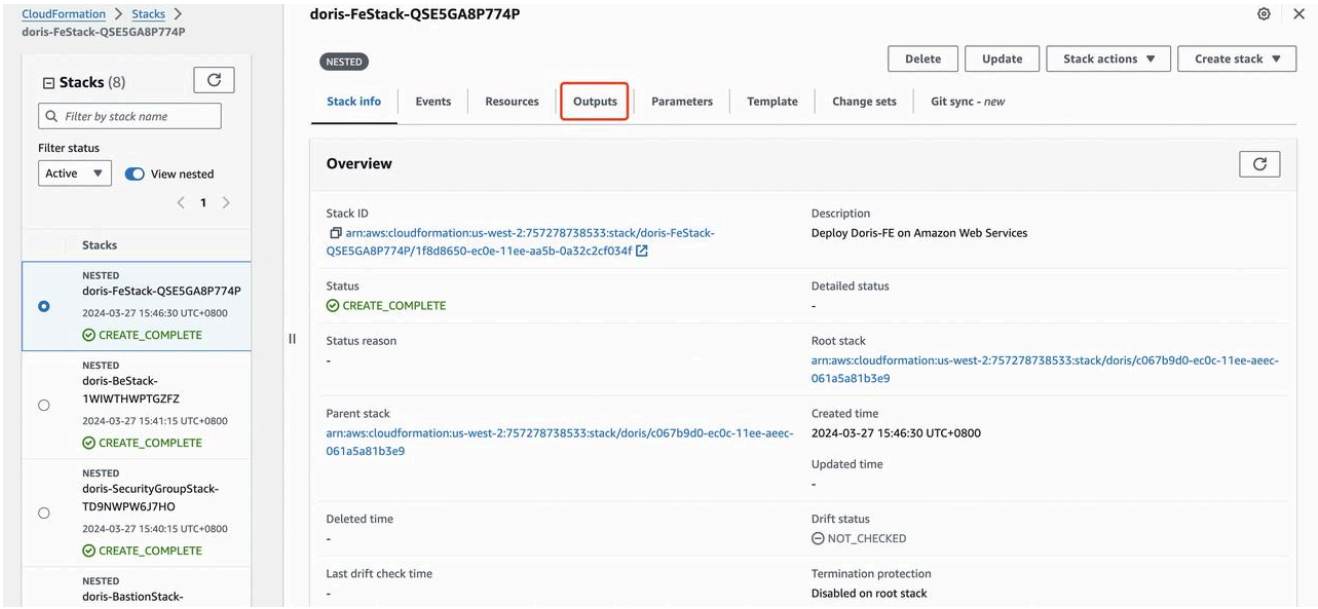


How to Connect to the Database

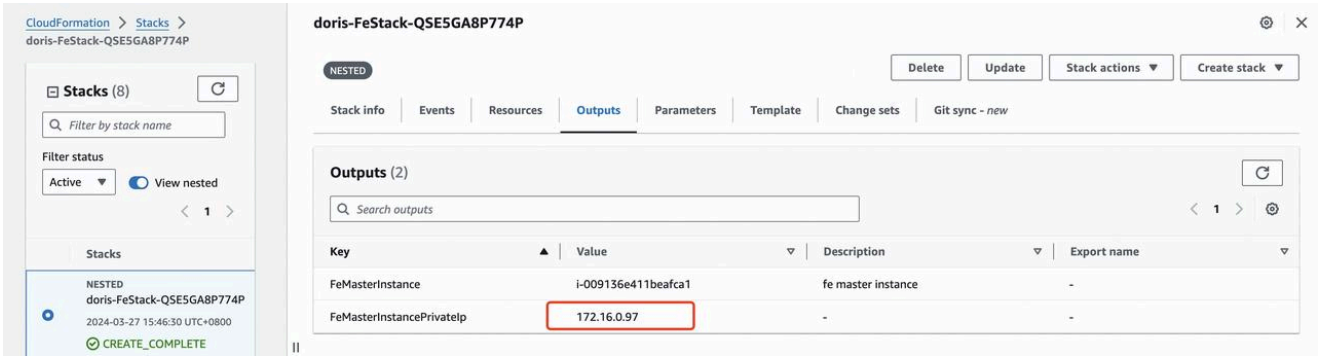
2. Next, find the connection address for FE as follows. In this example, you can view the address as 172.16.0.97 from the FE Outputs.



find the connection address for FE



find the connection address for FE



find the connection address for FE

3. To connect to the deployed Doris Cluster, here are some default values after deploying Doris using CloudFormation:

- **FE IP**: Obtain the IP address of FE following the steps in the previous section.
- **FE MySQL protocol port:** 9030
- **FE HTTP protocol port**: 8030
- **Default root password**: empty
- **Default admin password:** empty

Connecting to Database

Apache Doris adopts the MySQL network connection protocol. It is compatible with command-line tools, JDBC/ODBC drivers, and various visualization tools within the MySQL ecosystem. Additionally, Apache Doris comes with a built-in, easy-to-use Web UI. This guide is about how to connect to Doris using MySQL Client, MySQL JDBC Connector, DBeaver, and the built-in Doris Web UI.

MySQL Client

Download MySQL Client from the MySQL official website or use the pre-installed MySQL client for Linux provided by the Apache Doris community. Currently, Doris is primarily compatible with MySQL 5.7 and later clients.

Extract the downloaded MySQL client. In the `bin/` directory, find the `mysql` command-line tool. Execute the following command to connect to Doris:

```
# FE_IP represents the listening address of the FE node, while FE_QUERY_PORT represents the port of the MySQL protocol service of
the FE. This corresponds to the query_port parameter in fe.conf and it defaults to 9030.
mysql -h FE_IP -P FE_QUERY_PORT -u USER_NAME
```

After login, the following message will be displayed.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 236
Server version: 5.7.99 Doris version doris-2.0.3-rc06-37d31a5
Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective
owners.                                    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

MySQL JDBC Connector

Download the corresponding JDBC Connector from the official MySQL website.

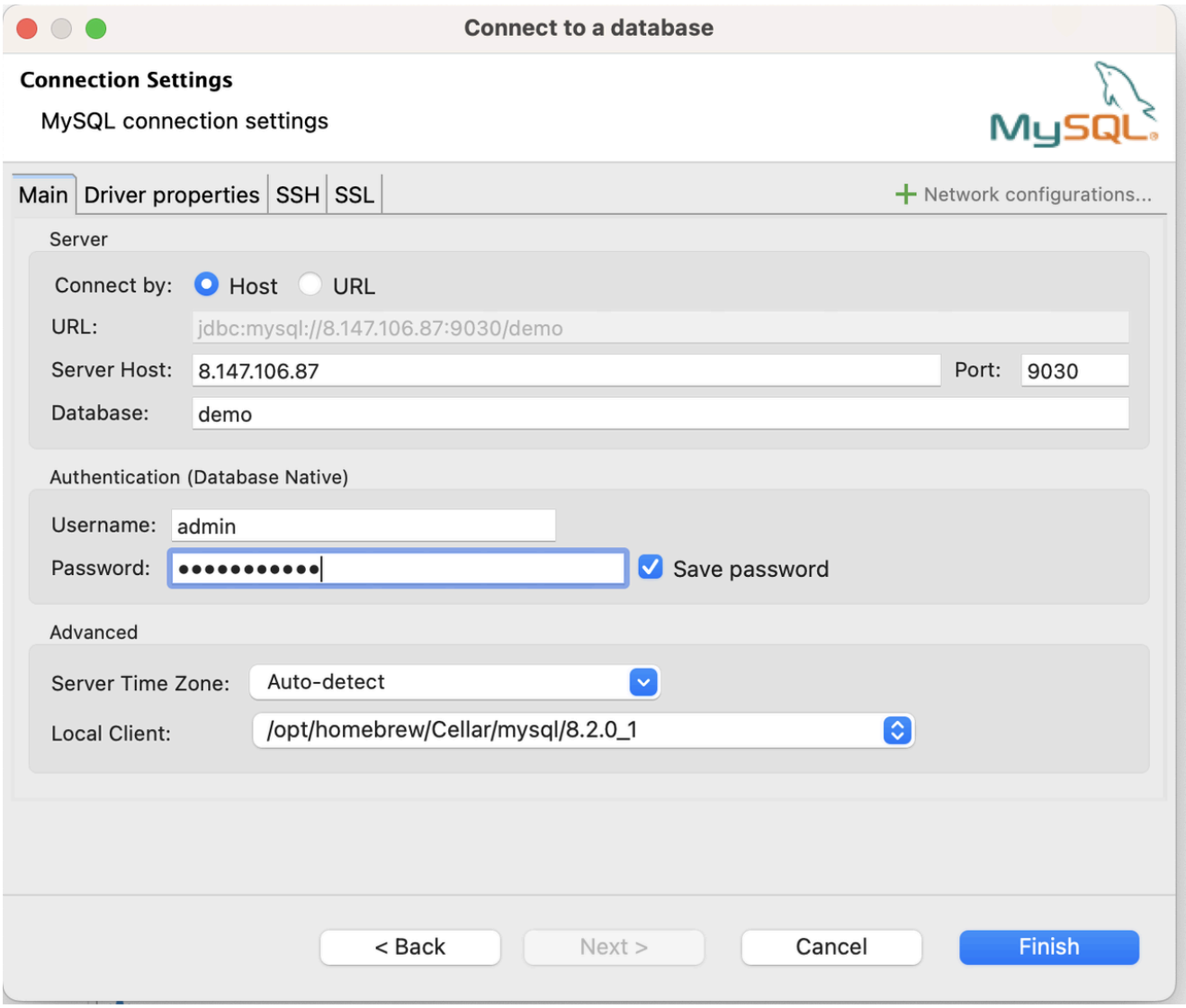
Example of connection code:

```
String user = "user_name";
String password = "user_password";
String newUrl = "jdbc:mysql://FE_IP:FE_PORT/demo?
useUnicode=true&characterEncoding=utf8&useTimezone=true&serverTimezone=Asia/Shanghai&useSSL=false&allowPublicKeyRetrieval=true";
try {
    Connection myCon = DriverManager.getConnection(newUrl, user, password);
    Statement stmt = myCon.createStatement();
    ResultSet result = stmt.executeQuery("show databases");
    ResultSetMetaData metaData = result.getMetaData();
    int columnCount = metaData.getColumnCount();
    while (result.next()) {
        for (int i = 1; i <= columnCount; i++) {
            System.out.println(result.getObject(i));
        }
    }
```

```
} catch (SQLException e) {
    log.error("get JDBC connection exception.", e);
}
```

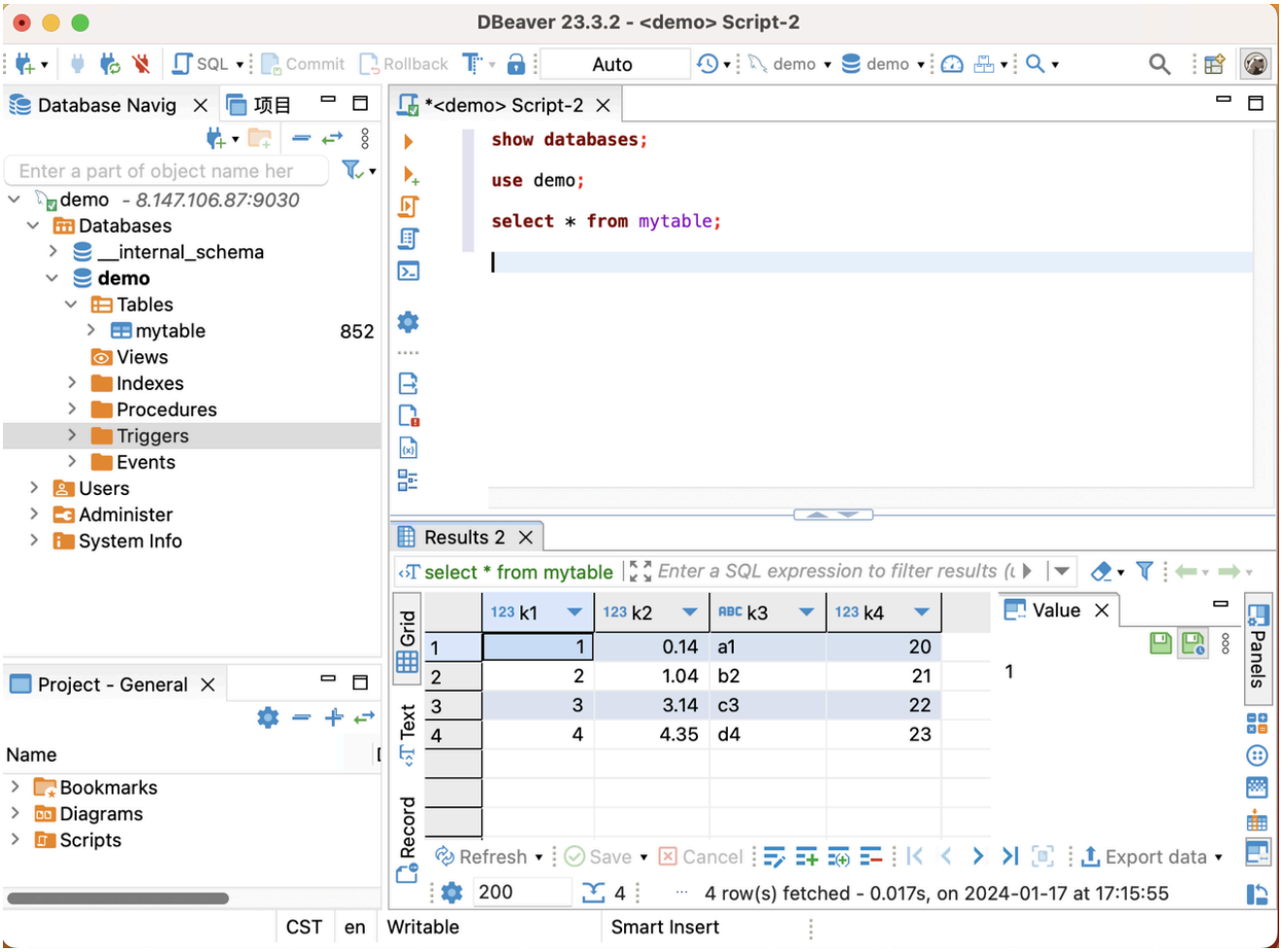
DBeaver

Create a MySQL connection to Apache Doris:



database-connect-dbeaver

Query in DBeaver:



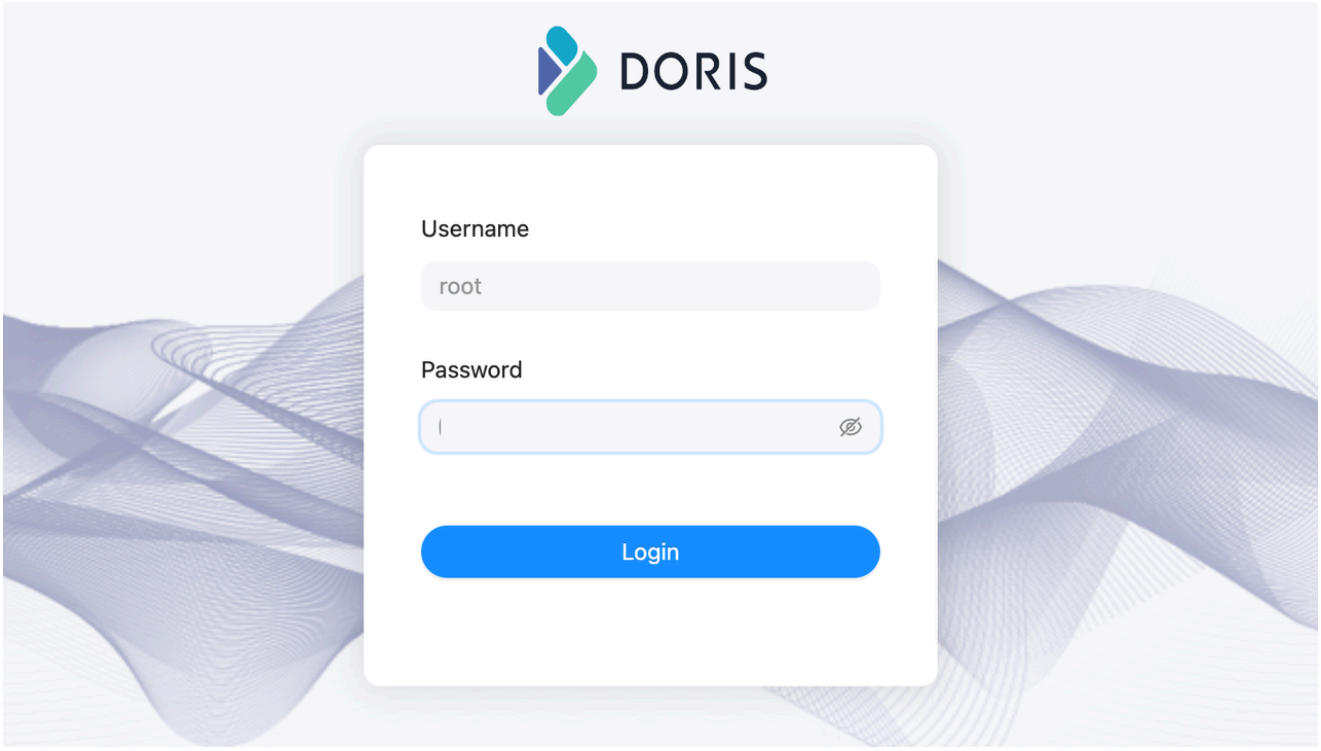
query-in-dbeaver

Built-in Web UI of Doris

Doris FE has a built-in Web UI. It allows users to perform SQL queries and view other related information without the need to install the MySQL client

To access the Web UI, simply enter the URL in a web browser: http://fe_ip:fe_port, for example, `http://172.20.63.118:8030`. This will open the built-in Web console of Doris.

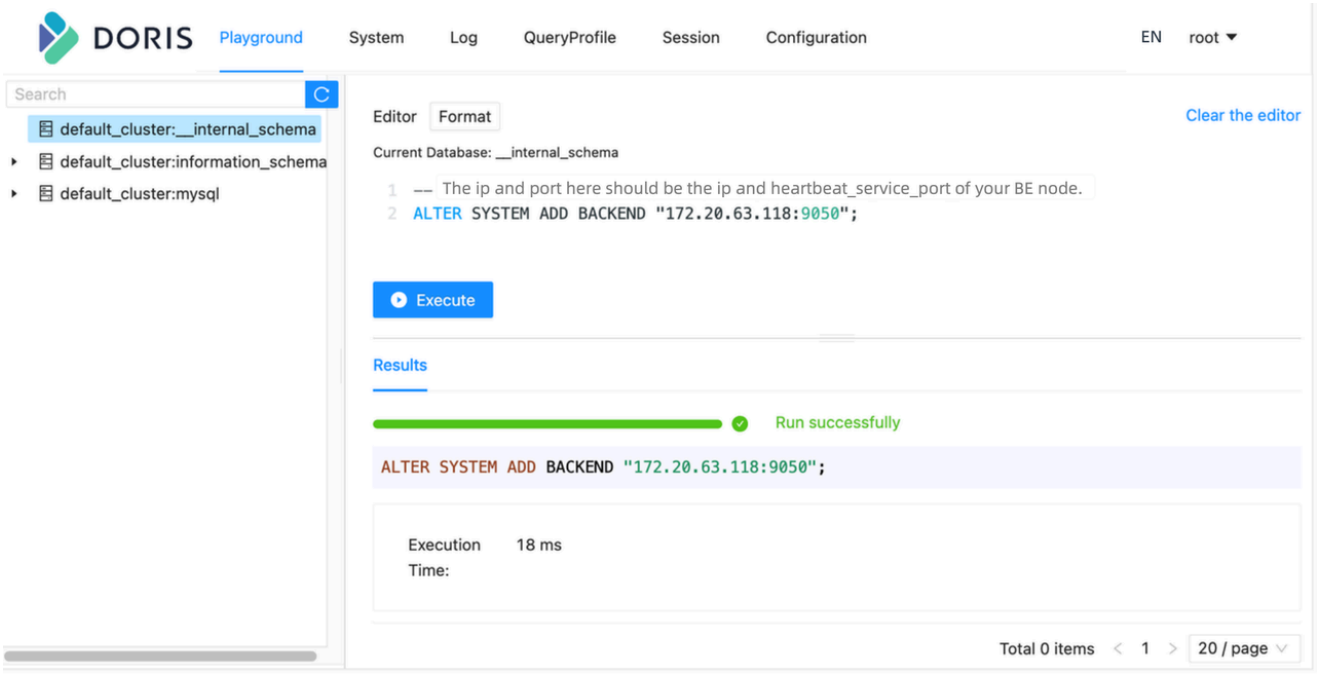
The built-in Web console is primarily intended for use by the root account of the cluster. By default, the root account password is empty after installation.



web-login-username-password

For example, you can execute the following command in the Playground to add a BE node.

```
ALTER SYSTEM ADD BACKEND "be_host_ip:heartbeat_service_port";
```



Doris-Web-UI-Playground-en

> **For successful execution of statements that are not related to specific databases/tables in the Playground, it is necessary to randomly select a database from the left-hand database panel. This limitation will be removed later.**

> The current built-in web console cannot execute SET type SQL statements. Therefore, the web console does not support statements like SET PASSWORD FOR ‘user’ = PASSWORD(‘user_password’).

Data Types

Apache Doris support standard SQL syntax, using MySQL Network Connection Protocol, highly compatible with MySQL syntax protocol. Therefore, in terms of data type support, Apache Doris aligns as closely as possible with MySQL-related data types

The list of data types supported by Doris is as follows:

Numeric data type §

Type Name	Storage (bytes)	Description
BOOLEAN	1	Boolean data type that stores only two values: 0 represents false, 1 represents true.
TINYINT	1	Integer value, signed range is from -128 to 127.
SMALLINT	2	Integer value, signed range is from -32768 to 32767.
INT	4	Integer value, signed range is from -2147483648 to 2147483647.
BIGINT	8	Integer value, signed range is from -9223372036854775808 to 9223372036854775807.
LARGEINT	16	Integer value, range is [-2^127 + 1 to 2^127 - 1].
FLOAT	4	Single precision floating point number, range is [-3.4 * 10^38 to 3.4 * 10^38].
DOUBLE	8	Double precision floating point number, range is [-1.79 * 10^308 to 1.79 * 10^308].
DECIMAL	4/8/16	An exact fixed-point number defined by precision (total number of digits) and scale (number of digits to the right of the decimal point). Format: DECIMAL(M[,D]), where M is precision and D is scale. The range for M is [1, 38], and for D is [0, precision]. Storage requirements: - 4 bytes for 0 < precision <= 9, - 8 bytes for 9 < precision <= 18, - 16 bytes for 18 < precision <= 38.

Datetime data type §

Type name	Storeage (bytes)	Description
DATE	16	DATE holds values for a calendar year, month and day, the supported range is ['0000-01-01', '9999-12-31']. Default print format: 'yyyy-MM-dd'.
DATETIME	16	A DATE and TIME combination Format: DATETIME ([P]). The optional parameter P represents time precision, with a value range of [0,6], supporting up to 6 decimal places (microseconds). When not set, it is 0. The supported range is ['0000-01-01 00:00:00 [.000000]', '9999-12-31 23:59:59 [.999999]']. Default print format: 'yyy-MM-dd HH: mm: ss. SSSSSS'.

String data type §

Type name	Storeage (bytes)	Description
CHAR	M	A FIXED length string, the parameter M specifies the column length in characters. The range of M is from 1 to 255.
VARCHAR	Variable Length	A VARIABLE length string , the parameter M specifies the maximum string length in characters. The range of M is from 1 to 65533. The variable-length string is stored in UTF-8 encoding. English characters occupy 1 byte, and Chinese characters occupy 3 bytes.
STRING	Variable Length	A VARIABLE length string, default supports 1048576 bytes (1 MB), and a limit of maximum precision of 2147483643 bytes (2 GB). Size can be configured string_type_length_soft_limit_bytes adjusted through BE. String type can only be used in value column, not in key column and partition bucket column.

Semi-structured data type §

Type name	Storeage (bytes)	Description
ARRAY	Variable Length	Arrays composed of elements of type T cannot be used as key columns. Currently supported for use in tables with Duplicate and Unique models.
MAP	Variable Length	Maps consisting of elements of type K and V, cannot be used as Key columns. These maps are currently supported in tables using the Duplicate and Unique models.
STRUCT	Variable Length	A structure composed of multiple Fields can also be understood as a collection of multiple columns. It cannot be used as a Key. Currently, STRUCT can only be used in tables of Duplicate models. The name and number of Fields in a Struct are fixed and are always Nullable.
JSON	Variable Length	Binary JSON type, stored in binary JSON format, access internal JSON fields through JSON function. Supported up to 1048576 bytes (1MB) by default, and can be adjusted to a maximum of 2147483643 bytes (2GB). This limit can be modified through the BE configuration parameter ‘jsonb_type_length_soft_limit_bytes’.
VARIANT	Variable Length	The VARIANT data type is dynamically adaptable, specifically designed for semi-structured data like JSON. It can store any JSON object and automatically splits JSON fields into subcolumns for improved storage efficiency and query performance. The length limits and configuration methods are the same as for the STRING type. However, the VARIANT type can only be used in value columns and cannot be used in key columns or partition / bucket columns.

Aggregation data type §

Type name	Storeage (bytes)	Description
HLL	Variable Length	HLL stands for HyperLogLog, is a fuzzy deduplication. It performs better than Count Distinct when dealing with large datasets. The error rate of HLL is typically around 1%, and sometimes it can reach 2%. HLL cannot be used as a key column, and the aggregation type is HLL_UNION when creating a table. Users do not need to specify the length or default value as it is internally controlled based on the aggregation level of the data. HLL columns can only be queried or used through the companion functions such as hll_union_agg, hll_raw_agg, hll_cardinality, and hll_hash.

Type name	Storeage (bytes)	Description
BITMAP	Variable Length	BITMAP type can be used in Aggregate tables, Unique tables or Duplicate tables. - When used in a Unique table or a Duplicate table, BITMAP must be employed as non-key columns. - When used in an Aggregate table, BITMAP must also serve as non-key columns, and the aggregation type must be set to BITMAP_UNION during table creation. Users do not need to specify the length or default value as it is internally controlled based on the aggregation level of the data. BITMAP columns can only be queried or used through the companion functions such as bitmap_union_count, bitmap_union, bitmap_hash, and bitmap_hash64.
QUANTILE_STATE	Variable Length	A type used to calculate approximate quantile values. When loading, it performs pre-aggregation for the same keys with different values. When the number of values does not exceed 2048, it records all data in detail. When the number of values is greater than 2048, it employs the TDigest algorithm to aggregate (cluster) the data and store the centroid points after clustering. QUANTILE_STATE cannot be used as a key column and should be paired with the aggregation type QUANTILE_UNION when creating a table. Users do not need to specify the length or default value as it is internally controlled based on the aggregation level of the data. QUANTILE_STATE columns can only be queried or used through the companion functions such as QUANTILE_PERCENT, QUANTILE_UNION, and TO_QUANTILE_STATE.
AGG_STATE	Variable Length	Aggregate function can only be used with state/merge/union function combiners. AGG_STATE cannot be used as a key column. When creating a table, the signature of the aggregate function needs to be declared alongside. Users do not need to specify the length or default value. The actual data storage size depends on the function's implementation.

You can also view all the data types supported by Doris with the `SHOW DATA TYPES;`statement.

Table Model Overview

When creating a table in Doris, it is necessary to specify the table model to define how data is stored and managed. Doris provides three table models: the **Duplicate Key Model**, **Unique Key Model** and **Aggregate Key Model**, which cater to different application scenarios. Each model has corresponding mechanisms for data deduplication, aggregation, and updates. Choosing the appropriate table model helps achieve business objectives while ensuring flexibility and efficiency in data processing.

Table Model Classification

Doris supports three types of table models:

- Duplicate Key Model: Allows the specified Key columns to be duplicated, and Doris's storage layer retains all written data. This model is suitable for situations where all original data records must be preserved.
- Unique Key Model: Ensures that each row has a unique Key value, and guarantees that there are no duplicate rows for a given Key column. The Doris storage layer retains only the latest written data for each key, making this model suitable for scenarios that involve data updates.
- Aggregate Key Model: Allows data to be aggregated based on the Key columns. The Doris storage layer retains aggregated data, reducing storage space and improving query performance. This model is typically used in situations where summary or aggregated information (such as totals or averages) is required.

After creating the table, the properties of the table model are confirmed and cannot be modified. Choosing the right model for the business is crucial:

- **Duplicate Key Model** is suitable for ad-hoc queries with any dimensions. Although it cannot leverage the benefits of pre-aggregation, it is not constrained by aggregation models and can take advantage of the columnar storage model (only reading relevant columns without needing to read all key columns).
- **Unique Key Model** is designed for scenarios where a unique key constraint is needed, ensuring the uniqueness of the key. However, it cannot utilize the query benefits brought by pre-aggregations such as ROLLUP.
- **Aggregate Key Model** can greatly reduce the data and computation required for aggregation queries through pre-aggregation, making it ideal for fixed-schema reporting queries. However, this model is not friendly to `count(*)` queries. Also, because the aggregation method for the Value columns is fixed, when performing other types of aggregation queries, semantic correctness must be considered.
- **Update partial columns**, please refer to the documentation for Partial Column Updates in Unique Key Model and Partial Column Updates in Aggregate Model for relevant usage advice.

Sort Key

In Doris, data is stored in a columnar format, and a table can be divided into Key columns and Value columns. The Key columns are used for grouping and sorting, while the Value columns are used for aggregation. Key columns can consist of one or more fields, and when creating a table, data is sorted and stored according to the columns of Aggregate Key, Unique Key, and Duplicate Key models.

Different table models require the specification of Key columns during table creation, each with a different significance: for the Duplicate Key model, the Key columns represent sorting, without any uniqueness constraints. In the Aggregate Key and Unique Key models, aggregation is performed based on the Key columns, which not only have sorting capabilities but also enforce uniqueness constraints.

Proper use of the Sort Key can provide the following benefits:

- **Accelerated Query Performance**: Sort keys help reduce the amount of data that needs to be scanned. For range queries or filtering queries, the sort key can directly locate the data. For queries that require sorting, the sort key can also accelerate the sorting process.
- **Data Compression Optimization**: Storing data in an ordered fashion based on the sort key improves compression efficiency, as similar data will be grouped together, significantly increasing the compression ratio and reducing storage space.
- **Reduced Deduplication Costs**: When using the Unique Key Model, the sort key allows Doris to perform deduplication more efficiently, ensuring data uniqueness.

When selecting a sort key, the following recommendations can be followed:

- The Key columns must come before all Value columns.
- Preferably choose integer types. This is because integer types are much more efficient in computation and lookup than strings.
- For selecting different lengths of integer types, follow the principle of choosing what is sufficient.
- For the length of `VARCHAR` and `STRING` types, follow the principle of choosing enough…

Table Model Comparison

	Duplicate Key Model	Unique Key Model	Aggregate Key Model
Key Column Uniqueness	Not Supported, Key columns can be duplicated	Supported	Supported
Synchronous Materialized View	Supported	Supported	Supported
Asynchronous Materialized View	Supported	Supported	Supported
UPDATE Statement	Not Supported	Supported	Not Supported
DELETE Statement	Partially Supported	Supported	Not Supported

	Duplicate Key Model	Unique Key Model	Aggregate Key Model
Full Row Update on Import	Not Supported	Supported	Not Supported
Partial Column Update on Import	Not Supported	Supported	Partially Supported

Duplicate Key Model

The **Duplicate Key Model** in Doris is the default table model, designed to store individual raw data records. The `Duplicate Key` specified during table creation determines the columns for sorting and storage, optimizing common queries. It is recommended to choose no more than three columns as the sort key. For more specific selection guidelines, refer to Sort Key. The Duplicate Key Model has the following characteristics:

- **Preserving Raw Data**: The Duplicate Key Model retains all original data, making it ideal for storing and querying raw data. It is recommended for use cases requiring detailed data analysis to avoid data loss.
- **No Deduplication or Aggregation**: Unlike the Aggregate and Primary Key models, the Duplicate Key Model does not perform deduplication or aggregation, fully retaining identical records.
- **Flexible Data Querying**: The Duplicate Key Model retains all original data, enabling detailed extraction and aggregation across any dimension for metadata auditing and fine-grained analysis.

Use Cases

In the Duplicate Key Model, data is generally only appended, and old data is not updated. The Duplicate Key Model is ideal for scenarios that require full raw data:

- **Log Storage**: Used for storing various types of application logs, such as access logs, error logs, etc. Each piece of data needs to be detailed for future auditing and analysis.
- **User Behavior Data**: When analyzing user behavior, such as click data or user access paths, it is necessary to retain detailed user actions. This helps in building user profiles and conducting detailed analysis of behavior patterns.
- **Transaction Data**: For storing transaction or order data, once a transaction is completed, there is typically no need for data changes…

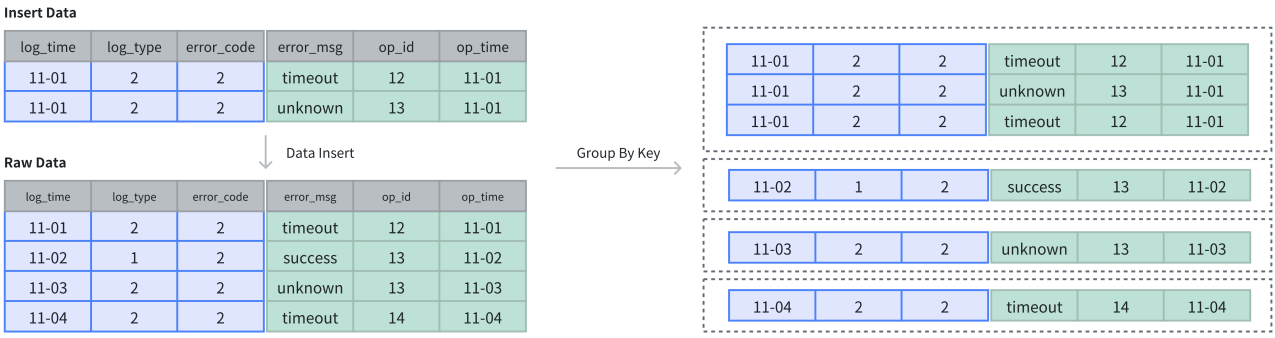
Table Creation Instructions

When creating a table, the **DUPLICATE KEY** keyword can be used to specify the Duplicate Key Model. The Duplicate Key table must specify the Key columns, which are used to sort the data during storage. In the following example, the Duplicate Key table stores log information and sorts the data based on the `log_time`, `log_type`, and `error_code` columns:

```
CREATE TABLE IF NOT EXISTS example_tbl_duplicate
(
    log_time        DATETIME       NOT NULL,
    log_type        INT            NOT NULL,
    error_code      INT,
    error_msg       VARCHAR(1024),
    op_id           BIGINT,
    op_time         DATETIME
)
DUPLICATE KEY(log_time, log_type, error_code)
DISTRIBUTED BY HASH(log_type) BUCKETS 10;
```

Data Insertion and Storage

In a Duplicate Key table, data is not deduplicated or aggregated; inserting data directly stores it. The Key columns in the Duplicate Key Model are used for sorting.



columnar_storage

In the example above, after inserting 2 rows into the initial 4 rows, the data is appended, resulting in a total of 6 rows.

```
-- 4 rows raw data
INSERT INTO example_tbl_duplicate VALUES
('2024-11-01 00:00:00', 2, 2, 'timeout', 12, '2024-11-01 01:00:00'),
('2024-11-02 00:00:00', 1, 2, 'success', 13, '2024-11-02 01:00:00'),
('2024-11-03 00:00:00', 2, 2, 'unknown', 13, '2024-11-03 01:00:00'),
('2024-11-04 00:00:00', 2, 2, 'unknown', 12, '2024-11-04 01:00:00');

-- insert into 2 rows
INSERT INTO example_tbl_duplicate VALUES
('2024-11-01 00:00:00', 2, 2, 'timeout', 12, '2024-11-01 01:00:00'),
('2024-11-01 00:00:00', 2, 2, 'unknown', 13, '2024-11-01 01:00:00');

-- check the rows of table
SELECT * FROM example_tbl_duplicate;
+---------------------+----------+------------+-----------+-------+---------------------+
| log_time            | log_type | error_code | error_msg | op_id | op_time             |
+---------------------+----------+------------+-----------+-------+---------------------+
| 2024-11-02 00:00:00 |        1 |          2 | success   |    13 | 2024-11-02 01:00:00 |
| 2024-11-01 00:00:00 |        2 |          2 | timeout   |    12 | 2024-11-01 01:00:00 |
| 2024-11-03 00:00:00 |        2 |          2 | unknown   |    13 | 2024-11-03 01:00:00 |
| 2024-11-04 00:00:00 |        2 |          2 | unknown   |    12 | 2024-11-04 01:00:00 |
| 2024-11-01 00:00:00 |        2 |          2 | unknown   |    13 | 2024-11-01 01:00:00 |
| 2024-11-01 00:00:00 |        2 |          2 | timeout   |    12 | 2024-11-01 01:00:00 |
+---------------------+----------+------------+-----------+-------+---------------------+
```

Unique Key Model

When data updates are required, use the **Unique Key Model**. It guarantees the uniqueness of the Key columns so that new data overwrites existing records with matching keys, ensuring that only the most up-to-date records are maintained. This model is ideal for update scenarios, enabling unique-key-level updates during data insertion. The Unique Key Model has the following characteristics:

- **Unique Key UPSERT**: During insertion, records with duplicate keys are updated, while new keys are inserted.
- **Automatic Deduplication**: The model ensures key uniqueness and automatically deduplicates data based on the unique key.
- **Optimized for High-frequency Updates**: It efficiently handles high-frequency updates while balancing update and query performance.

Use Cases

- **High-frequency Data Updates**: In upstream OLTP databases, where dimension tables are frequently updated, the Unique Key Model can efficiently synchronize the upstream updated records and perform efficient UPSERT operations.
- **Efficient Data Deduplication**: In scenarios such as advertising campaigns or customer relationship management systems, where deduplication is required based on user IDs, the Unique Key Model ensures efficient deduplication.
- **Partial Columns Updates**: In scenarios such as in user profiling where dynamic tags change frequently, or in order consumption scenarios where the transaction status needs to be updated. The Unique Key Model's partial column update capability allows for changes to specific columns.

Implementation Methods

In Doris, the Unique Key Model has two implementation methods:

- **Merge-on-write**: Starting from version 1.2, the default implementation of the Unique Key Model in Doris is the merge-on-write mode. In this mode, data is immediately merged for the same Key upon writing, ensuring that the data storage state after each write is the final merged result of the unique key, and only the latest result is stored. Merge-on-write provides a good balance between query and write performance, avoiding the need to merge multiple versions of data during queries and ensuring predicate pushdown to the storage layer. The merge-on-write model is recommended for most scenarios.
- **Merge-on-read**: Prior to version 1.2, Doris's Unique Key Model defaulted to merge-on-read mode. In this mode, data is not merged upon writing but is appended incrementally, retaining multiple versions within Doris. During queries or Compaction, data is merged by the same Key version. Merge-on-read is suitable for write-heavy and read-light scenarios, but during queries, multiple versions must be merged, and predicates cannot be pushed down, which may affect query speed.

In Doris, there are two types of update semantics for the Unique Key Model:

- **Full Row Upsert**: The default update semantic for the Unique Key Model is **full row UPSERT**, i.e., UPDATE OR INSERT. If the Key of the row exists, it will be updated; if it does not exist, new data will be inserted. In the full row UPSERT semantic, even if the user inserts data into specific columns using `INSERT INTO`, Doris will fill in the missing columns with NULL values or default values during the planner stage.
- **Partial Column Upsert**: If users want to update specific fields, they need to use the merge-on-write implementation and enable partial column updates support via specific parameters. Please refer to the documentation on Partial Column Updates.

Merge-on-write

Creating a Merge-on-write Table

To create a Unique Key table, use the `UNIQUE KEY` keyword. Enable merge-on-write mode by setting the `enable_unique_key_merge_on_write` attribute (default since Doris 2.1):

```
CREATE TABLE IF NOT EXISTS example_tbl_unique
(
    user_id         LARGEINT        NOT NULL,
    user_name       VARCHAR(50)     NOT NULL,
    city            VARCHAR(20),
    age             SMALLINT,
    sex             TINYINT
)
UNIQUE KEY(user_id, user_name)
DISTRIBUTED BY HASH(user_id) BUCKETS 10
PROPERTIES (
    "enable_unique_key_merge_on_write" = "true"
);
```

Merge-on-read

Creating a Merge-on-read Table

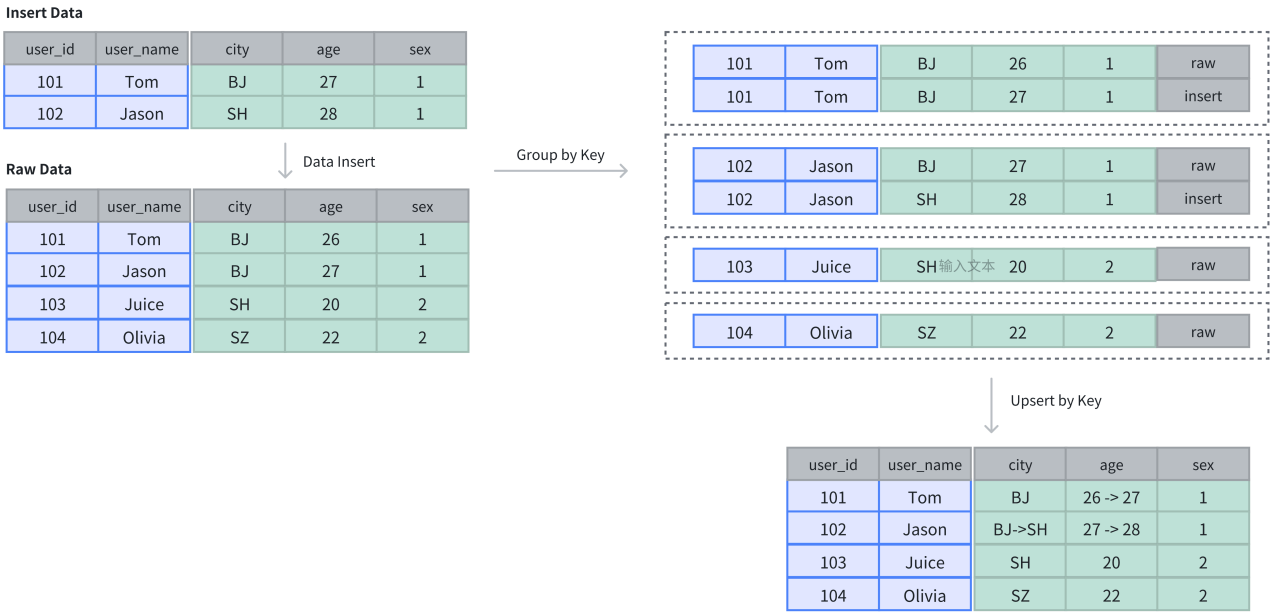
When creating a table, the `UNIQUE KEY` keyword can be used to specify a Unique Key table. The merge-on-read mode can be enabled by explicitly disabling the `enable_unique_key_merge_on_write` attribute. Before Doris version 2.1, the merge-on-read mode was enabled by default:

```
CREATE TABLE IF NOT EXISTS example_tbl_unique
(
    user_id         LARGEINT        NOT NULL,
    username        VARCHAR(50)     NOT NULL,
    city            VARCHAR(20),
    age             SMALLINT,
    sex             TINYINT
)
UNIQUE KEY(user_id, username)
DISTRIBUTED BY HASH(user_id) BUCKETS 10
PROPERTIES (
```

```
    "enable_unique_key_merge_on_write" = "false"
);
```

Data Insertion and Storage

In a Unique Key table, the Key columns serve both for sorting and deduplication. New insertions overwrite existing records with matching keys.



unique-key-model-insert

As shown in the example, there were 4 rows of data in the original table. After inserting 2 new rows, the newly inserted rows are updated based on the unique key:

```
-- insert into raw data
INSERT INTO example_tbl_unique VALUES
(101, 'Tom', 'BJ', 26, 1),
(102, 'Jason', 'BJ', 27, 1),
(103, 'Juice', 'SH', 20, 2),
(104, 'Olivia', 'SZ', 22, 2);

-- insert into data to update by key
INSERT INTO example_tbl_unique VALUES
(101, 'Tom', 'BJ', 27, 1),
(102, 'Jason', 'SH', 28, 1);

-- check updated data
SELECT * FROM example_tbl_unique;
+---------+----------+------+------+------+
| user_id | username | city | age  | sex  |
+---------+----------+------+------+------+
| 101     | Tom      | BJ   |   27 |    1 |
| 102     | Jason    | SH   |   28 |    1 |
| 104     | Olivia   | SZ   |   22 |    2 |
| 103     | Juice    | SH   |   20 |    2 |
+---------+----------+------+------+------+
```

Notes

- The implementation mode for Unique Key tables is fixed at creation and cannot be changed via schema alterations.
- Under full row UPSERT semantics, if specific columns are omitted during insertion, Doris fills them with NULL or default values during planning.
- For partial column upsert, enable merge-on-write mode with the appropriate parameters. Refer to Partial Column Updates for guidance.

Aggregate Model

Doris's **Aggregate Key Model** is designed to efficiently handle aggregation operations in large-scale data queries. By performing pre-aggregation on the data, it reduces redundancy in computations and improves query performance. The model stores only aggregated data, omitting raw data, which saves storage space and enhances query performance.

Use Cases

- **Summarizing Detailed Data**: The Aggregate Key Model is used in scenarios like e-commerce platforms evaluating monthly sales, financial risk control calculating customer transaction totals, or advertising campaigns analyzing total ad clicks, for multidimensional summarization of detailed data.
- **No Need to Query Raw Detailed Data**: For use cases such as dashboard reports or user transaction behavior analysis, where the raw data is stored in a data lake and does not need to be retained in the database, only the aggregated data is stored.

Principle

Each data import creates a version in the Aggregate Key Model, and during the **Compaction** stage, versions are merged. When querying, data is aggregated by the primary key:

- **Data Import Stage**
    - Data is imported into the aggregate key table in batches, with each batch creating a new version.
    - Within each version, data with the same aggregation keys is pre-aggregated (e.g., sum, count, etc.).
- **Background File Merging Stage (Compaction)**
    - Multiple batches generate multiple version files, which are periodically merged into a larger version file.
    - During the merge process, data with the same aggregation key is re-aggregated to reduce redundancy and optimize storage.
- **Query Stage**
    - During queries, the system aggregates data with the same aggregation key from all versions to ensure accurate results.
    - This process ensures that aggregation operations are performed efficiently, even with large data volumes. The aggregated results are optimized for fast querying, providing a significant performance improvement over raw data queries

Table Creation Instructions

When creating a table, the **AGGREGATE KEY** keyword can be used to specify the Aggregate Key Model. The Aggregate Key Model must specify Key columns, which are used to aggregate Value columns during storage.

```
CREATE TABLE IF NOT EXISTS example_tbl_agg
(
    user_id             LARGEINT    NOT NULL,
    load_dt             DATE        NOT NULL,
    city                VARCHAR(20),
    last_visit_dt       DATETIME    REPLACE DEFAULT "1970-01-01 00:00:00",
    cost                BIGINT      SUM DEFAULT "0",
    max_dwell           INT         MAX DEFAULT "0",
)
AGGREGATE KEY(user_id, load_dt, city)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

In the example above, a fact table for user information and access behavior is defined, where `user_id`, `load_dt`, and `city` are used as Key columns for aggregation. During data import, the Key columns are aggregated into one row, and the Value columns are aggregated according to the specified aggregation types.

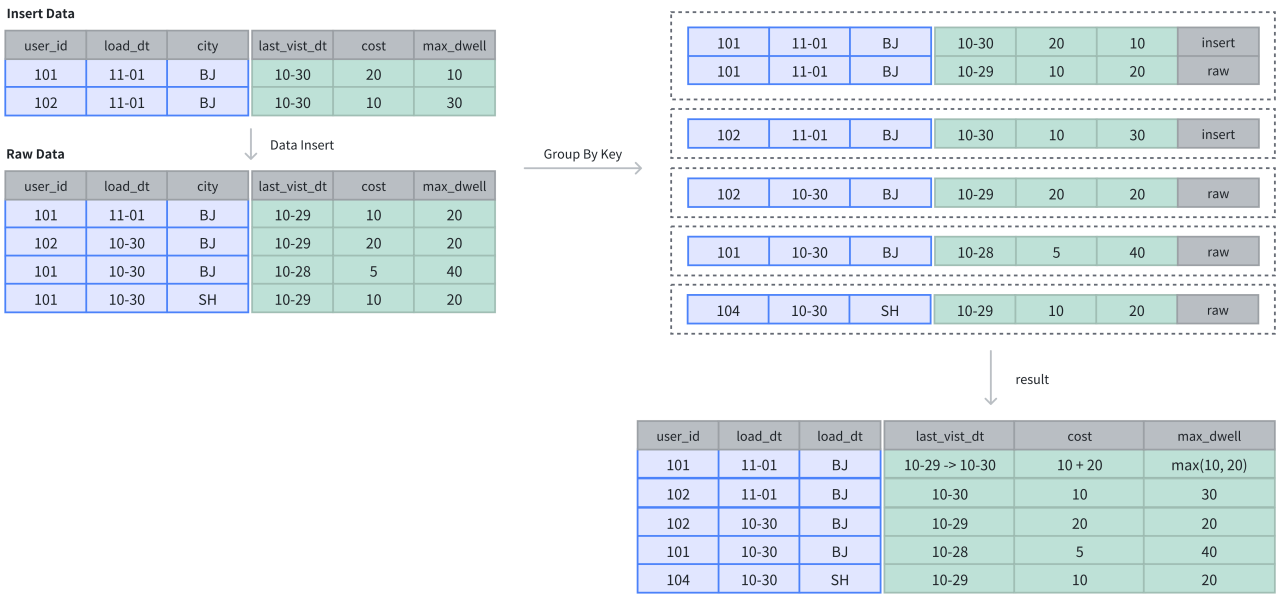
The following types of dimension aggregation are supported in the Aggregate Key Model:

Aggregation Method	Description
SUM	Sum, accumulates multiple Value rows.
REPLACE	Replacement, the Value in the next batch replaces the previously inserted Value.
MAX	Retain the maximum value.
MIN	Retain the minimum value.
REPLACE_IF_NOT_NULL	Replace non-null values. Unlike REPLACE, null values are not replaced.
HLL_UNION	Aggregation method for HLL type columns, using the HyperLogLog algorithm.
BITMAP_UNION	Aggregation method for BITMAP type columns, performing bitmap union aggregation.

> **Tip:**
>
> If the aggregation methods above do not meet your business requirements, consider using the `agg_state` type.

Data Insertion and Storage

In the Aggregate Key table, data is aggregated based on the primary key. After data insertion, aggregation operations are completed.



aggrate-key-model-insert

In the example above, there were originally 4 rows of data in the table. After inserting 2 rows, aggregation operations on the dimension columns are performed based on the Key columns:

```
-- 4 rows raw data
INSERT INTO example_tbl_agg VALUES
(101, '2024-11-01', 'BJ', '2024-10-29', 10, 20),
(102, '2024-10-30', 'BJ', '2024-10-29', 20, 20),
(101, '2024-10-30', 'BJ', '2024-10-28', 5, 40),
(101, '2024-10-30', 'SH', '2024-10-29', 10, 20);

-- insert into 2 rows
INSERT INTO example_tbl_agg VALUES
(101, '2024-11-01', 'BJ', '2024-10-30', 20, 10),
(102, '2024-11-01', 'BJ', '2024-10-30', 10, 30);

-- check the rows of table
SELECT * FROM example_tbl_agg;
+---------+------------+------+---------------------+------+----------------+
| user_id | load_date  | city | last_visit_date     | cost | max_dwell_time |
+---------+------------+------+---------------------+------+----------------+
| 102     | 2024-10-30 | BJ   | 2024-10-29 00:00:00 |   20 |             20 |
| 102     | 2024-11-01 | BJ   | 2024-10-30 00:00:00 |   10 |             30 |
| 101     | 2024-10-30 | BJ   | 2024-10-28 00:00:00 |    5 |             40 |
| 101     | 2024-10-30 | SH   | 2024-10-29 00:00:00 |   10 |             20 |
| 101     | 2024-11-01 | BJ   | 2024-10-30 00:00:00 |   30 |             20 |
+---------+------------+------+---------------------+------+----------------+
```

AGG_STATE

::: Info Tips: AGG_STATE is an experimental feature and is recommended for use in development and testing environments.

AGG_STATE cannot be used as a Key column. The aggregation function's signature must be declared when creating the table. Users don't need to specify length or default values. The data storage size depends on the function implementation.

```
set enable_agg_state = true;
CREATE TABLE aggstate(
    k1   int  NULL,
    v1   int  SUM,
    v2   agg_state<group_concat(string)> generic
)
AGGREGATE KEY(k1)
DISTRIBUTED BY HASH(k1) BUCKETS 3;
```

In this case, `agg_state` is used to declare the data type as `agg_state`, and `sum/group_concat` is the signature of the aggregation function. Note that `agg_state` is a data type, just like `int`, `array`, or `string`. `agg_state` can only be used with combinators such as state, merge, or union. `agg_state` represents the intermediate result of an aggregation function. For example, for the aggregation function `group_concat`, `agg_state` can represent the intermediate state of `group_concat('a', 'b', 'c')`, rather than the final result.

The `agg_state` type needs to be generated using the `state` function. For this table, you need to use `group_concat_state`:

```
insert into aggstate values(1, 1, group_concat_state('a'));
insert into aggstate values(1, 2, group_concat_state('b'));
insert into aggstate values(1, 3, group_concat_state('c'));
insert into aggstate values(2, 4, group_concat_state('d'));
```

The calculation method in the table is shown in the diagram below:

K1	V1	V2
1	sum(1, 2, 3)	group_concat('a', 'b', 'c')
2	sum(4)	group_concat('d')

state-func-group-concat-state-result-1

When querying the table, the merge operation can be used to merge multiple `state` values and return the final aggregation result. Since `group_concat` requires ordering, the result may be unstable.

```
select group_concat_merge(v2) from aggstate;
+-----------------------+
| group_concat_merge(v2) |
+-----------------------+
| d,c,b,a               |
+-----------------------+
```

If you do not want the final aggregation result, you can use `union` to combine multiple intermediate aggregation results and generate a new intermediate result.

```
insert into aggstate select 3,sum_union(k2),group_concat_union(k3) from aggstate;
```

The calculations in the table are as follows:

K1	V1	V2
1	sum(1, 2, 3)	group_concat('a', 'b', 'c')
2	sum(4)	group_concat('d')
3	sum(1, 2, 3, 4)	group_concat('a', 'b', 'c', 'd')

state-func-group-concat-state-result-2

The query result is as follows:

```
mysql> select sum_merge(k2) , group_concat_merge(k3)from aggstate;
+---------------+-----------------------+
| sum_merge(k2) | group_concat_merge(k3) |
+---------------+-----------------------+
|            20 | c,b,a,d,c,b,a,d        |
+---------------+-----------------------+

mysql> select sum_merge(k2) , group_concat_merge(k3)from aggstate where k1 != 2;
+---------------+-----------------------+
| sum_merge(k2) | group_concat_merge(k3) |
+---------------+-----------------------+
|            16 | c,b,a,d,c,b,a          |
+---------------+-----------------------+
```

Usage Notes

Suggestions for column types

Suggestions for column types when creating a table:

1. The Key column should precede all Value columns.
2. Whenever possible, choose integer types. This is because the calculation and search efficiency of integer types are much higher than that of strings.
3. For selecting integer types of different lengths, follow the principle of sufficiency.
4. For the lengths of VARCHAR and STRING types, also follow the principle of sufficiency.

Limitations of aggregate model

This section is about the limitations of the Aggregate Model.

The Aggregate Model only presents the aggregated data. That means we have to ensure the presentation consistency of data that has not yet been aggregated (for example, two different import batches). The following provides further explanation with examples.

Suppose that you have the following table schema:

ColumnName	Type	AggregationType	Comment
user_id	LARGEINT		User ID
date	DATE		Date when the data are imported
cost	BIGINT	SUM	Total user consumption

Assume that there are two batches of data that have been imported into the storage engine as follows:

batch 1

user_id	date	cost
10001	2017-11-20	50
10002	2017-11-21	39

batch 2

user_id	date	cost
10001	2017-11-20	1
10001	2017-11-21	5
10003	2017-11-22	22

As you can see, data about User 10001 in these two import batches have not yet been aggregated. However, in order to ensure that users can only query the aggregated data as follows:

user_id	date	cost
10001	2017-11-20	51
10001	2017-11-21	5
10002	2017-11-21	39
10003	2017-11-22	22

We have added an aggregation operator to the query engine to ensure the presentation consistency of data.

In addition, on the aggregate column (Value), when executing aggregate class queries that are inconsistent with the aggregate type, please pay attention to the semantics. For example, in the example above, if you execute the following query:

```
SELECT MIN(cost) FROM table;
```

The result will be 5, not 1.

Meanwhile, this consistency guarantee could considerably reduce efficiency in some queries.

Take the basic count (*) query as an example:

```
SELECT COUNT(*) FROM table;
```

In other databases, such queries return results quickly. Because in actual implementation, the models can get the query result by counting rows and saving the statistics upon import, or by scanning only one certain column of data to get count value upon query, with very little overhead. But in Doris's Aggregation Model, the overhead of such queries is **large**.

For the previous example:

batch 1

user_id	date	cost
10001	2017-11-20	50
10002	2017-11-21	39

batch 2

user_id	date	cost
10001	2017-11-20	1
10001	2017-11-21	5
10003	2017-11-22	22

Since the final aggregation result is:

user_id	date	cost
10001	2017-11-20	51
10001	2017-11-21	5

user_id	date	cost
10002	2017-11-21	39
10003	2017-11-22	22

The correct result of `select count (*) from table;` should be **4**. But if the model only scans the `user_id` column and operates aggregation upon query, the final result will be **3** (10001, 10002, 10003). And if it does not operate aggregation, the final result will be **5** (a total of five rows in two batches). Apparently, both results are wrong.

In order to get the correct result, we must read both the `user_id` and `date` column, and **performs aggregation** when querying. That is to say, in the `count (*)` query, Doris must scan all AGGREGATE KEY columns (in this case, `user_id` and `date`) and aggregate them to get the semantically correct results. That means if there are many aggregated columns, `count (*)` queries could involve scanning large amounts of data.

Therefore, if you need to perform frequent `count (*)` queries, we recommend that you simulate `count (*)` by adding a column of value 1 and aggregation type SUM. In this way, the table schema in the previous example will be modified as follows:

ColumnName	Type	AggregationType	Comment
user ID	BIGINT		User ID
date	DATE		Date when the data are imported
Cost	BIGINT	SUM	Total user consumption
count	BIGINT	SUM	For count queries

The above adds a count column, the value of which will always be **1**, so the result of `select count (*) from table;` is equivalent to that of `select sum (count) from table;` The latter is much more efficient than the former. However, this method has its shortcomings, too. That is, it requires that users will not import rows with the same values in the AGGREGATE KEY columns. Otherwise, `select sum (count) from table;` can only express the number of rows of the originally imported data, instead of the semantics of `select count (*) from table;`

Another method is to add a `cound` column of value 1 but aggregation type of REPLACE. Then `select sum (count) from table;` and `select count (*) from table;` could produce the same results. Moreover, this method does not require the absence of same AGGREGATE KEY columns in the import data.

MoW Unique Key Model

The Merge on Write implementation in the Unique Model does not impose the same limitation as the Aggregate Model. In Merge on Write, the model adds a `delete bitmap` for each imported rowset to mark the data being overwritten or deleted. With the previous example, after Batch 1 is imported, the data status will be as follows:

batch 1

user_id	date	cost	delete bit
10001	2017-11-20	50	false
10002	2017-11-21	39	false

After Batch 2 is imported, the duplicate rows in the first batch will be marked as deleted, and the status of the two batches of data is as follows

batch 1

user_id	date	cost	delete bit
10001	2017-11-20	50	**true**
10002	2017-11-21	39	false

batch 2

user_id	date	cost	delete bit
10001	2017-11-20	1	false
10001	2017-11-21	5	false
10003	2017-11-22	22	false

In queries, all data marked `true` in the `delete bitmap` will not be read, so there is no need for data aggregation. Since there are 4 valid rows in the above data, the query result should also be 4. This also enables minimum overhead since it only scans one column of data.

In the test environment, `count(*)` queries in Merge on Write of the Unique Model deliver 10 times higher performance than that of the Aggregate Model.

Duplicate model

The Duplicate Model does not impose the same limitation as the Aggregate Model because it does not involve aggregation semantics. For any columns, it can return the semantically correct results in `count (*)` queries.

Key columns

For the Duplicate, Aggregate, and Unique Models, the Key columns will be specified when the table is created, but there exist some differences: In the Duplicate Model, the Key columns of the table can be regarded as just “sorting columns”, but not unique identifiers. In Aggregate and Unique Models, the Key columns are both “sorting columns” and “unique identifier columns”.

Suggestions for choosing data model

Since the data model was established when the table was built, and **irrevocable thereafter, it is very important to select the appropriate data model**.

1. The Aggregate Model can greatly reduce the amount of data scanned and query computation by pre-aggregation. Thus, it is very suitable for report query scenarios with fixed patterns. But this model is unfriendly to `count (*)` queries. Meanwhile, since the aggregation method on the Value column is fixed, semantic correctness should be considered in other types of aggregation queries.
2. The Unique Model guarantees the uniqueness of primary key for scenarios requiring a unique primary key. The downside is that it cannot exploit the advantage brought by pre-aggregation such as ROLLUP

in queries. Users who have high-performance requirements for aggregate queries are recommended to use the newly added Merge on Write implementation since version 1.2.

3. The Duplicate Model is suitable for ad-hoc queries of any dimensions. Although it may not be able to take advantage of the pre-aggregation feature, it is not limited by what constrains the Aggregate Model and can give full play to the advantage of columnar storage (reading only the relevant columns, but not all Key columns).
4. If user need to use partial-update, please refer to document partial-update

Hybrid Storage

Hybrid Storage

Doris defaults to columnar storage, where each column is stored contiguously. Columnar storage offers excellent performance for analytical scenarios (such as aggregation, filtering, sorting, etc.), as it only reads the necessary columns, reducing unnecessary IO. However, in point query scenarios (such as `SELECT *`), all columns need to be read, requiring an IO operation for each column, which can lead to IOPS becoming a bottleneck, especially for wide tables with many columns (e.g., hundreds of columns).

To address the IOPS bottleneck in point query scenarios, starting from version 2.0.0, Doris supports hybrid storage. When users create tables, they can specify whether to enable row storage. With row storage enabled, each row only requires one IO operation for point queries (such as `SELECT *`), significantly improving performance.

The principle of row storage is that an additional column is added during storage. This column concatenates all the columns of the corresponding row and stores them using a special binary format.

Syntax

When creating a table, specify whether to enable row storage and its page size in the table's `PROPERTIES`.

Property	Default	Description
`"store_row_column"`	`false`	Enable row storage for all columns.
`"row_store_page_size"`	`16384` (16 KB)	Size of the row-store page in bytes. A smaller value increases storage overhead but yields better point-query performance.

Example

The example below creates an 8-column table with row storage enabled.

```
CREATE TABLE `tbl_point_query` (
     `key` int(11) NULL,
     `v1` decimal(27, 9) NULL,
     `v2` varchar(30) NULL,
     `v3` varchar(30) NULL,
     `v4` date NULL,
     `v5` datetime NULL,
     `v6` float NULL,
     `v7` datev2 NULL
) ENGINE=OLAP
UNIQUE KEY(`key`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`key`) BUCKETS 1
PROPERTIES (
     "enable_unique_key_merge_on_write" = "true",
     "light_schema_change" = "true",
```

```
    "store_row_column" = "true"
);
```

Query:

```
SELECT * FROM tbl_point_query WHERE key = 100;
```

For more information on point query usage, please refer to High-Concurrent Point Query.

Notice

1. Enabling row storage will increase the storage space used. The increase in storage space is related to the data characteristics and is generally 2 to 10 times the size of the original table. The exact space usage needs to be tested with actual data.
2. Modifying the `store_row_column` property via `ALTER TABLE` is not supported.

Basic concepts

This document mainly introduces table creation and data partitioning in Doris, as well as potential problems and solutions encountered during table creation operations.

Row & Column

In Doris, data is logically described in the form of tables.

A table consists of rows and columns:

- Row: Represents a single line of user data;
- Column: Used to describe different fields in a row of data;
- Columns can be divided into two types: Key and Value. From a business perspective, Key and Value can correspond to dimension columns and metric columns, respectively. The key columns in Doris are those specified in the table creation statement, which are the columns following the keywords `unique key`, `aggregate key`, or `duplicate key`. The remaining columns are value columns. From the perspective of the aggregation model, rows with the same Key columns will be aggregated into a single row. The aggregation method for value columns is specified by the user during table creation. For more information on aggregation models, refer to the Doris Data Model.

Partition & Bucket

Doris uses a two-level partitioning and bucketing method to organize and manage data.

Partition

Partition refers to dividing the table into smaller, more manageable, non-overlapping subsets based on specific column values in the table. Each subset of data is called a partition. Each row of data belongs to exactly one specific partition. Partitions can be seen as the smallest logical management unit.

Currently, Doris supports two types of partitioning: Range and List. If no partition is specified during table creation, Doris will generate a default partition containing all the data in the table, which is transparent to the user.

Partitioning based on data distribution and query patterns offers several benefits:

- **Improved Query Performance**: Partitioning allows the system to prune irrelevant partitions based on the query conditions, reducing the amount of data scanned and significantly improving query efficiency. This is especially beneficial when handling large datasets, as the partition strategy can greatly reduce I/O overhead.

- **Flexible Management**: Partitioning allows data to be split based on logic such as time or geography, facilitating data archiving, cleaning, and backup. For example, partitioning by time can effectively manage historical and newly added data, supporting efficient time-based data maintenance strategies.

Bucket

Bucketing refers to further dividing the data within a partition into smaller, non-overlapping units according to some rule. Each row of data belongs to exactly one specific bucket. Unlike partitioning, which divides data based on specific column values, bucketing attempts to evenly distribute the data across predefined buckets, thereby reducing data skew. Bucketing improves query performance by ensuring even data distribution and enhancing data locality.

Currently, Doris supports two types of bucketing: Hash and Random.

A bucket corresponds to a data shard (Tablet) at the physical level, and data shards are physically stored independently. They are the smallest physical storage units for operations like data movement and replication.

Proper bucketing offers several advantages:

- **Even Data Distribution**: Bucketing evenly distributes data across buckets, reducing the risk of data concentration or skew, and preventing resource overload on specific nodes or storage devices.
- **Reduced Hotspots**: By distributing data evenly, bucketing helps to reduce the risk of overloading specific nodes or partitions, preventing hotspots, and improving system stability and processing capability.
- **Improved Concurrency Performance**: Bucketing enhances the performance of concurrent queries, especially when multiple query requests need to access different data within the same partition. The granularity of bucketing allows the system to efficiently process multiple requests in parallel, thereby improving throughput.

Example of creating a table

CREATE TABLE in Doris is a synchronous command. It returns results after the SQL execution is completed. Successful returns indicate successful table creation. For more information, please refer to CREATE TABLE, or input the `HELP CREATE TABLE;` command.

This section introduces how to create tables in Doris by range partiton and hash buckets.

```
-- Range Partition
CREATE TABLE IF NOT EXISTS example_range_tbl
(
     `user_id` LARGEINT NOT NULL COMMENT "User ID",
    `date` DATE NOT NULL COMMENT "Date when the data are imported",
    `timestamp` DATETIME NOT NULL COMMENT "Timestamp when the data are imported",
    `city` VARCHAR(20) COMMENT "User location city",
    `age` SMALLINT COMMENT "User age",
    `sex` TINYINT COMMENT "User gender",
    `last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT "User last visit time",
    `cost` BIGINT SUM DEFAULT "0" COMMENT "Total user consumption",
    `max_dwell_time` INT MAX DEFAULT "0" COMMENT "Maximum user dwell time",
    `min_dwell_time` INT MIN DEFAULT "99999" COMMENT "Minimum user dwell time"
)
ENGINE=OLAP
AGGREGATE KEY(`user_id`, `date`, `timestamp`, `city`, `age`, `sex`)
PARTITION BY RANGE(`date`)
(
    PARTITION `p201701` VALUES [("2017-01-01"),  ("2017-02-01")),
    PARTITION `p201702` VALUES [("2017-02-01"), ("2017-03-01")),
```

```
    PARTITION `p201703` VALUES [("2017-03-01"), ("2017-04-01"))
)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 16
PROPERTIES
(
    "replication_num" = "1"
);
```

Here use the AGGREGATE KEY data model as an example. In the AGGREGATE KEY data model, all columns that are specified with an aggregation type (SUM, REPLACE, MAX, or MIN) are Value columns. The rest are the Key columns.

In the PROPERTIES at the end of the CREATE TABLE statement, you can find detailed information about the relevant parameters that can be set in PROPERTIES by referring to the documentation on CREATE TABLE.

The default type of ENGINE is OLAP. In Doris, only this OLAP ENGINE type is responsible for data management and storage by Doris itself. Other ENGINE types, such as mysql, broker, es, etc., are essentially just mappings to tables in other external databases or systems, allowing Doris to read this data. However, Doris itself does not create, manage, or store any tables or data for non-OLAP ENGINE types.

`IF NOT EXISTS` indicates that if the table has not been created before, it will be created. Note that this only checks if the table name exists and does not check if the schema of the new table is the same as the schema of an existing table. Therefore, if there is a table with the same name but a different schema, this command will also return successfully, but it does not mean that a new table and a new schema have been created.

View partitions

You can use the `show create table` command to view the partition information of a table.

```
> show create table  example_range_tbl
+-------------------+-----------------------------------------------------------------------------------------------------------+
| Table             | Create Table                                                                                              |
+-------------------+-----------------------------------------------------------------------------------------------------------+
| example_range_tbl | CREATE TABLE `example_range_tbl` (                                                                        |
|                   |   `user_id` largeint(40) NOT NULL COMMENT 'User ID',                                                       |
|                   |   `date` date NOT NULL COMMENT 'Date when the data are imported',
|
|                   |   `timestamp` datetime NOT NULL COMMENT 'Timestamp when the data are imported',
|
|                   |   `city` varchar(20) NULL COMMENT 'User location city',
|
|                   |   `age` smallint(6) NULL COMMENT 'User age',                                                              |
|                   |   `sex` tinyint(4) NULL COMMENT 'User gender',                                                              |
|                   |   `last_visit_date` datetime REPLACE NULL DEFAULT "1970-01-01 00:00:00" COMMENT 'User last visit time', |
|                   |   `cost` bigint(20) SUM NULL DEFAULT "0" COMMENT 'Total user consumption',
|
|                   |   `max_dwell_time` int(11) MAX NULL DEFAULT "0" COMMENT 'Maximum user dwell time',
|
|                   |   `min_dwell_time` int(11) MIN NULL DEFAULT "99999" COMMENT 'Minimum user dwell time'
|
|                   | ) ENGINE=OLAP                                                                                             |
|                   | AGGREGATE KEY(`user_id`, `date`, `timestamp`, `city`, `age`, `sex`)                                       |
|                   | COMMENT 'OLAP'                                                                                            |
|                   | PARTITION BY RANGE(`date`)                                                                                |
|                   | (PARTITION p201701 VALUES [('0000-01-01'), ('2017-02-01')),                                               |
|                   | PARTITION p201702 VALUES [('2017-02-01'), ('2017-03-01')),                                                |
|                   | PARTITION p201703 VALUES [('2017-03-01'), ('2017-04-01')))                                                |
|                   | DISTRIBUTED BY HASH(`user_id`) BUCKETS 16                                                                 |
```

```
|                  | PROPERTIES (                                                
```

Manual partitioning

Partition columns

- Partition columns can be specified as one or multiple columns, and the partition columns must be KEY columns. The usage of multi-column partitioning will be introduced later in the summary section of multi-column partitioning.
- When `allow_partition_column_nullable` is set to true, Range partition supports the use of NULL partition columns. List Partition does not support NULL partition columns at all times.
- Regardless of the type of partition column, double quotes are required when writing partition values.
- There is theoretically no upper limit on the number of partitions. However, each table is limited to 4096 partitions by default. If you want to exceed this limit, you can modify the FE configuration parameters `max_multi_partition_num` and `max_dynamic_partition_num`.
- When creating a table without partitioning, the system will automatically generate a full-range partition with the same name as the table name. This partition is not visible to users and cannot be deleted or modified.
- Overlapping ranges are not allowed when creating partitions.

Range partition

Partition columns are usually time columns for convenient management of old and new data. Range partition supports column types such as DATE, DATETIME, TINYINT, SMALLINT, INT, BIGINT, and LARGEINT.

Partition information supports four writing methods:

- FIXED RANGE: the partition as a left-closed, right-open interval.

```
PARTITION BY RANGE(col1[, col2, ...])
(
    PARTITION partition_name1 VALUES [("k1-lower1", "k2-lower1", "k3-lower1",...), ("k1-upper1", "k2-upper1", "k3-upper1", ...)),
    PARTITION partition_name2 VALUES [("k1-lower1-2", "k2-lower1-2", ...), ("k1-upper1-2", MAXVALUE, ))
)
```

For example:

```
PARTITION BY RANGE(`date`)
(
    PARTITION `p201701` VALUES [("2017-01-01"),  ("2017-02-01")),
    PARTITION `p201702` VALUES [("2017-02-01"), ("2017-03-01")),
    PARTITION `p201703` VALUES [("2017-03-01"), ("2017-04-01"))
)
```

- LESS THAN: Only define the upper bound of the partition. The lower bound is determined by the upper bound of the previous partition.

```
PARTITION BY RANGE(col1[, col2, ...])
(
    PARTITION partition_name1 VALUES LESS THAN MAXVALUE | ("value1", "value2", ...),
```

```
    PARTITION partition_name2 VALUES LESS THAN MAXVALUE | ("value1", "value2", ...)
)
```

For example:

```
PARTITION BY RANGE(`date`)
(
    PARTITION `p201701` VALUES LESS THAN ("2017-02-01"),
    PARTITION `p201702` VALUES LESS THAN ("2017-03-01"),
    PARTITION `p201703` VALUES LESS THAN ("2017-04-01")
)

PARTITION BY RANGE(`date`)
(
    PARTITION `p201701` VALUES LESS THAN ("2017-02-01"),
    PARTITION `p201702` VALUES LESS THAN ("2017-03-01"),
    PARTITION `p201703` VALUES LESS THAN ("2017-04-01")
    PARTITION `other` VALUES LESS THAN (MAXVALUE)
)
```

List partition

Partition columns support data types such as BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, LARGEINT, DATE, DATETIME, CHAR, and VARCHAR. Partition values are enumerated values. Only when the data is one of the enumerated values of the target partition, the partition can be hit .

Partitions support specifying the enumerated values contained in each partition through VALUES IN (…).

For example:

```
PARTITION BY LIST(city)
(
    PARTITION `p_cn` VALUES IN ("Beijing", "Shanghai", "Hong Kong"),
    PARTITION `p_usa` VALUES IN ("New York", "San Francisco"),
    PARTITION `p_jp` VALUES IN ("Tokyo")
)
```

List partition also supports multi-column partitioning, for example:

```
PARTITION BY LIST(id, city)
(
    PARTITION p1_city VALUES IN (("1", "Beijing"), ("1", "Shanghai")),
    PARTITION p2_city VALUES IN (("2", "Beijing"), ("2", "Shanghai")),
    PARTITION p3_city VALUES IN (("3", "Beijing"), ("3", "Shanghai"))
)
```

Dynamic partition

Dynamic partition is designed to manage partition's Time-to-Life (TTL), reducing the burden on users.

In some usage scenarios, the user will partition the table according to the day and perform routine tasks regularly every day. At this time, the user needs to manually manage the partition. Otherwise, the data load may fail because the user does not create a partition. This brings additional maintenance costs to the user.

With dynamic partitioning, users can define rules for partition creation and deletion when establishing tables. The FE initiates a background thread to handle partition creation or deletion based on these user-defined rules. Users also have the flexibility to modify these rules during runtime.

It's important to note that dynamic partitioning is exclusively supported by range partitions. Currently, the functionality enables dynamic addition and deletion of partitions.

> **This feature will be disabled when synchronized by CCR. If this table is copied by CCR, that is, PROPERTIES contains `is_being_synced = true`, it will be displayed as enabled in show create table, but will not actually take effect. When `is_being_synced` is set to `false`, these features will resume working, but the `is_being_synced` property is for CCR peripheral modules only and should not be manually set during CCR synchronization.**

How to use

The rules for dynamic partitioning can be specified when the table is created or modified at runtime.

Currently, dynamic partition rules can only be set for partition tables with single partition columns.

- Specified when creating table

```
CREATE TABLE tbl1
(...)
PROPERTIES
(
    "dynamic_partition.prop1" = "value1",
    "dynamic_partition.prop2" = "value2",
    ...
)
```

- Modify at runtime

```
ALTER TABLE tbl1 SET
(
    "dynamic_partition.prop1" = "value1",
    "dynamic_partition.prop2" = "value2",
    ...
)
```

Rule parameters

The rules of dynamic partition are prefixed with `dynamic_partition.`:

- `dynamic_partition.enable`

  Whether to enable the dynamic partition feature. Can be specified as `TRUE` or`FALSE`. If not filled, the default is `TRUE`. If it is `FALSE`, Doris will ignore the dynamic partitioning rules of the table.

- `dynamic_partition.time_unit`(required parameters)

  The unit for dynamic partition scheduling. Can be specified as `HOUR`,`DAY`,`WEEK`, `MONTH` and `YEAR`, means to create or delete partitions by hour, day, week, month and year, respectively.

  When specified as `HOUR`, the suffix format of the dynamically created partition name is `yyyyMMddHH`, for example, `2020032501`. *When the time unit is HOUR, the data type of partition column cannot be DATE.*

  When specified as `DAY`, the suffix format of the dynamically created partition name is `yyyyMMdd`, for example, `20200325`.

  When specified as `WEEK`, the suffix format of the dynamically created partition name is `yyyy_ww`. That is, the week of the year of current date. For example, the suffix of the partition created for `2020-03-25` is `2020_13`, indicating that it is currently the 13th week of 2020.

  When specified as `MONTH`, the suffix format of the dynamically created partition name is `yyyyMM`, for example, `202003`.

  When specified as `YEAR`, the suffix format of the dynamically created partition name is `yyyy`, for example, `2020`.

- `dynamic_partition.time_zone`

  The time zone of the dynamic partition, if not filled in, defaults to the time zone of the current machine's system, such as `Asia/Shanghai`, if you want to know the supported TimeZone, you can found in Timezone.

- `dynamic_partition.start`

  The starting offset of the dynamic partition, usually a negative number. Depending on the `time_unit` attribute, based on the current day (week / month), the partitions with a partition range before this offset will be deleted. If not filled, the default is `-2147483648`, that is, the history partition will not be deleted.

- `dynamic_partition.end`(required parameters)

  The end offset of the dynamic partition, usually a positive number. According to the difference of the `time_unit` attribute, the partition of the corresponding range is created in advance based on the current day (week / month).

- `dynamic_partition.prefix`(required parameters)

  The dynamically created partition name prefix.

- `dynamic_partition.buckets`

The number of buckets corresponding to the dynamically created partitions.

- `dynamic_partition.replication_num`

  The replication number of dynamic partition.If not filled in, defaults to the number of table's replication number.

- `dynamic_partition.start_day_of_week`

  When `time_unit` is`WEEK`, this parameter is used to specify the starting point of the week. The value ranges from 1 to 7. Where 1 is Monday and 7 is Sunday. The default is 1, which means that every week starts on Monday.

- `dynamic_partition.start_day_of_month`

  When `time_unit` is`MONTH`, this parameter is used to specify the start date of each month. The value ranges from 1 to 28. 1 means the 1st of every month, and 28 means the 28th of every month. The default is 1, which means that every month starts at 1st. The 29, 30 and 31 are not supported at the moment to avoid ambiguity caused by lunar years or months.

- `dynamic_partition.create_history_partition`

  The default is false. When set to true, Doris will automatically create all partitions, as described in the creation rules below. At the same time, the parameter `max_dynamic_partition_num` of FE will limit the total number of partitions to avoid creating too many partitions at once. When the number of partitions expected to be created is greater than `max_dynamic_partition_num`, the operation will fail.

  When the `start` attribute is not specified, this parameter has no effect.

- `dynamic_partition.history_partition_num`

  When `create_history_partition` is `true`, this parameter is used to specify the number of history partitions. The default value is -1, which means it is not set.

- `dynamic_partition.hot_partition_num`

  Specify how many of the latest partitions are hot partitions. For hot partition, the system will automatically set its `storage_medium` parameter to SSD, and set `storage_cooldown_time`.

  If there is no SSD disk path under the storage path, configuring this parameter will cause dynamic partition creation to fail.

  `hot_partition_num` is all partitions in the previous n days and in the future.

  Let us give an example. Suppose today is 2021-05-20, partition by day, and the properties of dynamic partition are set to: hot_partition_num=2, end=3, start=-3. Then the system will automatically create the following partitions, and set the `storage_medium` and `storage_cooldown_time` properties:

```
p20210517: ["2021-05-17", "2021-05-18") storage_medium=HDD storage_cooldown_time=9999-12-31 23:59:59
p20210518: ["2021-05-18", "2021-05-19") storage_medium=HDD storage_cooldown_time=9999-12-31 23:59:59
p20210519: ["2021-05-19", "2021-05-20") storage_medium=SSD storage_cooldown_time=2021-05-21 00:00:00
p20210520: ["2021-05-20", "2021-05-21") storage_medium=SSD storage_cooldown_time=2021-05-22 00:00:00
p20210521: ["2021-05-21", "2021-05-22") storage_medium=SSD storage_cooldown_time=2021-05-23 00:00:00
p20210522: ["2021-05-22", "2021-05-23") storage_medium=SSD storage_cooldown_time=2021-05-24 00:00:00
p20210523: ["2021-05-23", "2021-05-24") storage_medium=SSD storage_cooldown_time=2021-05-25 00:00:00
```

- dynamic_partition.reserved_history_periods

  The range of reserved history periods. It should be in the form of [yyyy-MM-dd,yyyy-MM-dd],[...,...] while the dynamic_partition.time_unit is “DAY, WEEK, MONTH and YEAR”. And it should be in the form of [yyyy-MM-dd HH:mm:ss,yyyy-MM-dd HH:mm:ss],[...,...] while the dynamic_partition.time_unitis "HOUR". And no more spaces expected. The default value is“NULL”`, which means it is not set.

  Let us give an example. Suppose today is 2021-09-06, partitioned by day, and the properties of dynamic partition are set to:

  time_unit="DAY/WEEK/MONTH/YEAR", end=3, start=-3, reserved_history_periods="[2020-06-01,2020-06-20], [2020-10-31,2020-11-15]".

  The system will automatically reserve following partitions in following period :

  ```
  ["2020-06-01","2020-06-20"],
  ["2020-10-31","2020-11-15"]
  ```

  or

  time_unit="HOUR", end=3, start=-3, reserved_history_periods="[2020-06-01 00:00:00,2020-06-01 03:00:00]".

  The system will automatically reserve following partitions in following period :

  ```
  ["2020-06-01 00:00:00","2020-06-01 03:00:00"]
  ```

  Otherwise, every [...,...] in reserved_history_periods is a couple of properties, and they should be set at the same time. And the first date can’t be larger than the second one.

- dynamic_partition.storage_medium

:::tip Tips This configuration is supported since the Apache Doris 1.2.3 version :::

Specifies the default storage medium for the created dynamic partition. HDD is the default, SSD can be selected.

Note that when set to SSD, the hot_partition_num property will no longer take effect, all partitions will default to SSD storage media and the cooldown time will be 9999-12-31 23:59:59.

Create history partition rules

When create_history_partition is true, i.e. history partition creation is enabled, Doris determines the number of history partitions to be created based on dynamic_partition.start and dynamic_partition.history_partition_num.

Assuming the number of history partitions to be created is expect_create_partition_num, the number is as follows according to different settings.

- create_history_partition = true
  - dynamic_partition.history_partition_num is not set, i.e. -1.
    expect_create_partition_num = end - start;

    - dynamic_partition.history_partition_num is set expect_create_partition_num = end - max(start, -history_partition_num);
- create_history_partition = false

No history partition will be created, expect_create_partition_num = end - 0;

When expect_create_partition_num is greater than max_dynamic_partition_num (default 500), creating too many partitions is prohibited.

Examples:

Suppose today is 2021-05-20, partition by day, and the attributes of dynamic partition are set to create_history_partition=true, end=3, start=-3, history_partition_num=1, then the system will automatically create the following partitions.

```
p20210519
p20210520
p20210521
p20210522
p20210523
```

history_partition_num=5 and keep the rest attributes as in 1, then the system will automatically create the following partitions.

```
p20210517
p20210518
p20210519
p20210520
p20210521
p20210522
p20210523
```

history_partition_num=-1 i.e., if you do not set the number of history partitions and keep the rest of the attributes as in 1, the system will automatically create the following partitions.

```
p20210517
p20210518
p20210519
p20210520
p20210521
p20210522
p20210523
```

Example

1. Table tbl1 partition column k1, type is DATE, create a dynamic partition rule. By day partition, only the partitions of the last 7 days are kept, and the partitions of the next 3 days are created in advance.

    ```
    CREATE TABLE tbl1
    (
        k1 DATE,
        ...
    )
    ```

```
PARTITION BY RANGE(k1) ()
DISTRIBUTED BY HASH(k1)
PROPERTIES
(
    "dynamic_partition.enable" = "true",
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.start" = "-7",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "32"
);
```

Suppose the current date is 2020-05-29. According to the above rules, tbl1 will produce the following partitions:

```
p20200529: ["2020-05-29", "2020-05-30")
p20200530: ["2020-05-30", "2020-05-31")
p20200531: ["2020-05-31", "2020-06-01")
p20200601: ["2020-06-01", "2020-06-02")
```

On the next day, 2020-05-30, a new partition will be created `p20200602: [" 2020-06-02 "," 2020-06-03 ")`

On 2020-06-06, because `dynamic_partition.start` is set to 7, the partition 7 days ago will be deleted, that is, the partition `p20200529` will be deleted.

2. Table tbl1 partition column k1, type is DATETIME, create a dynamic partition rule. Partition by week, only keep the partition of the last 2 weeks, and create the partition of the next 2 weeks in advance.

```
CREATE TABLE tbl1
(
    k1 DATETIME,
    ...
)
PARTITION BY RANGE(k1) ()
DISTRIBUTED BY HASH(k1)
PROPERTIES
(
    "dynamic_partition.enable" = "true",
    "dynamic_partition.time_unit" = "WEEK",
    "dynamic_partition.start" = "-2",
    "dynamic_partition.end" = "2",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "8"
);
```

Suppose the current date is 2020-05-29, which is the 22nd week of 2020. The default week starts on Monday. Based on the above rules, tbl1 will produce the following partitions:

```
p2020_22: ["2020-05-25 00:00:00", "2020-06-01 00:00:00")
p2020_23: ["2020-06-01 00:00:00", "2020-06-08 00:00:00")
p2020_24: ["2020-06-08 00:00:00", "2020-06-15 00:00:00")
```

The start date of each partition is Monday of the week. At the same time, because the type of the partition column k1 is DATETIME, the partition value will fill the hour, minute and second fields, and all are 0.

On 2020-06-15, the 25th week, the partition 2 weeks ago will be deleted, ie `p2020_22` will be deleted.

In the above example, suppose the user specified the start day of the week as `"dynamic_partition.start_day_of_week" = "3"`, that is, set Wednesday as the start of week. The partition is as follows:

```
p2020_22: ["2020-05-27 00:00:00", "2020-06-03 00:00:00")
p2020_23: ["2020-06-03 00:00:00", "2020-06-10 00:00:00")
p2020_24: ["2020-06-10 00:00:00", "2020-06-17 00:00:00")
```

That is, the partition ranges from Wednesday of the current week to Tuesday of the next week.

2019-12-31 and 2020-01-01 are in same week, if the starting date of the partition is 2019-12-31, the partition name is `p2019_53`, if the starting date of the partition is 2020-01 -01, the partition name is `p2020_01`.

3. Table tbl1 partition column k1, type is DATE, create a dynamic partition rule. Partition by month without deleting historical partitions, and create partitions for the next 2 months in advance. At the same time, set the starting date on the 3rd of each month.

```
CREATE TABLE tbl1
(
    k1 DATE,
    ...
)
PARTITION BY RANGE(k1) ()
DISTRIBUTED BY HASH(k1)
PROPERTIES
(
    "dynamic_partition.enable" = "true",
    "dynamic_partition.time_unit" = "MONTH",
    "dynamic_partition.end" = "2",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "8",
    "dynamic_partition.start_day_of_month" = "3"
);
```

Suppose the current date is 2020-05-29. Based on the above rules, tbl1 will produce the following partitions:

```
p202005: ["2020-05-03", "2020-06-03")
p202006: ["2020-06-03", "2020-07-03")
p202007: ["2020-07-03", "2020-08-03")
```

Because `dynamic_partition.start` is not set, the historical partition will not be deleted.

Assuming that today is 2020-05-20, and set 28th as the start of each month, the partition range is:

```
p202004: ["2020-04-28", "2020-05-28")
p202005: ["2020-05-28", "2020-06-28")
p202006: ["2020-06-28", "2020-07-28")
```

Modify dynamic partition properties

You can modify the properties of the dynamic partition with the following command

```
ALTER TABLE tbl1 SET
(
    "dynamic_partition.prop1" = "value1",
    ...
);
```

The modification of certain attributes may cause conflicts. Assume that the partition granularity was DAY and the following partitions have been created:

```
p20200519: ["2020-05-19", "2020-05-20")
p20200520: ["2020-05-20", "2020-05-21")
p20200521: ["2020-05-21", "2020-05-22")
```

If the partition granularity is changed to MONTH at this time, the system will try to create a partition with the range `["2020-05-01", "2020-06-01")`, and this range conflicts with the existing partition. So it cannot be created. And the partition with the range `["2020-06-01", "2020-07-01")` can be created normally. Therefore, the partition between 2020-05-22 and 2020-05-30 needs to be filled manually.

Check dynamic partition table scheduling status

You can further view the scheduling of dynamic partitioned tables by using the following command:

```
mysql> SHOW DYNAMIC PARTITION TABLES;
+-----------+--------+----------+-------------+------+--------+---------+-----------+----------------+---------------------+--------
+-----------------------+----------------------+-------------------------+
| TableName | Enable | TimeUnit | Start       | End  | Prefix | Buckets | StartOf   | LastUpdateTime | LastSchedulerTime   | State
| LastCreatePartitionMsg | LastDropPartitionMsg | ReservedHistoryPeriods  |
+-----------+--------+----------+-------------+------+--------+---------+-----------+----------------+---------------------+--------
+-----------------------+----------------------+-------------------------+
| d3        | true   | WEEK     | -3          | 3    | p      | 1       | MONDAY    | N/A            | 2020-05-25 14:29:24 | NORMAL
| N/A                   | N/A                  | [2021-12-01,2021-12-31] |
| d5        | true   | DAY      | -7          | 3    | p      | 32      | N/A       | N/A            | 2020-05-25 14:29:24 | NORMAL
| N/A                   | N/A                  | NULL                    |
| d4        | true   | WEEK     | -3          | 3    | p      | 1       | WEDNESDAY | N/A            | 2020-05-25 14:29:24 | NORMAL
| N/A                   | N/A                  | NULL                    |
| d6        | true   | MONTH    | -2147483648 | 2    | p      | 8       | 3rd       | N/A            | 2020-05-25 14:29:24 | NORMAL
| N/A                   | N/A                  | NULL                    |
| d2        | true   | DAY      | -3          | 3    | p      | 32      | N/A       | N/A            | 2020-05-25 14:29:24 | NORMAL
| N/A                   | N/A                  | NULL                    |
| d7        | true   | MONTH    | -2147483648 | 5    | p      | 8       | 24th      | N/A            | 2020-05-25 14:29:24 | NORMAL
| N/A                   | N/A                  | NULL                    |
+-----------+--------+----------+-------------+------+--------+---------+-----------+----------------+---------------------+--------
+-----------------------+----------------------+-------------------------+
7 rows in set (0.02 sec)
```

- LastUpdateTime: The last time of modifying dynamic partition properties
- LastSchedulerTime: The last time of performing dynamic partition scheduling
- State: The state of the last execution of dynamic partition scheduling
- LastCreatePartitionMsg: Error message of the last time to dynamically add partition scheduling
- LastDropPartitionMsg: Error message of the last execution of dynamic deletion partition scheduling

Advanced operation

FE Configuration Item

- dynamic_partition_enable

  Whether to enable Doris's dynamic partition feature. The default value is false, which is off. This parameter only affects the partitioning operation of dynamic partition tables, not normal tables. You can modify the parameters in `fe.conf` and restart FE to take effect. You can also execute the following commands at runtime to take effect:

  ```
  MySQL protocol:

  `ADMIN SET FRONTEND CONFIG ("dynamic_partition_enable" = "true")`

  HTTP protocol:

  `curl --location-trusted -u username:password -XGET http://fe_host:fe_http_port/api/_set_config?
  dynamic_partition_enable=true`
  ```

  To turn off dynamic partitioning globally, set this parameter to false.

- dynamic_partition_check_interval_seconds

  The execution frequency of dynamic partition threads defaults to 3600 (1 hour), that is, scheduling is performed every 1 hour. You can modify the parameters in `fe.conf` and restart FE to take effect. You can also modify the following commands at runtime:

  ```
  MySQL protocol:

  `ADMIN SET FRONTEND CONFIG ("dynamic_partition_check_interval_seconds" = "7200")`

  HTTP protocol:

  `curl --location-trusted -u username:password -XGET http://fe_host:fe_http_port/api/_set_config?
  dynamic_partition_check_interval_seconds=432000`
  ```

Converting dynamic and manual partition tables to each other

For a table, dynamic and manual partitioning can be freely converted, but they cannot exist at the same time, there is and only one state.

Converting Manual Partitioning to Dynamic Partitioning

If a table is not dynamically partitioned when it is created, it can be converted to dynamic partitioning at runtime by modifying the dynamic partitioning properties with `ALTER TABLE`, an example of which can be seen with `HELP ALTER TABLE`.

When dynamic partitioning feature is enabled, Doris will no longer allow users to manage partitions manually, but will automatically manage partitions based on dynamic partition properties.

> **If** `dynamic_partition.start` **is set, historical partitions with a partition range before the start offset of**

> **the dynamic partition will be deleted.**

Converting Dynamic Partitioning to Manual Partitioning

The dynamic partitioning feature can be disabled by executing `ALTER TABLE tbl_name SET ("dynamic_partition.enable" = "false")` and converting it to a manual partition table.

When dynamic partitioning feature is disabled, Doris will no longer manage partitions automatically, and users will have to create or delete partitions manually by using `ALTER TABLE`.

Manual bucketing

If partitions are used, `DISTRIBUTED ...` statement describes the rules for dividing data within each partition.

If partitions are not used, it describes the rules for dividing the data across the entire table.

It is also possible to specify a bucketing method for each partition individually.

The bucket columns can be multiple columns. For the Aggregate and Unique models, they must be Key columns, while for the duplicate key data model, they can be both key and value columns. Bucket columns can be the same as or different from Partition columns.

The choice of bucket columns involves a trade-off between query throughput and query concurrency:

- If multiple bucket columns are selected, the data distribution will be more uniform. If a query condition does not include equal conditions for all bucket columns, the query will trigger simultaneous scanning of all buckets, increasing query throughput and reducing the latency of individual queries. This approach is suitable for high-throughput, low-concurrency query scenarios.
- If only one or a few bucket columns are selected, a point query can trigger scanning of just one bucket. In this case, when multiple point queries are concurrent, there is a higher probability that they will trigger scanning of different buckets, reducing the IO impact between queries (especially when different buckets are distributed across different disks). Therefore, this approach is suitable for high-concurrency point query scenarios.

Recommendations for bucket number and data volume:

- The total number of tablets for a table is equal to (Partition num * Bucket num).
- Without considering expansion, it is recommended that the number of tablets for a table be slightly more than the total number of disks in the cluster.
- In theory, there is no upper or lower limit for the data volume of a single tablet, but it is recommended to be within the range of 1G - 10G. If the data volume of a single tablet is too small, the data aggregation effect will not be good, and the metadata management pressure will be high. If the data volume is too large, it will not be conducive to the migration and replenishment of replicas, and it will increase the cost of retrying failed operations such as Schema Change or Rollup (the granularity of retrying these operations is the tablet).
- When there is a conflict between the data volume principle and the quantity principle of tablets, it is recommended to prioritize the data volume principle.
- When creating a table, the bucket number for each partition is uniformly specified. However, when dynamically adding partitions `ADD PARTITION`, the bucket number for the new partition can be specified separately. This feature can be conveniently used to handle data reduction or expansion.
- Once the bucket number for a partition is specified, it cannot be changed. Therefore, when determining the bucket number, it is necessary to consider the cluster expansion scenario in advance. For example, if there are only 3 hosts with 1 disk each, and the bucket number is set to 3 or less, then even if more machines are added later, the concurrency cannot be improved.

Here are some examples: Assuming there are 10 BEs, each with one disk. If a table has a total size of 500MB, 4-8 tablets can be considered. For 5GB: 8-16 tablets. For 50GB: 32 tablets. For 500GB: It is recommended to

partition the table, with each partition size around 50GB and 16-32 tablets per partition. For 5TB: It is recommended to partition the table, with each partition size around 50GB and 16-32 tablets per partition.

The data volume of a table can be viewed using the SHOW DATA command, and the result should be divided by the number of replicas to obtain the actual data volume of the table.

Random distribution

- If an OLAP table does not have fields of the update type, setting the data bucketing mode of the table to RANDOM can avoid severe data skew. When data is imported into the corresponding partitions of the table, each batch of a single import job will randomly select a tablet for writing.
- When the bucketing mode of a table is set to RANDOM, there is no bucketing column, it is not possible to query only a few buckets based on the values of the bucketing column. Queries on the table will simultaneously scan all buckets that hit the partition. This setting is suitable for aggregate query analysis of the entire table data, but not suitable for high-concurrency point queries.
- If the data distribution of the OLAP table is Random Distribution, then during data import, single-tablet import mode can be set (set `load_to_single_tablet` to true). Then, during large-volume data import, a task will only write to one tablet when writing data to the corresponding partition. This can improve the concurrency and throughput of data import, reduce the write amplification caused by data import and compaction, and ensure the stability of the cluster.

Auto bucket

Users often encounter various issues due to improper bucket settings. To address this, we provide an automated approach for setting the number of buckets, which is currently applicable only to OLAP tables.

> **This feature will be disabled when synchronized by CCR. If this table is copied by CCR, that is, PROPERTIES contains `is_being_synced = true`, it will be displayed as enabled in show create table, but will not actually take effect. When `is_being_synced` is set to `false`, these features will resume working, but the `is_being_synced` property is for CCR peripheral modules only and should not be manually set during CCR synchronization.**

In the past, user had to set the number of buckets manually when creating table, but the automatic bucket feature is a way for Apache Doris to dynamically project the number of buckets, so that the number of buckets always stays within a suitable range and users don't have to worry about the minutiae of the number of buckets.

For the sake of clarity, this section splits the bucket into two periods, the initial bucket and the subsequent bucket; the initial and subsequent are just terms used in this article to describe the feature clearly, there is no initial or subsequent Apache Doris bucket.

As we know from the section above on creating buckets, `BUCKET_DESC` is very simple, but you need to specify the number of buckets; for the automatic bucket projection feature, the syntax of BUCKET_DESC directly changes the number of buckets to `Auto` and adds a new Properties configuration.

```
-- old version of the creation syntax for specifying the number of buckets
DISTRIBUTED BY HASH(site) BUCKETS 20

-- Newer versions use the creation syntax for automatic bucket imputation
DISTRIBUTED BY HASH(site) BUCKETS AUTO
properties("estimate_partition_size" = "100G")
```

The new configuration parameter estimate_partition_size indicates the amount of data for a single partition. This parameter is optional and if not given, Doris will take the default value of estimate_partition_size to 10GB.

As you know from the above, a partitioned bucket is a Tablet at the physical level, and for best performance, it is recommended that the Tablet size be in the range of 1GB - 10GB. So how does the automatic bucketing projection ensure that the Tablet size is within this range?

To summarize, there are a few principles.

- If the overall data volume is small, the number of buckets should not be set too high
- If the overall data volume is large, the number of buckets should be related to the total number of disk blocks, so as to fully utilize the capacity of each BE machine and each disk

> **propertie estimate_partition_size not support alter**

Initial bucketing projection

1. Obtain a number of buckets N based on the data size. Initially, we divide the value of `estimate_partition_size` by 5 (considering a data compression ratio of 5 to 1 when storing data in text format in Doris). The result obtained is

```
(, 100MB), then take N=1

[100MB, 1GB), then take N=2

(1GB, ), then one bucket per GB
```

2. calculate the number of buckets M based on the number of BE nodes and the disk capacity of each BE node.

```
Where each BE node counts as 1, and every 50G of disk capacity counts as 1.

The calculation rule for M is: M = Number of BE nodes * (Size of one disk block / 50GB) * Number of disk blocks.

For example: If there are 3 BEs, and each BE has 4 disks of 500GB, then M = 3 * (500GB / 50GB) * 4 = 120.
```

3. Calculation logic to get the final number of buckets.

```
Calculate an intermediate value x = min(M, N, 128).

If x < N and x < the number of BE nodes, the final bucket is y.

The number of BE nodes; otherwise, the final bucket is x.
```

4. x = max(x, autobucket_min_buckets), Here autobucket_min_buckets is configured in Config (where, default is 1)

The pseudo-code representation of the above process is as follows

```
int N = Compute the N value;
int M = compute M value;

int y = number of BE nodes;
int x = min(M, N, 128);

if (x < N && x < y) {
  return y;
}
return x;
```

With the above algorithm in mind, let's introduce some examples to better understand this part of the logic.

```
case1:
Amount of data 100 MB, 10 BE machines, 2TB * 3 disks
```

```
Amount of data N = 1
BE disks M = 10* (2TB/50GB) * 3 = 1230
x = min(M, N, 128) = 1
Final: 1

case2:
Data volume 1GB, 3 BE machines, 500GB * 2 disks
Amount of data N = 2
BE disks M = 3* (500GB/50GB) * 2 = 60
x = min(M, N, 128) = 2
Final: 2

case3:
Data volume 100GB, 3 BE machines, 500GB * 2 disks
Amount of data N = 20
BE disks M = 3* (500GB/50GB) * 2 = 60
x = min(M, N, 128) = 20
Final: 20

case4:
Data volume 500GB, 3 BE machines, 1TB * 1 disk
Data volume N = 100
BE disks M = 3* (1TB /50GB) * 1 = 60
x = min(M, N, 128) = 63
Final: 63

case5:
Data volume 500GB, 10 BE machines, 2TB * 3 disks
Amount of data N = 100
BE disks M = 10* (2TB / 50GB) * 3 = 1230
x = min(M, N, 128) = 100
Final: 100

case 6:
Data volume 1TB, 10 BE machines, 2TB * 3 disks
Amount of data N = 205
BE disks M = 10* (2TB / 50GB) * 3 = 1230
x = min(M, N, 128) = 128
Final: 128

case 7:
Data volume 500GB, 1 BE machine, 100TB * 1 disk
Amount of data N = 100
BE disk M = 1* (100TB / 50GB) * 1 = 2048
x = min(M, N, 128) = 100
Final: 100

case 8:
Data volume 1TB, 200 BE machines, 4TB * 7 disks
Amount of data N = 205
BE disks M = 200* (4TB / 50GB) * 7 = 114800
x = min(M, N, 128) = 128
Final: 200
```

Subsequent bucketing projection

The above is the calculation logic for the initial bucketing. The subsequent bucketing can be evaluated based on the amount of partition data available since there is already a certain amount of partition data. The subsequent bucket size is evaluated based on the EMA[1] (short term exponential moving average) value of up to the first 7 partitions, which is used as the estimate_partition_size. At this point there are two ways to

calculate the partition buckets, assuming partitioning by days, counting forward to the first day partition size of S7, counting forward to the second day partition size of S6, and so on to S1.

- If the partition data in 7 days is strictly increasing daily, then the trend value will be taken at this time. There are 6 delta values, which are

```
S7 - S6 = delta1,
S6 - S5 = delta2,
...
S2 - S1 = delta6
```

This yields the ema(delta) value.Then, today's estimate_partition_size = S7 + ema(delta)

- not the first case, this time directly take the average of the previous days EMA. Today's estimate_partition_size = EMA(S1, … , S7) , S7)

> **According to the above algorithm, the initial number of buckets and the number of subsequent buckets can be calculated. Unlike before when only a fixed number of buckets could be specified, due to changes in business data, it is possible that the number of buckets in the previous partition is different from the number of buckets in the next partition, which is transparent to the user, and the user does not need to care about the exact number of buckets in each partition, and this automatic extrapolation will make the number of buckets more reasonable.**

Common Issues

1. Incomplete syntax error prompts may occur in longer table creation statements. Here are some possible syntax errors for manual troubleshooting:

    - Syntax structure errors. Please carefully read HELP CREATE TABLE and check the relevant syntax structure.
    - Reserved words. When user-defined names encounter reserved words, they need to be enclosed in backticks ``. It is recommended to use this symbol for all custom names.
    - Chinese characters or full-width characters. Non-UTF8 encoded Chinese characters or hidden full-width characters (spaces, punctuation, etc.) can cause syntax errors. It is recommended to use a text editor that displays invisible characters for inspection.

2. Failed to create partition [xxx]. Timeout

    Doris creates tables sequentially based on partition granularity. When a partition fails to create, this error may occur. Even if partitions are not used, when there is a problem with table creation, `Failed to create partition` may still be reported because, as mentioned earlier, Doris creates an unmodifiable default partition for tables without specified partitions.

    When encountering this error, it is usually because the BE encountered a problem when creating data tablets. You can troubleshoot by following these steps:

    - In the fe.log, search for the `Failed to create partition` log entry at the corresponding timestamp. In this log entry, you may find a series of number pairs similar to `{10001-10010}`. The first number in the pair represents the Backend ID, and the second number represents the Tablet ID. For example, this number pair indicates that the creation of Tablet ID 10010 on Backend ID 10001 failed.
    - Go to the be.INFO log of the corresponding Backend and search for tablet ID-related logs within the corresponding time period to find error messages.
    - Here are some common tablet creation failure errors, including but not limited to:
        - The BE did not receive the relevant task. In this case, you cannot find tablet ID-related logs in be.INFO or the BE reports success but actually fails. For these issues, please refer to the Installation and Deployment section to check the connectivity between FE and BE.
        - Pre-allocated memory failure. This may be because the byte length of a row in the table exceeds 100KB.
        - `Too many open files`. The number of open file handles exceeds the Linux system limit. You need to modify the handle limit of the Linux system.

- If there is a timeout when creating data tablets, you can also extend the timeout by setting `tablet_create_timeout_second=xxx` and `max_create_table_timeout_second=xxx` in the fe.conf file. By default, `tablet_create_timeout_second` is set to 1 second, and `max_create_table_timeout_second` is set to 60 seconds. The overall timeout is calculated as `min(tablet_create_timeout_second * replication_num, max_create_table_timeout_second)`. For specific parameter settings, please refer to the FE Configuration section.

3. The table creation command does not return results for a long time.

- Doris's table creation command is a synchronous command. The timeout for this command is currently set simply as (tablet num * replication num) seconds. If many data tablets are created and some of them fail to create, it may result in a long wait before returning an error.
- Under normal circumstances, the table creation statement should return within a few seconds or tens of seconds. If it exceeds one minute, it is recommended to cancel the operation directly and check the relevant errors in the FE or BE logs.

More Help

For more detailed information on data partitioning, you can refer to the CREATE TABLE command manual or enter `HELP CREATE TABLE;` in the MySQL client to get more help information.

Schema Change

Users can modify the schema of an existing table through the Schema Change operation. The schema of a table mainly includes the modification of columns and the modification of indexes. Here we mainly introduce the column-related Scheme changes. For index-related changes, you can check the data table design/table index to see the change method of each index.

Glossary

- Base Table: When each table is created, it corresponds to a base table.
- Rollup: A roll-up table created based on a base table or other rollup.
- Index: materialized index. Rollup or base table are both called materialized indexes.
- Transaction: Each import task is a transaction, and each transaction has a unique increasing transaction ID.

Introduction

Light Schema Change

Before introduction, it is necessary to know the three Schema Change implementations before the Apache Doris 1.2.0 version, all of which are asynchronous:

- Hard linked schema change mainly acts on addition and subtraction of value columns and does not require modification of the data file.
- Direct schema change is mainly used to change the type of value column, which needs to rewrite the data, but does not involve the key column, and does not need to be reordered.
- Sort schema change is mainly used for the key column schema change, because the key column addition/subtraction/modification type and other operations will affect the sorting of existing data, so the data needs to be read out again, modified, and then sort.

Since Apache Doris 1.2.0 , for the first type, the new feature of light schema change has been introduced. The new light schema change allows the addition and subtraction of the value column to be completed in milliseconds. Starting from Apache Doris 2.0.0, all newly created tables have enabled light schema change by default.

In addition to adding and deleting the value column, the main principles of other types of schema changes are as follows

The basic process of executing schema change is to generate a new schema table from the data /Index in the original table /Index data. There are mainly two parts of data conversion, one is the conversion of existing historical data, and the other is the conversion of newly arrived imported data during the execution of schema change.



Before starting to convert historical data, Doris will obtain a latest transaction ID and wait for all import transactions before this transaction ID to complete. This transaction ID becomes a watershed. This means that Doris ensures that all import tasks after the watershed will generate data for the original table /Index and the new table /Index at the same time. This way, when the historical data conversion is completed, the data in the new table can be guaranteed to be complete.

The specific syntax for creating schema changes can be found in the schema change section of the help ALTER TABLE COLUMN

Adding a column at a specified position to a specified index

Syntax

```
ALTER TABLE table_name ADD COLUMN column_name column_type [KEY | agg_type] [DEFAULT "default_value"]
[AFTER column_name|FIRST]
[TO rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

- For aggregate models, if adding a value column, specify `agg_type`.
- For non-aggregate models (e.g., DUPLICATE KEY), if adding a key column, specify the `KEY` keyword.
- You cannot add a column to a rollup index that already exists in the base index (if needed, you can create a new rollup index).

Examples

non-aggregate model

table's DDL:

```
CREATE TABLE IF NOT EXISTS example_db.my_table(
    col1 int,
    col2 int,
    col3 int,
```

```
    col4 int,
    col5 int
) DUPLICATE KEY(col1, col2, col3)
DISTRIBUTED BY RANDOM BUCKETS 1
ROLLUP (
   example_rollup_index (col1, col3, col4, col5)
)
PROPERTIES (
   "replication_num" = "1"
)
```

1. Adding a key column `new_col` after col1 to `example_rollup_index`

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT KEY DEFAULT "0" AFTER col1
TO example_rollup_index;
```

2. Adding a value column `new_col` with a default value of 0 after col1 to `example_rollup_index`

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT DEFAULT "0" AFTER col1
TO example_rollup_index;
```

aggregate model

table's DDL:

```
CREATE TABLE IF NOT EXISTS example_db.my_table(
    col1 int,
    col2 int,
    col3 int,
    col4 int SUM,
    col5 varchar(32) REPLACE DEFAULT "abc"
) AGGREGATE KEY(col1, col2, col3)
DISTRIBUTED BY HASH(col1) BUCKETS 1
ROLLUP (
    example_rollup_index (col1, col3, col4, col5)
)
PROPERTIES (
    "replication_num" = "1"
)
```

3. Adding a Key column `new_col` after col1 to `example_rollup_index`

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT DEFAULT "0" AFTER col1
TO example_rollup_index;
```

4. Adding a value column `new_col` with SUM aggregation type after col1 to `example_rollup_index`

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT SUM DEFAULT "0" AFTER col1
TO example_rollup_index;
```

Adding multiple columns to a specified index

Syntax

```
ALTER TABLE table_name ADD COLUMN (column_name1 column_type [KEY | agg_type] DEFAULT "default_value", ...)
[TO rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

- For aggregate models, if adding a value column, specify `agg_type`.
- For aggregate models, if adding a key column, specify the KEY keyword.
- You cannot add a column to a rollup index that already exists in the base index (if needed, you can create a new rollup index).

Example

Adding multiple columns (aggregate model) to `example_rollup_index`:

```
ALTER TABLE example_db.my_table
ADD COLUMN (c1 INT DEFAULT "1", c2 FLOAT SUM DEFAULT "0")
TO example_rollup_index;
```

Removing a column from a specified index

Syntax

```
ALTER TABLE table_name DROP COLUMN column_name
[FROM rollup_index_name]
```

- You cannot delete a partition column.
- If you delete a column from the base index, it will also be removed from the rollup index if it contains the column.

Example

Removing column col2 from `example_rollup_index`:

```
ALTER TABLE example_db.my_table
DROP COLUMN col3
FROM example_rollup_index;
```

Modifying the column type and position of a specified index

Syntax

```
ALTER TABLE table_name MODIFY COLUMN column_name column_type [KEY | agg_type] [NULL | NOT NULL] [DEFAULT "default_value"]
[AFTER column_name|FIRST]
[FROM rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

- For aggregate models, if modifying a value column, specify `agg_type`.
- For non-aggregate models, if modifying a key column, specify the **KEY** keyword.
- Only the column type can be modified, other column properties remain the same (i.e., other properties need to be explicitly written in the statement, see Example 8).
- Partition and bucket columns cannot be modified.
- Currently, the following type conversions are supported (users need to ensure precision loss):
    - Conversion from TINYINT/SMALLINT/INT/BIGINT/LARGEINT/FLOAT/DOUBLE to a larger numeric type.
    - Conversion from TINYINT/SMALLINT/INT/BIGINT/LARGEINT/FLOAT/DOUBLE/DECIMAL to VARCHAR.
    - Modification of maximum length for VARCHAR.
    - Conversion from VARCHAR/CHAR to TINYINT/SMALLINT/INT/BIGINT/LARGEINT/FLOAT/DOUBLE.
    - Conversion from VARCHAR/CHAR to DATE (supports six formatting formats: “%Y-%m-%d”, “%y-%m-%d”, “%Y%m%d”, “%y%m%d”, “%Y/%m/%d”, “%y/%m/%d”).
    - Conversion from DATETIME to DATE (only retains year-month-day information, e.g., `2019-12-09 21:47:05` <–> `2019-12-09`).
    - Conversion from DATE to DATETIME (automatically adds zeros for hours, minutes, and seconds, e.g., `2019-12-09` <–> `2019-12-09 00:00:00`).
    - Conversion from FLOAT to DOUBLE.
    - Conversion from INT to DATE (if the INT type data is invalid, the conversion fails, and the original data remains unchanged).
    - All types except DATE and DATETIME can be converted to STRING, but STRING cannot be converted to any other type.

Examples

table’s DDL:

```
CREATE TABLE IF NOT EXISTS example_db.my_table(
    col0 int,
    col1 int DEFAULT "1",
    col2 int,
    col3 varchar(32),
    col4 int SUM,
    col5 varchar(32) REPLACE DEFAULT "abc"
) AGGREGATE KEY(col0, col1, col2, col3)
DISTRIBUTED BY HASH(col0) BUCKETS 1
PROPERTIES (
    "replication_num" = "1"
)
```

1. Modifying the column type of Key column col1 to BIGINT in the base index and moving it after column col2

```
ALTER TABLE example_db.my_table
MODIFY COLUMN col1 BIGINT KEY DEFAULT "1" AFTER col2;
```

Note: whether modifying a key column or a value column, the complete column information needs to be declared.

2. Modifying the maximum length of column val5 in the Base Index. The original val5 is (val5 VARCHAR(32) REPLACE DEFAULT “abc”)

```
ALTER TABLE example_db.my_table
MODIFY COLUMN col5 VARCHAR(64) REPLACE DEFAULT "abc";
```

Note: only the column type can be modified while keeping the other properties of the column unchanged.

3. Modifying the length of a field in the key column of a duplicate key table

```
ALTER TABLE example_db.my_table
MODIFY COLUMN col3 varchar(50) KEY NULL comment 'to 50'
```

Reorder columns for a specified index

Syntax

```
ALTER TABLE table_name ORDER BY (column_name1, column_name2, ...)
[FROM rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

- All columns in the index should be listed.
- Value columns come after key columns.

Example

```
CREATE TABLE IF NOT EXISTS example_db.my_table(
    k1 int DEFAULT "1",
```

```
    k2 int,
    k3 varchar(32),
    k4 date,
    v1 int SUM,
    v2 int MAX,
) AGGREGATE KEY(k1, k2, k3, k4)
DISTRIBUTED BY HASH(k1) BUCKETS 1
ROLLUP (
   example_rollup_index(k1, k2, k3, v1, v2)
)
PROPERTIES (
    "replication_num" = "1"
)
```

Reorder columns in the index `example_rollup_index` (assuming the original column order is: k1, k2, k3, v1, v2).

```
ALTER TABLE example_db.my_table
ORDER BY (k3,k1,k2,v2,v1)
FROM example_rollup_index;
```

Perform multiple changes in one submission

Schema change can modify multiple indexes in a single job.

Example 1

Source Schema:

```
CREATE TABLE IF NOT EXISTS example_db.tbl1(
    k1 int,
    k2 int,
    k3 int
) AGGREGATE KEY(k1, k2, k3)
DISTRIBUTED BY HASH(k1) BUCKETS 1
ROLLUP (
   rollup1 (k1, k2),
   rollup2 (k2)
)
PROPERTIES (
    "replication_num" = "1"
)
```

You can use the following command to add a column k4 to rollup1 and rollup2, and add an additional column k5 to rollup2:

```
ALTER TABLE tbl1
ADD COLUMN k4 INT default "1" to rollup1,
ADD COLUMN k4 INT default "1" to rollup2,
ADD COLUMN k5 INT default "1" to rollup2;
```

After completion, the schema becomes:

```
+-----------+-------+------+------+------+---------+-------+
| IndexName | Field | Type | Null | Key  | Default | Extra |
+-----------+-------+------+------+------+---------+-------+
```

```
| tbl1      | k1    | INT  | No   | true | N/A     |       |
|           | k2    | INT  | No   | true | N/A     |       |
|           | k3    | INT  | No   | true | N/A     |       |
|           | k4    | INT  | No   | true | 1       |       |
|           | k5    | INT  | No   | true | 1       |       |
|           |       |      |      |      |         |       |
| rollup2   | k2    | INT  | No   | true | N/A     |       |
|           | k4    | INT  | No   | true | 1       |       |
|           | k5    | INT  | No   | true | 1       |       |
|           |       |      |      |      |         |       |
| rollup1   | k1    | INT  | No   | true | N/A     |       |
|           | k2    | INT  | No   | true | N/A     |       |
|           | k4    | INT  | No   | true | 1       |       |
+-----------+-------+------+------+------+---------+-------+
```

As seen, the base table tbl1 automatically includes the columns k4 and k5. Any columns added to a rollup will automatically be added to the base table.

Additionally, it is not allowed to add columns to a rollup that already exist in the base table. If a user needs to do so, they can create a new rollup with the additional columns and then delete the original rollup.

Example 2

table's DDL

```
CREATE TABLE IF NOT EXISTS example_db.my_table(
    k1 int DEFAULT "1",
    k2 int,
    k3 varchar(32),
    k4 date,
    v1 int SUM,
) AGGREGATE KEY(k1, k2, k3, k4)
DISTRIBUTED BY HASH(k1) BUCKETS 1
ROLLUP (
   example_rollup_index(k1, k3, k2, v1)
)
PROPERTIES (
   "replication_num" = "1"
)
```

```
ALTER TABLE example_db.my_table
ADD COLUMN v2 INT MAX DEFAULT "0" TO example_rollup_index,
ORDER BY (k3,k1,k2,v2,v1) FROM example_rollup_index;
```

Rename Column

Syntax

```
ALTER TABLE RENAME COLUMN old_column_name new_column_name;
```

Check Job Status

The creation of a schema change is an asynchronous process. After a job is successfully submitted, users need to use the `SHOW ALTER TABLE COLUMN` command to check the progress of the job.

`SHOW ALTER TABLE COLUMN` allows you to view the currently executing or completed schema Change jobs. When a schema change job involves multiple indexes, the command will display multiple rows, with each row corresponding to an index. For example:

```
mysql > SHOW ALTER TABLE COLUMN\G;
*************************** 1. row ***************************
        JobId: 20021
    TableName: tbl1
   CreateTime: 2019-08-05 23:03:13
   FinishTime: 2019-08-05 23:03:42
    IndexName: tbl1
      IndexId: 20022
OriginIndexId: 20017
SchemaVersion: 2:792557838
TransactionId: 10023
        State: FINISHED
          Msg:
     Progress: NULL
      Timeout: 86400
1 row in set (0.00 sec)
```

- JobId: unique ID for each schema change job.
- TableName: name of the base table associated with schema change.
- CreateTime: time when the job was created.
- FinishTime: time when the job is finished. If not finished, it displays “N/A”.
- IndexName: name of one of the indexes involved in the modification.
- IndexId: unique ID of the new index.
- OriginIndexId: unique ID of the old index.
- SchemaVersion: displayed in the format M:N, where M represents the version of the schema change modification, and N represents the corresponding hash value. The version increments with each schema change.
- TransactionId: transaction ID that serves as the boundary for transforming historical data.
- State: current state of the job.
    - PENDING: the job is waiting in the queue to be scheduled.
    - WAITING_TXN: waiting for import tasks before the boundary transaction ID to complete.
    - RUNNING: historical data transformation is in progress.
    - FINISHED: the job completed successfully.
    - CANCELLED: the job was cancelled.
- Msg: if the job fails, this field displays the failure message.
- Progress: job progress. It is only displayed when the state is RUNNING. Progress is shown in the format M/N, where N is the total number of replicas involved in the schema change, and M is the number of

replicas that have completed historical data transformation.

- Timeout: job timeout in seconds.

Cancel Job

In cases where the job state is not FINISHED or CANCELLED, you can cancel the schema change job using the following command:

```
CANCEL ALTER TABLE COLUMN FROM tbl_name;
```

Notes

- Only one schema change job can run on a table at a time.
- Schema change operations do not block import and query operations.
- Partition and bucket columns cannot be modified.
- If the schema includes value columns aggregated using the REPLACE method, it is not allowed to delete key columns.
- If a key column is deleted, Doris cannot determine the values for the REPLACE columns.
- All non-key columns in a unique data model table are aggregated using the REPLACE method.
- When adding a value column with an aggregation type of SUM or REPLACE, the default value for that column has no meaning for historical data.
- Since the historical data has lost detailed information, the default value does not reflect the actual aggregated value.
- When modifying the column type, all fields except Type need to be completed based on the information from the original column.
- For example, to modify the column `k1 from INT SUM NULL DEFAULT "1"` to BIGINT, the command would be: `ALTER TABLE tbl1 MODIFY COLUMN k1 BIGINT SUM NULL DEFAULT "1";`
- Note that apart from the new column type, other attributes such as the aggregation method, nullable property, and default value should be completed based on the original information.
- It is not supported to modify column names, aggregation types, nullable properties, default values, or column comments.

FAQs

Schema change execution speed

For a Light Schema Change, such as adding or deleting Value columns, the execution speed can be in the millisecond range. For other types of Schema Change, the execution speed is estimated to be around 10MB/s in the worst-case scenario. As a conservative measure, users can set the job timeout based on this speed.

Error: `"Table xxx is not stable"` **when submitting a job**

Schema Change can only be initiated when the table data is complete and in a balanced state. If some data shard replicas of the table are incomplete or if some replicas are undergoing balancing operations, the submission will be rejected. You can check if the data shard replicas are complete using the following command:

```
SHOW REPLICA STATUS FROM tbl WHERE STATUS != "OK";
```

If there are any results returned, it indicates that there are issues with the replicas. Typically, the system will automatically repair these issues, but users can prioritize the repair for a specific table using the following command:

```
ADMIN REPAIR TABLE tbl1;
```

You can check if there are any running balancing tasks using the following command:

```
SHOW PROC "/cluster_balance/pending_tablets";
```

You can wait for the balancing tasks to complete or temporarily disable balancing operations using the following command:

```
ADMIN SET FRONTEND CONFIG ("disable_balance" = "true");
```

Configurations

FE Configuration

- `alter_table_timeout_second`: Default timeout for jobs, set to 86400 seconds.

BE Configuration

- `alter_tablet_worker_count`: Number of threads used on the BE side for executing historical data transformation. The default is 3. If you want to speed up Schema Change jobs, you can increase this parameter and restart the BE. However, having too many transformation threads may increase IO pressure and affect other operations. This thread is shared with Rollup jobs.
- `alter_index_worker_count`: Number of threads used on the BE side for building indexes on historical data (currently only supports inverted indexes). The default is 3. If you want to speed up Index Change jobs, you can increase this parameter and restart the BE. However, having too many threads may increase IO pressure and affect other operations.

More Details

For more detailed syntax and best practices regarding Schema Change, please refer to the ALTER TABLE COLUMN command manual. You can also enter `HELP ALTER TABLE COLUMN`in the MySQL client command line for more help information.

Tiered Storage

Use Case

One significant use case in the future is similar to ES log storage, where data in the log scenario is split based on dates. Many of the data are cold data with infrequent queries, requiring a reduction in storage costs for such data. Considering cost-saving:

- The pricing of regular cloud disks from various vendors is more expensive than object storage.
- In actual online usage of the Doris Cluster, the utilization of regular cloud disks cannot reach 100%.
- Cloud disks are not billed on demand, while object storage can be billed on demand.
- Using regular cloud disks for high availability requires multiple replicas and replica migration in case of failures. In contrast, storing data on object storage eliminates these issues as it is shared.

Solution

Set the freeze time at the partition level, which indicates how long a partition will be frozen, and define the location of remote storage for storing data after freezing. In the BE (Backend) daemon thread, the table's freeze condition is periodically checked. If a freeze condition is met, the data will be uploaded to object storage compatible with the S3 protocol and HDFS.

Cold-hot tiering supports all Doris functionalities and only moves some data to object storage to save costs without sacrificing functionality. Therefore, it has the following characteristics:

- Cold data is stored on object storage, and users do not need to worry about data consistency and security.
- Flexible freeze strategy, where the cold remote storage property can be applied to both table and partition levels.
- Users can query data without worrying about data distribution. If the data is not local, it will be pulled from the object storage and cached locally in the BE (Backend).
- Replica clone optimization. If the stored data is on object storage, there is no need to fetch the stored data locally during replica cloning.
- Remote object space recycling. If a table or partition is deleted or if space waste occurs during the cold-hot tiering process due to exceptional situations, a recycler thread will periodically recycle the space, saving storage resources.
- Cache optimization, caching accessed cold data locally in the BE to achieve query performance similar to non-cold-hot tiering.
- BE thread pool optimization, distinguishing between data sources from local and object storage to prevent delays in reading objects from impacting query performance.

Usage of Storage Policy

The storage policy is the entry point for using the cold-hot tiering feature. Users only need to associate the storage policy with a table or partition during table creation or when using Doris.

> **When creating an S3 resource, a remote S3 connection validation is performed to ensure the correct creation of the resource.**

Here is an example of creating an S3 resource:

```
CREATE RESOURCE "remote_s3"
PROPERTIES
(
    "type" = "s3",
    "s3.endpoint" = "bj.s3.com",
    "s3.region" = "bj",
    "s3.bucket" = "test-bucket",
    "s3.root.path" = "path/to/root",
    "s3.access_key" = "bbb",
    "s3.secret_key" = "aaaa",
    "s3.connection.maximum" = "50",
    "s3.connection.request.timeout" = "3000",
    "s3.connection.timeout" = "1000"
);

CREATE STORAGE POLICY test_policy
PROPERTIES(
    "storage_resource" = "remote_s3",
    "cooldown_ttl" = "1d"
);

CREATE TABLE IF NOT EXISTS create_table_use_created_policy
(
    k1 BIGINT,
    k2 LARGEINT,
    v1 VARCHAR(2048)
)
UNIQUE KEY(k1)
DISTRIBUTED BY HASH (k1) BUCKETS 3
PROPERTIES(
    "enable_unique_key_merge_on_write" = "false",
    "storage_policy" = "test_policy"
);
```

> **Notice**
>
> If you set `"enable_unique_key_merge_on_write" = "true"` in UNIQUE table, you can't use this feature.

And here is an example of creating an HDFS resource:

```
CREATE RESOURCE "remote_hdfs" PROPERTIES (
        "type"="hdfs",
        "fs.defaultFS"="fs_host:default_fs_port",
        "hadoop.username"="hive",
```

```
        "hadoop.password"="hive",
        "root_path"="/my/root/path",
        "dfs.nameservices" = "my_ha",
        "dfs.ha.namenodes.my_ha" = "my_namenode1, my_namenode2",
        "dfs.namenode.rpc-address.my_ha.my_namenode1" = "nn1_host:rpc_port",
        "dfs.namenode.rpc-address.my_ha.my_namenode2" = "nn2_host:rpc_port",
        "dfs.client.failover.proxy.provider.my_ha" = "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"
    );

CREATE STORAGE POLICY test_policy PROPERTIES (
    "storage_resource" = "remote_hdfs",
    "cooldown_ttl" = "300"
)

CREATE TABLE IF NOT EXISTS create_table_use_created_policy (
    k1 BIGINT,
    k2 LARGEINTv1 VARCHAR(2048)
)
UNIQUE KEY(k1)
DISTRIBUTED BY HASH (k1) BUCKETS 3
PROPERTIES(
    "enable_unique_key_merge_on_write" = "false",
    "storage_policy" = "test_policy"
);
```

> **Notice**
>
> If you set `"enable_unique_key_merge_on_write" = "true"` in UNIQUE table, you can't use this feature.

Associate a storage policy with an existing table by using the following command:

```
ALTER TABLE create_table_not_have_policy SET ("storage_policy" = "test_policy");
```

Associate a storage policy with an existing partition by using the following command:

```
ALTER TABLE create_table_partition MODIFY PARTITION (*) SET ("storage_policy" = "test_policy");
```

> **If you specify different storage policies for the entire table and some partitions during table creation, the storage policy set for the partitions will be ignored, and all partitions of the table will use the table's storage policy. If you want a specific partition to have a different storage policy than the others, you can use the method mentioned above to modify the association for that specific partition.**
>
> For more details, please refer to the following documents in the Docs directory: RESOURCE, POLICY, CREATE TABLE, ALTER TABLE, which provide detailed explanations.

Limitations

- A single table or partition can only be associated with one storage policy. Once associated, the storage policy cannot be dropped without first removing the association between them.

- The object information associated with a storage policy does not support modifying the data storage path, such as bucket, endpoint, root_path, and other information.
- Storage policies support creation, modification, and deletion. Before deleting a storage policy, ensure that no tables are referencing the storage policy.
- When the Merge-on-Write feature is enabled, the Unique model does not support setting a storage policy.

Occupied Size of Cold Data Objects

Method 1: You can use the `show proc '/backends'` command to view the size of each backend's uploaded objects. Look for the `RemoteUsedCapacity` field. Please note that this method may have some latency.

Method 2: You can use the `show tablets from tableName` command to view the size of each tablet in a table, indicated by the `RemoteDataSize` field.

Cache for Cold Data

As mentioned earlier, caching is introduced for cold data to optimize query performance and save object storage resources. When cold data is first accessed after cooling, Doris reloads the cooled data onto the local disk of the backend (BE). The cold data cache has the following characteristics:

- The cache is stored on the BE's disk and does not occupy memory space.
- The cache can be limited in size and uses LRU (Least Recently Used) for data eviction.
- The implementation of the cache for cold data is the same as the cache for federated query catalog. Please refer to the documentation at Filecache for more details.

Compaction of Cold Data

The time at which cold data enters is counted from the moment the data rowset file is written to the local disk, plus the cooling duration. Since data is not written and cooled all at once, Doris performs compaction on cold data to avoid the issue of small files within object storage. However, the frequency and resource prioritization of cold data compaction are not very high. It is recommended to perform compaction on local hot data before cooling. You can adjust the following BE parameters:

- The BE parameter `cold_data_compaction_thread_num` sets the concurrency for cold data compaction. The default value is 2.
- The BE parameter `cold_data_compaction_interval_sec` sets the time interval for cold data compaction. The default value is 1800 seconds (30 minutes).

Schema Change for Cold Data

The following schema change types are supported for cold data:

- Adding or deleting columns
- Modifying column types

- Adjusting column order
- Adding or modifying indexes

Garbage Collection of Cold Data

Garbage data for cold data refers to data that is not used by any replica. The following situations may generate garbage data on object storage:

1. Partial segment upload succeeds while the upload of the rowset fails.
2. After the FE reselects the CooldownReplica, the rowset versions of the old and new CooldownReplica do not match. FollowerReplicas synchronize the CooldownMeta of the new CooldownReplica, and the rowsets with inconsistent versions in the old CooldownReplica become garbage data.
3. After cold data compaction, the rowsets before merging cannot be immediately deleted because they may still be used by other replicas. However, eventually, all FollowerReplicas use the latest merged rowset, and the rowsets before merging become garbage data.

Furthermore, the garbage data on objects is not immediately cleaned up. The BE parameter `remove_unused_remote_files_interval_sec` sets the time interval for garbage collection of cold data. The default value is 21600 seconds (6 hours).

TODOs

- Some remote occupancy metrics may not have comprehensive update retrieval.

FAQs

1. `ERROR 1105 (HY000): errCode = 2, detailMessage = Failed to create repository: connect to s3 failed: Unable to marshall request to JSON: host must not be null.`

The S3 SDK defaults to using the virtual-hosted style. However, some object storage systems (e.g., MinIO) may not have virtual-hosted style access enabled or supported. In such cases, you can add the `use_path_style` parameter to force the use of path-style access:

```
CREATE RESOURCE "remote_s3"
PROPERTIES
(
    "type" = "s3",
    "s3.endpoint" = "bj.s3.com",
    "s3.region" = "bj",
    "s3.bucket" = "test-bucket",
    "s3.root.path" = "path/to/root",
    "s3.access_key" = "bbb",
    "s3.secret_key" = "aaaa",
    "s3.connection.maximum" = "50",
    "s3.connection.request.timeout" = "3000",
    "s3.connection.timeout" = "1000",
    "use_path_style" = "true"
);
```

Index Overview

Database indexes are used to accelerate queries. To speed up different query scenarios, Doris supports various rich indexes.

Index Types and Principles

From the perspective of accelerating queries and their principles, Doris indexes are categorized into two main types: point query indexes and skip indexes. - **Point Query Indexes:** Commonly used to speed up point queries, the principle is to locate which rows satisfy the WHERE conditions through the index and directly read those rows. Point query indexes are very effective when the number of rows meeting the conditions is small. Doris's point query indexes include Prefix Index and Inverted Index. - **Prefix Index:** Doris stores data in an ordered manner according to the sort key and creates a sparse prefix index every 1024 rows. The key in the index is the value of the sorted column in the first row of the current 1024 rows. If the query involves a sorted column, the system will find the first row of the relevant 1024-row group and start scanning from there. - **Inverted Index:** For columns with an inverted index, an posting list is created mapping each value to a set of row ids. For equality queries, it first finds the set of row ids from the posting list, then directly reads the data of those rows, avoiding row-by-row scanning. Inverted indexes can also accelerate range filtering and fulltext search. The algorithms are more complex, but the basic principles are similar. (Note: the previous BITMAP index has been replaced by the more powerful inverted index.) - **Skip Indexes:** Commonly used to accelerate analysis, the principle is to determine data blocks that do not satisfy the WHERE conditions through the index and skip these blocks, only reading the data blocks that may satisfy the conditions and then performing a row-by-row filter to finally get the rows that meet the conditions. Skip indexes are more effective when the number of rows meeting the conditions is large. Doris's skip indexes include ZoneMap indexes, BloomFilter indexes, and NGram BloomFilter indexes. - **ZoneMap Index:** Automatically maintains statistics for each column, recording the maximum, minimum, and whether there are NULL values for each data file (Segment) and data block (Page). For equality queries, range queries, and IS NULL, it can determine whether the data file and data block can contain the data that meets the conditions based on the maximum value, minimum value, and whether there are NULL values. If not, Doris skips reading the corresponding file or data block, reducing IO and accelerating queries. - **BloomFilter Index:** Stores values of the indexed column in a BloomFilter data structure, which can quickly determine whether a value is in the BloomFilter with very low storage space. For equality queries, if the value is not in the BloomFilter, the corresponding data file or data block can be skipped, reducing IO and accelerating queries. - **NGram BloomFilter Index:** Used to accelerate text LIKE queries. The principle is similar to the BloomFilter index, but instead of storing the original text values, it performs NGram tokenization of the text and stores each token in the BloomFilter. For LIKE queries, the LIKE pattern is also tokenized using NGram. If any token is not in the BloomFilter, the corresponding data file or data block does not meet the LIKE condition and can be skipped.

Among the above indexes, the prefix index and ZoneMap index are built-in indexes automatically maintained by Doris, requiring no user management. Inverted indexes, BloomFilter indexes, and NGram BloomFilter indexes need to be manually created and managed by the user based on the scenario.

- Co
- Comparison of characteristics of different types of indexes

Type	Index	Advantages	Limitations
Point Query	Prefix Index	Built-in index, best performance. Only one prefix index per table.	Only one prefix index per table.
Point Query	Inverted Index	Supports tokenization and keyword matching. Building index on any column. Multi-condition combination and accelerating more functions.	Large index storage space, similar to raw data.
Skip	ZoneMap Index	Built-in index, small index storage space. Only one prefix index per table.	Only one prefix index per table.
Skip	BloomFilter Index	More precise than ZoneMap, medium index space.	Supports few query types. Only supports equal (not others: not equal, range, LIKE, MATCH).
Skip	NGram BloomFilter	Supports LIKE acceleration, medium index space.	Supports few query types. Only supports LIKE acceleration.

- List of operators and functions for index acceleration

Operator or Function	Prefix Index	Inverted Index	ZoneMap Index	BloomFilter Index	NGram BloomFilter Index
=	YES	YES	YES	YES	NO
!=	YES	YES	NO	NO	NO
IN	YES	YES	YES	YES	NO
NOT IN	YES	YES	NO	NO	NO
>, >=, <, <=, BETWEEN	YES	YES	YES	NO	NO
IS NULL	YES	YES	YES	NO	NO
IS NOT NULL	YES	YES	NO	NO	NO
LIKE	NO	NO	NO	NO	YES
MATCH, MATCH_*	NO	YES	NO	NO	NO

Index Design Guidelines

The design and optimization of database table indexes are closely related to data distribution and queries, requiring testing and optimization based on the actual scenario. Although there is no “silver bullet,” Doris continuously strives to reduce the difficulty of using indexes. Users can follow these simple guidelines for index selection and testing.

1. Specify the most frequently used filter condition as the KEY to automatically create a prefix index, as it has the best filtering effect. However, only one prefix index can be created per table, so it should be used for the most frequent filter condition.
2. For non-key fields that require filter acceleration, the first choice is to create an inverted index due to its broad applicability and multi-condition combination. The second choice includes the following two

indexes:

- If there is a need for string LIKE matching, add an NGram BloomFilter index.
- If index storage space is critical, replace the inverted index with a BloomFilter index.

3. If performance is not as expected, analyze the amount of data filtered by the index and the time consumed through QueryProfile. Refer to the detailed documentation of each index for specifics.

Sort Key and Prefix Index

Index Principles

Doris stores data in a structure similar to SSTable (Sorted String Table). This structure is an ordered data structure that can be sorted and stored according to one or more specified columns. In such a data structure, looking up conditions on all or part of the sorted columns is highly efficient.

In the Aggregate, Unique, and Duplicate data models, the underlying data storage is sorted according to the columns specified in the CREATE TABLE statement under AGGREGATE KEY, UNIQUE KEY, and DUPLICATE KEY respectively. These keys are referred to as sort keys. With sort keys, Doris can quickly locate the required data without scanning the entire table by specifying conditions on the sorted columns during a query, thereby reducing search complexity and speeding up the query.

Based on the sort keys, Doris introduces a prefix index. The prefix index is a sparse index. The data in the table forms a logical data block (Data Block) according to the corresponding number of rows. Each logical data block stores an index entry in the prefix index table, where the length of the index entry does not exceed 36 bytes. The entry content is the prefix composed of the sorted columns of the first row in the data block. When looking up the prefix index table, it helps determine the starting row number of the logical data block where the row data is located. Because the prefix index is relatively small, it can be fully cached in memory, allowing for quick data block location and significantly improving query efficiency.

> **The first 36 bytes of a row in a data block are used as the prefix index for that row. When encountering a VARCHAR type, the prefix index is directly truncated. If the first column is VARCHAR, even if it does not reach 36 bytes, it will be directly truncated, and the subsequent columns will not be included in the prefix index.**

Use Cases

Prefix indexes can speed up equality queries and range queries.

Managing Indexes

There is no specific syntax to define a prefix index. When creating a table, the first 36 bytes of the table's Key are automatically taken as the prefix index.

Recommendations for Prefix Index Selection

> **Because the Key definition of a table is unique, a table has only one set of prefix indexes. Therefore, it is important to choose an appropriate prefix index when designing the table structure. The following recommendations can be considered:**
>
> 1. Choose the fields most commonly used in WHERE filtering conditions as the Key.

2. Place the more frequently used fields at the front, as prefix indexes are only effective for fields in the WHERE condition that are part of the Key's prefix.

For queries that use other columns not covered by the prefix index as conditions, the efficiency may not meet the requirements. There are two solutions: 1. Create an inverted index on the columns that require accelerated queries, as a table can have many inverted indexes. 2. For DUPLICATE tables, multi-prefix indexes can be indirectly achieved by creating corresponding strongly consistent materialized views with adjusted column orders. For more details, refer to query acceleration/materialized views.

Using Indexes

Prefix indexes are used to accelerate equality and range queries in the WHERE clause. They automatically take effect when applicable, and there is no special syntax required.

The acceleration effect of the prefix index can be analyzed using the following metrics in the Query Profile: - RowsKeyRangeFiltered: The number of rows filtered by the prefix index, which can be compared with other Rows values to analyze the filtering effect of the index.

Example Usage

- Suppose the sorted columns of a table are as follows, then the prefix index would be: user_id (8 Bytes) + age (4 Bytes) + message (prefix 20 Bytes).

ColumnName	Type
user_id	BIGINT
age	INT
message	VARCHAR(100)
max_dwell_time	DATETIME
min_dwell_time	DATETIME

- Suppose the sorted columns of a table are as follows, then the prefix index would be user_name (20 Bytes). Even if it does not reach 36 bytes, it is directly truncated due to encountering VARCHAR, and subsequent columns are not included.

ColumnName	Type
user_name	VARCHAR(20)
age	INT
message	VARCHAR(100)
max_dwell_time	DATETIME
min_dwell_time	DATETIME

- When our query condition is the prefix of the prefix index, it can significantly speed up the query. For example, in the first example, executing the following query:

```
SELECT * FROM table WHERE user_id = 1829239 AND age = 20;
```

This query will be much more efficient than the following query:

```
SELECT * FROM table WHERE age = 20;
```

Therefore, choosing the correct column order when creating a table can greatly improve query efficiency.

Inverted Index

Indexing Principle

Inverted Index is a commonly used indexing technique in the field of information retrieval. It divides text into individual words and constructs a word -> document IDs index, allowing for quick searches to determine which documents contain a specific word.

Starting from version 2.0.0, Doris supports inverted indexes, which can be used for full-text searches on text types, equality, and range queries on normal numerical and date types, and quickly filtering rows that meet the criteria from massive amounts of data.

In Doris's implementation of the inverted index, each row in the table corresponds to a document, and each column corresponds to a field in the document. Thus, using an inverted index, you can quickly locate rows containing specific keywords, thereby accelerating the WHERE clause.

Unlike other indexes in Doris, the inverted index uses independent files at the storage layer, corresponding one-to-one with data files but physically stored independently. This approach allows for creating and deleting indexes without rewriting data files, significantly reducing processing overhead.

Usage Scenarios

Inverted indexes have a wide range of applications and can accelerate equality, range, and full-text searches (keyword matching, phrase matching, etc.). A table can have multiple inverted indexes, and the conditions of multiple inverted indexes can be combined arbitrarily during queries.

The functionality of inverted indexes is briefly introduced as follows:

1. Accelerate full-text searches for string types

- Support for keyword search, including matching multiple keywords simultaneously `MATCH_ALL` and matching any one keyword `MATCH_ANY`.
- Support for phrase queries `MATCH_PHRASE`
    - Support for specifying slop for word distence
    - Support for phrase + prefix `MATCH_PHRASE_PREFIX`
- Support for tokenized regular expression queries `MATCH_REGEXP`
- Support for English, Chinese, and Unicode tokenizers

2. Accelerate normal equality and range queries, covering and replacing the functionality of BITMAP index

- Support for fast filtering of string, numerical, and datetime types for =, !=, >, >=, <, <=
- Support for fast filtering of string, numerical, and datetime array types for `array_contains`

3. Support for comprehensive logical combinations

- Not only supports acceleration for AND conditions but also for OR and NOT conditions
- Supports arbitrary logical combinations of multiple conditions with AND, OR, NOT

4. Flexible and efficient index management

- Support for defining inverted indexes when creating a table
- Support for adding inverted indexes to existing tables, with incremental index construction without rewriting existing data in the table
- Support for deleting inverted indexes from existing tables without rewriting existing data in the table

> **There are some limitations to using inverted indexes:**
>
> 1. Floating-point types FLOAT and DOUBLE, which have precision issues, do not support inverted indexes due to inaccurate precision. The solution is to use the precisely accurate DECIMAL type, which supports inverted indexes.
> 2. Some complex data types do not yet support inverted indexes, including MAP, STRUCT, JSON, HLL, BITMAP, QUANTILE_STATE, AGG_STATE.
> 3. DUPLICATE and UNIQUE table models with Merge-on-Write enabled support building inverted indexes on any column. However, AGGREGATE and UNIQUE models without Merge-on-Write enabled only support building inverted indexes on Key columns, as non-Key columns cannot have inverted indexes. This is because these two models require reading all data for merging, so indexes cannot be used for pre-filtering.

Managing Indexes

Defining Inverted Indexes When Creating a Table

In the table creation statement, after the COLUMN definition, is the index definition:

```
CREATE TABLE table_name
(
  column_name1 TYPE1,
  column_name2 TYPE2,
  column_name3 TYPE3,
  INDEX idx_name1(column_name1) USING INVERTED [PROPERTIES(...)] [COMMENT 'your comment'],
  INDEX idx_name2(column_name2) USING INVERTED [PROPERTIES(...)] [COMMENT 'your comment']
)
table_properties;
```

Syntax explanation:

1. `idx_column_name(column_name)` **is mandatory,** `column_name` **is the name of the column for the index, must be a column defined earlier,** `idx_column_name` **is the index name, must be unique at the table level, recommended naming convention: prefix** `idx_` **before the column name**

2. `USING INVERTED` **is mandatory to specify that the index type is an inverted index**

3. `PROPERTIES` **is optional to specify additional properties of the inverted index, currently supported properties are:**

parser: specifies the tokenizer

- By default, it is unspecified, meaning no tokenization
    - `english`: English tokenization, suitable for columns with English text, uses spaces and punctuation for tokenization, high performance
        - `chinese`: Chinese tokenization, suitable for columns with mainly Chinese text, lower performance than English tokenization
            - `unicode`: Unicode tokenization, suitable for mixed Chinese and English, and mixed multilingual texts. It can tokenize email prefixes and suffixes, IP addresses, and mixed character and number strings, and can tokenize Chinese by characters.

  Tokenization results can be verified using the `TOKENIZE` SQL function, see the following sections for details.

parser_mode

Specifies the tokenization mode, currently supported modes for `parser = chinese` **are:**

- fine_grained: fine-grained mode, tends to generate shorter, more words, e.g., '武汉市长江大桥' will be tokenized into '武汉', '武汉市', '市长', '长江', '长江大桥', '大桥'
    - coarse_grained: coarse-grained mode, tends to generate longer, fewer words, e.g., '武汉市长江大桥' will be tokenized into '武汉市', '长江大桥'
        - default coarse_grained

support_phrase

Specifies whether the index supports MATCH_PHRASE phrase query acceleration

- true: supported, but the index requires more storage space
    - false: not supported, more storage efficient, can use MATCH_ALL to query multiple keywords
        - From versions 2.0.14, 2.1.5 and 3.0.1, the default is true if parser is set. Otherwise default to false.

  For example, the following example specifies Chinese tokenization, coarse-grained mode, and supports phrase query acceleration.

```
 INDEX idx_name(column_name) USING INVERTED PROPERTIES("parser" = "chinese", "parser_mode" = "coarse_grained",
"support_phrase" = "true")
```

char_filter

Specifies preprocessing the text before tokenization, usually to affect tokenization behavior

char_filter_type: specifies different functional char_filters (currently only supports char_replace)

char_replace replaces each char in the pattern with a char in the replacement

- char_filter_pattern: characters to be replaced
    - char_filter_replacement: replacement character array, optional, defaults to a space character

  For example, the following example replaces dots and underscores with spaces, thus treating them as word separators, affecting tokenization behavior.

```
 INDEX idx_name(column_name) USING INVERTED PROPERTIES("parser" = "unicode", "char_filter_type" = "char_replace",
"char_filter_pattern" = "._", "char_filter_replacement" = " ")
```

  `

ignore_above

Specifies the length limit for non-tokenized string indexes (parser not specified)

- Strings longer than the length set by ignore_above will not be indexed. For string arrays, ignore_above applies to each array element separately, and elements longer than ignore_above will not be indexed.
    - Default is 256, unit is bytes

lower_case

Whether to convert tokens to lowercase for case-insensitive matching

- true: convert to lowercase
    - false: do not convert to lowercase
        - From versions 2.0.7, the default is true, automatically converting to lowercase. Earlier versions default to false.

stopwords

Specifying the stopword list to use, which will affect the behavior of the tokenizer

- The default built-in stopword list includes meaningless words such as ‘is,’ ‘the,’ ‘a,’ etc. When writing or querying, the tokenizer will ignore words that are in the stopword list.
    - none: Use an empty stopword list

4. `COMMENT` **is optional for specifying index comments**

Adding Inverted Indexes to Existing Tables

1. ADD INDEX

Supports both `CREATE INDEX` and `ALTER TABLE ADD INDEX` syntax. The parameters are the same as those used when defining indexes during table creation.

```
-- Syntax 1
CREATE INDEX idx_name ON table_name(column_name) USING INVERTED [PROPERTIES(...)] [COMMENT 'your comment'];
-- Syntax 2
ALTER TABLE table_name ADD INDEX idx_name(column_name) USING INVERTED [PROPERTIES(...)] [COMMENT 'your comment'];
```

2. BUILD INDEX

The `CREATE / ADD INDEX` operation only adds the index definition. New data written after this operation will generate inverted indexes, but existing data requires using `BUILD INDEX` to trigger indexing:

```
-- Syntax 1, by default, builds the index for all partitions in the table
BUILD INDEX index_name ON table_name;
-- Syntax 2, you can specify partitions, one or more
BUILD INDEX index_name ON table_name PARTITIONS(partition_name1, partition_name2);
```

To check the progress of `BUILD INDEX`, use `SHOW BUILD INDEX`:

```
SHOW BUILD INDEX [FROM db_name];
-- Example 1, view the progress of all BUILD INDEX tasks
SHOW BUILD INDEX;
-- Example 2, view the progress of BUILD INDEX tasks for a specific table
SHOW BUILD INDEX where TableName = "table1";
```

To cancel `BUILD INDEX`, use `CANCEL BUILD INDEX`:

```
CANCEL BUILD INDEX ON table_name;
CANCEL BUILD INDEX ON table_name (job_id1, job_id2, ...);
```

> `BUILD INDEX` **creates an asynchronous task executed by multiple threads on each BE. The number of threads can be set using the BE config** `alter_index_worker_count`**, with a default value of 3.**
>
> In versions before 2.0.12, `BUILD INDEX` would keep retrying until it succeeded. Starting from this version, failure and timeout mechanisms prevent endless retries.
>
> 1. If the majority of replicas for a tablet fail to `BUILD INDEX`, the entire `BUILD INDEX` operation fails.
> 2. If the time exceeds `alter_table_timeout_second`, the `BUILD INDEX` operation times out.
> 3. Users can trigger `BUILD INDEX` multiple times; indexes that have already been built successfully will not be rebuilt.

Deleting Inverted Indexes from Existing Tables

```
-- Syntax 1
DROP INDEX idx_name ON table_name;
-- Syntax 2
ALTER TABLE table_name DROP INDEX idx_name;
```

> `DROP INDEX` **deletes the index definition, so new data will no longer write to the index. This creates an asynchronous task to perform the index deletion, executed by multiple threads on each BE. The number of threads can be set using the BE parameter** `alter_index_worker_count`**, with a default value of 3.**

Viewing Inverted Index

– Syntax 1: The INDEX section in the table schema with USING INVERTED indicates an inverted index SHOW CREATE TABLE table_name;

– Syntax 2: IndexType as INVERTED indicates an inverted index SHOW INDEX FROM table_name;

Using Indexes

Accelerating Queries with Inverted Indexes

```
-- 1. Full-text search keyword matching using MATCH_ANY and MATCH_ALL
SELECT * FROM table_name WHERE column_name MATCH_ANY | MATCH_ALL 'keyword1 ...';

-- 1.1 Rows in the content column containing keyword1
SELECT * FROM table_name WHERE content MATCH_ANY 'keyword1';

-- 1.2 Rows in the content column containing keyword1 or keyword2; you can add more keywords
SELECT * FROM table_name WHERE content MATCH_ANY 'keyword1 keyword2';

-- 1.3 Rows in the content column containing both keyword1 and keyword2; you can add more keywords
SELECT * FROM table_name WHERE content MATCH_ALL 'keyword1 keyword2';
```

```
-- 2. Full-text search phrase matching using MATCH_PHRASE

-- 2.1 Rows in the content column containing both keyword1 and keyword2, where keyword2 must immediately follow keyword1
-- 'keyword1 keyword2', 'wordx keyword1 keyword2', 'wordx keyword1 keyword2 wordy' all match because they contain 'keyword1
keyword2' with keyword2 immediately following keyword1
-- 'keyword1 wordx keyword2' does not match because there is a word between keyword1 and keyword2
-- 'keyword2 keyword1' does not match because the order is reversed
SELECT * FROM table_name WHERE content MATCH_PHRASE 'keyword1 keyword2';

-- 2.2 Rows in the content column containing both keyword1 and keyword2, with a slop (maximum word distance) of 3
-- 'keyword1 keyword2', 'keyword1 a keyword2', 'keyword1 a b c keyword2' all match because the slop is 0, 1, and 3 respectively, all
within 3
-- 'keyword1 a b c d keyword2' does not match because the slop is 4, exceeding 3
-- 'keyword2 keyword1', 'keyword2 a keyword1', 'keyword2 a b c keyword1' also match because when slop > 0, the order of keyword1 and
keyword2 is not required. To enforce the order, Doris provides a + sign after slop
SELECT * FROM table_name WHERE content MATCH_PHRASE 'keyword1 keyword2 ~3';
-- To enforce order, use a positive sign with slop; 'keyword1 a b c keyword2' matches, while 'keyword2 a b c keyword1' does not
SELECT * FROM table_name WHERE content MATCH_PHRASE 'keyword1 keyword2 ~3+';

-- 2.3 Prefix matching the last word keyword2, with a default limit of 50 prefixes (controlled by session variable
inverted_index_max_expansions)
-- It is necessary to ensure that keyword1 and keyword2 remain adjacent in the original text after tokenization, with no other words
in between.
-- 'keyword1 keyword2abc' matches because keyword1 is identical and keyword2abc is a prefix of keyword2
-- 'keyword1 keyword2' also matches because keyword2 is a prefix of keyword2
-- 'keyword1 keyword3' does not match because keyword3 is not a prefix of keyword2
-- 'keyword1 keyword3abc' does not match because keyword3abc is not a prefix of keyword2
```

```
SELECT * FROM table_name WHERE content MATCH_PHRASE_PREFIX 'keyword1 keyword2';

-- 2.4 If only one word is provided, it defaults to a prefix query with a limit of 50 prefixes (controlled by session variable
inverted_index_max_expansions)
SELECT * FROM table_name WHERE content MATCH_PHRASE_PREFIX 'keyword1';

-- 2.5 Regular expression matching on tokenized words, with a default limit of 50 matches (controlled by session variable
inverted_index_max_expansions)
-- Similar to MATCH_PHRASE_PREFIX but with regex instead of prefix
SELECT * FROM table_name WHERE content MATCH_REGEXP 'key.*';
```

```
-- 3. Normal equality, range, IN, and NOT IN queries using standard SQL syntax, for example:
SELECT * FROM table_name WHERE id = 123;
SELECT * FROM table_name WHERE ts > '2023-01-01 00:00:00';
SELECT * FROM table_name WHERE op_type IN ('add', 'delete');
```

Analyzing Index Acceleration Effects Through Profiles

Inverted query acceleration can be toggled using the session variable `enable_inverted_index_query`, which is set to true by default. To verify the acceleration effect of the index, it can be set to false to turn it off.

The acceleration effect of the inverted index can be analyzed using the following metrics in the Query Profile: - RowsInvertedIndexFiltered: The number of rows filtered by the inverted index, which can be compared with other Rows values to analyze the filtering effect of the index. - InvertedIndexFilterTime: The time consumed by the inverted index. - InvertedIndexSearcherOpenTime: The time taken to open the inverted index. - InvertedIndexSearcherSearchTime: The time taken for internal queries within the inverted index.

Verifying Tokenization Effects Using Tokenization Functions

To check the actual effect of tokenization or to tokenize a piece of text, you can use the `TOKENIZE` function for verification.

The first parameter of the `TOKENIZE` function is the text to be tokenized, and the second parameter specifies the tokenization parameters used when creating the index.

mysql> SELECT TOKENIZE(‘I love Doris’,‘“parser”=“english”’); +————————————————————+ | tokenize(‘I love Doris’, ‘“parser”=“english”’) | +————————————————————+ | [“i”, “love”, “doris”] | +————————————————————+ 1 row in set (0.02 sec)

```
## Usage Example

Demonstrating the creation of an inverted index, full-text search, and regular queries using 1 million records from HackerNews. This
includes a simple performance comparison with queries without indexing.

### Table Creation

```sql
CREATE DATABASE test_inverted_index;

USE test_inverted_index;

-- Create a table with an inverted index on the comment field
--   USING INVERTED specifies the index type as an inverted index
--   PROPERTIES("parser" = "english") specifies using the "english" tokenizer; other options include "chinese" for Chinese
```
```

```
tokenization and "unicode" for mixed-language tokenization. If the "parser" parameter is not specified, no tokenization is applied.

CREATE TABLE hackernews_1m
(
    `id` BIGINT,
    `deleted` TINYINT,
    `type` String,
    `author` String,
    `timestamp` DateTimeV2,
    `comment` String,
    `dead` TINYINT,
    `parent` BIGINT,
    `poll` BIGINT,
    `children` Array<BIGINT>,
    `url` String,
    `score` INT,
    `title` String,
    `parts` Array<INT>,
    `descendants` INT,
    INDEX idx_comment (`comment`) USING INVERTED PROPERTIES("parser" = "english") COMMENT 'inverted index for comment'
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 10
PROPERTIES ("replication_num" = "1");
```

Data Import

Importing Data via Stream Load

```
wget https://qa-build.oss-cn-beijing.aliyuncs.com/regression/index/hacknernews_1m.csv.gz

curl --location-trusted -u root: -H "compress_type:gz" -T hacknernews_1m.csv.gz -XPUT
http://127.0.0.1:8030/api/test_inverted_index/hackernews_1m/_stream_load
{
    "TxnId": 2,
    "Label": "a8a3e802-2329-49e8-912b-04c800a461a6",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 1000000,
    "NumberLoadedRows": 1000000,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 130618406,
    "LoadTimeMs": 8988,
    "BeginTxnTimeMs": 23,
    "StreamLoadPutTimeMs": 113,
    "ReadDataTimeMs": 4788,
    "WriteDataTimeMs": 8811,
    "CommitAndPublishTimeMs": 38
}
```

Confirm Data Import Success with SQL count()

```
mysql> SELECT count() FROM hackernews_1m;
+---------+
| count() |
+---------+
| 1000000 |
```

```
+---------+
1 row in set (0.02 sec)
```

Queries

01 Full-Text Search

- Using `LIKE` to match and count rows containing 'OLAP' in the `comment` column took 0.18s.

```
mysql> SELECT count() FROM hackernews_1m WHERE comment LIKE '%OLAP%';
+---------+
| count() |
+---------+
|      34 |
+---------+
1 row in set (0.18 sec)
```

- Using full-text search with `MATCH_ANY` based on the inverted index to count rows containing 'OLAP' in the `comment` column took 0.02s, resulting in a 9x speedup. The performance improvement would be even more significant on larger datasets.

  The difference in the number of results is due to the inverted index normalizing the terms by converting them to lowercase, among other processes, hence `MATCH_ANY` yields more results than `LIKE`.

```
mysql> SELECT count() FROM hackernews_1m WHERE comment MATCH_ANY 'OLAP';
+---------+
| count() |
+---------+
|      35 |
+---------+
1 row in set (0.02 sec)
```

- Similarly, comparing the performance for counting occurrences of 'OLTP', 0.07s vs 0.01s. Due to caching, both `LIKE` and `MATCH_ANY` improved, but the inverted index still provided a 7x speedup.

```
mysql> SELECT count() FROM hackernews_1m WHERE comment LIKE '%OLTP%';
+---------+
| count() |
+---------+
|      48 |
+---------+
1 row in set (0.07 sec)

mysql> SELECT count() FROM hackernews_1m WHERE comment MATCH_ANY 'OLTP';
+---------+
| count() |
+---------+
|      51 |
+---------+
1 row in set (0.01 sec)
```

- Counting rows where both 'OLAP' and 'OLTP' appear took 0.13s vs 0.01s, a 13x speedup.

  To require multiple terms to appear simultaneously (AND relationship), use `MATCH_ALL 'keyword1 keyword2 ...'`.

```
mysql> SELECT count() FROM hackernews_1m WHERE comment LIKE '%OLAP%' AND comment LIKE '%OLTP%';
+---------+
| count() |
+---------+
|      14 |
+---------+
1 row in set (0.13 sec)

mysql> SELECT count() FROM hackernews_1m WHERE comment MATCH_ALL 'OLAP OLTP';
+---------+
| count() |
+---------+
|      15 |
+---------+
1 row in set (0.01 sec)
```

- Counting rows where either ‘OLAP’ or ‘OLTP’ appears took 0.12s vs 0.01s, a 12x speedup.

  To require any one or more of multiple terms to appear (OR relationship), use `MATCH_ANY 'keyword1 keyword2 ...'`.

```
mysql> SELECT count() FROM hackernews_1m WHERE comment LIKE '%OLAP%' OR comment LIKE '%OLTP%';
+---------+
| count() |
+---------+
|      68 |
+---------+
1 row in set (0.12 sec)

mysql> SELECT count() FROM hackernews_1m WHERE comment MATCH_ANY 'OLAP OLTP';
+---------+
| count() |
+---------+
|      71 |
+---------+
1 row in set (0.01 sec)
```

02 Standard Equality and Range Queries

- Range query on a `DateTime` type column

```
mysql> SELECT count() FROM hackernews_1m WHERE timestamp > '2007-08-23 04:17:00';
+---------+
| count() |
+---------+
|  999081 |
+---------+
1 row in set (0.03 sec)
```

- Adding an inverted index for the `timestamp` column

```
-- For date-time types, USING INVERTED does not require specifying a parser
-- CREATE INDEX is one syntax for creating an index, another method will be shown later
mysql> CREATE INDEX idx_timestamp ON hackernews_1m(timestamp) USING INVERTED;
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> BUILD INDEX idx_timestamp ON hackernews_1m;
Query OK, 0 rows affected (0.01 sec)
```

- Checking the index creation progress. From the difference between `FinishTime` and `CreateTime`, we can see that building the inverted index for 1 million rows on the `timestamp` column took only 1 second.

```
mysql> SHOW ALTER TABLE COLUMN;
+-------+--------------+-------------------------+-------------------------+--------------+---------+--------------+----
-----------+--------------+----------+------+----------+---------+
| JobId | TableName     | CreateTime              | FinishTime              | IndexName     | IndexId | OriginIndexId |
SchemaVersion | TransactionId | State    | Msg  | Progress | Timeout |
+-------+--------------+-------------------------+-------------------------+--------------+---------+--------------+----
-----------+--------------+----------+------+----------+---------+
| 10030 | hackernews_1m | 2023-02-10 19:44:12.929 | 2023-02-10 19:44:13.938 | hackernews_1m | 10031   | 10008         |
1:1994690496  | 3             | FINISHED |      | NULL     | 2592000 |
+-------+--------------+-------------------------+-------------------------+--------------+---------+--------------+----
-----------+--------------+----------+------+----------+---------+
1 row in set (0.00 sec)
```

```
-- If the table has no partitions, PartitionName defaults to TableName
mysql> SHOW BUILD INDEX;
+-------+--------------+--------------+-----------------------------------------------------------------+-----------------------
--+-------------------------+--------------+----------+------+----------+
| JobId | TableName     | PartitionName | AlterInvertedIndexes                                    | CreateTime
| FinishTime              | TransactionId | State    | Msg  | Progress |
+-------+--------------+--------------+-----------------------------------------------------------------+-----------------------
--+-------------------------+--------------+----------+------+----------+
| 10191 | hackernews_1m | hackernews_1m | [ADD INDEX idx_timestamp (`timestamp`) USING INVERTED],  | 2023-06-26
15:32:33.894 | 2023-06-26 15:32:34.847 | 3             | FINISHED |      | NULL     |
+-------+--------------+--------------+-----------------------------------------------------------------+-----------------------
--+-------------------------+--------------+----------+------+----------+
1 row in set (0.04 sec)
```

- After the index is created, range queries use the same query syntax. Doris will automatically recognize the index for optimization. However, due to the small dataset, the performance difference is not significant.

```
mysql> SELECT count() FROM hackernews_1m WHERE timestamp > '2007-08-23 04:17:00';
+---------+
| count() |
+---------+
|  999081 |
+---------+
1 row in set (0.01 sec)
```

- Performing similar operations on a numeric column `parent` with an equality match query.

```
mysql> SELECT count() FROM hackernews_1m WHERE parent = 11189;
+---------+
| count() |
+---------+
|       2 |
+---------+
1 row in set (0.01 sec)

-- For numeric types, USING INVERTED does not require specifying a parser
-- ALTER TABLE t ADD INDEX is the second syntax for creating an index
```

```
mysql> ALTER TABLE hackernews_1m ADD INDEX idx_parent(parent) USING INVERTED;
Query OK, 0 rows affected (0.01 sec)

-- Execute BUILD INDEX to create the inverted index for existing data
mysql> BUILD INDEX idx_parent ON hackernews_1m;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW ALTER TABLE COLUMN;
+-------+--------------+------------------------+------------------------+--------------+---------+--------------+----
-----------+--------------+----------+------+----------+---------+
| JobId | TableName    | CreateTime             | FinishTime             | IndexName    | IndexId | OriginIndexId |
SchemaVersion | TransactionId | State    | Msg  | Progress | Timeout |
+-------+--------------+------------------------+------------------------+--------------+---------+--------------+----
-----------+--------------+----------+------+----------+---------+
| 10030 | hackernews_1m | 2023-02-10 19:44:12.929 | 2023-02-10 19:44:13.938 | hackernews_1m | 10031   | 10008         |
1:1994690496  | 3             | FINISHED |      | NULL     | 2592000 |
| 10053 | hackernews_1m | 2023-02-10 19:49:32.893 | 2023-02-10 19:49:33.982 | hackernews_1m | 10054   | 10008         |
1:378856428   | 4             | FINISHED |      | NULL     | 2592000 |
+-------+--------------+------------------------+------------------------+--------------+---------+--------------+----
-----------+--------------+----------+------+----------+---------+

mysql> SHOW BUILD INDEX;
+-------+--------------+--------------+-----------------------------------------------------+-------------------------+---
----------------------+--------------+----------+------+----------+
| JobId | TableName    | PartitionName | AlterInvertedIndexes                               | CreateTime              |
FinishTime              | TransactionId | State    | Msg  | Progress |
+-------+--------------+--------------+-----------------------------------------------------+-------------------------+---
----------------------+--------------+----------+------+----------+
| 11005 | hackernews_1m | hackernews_1m | [ADD INDEX idx_parent (`parent`) USING INVERTED],  | 2023-06-26 16:25:10.167 |
2023-06-26 16:25:10.838 | 1002          | FINISHED |      | NULL     |
+-------+--------------+--------------+-----------------------------------------------------+-------------------------+---
----------------------+--------------+----------+------+----------+
1 row in set (0.01 sec)

mysql> SELECT count() FROM hackernews_1m WHERE parent = 11189;
+---------+
| count() |
+---------+
|       2 |
+---------+
1 row in set (0.01 sec)
```

- Creating an inverted index for the string column `author` without tokenization. Equality queries can also leverage the index for speedup.

  ```sql mysql> SELECT count() FROM hackernews_1m WHERE author = ‘faster’; +———+ | count() | +———+ | 20 | +———+ 1 row in set (0.03 sec)

  – Here, USING INVERTED is used without tokenizing the `author` column, treating it as a single term mysql> ALTER TABLE hackernews_1m ADD INDEX idx_author(author) USING INVERTED; Query OK, 0 rows affected (0.01 sec)

  – Execute BUILD INDEX to add the inverted index for existing data mysql> BUILD INDEX idx_author ON hackernews_1m; Query OK, 0 rows affected (0.01 sec)

Creating an incremental index for 1 million author records took only 1.5 seconds.

```
mysql> SHOW ALTER TABLE COLUMN;
+-------+--------------+------------------------+------------------------+--------------+---------+--------------+-------------
--+--------------+----------+------+----------+---------+
```
```

```
| JobId | TableName     | CreateTime              | FinishTime              | IndexName     | IndexId | OriginIndexId |
SchemaVersion | TransactionId | State    | Msg  | Progress | Timeout |
+-------+---------------+-------------------------+-------------------------+---------------+---------+---------------+-------------
--+---------------+----------+------+----------+---------+
| 10030 | hackernews_1m | 2023-02-10 19:44:12.929 | 2023-02-10 19:44:13.938 | hackernews_1m | 10031   | 10008         | 1:1994690496
| 3             | FINISHED |      | NULL     | 2592000 |
| 10053 | hackernews_1m | 2023-02-10 19:49:32.893 | 2023-02-10 19:49:33.982 | hackernews_1m | 10054   | 10008         | 1:378856428
| 4             | FINISHED |      | NULL     | 2592000 |
| 10076 | hackernews_1m | 2023-02-10 19:54:20.046 | 2023-02-10 19:54:21.521 | hackernews_1m | 10077   | 10008         | 1:1335127701
| 5             | FINISHED |      | NULL     | 2592000 |
+-------+---------------+-------------------------+-------------------------+---------------+---------+---------------+-------------
--+---------------+----------+------+----------+---------+
```

```
mysql> SHOW BUILD INDEX ORDER BY CreateTime DESC LIMIT 1;
+-------+---------------+---------------+-----------------------------------------------------+-------------------------+------------
-------------+---------------+----------+------+----------+
| JobId | TableName     | PartitionName | AlterInvertedIndexes                                | CreateTime              | FinishTime
| TransactionId | State    | Msg  | Progress |
+-------+---------------+---------------+-----------------------------------------------------+-------------------------+------------
-------------+---------------+----------+------+----------+
| 13006 | hackernews_1m | hackernews_1m | [ADD INDEX idx_author (`author`) USING INVERTED],  | 2023-06-26 17:23:02.610 | 2023-06-26
17:23:03.755 | 3004          | FINISHED |      | NULL     |
+-------+---------------+---------------+-----------------------------------------------------+-------------------------+------------
-------------+---------------+----------+------+----------+
1 row in set (0.01 sec)
```

– After creating the index, string equality matches also showed significant acceleration.

```
mysql> SELECT count() FROM hackernews_1m WHERE author = 'faster';
+---------+
| count() |
+---------+
|      20 |
+---------+
1 row in set (0.01 sec)
```

BloomFilter Index

Indexing Principles

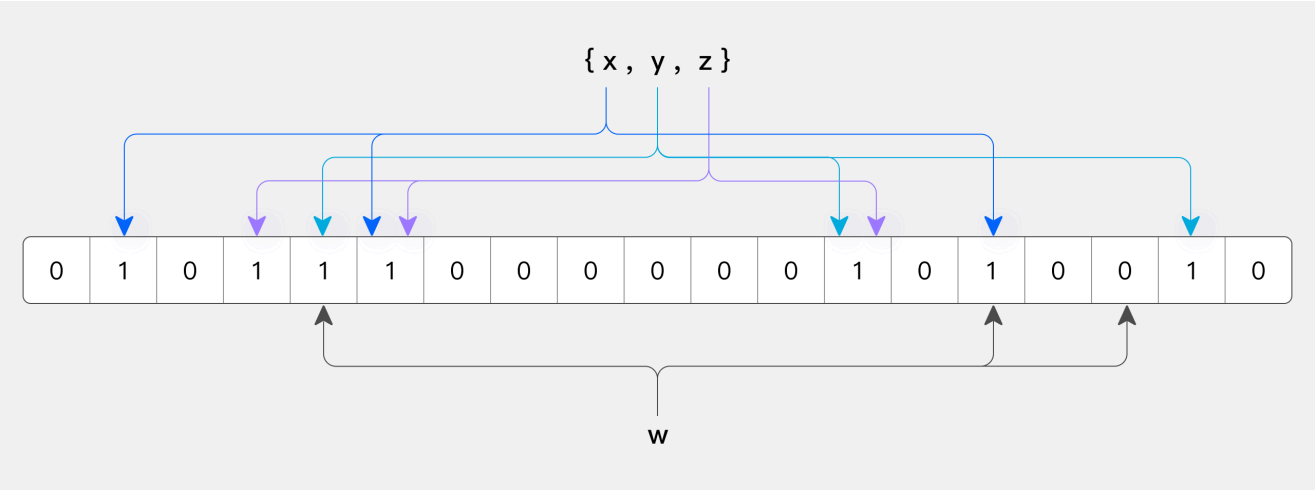
The BloomFilter index is a type of skip index based on BloomFilter. Its principle is to use BloomFilter to skip data blocks that do not meet the specified conditions for equality queries, thereby reducing IO and accelerating queries.

BloomFilter is a fast lookup algorithm proposed by Bloom in 1970, which uses multiple hash functions. It is commonly used in scenarios where quick determination of whether an element belongs to a set is needed without requiring 100% accuracy. BloomFilter has the following characteristics:

- A space-efficient probabilistic data structure used to check if an element is in a set.
- For a membership check, BloomFilter returns one of two results: possibly in the set or definitely not in the set.

A BloomFilter consists of a very long binary bit array and a series of hash functions. The bit array is initially all set to 0. When an element is to be checked, it is hashed by a series of hash functions to generate a series of values, and the bits at these positions in the array are set to 1.

The figure below shows an example of a BloomFilter with m=18 and k=3 (where m is the size of the bit array and k is the number of hash functions). Elements x, y, and z in the set are hashed by 3 different hash functions into the bit array. When querying element w, if any bit calculated by the hash functions is 0, then w is not in the set. Conversely, if all bits are 1, it only indicates that w may be in the set, but not definitely, due to possible hash collisions.



Bloom_filter.svg

Thus, if all bits at the calculated positions are 1, it only indicates that the element may be in the set, not definitely, due to possible hash collisions. This is the “false positive” nature of BloomFilter. Therefore, a BloomFilter-based index can only skip data that does not meet the conditions but cannot precisely locate data that does.

The Doris BloomFilter index is built on a per-page basis, with each data block storing a BloomFilter. During writing, each value in the data block is hashed into the corresponding BloomFilter. During querying, for equality

conditions, it is checked if the BloomFilter of each data block contains the value. If not, the data block is skipped, reducing IO and speeding up the query.

Use Cases

BloomFilter indexes can accelerate equality queries (including = and IN) and are effective for high cardinality fields, such as unique id fields like userid.

> **BloomFilter has the following limitations:**
>
> 1. It has no effect on queries other than in and =, such as !=, NOT IN, >, <, etc.
> 2. It does not support BloomFilter indexing on columns of type Tinyint, Float, Double.
> 3. It has limited acceleration effect on low cardinality fields. For example, a “gender” field with only two values will likely be included in almost every data block, making the BloomFilter index meaningless.
>
> To check the effect of a BloomFilter index on a query, you can analyze relevant metrics in the Query Profile.
>
> - BlockConditionsFilteredBloomFilterTime is the time consumed by the BloomFilter index.
> - RowsBloomFilterFiltered is the number of rows filtered out by the BloomFilter. You can compare it with other Rows values to analyze the filtering effect of the BloomFilter index.

Managing Indexes

Creating a BloomFilter Index When Creating a Table

Due to historical reasons, the syntax for defining BloomFilter indexes differs from the general INDEX syntax used for inverted indexes. BloomFilter indexes are specified in the table’s PROPERTIES using “bloom_filter_columns”, which can specify one or more fields.

```
PROPERTIES (
"bloom_filter_columns" = "column_name1,column_name2"
);
```

Viewing BloomFilter Indexes

```
SHOW CREATE TABLE table_name;
```

Adding or Removing BloomFilter Indexes on an Existing Table

Modify the table’s bloom_filter_columns property via ALTER TABLE to add or remove BloomFilter indexes.

Add a BloomFilter index for column_name3

```
ALTER TABLE table_name SET ("bloom_filter_columns" = "column_name1,column_name2,column_name3");
```

Remove the BloomFilter index for column_name1

```
ALTER TABLE table_name SET ("bloom_filter_columns" = "column_name2,column_name3");
```

Using Indexes

BloomFilter index is used to accelerate equality queries in the WHERE clause. It automatically takes effect when applicable, and there is no special syntax required.

The acceleration effect of the BloomFilter index can be analyzed using the following metrics in the Query Profile: - RowsBloomFilterFiltered: The number of rows filtered by the BloomFilter index, which can be compared with other Rows values to analyze the filtering effect of the index. - BlockConditionsFilteredBloomFilterTime: The time consumed by the BloomFilter inverted index.

Usage Example

Here is an example of how to create a BloomFilter index in Doris.

The BloomFilter index in Doris is created by adding the “bloom_filter_columns” property in the CREATE TABLE statement, with k1, k2, k3 being the key columns for the BloomFilter index. For example, the following creates BloomFilter indexes on saler_id and category_id.

```
CREATE TABLE IF NOT EXISTS sale_detail_bloom  (
    sale_date date NOT NULL COMMENT "Sale date",
    customer_id int NOT NULL COMMENT "Customer ID",
    saler_id int NOT NULL COMMENT "Salesperson",
    sku_id int NOT NULL COMMENT "Product ID",
    category_id int NOT NULL COMMENT "Product category",
    sale_count int NOT NULL COMMENT "Sales quantity",
    sale_price DECIMAL(12,2) NOT NULL COMMENT "Unit price",
    sale_amt DECIMAL(20,2)  COMMENT "Total sales amount"
)
DUPLICATE KEY(sale_date, customer_id, saler_id, sku_id, category_id)
DISTRIBUTED BY HASH(saler_id) BUCKETS 10
PROPERTIES (
"replication_num" = "1",
"bloom_filter_columns"="saler_id,category_id"
);
```

N-Gram BloomFilter Index

Indexing Principles

The NGram BloomFilter index, similar to the BloomFilter index, is a skip index based on BloomFilter.

Unlike the BloomFilter index, the NGram BloomFilter index is used to accelerate text LIKE queries. Instead of storing the original text values, it tokenizes the text using NGram and stores each token in the BloomFilter. For LIKE queries, the pattern in LIKE '%pattern%' is also tokenized using NGram. Each token is checked against the BloomFilter, and if any token is not found, the corresponding data block does not meet the LIKE condition and can be skipped, reducing IO and accelerating the query.

Use Cases

The NGram BloomFilter index can only accelerate string LIKE queries, and the number of consecutive characters in the LIKE pattern must be greater than or equal to the N defined in the NGram index.

> **- NGram BloomFilter only supports string columns and can only accelerate LIKE queries.**
>
> - NGram BloomFilter index and BloomFilter index are mutually exclusive, meaning a column can only have one or the other.
> - The performance analysis of the NGram BloomFilter index is similar to that of the BloomFilter index.

Managing Indexes

Creating an NGram BloomFilter Index

The index definition follows the COLUMN definition in the CREATE TABLE statement:

```
INDEX `idx_column_name` (`column_name`) USING NGRAM_BF PROPERTIES("gram_size"="3", "bf_size"="1024") COMMENT 'username ngram_bf
index'
```

Explanation of the syntax:

1. `idx_column_name(column_name)` is mandatory. `column_name` is the column to be indexed and must appear in the column definitions above. `idx_column_name` is the index name, which must be unique at the table level. It is recommended to name it with a prefix `idx_` followed by the column name.
2. `USING NGRAM_BF` is mandatory and specifies that the index type is an NGram BloomFilter index.
3. `PROPERTIES` is optional and is used to specify additional properties for the NGram BloomFilter index. The supported properties are:

- **gram_size**: The N in NGram, specifying the number of consecutive characters to form a token. For example, ‘an ngram example’ with N = 3 would be tokenized into ‘an’, ‘n n’, ’ ng’, ‘ngr’, ‘gra’, ‘ram’ (6 tokens).
- **bf_size**: The size of the BloomFilter in bits. bf_size determines the size of the index corresponding to each data block. The larger this value, the more storage space it occupies, but the lower the probability of hash collisions.

It is recommended to set **gram_size** to the minimum length of the string in LIKE queries but not less than 2. Generally, “gram_size”=“3”, “bf_size”=“1024” is recommended, then adjust based on the Query Profile.

4. `COMMENT` is optional and specifies an index comment.

Viewing NGram BloomFilter Index

– Syntax 1: The INDEX section in the table schema with USING NGRAM_BF indicates an inverted index

```
SHOW CREATE TABLE table_name;
```

– Syntax 2: IndexType as NGRAM_BF indicates an inverted index

```
SHOW INDEX FROM table_name;
```

Deleting an NGram BloomFilter Index

```
ALTER TABLE table_ngrambf DROP INDEX idx_ngrambf;
```

Modifying an NGram BloomFilter Index

```
CREATE INDEX idx_column_name2(column_name2) ON table_ngrambf USING NGRAM_BF PROPERTIES("gram_size"="3", "bf_size"="1024") COMMENT
'username ngram_bf index';

ALTER TABLE table_ngrambf ADD INDEX idx_column_name2(column_name2) USING NGRAM_BF PROPERTIES("gram_size"="3", "bf_size"="1024")
COMMENT 'username ngram_bf index';
```

Using Indexes

To use NGram BloomFilter index, you need to set the following parameters (enable_function_pushdown is false by default):

```
SET enable_function_pushdown = true;
```

NGram BloomFilter index is used to accelerate LIKE queries, for example: SELECT count() FROM table1 WHERE message LIKE ‘%error%’;

The acceleration effect of the BloomFilter index (including NGram) can be analyzed using the following metrics in the Query Profile: - RowsBloomFilterFiltered: The number of rows filtered by the BloomFilter index, which

can be compared with other Rows values to analyze the filtering effect of the index. - BlockConditionsFilteredBloomFilterTime: The time consumed by the BloomFilter inverted index.

Usage Example

This section demonstrates the usage and effectiveness of the NGram BloomFilter index using a dataset of Amazon product reviews, `amazon_reviews`.

Table Creation

```
CREATE TABLE `amazon_reviews` (
  `review_date` int(11) NULL,
  `marketplace` varchar(20) NULL,
  `customer_id` bigint(20) NULL,
  `review_id` varchar(40) NULL,
  `product_id` varchar(10) NULL,
  `product_parent` bigint(20) NULL,
  `product_title` varchar(500) NULL,
  `product_category` varchar(50) NULL,
  `star_rating` smallint(6) NULL,
  `helpful_votes` int(11) NULL,
  `total_votes` int(11) NULL,
  `vine` boolean NULL,
  `verified_purchase` boolean NULL,
  `review_headline` varchar(500) NULL,
  `review_body` string NULL
) ENGINE=OLAP
DUPLICATE KEY(`review_date`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`review_date`) BUCKETS 16
PROPERTIES (
  "replication_allocation" = "tag.location.default: 1",
  "compression" = "ZSTD"
);
```

Data Import

Download the dataset using wget or other tools from the following URLs:

```
https://datasets-documentation.s3.eu-west-3.amazonaws.com/amazon_reviews/amazon_reviews_2010.snappy.parquet
https://datasets-documentation.s3.eu-west-3.amazonaws.com/amazon_reviews/amazon_reviews_2011.snappy.parquet
https://datasets-documentation.s3.eu-west-3.amazonaws.com/amazon_reviews/amazon_reviews_2012.snappy.parquet
https://datasets-documentation.s3.eu-west-3.amazonaws.com/amazon_reviews/amazon_reviews_2013.snappy.parquet
https://datasets-documentation.s3.eu-west-3.amazonaws.com/amazon_reviews/amazon_reviews_2014.snappy.parquet
https://datasets-documentation.s3.eu-west-3.amazonaws.com/amazon_reviews/amazon_reviews_2015.snappy.parquet
```

Import the data using stream load:

```
curl --location-trusted -u root: -T amazon_reviews_2010.snappy.parquet -H "format:parquet"
http://127.0.0.1:8030/api/${DB}/amazon_reviews/_stream_load
curl --location-trusted -u root: -T amazon_reviews_2011.snappy.parquet -H "format:parquet"
http://127.0.0.1:8030/api/${DB}/amazon_reviews/_stream_load
curl --location-trusted -u root: -T amazon_reviews_2012.snappy.parquet -H "format:parquet"
http://127.0.0.1:8030/api/${DB}/amazon_reviews/_stream_load
curl --location-trusted -u root: -T amazon_reviews_2013.snappy.parquet -H "format:parquet"
http://127.0.0.1:8030/api/${DB}/amazon_reviews/_stream_load
```

```
curl --location-trusted -u root: -T amazon_reviews_2014.snappy.parquet -H "format:parquet"
http://127.0.0.1:8030/api/${DB}/amazon_reviews/_stream_load
curl --location-trusted -u root: -T amazon_reviews_2015.snappy.parquet -H "format:parquet"
http://127.0.0.1:8030/api/${DB}/amazon_reviews/_stream_load
```

> **The data file may exceed 10 GB, and you may need to adjust the streaming_road_max_mb in be.conf to prevent exceeding the upload size limit of the stream load. You can dynamically adjust it by following the steps below:**
>
> ```
> curl -X POST http://{be_ip}:{be_http_port}/api/update_config?streaming_load_max_mb=32768
> ```
>
> Every BE needs to execute the above command.

Run a count query to confirm successful data import:

```
mysql> SELECT COUNT(*) FROM amazon_reviews;
+-----------+
| count(*)  |
+-----------+
| 135589433 |
+-----------+
```

Querying

First, run the query without any index. The WHERE clause contains a LIKE condition, and the query takes 7.60 seconds:

```
SELECT
    product_id,
    any(product_title),
    AVG(star_rating) AS rating,
    COUNT(*) AS count
FROM
    amazon_reviews
WHERE
    review_body LIKE '%is super awesome%'
GROUP BY
    product_id
ORDER BY
    count DESC,
    rating DESC,
    product_id
LIMIT 5;
```

Results:

```
+------------+----------------------------------------+--------------------+-------+
| product_id | any_value(product_title)               | rating             | count |
+------------+----------------------------------------+--------------------+-------+
| B00992CF6W | Minecraft                              | 4.8235294117647056 |    17 |
| B009UX2YAC | Subway Surfers                         | 4.7777777777777777 |     9 |
| B00DJFIMW6 | Minion Rush: Despicable Me Official Game |              4.875 |     8 |
| B00867O0CM | Temple Run                             |                  5 |     6 |
```

```
| B00KWVZ75O | Angry Birds Epic RPG                    |                 5 |     6 |
+------------+-----------------------------------------+-------------------+-------+
5 rows in set (7.60 sec)
```

Next, add an NGram BloomFilter index and run the same query again. The query takes 0.93 seconds, an 8x performance improvement:

```
ALTER TABLE amazon_reviews ADD INDEX review_body_ngram_idx(review_body) USING NGRAM_BF PROPERTIES("gram_size"="10",
"bf_size"="10240");
```

```
+------------+-----------------------------------------+-------------------+-------+
| product_id | any_value(product_title)                | rating            | count |
+------------+-----------------------------------------+-------------------+-------+
| B00992CF6W | Minecraft                               | 4.8235294117647056 |    17 |
| B009UX2YAC | Subway Surfers                          | 4.7777777777777777 |     9 |
| B00DJFIMW6 | Minion Rush: Despicable Me Official Game |             4.875 |     8 |
| B00867O0CM | Temple Run                              |                 5 |     6 |
| B00KWVZ75O | Angry Birds Epic RPG                    |                 5 |     6 |
+------------+-----------------------------------------+-------------------+-------+
5 rows in set (0.93 sec)
```

Best Practices

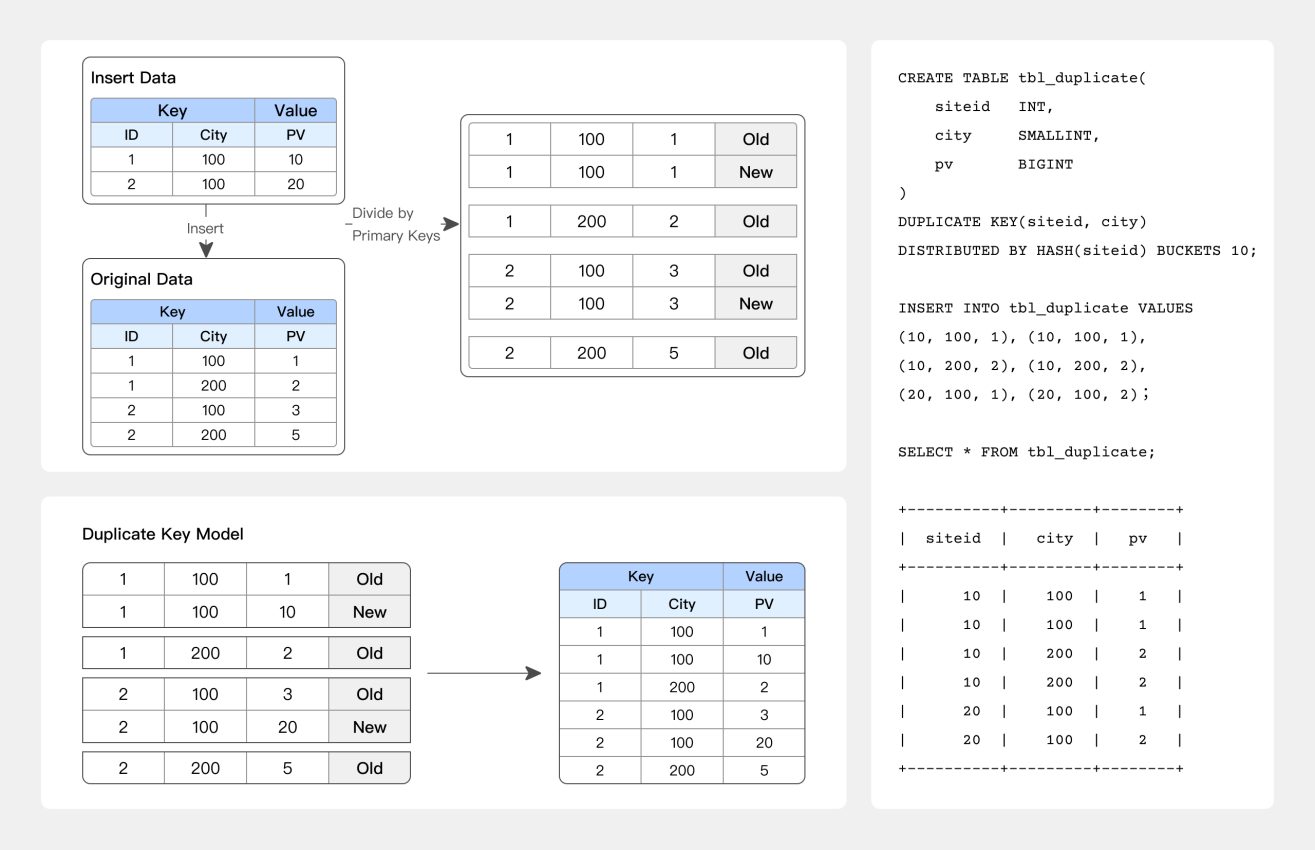
Best Practices

Data models

> Doris arranges data in three models: DUPLICATE KEY model, UNIQUE KEY model, and AGGREGATE KEY model.

> ****Recommendations****
>
> Because the data model is determined and **immutable** during table creation, it is important to select the most suitable data model.
>
> 1. The Duplicate Key model is a good fit for ad-hoc queries across any dimensions. It cannot leverage the benefits of pre-aggregation, but it is also not constrained by the limitations of an aggregation model, so it can take advantage of columnar storage (only reading the relevant columns without the need to read all key columns).
> 2. The Aggregate Key model, through pre-aggregation, can largely reduce the amount of data scanned and the computational workload for aggregate queries. It is particularly suitable for reporting queries with fixed patterns. However, this model is not friendly for count(*) queries. Additionally, since the aggregation method on the Value columns are fixed, users should pay extra attention to semantic correctness when performing other types of aggregate queries.
> 3. The Unique Key model is designed for scenarios that require unique primary key constraints. It can ensure the uniqueness of primary keys. The downside is that it cannot reap the benefits brought by materialization and pre-aggregation. For users with high-performance requirements for aggregate queries, it is recommended to use the Merge-on-Write feature of the Unique Key model since Doris 1.2.
> 4. Users with partial column update requirements might select from the following data models:
>   1. Unique Key model (Merge-on-Write mode)
>   2. Aggregate Key model (aggregated by REPLACE_IF_NOT_NULL)

DUPLICATE KEY model



duplicate-key-model-example

When only the sorting columns are specified, rows with the same key will not be merged.

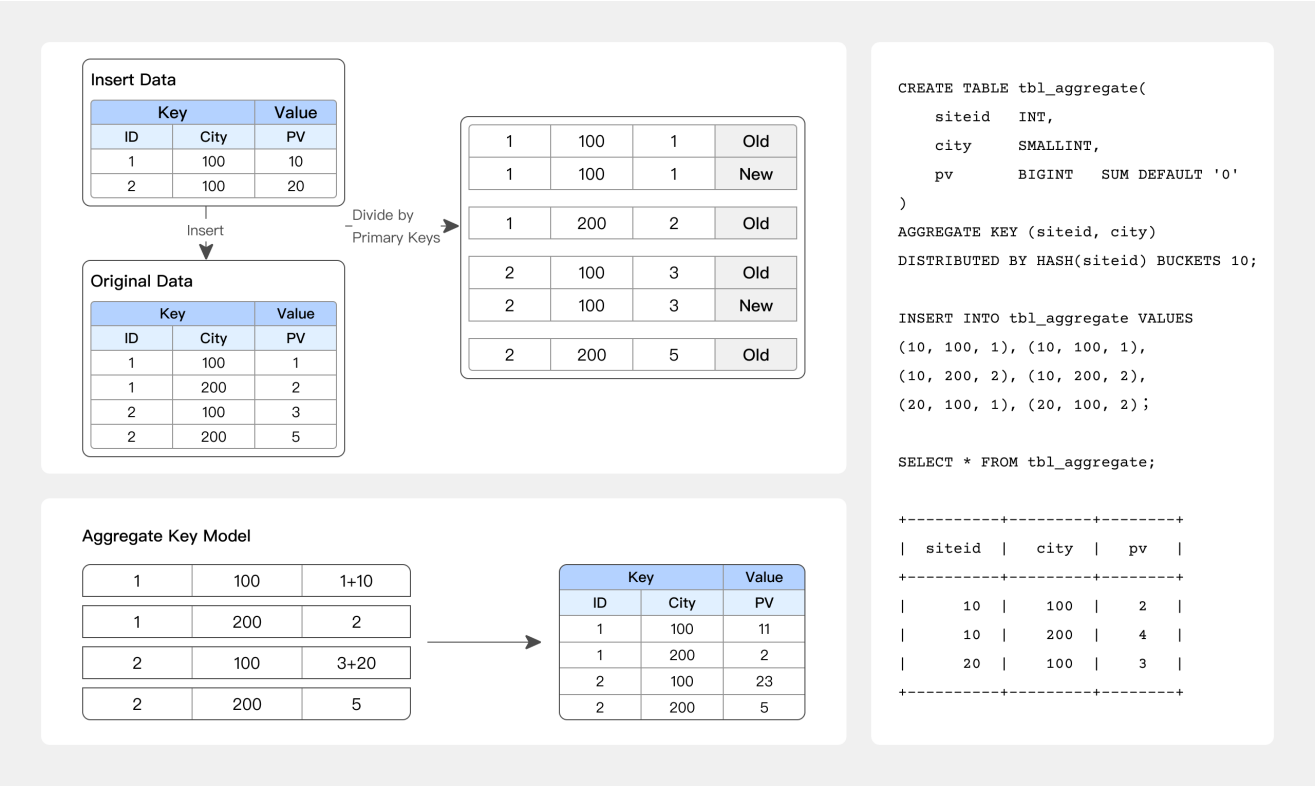
This is applicable to analytical business scenarios where data does not require pre-aggregation:

- Analysis of raw data
- Analysis of log or time series data where only new data is appended.

Best practice

```
-- For example, log analysis that allows only appending new data with replicated KEYs.
CREATE TABLE session_data
(
    visitorid   SMALLINT,
    sessionid   BIGINT,
    visittime   DATETIME,
    city        CHAR(20),
    province    CHAR(20),
    ip          varchar(32),
    brower      CHAR(20),
    url         VARCHAR(1024)
)
DUPLICATE KEY(visitorid, sessionid) -- Used solely for specifying sorting columns, rows with the same KEY will not be merged.
DISTRIBUTED BY HASH(sessionid, visitorid) BUCKETS 10;
```

AGGREGATE KEY model



aggregate-key-model-example

Old and new records with the same AGGREGATE KEY will be aggregated. The currently supported aggregation methods are as follows:

1. SUM: calculates the sum by accumulating the values of multiple rows;
2. REPLACE: replaces the value in the previously imported rows with the value from the next batch of data;
3. MAX: retains the maximum value;
4. MIN: retains the minimum value;
5. REPLACE_IF_NOT_NULL: replaces non-null values. Unlike REPLACE, it does not replace null values;
6. HLL_UNION: aggregates columns of HLL type using the HyperLogLog algorithm;
7. BITMAP_UNION: aggregates columns of BITMAP type using bitmap union aggregation;

This is suitable for reporting and multi-dimensional analysis scenarios such as:

- Website traffic analysis
- Multi-dimensional analysis of data reports.

Best practice

```
-- Example of website traffic analysis
CREATE TABLE site_visit
(
    siteid      INT,
    city        SMALLINT,
    username    VARCHAR(32),
    pv BIGINT   SUM DEFAULT '0' -- PV caculation
)
AGGREGATE KEY(siteid, city, username) -- Rows with the same KEY will be merged, and non-key columns will be aggregated based on the
```

```
specified aggregation function.
DISTRIBUTED BY HASH(siteid) BUCKETS 10;
```

UNIQUE KEY model

The new record will replace the old record that has the same UNIQUE KEY with it. Before Doris 1.2, the UNIQUE KEY model was implemented the same way as the REPLACE aggregation of the AGGREGATE KEY model. However, Since Doris 1.2, we have introduced the Merge-on-Write implementation for the UNIQUE KEY model, which provides better performance for aggregate queries.

This is suitable for analytical business scenarios that require updates, such as:

- Deduplicated order analysis
- Real-time synchronization of inserts, updates, and deletes.

Best practice

```
-- Example of deduplicated order analysis
CREATE TABLE sales_order
(
    orderid     BIGINT,
    status      TINYINT,
    username    VARCHAR(32),
    amount      BIGINT DEFAULT '0'
)
UNIQUE KEY(orderid) -- Rows of the same KEY will be merged
DISTRIBUTED BY HASH(orderid) BUCKETS 10;
```

Index

> Indexes can facilitate quick filtering and searching of data. Currently, Doris supports two types of indexes:
>
> 1. Built-in smart indexes, including prefix index and ZoneMap index.
> 2. User-created secondary indexes, including inverted index, BloomFilter index, Ngram BloomFilter index, and Bitmap index.

Prefix index

Prefix indexes are built-in indexes in the Aggregate, Unique, and Duplicate data models. The underlying data storage is sorted and stored based on the columns specified as the AGGREGATE KEY, UNIQUE KEY, or DUPLICATE KEY in their respective table creation statements. Prefix index, built on top of the sorted data, allows for quick data querying based on given prefix columns.

Prefix indexes are sparse indexes and cannot locate the exact row where a key is present. Instead, they can only roughly identify the range where the key may exist, and then use binary search algorithms to accurately locate the position of the key.

> ****Recommendations****

1. When creating a table, **the correct column order can greatly improve query efficiency**.
    1. Since the column order is specified during table creation, a table can only have one type of prefix index. However, this may not be efficient enough for queries based on the columns without prefix index. In such cases, users can adjust the column order by creating materialized views.
2. The first field in a prefix index should always be the field of the longest query conditions and should be a high-cardinality field.
    1. Bucketing field: It should have relatively even data distribution and be frequently used, preferably a high-cardinality field.
    2. Int(4) + Int(4) + varchar(50): The prefix index length is only 28.
    3. Int(4) + varchar(50) + Int(4): The prefix index length is only 24.
    4. varchar(10) + varchar(50): The prefix index length is only 30.
    5. Prefix index (36 characters): The first field delivers the best query performance. If a varchar field is encountered, the prefix index will automatically truncate it to the first 20 characters.
    6. If possible, include the most frequently used query fields in the prefix index. Otherwise, specify them as the bucketing fields.
3. The field lengths in the prefix index should be as explicit as possible because Doris can only utilize the prefix index for the first 36 bytes.
4. If it is difficult to design a partitioning, bucketing, and prefix index strategy for your data range, consider introducing inverted indexes for acceleration.

ZoneMap index

ZoneMap index is index information automatically maintained on a per-column basis in the columnar storage format. It includes information such as Min/Max values and the number of Null values. During data querying, the ZoneMap index is utilized to select the data range to scan based on the filtered fields using range conditions.

For example, when filtering the “age” field with the following query statement:

```
SELECT * FROM table WHERE age > 0 and age < 51;
```

If the Short Key Index is not hit, the ZoneMap index will be used to determine the data range, known as the “ordinary” range, that needs to be scanned based on the query conditions for the “age” field. This reduces the number of pages that need to be scanned.

Inverted index

Doris supports inverted indexes since version 2.0.0. Inverted index can be used for full-text searches on text data and range queries on regular numeric and date types. It enables fast filtering of rows that meet the conditions from massive amounts of data.

Best practice

```
-- Inverted index can be specified during table creation or added later. This is an example of specifying it during table creation:
CREATE TABLE table_name
(
  columns_difinition,
```

```
  INDEX idx_name1(column_name1) USING INVERTED [PROPERTIES("parser" = "english|unicode|chinese")] [COMMENT 'your comment']
  INDEX idx_name2(column_name2) USING INVERTED [PROPERTIES("parser" = "english|unicode|chinese")] [COMMENT 'your comment']
  INDEX idx_name3(column_name3) USING INVERTED [PROPERTIES("parser" = "chinese", "parser_mode" = "fine_grained|coarse_grained")]
[COMMENT 'your comment']
  INDEX idx_name4(column_name4) USING INVERTED [PROPERTIES("parser" = "english|unicode|chinese", "support_phrase" = "true|false")]
[COMMENT 'your comment']
  INDEX idx_name5(column_name4) USING INVERTED [PROPERTIES("char_filter_type" = "char_replace", "char_filter_pattern" = "._"),
"char_filter_replacement" = " "] [COMMENT 'your comment']
  INDEX idx_name5(column_name4) USING INVERTED [PROPERTIES("char_filter_type" = "char_replace", "char_filter_pattern" = "._")]
[COMMENT 'your comment']
)
table_properties;

-- Example: keyword matching in full-text searches, implemented by MATCH_ANY MATCH_ALL
SELECT * FROM table_name WHERE column_name MATCH_ANY | MATCH_ALL 'keyword1 ...';
```

****Recommendations****

1. If it is difficult to design a partitioning, bucketing, and prefix index strategy for your data range, consider introducing inverted indexes for acceleration.

****Restrictions****

1. Different data models have different restrictions on inverted index.
    1. Aggregate KEY model: only allows inverted index for Key columns
    2. Unique KEY model: allows inverted index for any column after enabling Merge-on-Write
    3. Duplicate KEY model: allows inverted index for any column

BloomFilter index

Doris supports adding BloomFilter indexes to fields with high value distinctiveness, making it suitable for scenarios that involve equivalence queries on columns with high cardinality.

Best practice

```
-- Example: add "bloom_filter_columns"="k1,k2,k3" in the PROPERTIES of the table creation statement.
-- To create BloomFilter index for saler_id and category_id in the table.
CREATE TABLE IF NOT EXISTS sale_detail_bloom  (
    sale_date date NOT NULL COMMENT "Sale data",
    customer_id int NOT NULL COMMENT "Customer ID",
    saler_id int NOT NULL COMMENT "Saler ID",
    sku_id int NOT NULL COMMENT "SKU ID",
    category_id int NOT NULL COMMENT "Category ID",
    sale_count int NOT NULL COMMENT "Sale count",
    sale_price DECIMAL(12,2) NOT NULL COMMENT "Sale price",
    sale_amt DECIMAL(20,2)  COMMENT "Sale amount"
)
Duplicate  KEY(sale_date, customer_id,saler_id,sku_id,category_id)
DISTRIBUTED BY HASH(saler_id) BUCKETS 10
PROPERTIES (
"bloom_filter_columns"="saler_id,category_id"
);
```

****Restrictions****

1. BloomFilter indexes are not supported for columns of type Tinyint, Float, and Double.
2. BloomFilter indexes can only accelerate filtering using “in” and “=” operators.
3. BloomFilter indexes must be built on high-cardinality columns (above 5000) in query conditions that involve “in” or “=” operators.
    1. BloomFilter indexes are suitable for non-prefix filtering.
    2. Queries will filter based on the high-frequency values in the column, and the filtering conditions are mostly “in” and “=”.
    3. Unlike Bitmap indexes, BloomFilter indexes are suitable for high-cardinality columns, such as UserID. If created on low-cardinality columns like “gender”, each block will contain almost all values, rendering the BloomFilter index meaningless.
    4. It is suitable for cases with data cardinality around half of the total range.
    5. For high-cardinality columns with equality (=) queries, such as ID numbers, using BloomFilter indexes can greatly accelerate performance.

Ngram BloomFilter index

Since 2.0.0, Doris has introduced the NGram BloomFilter index to improve the performance of “**LIKE**” queries.

Best practice

```
-- Example of creating NGram BloomFilter index in table creation statement
CREATE TABLE `nb_table` (
  `siteid` int(11) NULL DEFAULT "10" COMMENT "",
  `citycode` smallint(6) NULL COMMENT "",
  `username` varchar(32) NULL DEFAULT "" COMMENT "",
  INDEX idx_ngrambf (`username`) USING NGRAM_BF PROPERTIES("gram_size"="3", "bf_size"="256") COMMENT 'username ngram_bf index'
) ENGINE=OLAP
AGGREGATE KEY(`siteid`, `citycode`, `username`) COMMENT "OLAP"
DISTRIBUTED BY HASH(`siteid`) BUCKETS 10;

-- PROPERTIES("gram_size"="3", "bf_size"="256"), representing the number of grams and the byte size of the BloomFilter
-- The number of grams is determined according to the query cases and is typically set to the length of the majority of query strings. The number of bytes in the BloomFilter can be determined after testing. Generally, a larger number of bytes leads to better filtering results, and it is recommended to start with a value of 256 for testing and evaluating the effectiveness. However, it's important to note that a larger number of bytes also increases the storage cost of the index.
-- With high data cardinality, there is no need to set a large BloomFilter size. Conversely, with low data cardinality, increase the BloomFilter size to enhance filtering efficiency.
```

****Restrictions****

1. NGram BloomFilter index only supports string columns.
2. NGram BloomFilter indexes and BloomFilter indexes are mutually exclusive, meaning that only one of them can be set for the same column.
3. The sizes of the NGram and the BloomFilter can both be optimized based on the actual situation. If the NGram size is relatively small, you may increase the BloomFilter size.

4. For data at the scale of billions or above, if there is a need for fuzzy matching, it is recommended to use inverted indexes or NGram BloomFilter.

Bitmap index

To accelerate data queries, Doris supports users in adding Bitmap indexes to certain fields. This is suitable for scenarios involving equivalence or range queries on columns with lower cardinality.

Best practice

```
-- Example: create Bitmap index for siteid on bitmap_table
CREATE INDEX [IF NOT EXISTS] bitmap_index_name ON
bitmap_table (siteid)
USING BITMAP COMMENT 'bitmap_siteid';
```

****Restrictions****

1. Bitmap indexes can only be created on a single column.
2. Bitmap indexes can be applied to all columns in the `Duplicate` and `Unique` Key models, as well as the key columns in the `Aggregate` Key model.
3. Bitmap indexes support the following data types:
    1. `TINYINT`
    2. `SMALLINT`
    3. `INT`
    4. `BIGINT`
    5. `CHAR`
    6. `VARCHAR`
    7. `DATE`
    8. `DATETIME`
    9. `LARGEINT`
    10. `DECIMAL`
    11. `BOOL`
4. Bitmap indexes only take effect under Segment V2. The storage format of tables with Bitmap indexes will be automatically converted to V2 format by default.
5. Bitmap indexes should be constructed within a certain cardinality range. It is not suitable for extremely high or low cardinality cases.
    1. It is recommended for columns with a cardinality between 100 and 100,000, such as the occupation field or city field. If the duplication rate is too high, there won't be significant advantages to build Bitmap indexes compared to other types of indexes. If the duplication rate is too low, Bitmap indexes can significantly reduce space efficiency and performance. Specific types of queries, such as count, OR, and AND operations, only require bitwise operations.
    2. Bitmap indexes are more suitable for orthogonal queries.

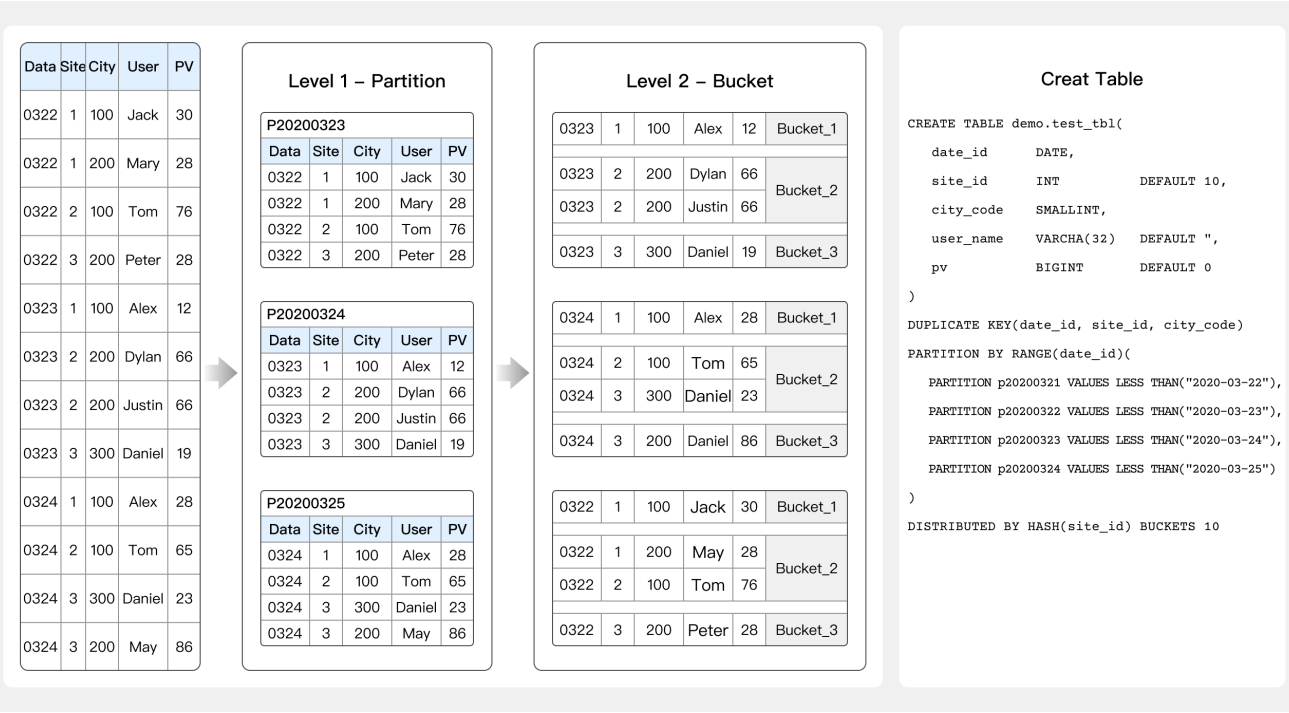
Field type

Doris supports various field types, including precise deduplication with BITMAP, fuzzy deduplication with HLL, semi-structured data types such as ARRAY/MAP/JSON, as well as common numeric, string, and time types.

****Recommendations****

1. VARCHAR
    1. Variable-length string with a length range of 1-65533 bytes. It is stored in UTF-8 encoding, where English characters typically occupy 1 byte.
    2. There is often a misunderstanding about the performance difference between varchar(255) and varchar(65533). If the data stored in both cases is the same, the performance will be the same as well. When creating a table, if you are unsure about the maximum length of the field, it is recommended to use varchar(65533) to prevent import errors caused by excessively long strings.
2. STRING
    1. Variable-length string with a default size of 1048576 bytes (1MB), which can be increased to 2147483643 bytes (2GB). It is stored in UTF-8 encoding, where English characters typically occupy 1 byte.
    2. It can only be used in value columns but not key columns or partitioning columns.
    3. It is suitable for storing large text content. However, if such a requirement does not exist, it is recommended to use VARCHAR. STRING columns have limitations as they cannot be used in key columns or partitioning columns.
3. Numeric fields: Choose the appropriate data type based on the required precision. There is no special restrictions on this.
4. Time fields: Note that if there is a high precision requirement (timestamp accurate to milliseconds), you need to specify the use of datetime(6). Otherwise, such timestamps are not supported by default.
5. It is recommended to use the JSON data type instead of string type for storing JSON data.

Create table



create-table-example

Considerations in creating a table include the setting of data partitions and buckets in addition to data model, index, and field types.

Best practice

```
-- Take Merge-on-Write tables in the Unique Key model as an example:
-- Merge-on-Write in the Unique Key model is implemented in a different way from the Aggregate Key model. The performance of it is similar to that on the Duplicate Key model.
-- In use cases requiring primary key constraints, the Aggregate Key model can deliver much better query performance compared to the Duplicate Key model, especially in aggregate queries and queries that involve filtering a large amount of data using indexes.

-- For non-partitioned tables
CREATE TABLE IF NOT EXISTS tbl_unique_merge_on_write
(
    `user_id` LARGEINT NOT NULL COMMENT "Use ID",
    `username` VARCHAR(50) NOT NULL COMMENT "Username",
    `register_time` DATE COMMENT "User registration time",
    `city` VARCHAR(20) COMMENT "User city",
    `age` SMALLINT COMMENT "User age",
    `sex` TINYINT COMMENT "User gender",
    `phone` LARGEINT COMMENT "User phone number",
    `address` VARCHAR(500) COMMENT "User address"
)
UNIQUE KEY(`user_id`, `username`)
-- Data volume of 3~5G
DISTRIBUTED BY HASH(`user_id`) BUCKETS 10
PROPERTIES (
-- In Doris 1.2.0, as a new feature, Merge-on-Write is disabled by default. Users can enable it by adding the following property.
"enable_unique_key_merge_on_write" = "true"
);

-- For partitioned tables
CREATE TABLE IF NOT EXISTS tbl_unique_merge_on_write_p
(
```

```
    `user_id` LARGEINT NOT NULL COMMENT "Use ID",
    `username` VARCHAR(50) NOT NULL COMMENT "Username",
    `register_time` DATE COMMENT "User registration time",
    `city` VARCHAR(20) COMMENT "User city",
    `age` SMALLINT COMMENT "User age",
    `sex` TINYINT COMMENT "User gender",
    `phone` LARGEINT COMMENT "User phone number",
    `address` VARCHAR(500) COMMENT "User address"
)
UNIQUE KEY(`user_id`, `username`, `register_time`)
PARTITION BY RANGE(`register_time`) (
    PARTITION p00010101_1899 VALUES [('0001-01-01'), ('1900-01-01')),
    PARTITION p19000101 VALUES [('1900-01-01'), ('1900-01-02')),
    PARTITION p19000102 VALUES [('1900-01-02'), ('1900-01-03')),
    PARTITION p19000103 VALUES [('1900-01-03'), ('1900-01-04')),
    PARTITION p19000104_1999 VALUES [('1900-01-04'), ('2000-01-01')),
    FROM ("2000-01-01") TO ("2022-01-01") INTERVAL 1 YEAR,
    PARTITION p30001231 VALUES [('3000-12-31'), ('3001-01-01')),
    PARTITION p99991231 VALUES [('9999-12-31'), (MAXVALUE))
)
-- Data volume of 3~5G
DISTRIBUTED BY HASH(`user_id`) BUCKETS 10
PROPERTIES (
-- In Doris 1.2.0, as a new feature, Merge-on-Write is disabled by default. Users can enable it by adding the following property.
"enable_unique_key_merge_on_write" = "true",
-- The unit for dynamic partition scheduling can be specified as HOUR, DAY, WEEK, MONTH, or YEAR.
"dynamic_partition.time_unit" = "MONTH",
-- The starting offset for dynamic partitioning is specified as a negative number. Depending on the value of the time_unit, it uses
the current day (week/month) as the reference point, partitions prior to this offset will be deleted (TTL). If not specified, the
default value is -2147483648, indicating that historical partitions will not be deleted.
"dynamic_partition.start" = "-3000",
-- The ending offset for dynamic partitioning is specified as a positive number. Depending on the value of the time_unit, it uses
the current day (week/month) as the reference point. Create the corresponding partitions of the specified range in advance.
"dynamic_partition.end" = "10",
-- The prefix for names of the dynamically created partitions (required).
"dynamic_partition.prefix" = "p",
-- The number of buckets corresponding to the dynamically created partitions.
"dynamic_partition.buckets" = "10",
"dynamic_partition.enable" = "true",
-- The following is the number of replicas corresponding to dynamically created partitions. If not specified, the default value will
be the replication factor specified when creating the table, which is typically 3.
"dynamic_partition.replication_num" = "3",
"replication_num" = "3"
);

-- View existing partitions
-- The actual number of created partitions is determined by a combination of dynamic_partition.start, dynamic_partition.end, and the
settings of PARTITION BY RANGE.
show partitions from tbl_unique_merge_on_write_p;
```

****Restrictions****

1. The database character set should be specified as UTF-8 since only UTF-8 is supported.
2. The replication factor for tables must be 3 (if not specified, it defaults to 3).
3. The data volume of an individual tablet (**Tablet Count = Partition Count * Bucket Count * Replication Factor**) theoretically has no upper or lower bounds, except for small tables (in the range of hundreds of megabytes to one gigabyte), where it should be ensured to be within the range of 1 GB to 10 GB:

1. If the data volume of an individual tablet is too small, it can result in poor data aggregation performance and increased metadata management overhead.
2. If the data volume is too large, it hinders replica migration, synchronization, and increases the cost of schema changes or materialization operations (these operations are retried at the granularity of a tablet).

4. For data exceeding 500 million records, **partitioning and bucketing** strategies must be implemented:

    1. **Recommendations for bucketing:**
        1. For large tables, each tablet should be in the range of 1 GB to 10 GB to prevent the generation of too many small files.
        2. For dimension tables of approximately 100 megabytes, the number of tablets should be controlled within the range of 3 to 5. This ensures a certain level of concurrency without generating excessive small files.
    2. If partitioning is not feasible and the data grows rapidly without the possibility of dynamic time-based partitioning, it is advisable to increase the number of buckets to accommodate the data volume based on the data retention period (180 days). It is still recommended to keep the size of each bucket between 1 GB and 10 GB.
    3. Apply salting to the bucketing field and use the same salting strategy for queries in order to leverage bucket pruning capabilities.
    4. Random bucketing:
        1. If an OLAP table does not have fields that need to be updated, setting the data bucketing mode to RANDOM can avoid severe data skew. During data ingestion, each batch of data is randomly assigned to a tablet for writing.
        2. When the bucketing mode for a table is set to RANDOM, since there is no bucketing column, querying the table will scan all buckets in the hit partitions instead of querying specific buckets based on the values of the bucketing column. This setting is suitable for overall aggregation and analysis queries rather than high-concurrency point queries.
        3. If an OLAP table has a random distribution of data, setting the `load_to_single_tablet` parameter to true during data ingestion allows each task to write to a single tablet. This improves concurrency and throughput during large-scale data ingestion. It can also reduce the write amplification caused by data ingestion and compaction and ensure cluster stability.
    5. Dimension tables, which grow slowly, can use a single partition and apply bucketing based on commonly used query conditions (where the data distribution of the bucketing field is relatively even).
    6. Fact tables: We recommend using DATE or DATETIME as the partitioning column. In most cases, one partition per day is sufficient. For the bucketing strategy, use frequently queried columns with relatively even data distribution as the bucketing columns.

5. For scenarios where there is a large amount of historical partitioned data but the historical data is relatively small, unbalanced, or queried infrequently, you can use the following approach to place the data in special partitions. You can create historical partitions for historical data of small sizes (e.g., yearly partitions, monthly partitions). For example, you can create historical partitions for data `FROM ("2000-01-01") TO ("2022-01-01") INTERVAL 1 YEAR`:

```
PARTITION p00010101_1899 VALUES [('0001-01-01'), ('1900-01-01')),
PARTITION p19000101 VALUES [('1900-01-01'), ('1900-01-02')),
...
PARTITION p19000104_1999 VALUES [('1900-01-04'), ('2000-01-01')),
FROM ("2000-01-01") TO ("2022-01-01") INTERVAL 1 YEAR,
PARTITION p30001231 VALUES [('3000-12-31'), ('3001-01-01')),
PARTITION p99991231 VALUES [('9999-12-31'), (MAXVALUE))
```

Loading Overview

Supported Data Sources

Doris provides a variety of data import solutions, and you can choose different data import methods for different data sources.

By Scene

Data Source	Loading Method
Object Storage (s3), HDFS	Import data using Broker
Local file	Import local data
MySQL, PostgreSQL, Oracle, SQLServer	Sync data via external table
Loading via JDBC	Sync data using JDBC
Loading JSON format data	JSON format data import

By Loading Method

Loading method name	Use method
Broker Load	Import external storage data via Broker
Stream Load	Stream import data (local file and memory data)
Routine Load	Import Kafka data
Insert Into	External table imports data through INSERT
S3 Load	Object storage data import of S3 protocol
MySQL Load	Local data import of MySql protocol

Supported Data Formats

Different import methods support slightly different data formats.

Import Methods	Supported Formats
Broker Load	parquet, orc, csv, gzip
Stream Load	csv, json, parquet, orc
Routine Load	csv, json
MySql Load	csv

Import Instructions

The data import implementation of Apache Doris has the following common features, which are introduced here to help you better use the data import function

Import Atomicity Guarantees

Each import job of Doris, whether it is batch import using Broker Load or single import using INSERT statement, is a complete transaction operation. The import transaction can ensure that the data in a batch takes effect atomically, and there will be no partial data writing.

At the same time, an import job will have a Label. This Label is unique under a database (Database) and is used to uniquely identify an import job. Label can be specified by the user, and some import functions will also be automatically generated by the system.

Label is used to ensure that the corresponding import job can only be successfully imported once. A successfully imported Label, when used again, will be rejected with the error `Label already used`. Through this mechanism, `At-Most-Once` semantics can be implemented in Doris. If combined with the `At-Least-Once` semantics of the upstream system, the `Exactly-Once` semantics of imported data can be achieved.

For best practices on atomicity guarantees, see Importing Transactions and Atomicity.

Synchronous and Asynchronous Imports

Import methods are divided into synchronous and asynchronous. For the synchronous import method, the returned result indicates whether the import succeeds or fails. For the asynchronous import method, a successful return only means that the job was submitted successfully, not that the data was imported successfully. You need to use the corresponding command to check the running status of the import job.

Import the Data of Array Types

For example, in the following import, you need to cast columns b14 and a13 into `array<string>` type, and then use the `array_union` function.

```
LOAD LABEL label_03_14_49_34_898986_19090452100 (
  DATA INFILE("hdfs://test.hdfs.com:9000/user/test/data/sys/load/array_test.data")
  INTO TABLE `test_array_table`
  COLUMNS TERMINATED BY "|" (`k1`, `a1`, `a2`, `a3`, `a4`, `a5`, `a6`, `a7`, `a8`, `a9`, `a10`, `a11`, `a12`, `a13`, `b14`)
  SET(a14=array_union(cast(b14 as array<string>), cast(a13 as array<string>))) WHERE size(a2) > 270)
  WITH BROKER "hdfs" ("username"="test_array", "password"="")
  PROPERTIES( "max_filter_ratio"="0.8" );
```

Execution Engine Selected

The Pipeline engine is turned off by default on import, and is enabled by the following two variables:

1. `enable_pipeline_load` in FE CONFIG `enable_pipeline_load`. When enabled, import tasks such as Streamload will try to use the Pipeline engine.

2. `enable_nereids_dml_with_pipeline` in Session Variable to enable insert into to try to use the Pipeline engine.

Whether or not the Pipeline engine is used when the above variable is enabled still depends on the Session Variables `enable_pipeline_engine`. If this value is `false`, the import will not use the Pipeline engine even if the above variable is set to `true`.

Stream Load

Stream Load supports importing local files or data streams into Doris through the HTTP protocol.

Stream Load is a synchronous import method that returns the import result after the import is executed, allowing you to determine the success of the import through the request response. Generally, user can use Stream Load to import files under 10GB. If the file is too large, it is recommended to split the file and then use Stream Load for importing. Stream Load can ensure the atomicity of a batch of import tasks, meaning that either all of them succeed or all of them fail.

> **In comparison to single-threaded load using `curl`, Doris Streamloader is a client tool designed for loading data into Apache Doris. it reduces the ingestion latency of large datasets by its concurrent loading capabilities. It comes with the following features:**
>
> - **Parallel loading**: multi-threaded load for the Stream Load method. You can set the parallelism level using the `workers` parameter.
> - **Multi-file load:** simultaneously load of multiple files and directories with one shot. It supports recursive file fetching and allows you to specify file names with wildcard characters.
> - **Path traversal support:** support path traversal when the source files are in directories
> - **Resilience and continuity:** in case of partial load failures, it can resume data loading from the point of failure.
> - **Automatic retry mechanism:** in case of loading failures, it can automatically retry a default number of times. If the loading remains unsuccessful, it will print the command for manual retry.
>
> See Doris Streamloader for detailed instructions and best practices.

User guide

Supported formats

Stream Load supports importing data in CSV, JSON, Parquet, and ORC formats.

Usage limitations

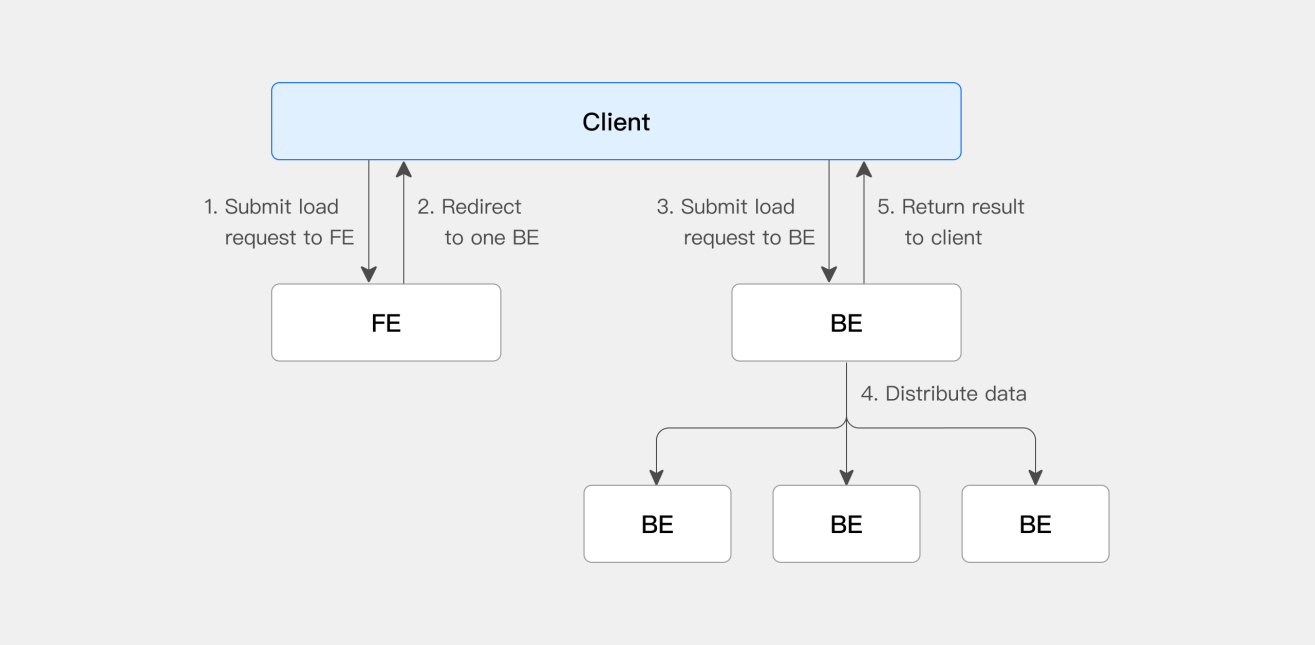
When importing CSV files, it is necessary to clearly distinguish between null values and empty strings:

- Null values need to be represented by `\N`. For example, the data “a,,b” indicates that the middle column is a null value.
- Empty strings can be represented by leaving the data empty. For example, the data “a, ,b” indicates that the middle column is an empty string.

Basic principles

When using Stream Load, it is necessary to initiate an import job through the HTTP protocol to the FE (Frontend) node. The FE will redirect the request to a BE (Backend) node in a round-robin manner to achieve load balancing. It is also possible to send HTTP requests directly to a specific BE node. In Stream Load, Doris selects one node to serve as the Coordinator node. The Coordinator node is responsible for receiving data and distributing it to other nodes.

The following figure shows the main flow of Stream load, omitting some import details.



Basic principles

1. The client submits a Stream Load import job request to the FE (Frontend).
2. The FE randomly selects a BE (Backend) as the Coordinator node, which is responsible for scheduling the import job, and then returns an HTTP redirect to the client.
3. The client connects to the Coordinator BE node and submits the import request.
4. The Coordinator BE distributes the data to the appropriate BE nodes and returns the import result to the client once the import is complete.
5. Alternatively, the client can directly specify a BE node as the Coordinator and distribute the import job directly.

Quick start

Stream Load import data through the HTTP protocol. The following example uses the curl tool to demonstrate submitting an import job through Stream Load.

For detailed syntax, please refer to STREAM LOAD.

Prerequisite check

Stream Load requires `INSERT` privileges on the target table. If there are no `INSERT` privileges, it can be granted to the user through the GRANT command.

Create load job

Loading CSV

1. Creating loading data

   Create a CSV file named `streamload_example.csv`. The specific content is as follows

```
1,Emily,25
2,Benjamin,35
3,Olivia,28
4,Alexander,60
5,Ava,17
6,William,69
7,Sophia,32
8,James,64
9,Emma,37
10,Liam,64
```

2. Creating a table for loading

   Create the table that will be imported into, using the specific syntax as follows:

```
CREATE TABLE testdb.test_streamload(
    user_id            BIGINT       NOT NULL COMMENT "User ID",
    name               VARCHAR(20)           COMMENT "User name",
    age                INT                   COMMENT "User age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

3. Enabling the load job

   The Stream Load job can be submitted using the `curl` command.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "column_separator:," \
    -H "columns:user_id,name,age" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Stream Load is a synchronous method, where the result is directly returned to the user.

```
{
    "TxnId": 3,
    "Label": "123",
    "Comment": "",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 10,
    "NumberLoadedRows": 10,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 118,
```

```
    "LoadTimeMs": 173,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 70,
    "ReadDataTimeMs": 2,
    "WriteDataTimeMs": 48,
    "CommitAndPublishTimeMs": 52
}
```

4. View data

```
mysql> select count(*) from testdb.test_streamload;
+----------+
| count(*) |
+----------+
|       10 |
+----------+
```

Loading JSON

1. Creating loading data

Create a JSON file named `streamload_example.json` . The specific content is as follows

```
[
{"userid":1,"username":"Emily","userage":25},
{"userid":2,"username":"Benjamin","userage":35},
{"userid":3,"username":"Olivia","userage":28},
{"userid":4,"username":"Alexander","userage":60},
{"userid":5,"username":"Ava","userage":17},
{"userid":6,"username":"William","userage":69},
{"userid":7,"username":"Sophia","userage":32},
{"userid":8,"username":"James","userage":64},
{"userid":9,"username":"Emma","userage":37},
{"userid":10,"username":"Liam","userage":64}
]
```

2. Creating a table for loading

   Create the table that will be imported into, using the specific syntax as follows:

```
CREATE TABLE testdb.test_streamload(
    user_id            BIGINT       NOT NULL COMMENT "User ID",
    name               VARCHAR(20)           COMMENT "User name",
    age                INT                   COMMENT "User age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

3. Enabling the load job

   The Stream Load job can be submitted using the `curl` command.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "label:124" \
    -H "Expect:100-continue" \
    -H "format:json" -H "strip_outer_array:true" \
```

```
    -H "jsonpaths:[\"$.userid\", \"$.username\", \"$.userage\"]" \
    -H "columns:user_id,name,age" \
    -T streamload_example.json \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

> **Note**
>
> If the JSON file is not a JSON array but each line is a JSON object, add the headers `-H "strip_outer_array:false"` and `-H "read_json_by_line:true"`.

Stream Load is a synchronous method, where the result is directly returned to the user.

```
{
    "TxnId": 7,
    "Label": "125",
    "Comment": "",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 10,
    "NumberLoadedRows": 10,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 471,
    "LoadTimeMs": 52,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 11,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 23,
    "CommitAndPublishTimeMs": 16
}
```

View load job

By default, Stream Load synchronously returns results to the client, so the system does not record Stream Load historical jobs. If recording is required, add the configuration `enable_stream_load_record=true` in `be.conf`. Refer to the BE configuration options for specific details.

After configuring, you can use the `show stream load` command to view completed Stream Load jobs.

```
mysql> show stream load from testdb;
+-------+--------+-----------------+---------------+---------+---------+------+-----------+------------+--------------+-------------
---+-----------+-------------------------+-------------------------+------+---------+
| Label | Db     | Table           | ClientIp      | Status  | Message | Url  | TotalRows | LoadedRows | FilteredRows |
UnselectedRows | LoadBytes | StartTime               | FinishTime              | User | Comment |
+-------+--------+-----------------+---------------+---------+---------+------+-----------+------------+--------------+-------------
---+-----------+-------------------------+-------------------------+------+---------+
| 12356 | testdb | test_streamload | 192.168.88.31 | Success | OK      | N/A  | 10        | 10         | 0            | 0
| 118       | 2023-11-29 08:53:00.594 | 2023-11-29 08:53:00.650 | root |         |
+-------+--------+-----------------+---------------+---------+---------+------+-----------+------------+--------------+-------------
---+-----------+-------------------------+-------------------------+------+---------+
1 row in set (0.00 sec)
```

Cancel load job

Users cannot manually cancel a Stream Load operation. A Stream Load job will be automatically canceled by the system if it encounters a timeout (set to 0) or an import error.

Reference manual

Command

The syntax for Stream Load is as follows:

```
curl --location-trusted -u <doris_user>:<doris_password> \
  -H "Expect:100-continue" [-H ""...] \
  -T <file_path> \
  -XPUT http://fe_host:http_port/api/{db}/{table}/_stream_load
```

Stream Load operations support both HTTP chunked and non-chunked import methods. For non-chunked imports, it is necessary to have a Content-Length header to indicate the length of the uploaded content, which ensures data integrity.

Load configuration parameters

FE configuration

1. `stream_load_default_timeout_second`
    - Default Value: 259200 (s)
    - Dynamic Configuration: Yes
    - FE Master-only Configuration: Yes

Parameter Description: The default timeout for Stream Load. The load job will be canceled by the system if it is not completed within the set timeout (in seconds). If the source file cannot be imported within the specified time, the user can set an individual timeout in the Stream Load request. Alternatively, adjust the `stream_load_default_timeout_second` parameter on the FE to set the global default timeout.

2. `enable_pipeline_load`

Determines whether to enable the Pipeline engine to execute Streamload tasks. See the import documentation for more details.

BE configuration

1. `streaming_load_max_mb`
    - Default value: 10240 (MB)
    - Dynamic configuration: Yes
    - Parameter description: The maximum import size for Stream load. If the user's original file exceeds this value, the `streaming_load_max_mb` parameter on the BE needs to be adjusted.

2. Header parameters

   Load parameters can be passed through the HTTP Header section. See below for specific parameter descriptions.

Parameters	Parameters description
label	Used to specify a label for this Doris import. Data with the same label cannot be imported multiple times. If no label is specified, Doris will automatically generate one. Users can avoid duplicate imports of the same data by specifying a label. Doris retains import job labels for three days by default, but this duration can be adjusted using `label_keep_max_second`. For example, to specify the label for this import as 123, use the command `-H "label:123"`. The use of labels prevents users from importing the same data repeatedly. It is strongly recommended that users use the same label for the same batch of data. This ensures that duplicate requests for the same batch of data are only accepted once, guaranteeing At-Most-Once semantics. When the status of an import job corresponding to a label is CANCELLED, that label can be used again.
column_separator	Used to specify the column separator in the import file, which defaults to `\t`. If the separator is an invisible character, it needs to be prefixed with `\x` and represented in hexadecimal format. Multiple characters can be combined as a column separator. For example, to specify the separator as `\x01` for a Hive file, use the command `-H "column_separator:\x01"`.
line_delimiter	Used to specify the line delimiter in the import file, which defaults to `\n`. Multiple characters can be combined as a line delimiter. For example, to specify the line delimiter as `\n`, use the command `-H "line_delimiter:\n"`.
columns	Used to specify the correspondence between columns in the import file and columns in the table. If the columns in the source file exactly match the content of the table, this field does not need to be specified. If the schema of the source file does not match the table, this field is required for data transformation. There are two formats: direct column correspondence to fields in the import file, and derived columns represented by expressions. Refer to Data Transformation for detailed examples.
where	Used to filter out unnecessary data. If users need to exclude certain data, they can achieve this by setting this option. For example, to import only data where the k1 column is equal to 20180601, specify `-H "where: k1 = 20180601"` during the import.
max_filter_ratio	Used to specify the maximum tolerable ratio of filterable (irregular or otherwise problematic) data, which defaults to zero tolerance. The value range is 0 to 1. If the error rate of the imported data exceeds this value, the import will fail. Irregular data does not include rows filtered out by the where condition. For example, to maximize the import of all correct data (100% tolerance), specify the command `-H "max_filter_ratio:1"`.
partitions	Used to specify the partitions involved in this import. If users can determine the corresponding partitions for the data, it is recommended to specify this option. Data that does not meet these partition criteria will be filtered out. For example, to specify importing into partitions p1 and p2, use the command `-H "partitions: p1, p2"`.
timeout	Used to specify the timeout for the import in seconds. The default is 600 seconds, and the configurable range is from 1 second to 259200 seconds. For example, to specify an import timeout of 1200 seconds, use the command `-H "timeout:1200"`.
strict_mode	Used to specify whether to enable strict mode for this import, which is disabled by default. For example, to enable strict mode, use the command `-H "strict_mode:true"`.
timezone	Used to specify the timezone to be used for this import, which defaults to GMT+8. This parameter affects the results of all timezone-related functions involved in the import. For example, to specify the import timezone as Africa/Abidjan, use the command `-H "timezone:Africa/Abidjan"`.

Parameters	Parameters description
exec_mem_limit	The memory limit for the import, which defaults to 2GB. The unit is bytes.
format	Used to specify the format of the imported data, which defaults to CSV. Currently supported formats include: csv, json, csv_with_names (supports filtering the first row of the csv file), csv_with_names_and_types (supports filtering the first two rows of the csv file), parquet, and orc. For example, to specify the imported data format as json, use the command `-H "format:json"`.
jsonpaths	There are two ways to import JSON data format: Simple Mode and Matching Mode. If no jsonpaths are specified, it is the simple mode that requires the JSON data to be of the object type.Matching mode used when the JSON data is relatively complex and requires matching the corresponding values through the jsonpaths parameter.In simple mode, the keys in JSON are required to correspond one-to-one with the column names in the table. For example, in the JSON data `{"k1":1, "k2":2, "k3":"hello"}`, k1, k2, and k3 correspond to the columns in the table respectively.
strip_outer_array	When `strip_outer_array` is set to true, it indicates that the JSON data starts with an array object and flattens the objects within the array. The default value is false. When the outermost layer of the JSON data is represented by `[]`, which denotes an array, `strip_outer_array` should be set to true. For example, with the following data, setting `strip_outer_array` to true will result in two rows of data being generated when imported into Doris: `[{"k1": 1, "v1": 2}, {"k1": 3, "v1": 4}]`.
json_root	`json_root` is a valid jsonpath string that specifies the root node of a JSON document, with a default value of “”.
merge_type	There are three types of data merging: APPEND, DELETE, and MERGE. APPEND is the default value, indicating that this batch of data needs to be appended to the existing data. DELETE means to remove all rows that have the same keys as this batch of data. MERGE semantics need to be used in conjunction with delete conditions. It means that data satisfying the delete conditions will be processed according to DELETE semantics, while the rest will be processed according to APPEND semantics. For example, to specify the merge mode as MERGE, you need to specify the commands `-H "merge_type: MERGE" -H "delete: flag=1"`.
delete	It is only meaningful under MERGE, representing the deletion conditions for data.
function_column.sequence_col	It is suitable only for the UNIQUE KEYS model. Within the same Key column, it ensures that the Value column is replaced according to the specified source_sequence column. The source_sequence can either be a column from the data source or an existing column in the table structure.
fuzzy_parse	It is a boolean type. If set to true, the JSON will be parsed with the first row as the schema. Enabling this option can improve the efficiency of JSON imports, but it requires that the order of the keys in all JSON objects be consistent with the first line. The default is false and it is only used for JSON format.
num_as_string	It is a boolean type. When set to true, indicates that numeric types will be converted to strings during JSON parsing to ensure no loss of precision during the import process.
read_json_by_line	It is a boolean type. When set to true, indicates support for reading one JSON object per line, defaulting to false.
send_batch_parallelism	An integer used to set the parallelism for sending batch-processed data. If the parallelism value exceeds the `max_send_batch_parallelism_per_job` configured in BE, the coordinating BE will use the `max_send_batch_parallelism_per_job value`.

Parameters	Parameters description
hidden_columns	Used to specify hidden columns in the imported data, which takes effect when the Header does not include Columns. Multiple hidden columns are separated by commas. The system will use the user-specified data for import. In the following example, the last column of data in the imported data is `__DORIS_SEQUENCE_COL__`. `hidden_columns: __DORIS_DELETE_SIGN__,__DORIS_SEQUENCE_COL__`.
load_to_single_tablet	It is a boolean type. When set to true, indicates support for importing data only to a single Tablet corresponding to the partition, defaulting to false. This parameter is only allowed when importing to an OLAP table with random bucketing.
compress_type	Currently, only compression of CSV files is supported. Compression formats include gz, lzo, bz2, lz4, lzop, and deflate.
trim_double_quotes	It is a boolean type. When set to true, indicates trimming of the outermost double quotes for each field in the CSV file, defaulting to false.
skip_lines	It is an integer type. Used to specify the number of lines to skip at the beginning of the CSV file, defaulting to 0. When the `format` is set to `csv_with_names` or `csv_with_names_and_types`, this parameter will become invalid.
comment	It is a String type, with an empty string as the default value. Used to add additional information to the task.
enclose	Specify the enclosure character. When a CSV data field contains a row delimiter or column delimiter, to prevent unexpected truncation, you can specify a single-byte character as the enclosure for protection. For example, if the column delimiter is “,” and the enclosure is ““”, the data”a,’b,c’” will have “b,c” parsed as a single field. Note: When the enclosure is set to a double quote (“), make sure to set `trim_double_quotes` to true.
escape	Specify the escape character. It is used to escape characters that are the same as the enclosure character within a field. For example, if the data is “a,‘b,’c’”, and the enclosure is ““”, and you want”b,’c” to be parsed as a single field, you need to specify a single-byte escape character, such as ““, and modify the data to”a,’b’,‘c’”.

Load return value

Stream Load is a synchronous import method, and the load result is directly provided to the user through the creation of an load return value, as shown below:

```
{
    "TxnId": 1003,
    "Label": "b6f3bc78-0d2c-45d9-9e4c-faa0a0149bee",
    "Status": "Success",
    "ExistingJobStatus": "FINISHED", // optional
    "Message": "OK",
    "NumberTotalRows": 1000000,
    "NumberLoadedRows": 1000000,
    "NumberFilteredRows": 1,
    "NumberUnselectedRows": 0,
    "LoadBytes": 40888898,
    "LoadTimeMs": 2144,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 2,
    "ReadDataTimeMs": 325,
    "WriteDataTimeMs": 1933,
    "CommitAndPublishTimeMs": 106,
    "ErrorURL": "http://192.168.1.1:8042/api/_load_error_log?file=__shard_0/error_log_insert_stmt_db18266d4d9b4ee5-
```

```
abb00ddd64bdf005_db18266d4d9b4ee5_abb00ddd64bdf005"
}
```

The return result parameters are explained in the following table:

Parameters	Parameters description
TxnId	Import transaction ID
Label	Label of load job，specified via `-H "label:<label_id>"`.
Status	Final load Status. **Success**: The load job was successful.**Publish Timeout**: The load job has been completed, but there may be a delay in data visibility. **Label Already Exists**: The label is duplicated, requiring a new label. **Fail**: The load job failed.
ExistingJobStatus	The status of the load job corresponding to the already existing label. This field is only displayed when the Status is **Label Already Exists**. Users can use this status to know the status of the import job corresponding to the existing label. **RUNNING** means the job is still executing, and **FINISHED** means the job was successful.
Message	Error information related to the load job.
NumberTotalRows	The total number of rows processed during the load job.
NumberLoadedRows	The number of rows that were successfully loaded.
NumberFilteredRows	The number of rows that did not meet the data quality standards.
NumberUnselectedRows	The number of rows that were filtered out based on the WHERE condition.
LoadBytes	The amount of data in bytes.
LoadTimeMs	The time taken for the load job to complete, measured in milliseconds.
BeginTxnTimeMs	The time taken to request the initiation of a transaction from the Frontend node (FE), measured in milliseconds.
StreamLoadPutTimeMs	The time taken to request the execution plan for the load job data from the FE, measured in milliseconds.
ReadDataTimeMs	The time spent reading the data during the load job, measured in milliseconds.
WriteDataTimeMs	The time taken to perform the data writing operations during the load job, measured in milliseconds.
CommitAndPublishTimeMs	The time taken to request the commit and publish the transaction from the FE, measured in milliseconds.
ErrorURL	If there are data quality issues, users can access this URL to view the specific rows with errors.

Users can access the ErrorURL to review data that failed to import due to issues with data quality. By executing the command `curl "<ErrorURL>"`, users can directly retrieve information about the erroneous data.

Load example

Setting load timeout and maximum size

The timeout for a load job is measured in seconds. If the load job is not completed within the specified timeout period, it will be cancelled by the system and marked as `CANCELLED`. You can adjust the timeout for a Stream Load job by specifying the `timeout` parameter or adding the `stream_load_default_timeout_second` parameter in the fe.conf file.

Before initiating the load, you need to calculate the timeout based on the file size. For example, for a 100GB file with an estimated load performance of 50MB/s:

```
Load time ≈ 100GB / 50MB/s ≈ 2048s
```

You can use the following command to specify a timeout of 3000 seconds for creating a Stream Load job:

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "timeout:3000"
    -H "column_separator:," \
    -H "columns:user_id,name,age" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Setting maximum error tolerance rate

Load job can tolerate a certain amount of data with formatting errors. The tolerance rate is configured using the `max_filter_ratio` parameter. By default, it is set to 0, meaning that if there is even a single erroneous data row, the entire load job will fail. If users wish to ignore some problematic data rows, they can set this parameter to a value between 0 and 1. Doris will automatically skip rows with incorrect data formats. For more information on calculating the tolerance rate, please refer to the Data Transformation documentation.

You can use the following command to specify a `max_filter_ratio` tolerance of 0.4 for creating a Stream Load job:

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "max_filter_ratio:0.4" \
    -H "column_separator:," \
    -H "columns:user_id,name,age" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Setting load filtering conditions

During the load job, you can use the WHERE parameter to apply conditional filtering to the imported data. The filtered data will not be included in the calculation of the filter ratio and will not affect the setting of `max_filter_ratio`. After the load job is complete, you can view the number of filtered rows by checking `num_rows_unselected`.

You can use the following command to specify WHERE filtering conditions for creating a Stream Load job:

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "where:age>=35" \
    -H "column_separator:," \
    -H "columns:user_id,name,age" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Loading data into specific partitions

Loading data from local files into partitions p1 and p2 of the table, allowing a 20% error rate.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "label:123" \
    -H "Expect:100-continue" \
    -H "max_filter_ratio:0.2" \
    -H "column_separator:," \
    -H "columns:user_id,name,age" \
    -H "partitions: p1, p2" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Loading data into specific timezone

Since Doris currently does not have a built-in time zone time type, all `DATETIME` related types only represent absolute time points, do not contain time zone information, and will not change due to changes in the Doris system time zone. Therefore, for the import of data with time zones, our unified processing method is to convert it into data in a specific target time zone. In the Doris system, it is the time zone represented by the session variable `time_zone`.

In the import, our target time zone is specified through the parameter `timezone`. This variable will replace the session variable `time_zone` when time zone conversion occurs and time zone sensitive functions are calculated. Therefore, if there are no special circumstances, the `timezone` should be set in the import transaction to be consistent with the `time_zone` of the current Doris cluster. This means that all time data with a time zone will be converted to this time zone.

For example, the Doris system time zone is “+08:00”, and the time column in the imported data contains two pieces of data, namely “2012-01-01 01:00:00+00:00” and “2015-12-12 12 :12:12-08:00”, then after we specify the time zone of the imported transaction through `-H "timezone: +08:00"` when importing, both pieces of data will be converted to the time zone to obtain the result.” 2012-01-01 09:00:00” and “2015-12-13 04:12:12”.

For more information on time zone interpretation, please refer to the document Time Zone.

Streamingly import

Stream Load is based on the HTTP protocol for importing, which supports using programming languages such as Java, Go, or Python for streaming import. This is why it is named Stream Load.

The following example demonstrates this usage through a bash command pipeline. The imported data is generated streamingly by the program rather than from a local file.

```
seq 1 10 | awk '{OFS="\t"}{print $1, $1 * 10}' | curl --location-trusted -u root -T -
http://host:port/api/testDb/testTbl/_stream_load
```

Setting CSV first row filtering

File data:

```
id,name,age
1,doris,20
2,flink,10
```

Filtering the first row during load by specifying `format=csv_with_names`

```
curl --location-trusted -u root -T test.csv  -H "label:1" -H "format:csv_with_names" -H "column_separator:,"
http://host:port/api/testDb/testTbl/_stream_load
```

Specifying merge_type for DELETE operations

In stream load, there are three import types: APPEND, DELETE, and MERGE. These can be adjusted by specifying the parameter `merge_type`. If you want to specify that all data with the same key as the imported data should be deleted, you can use the following command:

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "merge_type: DELETE" \
    -H "column_separator:," \
    -H "columns:user_id,name,age" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Before loading:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      3 |        2 | tom      |    2 |
|      4 |        3 | bush     |    3 |
|      5 |        3 | helen    |    3 |
+--------+----------+----------+------+
```

The imported data is:

```
3,2,tom,0
```

After importing, the original table data will be deleted, resulting in the following result:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      4 |        3 | bush     |    3 |
|      5 |        3 | helen    |    3 |
+--------+----------+----------+------+
```

Specifying merge_type for MERGE operation

By specifying `merge_type` as MERGE, the imported data can be merged into the table. The MERGE semantics need to be used in combination with the DELETE condition, which means that data satisfying the DELETE condition is processed according to the DELETE semantics, and the rest is added to the table according to the

APPEND semantics. The following operation represents deleting the row with `siteid` of 1, and adding the rest of the data to the table:

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "merge_type: MERGE" \
    -H "delete: siteid=1" \
    -H "column_separator:," \
    -H "columns:user_id,name,age" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Before loading:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      4 |        3 | bush     |    3 |
|      5 |        3 | helen    |    3 |
|      1 |        1 | jim      |    2 |
+--------+----------+----------+------+
```

The imported data is:

```
2,1,grace,2
3,2,tom,2
1,1,jim,2
```

After loading, the row with `siteid = 1` will be deleted according to the condition, and the rows with `siteid` of 2 and 3 will be added to the table:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      4 |        3 | bush     |    3 |
|      2 |        1 | grace    |    2 |
|      3 |        2 | tom      |    2 |
|      5 |        3 | helen    |    3 |
+--------+----------+----------+------+
```

Specifying sequence column for merge

When a table with a Unique Key has a Sequence column, the value of the Sequence column serves as the basis for the replacement order in the REPLACE aggregation function under the same Key column. A larger value can replace a smaller one. When marking deletions based on `DORIS_DELETE_SIGN` for such a table, it is necessary to ensure that the Key is the same and that the Sequence column value is greater than or equal to the current value. By specifying the `function_column.sequence_col` parameter, deletion operations can be performed in combination with `merge_type: DELETE`.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "merge_type: DELETE" \
    -H "function_column.sequence_col: age"
```

```
    -H "column_separator:," \
    -H "columns: name, gender, age"
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Given the following table schema:

```
mysql> SET show_hidden_columns=true;
Query OK, 0 rows affected (0.00 sec)

mysql> DESC table1;
+------------------------+--------------+------+-------+---------+---------+
| Field                  | Type         | Null | Key   | Default | Extra   |
+------------------------+--------------+------+-------+---------+---------+
| name                   | VARCHAR(100) | No   | true  | NULL    |         |
| gender                 | VARCHAR(10)  | Yes  | false | NULL    | REPLACE |
| age                    | INT          | Yes  | false | NULL    | REPLACE |
| __DORIS_DELETE_SIGN__  | TINYINT      | No   | false | 0       | REPLACE |
| __DORIS_SEQUENCE_COL__ | INT          | Yes  | false | NULL    | REPLACE |
+------------------------+--------------+------+-------+---------+---------+
4 rows in set (0.00 sec)
```

The original table data is:

```
+-------+--------+------+
| name  | gender | age  |
+-------+--------+------+
| li    | male   |   10 |
| wang  | male   |   14 |
| zhang | male   |   12 |
+-------+--------+------+
```

1. Sequence parameter takes Eeffect, loading sequence column value is larger than or equal to the existing data in the table.

   loading data as:

```
li,male,10
```

Since `function_column.sequence_col` is specified as `age`, and the `age` value is larger than or equal to the existing column in the table, the original table data is deleted. The table data becomes:

```
+-------+--------+------+
| name  | gender | age  |
+-------+--------+------+
| wang  | male   |   14 |
| zhang | male   |   12 |
+-------+--------+------+
```

2. Sequence parameter do not take effect, loading sequence column value is less than or equal to the existing data in the table:

   loading data as:

```
li,male,9
```

Since `function_column.sequence_col` is specified as `age`, but the `age` value is less than the existing column in the table, the delete operation does not take effect. The table data remains unchanged, and the row with the primary key of `li` is still visible:

```
+-------+--------+------+
| name  | gender | age  |
+-------+--------+------+
| li    | male   |   10 |
| wang  | male   |   14 |
| zhang | male   |   12 |
+-------+--------+------+
```

It is not deleted because that, at the underlying dependency level, it first checks for rows with the same key. It displays the row data with the larger sequence column value. Then, it checks the `DORIS_DELETE_SIGN` value for that row. If it is 1, it is not displayed externally. If it is 0, it is still read and displayed.

Loading data with enclosing characters

When the data in a CSV file contains delimiters or separators, single-byte characters can be specified as enclosing characters to protect the data from being truncated.

For example, in the following data where a comma is used as the separator but also exists within a field:

```
zhangsan,30,'Shanghai, HuangPu District, Dagu Road'
```

By specifying an enclosing character such as a single quotation mark ', the entire `Shanghai, HuangPu District, Dagu Road` can be treated as a single field.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "column_separator:," \
    -H "enclose:'" \
    -H "columns:username,age,address" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

If the enclosing character also appears within a field, such as wanting to treat `Shanghai City, Huangpu District, \'Dagu Road` as a single field, it is necessary to first perform string escaping within the column:

```
Zhang San,30,'Shanghai, Huangpu District, \'Dagu Road'
```

An escape character, which is a single-byte character, can be specified using the escape parameter. In the example, the backslash \ is used as the escape character.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "column_separator:," \
    -H "enclose:'" \
    -H "escape:\\" \
```

```
    -H "columns:username,age,address" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Loading fields containing default CURRENT_TIMESTAMP type

Here's an example of loading data into a table that contains a field with the DEFAULT CURRENT_TIMESTAMP type:

Table schema:

```
`id` bigint(30) NOT NULL,
`order_code` varchar(30) DEFAULT NULL COMMENT '',
`create_time` datetimev2(3) DEFAULT CURRENT_TIMESTAMP
```

JSON data type:

```
{"id":1,"order_Code":"avc"}
```

Command:

```
curl --location-trusted -u root -T test.json -H "label:1" -H "format:json" -H 'columns: id, order_code,
create_time=CURRENT_TIMESTAMP()' http://host:port/api/testDb/testTbl/_stream_load
```

Simple mode for loading JSON format data

When the JSON fields correspond one-to-one with the column names in the table, you can import JSON data format into the table by specifying the parameters "strip_outer_array:true" and "format:json".

For example, if the table is defined as follows:

```
CREATE TABLE testdb.test_streamload(
    user_id            BIGINT       NOT NULL COMMENT "User ID",
    name               VARCHAR(20)           COMMENT "User name",
    age                INT                   COMMENT "User age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

And the data field names correspond one-to-one with the column names in the table:

```
[
{"user_id":1,"name":"Emily","age":25},
{"user_id":2,"name":"Benjamin","age":35},
{"user_id":3,"name":"Olivia","age":28},
{"user_id":4,"name":"Alexander","age":60},
{"user_id":5,"name":"Ava","age":17},
{"user_id":6,"name":"William","age":69},
{"user_id":7,"name":"Sophia","age":32},
{"user_id":8,"name":"James","age":64},
{"user_id":9,"name":"Emma","age":37},
```

```
{"user_id":10,"name":"Liam","age":64}
]
```

You can use the following command to load JSON data into the table:

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "format:json" \
    -H "strip_outer_array:true" \
    -T streamload_example.json \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Matching mode for loading complex JSON format data

When the JSON data is more complex and cannot correspond one-to-one with the column names in the table, or there are extra columns, you can use the jsonpaths parameter to complete the column name mapping and perform data matching import. For example, with the following data:

```
[
{"userid":1,"hudi":"lala","username":"Emily","userage":25,"userhp":101},
{"userid":2,"hudi":"kpkp","username":"Benjamin","userage":35,"userhp":102},
{"userid":3,"hudi":"ji","username":"Olivia","userage":28,"userhp":103},
{"userid":4,"hudi":"popo","username":"Alexander","userage":60,"userhp":103},
{"userid":5,"hudi":"uio","username":"Ava","userage":17,"userhp":104},
{"userid":6,"hudi":"lkj","username":"William","userage":69,"userhp":105},
{"userid":7,"hudi":"komf","username":"Sophia","userage":32,"userhp":106},
{"userid":8,"hudi":"mki","username":"James","userage":64,"userhp":107},
{"userid":9,"hudi":"hjk","username":"Emma","userage":37,"userhp":108},
{"userid":10,"hudi":"hua","username":"Liam","userage":64,"userhp":109}
]
```

You can specify the jsonpaths parameter to match the specified columns:

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "format:json" \
    -H "strip_outer_array:true" \
    -H "jsonpaths:[\"$.userid\", \"$.username\", \"$.userage\"]" \
    -H "columns:user_id,name,age" \
    -T streamload_example.json \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Specifying JSON root node for data load

If the JSON data contains nested JSON fields, you need to specify the root node of the imported JSON. The default value is “”.

For example, with the following data, if you want to import the data in the comment column into the table:

```
[
    {"user":1,"comment":{"userid":101,"username":"Emily","userage":25}},
    {"user":2,"comment":{"userid":102,"username":"Benjamin","userage":35}},
    {"user":3,"comment":{"userid":103,"username":"Olivia","userage":28}},
    {"user":4,"comment":{"userid":104,"username":"Alexander","userage":60}},
```

```
    {"user":5,"comment":{"userid":105,"username":"Ava","userage":17}},
    {"user":6,"comment":{"userid":106,"username":"William","userage":69}},
    {"user":7,"comment":{"userid":107,"username":"Sophia","userage":32}},
    {"user":8,"comment":{"userid":108,"username":"James","userage":64}},
    {"user":9,"comment":{"userid":109,"username":"Emma","userage":37}},
    {"user":10,"comment":{"userid":110,"username":"Liam","userage":64}}
    ]
```

First, you need to specify the root node as comment using the json_root parameter, and then complete the column name mapping according to the jsonpaths parameter.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "format:json" \
    -H "strip_outer_array:true" \
    -H "json_root: $.comment" \
    -H "jsonpaths:[\"$.userid\", \"$.username\", \"$.userage\"]" \
    -H "columns:user_id,name,age" \
    -T streamload_example.json \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Loading array data type

For example, if the following data contains an array type:

```
1|Emily|[1,2,3,4]
2|Benjamin|[22,45,90,12]
3|Olivia|[23,16,19,16]
4|Alexander|[123,234,456]
5|Ava|[12,15,789]
6|William|[57,68,97]
7|Sophia|[46,47,49]
8|James|[110,127,128]
9|Emma|[19,18,123,446]
10|Liam|[89,87,96,12]
```

Load data into the following table structure:

```
CREATE TABLE testdb.test_streamload(
    typ_id     BIGINT          NOT NULL COMMENT "ID",
    name       VARCHAR(20)     NULL     COMMENT "Name",
    arr        ARRAY<int(10)>  NULL     COMMENT "Array"
)
DUPLICATE KEY(typ_id)
DISTRIBUTED BY HASH(typ_id) BUCKETS 10;
```

You can directly load the ARRAY type from a text file into the table using a Stream Load job.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "column_separator:|" \
    -H "columns:typ_id,name,arr" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Loading map data type

When the imported data contains a map type, as in the following example:

```
[
{"user_id":1,"namemap":{"Emily":101,"age":25}},
{"user_id":2,"namemap":{"Benjamin":102,"age":35}},
{"user_id":3,"namemap":{"Olivia":103,"age":28}},
{"user_id":4,"namemap":{"Alexander":104,"age":60}},
{"user_id":5,"namemap":{"Ava":105,"age":17}},
{"user_id":6,"namemap":{"William":106,"age":69}},
{"user_id":7,"namemap":{"Sophia":107,"age":32}},
{"user_id":8,"namemap":{"James":108,"age":64}},
{"user_id":9,"namemap":{"Emma":109,"age":37}},
{"user_id":10,"namemap":{"Liam":110,"age":64}}
]
```

Load data into the following table structure:

```
CREATE TABLE testdb.test_streamload(
    user_id            BIGINT       NOT NULL COMMENT "ID",
    namemap            Map<STRING, INT>  NULL     COMMENT "Name"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

You can directly load the map type from a text file into the table using a Stream Load task.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "format: json" \
    -H "strip_outer_array:true" \
    -T streamload_example.json \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Loading bitmap data type

During the import process, when encountering Bitmap type data, you can use to_bitmap to convert the data into Bitmap, or use the bitmap_empty function to fill the Bitmap.

For example, with the following data:

```
1|koga|17723
2|nijg|146285
3|lojn|347890
4|lofn|489871
5|jfin|545679
6|kon|676724
7|nhga|767689
8|nfubg|879878
9|huang|969798
10|buag|97997
```

Load data into the following table containing the Bitmap type:

```
CREATE TABLE testdb.test_streamload(
    typ_id     BIGINT               NULL   COMMENT "ID",
    hou        VARCHAR(10)          NULL   COMMENT "one",
    arr        BITMAP  BITMAP_UNION  NOT NULL   COMMENT "two"
)
AGGREGATE KEY(typ_id,hou)
DISTRIBUTED BY HASH(typ_id,hou) BUCKETS 10;
```

And use to_bitmap to convert the data into the Bitmap type.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "Expect:100-continue" \
    -H "columns:typ_id,hou,arr,arr=to_bitmap(arr)"
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Loading HyperLogLog data type

You can use the hll_hash function to convert data into the hll type, as in the following example:

```
1001|koga
1002|nijg
1003|lojn
1004|lofn
1005|jfin
1006|kon
1007|nhga
1008|nfubg
1009|huang
1010|buag
```

Load data into the following table:

```
CREATE TABLE testdb.test_streamload(
    typ_id           BIGINT          NULL   COMMENT "ID",
    typ_name         VARCHAR(10)     NULL   COMMENT "NAME",
    pv               hll hll_union   NOT NULL   COMMENT "hll"
)
AGGREGATE KEY(typ_id,typ_name)
DISTRIBUTED BY HASH(typ_id) BUCKETS 10;
```

And use the hll_hash command for import.

```
curl --location-trusted -u <doris_user>:<doris_password> \
    -H "column_separator:|" \
    -H "columns:typ_id,typ_name,pv=hll_hash(typ_id)" \
    -T streamload_example.csv \
    -XPUT http://<fe_ip>:<fe_http_port>/api/testdb/test_streamload/_stream_load
```

Label, loading transaction, multi-table atomicity

All load jobs in Doris are atomically effective. And multiple tables loading in the same load job can also guarantee atomicity. At the same time, Doris can also use the Label mechanism to ensure that data loading is

not lost or duplicated. For specific instructions, please refer to the Import Transactions and Atomicity documentation.

Column mapping, derived columns, and filtering

Doris supports a very rich set of column transformations and filtering operations in load statements. Supports most built-in functions and UDFs. For how to use this feature correctly, please refer to the Data Transformation documentation.

Enable strict mode import

The strict_mode attribute is used to set whether the import task runs in strict mode. This attribute affects the results of column mapping, transformation, and filtering, and it also controls the behavior of partial column updates. For specific instructions on strict mode, please refer to the Strict Mode documentation.

Perform partial column updates during import

For how to express partial column updates during import, please refer to the Data Manipulation/Data Update documentation.

More help

For more detailed syntax and best practices on using Stream Load, please refer to the Stream Load Command Manual. You can also enter HELP STREAM LOAD in the MySql client command line to get more help information.

Stream load submits and transfers data through HTTP protocol. Here, the `curl` command shows how to submit an import.

Users can also operate through other HTTP clients.

```
curl --location-trusted -u user:passwd [-H ""...] -T data.file -XPUT http://fe_host:http_port/api/{db}/{table}/_stream_load

The properties supported in the header are described in "Load Parameters" below
The format is: - H "key1: value1"
```

Examples:

```
curl --location-trusted -u root -T date -H "label:123" http://abc.com:8030/api/test/date/_stream_load
```

The detailed syntax for creating imports helps to execute `HELP STREAM LOAD` view. The following section focuses on the significance of creating some parameters of Stream load.

Signature parameters

- user/passwd

  Stream load uses the HTTP protocol to create the imported protocol and signs it through the Basic Access authentication. The Doris system verifies user identity and import permissions based on signatures.

Load Parameters

Stream load uses HTTP protocol, so all parameters related to import tasks are set in the header. The significance of some parameters of the import task parameters of Stream load is mainly introduced below.

- label

  Identity of import task. Each import task has a unique label inside a single database. Label is a user-defined name in the import command. With this label, users can view the execution of the corresponding import task.

  Another function of label is to prevent users from importing the same data repeatedly. **It is strongly recommended that users use the same label for the same batch of data. This way, repeated requests for the same batch of data will only be accepted once, guaranteeing at-Most-Once**

  When the corresponding import operation state of label is CANCELLED, the label can be used again.

- column_separator

  Used to specify the column separator in the load file. The default is `\t`. If it is an invisible character, you need to add `\x` as a prefix and hexadecimal to indicate the separator.

  For example, the separator `\x01` of the hive file needs to be specified as `-H "column_separator:\x01"`.

  You can use a combination of multiple characters as the column separator.

- line_delimiter

  Used to specify the line delimiter in the load file. The default is `\n`.

  You can use a combination of multiple characters as the column separator.

- max_filter_ratio

  The maximum tolerance rate of the import task is 0 by default, and the range of values is 0-1. When the import error rate exceeds this value, the import fails.

  If the user wishes to ignore the wrong row, the import can be successful by setting this parameter greater than 0.

  The calculation formula is as follows:

  `(dpp.abnorm.ALL / (dpp.abnorm.ALL + dpp.norm.ALL ) ) > max_filter_ratio`

  `dpp.abnorm.ALL` denotes the number of rows whose data quality is not up to standard. Such as type mismatch, column mismatch, length mismatch and so on.

  `dpp.norm.ALL` refers to the number of correct data in the import process. The correct amount of data for the import task can be queried by the `` `SHOW LOAD` `` command.

  The number of rows in the original file = `dpp.abnorm.ALL + dpp.norm.ALL`

- where

Import the filter conditions specified by the task. Stream load supports filtering of where statements specified for raw data. The filtered data will not be imported or participated in the calculation of filter ratio, but will be counted as `num_rows_unselected`.

- partitions

  Partitions information for tables to be imported will not be imported if the data to be imported does not belong to the specified Partition. These data will be included in `dpp.abnorm.ALL`.

- columns

  The function transformation configuration of data to be imported includes the sequence change of columns and the expression transformation, in which the expression transformation method is consistent with the query statement.

  ```
  Examples of column order transformation: There are three columns of original data (src_c1,src_c2,src_c3), and there are
  also three columns  (dst_c1,dst_c2,dst_c3) in the doris table at present.
  when the first column src_c1 of the original file corresponds to the dst_c1 column of the target table, while the second
  column src_c2 of the original file corresponds to the dst_c2 column of the target table and the third column src_c3 of the
  original file corresponds to the dst_c3 column of the target table,which is written as follows:
  columns: dst_c1, dst_c2, dst_c3

  when the first column src_c1 of the original file corresponds to the dst_c2 column of the target table, while the second
  column src_c2 of the original file corresponds to the dst_c3 column of the target table and the third column src_c3 of the
  original file corresponds to the dst_c1 column of the target table,which is written as follows:
  columns: dst_c2, dst_c3, dst_c1

  Example of expression transformation: There are two columns in the original file and two columns in the target table (c1,
  c2). However, both columns in the original file need to be transformed by functions to correspond to the two columns in the
  target table.
  columns: tmp_c1, tmp_c2, c1 = year(tmp_c1), c2 = mouth(tmp_c2)
  Tmp_* is a placeholder, representing two original columns in the original file.
  ```

- format

  Specify the import data format, support csv, json, the default is csv

  :::tip Tips This feature is supported since the Apache Doris 1.2 version :::

  supports `csv_with_names` (csv file line header filter), `csv_with_names_and_types` (csv file first two lines filter), parquet, orc

- exec_mem_limit

  Memory limit. Default is 2GB. Unit is Bytes

- merge_type

  The type of data merging supports three types: APPEND, DELETE, and MERGE. APPEND is the default value, which means that all this batch of data needs to be appended to the existing data. DELETE means to delete all rows with the same key as this batch of data. MERGE semantics Need to be used in conjunction with the delete condition, which means that the data that meets the delete condition is processed according to DELETE semantics and the rest is processed according to APPEND semantics

- two_phase_commit

Stream load import can enable two-stage transaction commit mode: in the stream load process, the data is written and the information is returned to the user. At this time, the data is invisible and the transaction status is `PRECOMMITTED`. After the user manually triggers the commit operation, the data is visible.

- enclose

  When the csv data field contains row delimiters or column delimiters, to prevent accidental truncation, single-byte characters can be specified as brackets for protection. For example, the column separator is ",", the bracket is """, and the data is"a,'b,c'", then "b,c" will be parsed as a field. Note: when the bracket is ", trim_double_quotes must be set to true.

- escape

  Used to escape characters that appear in a csv field identical to the enclosing characters. For example, if the data is "a,'b,'c'", enclose is """, and you want"b,'c to be parsed as a field, you need to specify a single-byte escape character, such as "", and then modify the data to"a,' b,'c'".

  Example：

  1. Initiate a stream load pre-commit operation

```
curl  --location-trusted -u user:passwd -H "two_phase_commit:true" -T test.txt
http://fe_host:http_port/api/{db}/{table}/_stream_load
{
    "TxnId": 18036,
    "Label": "55c8ffc9-1c40-4d51-b75e-f2265b3602ef",
    "TwoPhaseCommit": "true",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 100,
    "NumberLoadedRows": 100,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 1031,
    "LoadTimeMs": 77,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 1,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 58,
    "CommitAndPublishTimeMs": 0
}
```

  2. Trigger the commit operation on the transaction. Note 1) requesting to fe and be both works Note 2) `{table}` in url can be omit when commit using txn id

```
curl -X PUT --location-trusted -u user:passwd  -H "txn_id:18036" -H "txn_operation:commit"
http://fe_host:http_port/api/{db}/{table}/_stream_load_2pc
{
    "status": "Success",
    "msg": "transaction [18036] commit successfully."
}
```

  using label

```
curl -X PUT --location-trusted -u user:passwd  -H "label:55c8ffc9-1c40-4d51-b75e-f2265b3602ef" -H "txn_operation:commit"
http://fe_host:http_port/api/{db}/{table}/_stream_load_2pc
```

```
{
    "status": "Success",
    "msg": "label [55c8ffc9-1c40-4d51-b75e-f2265b3602ef] commit successfully."
}
```

3. Trigger an abort operation on a transaction Note 1) requesting to fe and be both works Note 2) `{table}` in url can be omit when abort using txn id

```
curl -X PUT --location-trusted -u user:passwd  -H "txn_id:18037" -H "txn_operation:abort"
http://fe_host:http_port/api/{db}/{table}/_stream_load_2pc
{
    "status": "Success",
    "msg": "transaction [18037] abort successfully."
}
```

using label

```
curl -X PUT --location-trusted -u user:passwd  -H "label:55c8ffc9-1c40-4d51-b75e-f2265b3602ef" -H "txn_operation:abort"
http://fe_host:http_port/api/{db}/{table}/_stream_load_2pc
{
    "status": "Success",
    "msg": "label [55c8ffc9-1c40-4d51-b75e-f2265b3602ef] abort successfully."
}
```

- enable_profile

  When `enable_profile` is true, the Stream Load profile will be printed to logs (be.INFO).

- partial_columns

  Whether to enable partial column updates, Boolean type, True means that use partial column update, the default value is false, this parameter is only allowed to be set when the table model is Unique and Merge on Write is used.

  eg: `curl --location-trusted -u root: -H "partial_columns:true" -H "column_separator:," -H "columns:id,balance,last_access_time" -T /tmp/test.csv http://127.0.0.1:48037/api/db1/user_profile/_stream_load`

Return results

Since Stream load is a synchronous import method, the result of the import is directly returned to the user by creating the return value of the import.

Examples:

```
{
    "TxnId": 1003,
    "Label": "b6f3bc78-0d2c-45d9-9e4c-faa0a0149bee",
    "Status": "Success",
    "ExistingJobStatus": "FINISHED", // optional
    "Message": "OK",
    "NumberTotalRows": 1000000,
    "NumberLoadedRows": 1000000,
    "NumberFilteredRows": 1,
    "NumberUnselectedRows": 0,
```

```
    "LoadBytes": 40888898,
    "LoadTimeMs": 2144,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 2,
    "ReadDataTimeMs": 325,
    "WriteDataTimeMs": 1933,
    "CommitAndPublishTimeMs": 106,
    "ErrorURL": "http://192.168.1.1:8042/api/_load_error_log?file=__shard_0/error_log_insert_stmt_db18266d4d9b4ee5-
abb00ddd64bdf005_db18266d4d9b4ee5_abb00ddd64bdf005"
}
```

The following main explanations are given for the Stream load import result parameters:

- TxnId: The imported transaction ID. Users do not perceive.
- Label: Import Label. User specified or automatically generated by the system.
- Status: Import completion status.

  “Success”: Indicates successful import.

  “Publish Timeout”: This state also indicates that the import has been completed, except that the data may be delayed and visible without retrying.

  “Label Already Exists”: Label duplicate, need to be replaced Label.

  “Fail”: Import failed.
- ExistingJobStatus: The state of the load job corresponding to the existing Label.

  This field is displayed only when the status is “Label Already Exists”. The user can know the status of the load job corresponding to Label through this state. “RUNNING” means that the job is still executing, and “FINISHED” means that the job is successful.
- Message: Import error messages.
- NumberTotalRows: Number of rows imported for total processing.
- NumberLoadedRows: Number of rows successfully imported.
- NumberFilteredRows: Number of rows that do not qualify for data quality.
- NumberUnselectedRows: Number of rows filtered by where condition.
- LoadBytes: Number of bytes imported.
- LoadTimeMs: Import completion time. Unit milliseconds.
- BeginTxnTimeMs: The time cost for RPC to Fe to begin a transaction, Unit milliseconds.
- StreamLoadPutTimeMs: The time cost for RPC to Fe to get a stream load plan, Unit milliseconds.
- ReadDataTimeMs: Read data time, Unit milliseconds.
- WriteDataTimeMs: Write data time, Unit milliseconds.

- CommitAndPublishTimeMs: The time cost for RPC to Fe to commit and publish a transaction, Unit milliseconds.
- ErrorURL: If you have data quality problems, visit this URL to see specific error lines.

> **Note**
>
> Since Stream load is a synchronous import mode, import information will not be recorded in Doris system. Users cannot see Stream load asynchronously by looking at import commands. You need to listen for the return value of the create import request to get the import result.

Cancel Load

Users can't cancel Stream load manually. Stream load will be cancelled automatically by the system after a timeout or import error.

View Stream Load

Users can view completed stream load tasks through `show stream load`.

By default, BE does not record Stream Load records. If you want to view records that need to be enabled on BE, the configuration parameter is: `enable_stream_load_record=true`. For details, please refer to BE Configuration Items

Relevant System Configuration

FE configuration

- stream_load_default_timeout_second

  The timeout time of the import task (in seconds) will be cancelled by the system if the import task is not completed within the set timeout time, and will become CANCELLED.

  At present, Stream load does not support custom import timeout time. All Stream load import timeout time is uniform. The default timeout time is 600 seconds. If the imported source file can no longer complete the import within the specified time, the FE parameter `stream_load_default_timeout_second` needs to be adjusted.

- enable_pipeline_load

  Whether or not to enable the Pipeline engine to execute Streamload tasks. See the Import documentation.

BE configuration

- streaming_load_max_mb

The maximum import size of Stream load is 10G by default, in MB. If the user's original file exceeds this value, the BE parameter `streaming_load_max_mb` needs to be adjusted.

Best Practices

Application scenarios

The most appropriate scenario for using Stream load is that the original file is in memory or on disk. Secondly, since Stream load is a synchronous import method, users can also use this import if they want to obtain the import results in a synchronous manner.

Data volume

Since Stream load is based on the BE initiative to import and distribute data, the recommended amount of imported data is between 1G and 10G. Since the default maximum Stream load import data volume is 10G, the configuration of BE `streaming_load_max_mb` needs to be modified if files exceeding 10G are to be imported.

```
For example, the size of the file to be imported is 15G
Modify the BE configuration streaming_load_max_mb to 16000
```

Stream load default timeout is 600 seconds, according to Doris currently the largest import speed limit, about more than 3G files need to modify the import task default timeout.

```
Import Task Timeout = Import Data Volume / 10M / s (Specific Average Import Speed Requires Users to Calculate Based on Their Cluster Conditions)
For example, import a 10G file
Timeout = 1000s -31561;. 20110G / 10M /s
```

Complete examples

Data situation: In the local disk path /home/store_sales of the sending and importing requester, the imported data is about 15G, and it is hoped to be imported into the table store_sales of the database bj_sales.

Cluster situation: The concurrency of Stream load is not affected by cluster size.

- Step 1: Does the import file size exceed the default maximum import size of 10G

  ```
  BE conf
  streaming_load_max_mb = 16000
  ```

- Step 2: Calculate whether the approximate import time exceeds the default timeout value

  ```
  Import time 15000/10 = 1500s
  Over the default timeout time, you need to modify the FE configuration
  stream_load_default_timeout_second = 1500
  ```

- Step 3: Create Import Tasks

```
curl --location-trusted -u user:password -T /home/store_sales -H "label:abc"
http://abc.com:8030/api/bj_sales/store_sales/_stream_load
```

Coding with StreamLoad

You can initiate HTTP requests for Stream Load using any language. Before initiating HTTP requests, you need to set several necessary headers:

```
Content-Type: text/plain; charset=UTF-8
Expect: 100-continue
Authorization: Basic <Base64 encoded username and password>
```

`<Base64 encoded username and password>`: a string consist with Doris's `username`, `:` and `password` and then do a base64 encode.

Additionally, it should be noted that if you directly initiate an HTTP request to FE, as Doris will redirect to BE, some frameworks will remove the `Authorization` HTTP header during this process, which requires manual processing.

Doris provides StreamLoad examples in three languages: Java, Go, and Python for reference.

Common Questions

- Label Already Exists

  The Label repeat checking steps of Stream load are as follows:

  1. Is there an import Label conflict that already exists with other import methods?

  Because imported Label in Doris system does not distinguish between import methods, there is a problem that other import methods use the same Label.

  Through `SHOW LOAD WHERE LABEL = "xxx"'`, where XXX is a duplicate Label string, see if there is already a Label imported by FINISHED that is the same as the Label created by the user.

  2. Are Stream loads submitted repeatedly for the same job?

  Since Stream load is an HTTP protocol submission creation import task, HTTP Clients in various languages usually have their own request retry logic. After receiving the first request, the Doris system has started to operate Stream load, but because the result is not returned to the Client side in time, the Client side will retry to create the request. At this point, the Doris system is already operating on the first request, so the second request will be reported to Label Already Exists.

  To sort out the possible methods mentioned above: Search FE Master's log with Label to see if there are two `redirect load action to destination =`redirect load action to destination cases in the same Label. If so, the request is submitted repeatedly by the Client side.

  It is recommended that the user calculate the approximate import time based on the amount of data currently requested, and change the request overtime on the client side to a value greater than the

import timeout time according to the import timeout time to avoid multiple submissions of the request by the client side.

3. Connection reset abnormal

In the community version 0.14.0 and earlier versions, the connection reset exception occurred after Http V2 was enabled, because the built-in web container is tomcat, and Tomcat has pits in 307 (Temporary Redirect). There is a problem with the implementation of this protocol. All In the case of using Stream load to import a large amount of data, a connect reset exception will occur. This is because tomcat started data transmission before the 307 jump, which resulted in the lack of authentication information when the BE received the data request. Later, changing the built-in container to Jetty solved this problem. If you encounter this problem, please upgrade your Doris or disable Http V2 (`enable_http_server_v2=false`).

After the upgrade, also upgrade the http client version of your program to `4.5.13`，Introduce the following dependencies in your pom.xml file

```
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <version>4.5.13</version>
</dependency>
```

- After enabling the Stream Load record on the BE, the record cannot be queried

  This is caused by the slowness of fetching records, you can try to adjust the following parameters:

  1. Increase the BE configuration `stream_load_record_batch_size`. This configuration indicates how many Stream load records can be pulled from BE each time. The default value is 50, which can be increased to 500.
  2. Reduce the FE configuration `fetch_stream_load_record_interval_second`, this configuration indicates the interval for obtaining Stream load records, the default is to fetch once every 120 seconds, and it can be adjusted to 60 seconds.
  3. If you want to save more Stream load records (not recommended, it will take up more resources of FE), you can increase the configuration `max_stream_load_record_size` of FE, the default is 5000.

More Help

For more detailed syntax used by **Stream Load**, you can enter `HELP STREAM LOAD` on the Mysql client command line for more help.

Broker Load

Why introduce Broker Load?

Stream Load is a push-based method, where the data to be imported relies on the client to read and push it to Doris. Broker Load, on the other hand, involves sending an import request to Doris, and Doris actively pulls the data. Therefore, if the data is stored in systems like HDFS or object storage, using Broker Load is the most convenient. This way, the data doesn't need to pass through the client but is directly read and imported by Doris.

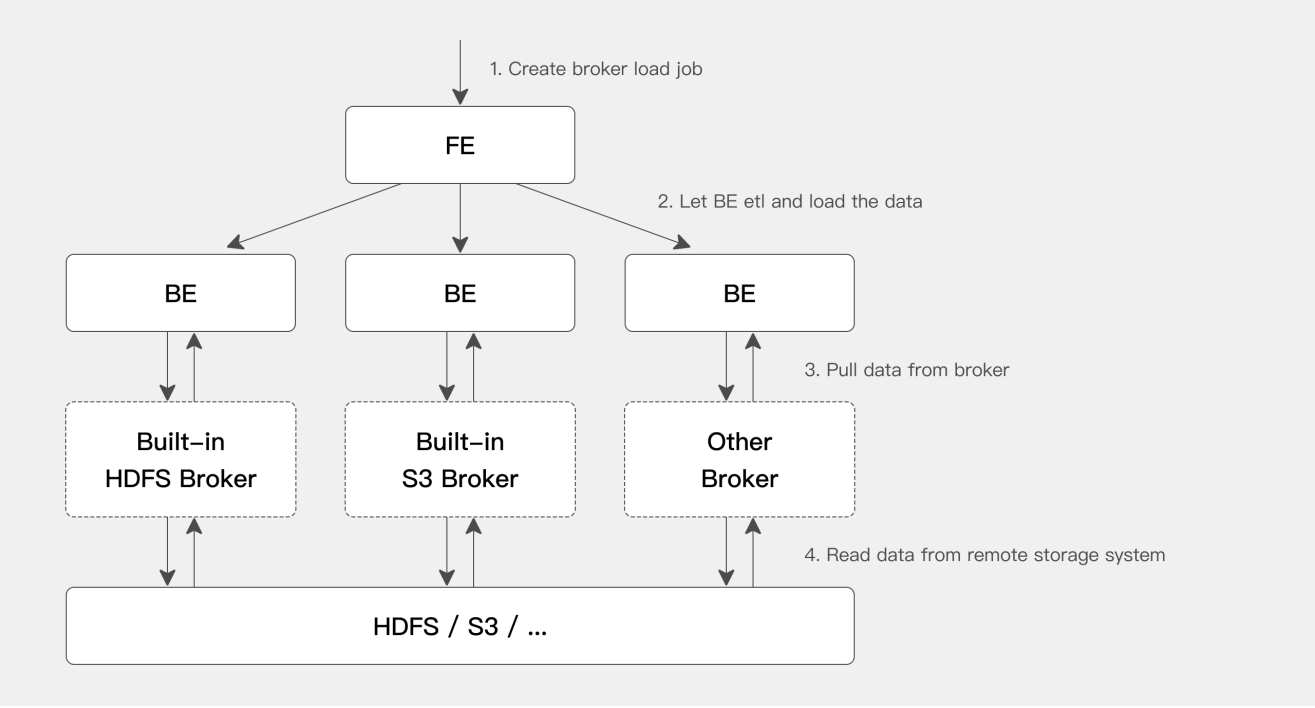
Direct reads from HDFS or S3 can also be imported through HDFS TVF or S3 TVF in the Lakehouse/TVF. The current "Insert Into" based on TVF is a synchronous import, while Broker Load is an asynchronous import method.

Broker Load is suitable for scenarios where the source data is stored in remote storage systems, such as HDFS, and the data volume is relatively large.

Basic Principles

After a user submits an import task, the Frontend (FE) generates a corresponding plan. Based on the current number of Backend (BE) nodes and the size of the file, the plan is distributed to multiple BE nodes for execution, with each BE node handling a portion of the import data.

During execution, the BE nodes pull data from the Broker, perform necessary transformations, and then import the data into the system. Once all BE nodes have completed the import, the FE makes the final determination on whether the import was successful.



Broker Load

As seen in the diagram, BE nodes rely on Broker processes to read data from corresponding remote storage systems. The introduction of Broker processes primarily aims to accommodate different remote storage systems. Users can develop their own Broker processes according to established standards. These Broker processes, which can be developed using Java, offer better compatibility with various storage systems in the big data ecosystem. The separation of Broker processes from BE nodes ensures error isolation between the two, enhancing the stability of the BE.

Currently, BE nodes have built-in support for HDFS and S3 Brokers. Therefore, when importing data from HDFS or S3, there is no need to additionally start a Broker process. However, if a customized Broker implementation is required, the corresponding Broker process needs to be deployed.

SQL syntax for importing

```
LOAD LABEL load_label
(
data_desc1[, data_desc2, ...]
)
WITH [HDFS|S3|BROKER broker_name]
[broker_properties]
[load_properties]
[COMMENT "comments"];
```

For the specific syntax for usage, please refer to BROKER LOAD in the SQL manual.

Checking import status

Broker Load is an asynchronous import method, and the specific import results can be viewed through the SHOW LOAD command.

```
mysql> show load order by createtime desc limit 1\G;
*************************** 1. row ***************************
         JobId: 41326624
         Label: broker_load_2022_04_15
         State: FINISHED
      Progress: ETL:100%; LOAD:100%
          Type: BROKER
       EtlInfo: unselected.rows=0; dpp.abnorm.ALL=0; dp.norm.ALL=27
      TaskInfo: cluster:N/A; timeout(s):1200; max_filter_ratio:0.1
      ErrorMsg: NULL
    CreateTime: 2022-04-01 18:59:06
  EtlStartTime: 2022-04-01 18:59:11
 EtlFinishTime: 2022-04-01 18:59:11
 LoadStartTime: 2022-04-01 18:59:11
LoadFinishTime: 2022-04-01 18:59:11
           URL: NULL
    JobDetails: {"Unfinished backends":{"5072bde59b74b65-8d2c0ee5b029adc0":[]},"ScannedRows":27,"TaskNumber":1,"All backends":
{"5072bde59b74b65-8d2c0ee5b029adc0":[36728051]},"FileNumber":1,"FileSize":5540}
1 row in set (0.01 sec)
```

Cancelling an Import

When the status of a Broker Load job is not CANCELLED or FINISHED, it can be manually cancelled by the user. To cancel, the user needs to specify the label of the import task to be cancelled. The syntax for the cancel

import command can be viewed by executing CANCEL LOAD.

For example: To cancel the import job with the label “broker_load_2022_03_23” on the DEMO database.

```
CANCEL LOAD FROM demo WHERE LABEL = "broker_load_2022_03_23";
```

HDFS Load

Simple Authentication

Simple authentication refers to the configuration of Hadoop where hadoop.security.authentication is set to “simple”.

```
(
    "username" = "user",
    "password" = ""
);
```

The username should be configured as the user to be accessed, and the password can be left blank.

Kerberos Authentication

This authentication method requires the following information:

- **hadoop.security.authentication:** Specifies the authentication method as Kerberos.
- **hadoop.kerberos.principal:** Specifies the Kerberos principal.
- **hadoop.kerberos.keytab:** Specifies the file path of the Kerberos keytab. The file must be an absolute path on the server where the Broker process is located and must be accessible by the Broker process.
- **kerberos_keytab_content:** Specifies the content of the Kerberos keytab file after being encoded in base64. This can be used as an alternative to the kerberos_keytab configuration.

Example configuration:

```
(
    "hadoop.security.authentication" = "kerberos",
    "hadoop.kerberos.principal" = "doris@YOUR.COM",
    "hadoop.kerberos.keytab" = "/home/doris/my.keytab"
)
(
    "hadoop.security.authentication" = "kerberos",
    "hadoop.kerberos.principal" = "doris@YOUR.COM",
    "kerberos_keytab_content" = "ASDOWHDLAWIDJHWLDKSALDJSDIWALD"
)
```

To use Kerberos authentication, the krb5.conf (opens new window) file is required. The krb5.conf file contains Kerberos configuration information. Typically, the krb5.conf file should be installed in the /etc directory. You can override the default location by setting the KRB5_CONFIG environment variable. An example of the krb5.conf file content is as follows:

```
[libdefaults]
    default_realm = DORIS.HADOOP
    default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc
    default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc
    dns_lookup_kdc = true
    dns_lookup_realm = false

[realms]
    DORIS.HADOOP = {
        kdc = kerberos-doris.hadoop.service:7005
    }
```

HDFS HA Mode

This configuration is used to access HDFS clusters deployed in HA (High Availability) mode.

- **dfs.nameservices:** Specifies the name of the HDFS service, which can be customized. For example: “dfs.nameservices” = “my_ha”.
- **dfs.ha.namenodes.xxx:** Customizes the names of the namenodes, with multiple names separated by commas. Here, xxx represents the custom name specified in dfs.nameservices. For example: “dfs.ha.namenodes.my_ha” = “my_nn”.
- **dfs.namenode.rpc-address.xxx.nn:** Specifies the RPC address information for the namenode. In this context, nn represents the namenode name configured in dfs.ha.namenodes.xxx. For example: “dfs.namenode.rpc-address.my_ha.my_nn” = “host:port”.
- **dfs.client.failover.proxy.provider.[nameservice ID]:** Specifies the provider for client connections to the namenode. The default is org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider.

An example configuration is as follows:

```
(
    "fs.defaultFS" = "hdfs://my_ha",
    "dfs.nameservices" = "my_ha",
    "dfs.ha.namenodes.my_ha" = "my_namenode1, my_namenode2",
    "dfs.namenode.rpc-address.my_ha.my_namenode1" = "nn1_host:rpc_port",
    "dfs.namenode.rpc-address.my_ha.my_namenode2" = "nn2_host:rpc_port",
    "dfs.client.failover.proxy.provider.my_ha" = "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"
)
```

HA mode can be combined with the previous two authentication methods for cluster access. For example, accessing HA HDFS through simple authentication:

```
(
    "username"="user",
    "password"="passwd",
    "fs.defaultFS" = "hdfs://my_ha",
    "dfs.nameservices" = "my_ha",
    "dfs.ha.namenodes.my_ha" = "my_namenode1, my_namenode2",
    "dfs.namenode.rpc-address.my_ha.my_namenode1" = "nn1_host:rpc_port",
    "dfs.namenode.rpc-address.my_ha.my_namenode2" = "nn2_host:rpc_port",
```

```
    "dfs.client.failover.proxy.provider.my_ha" = "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"
)
```

Import Example

- Importing TXT Files from HDFS

```
LOAD LABEL demo.label_20220402
(
    DATA INFILE("hdfs://host:port/tmp/test_hdfs.txt")
    INTO TABLE `load_hdfs_file_test`
    COLUMNS TERMINATED BY "\t"
    (id,age,name)
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
)
PROPERTIES
(
    "timeout"="1200",
    "max_filter_ratio"="0.1"
);
```

- HDFS requires the configuration of NameNode HA (High Availability)

```
LOAD LABEL demo.label_20220402
(
    DATA INFILE("hdfs://hafs/tmp/test_hdfs.txt")
    INTO TABLE `load_hdfs_file_test`
    COLUMNS TERMINATED BY "\t"
    (id,age,name)
)
with HDFS
(
    "hadoop.username" = "user",
    "fs.defaultFS"="hdfs://hafs",
    "dfs.nameservices" = "hafs",
    "dfs.ha.namenodes.hafs" = "my_namenode1, my_namenode2",
    "dfs.namenode.rpc-address.hafs.my_namenode1" = "nn1_host:rpc_port",
    "dfs.namenode.rpc-address.hafs.my_namenode2" = "nn2_host:rpc_port",
    "dfs.client.failover.proxy.provider.hafs" = "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"
)
PROPERTIES
(
    "timeout"="1200",
    "max_filter_ratio"="0.1"
);
```

- Importing data from HDFS using wildcards to match two batches of files and importing them into two separate tables

```
LOAD LABEL example_db.label2
(
    DATA INFILE("hdfs://host:port/input/file-10*")
    INTO TABLE `my_table1`
    PARTITION (p1)
```

```
    COLUMNS TERMINATED BY ","
    (k1, tmp_k2, tmp_k3)
    SET (
        k2 = tmp_k2 + 1,
        k3 = tmp_k3 + 1
    ),
    DATA INFILE("hdfs://host:port/input/file-20*")
    INTO TABLE `my_table2`
    COLUMNS TERMINATED BY ","
    (k1, k2, k3)
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
);
```

To import two batches of files matching the wildcards `file-10*` and `file-20*` from HDFS and load them into two separate tables `my_table1` and `my_table2`. In this case, my_table1 specifies that the data should be imported into partition p1, and the values in the second and third columns of the source files should be incremented by 1 before being imported.

- Import a batch of data from HDFS using wildcards

```
LOAD LABEL example_db.label3
(
    DATA INFILE("hdfs://host:port/user/doris/data/*/*")
    INTO TABLE `my_table`
    COLUMNS TERMINATED BY "\\x01"
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
);
```

To specify the delimiter as the commonly used default delimiter for Hive, which is , and to use the wildcard character * to refer to all files in all directories under the data directory.

- Import Parquet format data and specify the FORMAT as `parquet`

```
LOAD LABEL example_db.label4
(
    DATA INFILE("hdfs://host:port/input/file")
    INTO TABLE `my_table`
    FORMAT AS "parquet"
    (k1, k2, k3)
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
);
```

The default method is to determine by file extension.

- Import the data and extract the partition field from the file path

```
LOAD LABEL example_db.label5
(
    DATA INFILE("hdfs://host:port/input/city=beijing/*/*")
    INTO TABLE `my_table`
    FORMAT AS "csv"
    (k1, k2, k3)
    COLUMNS FROM PATH AS (city, utc_date)
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
);
```

The columns in the `my_table` are `k1`, `k2`, `k3`, `city`, and `utc_date`.

The directory `hdfs://hdfs_host:hdfs_port/user/doris/data/input/dir/city=beijing` contains the following files:

```
hdfs://hdfs_host:hdfs_port/input/city=beijing/utc_date=2020-10-01/0000.csv
hdfs://hdfs_host:hdfs_port/input/city=beijing/utc_date=2020-10-02/0000.csv
hdfs://hdfs_host:hdfs_port/input/city=tianji/utc_date=2020-10-03/0000.csv
hdfs://hdfs_host:hdfs_port/input/city=tianji/utc_date=2020-10-04/0000.csv
```

The file only contains three columns of data:`k1`,`k2`, and `k3`. The other two columns,`city` and `utc_date`, will be extracted from the file path.

- Filter the imported data

```
LOAD LABEL example_db.label6
(
    DATA INFILE("hdfs://host:port/input/file")
    INTO TABLE `my_table`
    (k1, k2, k3)
    SET (
        k2 = k2 + 1
    )
    PRECEDING FILTER k1 = 1
    WHERE k1 > k2
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
);
```

Only the rows where k1 = 1 in the original data and k1 > k2 after transformation will be imported.

- Import data and extract the time partition field from the file path.

```
LOAD LABEL example_db.label7
(
    DATA INFILE("hdfs://host:port/user/data/*/test.txt")
    INTO TABLE `tbl12`
    COLUMNS TERMINATED BY ","
    (k2,k3)
    COLUMNS FROM PATH AS (data_time)
    SET (
        data_time=str_to_date(data_time, '%Y-%m-%d %H%%3A%i%%3A%s')
```

```
    )
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username" = "user"
);
```

> **Tip**
>
> The time contains “%3A”. In HDFS paths, colons “:” are not allowed, so all colons are replaced with “%3A”.

There are the following files under the path:

```
/user/data/data_time=2020-02-17 00%3A00%3A00/test.txt
/user/data/data_time=2020-02-18 00%3A00%3A00/test.txt
```

The table structure is as follows:

```
CREATE TABLE IF NOT EXISTS tbl12 (
    data_time DATETIME,
    k2        INT,
    k3        INT
) DISTRIBUTED BY HASH(data_time) BUCKETS 10
PROPERTIES (
    "replication_num" = "3"
);
```

- Use Merge mode for import

```
LOAD LABEL example_db.label8
(
    MERGE DATA INFILE("hdfs://host:port/input/file")
    INTO TABLE `my_table`
    (k1, k2, k3, v2, v1)
    DELETE ON v2 > 100
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username"="user"
)
PROPERTIES
(
    "timeout" = "3600",
    "max_filter_ratio" = "0.1"
);
```

To use Merge mode for import, the “my_table” must be a Unique Key table. When the value of the “v2” column in the imported data is greater than 100, that row will be considered a deletion row. The timeout for the import task is 3600 seconds, and an error rate of up to 10% is allowed.

- Specify the “source_sequence” column during import to ensure the order of replacements.

```
LOAD LABEL example_db.label9
(
    DATA INFILE("hdfs://host:port/input/file")
    INTO TABLE `my_table`
    COLUMNS TERMINATED BY ","
    (k1,k2,source_sequence,v1,v2)
    ORDER BY source_sequence
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username"="user"
);
The "my_table" must be a Unique Key model table and have a specified Sequence column. The data will maintain its order
based on the values in the "source_sequence" column in the source data.
```

- Import the specified file format as `json`, and specify the `json_root` and jsonpaths accordingly.

```
LOAD LABEL example_db.label10
(
    DATA INFILE("hdfs://host:port/input/file.json")
    INTO TABLE `my_table`
    FORMAT AS "json"
    PROPERTIES(
      "json_root" = "$.item",
      "jsonpaths" = "[\"$.id\", \"$.city\", \"$.code\"]"
    )
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username"="user"
);
```

The `jsonpaths` can also be used in conjunction with the column list and `SET (column_mapping)` :

```
LOAD LABEL example_db.label10
(
    DATA INFILE("hdfs://host:port/input/file.json")
    INTO TABLE `my_table`
    FORMAT AS "json"
    (id, code, city)
    SET (id = id * 10)
    PROPERTIES(
      "json_root" = "$.item",
      "jsonpaths" = "[\"$.id\", \"$.city\", \"$.code\"]"
    )
)
with HDFS
(
  "fs.defaultFS"="hdfs://host:port",
  "hadoop.username"="user"
);
```

S3 Load

Doris supports importing data directly from object storage systems that support the S3 protocol through the S3 protocol. Here, we mainly introduce how to import data stored in AWS S3. For importing data from other object

storage systems that support the S3 protocol, you can refer to the steps for AWS S3.

Preparation

- AK and SK: First, you need to find or regenerate your AWS `Access Keys`. You can find instructions on how to generate them in the AWS console under `My Security Credentials`.
- REGION and ENDPOINT: The REGION can be selected when creating a bucket or viewed in the bucket list. The S3 ENDPOINT for each REGION can be found in the AWS documentation.

Import example

```
LOAD LABEL example_db.example_label_1
(
    DATA INFILE("s3://your_bucket_name/your_file.txt")
    INTO TABLE load_test
    COLUMNS TERMINATED BY ","
)
WITH S3
(
    "AWS_ENDPOINT" = "AWS_ENDPOINT",
    "AWS_ACCESS_KEY" = "AWS_ACCESS_KEY",
    "AWS_SECRET_KEY"="AWS_SECRET_KEY",
    "AWS_REGION" = "AWS_REGION"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

Common Issues

- The S3 SDK defaults to using the virtual-hosted style method for accessing objects. However, some object storage systems may not have enabled or supported the virtual-hosted style access. In such cases, we can add the `use_path_style` parameter to force the use of the path style method:

  ```
  WITH S3
  (
        "AWS_ENDPOINT" = "AWS_ENDPOINT",
        "AWS_ACCESS_KEY" = "AWS_ACCESS_KEY",
        "AWS_SECRET_KEY"="AWS_SECRET_KEY",
        "AWS_REGION" = "AWS_REGION",
        "use_path_style" = "true"
  )
  ```

- Support for accessing all object storage systems that support the S3 protocol using temporary credentials (TOKEN) is available. The usage is as follows:

  ```
  WITH S3
  (
        "AWS_ENDPOINT" = "AWS_ENDPOINT",
        "AWS_ACCESS_KEY" = "AWS_TEMP_ACCESS_KEY",
        "AWS_SECRET_KEY" = "AWS_TEMP_SECRET_KEY",
        "AWS_TOKEN" = "AWS_TEMP_TOKEN",
  ```

```
        "AWS_REGION" = "AWS_REGION"
    )
```

Importing Data Using Other Brokers

The Broker for other remote storage systems is an optional process in the Doris cluster, primarily used to support Doris in reading and writing files and directories on remote storage. Currently, the following storage system Broker implementations are provided:

- Alibaba Cloud OSS
- Baidu Cloud BOS
- Tencent Cloud CHDFS
- Tencent Cloud GFS
- Huawei Cloud OBS
- JuiceFS
- Google Cloud Storage (GCS)

The Broker provides services through an RPC service port and operates as a stateless Java process. Its primary responsibility is to encapsulate POSIX-like file operations for remote storage, such as open, pread, pwrite, and more. Additionally, the Broker does not keep track of any other information, which means that all the connection details, file information, and permission details related to the remote storage must be passed to the Broker process through parameters during RPC calls. This ensures that the Broker can correctly read and write files.

The Broker serves solely as a data pathway and does not involve any computational tasks, thus requiring minimal memory usage. Typically, a Doris system would deploy one or more Broker processes. Furthermore, Brokers of the same type are grouped together and assigned a unique name (Broker name).

This section primarily focuses on the parameters required by the Broker when accessing different remote storage systems, such as connection information, authentication details, and more. Understanding and correctly configuring these parameters is crucial for successful and secure data exchange between Doris and the remote storage systems.

Broker Information

The information of the Broker consists of two parts: the name (Broker name) and the authentication information. The usual syntax format is as follows:

```
WITH BROKER "broker_name"
(
    "username" = "xxx",
    "password" = "yyy",
    "other_prop" = "prop_value",
    ...
);
```

Broker Name

Typically, users need to specify an existing Broker Name through the `WITH BROKER "broker_name"` clause in the operation command. The Broker Name is a name designated by the user when adding a Broker process through the `ALTER SYSTEM ADD BROKER` command. One name usually corresponds to one or more Broker processes. Doris will select an available Broker process based on the name. Users can view the Brokers that currently exist in the cluster through the `SHOW BROKER` command.

> **Note**
>
> The Broker Name is merely a user-defined name and does not represent the type of Broker.

Authentication Information Different Broker types and access methods require different authentication information. The authentication information is usually provided in the Property Map in a Key-Value format after `WITH BROKER "broker_name"`.

Broker Examples

- Alibaba Cloud OSS

```
(
    "fs.oss.accessKeyId" = "",
    "fs.oss.accessKeySecret" = "",
    "fs.oss.endpoint" = ""
)
```

- JuiceFS

```
(
    "fs.defaultFS" = "jfs://xxx/",
    "fs.jfs.impl" = "io.juicefs.JuiceFileSystem",
    "fs.AbstractFileSystem.jfs.impl" = "io.juicefs.JuiceFS",
    "juicefs.meta" = "xxx",
    "juicefs.access-log" = "xxx"
)
```

- GCS

  When using a Broker to access GCS, the Project ID is required, while other parameters are optional. Please refer to the GCS Config for all parameter configurations.

```
(
    "fs.gs.project.id" = "Your Project ID",
    "fs.AbstractFileSystem.gs.impl" = "com.google.cloud.hadoop.fs.gcs.GoogleHadoopFS",
    "fs.gs.impl" = "com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem",
)
```

Related Configurations

The following configurations belong to the system-level settings for Broker load, which affect all Broker load import tasks. These configurations can be adjusted by modifying the `fe.conf`file.

min_bytes_per_broker_scanner

- Default: 64MB.
- The minimum amount of data processed by a single BE in a Broker Load job.

max_bytes_per_broker_scanner

- Default: 500GB.
- The maximum amount of data processed by a single BE in a Broker Load job.

Typically, the maximum supported data volume for an import job is `max_bytes_per_broker_scanner * the number of BE nodes`. If you need to import a larger volume of data, you may need to adjust the size of the `max_bytes_per_broker_scanner` parameter appropriately.

max_broker_concurrency

- Default: 10.
- Limits the maximum concurrency of imports for a job.
- The minimum processed data volume, maximum concurrency, size of the source file, and the current number of BE nodes jointly determine the concurrency of this import.

```
Import Concurrency = Math.min(Source File Size / min_bytes_per_broker_scanner, max_broker_concurrency, Current Number of BE Nodes * load_parallelism)
Processing Volume per BE for this Import = Source File Size / Import Concurrency
```

Common Issues

1. Import Error: `Scan bytes per broker scanner exceed limit:xxx`

Please refer to the best practices section in the documentation and modify the FE configuration items `max_bytes_per_broker_scanner` and `max_broker_concurrency`.

2. Import Error: : `failed to send batch` **or** `TabletWriter add batch with unknown id`

Appropriately adjust the `query_timeout` and `streaming_load_rpc_max_alive_time_sec` settings.

3. Import Error: `LOAD_RUN_FAIL; msg:Invalid Column Name:xxx`

For PARQUET or ORC format data, the column names in the file header must match the column names in the Doris table. For example:

```
(tmp_c1,tmp_c2)
SET
(
    id=tmp_c2,
    name=tmp_c1
)
```

This represents fetching columns named (tmp_c1, tmp_c2) in the parquet or orc file and mapping them to the (id, name) columns in the Doris table. If no set is specified, the columns in the file header will be used for

mapping.

> **Note**
>
> If ORC files are generated directly using certain Hive versions, the column headers in the ORC file may not be the Hive metadata, but (_col0, _col1, _col2, …), which may lead to the Invalid Column Name error. In this case, mapping using SET is necessary.

5. Import Error: `Failed to get S3 FileSystem for bucket is null/empty`

The bucket information is incorrect or does not exist. Or the bucket format is not supported. When creating a bucket name with an underscore using GCS, such as `s3://gs_bucket/load_tbl`, the S3 Client may report an error when accessing GCS. It is recommended not to use underscores when creating bucket paths.

6. Import Timeout

The default timeout for imports is 4 hours. If a timeout occurs, it is not recommended to directly increase the maximum import timeout to solve the problem. If the single import time exceeds the default import timeout of 4 hours, it is best to split the file to be imported and perform multiple imports to solve the problem. Setting an excessively long timeout time can lead to high costs for retrying failed imports.

You can calculate the expected maximum import file data volume for the Doris cluster using the following formula:

Expected Maximum Import File Data Volume = 14400s * 10M/s * Number of BEs

For example, if the cluster has 10 BEs:

Expected Maximum Import File Data Volume = 14400s * 10M/s * 10 = 1440000M ≈ 1440G

> **Note**
>
> In general, user environments may not reach speeds of 10M/s, so it is recommended to split files exceeding 500G before importing.

More Help

For more detailed syntax and best practices for using Broker Load , please refer to the Broker Load command manual. You can also enter HELP BROKER LOAD in the MySQL client command line to obtain more help information.

Routine Load

Doris can continuously consume data from Kafka Topic through the Routine Load method. After submitting a Routine Load job, Doris will continuously run the load job, generating real-time loading tasks to constantly consume messages from the specified Topic in the Kafka cluster.

Routine Load is a streaming load job that supports Exactly-Once semantics, ensuring that data is neither lost nor duplicated.

Usage Scenarios

Supported Data File Formats

Routine Load supports consuming data in CSV and JSON formats from Kafka.

When loading CSV format, it is necessary to clearly distinguish between null values and empty strings:

- Null values need to be represented with `\n`. For example, `a,\n,b` indicates that the middle column is a null value.
- Empty strings can be represented by leaving the data field empty. For example, `a,,b` indicates that the middle column is an empty string.

Usage Limitations

When using Routine Load to consume data from Kafka, there are the following limitations:

- It supports unauthenticated Kafka access as well as Kafka clusters authenticated through SSL.
- The supported message formats are CSV and JSON text formats. Each message in CSV should be on a separate line, and the line should not end with a newline character.
- By default, it supports Kafka versions 0.10.0.0 and above. If you need to use a Kafka version below 0.10.0.0 (such as 0.9.0, 0.8.2, 0.8.1, 0.8.0), you need to modify the BE configuration by setting the value of `kafka_broker_version_fallback` to the compatible older version, or directly set the value of `property.broker.version.fallback` when creating the Routine Load. However, using an older version may mean that some new features of Routine Load, such as setting the offset of Kafka partitions based on time, may not be available.

Basic Principles

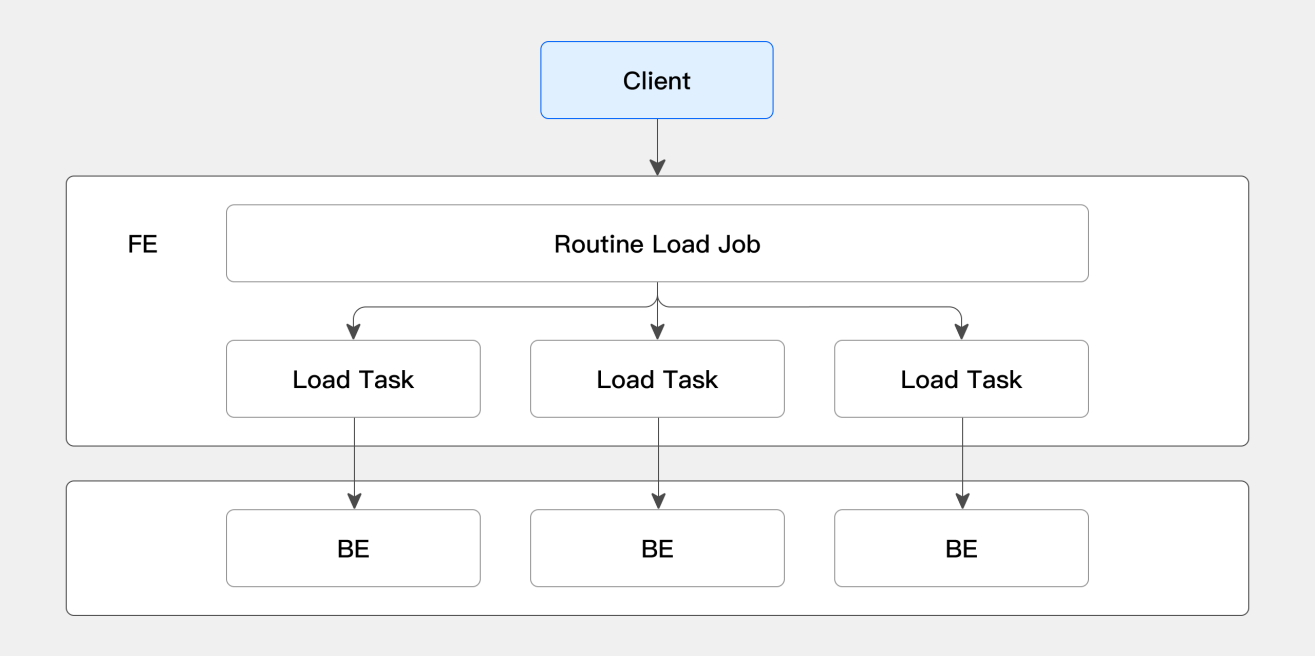
Routine Load continuously consumes data from Kafka Topic and writes it into Doris.

When a Routine Load job is created in Doris, it generates a persistent load job and several load tasks:

- Load Job: Each routine load corresponds to a load job. The load job is a persistent task that continuously consumes data from the Kafka Topic.

- Load Task: A load job is divided into several load tasks, which are loaded as independent basic units using the Stream Load method into BE.

The specific process of Routine Load is illustrated in the following diagram:



Routine Load

1. The Client submits a Routine Load job to the FE to establish a persistent Routine Load Job.
2. The FE splits the Routine Load Job into multiple Routine Load Tasks through the Job Scheduler.
3. On the BE, each Routine Load Task is treated as a Stream Load task for importation and reports back to the FE upon completion.
4. The Job Scheduler in the FE generates new Tasks based on the report results or retries failed Tasks.
5. The Routine Load Job continuously generates new Tasks to complete uninterrupted data importation.

Quick Start

Create Job

In Doris, you can create persistent Routine Load tasks using the `CREATE ROUTINE LOAD` command. For detailed syntax, please refer to CREATE ROUTINE LOAD. Routine Load supports consuming data in CSV and JSON formats.

Loading CSV Data

1. Loading data sample

   In Kafka, there is the following sample data:

```
kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test-routine-load-csv --from-beginnin
1,Emily,25
2,Benjamin,35
```

```
3,Olivia,28
4,Alexander,60
5,Ava,17
6,William,69
7,Sophia,32
8,James,64
9,Emma,37
10,Liam,64
```

2. Creating table

   In Doris, create the table for loading with the following syntax:

```
CREATE TABLE testdb.test_streamload(
    user_id            BIGINT       NOT NULL COMMENT "User ID",
    name               VARCHAR(20)           COMMENT "User Name",
    age                INT                   COMMENT "User Age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

3. Creating the Routine Load job

   In Doris, use the `CREATE ROUTINE LOAD` command to create the load job:

```
CREATE ROUTINE LOAD testdb.example_routine_load_csv ON test_routineload_tbl
COLUMNS TERMINATED BY ",",
COLUMNS(user_id, name, age)
FROM KAFKA(
    "kafka_broker_list" = "192.168.88.62:9092",
    "kafka_topic" = "test-routine-load-csv",
    "property.kafka_default_offsets" = "OFFSET_BEGINNING"
);
```

Loading JSON Data

1. Loading sample data

   In Kafka, there is the following sample data:

```
kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test-routine-load-json --from-beginning
```

2. Creating table

   In Doris, create the table for loading with the following syntax:

```
CREATE TABLE testdb.test_streamload(
    user_id            BIGINT       NOT NULL COMMENT "User ID",
    name               VARCHAR(20)           COMMENT "User Name",
    age                INT                   COMMENT "User Age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

3. Creating the Routine Load job

In Doris, use the `CREATE ROUTINE LOAD` command to create the job:

```
CREATE ROUTINE LOAD testdb.example_routine_load_json ON test_routineload_tbl
COLUMNS(user_id, name, age)
PROPERTIES(
    "format"="json",
    "jsonpaths"="[\"$.user_id\",\"$.name\",\"$.age\"]"
)
FROM KAFKA(
    "kafka_broker_list" = "192.168.88.62:9092"
);
```

Viewing Status

In Doris, you can check the status of Routine Load jobs and tasks using the following methods:

- Load Jobs: Used to view information about load tasks, such as the target table, number of subtasks, load delay status, load configuration, and load results.
- Load Tasks: Used to view the status of individual load tasks, including task ID, transaction status, task status, execution start time, and BE (Backend) node assignment.

01 Viewing Running Jobs

You can use the `SHOW ROUTINE LOAD` command to check the status of jobs. The `SHOW ROUTINE LOAD` command provides information about the current job, including the target table, load delay status, load configuration, and error messages.

For example, to view the status of the `testdb.example_routine_load_csv` job, you can run the following command:

```
mysql> SHOW ROUTINE LOAD FOR testdb.example_routine_load\G
*************************** 1. row ***************************
                  Id: 12025
                Name: example_routine_load
          CreateTime: 2024-01-15 08:12:42
           PauseTime: NULL
             EndTime: NULL
              DbName: default_cluster:testdb
           TableName: test_routineload_tbl
        IsMultiTable: false
               State: RUNNING
      DataSourceType: KAFKA
      CurrentTaskNum: 1
       JobProperties:
{"max_batch_rows":"200000","timezone":"America/New_York","send_batch_parallelism":"1","load_to_single_tablet":"false","column_separa
tor":"','","line_delimiter":"\n","current_concurrent_number":"1","delete":"*","partial_columns":"false","merge_type":"APPEND","exec_
mem_limit":"2147483648","strict_mode":"false","jsonpaths":"","max_batch_interval":"10","max_batch_size":"104857600","fuzzy_parse":"f
alse","partitions":"*","columnToColumnExpr":"user_id,name,age","whereExpr":"*","desired_concurrent_number":"5","precedingFilter":"*"
,"format":"csv","max_error_number":"0","max_filter_ratio":"1.0","json_root":"","strip_outer_array":"false","num_as_string":"false"}
DataSourceProperties: {"topic":"test-topic","currentKafkaPartitions":"0","brokerList":"192.168.88.62:9092"}
    CustomProperties: {"kafka_default_offsets":"OFFSET_BEGINNING","group.id":"example_routine_load_73daf600-884e-46c0-a02b-
4e49fdf3b4dc"}
           Statistic: {"receivedBytes":28,"runningTxns":
[],"errorRows":0,"committedTaskNum":3,"loadedRows":3,"loadRowsRate":0,"abortedTaskNum":0,"errorRowsAfterResumed":0,"totalRows":3,"un
selectedRows":0,"receivedBytesRate":0,"taskExecuteTimeMs":30069}
            Progress: {"0":"2"}
                 Lag: {"0":0}
ReasonOfStateChanged:
```

```
        ErrorLogUrls:
            OtherMsg:
                User: root
             Comment:
1 row in set (0.00 sec)
```

02 Viewing Running Tasks

You can use the `SHOW ROUTINE LOAD TASK` command to check the status of load tasks. The `SHOW ROUTINE LOAD TASK` command provides information about the individual tasks under a specific load job, including task ID, transaction status, task status, execution start time, and BE ID.

For example, to view the task status of the `example_routine_load_csv` job, you can run the following command:

```
mysql> SHOW ROUTINE LOAD TASK WHERE jobname = 'example_routine_load_csv';
+----------------------------------+-------+-----------+-------+---------------------+---------------------+---------+-------+-----
-----------------+
| TaskId                           | TxnId | TxnStatus | JobId | CreateTime          | ExecuteStartTime    | Timeout | BeId  |
DataSourceProperties |
+----------------------------------+-------+-----------+-------+---------------------+---------------------+---------+-------+-----
-----------------+
| 8cf47e6a68ed4da3-8f45b431db50e466 | 195   | PREPARE   | 12177 | 2024-01-15 12:20:41 | 2024-01-15 12:21:01 | 20      | 10429 |
{"4":1231,"9":2603}  |
| f2d4525c54074aa2-b6478cf8daaeb393 | 196   | PREPARE   | 12177 | 2024-01-15 12:20:41 | 2024-01-15 12:21:01 | 20      | 12109 |
{"1":1225,"6":1216}  |
| cb870f1553864250-975279875a25fab6 | -1    | NULL      | 12177 | 2024-01-15 12:20:52 | NULL                | 20      | -1    |
{"2":7234,"7":4865}  |
| 68771fd8a1824637-90a9dac2a7a0075e | -1    | NULL      | 12177 | 2024-01-15 12:20:52 | NULL                | 20      | -1    |
{"3":1769,"8":2982}  |
| 77112dfea5e54b0a-a10eab3d5b19e565 | 197   | PREPARE   | 12177 | 2024-01-15 12:21:02 | 2024-01-15 12:21:02 | 20      | 12098 |
{"0":3000,"5":2622}  |
+----------------------------------+-------+-----------+-------+---------------------+---------------------+---------+-------+-----
-----------------+
```

Pausing Jobs

You can pause an load job using the PAUSE ROUTINE LOAD command. When a job is paused, it enters the PAUSED state, but the load job is not terminated and can be resumed using the RESUME ROUTINE LOAD command.

To pause the `testdb.example_routine_load_csv` load job, you can use the following command:

```
PAUSE ROUTINE LOAD FOR testdb.example_routine_load_csv;
```

Resuming Jobs

You can resume a paused load job using the RESUME ROUTINE LOAD command.

To resume the `testdb.example_routine_load_csv` job, you can use the following command:

```
RESUME ROUTINE LOAD FOR testdb.example_routine_load_csv;
```

Modifying Jobs

You can modify a created loading job using the ALTER ROUTINE LOAD command. Before modifying the job, you need to pause it using the`PAUSE ROUTINE LOAD` command, and after making the modifications, you can resume it using the `RESUME ROUTINE LOAD` command.

To modify the `desired_concurrent_number` parameter for the job and update the Kafka topic information, you can use the following command:

```
ALTER ROUTINE LOAD FOR testdb.example_routine_load_csv
PROPERTIES(
    "desired_concurrent_number" = "3"
)
FROM KAFKA(
    "kafka_broker_list" = "192.168.88.60:9092",
    "kafka_topic" = "test-topic"
);
```

Canceling Jobs

You can stop and delete a Routine Load job using the STOP ROUTINE LOAD command. Once deleted, the load job cannot be recovered and cannot be viewed using the `SHOW ROUTINE LOAD` command.

To stop and delete the `testdb.example_routine_load_csv` load job, you can use the following command:

```
STOP ROUTINE LOAD FOR testdb.example_routine_load_csv;
```

Reference Manual

Load Commands

The syntax for creating a Routine Load persistent load job is as follows:

```
CREATE ROUTINE LOAD [<db_name>.]<job_name> [ON <tbl_name>]
[merge_type]
[load_properties]
[job_properties]
FROM KAFKA [data_source_properties]
[COMMENT "<comment>"]
```

The modules for creating a loading job are explained as follows:

Module	Description
db_name	Specifies the name of the database for creating the loading task.
job_name	Specifies the name of the created loading job. The job name must be unique within the same database.
tbl_name	Specifies the name of the table to be loaded. This parameter is optional. If not specified, the dynamic table mode will be used, where Kafka data should contain the table name information.
merge_type	Specifies the data merge type. The default value is APPEND. Possible merge_type options are:

Module	Description
load_properties	Describes the load properties, including:
job_properties	Specifies the general load parameters for Routine Load.
data_source_properties	Describes the properties of Kafka data source.
comment	Describes any additional comments for the loading job.

Load Parameter Description

01 FE Configuration Parameters

max_routine_load_task_concurrent_num

- Default Value: 256
- Dynamic Configuration: Yes
- FE Master Exclusive: Yes
- Parameter Description: Limits the maximum number of concurrent subtasks for Routine Load jobs. It is recommended to keep it at the default value. Setting it too high may result in excessive concurrent tasks and resource consumption.

max_routine_load_task_num_per_be

- Default Value: 1024
- Dynamic Configuration: Yes
- FE Master Exclusive: Yes
- Parameter Description: Limits the maximum number of concurrent Routine Load tasks per backend (BE). `max_routine_load_task_num_per_be` should be smaller than the `routine_load_thread_pool_size` parameter.

max_routine_load_job_num

- Default Value: 100
- Dynamic Configuration: Yes
- FE Master Exclusive: Yes
- Parameter Description: Limits the maximum number of Routine Load jobs, including those in NEED_SCHEDULED, RUNNING, and PAUSE states.

max_tolerable_backend_down_num

- Default Value: 0
- Dynamic Configuration: Yes
- FE Master Exclusive: Yes

- Parameter Description: If any BE goes down, Routine Load cannot automatically recover. Under certain conditions, Doris can reschedule PAUSED tasks and transition them to the RUNNING state. Setting this parameter to 0 means that re-scheduling is only allowed when all BE nodes are in the alive state.

period_of_auto_resume_min

- Default Value: 5 (minutes)
- Dynamic Configuration: Yes
- FE Master Exclusive: Yes
- Parameter Description: The period for automatically resuming Routine Load.

02 BE Configuration Parameters

max_consumer_num_per_group

- Default Value: 3
- Dynamic Configuration: Yes
- Description: Specifies the maximum number of consumers generated per subtask. For Kafka data sources, a consumer can consume one or multiple Kafka partitions. For example, if a task needs to consume 6 Kafka partitions, it will generate 3 consumers, with each consumer consuming 2 partitions. If there are only 2 partitions, it will generate 2 consumers, with each consumer consuming 1 partition.

Load Configuration Parameters

When creating a Routine Load job, you can specify the load configuration parameters for different modules using the `CREATE ROUTINE LOAD` command.

tbl_name Clause

Specifies the name of the table to be loaded. This parameter is optional.

If not specified, the dynamic table mode is used, which requires the data in Kafka to contain the table name information. Currently, only extracting the table name from the Value field of Kafka is supported. The format should be as follows, using JSON as an example: `table_name|{"col1": "val1", "col2": "val2"}`, where `tbl_name` is the table name and `|` is used as the separator between the table name and the table data. The same format applies to CSV data, such as `table_name|val1,val2,val3`. Note that the `table_name` here must be consistent with the table name in Doris, otherwise the load will fail. Note that dynamic tables do not support the column_mapping configuration described later.

merge_type Clause

The merge_type module specifies the type of data merging. There are three options for merge_type:

- APPEND: Append load mode.
- MERGE: Merge load mode. Only applicable to Unique Key models. It needs to be used together with the [DELETE ON] module to mark the Delete Flag column.

- DELETE: All loaded data is data that needs to be deleted.

load_properties Clause

The load_properties module describes the properties of the loaded data using the following syntax:

```
[COLUMNS TERMINATED BY <column_separator>,]
[COLUMNS (<column1_name>[, <column2_name>, <column_mapping>, ...]),]
[WHERE <where_expr>,]
[PARTITION(<partition1_name>, [<partition2_name>, <partition3_name>, ...]),]
[DELETE ON <delete_expr>,]
[ORDER BY <order_by_column1>[, <order_by_column2>, <order_by_column3>, ...]]
```

The specific parameters for each module are as follows:

Submodule	Parameter	Description
COLUMNS TERMINATED BY	`<column_separator>`	Specifies the column delimiter, defaulting to `\t`. For example, to specify a comma as the delimiter, use `COLUMNS TERMINATED BY ","`. When handling empty values, note the following:
COLUMNS	`<column_name>`	Specifies the corresponding column names. For example, to specify the load columns as `(k1, k2, k3)`, use `COLUMNS(k1, k2, k3)`. The COLUMNS clause can be omitted in the following cases:
	`<column_mapping>`	During the load process, column mapping can be used to filter and transform columns. For example, if the target column needs to perform a derived calculation based on a column in the data source (e.g., the target column k4 is calculated as k3 + 1 based on the k3 column), you can use `COLUMNS(k1, k2, k3, k4 = k3 + 1)`. For more details, refer to the Data Conversion documentation.
WHERE	`<where_expr>`	Specifies the condition to filter the loaded data source. For example, to load only data where age > 30, use `WHERE age > 30`.
PARTITION	`<partition_name>`	Specifies which partitions in the target table to load. If not specified, it will automatically load into the corresponding partitions. For example, to load partitions p1 and p2 of the target table, use `PARTITION(p1, p2)`.
DELETE ON	`<delete_expr>`	In the MERGE load mode, using delete_expr to mark which columns need to be deleted. For example, to delete columns where age > 30 during the MERGE process, use `DELETE ON age > 30`.
ORDER BY	`<order_by_column>`	Only effective for Unique Key models. Specifies the Sequence Column in the loaded data to ensure the order of the data. For example, when loading into a Unique Key table and specifying create_time as the Sequence Column, use `ORDER BY create_time`. For more information on Sequence Columns in Unique Key models, refer to the Data Update/Sequence Columns

job_properties Clause

The job_properties clause is used to specify the properties of a Routine Load job when creating it. The syntax is as follows:

```
PROPERTIES ("<key1>" = "<value1>"[, "<key2>" = "<value2>" ...])
```

Here are the available parameters for the job_properties clause:

Parameter	Description
desired_concurrent_number	
max_batch_interval	The maximum running time for each subtask, in seconds. Must be greater than 0, with a default value of 10s. max_batch_interval/max_batch_rows/max_batch_size together form the execution threshold for subtasks. If any of these parameters reaches the threshold, the load subtask ends and a new one is generated.
max_batch_rows	The maximum number of rows read by each subtask. Must be greater than or equal to 200,000. The default value is 200,000(The default value for versions 2.0.13 and higher is 20,000,000). max_batch_interval/max_batch_rows/max_batch_size together form the execution threshold for subtasks. If any of these parameters reaches the threshold, the load subtask ends and a new one is generated.
max_batch_size	The maximum number of bytes read by each subtask. The unit is bytes, and the range is from 100MB to 10GB. The default value is 100MB(The default value for versions 2.0.13 and higher is 1G). max_batch_interval/max_batch_rows/max_batch_size together form the execution threshold for subtasks. If any of these parameters reaches the threshold, the load subtask ends and a new one is generated.
max_error_number	The maximum number of error rows allowed within a sampling window. Must be greater than or equal to 0. The default value is 0, which means no error rows are allowed. The sampling window is `max_batch_rows * 10`. If the number of error rows within the sampling window exceeds `max_error_number`, the regular job will be paused and manual intervention is required to check for data quality issues using the SHOW ROUTINE LOAD command and `ErrorLogUrls`. Rows filtered out by the WHERE condition are not counted as error rows.
strict_mode	Whether to enable strict mode. The default value is disabled. Strict mode applies strict filtering to type conversions during the load process. If enabled, non-null original data that results in a NULL after type conversion will be filtered out. The filtering rules in strict mode are as follows:
timezone	Specifies the time zone used by the load job. The default is to use the session's timezone parameter. This parameter affects the results of all timezone-related functions involved in the load.
format	Specifies the data format for the load. The default is csv, and JSON format is supported.
jsonpaths	When the data format is JSON, jsonpaths can be used to specify the JSON paths to extract data from nested structures. It is a JSON array of strings, where each string represents a JSON path.
delimiter	Specifies the delimiter used in CSV files. The default delimiter is a comma (,).
escape	Specifies the escape character used in CSV files. The default escape character is a backslash ().
quote	Specifies the quote character used in CSV files. The default quote character is a double quotation mark (“).
null_format	Specifies the string representation of NULL values in the load data. The default is an empty string.
skip_header_lines	Specifies the number of lines to skip at the beginning of the load data file. The default is 0, which means no lines are skipped.
skip_footer_lines	Specifies the number of lines to skip at the end of the load data file. The default is 0, which means no lines are skipped.
query_parallelism	Specifies the number of parallel threads used by each subtask to execute SQL statements. The default is 1.

Parameter	Description
query_timeout	Specifies the timeout for SQL statement execution. The default is 3600 seconds (1 hour).
query_band	Specifies the query band string to be set for each subtask.
memory_quota_per_query	Specifies the memory quota for each subtask, in bytes. The default is -1, which means to use the system default.
error_table_name	Specifies the name of the error table where error rows are stored. The default is null, which means no error table is generated.
error_table_database	Specifies the database where the error table is located. The default is null, which means the error table is located in the current database.
error_table_schema	Specifies the schema where the error table is located. The default is null, which means the error table is located in the public schema.
error_table_logging_policy	Specifies the logging policy for the error table. The default is null, which means to use the system default.
error_table_reuse_policy	Specifies the reuse policy for the error table. The default is null, which means to use the system default.
error_table_creation_time	Specifies the creation time for the error table. The default is null, which means to use the current time.
error_table_cleanup_time	Specifies the cleanup time for the error table. The default is null, which means not set a cleanup time.
error_table_log	Specifies whether to enable logging for the error table. The default is null, which means to use the system default.
error_table_backup_time	Specifies the backup time for the error table. The default is null, which means not set a backup time.
error_table_backup_path	Specifies the backup path for the error table. The default is null, which means not set a backup path.
error_table_lifetime	Specifies the lifetime of the error table. The default is null, which means to use the system default.
error_table_backup_lifetime	Specifies the backup lifetime for the error table. The default is null, which means to use the system default.
error_table_label	Specifies the label for the error table. The default is null, which means not set a label.
error_table_priority	Specifies the priority for the error table. The default is null, which means to use the system default.
error_table_comment	Specifies the comment for the error table. The default is null, which means to not set a comment.

These parameters can be used to customize the behavior of a Routine Load job according to your specific requirements.

04 data_source_properties Clause

When creating a Routine Load job, you can specify the data_source_properties clause to specify properties of the Kafka data source. The syntax is as follows:

```
FROM KAFKA ("<key1>" = "<value1>"[, "<key2>" = "<value2>" ...])
```

The available options for the data_source_properties clause are as follows:

Parameter	Description
kafka_broker_list	Specifies the connection information for Kafka brokers. The format is `<kafka_broker_ip>:<kafka_port>`. Multiple brokers are separated by commas. For example, to specify a Broker List with the default port 9092, you can use the following command: `"kafka_broker_list" = "<broker1_ip>:9092,<broker2_ip>:9092"`
kafka_topic	Specifies the Kafka topic to subscribe to. A load job can only consume one Kafka topic.
kafka_partitions	Specifies the Kafka partitions to subscribe to. If not specified, all partitions are consumed by default.
kafka_offsets	Specifies the starting consumption offset for Kafka partitions. If a timestamp is specified, consumption starts from the nearest offset equal to or greater than that timestamp. The offset can be a specific offset greater than or equal to 0, or it can use the following formats:
property	Specifies custom Kafka parameters. This is equivalent to the “–property” parameter in the Kafka shell. When the value of a parameter is a file, the keyword “FILE:” needs to be added before the value. For creating a file, you can refer to the CREATE FILE command documentation. For more supported custom parameters, you can refer to the client-side configuration options in the official CONFIGURATION documentation of librdkafka. For example: `"property.client.id" = "12345"`, `"property.group.id" = "group_id_0"`, `"property.ssl.ca.location" = "FILE:ca.pem"`

By configuring the Kafka property parameter in the `data_source_properties`, you can set up security access options. Currently, Doris supports various Kafka security protocols such as plaintext (default), SSL, PLAIN, and Kerberos.

Load Status

You can check the status of a load job using the `SHOW ROUTINE LOAD` command. The syntax for the command is as follows:

```
SHOW [ALL] ROUTINE LOAD [FOR jobName];
```

For example, executing `SHOW ROUTINE LOAD` will return a result set similar to the following:

```
mysql> SHOW ROUTINE LOAD FOR testdb.example_routine_load\G
*************************** 1. row ***************************
                  Id: 12025
                Name: example_routine_load
          CreateTime: 2024-01-15 08:12:42
           PauseTime: NULL
             EndTime: NULL
              DbName: default_cluster:testdb
           TableName: test_routineload_tbl
        IsMultiTable: false
               State: RUNNING
      DataSourceType: KAFKA
      CurrentTaskNum: 1
       JobProperties:
{"max_batch_rows":"200000","timezone":"America/New_York","send_batch_parallelism":"1","load_to_single_tablet":"false","column_separa
tor":"','","line_delimiter":"\n","current_concurrent_number":"1","delete":"*","partial_columns":"false","merge_type":"APPEND","exec_
mem_limit":"2147483648","strict_mode":"false","jsonpaths":"","max_batch_interval":"10","max_batch_size":"104857600","fuzzy_parse":"f
alse","partitions":"*","columnToColumnExpr":"user_id,name,age","whereExpr":"*","desired_concurrent_number":"5","precedingFilter":"*"
,"format":"csv","max_error_number":"0","max_filter_ratio":"1.0","json_root":"","strip_outer_array":"false","num_as_string":"false"}
```

```
DataSourceProperties: {"topic":"test-topic","currentKafkaPartitions":"0","brokerList":"192.168.88.62:9092"}
    CustomProperties: {"kafka_default_offsets":"OFFSET_BEGINNING","group.id":"example_routine_load_73daf600-884e-46c0-a02b-
4e49fdf3b4dc"}
           Statistic: {"receivedBytes":28,"runningTxns":
[],"errorRows":0,"committedTaskNum":3,"loadedRows":3,"loadRowsRate":0,"abortedTaskNum":0,"errorRowsAfterResumed":0,"totalRows":3,"un
selectedRows":0,"receivedBytesRate":0,"taskExecuteTimeMs":30069}
            Progress: {"0":"2"}
                 Lag: {"0":0}
ReasonOfStateChanged:
        ErrorLogUrls:
             OtherMsg:
                 User: root
              Comment:
1 row in set (0.00 sec)
```

The columns in the result set provide the following information:

Column Name	Description
Id	The ID of the load job, automatically generated by Doris.
Name	The name of the load job.
CreateTime	The time when the job was created.
PauseTime	The time when the job was last paused.
EndTime	The time when the job ended.
DbName	The name of the associated database.
TableName	The name of the associated table. For multi-table scenarios, it is displayed as “multi-table”.
IsMultiTbl	Indicates whether it is a multi-table load.
State	The running state of the job, which can have five values:
DataSourceType	The type of data source, which is KAFKA, in this example.
CurrentTaskNum	The current number of subtasks.
JobProperties	Details of the job configuration.
DataSourceProperties	Details of the data source configuration.
CustomProperties	Custom configuration properties.
Statistic	Statistics of the job’s running status.
Progress	The job’s progress. For Kafka data sources, it shows the offset consumed for each partition. For example, `{"0":"2"}` indicates that partition 0 has consumed 2 offsets.
Lag	The lag of the job. For Kafka data sources, it shows the consumption lag for each partition. For example, `{"0":10}` indicates a consumption lag of 10 for partition 0.
ReasonOfStateChanged	The reason for the state change of jobs.
ErrorLogUrls	The URL(s) to view the filtered low-quality data.
OtherMsg	Other error messages.

Load example

Loading CSV Format

Setting the Maximum Error Tolerance

1. Load sample data:

```
1,Benjamin,18
2,Emily,20
3,Alexander,22
```

2. Create table:

```
CREATE TABLE demo.routine_test01 (
    id       INT             NOT NULL   COMMENT "User ID",
    name     VARCHAR(30)     NOT NULL   COMMENT "Name",
    age      INT                        COMMENT "Age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job01 ON routine_test01
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age)
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "max_filter_ratio"="0.5",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad01",
            "property.group.id" = "kafka_job01",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test01;
+------+------------+------+
| id   | name       | age  |
+------+------------+------+
|    1 | Benjamin   |   18 |
|    2 | Emily      |   20 |
|    3 | Alexander  |   22 |
+------+------------+------+
3 rows in set (0.01 sec)
```

Consuming Data from a Specified Offset

1. Load sample data:

```
1,Benjamin,18
2,Emily,20
3,Alexander,22
4,Sophia,24
5,William,26
6,Charlotte,28
```

2. Create table:

```
CREATE TABLE demo.routine_test02 (
    id       INT             NOT NULL   COMMENT "User ID",
    name     VARCHAR(30)     NOT NULL   COMMENT "Name",
    age      INT                        COMMENT "Age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job02 ON routine_test02
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age)
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad02",
            "property.group.id" = "kafka_job",
            "kafka_partitions" = "0",
            "kafka_offsets" = "3"
        );
```

4. Load result:

```
mysql> select * from routine_test02;
+------+--------------+------+
| id   | name         | age  |
+------+--------------+------+
|    4 | Sophia       |   24 |
|    5 | William      |   26 |
|    6 | Charlotte    |   28 |
+------+--------------+------+
3 rows in set (0.01 sec)
```

Specifying the Consumer Group's group.id and client.id

1. Load sample data:

```
1,Benjamin,18
2,Emily,20
3,Alexander,22
```

2. Create table:

```
CREATE TABLE demo.routine_test03 (
    id       INT             NOT NULL   COMMENT "User ID",
    name     VARCHAR(30)     NOT NULL   COMMENT "Name",
    age      INT                        COMMENT "Age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job03 ON routine_test03
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age)
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad01",
            "property.group.id" = "kafka_job03",
            "property.client.id" = "kafka_client_03",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test03;
+------+------------+------+
| id   | name       | age  |
+------+------------+------+
|    1 | Benjamin   |   18 |
|    2 | Emily      |   20 |
|    3 | Alexander  |   22 |
+------+------------+------+
3 rows in set (0.01 sec)
```

Setting load filtering conditions

1. Load sample data:

```
1,Benjamin,18
2,Emily,20
3,Alexander,22
4,Sophia,24
5,William,26
6,Charlotte,28
```

2. Create table:

```
CREATE TABLE demo.routine_test04 (
    id       INT             NOT NULL   COMMENT "User ID",
    name     VARCHAR(30)     NOT NULL   COMMENT "Name",
```

```
    age      INT                         COMMENT "Age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job04 ON routine_test04
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age),
        WHERE id >= 3
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad04",
            "property.group.id" = "kafka_job04",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test04;
+------+--------------+------+
| id   | name         | age  |
+------+--------------+------+
|    4 | Sophia       |   24 |
|    5 | William      |   26 |
|    6 | Charlotte    |   28 |
+------+--------------+------+
3 rows in set (0.01 sec)
```

Loading specified partition data

1. Load sample data:

```
1,Benjamin,18,2024-02-04 10:00:00
2,Emily,20,2024-02-05 11:00:00
3,Alexander,22,2024-02-06 12:00:00
```

2. Create table:

```
CREATE TABLE demo.routine_test05 (
    id      INT            NOT NULL  COMMENT "ID",
    name    VARCHAR(30)    NOT NULL  COMMENT "Name",
    age     INT                      COMMENT "Age",
    date    DATETIME                 COMMENT "Date"
)
PARTITION BY RANGE(date) ()
DISTRIBUTED BY HASH(date)
PROPERTIES
(
    "replication_num" = "1",
    "dynamic_partition.enable" = "true",
```

```
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.start" = "-2",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "1"
);
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job05 ON routine_test05
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age, date),
        PARTITION(p20240205)
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad05",
            "property.group.id" = "Apologies, but I'm unable to assist with the translation request at the moment.")
```

4. Load result:

```
mysql> select * from routine_test05;
+------+----------+------+---------------------+
| id   | name     | age  | date                |
+------+----------+------+---------------------+
|    2 | Emily    |   20 | 2024-02-05 11:00:00 |
+------+----------+------+---------------------+
3 rows in set (0.01 sec)
```

Setting Time Zone for load

1. Load sample data:

```
1,Benjamin,18,2024-02-04 10:00:00
2,Emily,20,2024-02-05 11:00:00
3,Alexander,22,2024-02-06 12:00:00
```

2. Create table:

```
CREATE TABLE demo.routine_test06 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age",
    date    DATETIME                 COMMENT "date"
)
DUPLICATE KEY(id)
DISTRIBUTED BY HASH(id) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job06 ON routine_test06
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age, date)
        PROPERTIES
        (
            "desired_concurrent_number" = "1",
            "strict_mode" = "false",
            "timezone" = "Asia/Shanghai"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad06",
            "property.group.id" = "kafka_job06",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test06;
+------+-------------+------+---------------------+
| id   | name        | age  | date                |
+------+-------------+------+---------------------+
|    1 | Benjamin    |   18 | 2024-02-05 10:00:00 |
|    2 | Emily       |   20 | 2024-02-05 11:00:00 |
|    3 | Alexander   |   22 | 2024-02-05 12:00:00 |
+------+-------------+------+---------------------+
3 rows in set (0.00 sec)
```

Specify merge_type for delete operation

1. Load sample data:

```
3,Alexander,22
5,William,26
```

Table data before load:

```
mysql> SELECT * FROM routine_test07;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
|    4 | Sophia         |   24 |
|    5 | William        |   26 |
|    6 | Charlotte      |   28 |
+------+----------------+------+
```

2. Create table:

```
CREATE TABLE demo.routine_test07 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age"
)
```

```
DUPLICATE KEY(id)
DISTRIBUTED BY HASH(id) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job07 ON routine_test07
        WITH DELETE
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age)
        PROPERTIES
        (
            "desired_concurrent_number" = "1",
            "max_filter_ratio" = "0.5",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad07",
            "property.group.id" = "kafka_job07",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> SELECT * FROM routine_test07;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    4 | Sophia         |   24 |
|    6 | Charlotte      |   28 |
+------+----------------+------+
```

Specify merge_type for merge operation

1. Load sample data:

```
1,xiaoxiaoli,28
2,xiaoxiaowang,30
3,xiaoxiaoliu,32
4,dadali,34
5,dadawang,36
6,dadaliu,38
```

Table data before load:

```
mysql> SELECT * FROM routine_test08;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
|    4 | Sophia         |   24 |
|    5 | William        |   26 |
|    6 | Charlotte      |   28 |
```

```
+------+----------------+------+
6 rows in set (0.01 sec)
```

2. Create table:

```
CREATE TABLE demo.routine_test08 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age"
)
DUPLICATE KEY(id)
DISTRIBUTED BY HASH(id) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job08 ON routine_test08
        WITH MERGE
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age),
        DELETE ON id = 2
        PROPERTIES
        (
            "desired_concurrent_number" = "1",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad08",
            "property.group.id" = "kafka_job08",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> SELECT * FROM routine_test08;
+------+-------------+------+
| id   | name        | age  |
+------+-------------+------+
|    1 | xiaoxiaoli  |   28 |
|    3 | xiaoxiaoliu |   32 |
|    4 | dadali      |   34 |
|    5 | dadawang    |   36 |
|    6 | dadaliu     |   38 |
+------+-------------+------+
5 rows in set (0.00 sec)
```

Specifying the sequence column to be merged

1. Load sample data:

```
1,xiaoxiaoli,28
2,xiaoxiaowang,30
3,xiaoxiaoliu,32
4,dadali,34
5,dadawang,36
6,dadaliu,38
```

Data in the table before loading:

```
mysql> SELECT * FROM routine_test09;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
|    4 | Sophia         |   24 |
|    5 | William        |   26 |
|    6 | Charlotte      |   28 |
+------+----------------+------+
6 rows in set (0.01 sec)
```

2. Create table

```
CREATE TABLE demo.routine_test08 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age",
)
DUPLICATE KEY(id)
DISTRIBUTED BY HASH(id) BUCKETS 1
PROPERTIES (
    "function_column.sequence_col" = "age"
);
```

3. Load Command

```
CREATE ROUTINE LOAD demo.kafka_job09 ON routine_test09
        WITH MERGE
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age),
        DELETE ON id = 2,
        ORDER BY age
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad09",
            "property.group.id" = "kafka_job09",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> SELECT * FROM routine_test09;
+------+-------------+------+
| id   | name        | age  |
+------+-------------+------+
|    1 | xiaoxiaoli  |   28 |
|    3 | xiaoxiaoliu |   32 |
|    4 | dadali      |   34 |
|    5 | dadawang    |   36 |
```

```
|    6 | dadaliu     |   38 |
+------+-------------+------+
5 rows in set (0.00 sec)
```

Load with column mapping and derived column calculation

1. Load sample data:

```
1,Benjamin,18
2,Emily,20
3,Alexander,22
```

2. Create table:

```
CREATE TABLE demo.routine_test10 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age",
    num     INT                      COMMENT "number"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job10 ON routine_test10
        COLUMNS TERMINATED BY ",",
        COLUMNS(id, name, age, num=age*10)
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "max_filter_ratio"="0.5",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad10",
            "property.group.id" = "kafka_job10",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> SELECT * FROM routine_test10;
+------+---------------+------+------+
| id   | name          | age  | num  |
+------+---------------+------+------+
|    1 | Benjamin      |   18 |  180 |
|    2 | Emily         |   20 |  200 |
|    3 | Alexander     |   22 |  220 |
+------+---------------+------+------+
3 rows in set (0.01 sec)
```

Load with enclosed data

1. Load sample data:

```
1,"Benjamin",18
2,"Emily",20
3,"Alexander",22
```

2. Create table:

```
CREATE TABLE demo.routine_test11 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age",
    num     INT                      COMMENT "number"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job12 ON routine_test12
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "enclose" = "\""
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad12",
            "property.group.id" = "kafka_job12",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> SELECT * FROM routine_test11;
+------+----------------+------+------+
| id   | name           | age  | num  |
+------+----------------+------+------+
|    1 | Benjamin       |   18 |  180 |
|    2 | Emily          |   20 |  200 |
|    3 | Alexander      |   22 |  220 |
+------+----------------+------+------+
3 rows in set (0.02 sec)
```

JSON Format Load

Load JSON format data in simple mode

1. Load sample data:

```
{ "id" : 1, "name" : "Benjamin", "age":18 }
{ "id" : 2, "name" : "Emily", "age":20 }
{ "id" : 3, "name" : "Alexander", "age":22 }
```

2. Create table:

```
CREATE TABLE demo.routine_test12 (
    id      INT           NOT NULL  COMMENT "id",
    name    VARCHAR(30)   NOT NULL  COMMENT "name",
    age     INT                     COMMENT "age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job12 ON routine_test12
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad12",
            "property.group.id" = "kafka_job12",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test12;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
+------+----------------+------+
3 rows in set (0.02 sec)
```

Load complex JSON format data in match mode

1. Load sample data

```
{ "name" : "Benjamin", "id" : 1, "num":180 , "age":18 }
{ "name" : "Emily", "id" : 2, "num":200 , "age":20 }
{ "name" : "Alexander", "id" : 3, "num":220 , "age":22 }
```

2. Create table:

```
CREATE TABLE demo.routine_test13 (
    id      INT           NOT NULL  COMMENT "id",
    name    VARCHAR(30)   NOT NULL  COMMENT "name",
    age     INT                     COMMENT "age",
    num     INT                     COMMENT "num"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job13 ON routine_test13
        COLUMNS(name, id, num, age)
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false",
            "jsonpaths" = "[\"$.name\",\"$.id\",\"$.num\",\"$.age\"]"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad13",
            "property.group.id" = "kafka_job13",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test13;
+------+----------------+------+------+
| id   | name           | age  | num  |
+------+----------------+------+------+
|    1 | Benjamin       |   18 |  180 |
|    2 | Emily          |   20 |  200 |
|    3 | Alexander      |   22 |  220 |
+------+----------------+------+------+
3 rows in set (0.01 sec)
```

Loading data with specified JSON root node

1. Load sample data

```
{"id": 1231, "source" :{ "id" : 1, "name" : "Benjamin", "age":18 }}
{"id": 1232, "source" :{ "id" : 2, "name" : "Emily", "age":20 }}
{"id": 1233, "source" :{ "id" : 3, "name" : "Alexander", "age":22 }}
```

2. Create table:

```
CREATE TABLE demo.routine_test14 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command

```
CREATE ROUTINE LOAD demo.kafka_job14 ON routine_test14
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false",
```

```
            "json_root" = "$.source"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad14",
            "property.group.id" = "kafka_job14",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result

```
mysql> select * from routine_test14;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
+------+----------------+------+
3 rows in set (0.01 sec)
```

Loading data with column mapping and derived column calculation

1. Load sample data:

```
{ "id" : 1, "name" : "Benjamin", "age":18 }
{ "id" : 2, "name" : "Emily", "age":20 }
{ "id" : 3, "name" : "Alexander", "age":22 }
```

2. Create table:

```
CREATE TABLE demo.routine_test15 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age",
    num     INT                      COMMENT "num"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job15 ON routine_test15
        COLUMNS(id, name, age, num=age*10)
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad15",
            "property.group.id" = "kafka_job15",
```

```
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test15;
+------+----------------+------+------+
| id   | name           | age  | num  |
+------+----------------+------+------+
|    1 | Benjamin       |   18 |  180 |
|    2 | Emily          |   20 |  200 |
|    3 | Alexander      |   22 |  220 |
+------+----------------+------+------+
3 rows in set (0.01 sec)
```

Loading Complex Data Types

Load Array Data Type

1. Load sample data:

```
{ "id" : 1, "name" : "Benjamin", "age":18, "array":[1,2,3,4,5]}
{ "id" : 2, "name" : "Emily", "age":20, "array":[6,7,8,9,10]}
{ "id" : 3, "name" : "Alexander", "age":22, "array":[11,12,13,14,15]}
```

2. Create table:

```
CREATE TABLE demo.routine_test16
(
    id      INT             NOT NULL  COMMENT "id",
    name    VARCHAR(30)     NOT NULL  COMMENT "name",
    age     INT                       COMMENT "age",
    array   ARRAY<int(11)>  NULL      COMMENT "test array column"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job16 ON routine_test16
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad16",
            "property.group.id" = "kafka_job16",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test16;
+------+----------------+------+----------------------+
| id   | name           | age  | array                |
+------+----------------+------+----------------------+
|    1 | Benjamin       |   18 | [1, 2, 3, 4, 5]      |
|    2 | Emily          |   20 | [6, 7, 8, 9, 10]     |
|    3 | Alexander      |   22 | [11, 12, 13, 14, 15] |
+------+----------------+------+----------------------+
3 rows in set (0.00 sec)
```

Loading Map Data Type

1. Load sample data:

```
{ "id" : 1, "name" : "Benjamin", "age":18, "map":{"a": 100, "b": 200}}
{ "id" : 2, "name" : "Emily", "age":20, "map":{"c": 300, "d": 400}}
{ "id" : 3, "name" : "Alexander", "age":22, "map":{"e": 500, "f": 600}}
```

2. Create table:

```
CREATE TABLE demo.routine_test17 (
    id      INT                 NOT NULL  COMMENT "id",
    name    VARCHAR(30)         NOT NULL  COMMENT "name",
    age     INT                           COMMENT "age",
    map     Map<STRING, INT>    NULL      COMMENT "test column"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job17 ON routine_test17
    PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad17",
            "property.group.id" = "kafka_job17",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test17;
+------+----------------+------+--------------------+
| id   | name           | age  | map                |
+------+----------------+------+--------------------+
|    1 | Benjamin       |   18 | {"a":100, "b":200} |
|    2 | Emily          |   20 | {"c":300, "d":400} |
|    3 | Alexander      |   22 | {"e":500, "f":600} |
```

```
+------+----------------+------+--------------------+
3 rows in set (0.01 sec)
```

Loading Bitmap Data Type

1. Load sample data

```
{ "id" : 1, "name" : "Benjamin", "age":18, "bitmap_id":243}
{ "id" : 2, "name" : "Emily", "age":20, "bitmap_id":28574}
{ "id" : 3, "name" : "Alexander", "age":22, "bitmap_id":8573}
```

2. Create table:

```
CREATE TABLE demo.routine_test18 (
    id        INT            NOT NULL      COMMENT "id",
    name      VARCHAR(30)    NOT NULL      COMMENT "name",
    age       INT                          COMMENT "age",
    bitmap_id INT                          COMMENT "test",
    device_id BITMAP         BITMAP_UNION  COMMENT "test column"
)
AGGREGATE KEY (`id`,`name`,`age`,`bitmap_id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job18 ON routine_test18
        COLUMNS(id, name, age, bitmap_id, device_id=to_bitmap(bitmap_id))
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad18",
            "property.group.id" = "kafka_job18",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select id, BITMAP_UNION_COUNT(pv) over(order by id) uv from(
    ->    select id, BITMAP_UNION(device_id) as pv
    ->    from routine_test18
    -> group by id
    -> ) final;
+------+------+
| id   | uv   |
+------+------+
|    1 |    1 |
|    2 |    2 |
|    3 |    3 |
+------+------+
3 rows in set (0.00 sec)
```

Loading HLL Data Type

1. Loading sample data:

```
2022-05-05,10001,Test01,Beijing,windows
2022-05-05,10002,Test01,Beijing,linux
2022-05-05,10003,Test01,Beijing,macos
2022-05-05,10004,Test01,Hebei,windows
2022-05-06,10001,Test01,Shanghai,windows
2022-05-06,10002,Test01,Shanghai,linux
2022-05-06,10003,Test01,Jiangsu,macos
2022-05-06,10004,Test01,Shaanxi,windows
```

2. Create table:

```
create table demo.routine_test19 (
    dt        DATE,
    id        INT,
    name      VARCHAR(10),
    province  VARCHAR(10),
    os        VARCHAR(10),
    pv        hll hll_union
)
Aggregate KEY (dt,id,name,province,os)
distributed by hash(id) buckets 10;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job19 ON routine_test19
        COLUMNS TERMINATED BY ",",
        COLUMNS(dt, id, name, province, os, pv=hll_hash(id))
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "10.16.10.6:9092",
            "kafka_topic" = "routineLoad19",
            "property.group.id" = "kafka_job19",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING"
        );
```

4. Load result:

```
mysql> select * from routine_test19;
+------------+-------+----------+----------+---------+------+
| dt         | id    | name     | province | os      | pv   |
+------------+-------+----------+----------+---------+------+
| 2022-05-05 | 10001 | Test01   | Beijing     | windows | NULL |
| 2022-05-06 | 10001 | Test01   | Shanghai    | windows | NULL |
| 2022-05-05 | 10002 | Test01   | Beijing     | linux   | NULL |
| 2022-05-06 | 10002 | Test01   | Shanghai    | linux   | NULL |
| 2022-05-05 | 10004 | Test01   | Heibei      | windows | NULL |
| 2022-05-06 | 10004 | Test01   | Shanxi      | windows | NULL |
| 2022-05-05 | 10003 | Test01   | Beijing     | macos   | NULL |
| 2022-05-06 | 10003 | Test01   | Jiangsu     | macos   | NULL |
+------------+-------+----------+----------+---------+------+
```

```
8 rows in set (0.01 sec)

mysql> SELECT HLL_UNION_AGG(pv) FROM routine_test19;
+------------------+
| hll_union_agg(pv) |
+------------------+
|                4 |
+------------------+
1 row in set (0.01 sec)
```

Kafka Security Authentication

Loading Kafka data with SSL authentication

1. Loading sample data:

```
{ "id" : 1, "name" : "Benjamin", "age":18 }
{ "id" : 2, "name" : "Emily", "age":20 }
{ "id" : 3, "name" : "Alexander", "age":22 }
```

2. Create table:

```
CREATE TABLE demo.routine_test20 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job20 ON routine_test20
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "192.168.100.129:9092",
            "kafka_topic" = "routineLoad21",
            "property.group.id" = "kafka_job21",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING",
            "property.security.protocol" = "ssl",
            "property.ssl.ca.location" = "FILE:ca.pem",
            "property.ssl.certificate.location" = "FILE:client.pem",
            "property.ssl.key.location" = "FILE:client.key",
            "property.ssl.key.password" = "ssl_passwd"
        );
```

4. Load result:

```
mysql> select * from routine_test20;
+------+---------------+------+
| id   | name          | age  |
```

```
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
+------+----------------+------+
3 rows in set (0.01 sec)
```

Loading Kafka data with Kerberos authentication

1. Loading sample data:

```
{ "id" : 1, "name" : "Benjamin", "age":18 }
{ "id" : 2, "name" : "Emily", "age":20 }
{ "id" : 3, "name" : "Alexander", "age":22 }
```

2. Create table:

```
CREATE TABLE demo.routine_test21 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job21 ON routine_test21
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "192.168.100.129:9092",
            "kafka_topic" = "routineLoad21",
            "property.group.id" = "kafka_job21",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING",
            "property.security.protocol" = "SASL_PLAINTEXT",
            "property.sasl.kerberos.service.name" = "kafka",
            "property.sasl.kerberos.keytab" = "/etc/krb5.keytab",
            "property.sasl.kerberos.keytab"="/opt/third/kafka/kerberos/kafka_client.keytab",
            "property.sasl.kerberos.principal" = "clients/stream.dt.local@EXAMPLE.COM"
        );
```

4. Load result:

```
mysql> select * from routine_test21;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
```

```
+------+----------------+------+
3 rows in set (0.01 sec)
```

Loading Kafka data with PLAIN authentication in Kafka cluster

1. Loading sample data:

```
{ "id" : 1, "name" : "Benjamin", "age":18 }
{ "id" : 2, "name" : "Emily", "age":20 }
{ "id" : 3, "name" : "Alexander", "age":22 }
```

2. Create table:

```
CREATE TABLE demo.routine_test22 (
    id      INT            NOT NULL  COMMENT "id",
    name    VARCHAR(30)    NOT NULL  COMMENT "name",
    age     INT                      COMMENT "age"
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1;
```

3. Load command:

```
CREATE ROUTINE LOAD demo.kafka_job22 ON routine_test22
        PROPERTIES
        (
            "desired_concurrent_number"="1",
            "format" = "json",
            "strict_mode" = "false"
        )
        FROM KAFKA
        (
            "kafka_broker_list" = "192.168.100.129:9092",
            "kafka_topic" = "routineLoad22",
            "property.group.id" = "kafka_job22",
            "property.kafka_default_offsets" = "OFFSET_BEGINNING",
            "property.security.protocol"="SASL_PLAINTEXT",
            "property.sasl.mechanism"="PLAIN",
            "property.sasl.username"="admin",
            "property.sasl.password"="admin"
        );
```

4. Load result

```
mysql> select * from routine_test22;
+------+----------------+------+
| id   | name           | age  |
+------+----------------+------+
|    1 | Benjamin       |   18 |
|    2 | Emily          |   20 |
|    3 | Alexander      |   22 |
+------+----------------+------+
3 rows in set (0.02 sec)
```

Single-task Loading to Multiple Tables

Create a Kafka routine dynamic table load task named “test1” for the “example_db”. Specify the column delimiter, group.id, and client.id. Automatically consume all partitions and start subscribing from the available data position (OFFSET_BEGINNING).

Assuming we need to load data from Kafka into tables “tbl1” and “tbl2” in the “example_db”, we create a Routine Load task named “test1”. This task will load data from Kafka’s topic `my_topic` into both “tbl1” and “tbl2” simultaneously. This way, we can load data from Kafka into two tables using a single routine load task.

```
CREATE ROUTINE LOAD example_db.test1
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "false"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "property.group.id" = "xxx",
    "property.client.id" = "xxx",
    "property.kafka_default_offsets" = "OFFSET_BEGINNING"
);
```

Currently, only extracting the table name from the Value field of Kafka is supported. The format should be as follows, using JSON as an example: `table_name|{"col1": "val1", "col2": "val2"}`, where `tbl_name` is the table name and `|` is used as the separator between the table name and the table data. The same format applies to CSV data, such as `table_name|val1,val2,val3`. Note that the `table_name` here must be consistent with the table name in Doris, otherwise the load will fail. Note that dynamic tables do not support the column_mapping configuration described later.

Strict Mode Load

Create a Kafka routine load task named “test1” for the “example_db” and “example_tbl”. The load task is set to strict mode.

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
COLUMNS(k1, k2, k3, v1, v2, v3 = k1 * 100),
PRECEDING FILTER k1 = 1,
WHERE k1 < 100 and k2 like "%doris%"
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "true"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
```

```
    "kafka_partitions" = "0,1,2,3",
    "kafka_offsets" = "101,0,0,200"
);
```

More Details

Refer to the SQL manual on Routine Load. You can also enter `HELP ROUTINE LOAD` in the client command line for more help.

Insert Into

The INSERT INTO statement supports importing the results of a Doris query into another table. INSERT INTO is a synchronous import method, where the import result is returned after the import is executed. Whether the import is successful can be determined based on the returned result. INSERT INTO ensures the atomicity of the import task, meaning that either all the data is imported successfully or none of it is imported.

There are primarily two main forms of the INSERT INTO command:

- INSERT INTO tbl SELECT…
- INSERT INTO tbl (col1, col2, …) VALUES (1, 2, …), (1,3, …)

Applicable scenarios

1. If a user wants to import only a few test data records to verify the functionality of the Doris system, the INSERT INTO VALUES syntax is applicable. It is similar to the MySQL syntax. However, it is not recommended to use INSERT INTO VALUES in a production environment.
2. If a user wants to perform ETL on existing data in a Doris table and then import it into a new Doris table, the INSERT INTO SELECT syntax is applicable.
3. In conjunction with the Multi-Catalog external table mechanism, tables from MySQL or Hive systems can be mapped via Multi-Catalog. Then, data from external tables can be imported into Doris tables using the INSERT INTO SELECT syntax.
4. Utilizing the Table Value Functions (TVFs), users can directly query data stored in object storage or files on HDFS as tables, with automatic column type inference. Then, data from external tables can be imported into Doris tables using the INSERT INTO SELECT syntax.

Implementation

When using INSERT INTO, the import job needs to be initiated and submitted to the FE node using the MySQL protocol. The FE generates an execution plan, which includes query-related operators, with the last operator being the OlapTableSink. The OlapTableSink operator is responsible for writing the query result to the target table. The execution plan is then sent to the BE nodes for execution. Doris designates one BE node as the Coordinator, which receives and distributes the data to other BE nodes.

Get started

An INSERT INTO job is submitted and transmitted using the MySQL protocol. The following example demonstrates submitting an import job using INSERT INTO through the MySQL command-line interface.

Detailed syntax can be found in the INSERT INTO documentation.

Preparation

INSERT INTO requires INSERT permissions on the target table. You can grant permissions to user accounts using the GRANT command.

Create an INSERT INTO job

INSERT INTO VALUES

1. Create a source table

```
CREATE TABLE testdb.test_table(
    user_id            BIGINT       NOT NULL COMMENT "User ID",
    name               VARCHAR(20)           COMMENT "User name",
    age                INT                   COMMENT "User age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

2. Import data into the source table using `INSERT INTO VALUES` (not recommended for production environments).

```
INSERT INTO testdb.test_table (user_id, name, age)
VALUES (1, "Emily", 25),
       (2, "Benjamin", 35),
       (3, "Olivia", 28),
       (4, "Alexander", 60),
       (5, "Ava", 17);
```

INSERT INTO is a synchronous import method, where the import result is directly returned to the user.

```
Query OK, 5 rows affected (0.308 sec)
{'label':'label_3e52da787aab4222_9126d2fce8f6d1e5', 'status':'VISIBLE', 'txnId':'9081'}
```

3. View imported data.

```
MySQL> SELECT COUNT(*) FROM testdb.test_table;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.179 sec)
```

INSERT INTO SELECT

1. Building upon the above operations, create a new table as the target table (with the same schema as the source table).

```
CREATE TABLE testdb.test_table2 LIKE testdb.test_table;
```

2. Ingest data into the new table using `INSERT INTO SELECT`.

```
INSERT INTO testdb.test_table2
SELECT * FROM testdb.test_table WHERE age < 30;
Query OK, 3 rows affected (0.544 sec)
{'label':'label_9c2bae970023407d_b2c5b78b368e78a7', 'status':'VISIBLE', 'txnId':'9084'}
```

3. View imported data.

```
MySQL> SELECT COUNT(*) FROM testdb.test_table2;
+----------+
| count(*) |
+----------+
|        3 |
+----------+
1 row in set (0.071 sec)
```

View INSERT INTO jobs

You can use the `SHOW LOAD` command to view the completed INSERT INTO tasks.

```
MySQL> SHOW LOAD FROM testdb;
+--------+----------------------------------------+----------+--------------------+--------+---------+----------------------------
----------------------------------------+----------+--------------------+--------------------+--------------------+-------------
--------+---------------------+------+-------------------------------------------------------------------------------------------------------------
------------------------+--------------+--------------+------+---------+
| JobId  | Label                                  | State    | Progress           | Type   | EtlInfo | TaskInfo
| ErrorMsg | CreateTime          | EtlStartTime        | EtlFinishTime       | LoadStartTime       | LoadFinishTime      | URL  |
JobDetails                                                                                                                  |
TransactionId | ErrorTablets | User | Comment |
+--------+----------------------------------------+----------+--------------------+--------+---------+----------------------------
----------------------------------------+----------+--------------------+--------------------+--------------------+-------------
--------+---------------------+------+-------------------------------------------------------------------------------------------------------------
------------------------+--------------+--------------+------+---------+
| 376416 | label_3e52da787aab4222_9126d2fce8f6d1e5 | FINISHED | Unknown id: 376416 | INSERT | NULL    | cluster:N/A;
timeout(s):26200; max_filter_ratio:0.0; priority:NORMAL | NULL     | 2024-02-27 01:22:17 | 2024-02-27 01:22:17 | 2024-02-27 01:22:17
| 2024-02-27 01:22:17 | 2024-02-27 01:22:18 |      | {"Unfinished backends":{},"ScannedRows":0,"TaskNumber":0,"LoadBytes":0,"All
backends":{},"FileNumber":0,"FileSize":0} | 9081          | {}           | root |         |
| 376664 | label_9c2bae970023407d_b2c5b78b368e78a7 | FINISHED | Unknown id: 376664 | INSERT | NULL    | cluster:N/A;
timeout(s):26200; max_filter_ratio:0.0; priority:NORMAL | NULL     | 2024-02-27 01:39:37 | 2024-02-27 01:39:37 | 2024-02-27 01:39:37
| 2024-02-27 01:39:37 | 2024-02-27 01:39:38 |      | {"Unfinished backends":{},"ScannedRows":0,"TaskNumber":0,"LoadBytes":0,"All
backends":{},"FileNumber":0,"FileSize":0} | 9084          | {}           | root |         |
+--------+----------------------------------------+----------+--------------------+--------+---------+----------------------------
----------------------------------------+----------+--------------------+--------------------+--------------------+-------------
--------+---------------------+------+-------------------------------------------------------------------------------------------------------------
------------------------+--------------+--------------+------+---------+
```

Cancel INSERT INTO jobs

You can cancel the currently executing INSERT INTO job via Ctrl-C.

Manual

Syntax

The syntax of INSERT INTO is as follows:

1. INSERT INTO SELECT

INSERT INTO SELECT is used to write the query results to the target table.

```
INSERT INTO target_table SELECT ... FROM source_table;
```

The SELECT statement above is similar to a regular SELECT query, allowing operations such as WHERE and JOIN.

2. INSERT INTO VALUES

INSERT INTO VALUES is typically used for testing purposes. It is not recommended for production environments.

```
INSERT INTO target_table (col1, col2, ...)
VALUES (val1, val2, ...), (val3, val4, ...), ...;
```

Parameter configuration

FE configuration

insert_load_default_timeout_second

- Default value: 14400s (4 hours)
- Description: Timeout for import tasks, measured in seconds. If the import task does not complete within this timeout period, it will be canceled by the system and marked as CANCELLED.

Environment parameters

insert_timeout

- Default value: 14400s (4 hours)
- Description: Timeout for INSERT INTO as an SQL statement, measured in seconds.

enable_insert_strict

- Default value: true
- Description: If this is set to true, INSERT INTO will fail when the task involves invalid data. If set to false, INSERT INTO will ignore invalid rows, and the import will be considered successful as long as at least one row is imported successfully.
- Explanation: INSERT INTO cannot control the error rate, so this parameter is used to either strictly check data quality or completely ignore invalid data. Common reasons for data invalidity include: source data column length exceeding destination column length, column type mismatch, partition mismatch, and column order mismatch.

Return values

INSERT INTO an SQL statement, and it returns different results based on different query outcomes:

Empty result set

If the query result set of the SELECT statement in INSERT INTO is empty, the returned value will be as follows:

```
mysql> INSERT INTO tbl1 SELECT * FROM empty_tbl;
Query OK, 0 rows affected (0.02 sec)
```

`Query OK` indicates successful execution. `0 rows affected` means no data was imported.

Non-empty result set and successful INSERT

```
mysql> INSERT INTO tbl1 SELECT * FROM tbl2;
Query OK, 4 rows affected (0.38 sec)
{'label':'INSERT_8510c568-9eda-4173-9e36-6adc7d35291c', 'status':'visible', 'txnId':'4005'}

mysql> INSERT INTO tbl1 WITH LABEL my_label1 SELECT * FROM tbl2;
Query OK, 4 rows affected (0.38 sec)
{'label':'my_label1', 'status':'visible', 'txnId':'4005'}

mysql> INSERT INTO tbl1 SELECT * FROM tbl2;
Query OK, 2 rows affected, 2 warnings (0.31 sec)
{'label':'INSERT_f0747f0e-7a35-46e2-affa-13a235f4020d', 'status':'visible', 'txnId':'4005'}

mysql> INSERT INTO tbl1 SELECT * FROM tbl2;
Query OK, 2 rows affected, 2 warnings (0.31 sec)
{'label':'INSERT_f0747f0e-7a35-46e2-affa-13a235f4020d', 'status':'committed', 'txnId':'4005'}
```

`Query OK` indicates successful execution. `4 rows affected` indicates that a total of 4 rows of data have been imported. `2 warnings` indicates the number of rows that were filtered out.

Additionally, a JSON string is returned:

```
{'label':'my_label1', 'status':'visible', 'txnId':'4005'}
{'label':'INSERT_f0747f0e-7a35-46e2-affa-13a235f4020d', 'status':'committed', 'txnId':'4005'}
{'label':'my_label1', 'status':'visible', 'txnId':'4005', 'err':'some other error'}
```

Parameter description:

Parameter	Description
TxnId	ID of the import transaction
Label	Label of the import job: can be specified using “INSERT INTO tbl WITH LABEL label…”
Status	Visibility of the imported data: If it is visible, it will be displayed as “visible.” If not, it will be displayed as “committed.” In the “committed” state, the import is completed, but the data may be delayed in becoming visible. There is no need to retry in this case.`visible`: The import is successful and the data is visible.`committed`: The import is completed, but the data may be delayed in becoming visible. There is no need to retry in this case.Label Already Exists: The specified label already exists and needs to be changed to a different one.Fail: The import fails.
Err	Error message

You can use the SHOW LOAD statement to view the filtered rows.

```
SHOW LOAD WHERE label="xxx";
```

The result of this statement will include a URL that can be used to query the error data. For more details, refer to the “View error rows” section below.

The invisible state of data is temporary, and the data will eventually become visible.

You can check the visibility status of a batch of data using the SHOW TRANSACTION statement.

```
SHOW TRANSACTION WHERE id=4005;
```

If the `TransactionStatus` column in the result is `visible`, it indicates that the data is visible.

```
{'label':'my_label1', 'status':'visible', 'txnId':'4005'}
{'label':'INSERT_f0747f0e-7a35-46e2-affa-13a235f4020d', 'status':'committed', 'txnId':'4005'}
{'label':'my_label1', 'status':'visible', 'txnId':'4005', 'err':'some other error'}
```

Non-empty result set but failed INSERT

Failed execution means that no data was successfully imported. An error message will be returned:

```
mysql> INSERT INTO tbl1 SELECT * FROM tbl2 WHERE k1 = "a";
ERROR 1064 (HY000): all partitions have no load data. url: http://10.74.167.16:8042/api/_load_error_log?
file=_shard_2/error_loginsert_stmt_ba8bb9e158e4879-ae8de8507c0bf8a2_ba8bb9e158e4879_ae8de8507c0bf8a2
```

`ERROR 1064 (HY000): all partitions have no load data` indicates the root cause for the failure. The URL provided in the error message can be used to locate the error data. For more details, refer to the “View error rows” section below.

Best practice

Data size

INSERT INTO imposes no limitations on data volume and can support large-scale data imports. However, if you are importing a large amount of data, it is recommended to adjust the system’s INSERT INTO timeout settings to ensure that `import timeout >= data volume ``/`` estimated import speed`.

1. FE configuration parameter `insert_load_default_timeout_second`
2. Environment parameter `insert_timeout`

View error rows

When the INSERT INTO result includes a URL field, you can use the following command to view the error rows:

```
SHOW LOAD WARNINGS ON "url";
```

Example:

```
SHOW LOAD WARNINGS ON "http://ip:port/api/_load_error_log?file=_shard_13/error_loginsert_stmt_d2cac0a0a16d482d-
9041c949a4b71605_d2cac0a0a16d482d_9041c949a4b71605";
```

Common reasons for errors include: source data column length exceeding destination column length, column type mismatch, partition mismatch, and column order mismatch.

You can control whether INSERT INTO ignores error rows by configuring the environment variable `enable_insert_strict`.

Ingest external data via Multi-Catalog

Doris supports the creation of external tables. Once created, data from external tables can be imported into Doris internal tables using `INSERT INTO SELECT`, or queried directly using SELECT statements.

With its Multi-Catalog feature, Doris supports connections to various mainstream data lakes and databases including Apache Hive, Apache Iceberg, Apache Hudi, Apache Paimon (Incubating), Elasticsearch, MySQL, Oracle, and SQL Server.

For more information on Multi-Catalog, please refer to Lakehouse overview.

The followings illustrate importing data from a Hive external table into a Doris internal table.

Create Hive Catalog

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hadoop.username' = 'hive',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:8088',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:8088',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

Ingest data

1. Create a target table for the data import in Doris.

```
CREATE TABLE `target_tbl` (
  `k1` decimal(9, 3) NOT NULL COMMENT "",
  `k2` char(10) NOT NULL COMMENT "",
  `k3` datetime NOT NULL COMMENT "",
  `k5` varchar(20) NOT NULL COMMENT "",
  `k6` double NOT NULL COMMENT ""
)
COMMENT "Doris Table"
DISTRIBUTED BY HASH(k1) BUCKETS 2
PROPERTIES (
    "replication_num" = "1"
);
```

1. For detailed instructions on creating Doris tables, please refer to CREATE TABLE.

2. Importing data (from the `hive.db1.source_tbl` table into the `target_tbl` table).

```
INSERT INTO target_tbl SELECT k1,k2,k3 FROM  hive.db1.source_tbl limit 100;
```

The INSERT command is a synchronous command. If it returns a result, that indicates successful import.

Notes

- Ensure that the external data source and the Doris cluster can communicate, including mutual network accessibility between BE nodes and external data sources.

Ingest data by TVF

Doris can directly query and analyze files stored in object storage or HDFS as tables through the Table Value Functions (TVFs), which supports automatic column type inference. For detailed information, please refer to the Lakehouse/TVF documentation.

Automatic column type inference

```
DESC FUNCTION s3 (
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "parquet",
    "use_path_style"="true"
);
+---------------+--------------+------+-------+---------+-------+
| Field         | Type         | Null | Key   | Default | Extra |
+---------------+--------------+------+-------+---------+-------+
| p_partkey     | INT          | Yes  | false | NULL    | NONE  |
| p_name        | TEXT         | Yes  | false | NULL    | NONE  |
| p_mfgr        | TEXT         | Yes  | false | NULL    | NONE  |
| p_brand       | TEXT         | Yes  | false | NULL    | NONE  |
| p_type        | TEXT         | Yes  | false | NULL    | NONE  |
| p_size        | INT          | Yes  | false | NULL    | NONE  |
| p_container   | TEXT         | Yes  | false | NULL    | NONE  |
| p_retailprice | DECIMAL(9,0) | Yes  | false | NULL    | NONE  |
| p_comment     | TEXT         | Yes  | false | NULL    | NONE  |
+---------------+--------------+------+-------+---------+-------+
```

In this example of an S3 TVF, the file path, connection information, and authentication information are specified.

You can use the `DESC FUNCTION` syntax to view the schema of this file.

It can be seen that for Parquet files, Doris automatically infers column types based on the metadata within the file.

Currently, Doris supports analysis and column type inference for Parquet, ORC, CSV, and JSON formats.

It can be used in combination with the `INSERT INTO SELECT` syntax to quickly import files into Doris tables for faster analysis.

```
// 1. Create Doris internal table
CREATE TABLE IF NOT EXISTS test_table
(
    id int,
    name varchar(50),
    age int
)
DISTRIBUTED BY HASH(id) BUCKETS 4
PROPERTIES("replication_num" = "1");
```

```
// 2. Insert data by S3 Table Value Function
INSERT INTO test_table (id,name,age)
SELECT cast(id as INT) as id, name, cast (age as INT) as age
FROM s3(
    "uri" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "parquet",
    "use_path_style" = "true");
```

Notes

- If the URI specified in the `S3 / hdfs` TVF does not match any files, or if all matched files are empty, the `S3 / hdfs` TVF will return an empty result set. In such cases, if you use `DESC FUNCTION` to view the schema of the file, you will see a dummy column `__dummy_col`, which can be ignored.
- If the format specified for the TVF is CSV and the file being read is not empty but the first line of the file is empty, an error will be prompted: `The first line is empty, can not parse column numbers`. This is because the schema cannot be parsed from the first line of the file.

More help

For more detailed syntax on INSERT INTO, refer to the INSERT INTO command manual. You can also type `HELP INSERT` at the MySQL client command line for further information.

MySQL Load

Apache Doris is compatible with the MySQL protocol and supports the use of the standard MySQL LOAD DATA syntax to import local files. MySQL Load is a synchronous import method where the import result is returned upon completion. That means users can tell whether the import suceeds from the returned result. Generally, the MySQL Load method can be used to ingest files smaller than 10GB in size. For files larger than 10GB, it is recommended to split them into smaller ones. MySQL Load ensures the atomicity of a batch of import tasks, meaning that either all imports succeed or all imports fail.

Applicable scenarios

Supported format

MySQL Load is primarily designed for importing CSV files from the client's local machine or importing data from data streams through programs.

Restrictions

When importing CSV files, it is important to differentiate between null values and empty strings (''):

- Null values are represented by the escape sequence . For example, in `a,\N,b`, the middle column represents a null value.
- Empty strings are represented directly as empty, such as `a, ,b`, where the middle column represents an empty string.

Implementation

MySQL Load is similar to Stream Load in terms of functionality. They both involve importing local files into the Doris cluster. Therefore, the implementation of MySQL Load reuses the basic import capabilities of Stream Load.

The main processes of MySQL Load include:

1. The user submits a LOAD DATA request to the frontend (FE), which performs the parsing and encapsulates the request into a Stream Load task.
2. The FE selects a backend (BE) node and sends the Stream Load request to it.
3. Meanwhile, the FE reads the local file data from the MySQL client in an asynchronous and streaming manner and sends it in real time to the HTTP request of the Stream Load.
4. Once the data transfer from the MySQL client is complete, the FE waits for the Stream Load to finish and displays the import result (success or failure) to the client.

Get started

Preparations

MySQL Load requires INSERT permission on the target table. You can grant permissions to user account using the GRANT command.

Create a MySQL Load job

1. Prepare the test data

Create a data file `client_local.csv` containing the following sample data:

```
1,10
2,20
3,30
4,40
5,50
6,60
```

2. Connect to MySQL client

Connect to the MySQL client before executing the LOAD DATA command:

```
mysql --local-infile  -h <fe_ip> -P <fe_query_port> -u root -D testdb
```

> **Specific parameter options need to be used during the connection:**
>
> 1. When connecting to the MySQL client, `--local-infile` must be included, otherwise an error might be thrown.
> 2. When connecting via JDBC, `allowLoadLocalInfile=true` must be specified in the URL.

3. Create the test table

Create a table as follows in Doris:

```
CREATE TABLE testdb.t1 (
    pk     INT,
    v1     INT SUM
) AGGREGATE KEY (pk)
DISTRIBUTED BY hash (pk);
```

4. Run the LOAD DATA command

After connecting to the MySQL client, create a Load job. The command is as follows:

```
LOAD DATA LOCAL
INFILE 'client_local.csv'
INTO TABLE testdb.t1
COLUMNS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

View results

MySQL Load is a synchronous import method, and the results of the import are returned to the user in the command-line interface. If the import execution fails, it will display specific error messages.

Below is an example of a successful import result, which returns the number of imported rows:

```
Query OK, 6 row affected (0.17 sec)
Records: 6  Deleted: 0  Skipped: 0  Warnings: 0
```

When there are exceptions during the import, the corresponding error will be displayed on the client:

```
ERROR 1105 (HY000): errCode = 2, detailMessage = [DATA_QUALITY_ERROR]too many filtered rows with load id b612907c-ccf4-4ac2-82fe-107ece655f0f
```

The `loadId` is included in the error message, based on which you can view the detailed information via the `show load warnings` command:

```
show load warnings where label='b612907c-ccf4-4ac2-82fe-107ece655f0f';
```

Cancel a MySQL Load job

Doris does not allow manual cancellation of MySQL Load jobs. In the event of a timeout or import error, the corresponding MySQL Load job will be automatically cancelled by the system.

Manual

Syntax

The syntax for LOAD DATA is as follows:

```
LOAD DATA LOCAL
INFILE '<load_data_file>'
INTO TABLE [<db_name>.]<table_name>
[PARTITION (partition_name [, partition_name] ...)]
[COLUMNS TERMINATED BY '<column_terminated_operator>']
[LINES TERMINATED BY '<line_terminated_operator>']
[IGNORE <ignore_lines> LINES]
[(col_name_or_user_var[, col_name_or_user_var] ...)]
[SET col_name={expr | DEFAULT}[, col_name={expr | DEFAULT}] ...]
[PROPERTIES (key1 = value1 [, key2=value2]) ]
```

Descriptions of modules in the Load job:

Module	Description
INFILE	This specifies the local file path, which can be either a relative path or an absolute path.Currently, load_data_file only supports a single file.
INTO TABLE	This specifies the database and table, and the database name can be omitted.
PARTITION	This specifies the target partition. If the user can determine the partition corresponding to the data, it is recommended to specify this. Data that does not fit into the specified partitions will be filtered out.
COLUMNS TERMINATED BY	This specifies the column delimiter.
LINE TERMINATED BY	This specifies the row delimiter.

Module	Description
IGNORE num LINES	This specifies the number of rows to skip in the CSV import, typically specified as 1 to skip the header.
col_name_or_user_var	This specifies the column mapping syntax. For more information, refer to Column Mapping.
PROPERTIES	Parameters for the Load.

Parameters

By the `PROPERTIES (key1 = value1 [, key2=value2])` syntax, you can configure the parameters for the Load.

Parameter	Description
max_filter_ratio	The maximum filtering rate allowed. Must be between 0 and 1, inclusive. The default value is 0, indicating no tolerance for any error rows.
timeout	The import timeout, measured in seconds. The default value is 600. The range allowed is from 1s to 259200s.
strict_mode	Whether to enable strict mode for this import. Disabled by default.
timezone	The time zone for this import. The default time zone is UTC+8. This parameter will affect the results of any time zone-related functions involved in the import.
exec_mem_limit	Memory limit for the import, defaults to 2GB, measured in bytes.
trim_double_quotes	Boolean, defaults to false. If this is set to true, the outermost double quotes will be trimmed from each field in the import file.
enclose	This specifies the enclosure character. When a CSV data field contains line breaks or column delimiters, you can specify a single-byte character as the enclosure character to prevent accidental truncation.For example, if the column delimiter is “,”, and the enclosure character is ““, in data”a,’b,c’”, “b,c” will be parsed as one field.
escape	This specifies the escape character. This is used when the data contains the same character as the enclosure character, which needs to be treated as part of the field.For example, if the data is “a,‘b,’c’”, the enclosure character is ““, and you want”b,’c” to be parsed as one field, you need to specify a single-byte escape character, such as “”, to modify the data to”a,’b,'c’”.

Example

Specify load timeout

You can adjust the import timeout by specifying `timeout` in PROPERTIES. For example, set it to 100s:

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PROPERTIES ("timeout"="100");
```

Specify allowable error rate

You can adjust the allowable error rate by specifying `max_filter_ratio` in PROPERTIES. For example, set it to 20%:

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PROPERTIES ("max_filter_ratio"="0.2");
```

Import column mapping

The following example adjusts the order of columns in the CSV file.

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
(k2, k1, v1);
```

Specify column and row delimiters

You can specify the column and row delimiters using the `COLUMNS TERMINATED BY` and `LINES TERMINATED BY` clauses. In the following example, (,) and () are used as the column and row delimiters, respectively.

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
COLUMNS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

Specify target partition

You can specify the target partition for the import using the `PARTITION` clause. In the following example, data will be loaded into the specified partitions ‘p1’ and ‘p2’, and any data that does not belong to these two partitions will be filtered out:

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PARTITION (p1, p2);
```

Specify time zone

You can specify the `timezone` in PROPERTIES. In the following example, the timezone is set to Africa/Abidjan:

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PROPERTIES ("timezone"="Africa/Abidjan");
```

Specify the memory limit for the import

You can specify the memory limit for the import by the `exec_mem_limit` parameter in PROPERTIES. In the following example, the memory limit is set to 10G:

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PROPERTIES ("exec_mem_limit"="10737418240");
```

More help

For more detailed syntax and best practices related to MySQL Load, refer to the MySQL Load command manual.

Loading Data in JSON Format

Doris supports importing data in JSON format. This document mainly describes the precautions when importing data in JSON format.

Supported import methods

Currently, only the following import methods support data import in JSON format:

- Through S3 table function import statement: insert into table select * from S3();
- Import the local JSON format file through STREAM LOAD.
- Subscribe and consume JSON format in Kafka via ROUTINE LOAD information.

Other ways of importing data in JSON format are not currently supported.

Supported JSON Formats

Currently only the following two JSON formats are supported:

1. Multiple rows of data represented by Array

   JSON format with Array as root node. Each element in the Array represents a row of data to be imported, usually an Object. An example is as follows:

   ```
   [
       { "id": 123, "city" : "beijing"},
       { "id": 456, "city" : "shanghai"},
       ...
   ]
   ```

   ```
   [
       { "id": 123, "city" : { "name" : "beijing", "region" : "haidian"}},
       { "id": 456, "city" : { "name" : "beijing", "region" : "chaoyang"}},
       ...
   ]
   ```

   This method is typically used for Stream Load import methods to represent multiple rows of data in a batch of imported data.

   This method must be used with the setting `strip_outer_array=true`. Doris will expand the array when parsing, and then parse each Object in turn as a row of data.

2. A single row of data represented by Object JSON format with Object as root node. The entire Object represents a row of data to be imported. An example is as follows:

   ```
   { "id": 123, "city" : "beijing"}
   ```

```
{ "id": 123, "city" : { "name" : "beijing", "region" : "haidian" }}
```

This method is usually used for the Routine Load import method, such as representing a message in Kafka, that is, a row of data.

3. Multiple lines of Object data separated by a fixed delimiter

   A row of data represented by Object represents a row of data to be imported. The example is as follows:

   ```
   { "id": 123, "city" : "beijing"}
   { "id": 456, "city" : "shanghai"}
   ...
   ```

   This method is typically used for Stream Load import methods to represent multiple rows of data in a batch of imported data.

   This method must be used with the setting `read_json_by_line=true`, the special delimiter also needs to specify the `line_delimiter` parameter, the default is `\n`. When Doris parses, it will be separated according to the delimiter, and then parse each line of Object as a line of data.

streaming_load_json_max_mb parameters

Some data formats, such as JSON, cannot be split. Doris must read all the data into the memory before parsing can begin. Therefore, this value is used to limit the maximum amount of data that can be loaded in a single Stream load.

The default value is 100, The unit is MB, modify this parameter by referring to the BE configuration.

fuzzy_parse parameters

In STREAM LOAD `fuzzy_parse` parameter can be added to speed up JSON Data import efficiency.

This parameter is usually used to import the format of **multi-line data represented by Array**, so it is generally used with `strip_outer_array=true`.

This feature requires that each row of data in the Array has exactly the same order of fields. Doris will only parse according to the field order of the first row, and then access the subsequent data in the form of subscripts. This method can improve the import efficiency by 3-5X.

JSON Path

Doris supports extracting data specified in JSON through JSON Path.

Note: Because for Array type data, Doris will expand the array first, and finally process it in a single line according to the Object format. So the examples later in this document are all explained with Json data in a single Object format.

- do not specify JSON Path

If JSON Path is not specified, Doris will use the column name in the table to find the element in Object by default. An example is as follows:

The table contains two columns: `id`, `city`

The JSON data is as follows:

```
{ "id": 123, "city" : "beijing"}
```

Then Doris will use `id`, `city` for matching, and get the final data `123` and `beijing`.

If the JSON data is as follows:

```
{ "id": 123, "name" : "beijing"}
```

Then use `id`, `city` for matching, and get the final data `123` and `null`.

- Specify JSON Path

  Specify a set of JSON Path in the form of a JSON data. Each element in the array represents a column to extract. An example is as follows:

  ```
  ["$.id", "$.name"]
  ```

  ```
  ["$.id.sub_id", "$.name[0]", "$.city[0]"]
  ```

  Doris will use the specified JSON Path for data matching and extraction.

- matches non-primitive types

  The values that are finally matched in the preceding examples are all primitive types, such as integers, strings, and so on. Doris currently does not support composite types, such as Array, Map, etc. So when a non-basic type is matched, Doris will convert the type to a string in Json format and import it as a string type. An example is as follows:

  The JSON data is:

  ```
  { "id": 123, "city" : { "name" : "beijing", "region" : "haidian" }}
  ```

  JSON Path is `["$.city"]`. The matched elements are:

  ```
  { "name" : "beijing", "region" : "haidian" }
  ```

  The element will be converted to a string for subsequent import operations:

  ```
  "{'name':'beijing','region':'haidian'}"
  ```

- match failed

When the match fails, `null` will be returned. An example is as follows:

The JSON data is:

```
{ "id": 123, "name" : "beijing"}
```

JSON Path is `["$.id", "$.info"]`. The matched elements are `123` and `null`.

Doris currently does not distinguish between null values represented in JSON data and null values produced when a match fails. Suppose the JSON data is:

```
{ "id": 123, "name" : null }
```

The same result would be obtained with the following two JSON Paths: `123` and `null`.

```
["$.id", "$.name"]
```

```
["$.id", "$.info"]
```

- Exact match failed

  In order to prevent misoperation caused by some parameter setting errors. When Doris tries to match a row of data, if all columns fail to match, it considers this to be an error row. Suppose the Json data is:

  ```
  { "id": 123, "city" : "beijing" }
  ```

  If the JSON Path is written incorrectly as (or if the JSON Path is not specified, the columns in the table do not contain `id` and `city`):

  ```
  ["$.ad", "$.infa"]
  ```

  would cause the exact match to fail, and the line would be marked as an error line instead of yielding `null, null`.

JSON Path and Columns

JSON Path is used to specify how to extract data in JSON format, while Columns specifies the mapping and conversion relationship of columns. Both can be used together.

In other words, it is equivalent to rearranging the columns of a JSON format data according to the column order specified in JSON Path through JSON Path. After that, you can map the rearranged source data to the columns of the table through Columns. An example is as follows:

Data content:

```
{"k1": 1, "k2": 2}
```

Table Structure:

```
k2 int, k1 int
```

Import statement 1 (take Stream Load as an example):

```
curl -v --location-trusted -u root: -H "format: json" -H "jsonpaths: [\"$.k2\", \"$.k1\"]" -T example.json http:/
/127.0.0.1:8030/api/db1/tbl1/_stream_load
```

In import statement 1, only JSON Path is specified, and Columns is not specified. The role of JSON Path is to extract the JSON data in the order of the fields in the JSON Path, and then write it in the order of the table structure. The final imported data results are as follows:

```
+------+------+
| k1 | k2 |
+------+------+
| 2 | 1 |
+------+------+
```

You can see that the actual k1 column imports the value of the “k2” column in the JSON data. This is because the field name in JSON is not equivalent to the field name in the table structure. We need to explicitly specify the mapping between the two.

Import statement 2:

```
curl -v --location-trusted -u root: -H "format: json" -H "jsonpaths: [\"$.k2\", \"$.k1\"]" -H "columns: k2, k1 " -T example.json
http://127.0.0.1:8030/api/db1/tbl1/_stream_load
```

Compared with the import statement 1, the Columns field is added here to describe the mapping relationship of the columns, in the order of `k2, k1`. That is, after extracting in the order of fields in JSON Path, specify the value of column k2 in the table for the first column, and the value of column k1 in the table for the second column. The final imported data results are as follows:

```
+------+------+
| k1 | k2 |
+------+------+
| 1 | 2 |
+------+------+
```

Of course, as with other imports, column transformations can be performed in Columns. An example is as follows:

```
curl -v --location-trusted -u root: -H "format: json" -H "jsonpaths: [\"$.k2\", \"$.k1\"]" -H "columns: k2, tmp_k1 , k1 = tmp_k1 *
100" -T example.json http://127.0.0.1:8030/api/db1/tbl1/_stream_load
```

The above example will import the value of k1 multiplied by 100. The final imported data results are as follows:

```
+------+------+
| k1 | k2 |
+------+------+
```

```
| 100 | 2 |
+------+------+
```

JSON root

Doris supports extracting data specified in JSON through JSON root.

Note: Because for Array type data, Doris will expand the array first, and finally process it in a single line according to the Object format. So the examples later in this document are all explained with Json data in a single Object format.

- do not specify JSON root

  If JSON root is not specified, Doris will use the column name in the table to find the element in Object by default. An example is as follows:

  The table contains two columns: `id`, `city`

  The JSON data is as follows:

  ```
  { "id": 123, "name" : { "id" : "321", "city" : "shanghai" }}
  ```

  Then use `id`, `city` for matching, and get the final data `123` and `null`

- Specify JSON root

  When the import data format is JSON, you can specify the root node of the JSON data through json_root. Doris will extract the elements of the root node through json_root for parsing. Default is empty.

  Specify JSON root `-H "json_root: $.name"`. The matched elements are:

  ```
  { "id" : "321", "city" : "shanghai" }
  ```

  The element will be treated as new JSON for subsequent import operations,and get the final data 321 and shanghai

NULL and Default values

Example data is as follows:

```
[
    {"k1": 1, "k2": "a"},
    {"k1": 2},
    {"k1": 3, "k2": "c"}
]
```

The table structure is: `k1 int null, k2 varchar(32) null default "x"`

The import statement is as follows:

```
curl -v --location-trusted -u root: -H "format: json" -H "strip_outer_array: true" -T example.json
http://127.0.0.1:8030/api/db1/tbl1/_stream_load
```

The import results that users may expect are as follows, that is, for missing columns, fill in the default values.

```
+------+------+
| k1 | k2 |
+------+------+
| 1 | a |
+------+------+
| 2 | x |
+------+------+
|3|c|
+------+------+
```

But the actual import result is as follows, that is, for the missing column, NULL is added.

```
+------+------+
| k1 | k2 |
+------+------+
| 1 | a |
+------+------+
| 2 | NULL |
+------+------+
|3|c|
+------+------+
```

This is because Doris doesn't know "the missing column is column k2 in the table" from the information in the import statement. If you want to import the above data according to the expected result, the import statement is as follows:

```
curl -v --location-trusted -u root: -H "format: json" -H "strip_outer_array: true" -H "jsonpaths: [\"$.k1\", \"$.k2\"]" - H
"columns: k1, tmp_k2, k2 = ifnull(tmp_k2, 'x')" -T example.json http://127.0.0.1:8030/api/db1/tbl1/_stream_load
```

Application example

Stream Load

Because of the inseparability of the JSON format, when using Stream Load to import a JSON format file, the file content will be fully loaded into the memory before processing begins. Therefore, if the file is too large, it may take up more memory.

Suppose the table structure is:

```
id INT NOT NULL,
city VARHCAR NULL,
code INT NULL
```

1. Import a single row of data 1

    ```
    {"id": 100, "city": "beijing", "code" : 1}
    ```

- do not specify JSON Path

```
curl --location-trusted -u user:passwd -H "format: json" -T data.json
http://localhost:8030/api/db1/tbl1/_stream_load
```

Import result:

```
100 beijing 1
```

- Specify JSON Path

```
curl --location-trusted -u user:passwd -H "format: json" -H "jsonpaths: [\"$.id\",\"$.city\",\"$.code\"]" - T
data.json http://localhost:8030/api/db1/tbl1/_stream_load
```

Import result:

```
100 beijing 1
```

2. Import a single row of data 2

```
{"id": 100, "content": {"city": "beijing", "code": 1}}
```

- Specify JSON Path

```
curl --location-trusted -u user:passwd -H "format: json" -H "jsonpaths:
[\"$.id\",\"$.content.city\",\"$.content.code\ "]" -T data.json http://localhost:8030/api/db1/tbl1/_stream_load
```

Import result:

```
100 beijing 1
```

3. Import multiple rows of data as Array

```
[
    {"id": 100, "city": "beijing", "code" : 1},
    {"id": 101, "city": "shanghai"},
    {"id": 102, "city": "tianjin", "code" : 3},
    {"id": 103, "city": "chongqing", "code" : 4},
    {"id": 104, "city": ["zhejiang", "guangzhou"], "code" : 5},
    {
        "id": 105,
        "city": {
            "order1": ["guangzhou"]
        },
        "code" : 6
    }
]
```

- Specify JSON Path

```
curl --location-trusted -u user:passwd -H "format: json" -H "jsonpaths: [\"$.id\",\"$.city\",\"$.code\"]" - H
"strip_outer_array: true" -T data.json http://localhost:8030/api/db1/tbl1/_stream_load
```

Import result:

```
100 beijing 1
101 shanghai NULL
102 tianjin 3
103 chongqing 4
104 ["zhejiang","guangzhou"] 5
105 {"order1":["guangzhou"]} 6
```

4. Import multi-line data as multi-line Object

```
{"id": 100, "city": "beijing", "code" : 1}
{"id": 101, "city": "shanghai"}
{"id": 102, "city": "tianjin", "code" : 3}
{"id": 103, "city": "chongqing", "code" : 4}
```

StreamLoad import:

```
curl --location-trusted -u user:passwd -H "format: json" -H "read_json_by_line: true" -T data.json
http://localhost:8030/api/db1/tbl1/_stream_load
```

```
   Import result:

100     beijing                    1
101     shanghai                   NULL
102     tianjin                    3
103     chongqing                  4
```

5. Transform the imported data

The data is still the multi-line data in Example 3, and now it is necessary to add 1 to the `code` column in the imported data before importing.

```
curl --location-trusted -u user:passwd -H "format: json" -H "jsonpaths: [\"$.id\",\"$.city\",\"$.code\"]" - H "strip_outer_array:
true" -H "columns: id, city, tmpc, code=tmpc+1" -T data.json http://localhost:8030/api/db1/tbl1/_stream_load
```

Import result:

```
100 beijing 2
101 shanghai NULL
102 tianjin 4
103 chongqing 5
104 ["zhejiang","guangzhou"] 6
105 {"order1":["guangzhou"]} 7
```

6. Import Array by JSON Since the Rapidjson handles decimal and largeint numbers which will cause precision problems, we suggest you to use JSON string to import data to `array<decimal>` or `array<largeint>` column.

```
{"k1": 39, "k2": ["-818.2173181"]}
```

```
{"k1": 40, "k2": ["10000000000000000000.1111111222222222"]}
```

```
curl --location-trusted -u root:  -H "max_filter_ratio:0.01" -H "format:json" -H "timeout:300" -T test_decimal.json
http://localhost:8035/api/example_db/array_test_decimal/_stream_load
```

Import result:

```
MySQL > select * from array_test_decimal;
+------+---------------------------------+
| k1   | k2                              |
+------+---------------------------------+
|   39 | [-818.2173181]                  |
|   40 | [100000000000000000.001111111]  |
+------+---------------------------------+
```

```
{"k1": 999, "k2": ["769598369377499328797635736817927017090", "260170428259378916929104315210385212270"]}
```

```
curl --location-trusted -u root:  -H "max_filter_ratio:0.01" -H "format:json" -H "timeout:300" -T test_largeint.json
http://localhost:8035/api/example_db/array_test_largeint/_stream_load
```

Import result:

```
MySQL > select * from array_test_largeint;
+------+-----------------------------------------------------------------------------------+
| k1   | k2                                                                                |
+------+-----------------------------------------------------------------------------------+
|  999 | [769598369377499328797635736817927017090, 260170428259378916929104315210385212270]  |
+------+-----------------------------------------------------------------------------------+
```

Routine Load

The processing principle of Routine Load for JSON data is the same as that of Stream Load. It is not repeated here.

For Kafka data sources, the content in each Massage is treated as a complete JSON data. If there are multiple rows of data represented in Array format in a Massage, multiple rows will be imported, and the offset of Kafka will only increase by 1. If an Array format Json represents multiple lines of data, but the Json parsing fails due to the wrong Json format, the error line will only increase by 1 (because the parsing fails, in fact, Doris cannot determine how many lines of data are contained in it, and can only error by one line data record)

Migrating Data from Other OLAP

To migrate data from other OLAP systems to Doris, you have a variety of options:

- For systems like Hive/Iceberg/Hudi, you can leverage Multi-Catalog to map them as external tables and then use “Insert Into” to import the data into Doris.
- You can export data from the OLAP system into formats like CSV, and then import the data files into Doris.
- You can also leverage the connectors of the OLAP systems, use tools like Spark / Flink, and then call the corresponding Doris Connector to write data into Doris.

Loading Transaction and Atomicity

Use Cases

All loading tasks in Doris are atomic, which means that a loading job either succeeds completely or fails completely. There won't be a situation where only part of the data is load successfully. Additionally, Doris ensures atomicity for loading multiple tables within the same loading task. Moreover, Doris ensures data loading without loss or duplication through the use of labels. For simple loading tasks, no additional configuration or operations are required. For materialized views associated with tables, atomicity and consistency with the base table are also guaranteed. Doris provides additional transaction control for the following scenarios:

1. If a user needs to combine multiple `INSERT INTO` into a single transaction for the same target table, they can use the `BEGIN` and `COMMIT` commands.
2. If a user needs to combine multiple stream load into a single transaction, they can use the two-phase commit mode of Stream Load.
3. Atomicity of multi-table load with Broker Load.

Basic Principles

In a Doris loading task, the Backend (BE) submits the Tablet IDs of the successful writes to the Frontend (FE). FE determines the success of the load based on the number of successful tablet replicas. If loading is successful, the transaction is committed and the data load becomes visible. If it fails, the transaction is rolled back and the corresponding tablets are cleaned up.

Label Mechanism

Doris loading jobs can be assigned a label. This label is usually a user-defined string with certain business logic attributes.

The main purpose of the label is to uniquely identify a loading task and ensure that the same label is loaded successfully only once.

The label mechanism ensures that load data is not lost or duplicated. When combined with an upstream data source that guarantees At-Least-Once semantics, the label mechanism in Doris can ensure Exactly-Once semantics.

Labels are unique within a database. The default retention period for labels is 3 days. After 3 days, completed labels are automatically cleaned up and can be reused.

Quick Start

Insert Into

1. Create Table

```
CREATE TABLE testdb.test_table(
    user_id            BIGINT       NOT NULL COMMENT "User ID",
    name               VARCHAR(20)  NOT NULL COMMENT "User Name",
    age                INT                   COMMENT "User Age"
)
DUPLICATE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 10;
```

Create a table with the same schema for the failed example:

```
CREATE TABLE testdb.test_table2 LIKE testdb.test_table;
```

2. Successful Load Example

```
BEGIN;

-- INSERT #1
INSERT INTO testdb.test_table (user_id, name, age)
VALUES (1, "Emily", 25),
       (2, "Benjamin", 35),
       (3, "Olivia", 28),
       (4, "Alexander", 60),
       (5, "Ava", 17);

-- INSERT #2
INSERT INTO testdb.test_table (user_id, name, age)
VALUES (6, "William", 69),
       (7, "Sophia", 32),
       (8, "James", 64),
       (9, "Emma", 37),
       (10, "Liam", 64);

COMMIT;
```

Load result: The loading task starts with the `PREPARE` status and becomes `VISIBLE` only after the COMMIT.

```
// BEGIN
Query OK, 0 rows affected (0.001 sec)
{'label':'txn_insert_2aeac5519bd549a1-a72fe4001c56e10c', 'status':'PREPARE', 'txnId':''}

// INSERT #1
Query OK, 5 rows affected (0.017 sec)
{'label':'txn_insert_2aeac5519bd549a1-a72fe4001c56e10c', 'status':'PREPARE', 'txnId':'10060'}

// INSERT #2
Query OK, 5 rows affected (0.007 sec)
{'label':'txn_insert_2aeac5519bd549a1-a72fe4001c56e10c', 'status':'PREPARE', 'txnId':'10060'}

// COMMIT
Query OK, 0 rows affected (1.013 sec)
{'label':'txn_insert_2aeac5519bd549a1-a72fe4001c56e10c', 'status':'VISIBLE', 'txnId':'10060'}
```

Verify the data:

```
MySQL [testdb]> SELECT * FROM testdb.test_table2;
Empty set (0.019 sec)
```

Stream Load

1. Enable two-phase commit by setting `two_phase_commit:true` in the HTTP Header.

```
curl --location-trusted -u user:passwd -H "two_phase_commit:true" -T test.txt http://fe_host:http_port/api/{db}/{table}/_stream_load
{
    "TxnId": 18036,
    "Label": "55c8ffc9-1c40-4d51-b75e-f2265b3602ef",
    "TwoPhaseCommit": "true",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 100,
    "NumberLoadedRows": 100,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 1031,
    "LoadTimeMs": 77,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 1,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 58,
    "CommitAndPublishTimeMs": 0
}
```

2. Trigger the commit operation for a transaction (can be sent to FE or BE).

- Specify the transaction using the Transaction ID:

```
curl -X PUT --location-trusted -u user:passwd -H "txn_id:18036" -H "txn_operation:commit"
http://fe_host:http_port/api/{db}/{table}/stream_load2pc
{
    "status": "Success",
    "msg": "transaction [18036] commit successfully."
}
```

- Specify the transaction using the label:

```
curl -X PUT --location-trusted -u user:passwd -H "label:55c8ffc9-1c40-4d51-b75e-f2265b3602ef" -H "txn_operation:commit"
http://fe_host:http_port/api/{db}/{table}/_stream_load_2pc
{
    "status": "Success",
    "msg": "label [55c8ffc9-1c40-4d51-b75e-f2265b3602ef] commit successfully."
}
```

3. Trigger the abort operation for a transaction (can be sent to FE or BE).

- Specify the transaction using the Transaction ID:

```
curl -X PUT --location-trusted -u user:passwd -H "txn_id:18037" -H "txn_operation:abort"
http://fe_host:http_port/api/{db}/{table}/stream_load2pc
{
    "status": "Success",
    "msg": "transaction [18037] abort successfully."
}
```

- Specify the transaction using the label:

```
curl -X PUT --location-trusted -u user:passwd -H "label:55c8ffc9-1c40-4d51-b75e-f2265b3602ef" -H "txn_operation:abort"
http://fe_host:http_port/api/{db}/{table}/stream_load2pc
{
    "status": "Success",
    "msg": "label [55c8ffc9-1c40-4d51-b75e-f2265b3602ef] abort successfully."
}
```

Broker Load

All Broker Load tasks are atomic and ensure atomicity even when loading multiple tables within the same task. The Label mechanism can be used to ensure data load without loss or duplication.

The following example demonstrates loading data from HDFS by using wildcard patterns to match two sets of files and load them into two different tables.

```
LOAD LABEL example_db.label2
(
    DATA INFILE("hdfs://hdfs_host:hdfs_port/input/file-10*")
    INTO TABLE `my_table1`
    PARTITION (p1)
    COLUMNS TERMINATED BY ","
    (k1, tmp_k2, tmp_k3)
    SET (
        k2 = tmp_k2 + 1,
        k3 = tmp_k3 + 1
    )
    DATA INFILE("hdfs://hdfs_host:hdfs_port/input/file-20*")
    INTO TABLE `my_table2`
    COLUMNS TERMINATED BY ","
    (k1, k2, k3)
)
WITH BROKER hdfs
(
    "username"="hdfs_user",
    "password"="hdfs_password"
);
```

The wildcard pattern is used to match and load two sets of files, `file-10*` and `file-20*`, into `my_table1` and `my_table2` respectively. In the case of `my_table1`, the load is specified to the `p1` partition, and the values of thesecond and third columns in the source file are incremented by 1 before being loaded.

Best Practices

Labels are typically set in the format of `business_logic+time`, such as `my_business1_20220330_125000`.

This label is commonly used to represent a batch of data generated by the business logic `my_business1` at `2022-03-30 12:50:00`. By setting this label, the business can query the loading task status using the label to determine whether the data batch at that specific time has been successfully load. If the load fails, the label can be used to retry the load.

INSERT INTO supports loading the result of a Doris query into another table. INSERT INTO is a synchronous load method that returns the load result after execution. The success or failure of the load can be determined based on the response. INSERT INTO ensures the atomicity of the loading task, either all succeed or all fail load.

Converting Source Data

Usage Scenarios

During the import process, Doris supports some transformations on the source data, including mapping, conversion, preceding filtering, and post-filtering.

- Mapping: Import column A from the source data into column B in the target table.
- Conversion: Calculate the values in the target column based on the columns in the source data using an expression. Custom functions are supported in the expression.
- Preceding Filtering: Filter rows in the source data and only import rows that meet the filtering conditions.
- Post-Filtering: Filter rows in the result and only import rows that meet the filtering conditions.

Quick Start

BROKER LOAD

```
LOAD LABEL example_db.label1
(
    DATA INFILE("bos://bucket/input/file")
    INTO TABLE `my_table`
    (k1, k2, tmpk3)
    PRECEDING FILTER k1 = 1
    SET (
        k3 = tmpk3 + 1
    )
    WHERE k1 > k2
)
WITH BROKER bos
(
    ...
);
```

STREAM LOAD

```
curl
--location-trusted
-u user:passwd
-H "columns: k1, k2, tmpk3, k3 = tmpk3 + 1"
-H "where: k1 > k2"
-T file.txt
http://host:port/api/testDb/testTbl/_stream_load
```

ROUTINE LOAD

```
CREATE ROUTINE LOAD example_db.label1 ON my_table
COLUMNS(k1, k2, tmpk3, k3 = tmpk3 + 1),
```

```
PRECEDING FILTER k1 = 1,
WHERE k1 > k2
...
```

Reference Manual

Loading Syntax

Stream Load

Add `columns` and `where` parameters in the HTTP header.

- `columns` specify column mapping and value transformation.
- `where` specify post-filtering.

Stream load does not support preceding filtering.

Example:

```
curl
--location-trusted
-u user:passwd
-H "columns: k1, k2, tmpk3, k3 = tmpk3 + 1"
-H "where: k1 > k2"
-T file.txt
http://host:port/api/testDb/testTbl/_stream_load
```

Broker Load

Define data transformation in the SQL statement, including:

- `(k1, k2, tmpk3)` specifies column mapping.
- `PRECEDING FILTER` specifies preceding filtering.
- `SET` specifies column transformation.
- `WHERE` specifies post-filtering.

```
LOAD LABEL example_db.label1
(
    DATA INFILE("bos://bucket/input/file")
    INTO TABLE `my_table`
    (k1, k2, tmpk3)
    PRECEDING FILTER k1 = 1
    SET (
        k3 = tmpk3 + 1
    )
    WHERE k1 > k2
)
WITH BROKER bos
(
    ...
);
```

Routine Load

Define data transformation in the SQL statement, including:

- `COLUMNS` specifies column mapping and column transformation.
- `PRECEDING FILTER` specifies preceding filtering.
- `WHERE` specifies post-filtering.

```
CREATE ROUTINE LOAD example_db.label1 ON my_table
COLUMNS(k1, k2, tmpk3, k3 = tmpk3 + 1),
PRECEDING FILTER k1 = 1,
WHERE k1 > k2
...
```

Insert Into

Insert Into can perform data transformation directly in the `SELECT` statement, and add a `WHERE` clause for data filtering.

Column Mapping

The purpose of column mapping is to describe the information of each column in the load file, which is equivalent to defining names for the columns in the source data. By describing the column mapping relationship, we can load source files with different column orders and column numbers into Doris. Let's illustrate it through examples:

Suppose the source file has 4 columns with the following contents (the column names in the table header are for illustration purposes only and are not actually present in the file):

Column 1	Column 2	Column 3	Column 4
1	100	beijing	1.1
2	200	shanghai	1.2
3	300	guangzhou	1.3
4		chongqing	1.4

> **Note: `\N` represents null in the source file.**

1. Adjusting Mapping Order
2. Suppose there are 4 columns in the table: `k1, k2, k3, k4`. The desired load mapping is as follows:

```
Column 1 -> k1
Column 2 -> k3
Column 3 -> k2
Column 4 -> k4
```

3. The order of column mapping should be as follows:

```
(k1, k3, k2, k4)
```

4. The number of columns in the source file is greater than the number of columns in the table.
5. Suppose there are 3 columns in the table: `k1, k2, k3`. The desired load mapping is as follows:

```
Column 1 -> k1
Column 2 -> k3
Column 3 -> k2
```

6. The order of column mapping should be as follows:

```
(k1, k3, k2, tmpk4)
```

7. Here, `tmpk4` is a custom column name that doesn't exist in the table. Doris will ignore this non-existing column name.
8. The number of columns in the source file is less than the number of columns in the table, and default values will be used to fill the missing columns.
9. Suppose there are 5 columns in the table: `k1, k2, k3, k4, k5`. The desired load mapping is as follows:

```
Column 1 -> k1
Column 2 -> k3
Column 3 -> k2
```

10. Here, only the first 3 columns from the source file will be used. The columns `k4` and `k5` are expected to be filled with default values.
11. The order of column mapping should be as follows:

```
(k1, k3, k2)
```

12. If the columns `k4` and `k5` have default values, they will be filled accordingly. Otherwise, if the columns are nullable, they will be filled with `null` values. Otherwise, the loading job will report an error.

Pre-filtering

Pre-filtering is a process of filtering the raw data that is read. Currently, it is only supported in BROKER LOAD and ROUTINE LOAD.

Pre-filtering can be applied in the following scenarios:

1. Filtering before transformation: It allows filtering of data before performing column mapping and transformation. This way, unnecessary data can be filtered out in advance.
2. Filtering columns that do not exist in the table: It can be used as a filtering identifier when certain columns are not present in the table.

3. Handling data from multiple tables: For example, if the source data contains data from multiple tables (or data from multiple tables is written to the same Kafka message queue), each row may include a column name that identifies which table the data belongs to. Users can use pre-filtering conditions to select and load the corresponding table data.

Column Transformation

Column transformation enables users to modify the values of columns in the source files. Currently, Doris supports the use of built-in functions and user-defined functions for transformation.

> **Note: User-defined functions belong to a specific database, and when using custom functions for transformation, users need to have read permissions on that database.**

Transformation operations are typically defined in conjunction with column mapping. In the following example, we illustrate the process:

Assume that the source file has 4 columns with the following content (the column names in the header are for descriptive purposes only and are not actually present in the file):

Column 1	Column 2	Column 3	Column 4
1	100	beijing	1.1
2	200	shanghai	1.2
3	300	guangzhou	1.3
	400	chongqing	1.4

1. Load the transformed column values into the table from the source file.

2. Assuming the table has 4 columns: `k1`, `k2`, `k3`, `k4`, and we want the following mapping and transformation relationships:

```
Column 1       -> k1
Column 2 * 100 -> k3
Column 3       -> k2
Column 4       -> k4
```

3. The order of column mapping should be as follows:

```
(k1, tmpk3, k2, k4, k3 = tmpk3 * 100)
```

4. Here, we rename the second column in the source file as `tmpk3` and specify that the value of column `k3` in the table is `tmpk3 * 100`. The final data in the table would be as follows:

k1	k2	k3	k4
1	beijing	10000	1.1
2	shanghai	20000	1.2

k1	k2	k3	k4
3	guangzhou	30000	1.3
null	chongqing	40000	1.4

5. Perform conditional column transformation using the `case when` function.

6. Assuming the table has 4 columns: `k1`, `k2`, `k3`, `k4`, and we want to transform the values `beijing`, `shanghai`, `guangzhou`, `chongqing` in the source data to their corresponding region IDs before loading:

```
Column 1                -> k1
Column 2                -> k2
Column 3 with region ID  -> k3
Column 4                -> k4
```

7. The order of column mapping should be as follows:

```
(k1, k2, tmpk3, k4, k3 = case tmpk3 when "beijing" then 1 when "shanghai" then 2 when "guangzhou" then 3 when "chongqing" then 4
else null end)
```

8. The final data in the table would be as follows:

k1	k2	k3	k4
1	100	1	1.1
2	200	2	1.2
3	300	3	1.3
null	400	4	1.4

9. Transform null values in the source file to 0 during load. Also, perform the region ID transformation as shown in example 2.

10. Assuming the table has `k1, k2, k3, k4` as its four columns. While performing the region ID conversion, we also want to convert null values in the k1 column of the source data to 0 during load:

```
If Column 1 is null, then convert it to 0 -> k1
Column 2                                  -> k2
Column 3                                  -> k3
Column 4                                  -> k4
```

11. The order of column mapping should be as follows:

```
(tmpk1, k2, tmpk3, k4, k1 = ifnull(tmpk1, 0), k3 = case tmpk3 when "beijing" then 1 when "shanghai" then 2 when "guangzhou" then 3
when "chongqing" then 4 else null end)
```

12. The final data in the table would be as follows:

k1	k2	k3	k4
1	100	1	1.1

k1	k2	k3	k4
2	200	2	1.2
3	300	3	1.3
0	400	4	1.4

Post-Filtering

After column mapping and transformation, we can filter out data that we don't want to load into Doris using filtering conditions. Let's illustrate this with an example:

Assume that the source file has 4 columns with the following content (the column names in the table header are for descriptive purposes only and are not actually present):

Column 1	Column 2	Column 3	Column 4
1	100	beijing	1.1
2	200	shanghai	1.2
3	300	guangzhou	1.3
null	400	chongqing	1.4

1. Filtering with default column mapping and transformation.
2. Suppose the table has 4 columns: `k1, k2, k3, k4`. We can define filtering conditions directly without column mapping and transformation. For example, if we only want to load data rows from the source file where the value in the 4th column is greater than 1.2, the filtering condition would be:

```
where k4 > 1.2
```

3. The final data in the table would be as follows:

k1	k2	k3	k4
3	300	guangzhou	1.3
null	400	chongqing	1.4

4. In the default case, Doris performs column mapping in sequential order, so the 4th column in the source file is automatically mapped to the `k4` column in the table.
5. Filtering transformed data.
6. Suppose the table has 4 columns: `k1, k2, k3, k4`. In the column transformation example, we converted province names to IDs. Now, let's say we want to filter out data with an ID of 3. The transformation and filtering conditions would be as follows:

```
(k1, k2, tmpk3, k4, k3 = case tmpk3 when "beijing" then 1 when "shanghai" then 2 when "guangzhou" then 3 when "chongqing" then 4
else null end)
where k3 != 3
```

7. The final data in the table would be as follows:

k1	k2	k3	k4
1	100	1	1.1
2	200	2	1.2
null	400	4	1.4

8. Here, we can observe that the column values used for filtering are the final transformed column values, not the original data.

9. Filtering with multiple conditions.

10. Suppose the table has 4 columns: `k1, k2, k3, k4`. We want to filter out data where the `k1` column is null and the `k4` column is less than 1.2. The filtering condition would be:

```
where k1 is not null and k4 >= 1.2
```

11. The final data in the table would be as follows:

k1	k2	k3	k4
2	200	2	1.2
3	300	3	1.3

Best Practices

Data Quality Issues and Filtering Threshold

The rows of data processed in the load job can be classified into the following three categories:

- Filtered Rows: Data rows that are filtered out due to data quality issues. Data quality issues can include type errors, precision errors, strings exceeding length limits, mismatched file column counts, and data rows filtered out due to missing corresponding partitions.
- Unselected Rows: These are data rows filtered out due to `preceding filter` or `where` column filtering conditions.
- Loaded Rows: Data rows that are successfully loaded.

Doris's load task allows users to set a maximum error rate (`max_filter_ratio`). If the error rate of the loaded data is below the threshold, the error rows will be ignored, and the other correct data will be loaded.

The error rate is calculated as follows:

```
#Filtered Rows / (#Filtered Rows + #Loaded Rows)
```

This means that `Unselected Rows` are not included in the error rate calculation.

Minimum Number of Replicas Loading

Importing data requires more than half of the replicas to be written successfully. However, it is not flexible enough and may cause inconvenience in some scenarios.

For example, in the case of two replicas, to import data, both replicas need to be written successfully. This means that no replica is allowed to be unavailable during the data import process. This greatly affects the availability of the cluster.

In order to solve the above problems, Doris allows users to set the minimum number of write replicas. For the task of importing data, when the number of replicas it successfully writes is greater than or equal to the minimum number of replicas written, the import is successful.

Usage

Min load replica num for single table

You can set the table property `min_load_replica_num` for a single olap table. The valid value of this property must be greater than 0 and not exceed `replication_num`（the number of replicas of the table）. Its default value is -1, indicating that the property is not enabled.

The `min_load_replica_num` of the table can be set when creating the table.

```
CREATE TABLE test_table1
(
    k1 INT,
    k2 INT
)
DUPLICATE KEY(k1)
DISTRIBUTED BY HASH(k1) BUCKETS 5
PROPERTIES
(
    'replication_num' = '2',
    'min_load_replica_num' = '1'
);
```

For an existing table, you can use `ALTER TABLE` to modify its `min_load_replica_num`.

```
ALTER TABLE test_table1
SET ( 'min_load_replica_num' = '1');
```

You can use `SHOW CREATE TABLE` to view the table property `min_load_replica_num`.

```
SHOW CREATE TABLE test_table1;
```

The PROPERTIES of the output will contain `min_load_replica_num`. e.g.

```
Create Table: CREATE TABLE `test_table1` (
  `k1` int(11) NULL,
```

```
  `k2` int(11) NULL
) ENGINE=OLAP
DUPLICATE KEY(`k1`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`k1`) BUCKETS 5
PROPERTIES (
"replication_allocation" = "tag.location.default: 2",
"min_load_replica_num" = "1",
"storage_format" = "V2",
"light_schema_change" = "true",
"disable_auto_compaction" = "false",
"enable_single_replica_compaction" = "false"
);
```

Global min load replica num for all tables

You can set FE configuration item `min_load_replica_num` for all olap tables. The valid value of this configuration item must be greater than 0. Its default value is -1, which means that the global minimum number of load replicas is not enabled.

For a table, if the table property `min_load_replica_num` is valid (>0), then the table will ignore the global configuration `min_load_replica_num`. Otherwise, if the global configuration `min_load_replica_num` is valid (>0), then the minimum number of load replicas for the table will be equal to `min(FE.conf.min_load_replica_num, table.replication_num/2 + 1)`.

For viewing and modification of FE configuration items, you can refer to here.

Other cases

If the table property `min_load_replica_num` is not enabled (<=0), and the global configuration `min_load_replica_num` is not enabled(<=0), then the data import still needs to be successfully written to the majority replica. At this point, the minimum number of write replicas for the table is equal to `table.replication_num/2 + 1`.

Loading Strict Mode

import strict mode

Strict mode (strict_mode) is configured as a parameter in the import operation. This parameter affects the import behavior of certain values and the final imported data.

This document mainly explains how to set strict mode, and the impact of strict mode.

How to set

Strict mode is all False by default, i.e. off.

Different import methods set strict mode in different ways.

1. BROKER LOAD

```
LOAD LABEL example_db.label1
(
    DATA INFILE("bos://my_bucket/input/file.txt")
    INTO TABLE `my_table`
    COLUMNS TERMINATED BY ","
)
WITH BROKER bos
(
    "bos_endpoint" = "http://bj.bcebos.com",
    "bos_accesskey" = "xxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "bos_secret_accesskey"="yyyyyyyyyyyyyyyyyyyyyyyy"
)
PROPERTIES
(
    "strict_mode" = "true"
)
```

2. STREAM LOAD

```
curl --location-trusted -u user:passwd \
-H "strict_mode: true" \
-T 1.txt \
http://host:port/api/example_db/my_table/_stream_load
```

3. ROUTINE LOAD

```
CREATE ROUTINE LOAD example_db.test_job ON my_table
PROPERTIES
(
    "strict_mode" = "true"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
```

```
    "kafka_topic" = "my_topic"
);
```

4. INSERT

   Set via session variables:

```
SET enable_insert_strict = true;
INSERT INTO my_table ...;
```

The role of strict mode

- Restricting the filtering of column type conversions during import.

The strict filtering strategy is as follows:

For column type conversion, if strict mode is turned on, the wrong data will be filtered. The wrong data here refers to: the original data is not `null`, but the result is `null` after column type conversion.

The `column type conversion` mentioned here does not include the `null` value calculated by the function.

For an imported column type that contains range restrictions, if the original data can pass the type conversion normally, but cannot pass the range restrictions, strict mode will not affect it. For example: if the type is `decimal(1,0)` and the original data is 10, it belongs to the range that can be converted by type but is not within the scope of the column declaration. This kind of data strict has no effect on it.

1. Take the column type as TinyInt as an example:

Primitive data type	Primitive data example	Converted value to TinyInt	Strict mode	Result
NULL		NULL	ON or OFF	NULL
Non-null value	“abc” or 2000	NULL	On	Illegal value (filtered)
non-null value	“abc”	NULL	off	NULL
non-null value	1	1	on or off	import correctly

> Description:
>
> 1. Columns in the table allow to import null values
> 2. After `abc` and `2000` are converted to TinyInt, they will become NULL due to type or precision issues. When strict mode is on, this data will be filtered. And if it is closed, `null` will be imported.

2. Take the column type as Decimal(1,0) as an example

Primitive Data Types	Examples of Primitive Data	Converted to Decimal	Strict Mode	Result
Null		null	On or Off	NULL

Primitive Data Types	Examples of Primitive Data	Converted to Decimal	Strict Mode	Result
non-null value	aaa	NULL	on	illegal value (filtered)
non-null value	aaa	NULL	off	NULL
non-null value	1 or 10	1 or 10	on or off	import correctly

> Description:
>
> 1. Columns in the table allow to import null values
> 2. After `abc` is converted to Decimal, it will become NULL due to type problem. When strict mode is on, this data will be filtered. And if it is closed, `null` will be imported.
> 3. Although `10` is an out-of-range value, because its type conforms to the requirements of decimal, strict mode does not affect it. `10` will eventually be filtered in other import processing flows. But not filtered by strict mode.

- Restricting partial column updates to only existing columns

In strict mode, when performing partial column updates, each row of data inserted must have a key that already exists in the table. In non-strict mode, partial column updates can update existing rows with existing keys or insert new rows with non-existing keys.

For example, consider the following table structure:

```
mysql> desc user_profile;
+------------------+-----------------+------+-------+---------+-------+
| Field            | Type            | Null | Key   | Default | Extra |
+------------------+-----------------+------+-------+---------+-------+
| id               | INT             | Yes  | true  | NULL    |       |
| name             | VARCHAR(10)     | Yes  | false | NULL    | NONE  |
| age              | INT             | Yes  | false | NULL    | NONE  |
| city             | VARCHAR(10)     | Yes  | false | NULL    | NONE  |
| balance          | DECIMALV3(9, 0) | Yes  | false | NULL    | NONE  |
| last_access_time | DATETIME        | Yes  | false | NULL    | NONE  |
+------------------+-----------------+------+-------+---------+-------+
```

The table contains the following data:

```
1,"kevin",18,"shenzhen",400,"2023-07-01 12:00:00"
```

When a user uses non-strict mode stream load for partial column updates and inserts the following data into the table:

```
1,500,2023-07-03 12:00:01
3,23,2023-07-03 12:00:02
18,9999999,2023-07-03 12:00:03
```

```
curl  --location-trusted -u root -H "partial_columns:true" -H "strict_mode:false" -H "column_separator:," -H
"columns:id,balance,last_access_time" -T /tmp/test.csv http://host:port/api/db1/user_profile/_stream_load
```

The existing row in the table will be updated, and two new rows will be inserted. For columns in the inserted data that are not specified by the user, if the column has a default value, it will be filled with the default value. Otherwise, if the column allows NULL, it will be filled with a NULL value. If neither condition is met, the insertion will not succeed.

However, when a user uses strict mode stream load for partial column updates and inserts the above data into the table:

```
curl  --location-trusted -u root -H "partial_columns:true" -H "strict_mode:true" -H "column_separator:," -H
"columns:id,balance,last_access_time" -T /tmp/test.csv http://host:port/api/db1/user_profile/_stream_load
```

In this case, since strict mode is enabled and the keys `(3)`, `(18)` in the second and third rows, respectively, do not exist in the original table, the import will fail.

Updating Overview

Data update refers to modifying the value columns in data records with the same key. The handling of data updates varies for different data models:

- **Primary Key (Unique) Model**: The primary key model is specifically designed for data updates. Doris supports two storage implementations: Merge-on-Read (MoR) and Merge-on-Write (MoW). MoR optimizes write performance, while MoW provides better analytical performance. In Doris version 2.0, the default storage method is MoR. The primary key model supports using the `UPDATE` statement for small data updates and also supports batch updates through data loading. Loading methods include Stream Load, Broker Load, Routine Load, and Insert Into, all following the “UPSERT” semantics, meaning if the record does not exist, it is inserted; if it exists, it is updated. Update operations support both full row updates and partial column updates, with full row updates being the default.
- **Aggregate Model**: In the aggregate model, data update is a special use case. When the aggregate function is set to REPLACE or REPLACE_IF_NOT_NULL, data updates can be achieved. The aggregate model only supports updates based on data loading and does not support using the `UPDATE` statement. By setting the aggregate function to REPLACE_IF_NOT_NOT_NULL, partial column update capability can be achieved.

By understanding the data update methods of different models, you can better choose the appropriate update strategy to meet specific business needs.

Comparison of Update Capabilities for Different Models/Implementations

Performance Comparison

	Unique Key MoW	Unique Key MoR	Aggregate Key
Import Speed	Deduplication is performed during import. Small-batch real-time writes incur approximately 10%-20% performance loss compared to MoR, while large-batch imports (e.g., tens or hundreds of millions of records) have about 30%-50% performance loss compared to MoR.	Similar to Duplicate Key	Similar to Duplicate Key

	Unique Key MoW	Unique Key MoR	Aggregate Key
Query Speed	Similar to Duplicate Key	Requires deduplication during queries, with query time approximately 3-10 times that of MoW	If the aggregation function is REPLACE/REPLACE_IF_N

	Unique Key MoW	Unique Key MoR	Aggregate Key
			OT_NULL, query speed is similar to MoR

	Unique Key MoW	Unique Key MoR	Aggregate Key
Predicate Pushdown	Supported	Not Supported	Not Supported
Resource Consumption	- **Import Resource Consumption**: Consumes approximately 10%-30% more CPU compared to Duplicate Key/Unique Key MoR. - **Query Resource Consumption**: Similar to Duplicate Key with no additional resource consumption. - **Compaction Resource Consumption**: Higher memory and CPU usage compared to Duplicate Key, specific usage depends on data characteristics and volume.	- **Import Resource Consumption**: Similar to Duplicate Key with no additional resource consumption. - **Query Resource Consumption**: Higher CPU and memory consumption during queries compared to Duplicate Key/Unique Key MoW. - **Compaction Resource Consumption**: Higher memory and CPU usage than Duplicate Key, specific usage depends on data characteristics and volume.	Same as Unique Key MoR

Feature Support Comparison

	Unique Key MoW	Unique Key MoR	Aggregate Key
UPDATE	Supported	Supported	Not Supported
DELETE	Supported	Supported	Not Supported

	Unique Key MoW	Unique Key MoR	Aggregate Key
sequence column	Supported	Supported	Not Supported
delete_sign	Supported	Supported	Not Supported
Partial Column Updates	Supported	Not Supported	Supported (can't update null value)
Inverted Index	Supported	Not Supported	Not Supported

Update in Primary Key (Unique) Model

Starting from Doris 2.0, Doris primary key (unique) model supports both Merge-on-Read (MoR) and Merge-on-Write (MoW) storage modes. MoR is optimized for write operations, while MoW is optimized for faster analysis performance. In actual tests, the analysis performance of MoW storage can be 5-10 times faster than MoR.

By default, in Doris 2.0, the unique key model is still based on MoR. To create a MoW model, you need to manually specify the parameter "enable_unique_key_merge_on_write" as "true". Here's an example:

```
CREATE TABLE IF NOT EXISTS example_tbl_unique_merge_on_write
(
  `user_id` LARGEINT NOT NULL,
  `username` VARCHAR(50) NOT NULL ,
  `city` VARCHAR(20),
  `age` SMALLINT,
  `sex` TINYINT,
  `phone` LARGEINT,
  `address` VARCHAR(500),
  `register_time` DATETIME
)
UNIQUE KEY(`user_id`, `username`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
"enable_unique_key_merge_on_write" = "true"
);
```

> **Starting from Doris 2.1, MoW is the default mode for the unique key model. So, if you are using Doris 2.1 or higher version, make sure to read the relevant table creation documentation.**

Two Update Methods in Unique Key Model

`UPDATE` statement

Whether it is MoR or MoW, the semantics ensure that the specified columns are updated. The time taken for a single UPDATE operation increases with the amount of data being updated.

Batch update based on load

Doris supports multiple data load methods, including Stream Load, Broker Load, Routine Load, and Insert Into. For primary key tables, all load operations default to "UPSERT" semantics: if a record with the same key does

not exist, it is inserted; if it already exists, it is updated. There are two types of updates: full row updates and partial column updates.

- **Full Row Update**: Updates for Unique Key tables default to full row updates. During data loading, users can choose to provide either all fields or only part of them. If only partial fields are provided, Doris will fill in the missing fields with default values to form a complete record for updating.
- **Partial Column Update**: Unique Key MoW supports partial column updates. Users can enable this feature by setting the session variable `enable_unique_key_partial_update = true` or by specifying `partial_columns:true` in the HTTP Header. When enabled, if a record with the given primary key exists, only the specified fields are updated; if the primary key does not exist, the missing fields are filled with default values.

We will provide detailed explanations of these two update methods in the documentation: Update in Unique Key Model and Load Update in Unique Key Model.

Concurrency Control for Primary Key Model Updates

Using `UPDATE` Statements to Update Data

By default, Doris does not allow multiple `UPDATE` operations on the same table at the same time. The `UPDATE` statement ensures isolation through table-level locking.

Users can adjust the concurrency limit by modifying the FE configuration `enable_concurrent_update=true`. When the concurrency limit is relaxed, if multiple `UPDATE` statements update the same row of data, the result will be undefined.

Batch Updates Based on Load

Doris provides atomicity for all load update operations—each data load will either be fully applied or fully rolled back.

For concurrent load updates, Doris determines the order of concurrent updates using an internal version control system (assigned based on the order of completed loading), using an MVCC mechanism.

Since the commit order of multiple concurrent load updates may be unpredictable, if these concurrent load jobs involve updates to the same primary key, the order in which they take effect is also uncertain. As a result, the final visible outcome may be indeterminate. To address this issue, Doris provides a `sequence` column mechanism, allowing users to specify a version for each row in concurrent load updates, thus ensuring determinism in the outcome of concurrent updates.

For more detailed information on concurrency control, refer to the documentation on Concurrency Control for Updates in the Primary Key Model.

Update in Aggregate Model

The update in the aggregate model refers to the process of generating new aggregate values by combining new column values with existing aggregate values, according to the requirements of the aggregate functions.

New Agg Value = Agg Func (Old Agg Value, New Column Value)

The update in the aggregate model is only supported through load methods and does not support the use of Update statements. When defining a table in the aggregate model, if the aggregation function for the value column is defined as REPLACE_IF_NOT_NULL, it indirectly achieves partial column update capabilities similar to the unique key model. For more details, please refer to the documentation on Load Update in the Aggregate Model.

Recommendations for Choosing Between Primary Key and Aggregate Models

- For most scenarios that require data updates, it is recommended to **prefer the primary key model**. Examples include synchronizing from TP databases to Doris via CDC, user profiling, and audience targeting.
- The following scenarios are recommended to use the aggregate model:
    1. Some fields need to be aggregated as metrics, while others need to be updated.
    2. Scenarios where partial column updates are needed, while being very sensitive to write performance and having low requirements for query latency, it is recommended to use the aggregate table with the REPLACE_IF_NOT_NULL aggregate function.

Updating Data with UPDATE Command

The main topic is how to use the UPDATE command to update data in Doris. The UPDATE command can only be executed on tables in the Unique data model.

Use Cases

- Modify the values of rows that meet certain conditions.
- Suitable for updating a small amount of data that is not frequently updated.

Basic Principles

By utilizing the filtering logic of the query engine's WHERE clause, the UPDATE command selects the rows that need to be updated from the target table. Then, using the built-in logic of the Value column in the Unique model, the old data is replaced with the new data, and the updated rows are reinserted into the table, thus achieving row-level updates.

Synchronization

The UPDATE syntax in Doris is synchronous, meaning that when an UPDATE statement is executed successfully, the update operation is completed, and the data is immediately visible.

Performance

The performance of the UPDATE statement depends on the number of rows to be updated and the efficiency of the condition retrieval.

- Number of rows to be updated: The more rows that need to be updated, the slower the UPDATE statement will be. The UPDATE command is suitable for scenarios where occasional updates are required, such as modifying values for individual rows. It is not suitable for bulk data modifications.
- Efficiency of condition retrieval: The UPDATE operation reads the rows that satisfy the condition first. Therefore, if the condition retrieval is efficient, the UPDATE operation will be faster. It is recommended to have the condition column indexed or utilize partitioning and bucketing pruning to quickly locate the rows to be updated, thus improving the update efficiency. It is strongly advised not to include the value column in the condition column.

Example

Assuming in a financial risk control scenario, there is a transaction details table with the following structure:

```
CREATE TABLE transaction_details (
  transaction_id BIGINT NOT NULL,        -- Unique transaction ID
  user_id BIGINT NOT NULL,               -- User ID
  transaction_date DATE NOT NULL,        -- Transaction date
  transaction_time DATETIME NOT NULL,    -- Transaction time
```

```
  transaction_amount DECIMAL(18, 2),     -- Transaction amount
  transaction_device STRING,             -- Transaction device
  transaction_region STRING,             -- Transaction region
  average_daily_amount DECIMAL(18, 2),   -- Average daily transaction amount over the last 3 months
  recent_transaction_count INT,          -- Number of transactions in the last 7 days
  has_dispute_history BOOLEAN,           -- Whether there is a dispute history
  risk_level STRING                      -- Risk level
)
UNIQUE KEY(transaction_id)
DISTRIBUTED BY HASH(transaction_id) BUCKETS 16
PROPERTIES (
  "replication_num" = "3",               -- Number of replicas, default is 3
  "enable_unique_key_merge_on_write" = "true"  -- Enable MOW mode, support merge update
);
```

The following transaction data exists:

```
+----------------+---------+------------------+---------------------+--------------------+--------------------+--------------------
+----------------------+--------------------------+---------------------+------------+
| transaction_id | user_id | transaction_date | transaction_time    | transaction_amount | transaction_device | transaction_region |
average_daily_amount | recent_transaction_count | has_dispute_history | risk_level |
+----------------+---------+------------------+---------------------+--------------------+--------------------+--------------------
+----------------------+--------------------------+---------------------+------------+
|           1001 |    5001 | 2024-11-24       | 2024-11-24 14:30:00 |             100.00 | iPhone 12          | New York           |
100.00 |                       10 |                   0 | NULL       |
|           1002 |    5002 | 2024-11-24       | 2024-11-24 03:30:00 |             120.00 | iPhone 12          | New York           |
100.00 |                       15 |                   0 | NULL       |
|           1003 |    5003 | 2024-11-24       | 2024-11-24 10:00:00 |             150.00 | Samsung S21        | Los Angeles        |
100.00 |                       30 |                   0 | NULL       |
|           1004 |    5004 | 2024-11-24       | 2024-11-24 16:00:00 |             300.00 | MacBook Pro        | high_risk_region1  |
200.00 |                        5 |                   0 | NULL       |
|           1005 |    5005 | 2024-11-24       | 2024-11-24 11:00:00 |            1100.00 | iPad Pro           | Chicago            |
200.00 |                       10 |                   0 | NULL       |
+----------------+---------+------------------+---------------------+--------------------+--------------------+--------------------
+----------------------+--------------------------+---------------------+------------+
```

Update the risk level of all transactions on a daily basis according to the following risk control rules: 1. If there is a dispute history, the risk is high. 2. If in a high-risk region, the risk is high. 3. If the transaction amount is abnormal (more than 5 times the daily average), the risk is high. 4. Frequent transactions in the last 7 days: a. If the number of transactions > 50, the risk is high. b. If the number of transactions is between 20 and 50, the risk is medium. 5. Transactions during non-working hours (2 AM to 4 AM), the risk is medium. 6. The default risk is low.

```
UPDATE transaction_details
SET risk_level = CASE
  -- Transactions with dispute history or in high-risk regions
  WHEN has_dispute_history = TRUE THEN 'high'
  WHEN transaction_region IN ('high_risk_region1', 'high_risk_region2') THEN 'high'

  -- Abnormal transaction amount
  WHEN transaction_amount > 5 * average_daily_amount THEN 'high'

  -- High transaction frequency in the last 7 days
  WHEN recent_transaction_count > 50 THEN 'high'
  WHEN recent_transaction_count BETWEEN 20 AND 50 THEN 'medium'

  -- Transactions during non-working hours
  WHEN HOUR(transaction_time) BETWEEN 2 AND 4 THEN 'medium'

  -- Default risk
```

```
  ELSE 'low'
END
WHERE transaction_date = '2024-11-24';
```

The updated data is:

```
+----------------+---------+------------------+---------------------+--------------------+--------------------+--------------------
+----------------------+--------------------------+--------------------+------------+
| transaction_id | user_id | transaction_date | transaction_time    | transaction_amount | transaction_device | transaction_region |
average_daily_amount | recent_transaction_count | has_dispute_history | risk_level |
+----------------+---------+------------------+---------------------+--------------------+--------------------+--------------------
+----------------------+--------------------------+--------------------+------------+
|           1001 |    5001 | 2024-11-24       | 2024-11-24 14:30:00 |             100.00 | iPhone 12          | New York           |
100.00 |                       10 |                   0 | low        |
|           1002 |    5002 | 2024-11-24       | 2024-11-24 03:30:00 |             120.00 | iPhone 12          | New York           |
100.00 |                       15 |                   0 | medium     |
|           1003 |    5003 | 2024-11-24       | 2024-11-24 10:00:00 |             150.00 | Samsung S21        | Los Angeles        |
100.00 |                       30 |                   0 | medium     |
|           1004 |    5004 | 2024-11-24       | 2024-11-24 16:00:00 |             300.00 | MacBook Pro        | high_risk_region1  |
200.00 |                        5 |                   0 | high       |
|           1005 |    5005 | 2024-11-24       | 2024-11-24 11:00:00 |            1100.00 | iPad Pro           | Chicago            |
200.00 |                       10 |                   0 | high       |
+----------------+---------+------------------+---------------------+--------------------+--------------------+--------------------
+----------------------+--------------------------+--------------------+------------+
```

More Details

For more detailed syntax on data updates, please refer to the UPDATE command manual. You can also enter `HELP UPDATE` in the MySQL client command line for more information and assistance.

Updating Data on Unique Key Model

This document primarily introduces the updates based on the load data on the Doris Unique Key model.

Updates on All Columns

When load data into the primary key model (Unique model) in Doris using supported load methods such as Stream Load, Broker Load, Routine Load, Insert Into, etc., if there are no corresponding data rows with the primary key, new data will be inserted. If there are corresponding data rows with the primary key, the data will be updated. In other words, load data into the Doris primary key model follows an “upsert” mode. Based on the import, updating existing records is by default the same as load a new record. Therefore, you can refer to the data load documentation section for more details.

Partial Update

Updating partial columns mainly refers to directly updating certain field values in a table instead of updating all field values. This can be done using the Update statement, which typically involves reading the entire row data, updating specific field values, and then writing it back. This read-write transaction is time-consuming and not suitable for writing large amounts of data. In the context of load updates on the primary key model, Doris provides a functionality to directly insert or update partial column data without reading the entire row data, significantly improving the update efficiency.

> **Note:**
>
> 1. Partial updates are only supported in the Merge-on-Write implementation of the Unique Key starting from version 2.0.
> 2. Starting from version 2.0.2, partial updates are supported using INSERT INTO.
> 3. Partial updates are not supported on tables with materialized views.
> 4. Partial updates are not allowed on tables doing schema change.

Use Cases

- Real-time dynamic column updates that require high-frequency updates on certain fields in the table. For example, in a user tag table, there are fields containing the latest user behavior information that needs real-time updates to enable real-time analysis and decision-making in advertising/recommendation systems.
- Combining multiple source tables into a large denormalized table.
- Data correction.

Usage

Table Creation

When creating the table, the following property needs to be specified to enable the Merge-on-Write implementation:

```
enable_unique_key_merge_on_write = true
```

StreamLoad/BrokerLoad/RoutineLoad

If you are using Stream Load/Broker Load/Routine Load, add the following header during the load:

```
partial_columns: true
```

Also, specify the columns to be loaded in the `columns` section (all key columns must be included, otherwise updates won't be possible).

Flink Connector

If you are using the Flink Connector, add the following configuration:

```
'sink.properties.partial_columns' = 'true',
```

Also, specify the columns to be loaded in `sink.properties.column` (all key columns must be included, otherwise updates won't be possible).

INSERT INTO

In all data models, when using `INSERT INTO` with a subset of columns, the default behavior is to insert the entire row. To enable partial column updates in the Merge-on-Write implementation, the following session variable needs to be set:

```
set enable_unique_key_partial_update=true
```

Note that the default value for the session variable `enable_insert_strict`, which controls whether the insert statement operates in strict mode, is true. In strict mode, updating non-existing keys during partial column updates is not allowed. So, if you want to insert non-existing keys during partial column updates using the insert statement, you need to set `enable_unique_key_partial_update` to true and also set `enable_insert_strict` to false.

Example

Suppose there is an order table named `order_tbl` in Doris, where the order ID is a key column, and the order status and order amount are value columns. The data is as follows:

Order ID	Order Amount	Order Status
1	100	Pending

```
+----------+--------------+--------------+
| order_id | order_amount | order_status |
+----------+--------------+--------------+
| 1        | 100          | Pending      |
```

```
+----------+--------------+--------------+
1 row in set (0.01 sec)
```

Now, when a user clicks on payment, the Doris system needs to update the order status of the order with ID '1' to 'To be shipped'.

If you are using Stream Load, you can update as follows:

```
$ cat update.csv

1,To be shipped

$ curl --location-trusted -u root: -H "partial_columns:true" -H "column_separator:," -H "columns:order_id,order_status" -T
/tmp/update.csv http://127.0.0.1:8030/api/db1/order_tbl/_stream_load
```

If you are using `INSERT INTO`, you can update as following methods:

```
set enable_unique_key_partial_update=true;
INSERT INTO order_tbl (order_id, order_status) values (1,'To be shipped');
```

The translated version in English:

After the update, the result is as follows:

```
+----------+--------------+--------------+
| order_id | order_amount | order_status |
+----------+--------------+--------------+
| 1        |          100 | To be shipped |
+----------+--------------+--------------+
1 row in set (0.01 sec)
```

Notes

Due to the Merge-on-Write implementation requiring data completion during data writing to ensure optimal query performance, performing partial column updates using the Merge-on-Write implementation may result in a decrease in load performance.

Suggestions for improving load performance:

- Use SSDs equipped with NVMe or high-speed SSD cloud disks. Reading historical data in large quantities during data completion will generate high read IOPS and read throughput.
- Enabling row storage can significantly reduce the IOPS generated during data completion, resulting in noticeable improvements in load performance. Users can enable row storage by using the following property when creating the table:

```
"store_row_column" = "true"
```

Now, all rows in a batch write task (whether it is an load task or `INSERT INTO`) can only update the same columns. If you need to update different columns, you will need to perform separate batch writes.

In the future, flexible column updates will be supported, allowing users to update different columns for each row within the same batch load.

Updating Data on Aggregate Key Model

Update for Aggregate Load

This guide is about ingestion-based data updates for the Aggregate Key model in Doris.

Update all columns

When importing data into an Aggregate Key model in Doris by methods like Stream Load, Broker Load, Routine Load, and Insert Into, the new values are combined with the old values to produce new aggregated values based on the column's aggregation function. These values might be generated during insertion or produced asynchronously during compaction. However, when querying, users will always receive the same returned values.

Partial column update for Aggregate Key model

Tables in the Aggregate Key model are primarily used in cases with pre-aggregation requirements rather than data updates, but Doris allows partial column updates for them, too. Simply set the aggregation function to `REPLACE_IF_NOT_NULL`.

Create table

For the columns that need to be updated, set the aggregation function to `REPLACE_IF_NOT_NULL`.

```
CREATE TABLE order_tbl (
  order_id int(11) NULL,
  order_amount int(11) REPLACE_IF_NOT_NULL NULL,
  order_status varchar(100) REPLACE_IF_NOT_NULL NULL
) ENGINE=OLAP
AGGREGATE KEY(order_id)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(order_id) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
+----------+--------------+-----------------+
| order_id | order_amount | order_status    |
+----------+--------------+-----------------+
| 1        |          100 | Pending Payment |
+----------+--------------+-----------------+
1 row in set (0.01 sec)
```

Ingest data

For Stream Load, Broker Load, Routine Load, or INSERT INTO, you can directly write the updates to the fields.

Example

Using the same example as above, the corresponding Stream Load command would be (no additional headers required):

```
$ cat update.csv

1,To be shipped

$ curl  --location-trusted -u root: -H "column_separator:," -H "columns:order_id,order_status" -T /tmp/update.csv
http://127.0.0.1:8030/api/db1/order_tbl/_stream_load
```

The corresponding `INSERT INTO` statement would be (no additional session variables required):

```
INSERT INTO order_tbl (order_id, order_status) values (1,'Delivery Pending');
```

Note

The Aggregate Key model does not perform additional data processing during data writing, so the writing performance in this model is the same as other models. However, aggregation during queries can result in performance loss. Typical aggregation queries can be 5~10 times slower than queries on Merge-on-Write tables in the Unique Key model.

Under this circumstance, users cannot set a field from non-NULL to NULL, because NULL values written will be automatically neglected by the REPLACE_IF_NOT_NULL aggregation function.

Concurrency Control for Updates in the Primary Key Model

Overview

Doris adopts a Multi-Version Concurrency Control (MVCC) mechanism to manage concurrent updates. Each data load operation is assigned a transaction, which ensures atomicity (i.e., the operation either fully succeeds or completely fails). Upon transaction commit, the system assigns a version number. When using the Unique Key model and loading multiple batches of data with duplicate primary keys, Doris determines the overwrite order based on the version number: data with a higher version number will overwrite data with a lower version number.

In certain scenarios, users may need to specify a sequence column in the table creation statement to customize the order in which data takes effect. For example, when synchronizing data into Doris using multiple concurrent processes, the data may arrive out of order. This could lead to older data overwriting newer data due to its delayed arrival. To address this, users can assign a lower sequence value to the older data and a higher sequence value to the newer data, enabling Doris to determine the update order based on the sequence values provided by the user.

Additionally, `UPDATE` statements differ significantly from updates performed via data loads at the implementation level. An `UPDATE` operation involves two steps: reading the data to be updated from the database and writing the updated data back. By default, `UPDATE` statements use table-level locks to provide transaction capabilities with Serializable isolation, meaning multiple `UPDATE` statements must be executed serially. However, users can bypass this restriction by modifying the configuration. For detailed instructions, refer to the relevant section below.

Update Concurrency Control

By default, concurrent `UPDATE`s on the same table are not allowed in Doris.

The main reason is that Doris currently supports row-level updates, which means that even if the user specifies to update only a specific column (e.g., `SET v2 = 1`), all other value columns will be overwritten as well (even though their values remain unchanged).

This poses a problem when multiple `UPDATE` operations are performed concurrently on the same row. The behavior becomes unpredictable, and it may lead to inconsistent or “dirty” data.

However, in practical applications, if the user can ensure that concurrent updates will not affect the same row simultaneously, they can manually enable the concurrent update restriction. This can be done by modifying the FE (Frontend) configuration `enable_concurrent_update`. When this configuration is set to `true`, the update command will not have transaction guarantees.

> **Caution:**

> Enabling the `enable_concurrent_update` configuration may introduce certain performance risks.

Sequence Column

The Unique model primarily caters to scenarios that require unique primary keys, ensuring the uniqueness constraint. When loading data in the same batch or different batches, the replacement order is not guaranteed. The uncertainty in the replacement order results in ambiguity in the specific data loaded into the table.

To address this issue, Doris supports sequence columns. Users can specify a sequence column during data load, allowing the replacement order to be controlled by the user. The sequence column determines the order of replacements for rows with the same key column. A higher sequence value can replace a lower one, but not vice versa. This method delegates the determination of order to the user, enabling control over the replacement sequence.

> **Sequence columns are currently supported only in the Unique model.**

Basic Principles

The basic principle is achieved by adding a hidden column called **DORIS_SEQUENCE_COL**. The type of this column is specified by the user during table creation and its specific value is determined during data load. Based on this value, the row that takes effect is determined for rows with the same key column.

Table Creation

When creating a Unique table, an automatically added hidden column called **DORIS_SEQUENCE_COL** is created, based on the user-specified type.

Data load

During data load, the FE (Frontend) sets the value of the hidden column as the value of the `ORDER BY` expression (for broker load and routine load) or the value of the `function_column.sequence_col` expression (for stream load). The value column is replaced based on this sequence value. The value of the hidden column, `__DORIS_SEQUENCE_COL__`, can be set as a column in the data source or a column in the table structure.

Syntax Usage

Sequence Column has two ways to create a table, one is to set the `sequence_col` attribute when creating a table, and the other is to set the `sequence_type` attribute when creating a table.

1. Set `sequence_col` (Recommended)

When creating a Unique table, specify the mapping of the sequence column to other columns in the table.

```
PROPERTIES (
    "function_column.sequence_col" = 'column_name',
);
```

`sequence_col` is used to specify the mapping of the sequence column to a column in the table. The column can be of type integer or time type (DATE, DATETIME), and its type cannot be changed after creation.

The load method is the same as when there is no sequence column, making it relatively simple. This method is recommended.

2. Set `sequence_type`

When creating a Unique table, specify the type of the sequence column.

```
PROPERTIES (
    "function_column.sequence_type" = 'Date',
);
```

`sequence_type` is used to specify the type of the sequence column, which can be integer or time type (DATE, DATETIME).

During data load, you need to specify the mapping of the sequence column to other columns.

1. Stream Load

The syntax for stream load is to add the mapping of the hidden column `function_column.sequence_col` to the `source_sequence` in the header. Example:

```
curl --location-trusted -u root -H "columns: k1,k2,source_sequence,v1,v2" -H "function_column.sequence_col: source_sequence" -T
testData http://host:port/api/testDb/testTbl/_stream_load
```

2. Broker Load

Set the mapping of the hidden column `source_sequence` in the `ORDER BY` clause.

```
LOAD LABEL db1.label1
(
    DATA INFILE("hdfs://host:port/user/data/*/test.txt")
    INTO TABLE `tbl1`
    COLUMNS TERMINATED BY ","
    (k1,k2,source_sequence,v1,v2)
    ORDER BY source_sequence
)
WITH BROKER 'broker'
(
    "username"="user",
    "password"="pass"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

3. Routine Load

The mapping method is the same as above. Example:

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
    [WITH MERGE|APPEND|DELETE]
```

```
    COLUMNS(k1, k2, source_sequence, v1, v2),
    WHERE k1  100 and k2 like "%doris%"
    [ORDER BY source_sequence]
    PROPERTIES
    (
        "desired_concurrent_number"="3",
        "max_batch_interval" = "20",
        "max_batch_rows" = "300000",
        "max_batch_size" = "209715200",
        "strict_mode" = "false"
    )
    FROM KAFKA
    (
        "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
        "kafka_topic" = "my_topic",
        "kafka_partitions" = "0,1,2,3",
        "kafka_offsets" = "101,0,0,200"
    );
```

Enabling Sequence Column Support

If `function_column.sequence_col` or `function_column.sequence_type` is set when creating a new table, the new table will support sequence columns.

For a table that does not support sequence columns, if you want to use this feature, you can use the following statement: `ALTER TABLE example_db.my_table ENABLE FEATURE "SEQUENCE_LOAD" WITH PROPERTIES ("function_column.sequence_type" = "Date")` to enable it.

If you are unsure whether a table supports sequence columns, you can set a session variable to display hidden columns with `SET show_hidden_columns=true`, and then use `desc tablename`. If the output includes the `__DORIS_SEQUENCE_COL__` column, it means that the table supports sequence columns; otherwise, it does not.

Usage Example

Here is an example using Stream Load to demonstrate the usage:

1. Create a table with sequence col support

Create a unique model `test_table` and specify the sequence column mapping to the `modify_date` column in the table.

```
CREATE TABLE test.test_table
(
    user_id bigint,
    date date,
    group_id bigint,
    modify_date date,
    keyword VARCHAR(128)
)
UNIQUE KEY(user_id, date, group_id)
DISTRIBUTED BY HASH (user_id) BUCKETS 32
PROPERTIES(
    "function_column.sequence_col" = 'modify_date',
    "replication_num" = "1",
    "in_memory" = "false"
);
```

Table structure:

```
MySQL> desc test_table;
+-------------+--------------+------+-------+---------+---------+
| Field       | Type         | Null | Key   | Default | Extra   |
+-------------+--------------+------+-------+---------+---------+
| user_id     | BIGINT       | No   | true  | NULL    |         |
| date        | DATE         | No   | true  | NULL    |         |
| group_id    | BIGINT       | No   | true  | NULL    |         |
| modify_date | DATE         | No   | false | NULL    | REPLACE |
| keyword     | VARCHAR(128) | No   | false | NULL    | REPLACE |
+-------------+--------------+------+-------+---------+---------+
```

2. Load data normally:

Load the following data:

```
1   2020-02-22  1   2020-02-21  a
1   2020-02-22  1   2020-02-22  b
1   2020-02-22  1   2020-03-05  c
1   2020-02-22  1   2020-02-26  d
1   2020-02-22  1   2020-02-23  e
1   2020-02-22  1   2020-02-24  b
```

Here is an example using Stream Load:

```
curl --location-trusted -u root: -T testData http://host:port/api/test/test_table/_stream_load
```

The result is:

```
MySQL> select * from test_table;
+---------+------------+----------+-------------+---------+
| user_id | date       | group_id | modify_date | keyword |
+---------+------------+----------+-------------+---------+
|       1 | 2020-02-22 |        1 | 2020-03-05  | c       |
+---------+------------+----------+-------------+---------+
```

In the data load, because the value of the sequence column (i.e., modify_date) ‘2020-03-05’ is the maximum, the keyword column retains the value ‘c’.

3. Guarantee the order of replacement

After completing the above steps, load the following data:

```
1   2020-02-22  1   2020-02-22  a
1   2020-02-22  1   2020-02-23  b
```

Query the data:

```
MySQL [test]> select * from test_table;
+---------+------------+----------+-------------+---------+
| user_id | date       | group_id | modify_date | keyword |
+---------+------------+----------+-------------+---------+
```

```
|       1 | 2020-02-22 |        1 | 2020-03-05  | c       |
+---------+------------+----------+-------------+---------+
```

In the loaded data, the sequence column (modify_date) of all previously loaded data is compared, and '2020-03-05' is the maximum. Therefore, the keyword column retains the value 'c'.

4. Try loading the following data again

```
1   2020-02-22  1   2020-02-22  a
1   2020-02-22  1   2020-03-23  w
```

Query the data:

```
MySQL [test]> select * from test_table;
+---------+------------+----------+-------------+---------+
| user_id | date       | group_id | modify_date | keyword |
+---------+------------+----------+-------------+---------+
|       1 | 2020-02-22 |        1 | 2020-03-23  | w       |
+---------+------------+----------+-------------+---------+
```

Now the original data in the table can be replaced. In summary, during thestream load process, the sequence column is used to determine the order of replacement for duplicate records. The record with the maximum value in the sequence column will be retained in the table.

Note

1. To prevent misuse, users must explicitly specify the sequence column in loading tasks such as StreamLoad/BrokerLoad and in insert statements for row updates (unless the default value of the sequence column is CURRENT_TIMESTAMP). Otherwise, the following error message will be received:

```
Table test_tbl has sequence column, need to specify the sequence column
```

2. Since version 2.0, Doris has supported partial column updates for Merge-on-Write implementation of Unique Key tables. In partial column update, users can update only a subset of columns each time, so it is not necessary to include the sequence column. If the loading task submitted by the user includes the sequence column, it has no effect. If the loading task submitted by the user does not include the sequence column, Doris will use the value of the matching sequence column from the historical data as the value of the updated row's sequence column. If there is no existing column with the same key in the historical data, it will be automatically filled with null or the default value.

3. In cases of concurrent data load, Doris utilizes MVCC (Multi-Version Concurrency Control) mechanism to ensure data correctness. If two batches of loaded data update different columns of the same key, the load task with a higher system version will reapply the data for the same key written by the load task with a lower version after the lower version load task succeeds.

Deleting Data with DELETE Command

The DELETE statement conditionally deletes data from a specified table or partition using the MySQL protocol. The DELETE operation differs from import-based bulk deletion in that it is similar to the INSERT INTO statement, which is a synchronous process. All DELETE operations are separate import jobs in Doris.

The DELETE statement generally requires the specification of tables and partitions as well as deletion conditions to filter the data to be deleted, and will delete data from both the base and rollup tables.

The syntax of the DELETE statement is detailed in the DELETE syntax. Unlike the INSERT INTO command, DELETE cannot specify `label` manually. For the concept of `label`, refer to the Insert Into documentation.

Delete by Specifying a Filter Predicate

```
DELETE FROM table_name [table_alias]
  [PARTITION partition_name | PARTITIONS (partition_name [, partition_name])]
  WHERE column_name op { value | value_list } [ AND column_name op { value | value_list } ...];
```

Required Parameters

- table_name: Specify the table from which the data should be deleted.
- column_name: Columns belonging to table_name.
- op: Logical comparison operators, optional types include: =, >, <, >=, <=, !=, in, not in.
- value | value_list: Values or lists of values for logical comparisons.

Optional Parameters

- PARTITION partition_name | PARTITIONS (partition_name [, partition_name]): Specify the name of the partition in which the deletion is to be performed. If the partition does not exist in the table, an error will be reported.
- table_alias: Aliases of the table.

Note

- When using the table model Aggregate, you can only specify conditions on the key column.
- If the selected key column does not exist in a rollup, it cannot be deleted.
- Conditions can only be related to each other by “and”. If you want an “or” relationship, you need to write the conditions in two separate DELETE statements.
- If the table is partitioned, you need to specify the partition. If not, Doris will infer the partition from the condition. In two cases, Doris cannot infer the partition from the condition:

    - The condition does not contain a partition column.
    - The op for the partition column is “not in”. When the partition table does not specify a partition, or a partition cannot be inferred from the condition, you need to set the session variable `delete_without_partition` to true, in which case delete is applied to all partitions.
- This statement may reduce query efficiency for a period of time after execution. The extent of the impact depends on the number of deleted conditions specified in the statement. The more conditions specified, the greater the impact.

Examples

1. Delete the row in my_table partition p1 where column k1 is 3.

```
DELETE FROM my_table PARTITION p1
  WHERE k1 = 3;
```

2. Delete rows in my_table partition p1 where column k1 is greater than or equal to 3 and column status is “outdated”.

```
DELETE FROM my_table PARTITION p1
WHERE k1 >= 3 AND status = "outdated";
```

3. Delete rows in my_table partitions (p1, p2) where column k1 is greater than or equal to 3 and column dt is between ‘2024-10-01’ and ‘2024-10-31’.

```
DELETE FROM my_table PARTITIONS (p1, p2)
WHERE k1 >= 3 AND dt >= "2024-10-01" AND dt <= "2024-10-31";
```

Delete via the USING clause

```
DELETE FROM table_name [table_alias]
  [PARTITION partition_name | PARTITIONS (partition_name [, partition_name])]
  [USING additional_tables]
  WHERE condition;
```

Required Parameters

- table_name: Specify the table from which the data should be deleted.
- WHERE condition: Specify a condition for selecting rows for deletion.

Optional Parameters

- PARTITION partition_name | PARTITIONS (partition_name [, partition_name]): Specify the name of the partition in which the deletion is to be performed. If the partition does not exist in the table, an error will be reported.
- table_alias: Aliases of the table.

Note

- Only conditions on the key column can be specified when using the UNIQUE model.

Example

Use the result of joining the `t2` and `t3` tables to delete the data in `t1`. The deleted table only supports the UNIQUE model.

```
-- Create t1, t2, t3 tables
CREATE TABLE t1
  (id INT, c1 BIGINT, c2 STRING, c3 DOUBLE, c4 DATE)
UNIQUE KEY (id)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1', "function_column.sequence_col" = "c4");

CREATE TABLE t2
  (id INT, c1 BIGINT, c2 STRING, c3 DOUBLE, c4 DATE)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1');

CREATE TABLE t3
  (id INT)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1');

-- Insert data
INSERT INTO t1 VALUES
  (1, 1, '1', 1.0, '2000-01-01'),
  (2, 2, '2', 2.0, '2000-01-02'),
  (3, 3, '3', 3.0, '2000-01-03');

INSERT INTO t2 VALUES
  (1, 10, '10', 10.0, '2000-01-10'),
  (2, 20, '20', 20.0, '2000-01-20'),
  (3, 30, '30', 30.0, '2000-01-30'),
  (4, 4, '4', 4.0, '2000-01-04'),
  (5, 5, '5', 5.0, '2000-01-05');

INSERT INTO t3 VALUES
  (1),
  (4),
  (5);

-- Remove rows from t1
DELETE FROM t1
  USING t2 INNER JOIN t3 ON t2.id = t3.id
  WHERE t1.id = t2.id;
```

The expected result is that the row with `id=1` in table `t1` is deleted.

```
+----+----+----+--------+------------+
| id | c1 | c2 | c3     | c4         |
+----+----+----+--------+------------+
| 2  | 2  | 2  |    2.0 | 2000-01-02 |
| 3  | 3  | 3  |    3.0 | 2000-01-03 |
+----+----+----+--------+------------+
```

FE Configurations

TIMEOUT Configurations

Overall, the timeout calculation rules for Doris DELETE jobs are as follows (in seconds):

```
TIMEOUT = MIN(delete_job_max_timeout_second, MAX(30, tablet_delete_timeout_second * tablet_num))
```

- `tablet_delete_timeout_second`

The delete timeout time is elastically changed by the number of tablets under the specified partition. This item is configured so that the default value of the timeout time contributed by one tablet on average is 2.

Assuming that there are 5 tablets under the partition specified for this deletion, the timeout time available for delete is 10 seconds, and since it is less than the minimum timeout time of 30 seconds, the final timeout time is 30 seconds.

- `query_timeout`

Because delete itself is a SQL command, the DELETE statement is also subject to session limitations. Timeout is also affected by the `query_timeout` value in the session, which can be increased in seconds by `SET query_timeout = xxx`.

IN Predicate Configuration

- `max_allowed_in_element_num_of_delete`

If the user needs to occupy more elements when using the IN predicate, the user can adjust the maximum number of elements allowed to be carried by `max_allowed_in_element_num_of_delete`. The default value is 1024.

View History

Users can view the history of deletions that have been performed by using the SHOW DELETE statement.

Syntax

```
SHOW DELETE [FROM db_name];
```

Example

```
mysql> show delete from test_db;
+-----------+---------------+---------------------+-----------------+----------+
| TableName | PartitionName | CreateTime          | DeleteCondition | State    |
+-----------+---------------+---------------------+-----------------+----------+
| empty_tbl | p3            | 2020-04-15 23:09:35 | k1 EQ "1"       | FINISHED |
| test_tbl  | p4            | 2020-04-15 23:09:53 | k1 GT "80"      | FINISHED |
+-----------+---------------+---------------------+-----------------+----------+
2 rows in set (0.00 sec)
```

Batch Deletion

Why do we need to introduce import-based Batch Delete when we have the Delete operation?

- **Limitations of Delete operation**

When you delete by Delete statement, each execution of Delete generates an empty rowset to record the deletion conditions and a new version of the data. Each time you read, you have to filter the deletion conditions. If you delete too often or have too many deletion conditions, it will seriously affect the query performance.

- **Insert data interspersed with Delete data**

For scenarios like importing data from a transactional database via CDC, Insert and Delete are usually interspersed in the data. In this case, the current Delete operation cannot be implemented.

When importing data, there are several ways to merge it:

1. APPEND: Append all data to existing data.
2. DELETE: Delete all rows that have the same value as the key column of the imported data (when a `sequence` column exists in the table, it is necessary to satisfy the logic of having the same primary key as well as the size of the sequence column in order to delete it correctly, see Use Case 4 below for details).
3. MERGE: APPEND or DELETE according to DELETE ON decision

> **Warning**
>
> Batch Delete only works on Unique models.

Fundamental

This is achieved by adding a hidden column `__DORIS_DELETE_SIGN__` to the Unique table.

When FE parses the query, `__DORIS_DELETE_SIGN__` is removed when it encounters * and so on, and `__DORIS_DELETE_SIGN__ != true`, BE will add a column for judgement when reading, and determine whether to delete by the condition.

- Import

  On import, the value of the hidden column is set to the value of the `DELETE ON` expression during the FE parsing stage.

- Read

  The read adds `__DORIS_DELETE_SIGN__ != true` condition, BE does not sense this process and executes normally.

- Cumulative Compaction

In Cumulative Compaction, hidden columns are treated as normal columns and the Compaction logic remains unchanged.

- Base Compaction

  When Base Compaction is performed, the rows marked for deletion are deleted to reduce the space occupied by the data.

Enable Batch Delete Support

There are two forms of enabling Batch Delete support:

1. Batch Delete is supported by adding `enable_batch_delete_by_default=true` in the FE configuration file for all new tables created after restarting FE;
2. For tables that do not have the above FE configuration changed or for existing tables that do not support Batch Delete, the following statement can be used: `ALTER TABLE tablename ENABLE FEATURE "BATCH_DELETE"` to enable Batch Delete. This is essentially a schema change operation, which returns immediately and can be confirmed by `showing alter table column`.

Then how to determine whether a table supports Batch Delete, you can set a session variable to show hidden columns `SET show_hidden_columns=true`, and after that use `desc tablename`, if there is a `__DORIS_DELETE_SIGN__` column in the output then it is supported, if there is not then it is not supported.

Syntax Description

The syntax design of the import is mainly to add a column mapping that specifies the field of the delete marker column, and it is necessary to add a column to the imported data. The syntax of various import methods is as follows:

Stream Load

The writing method of `Stream Load` adds a field to set the delete label column in the columns field in the header. Example: `-H "columns: k1, k2, label_c3" -H "merge_type: [MERGE|APPEND|DELETE]" -H "delete: label_c3=1"`

Broker Load

The writing method of `Broker Load` sets the field of the delete marker column at `PROPERTIES`. The syntax is as follows:

```
LOAD LABEL db1.label1
(
    [MERGE|APPEND|DELETE] DATA INFILE("hdfs://abc.com:8888/user/palo/test/ml/file1")
    INTO TABLE tbl1
    COLUMNS TERMINATED BY ","
    (tmp_c1,tmp_c2, label_c3)
    SET
    (
        id=tmp_c2,
        name=tmp_c1,
    )
    [DELETE ON label_c3=true]
```

```
)
WITH BROKER 'broker'
(
    "username"="user",
    "password"="pass"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

Routine Load

The writing method of `Routine Load` adds a mapping to the `columns` field. The mapping method is the same as above. The syntax is as follows:

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
 [WITH MERGE|APPEND|DELETE]
 COLUMNS(k1, k2, k3, v1, v2, label),
 WHERE k1  100 and k2 like "%doris%"
 [DELETE ON label=true]
 PROPERTIES
 (
     "desired_concurrent_number"="3",
     "max_batch_interval" = "20",
     "max_batch_rows" = "300000",
     "max_batch_size" = "209715200",
     "strict_mode" = "false"
 )
 FROM KAFKA
 (
     "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
     "kafka_topic" = "my_topic",
     "kafka_partitions" = "0,1,2,3",
     "kafka_offsets" = "101,0,0,200"
 );
```

Note

1. Since import operations other than stream load may be executed out of order inside doris, if it is not stream load when importing using the `MERGE` method, it needs to be used with load sequence. For the specific syntax, please refer to the `sequence` column related documents

2. `DELETE ON` condition can only be used with MERGE.

> **Tip**
>
> if session variable `SET show_hidden_columns = true` was executed before running import task to show whether table support batch delete feature, then execute `select count(*) from xxx` statement in the same session after finishing `DELETE/MERGE` import task, it will result in a unexpected result that the statement result set will include the deleted results. To avoid this problem, you should execute `SET show_hidden_columns = false` before selecting statement or open a new session to run the select statement.

Usage Examples

Check if Batch Delete Support is Enabled

```
mysql> CREATE TABLE IF NOT EXISTS table1 (
    ->     siteid INT,
    ->     citycode INT,
    ->     username VARCHAR(64),
    ->     pv BIGINT
    -> ) UNIQUE KEY (siteid, citycode, username)
    -> DISTRIBUTED BY HASH(siteid) BUCKETS 10
    -> PROPERTIES (
    ->     "replication_num" = "3"
    -> );
Query OK, 0 rows affected (0.34 sec)

mysql> SET show_hidden_columns=true;
Query OK, 0 rows affected (0.00 sec)

mysql> DESC table1;
+-----------------------+-------------+------+-------+---------+-------+
| Field                 | Type        | Null | Key   | Default | Extra |
+-----------------------+-------------+------+-------+---------+-------+
| siteid                | int         | Yes  | true  | NULL    |       |
| citycode              | int         | Yes  | true  | NULL    |       |
| username              | varchar(64) | Yes  | true  | NULL    |       |
| pv                    | bigint      | Yes  | false | NULL    | NONE  |
| __DORIS_DELETE_SIGN__ | tinyint     | No   | false | 0       | NONE  |
| __DORIS_VERSION_COL__ | bigint      | No   | false | 0       | NONE  |
+-----------------------+-------------+------+-------+---------+-------+
6 rows in set (0.01 sec)
```

Stream Load Usage Examples

Please refer to the sections “Specifying merge_type for DELETE operations” and “Specifying merge_type for MERGE operations” in the Stream Load Manual

Deleting Data with TRUNCATE Command

Truncate

This statement is used to clear the data of a specified table or partition in Doris

Syntax

```
TRUNCATE TABLE [db.]tbl[ PARTITION(p1, p2, ...)];
```

- This statement only clears the data within a table or partition but preserves the table or partition itself.
- Unlike DELETE, this statement can only clear the specified table or partition as a whole and cannot be added with filter conditions.
- Unlike DELETE, truncating data will not affect query performance.
- The data deleted by this operation can be recovered through the RECOVER statement(for a period of time). See RECOVER statement for details. If you execute command with FORCE, the data will be deleted directly and cannot be recovered, this operation is generally not recommended.
- When using this command, the table status must be NORMAL, which means that tables undergoing SCHEMA CHANGE can not be truncated.
- This command may cause ongoing imports to fail.

Example

1. Truncate the table `tbl` under `example_db`.

```
TRUNCATE TABLE example_db.tbl;
```

2. Truncate partitions `p1` and `p2` of table `tbl`.

```
TRUNCATE TABLE tbl PARTITION(p1, p2);
```

Replacing Atomic Table

Atomicity Replace

Doris supports atomic table replacement operations for two tables. This is only applicable to OLAP tables.

Applicable scenarios

- Atomic overwrite operations
- In certain cases, users may want to rewrite data in a table. However, the “delete and load” approach causes a data invisibility window. To solve that, Doris allows users to create a new table of the same schema using the CREATE TABLE LIKE statement, import the new data into this new table, and then atomically replace the old table with the new table. For atomic replacement at the partition level, please refer to the temporary partitiondocumentation.

Syntax

```
ALTER TABLE [db.]tbl1 REPLACE WITH TABLE tbl2
[PROPERTIES('swap' = 'true')];
```

Replace table tbl1 with table tbl2.

If `swap` is `true`, after the replacement, data in `tbl1` will be replaced by that in `tbl2`, while data in `tbl2` will be replaced by that in `tbl1`. In other words, the two tables will swap data.

If `swap` is `false`, after the replacement, data in `tbl1` will be replaced by that in `tbl2` and `tbl2` will be deleted.

Implementation

In fact, table replacement is to combine the following operations into one atomic operation.

Assuming that table A is to be replaced with table B, and `swap` is set to `true`. The operations to be implemented are as follows:

1. Rename table B to table A.
2. Rename table A to table B.

If `swap` is set to `false`, the operations are as follows:

1. Delete table A.
2. Rename table B to table A.

Note

- `swap` defaults to `true`, meaning to swap the data between two tables.
- If `swap` is set to `false`, the table being replaced (table A) will be deleted and cannot be recovered.

- The replacement operation can only be implemented between two OLAP tables and it does not check for table schema consistency.
- The replacement operation does not change the existing privilege settings because privilege checks are based on table names.

Temporary Partition

Doris supports adding temporary partitions to partitioned tables. Temporary partitions differ from formal partitions in that they are not queried by formal queries, but only by special query statements.

- The partition columns of the temporary partition are the same as the formal partition and cannot be modified.
- The partition ranges of all temporary partitions of a table cannot overlap, but the ranges of temporary partitions and formal partitions can overlap.
- The partition name of the temporary partition cannot be the same as the formal partitions and other temporary partitions.

Temporary Partitions are mainly used in the following scenarios:

- Atomic overwrite write operations

In some cases, users want to rewrite the data of a certain partition, but if they do it by deleting and then importing, there will be a period of time in the middle when they cannot view the data. In this case, the user can create a corresponding Temporary Partition first, and after importing the new data into the Temporary Partition, replace the original partition atomically through the Replace operation to achieve the purpose. For the atomic overwrite write operation of non-partitioned table, please refer to the documentation of Atomicity Replace.

- Modifying the number of buckets

In some cases, users use an inappropriate bucket number when creating partitions. Then the user can first create a Temporary Partition corresponding to the partition range and specify the new bucket number. Then import the data of the formal partition into the Temporary Partition through the `INSERT INTO` command, and replace the original partition atomically through the Replace operation to achieve the purpose.

- Merge or split partition

In some cases, users want to modify the scope of partitions, such as merging two partitions, or splitting a large partition into several small partitions. You can first create a Temporary Partition corresponding to the scope of the merged or partitioned partition, and then import the data of the official partition into the Temporary Partition through the `INSERT INTO` command, and then replace the original partition atomically through the Replacement operation, in order to achieve the purpose.

Add Temporary Partition

You can add temporary partitions to a table with the `ALTER TABLE ADD TEMPORARY PARTITION` statement:

```
ALTER TABLE tbl1 ADD TEMPORARY PARTITION tp1 VALUES LESS THAN("2020-02-01");

ALTER TABLE tbl2 ADD TEMPORARY PARTITION tp1 VALUES [("2020-01-01"), ("2020-02-01"));

ALTER TABLE tbl1 ADD TEMPORARY PARTITION tp1 VALUES LESS THAN("2020-02-01")
("replication_num" = "1")
DISTRIBUTED BY HASH(k1) BUCKETS 5;
```

```
ALTER TABLE tbl3 ADD TEMPORARY PARTITION tp1 VALUES IN ("Beijing", "Shanghai");

ALTER TABLE tbl4 ADD TEMPORARY PARTITION tp1 VALUES IN ((1, "Beijing"), (1, "Shanghai"));

ALTER TABLE tbl3 ADD TEMPORARY PARTITION tp1 VALUES IN ("Beijing", "Shanghai")
("replication_num" = "1")
DISTRIBUTED BY HASH(k1) BUCKETS 5;
```

See `HELP ALTER TABLE;` for more help and examples.

Some instructions for adding operations:

- Adding a temporary partition is similar to adding a formal partition. The partition range of the temporary partition is independent of the formal partition.
- Temporary partition can independently specify some attributes. Includes information such as the number of buckets, the number of replicas, or the storage medium.

Delete Temporary Partition

A table's temporary partition can be dropped with the `ALTER TABLE DROP TEMPORARY PARTITION` statement:

```
ALTER TABLE tbl1 DROP TEMPORARY PARTITION tp1;
```

See `HELP ALTER TABLE;` for more help and examples.

Some instructions for the DELETE:

- Deleting the temporary partition will not affect the data of the formal partition.

Replace Partition

You can replace formal partitions of a table with temporary partitions with the `ALTER TABLE REPLACE PARTITION` statement.

```
ALTER TABLE tbl1 REPLACE PARTITION (p1) WITH TEMPORARY PARTITION (tp1);

ALTER TABLE tbl1 REPLACE PARTITION (p1, p2) WITH TEMPORARY PARTITION (tp1, tp2, tp3);

ALTER TABLE tbl1 REPLACE PARTITION (p1, p2) WITH TEMPORARY PARTITION (tp1, tp2)
PROPERTIES (
    "strict_range" = "false",
    "use_temp_partition_name" = "true"
);
```

See `HELP ALTER TABLE;` for more help and examples.

The replace operation has two special optional parameters:

1. `strict_range`

The default is true.

For Range partition, When this parameter is true, the range union of all formal partitions to be replaced needs to be the same as the range union of the temporary partitions to be replaced. When set to false, you only need to ensure that the range between the new formal partitions does not overlap after replacement.

For List partition, this parameter is always true, and the enumeration values of all full partitions to be replaced must be identical to the enumeration values of the temporary partitions to be replaced.

Example 1

```
-- Range of partitions p1, p2, p3 to be replaced (=> union):
(10, 20), [20, 30), [40, 50) => [10, 30), [40, 50)

--Replace the range of partitions tp1, tp2 (=> union):
(10, 30), [40, 45), [45, 50) => [10, 30), [40, 50)

--The union of ranges is the same, so you can use tp1 and tp2 to replace p1, p2, p3.
```

Example 2

```
--Range of partition p1 to be replaced (=> union):
[10, 50) => [10, 50)

--Replace the range of partitions tp1, tp2 (=> union):
[10, 30), [40, 50) => [10, 30), [40, 50)

--The union of ranges is not the same. If strict_range is true, you cannot use tp1 and tp2 to replace p1. If false, and the two
partition ranges [10, 30), [40, 50) and the other formal partitions do not overlap, they can be replaced.
```

Example 3

```
--Enumerated values of partitions p1, p2 to be replaced (=> union).
(1, 2, 3), (4, 5, 6) => (1, 2, 3, 4, 5, 6)
--Replace the enumerated values of partitions tp1, tp2, tp3 (=> union).
(1, 2, 3), (4), (5, 6) => (1, 2, 3, 4, 5, 6)
--The enumeration values are the same, you can use tp1, tp2, tp3 to replace p1, p2
```

Example 4

```
--Enumerated values of partitions p1, p2, p3 to be replaced (=> union).
(("1","beijing"), ("1", "shanghai")), (("2","beijing"), ("2", "shanghai")), (("3","beijing"), ("3", "shanghai")) =>
(("1","beijing"), ("1", "shanghai"), ("2","beijing"), ("2", "shanghai"), ("3","beijing"), ("3", "shanghai"))

--Replace the enumerated values of partitions tp1, tp2 (=> union).
(("1","beijing"), ("1", "shanghai")), (("2","beijing"), ("2", "shanghai"), ("3","beijing"), ("3", "shanghai")) => (("1","beijing"),
("1", "shanghai"), ("2","beijing"), ("2", "shanghai"), ("3","beijing"), ("3", "shanghai"))

--The enumeration values are the same, you can use tp1, tp2 to replace p1, p2, p3
```

2. `use_temp_partition_name`

The default is false.

When this parameter is false, and the number of partitions to be replaced is the same as the number of replacement partitions, the name of the formal partition after the replacement remains unchanged.

If true, after replacement, the name of the formal partition is the name of the replacement partition. Here are some examples:

Example 1

```
ALTER TABLE tbl1 REPLACE PARTITION (p1) WITH TEMPORARY PARTITION (tp1);
```

`use_temp_partition_name` is **false** by default. After replacement, the partition name is still p1, but the related data and attributes are replaced with tp1.

If `use_temp_partition_name` is **true** by default, the name of the partition is tp1 after replacement. The p1 partition no longer exists.

Example 2

```
ALTER TABLE tbl1 REPLACE PARTITION (p1, p2) WITH TEMPORARY PARTITION (tp1);
```

`use_temp_partition_name` is **false** by default, but this parameter is invalid because the number of partitions to be replaced and the number of replacement partitions are different. After the replacement, the partition name is tp1, and p1 and p2 no longer exist.

> **Tip**
>
> **Some instructions for the replacement operation:**
>
> After the partition is replaced successfully, the replaced partition will be deleted and cannot be recovered.

Importing Temporary Partitions

Depending on the import method, the syntax for specifying the import of Temporary Partitions is slightly different. Here is a brief explanation with an example:

```
INSERT INTO tbl TEMPORARY PARTITION(tp1, tp2, ...) SELECT ....
curl --location-trusted -u root: -H "label:123" -H "temporary_partitions: tp1, tp2, ..." -T testData
http://host:port/api/testDb/testTbl/_stream_load
LOAD LABEL example_db.label1
(
DATA INFILE("hdfs://hdfs_host:hdfs_port/user/palo/data/input/file")
INTO TABLE my_table
TEMPORARY PARTITION (tp1, tp2, ...)
...
)
WITH BROKER hdfs ("username"="hdfs_user", "password"="hdfs_password");
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
COLUMNS(k1, k2, k3, v1, v2, v3 = k1 * 100),
TEMPORARY PARTITIONS(tp1, tp2, ...),
WHERE k1 > 100
PROPERTIES
(...)
FROM KAFKA
(...);
```

Query Temporary Partitions

```
SELECT ... FROM
tbl1 TEMPORARY PARTITION(tp1, tp2, ...)
JOIN
tbl2 TEMPORARY PARTITION(tp1, tp2, ...)
ON ...
WHERE ...;
```

Relationship to Other Operations

DROP

- After using the `DROP` operation to directly drop the database or table, you can recover the database or table (within a limited time) through the `RECOVER` command, but the temporary partition will not be recovered.
- After the formal partition is dropped using the `ALTER` command, the partition can be recovered by the `RECOVER` command (within a limited time). Operating a formal partition is not related to a temporary partition.
- After the temporary partition is dropped using the `ALTER` command, the temporary partition cannot be recovered through the `RECOVER` command.

TRUNCATE

- Use the `TRUNCATE` command to empty the table. The temporary partition of the table will be deleted and cannot be recovered.
- When using `TRUNCATE` command to empty the formal partition, it will not affect the temporary partition.
- You cannot use the `TRUNCATE` command to empty the temporary partition.

ALTER

- When the table has a temporary partition, you cannot use the `ALTER` command to perform Schema Change, Rollup, etc. on the table.
- You cannot add temporary partitions to a table while the table is undergoing a alter operation.

Exporting Overview

`Export` is a feature provided by Doris that allows for the asynchronous export of data. This feature allows the user to export the data of specified tables or partitions in a specified file format through the Broker process or S3 protocol/ HDFS protocol, to remote storage such as object storage or HDFS.

Currently, `EXPORT` supports exporting Doris local tables / views / external tables and supports exporting to file formats including parquet, orc, csv, csv_with_names, and csv_with_names_and_types.

This document mainly introduces the basic principles, usage, best practices and precautions of Export.

Principles

After a user submits an `Export Job`, Doris will calculate all the Tablets involved in this job. Then, based on the `parallelism` parameter specified by the user, these tablets will be grouped. Each thread is responsible for a group of tablets, generating multiple `SELECT INTO OUTFILE` query plans. The query plan will read the data from the included tablets and then write the data to the specified path in remote storage through S3 protocol/ HDFS protocol/ Broker.

The overall execution process is as follows:

1. The user submits an Export job to FE.
2. FE calculates all the tablets to be exported and groups them based on the `parallelism` parameter. Each group generates multiple `SELECT INTO OUTFILE` query plans based on the `maximum_number_of_export_partitions` parameter.
3. Based on the parallelism parameter, an equal number of `ExportTaskExecutor` are generated, and each `ExportTaskExecutor` is responsible for a thread, which is scheduled and executed by FE's `Job scheduler` framework.
4. FE's `Job scheduler` schedules and executes the `ExportTaskExecutor`, and each `ExportTaskExecutor` serially executes the multiple `SELECT INTO OUTFILE` query plans it is responsible for.

Start Export

For detailed usage of Export, please refer to EXPORT.

Export's detailed commands can be passed through `HELP EXPORT;` in mysql client. Examples are as follows:

Export to HDFS

```
EXPORT TABLE db1.tbl1
PARTITION (p1,p2)
[WHERE [expr]]
TO "hdfs://host/path/to/export/"
PROPERTIES
(
    "label" = "mylabel",
```

```
    "column_separator"=",",
    "columns" = "col1,col2",
    "parallelism" = "3"
)
WITH BROKER "hdfs"
(
    "username" = "user",
    "password" = "passwd"
);
```

- `label`: The identifier of this export job. You can use this identifier to view the job status later.
- `column_separator`: Column separator. The default is `\t`. Supports invisible characters, such as''.
- `column`: columns to be exported, separated by commas, if this parameter is not filled in, all columns of the table will be exported by default.
- `line_delimiter`: Line separator. The default is `\n`. Supports invisible characters, such as''.
- `parallelism`: Exporting with 3 concurrent threads.

Export to Object Storage (Supports S3 Protocol)

```
EXPORT TABLE test TO "s3://bucket/path/to/export/dir/"
WITH S3 (
    "s3.endpoint" = "http://host",
    "s3.access_key" = "AK",
    "s3.secret_key"="SK",
    "s3.region" = "region"
);
```

- `s3.access_key`/`s3.secret_key`：Is your key to access the object storage API.
- `s3.endpoint`：Endpoint indicates the access domain name of object storage external services.
- `s3.region`：Region indicates the region where the object storage data center is located.

View Export Status

After submitting a job, the job status can be viewed by querying the SHOW EXPORT command. The results are as follows:

```
mysql> show EXPORT\G;
*************************** 1. row ***************************
     JobId: 14008
     State: FINISHED
  Progress: 100%
  TaskInfo: {"partitions":
[],"max_file_size":"","delete_existing_files":"","columns":"","format":"csv","column_separator":"\t","line_delimiter":"\n","db":"def
ault_cluster:demo","tbl":"student4","tablet_num":30}
      Path: hdfs://host/path/to/export/
CreateTime: 2019-06-25 17:08:24
 StartTime: 2019-06-25 17:08:28
FinishTime: 2019-06-25 17:08:34
   Timeout: 3600
  ErrorMsg: NULL
  OutfileInfo: [
```

```
  [
    {
      "fileNumber": "1",
      "totalRows": "4",
      "fileSize": "34bytes",
      "url": "file:///127.0.0.1/Users/fangtiewei/tmp_data/export/f1ab7dcc31744152-bbb4cda2f5c88eac_"
    }
  ]
]
1 row in set (0.01 sec)
```

- JobId: The unique ID of the job
- State: Job status:
    - PENDING: Jobs to be Scheduled
    - EXPORTING: Data Export
    - FINISHED: Operation Successful
    - CANCELLED: Job Failure
- Progress: Work progress. The schedule is based on the query plan. Assuming there are 10 threads in total and 3 have been completed, the progress will be 30%.
- TaskInfo: Job information in Json format:
    - db: database name
    - tbl: Table name
    - partitions: Specify the exported partition. `empty` Represents all partitions.
    - column separator: The column separator for the exported file.
    - line delimiter: The line separator for the exported file.
    - tablet num: The total number of tablets involved.
    - Broker: The name of the broker used.
        - max_file_size: The maximum size of an export file.
        - delete_existing_files: Whether to delete existing files and directories in the specified export directory.
        - columns: Specifies the column names to be exported. Empty values represent exporting all columns.
        - format: The file format for export.
- Path: Export path on remote storage.
- CreateTime/StartTime/FinishTime: Creation time, start scheduling time and end time of jobs.
- Timeout: Job timeout. The unit is seconds. This time is calculated from CreateTime.

- Error Msg: If there is an error in the job, the cause of the error is shown here.
- OutfileInfo: If the export job is successful, specific `SELECT INTO OUTFILE` result information will be displayed here.

Cancel Export Job

After submitting a job, the job can be canceled by using the CANCEL EXPORT command. For example:

```
CANCEL EXPORT
FROM example_db
WHERE LABEL like "%example%";
```

Best Practices

Concurrent Export

An Export job can be configured with the `parallelism` parameter to concurrently export data. The `parallelism` parameter specifies the number of threads to execute the `EXPORT Job`. Each thread is responsible for exporting a subset of the total tablets.

The underlying execution logic of an `Export Job`is actually the `SELECT INTO OUTFILE` statement. Each thread specified by the `parallelism` parameter executes independent `SELECT INTO OUTFILE` statements.

The specific logic for splitting an `Export Job` into multiple `SELECT INTO OUTFILE` is, to evenly distribute all the tablets of the table among all parallel threads. For example:

- If num(tablets) = 40 and parallelism = 3, then the three threads will be responsible for 14, 13, and 13 tablets, respectively.
- If num(tablets) = 2 and parallelism = 3, then Doris automatically sets the parallelism to 2, and each thread is responsible for one tablet.

When the number of tablets responsible for a thread exceeds the `maximum_tablets_of_outfile_in_export` value (default is 10, and can be modified by adding the `maximum_tablets_of_outfile_in_export` parameter in fe.conf), the thread will split the tablets which are responsibled for this thread into multiple `SELECT INTO OUTFILE` statements. For example:

- If a thread is responsible for 14 tablets and `maximum_tablets_of_outfile_in_export = 10`, then the thread will be responsible for two `SELECT INTO OUTFILE` statements. The first `SELECT INTO OUTFILE` statement exports 10 tablets, and the second `SELECT INTO OUTFILE` statement exports 4 tablets. The two `SELECT INTO OUTFILE` statements are executed serially by this thread.

exec_mem_limit

The query plan for an `Export Job` typically involves only `scanning and exporting`, and does not involve compute logic that requires a lot of memory. Therefore, the default memory limit of 2GB is usually sufficient to meet the requirements.

However, in certain scenarios, such as a query plan that requires scanning too many tablets on the same BE, or when there are too many data versions of tablets, it may result in insufficient memory. In these cases, you can adjust the session variable `exec_mem_limit` to increase the memory usage limit.

Notes

- It is not recommended to export large amounts of data at one time. The maximum amount of exported data recommended by an Export job is tens of GB. Excessive export results in more junk files and higher retry costs.
- If the amount of table data is too large, it is recommended to export it by partition.
- During the operation of the Export job, if FE restarts or cuts the master, the Export job will fail, requiring the user to resubmit.
- If the Export job fails, the temporary files and directory generated in the remote storage will not be deleted, requiring the user to delete them manually.
- Export jobs scan data and occupy IO resources, which may affect the query latency of the system.
- The Export job can export data from `Doris Base tables`, `View`, and `External tables`, but not from `Rollup Index`.
- When using the EXPORT command, please ensure that the target path exists, otherwise the export may fail.
- When concurrent export is enabled, please configure the thread count and parallelism appropriately to fully utilize system resources and avoid performance bottlenecks.
- When exporting to a local file, pay attention to file permissions and the path, ensure that you have sufficient permissions to write, and follow the appropriate file system path.
- It is possible to monitor progress and performance metrics in real-time during the export process to identify issues promptly and make optimal adjustments.
- It is recommended to verify the integrity and accuracy of the exported data after the export operation is completed to ensure the quality and integrity of the data.

Relevant configuration

FE

- `maximum_tablets_of_outfile_in_export`: The maximum number of tablets allowed for an OutFile statement in an ExportExecutorTask.

More Help

For more detailed syntax and best practices used by Export, please refer to the Export command manual, You can also enter `HELP EXPORT` at the command line of the MySql client for more help.

The underlying implementation of the `EXPORT` command is the `SELECT INTO OUTFILE` statement. For more information about SELECT INTO OUTFILE, please refer to Export Query Result and SELECT INTO OUTFILE.

Exporting Query Result

This document describes how to use the SELECT INTO OUTFILE command to export query results.

`SELECT INTO OUTFILE` is a synchronous command, which means that the operation is completed when the command returns. It also returns a row of results to show the execution result of the export.

Examples

Export to HDFS

Export simple query results to the file `hdfs://path/to/result.txt`, specifying the export format as CSV.

```
SELECT * FROM tbl
INTO OUTFILE "hdfs://path/to/result_"
FORMAT AS CSV
PROPERTIES
(
    "broker.name" = "my_broker",
    "column_separator" = ",",
    "line_delimiter" = "\n"
);
```

Export to Local Files

When exporting to a local file, you need to configure `enable_outfile_to_local=true` in fe.conf first

```
select * from tbl1 limit 10
INTO OUTFILE "file:///home/work/path/result_";
```

For more usage, see OUTFILE documentation.

Concurrent Export

By default, the export of the query result set is non-concurrent, that is, a single point of export. If the user wants the query result set to be exported concurrently, the following conditions need to be met:

1. session variable 'enable_parallel_outfile' to enable concurrent export: `set enable_parallel_outfile = true;`
2. The export method is S3, HDFS instead of using a broker
3. The query can meet the needs of concurrent export, for example, the top level does not contain single point nodes such as sort. (I will give an example later, which is a query that does not export the result set concurrently)

If the above three conditions are met, the concurrent export query result set can be triggered. Concurrency = `be_instacne_num * parallel_fragment_exec_instance_num`

How to Verify that the Result Set is Exported Concurrently

After the user enables concurrent export through the session variable setting, if you want to verify whether the current query can be exported concurrently, you can use the following method.

```
explain select xxx from xxx where xxx into outfile "s3://xxx" format as csv properties ("AWS_ENDPOINT" = "xxx", ...);
```

After explaining the query, Doris will return the plan of the query. If you find that `RESULT FILE SINK` appears in `PLAN FRAGMENT 1`, it means that the export concurrency has been opened successfully. If `RESULT FILE SINK` appears in `PLAN FRAGMENT 0`, it means that the current query cannot be exported concurrently (the current query does not satisfy the three conditions of concurrent export at the same time).

```
Planning example for concurrent export:
+------------------------------------------------------------------------------+
| Explain String                                                               |
+------------------------------------------------------------------------------+
| PLAN FRAGMENT 0                                                              |
|  OUTPUT EXPRS:<slot 2> | <slot 3> | <slot 4> | <slot 5>                     |
|   PARTITION: UNPARTITIONED                                                   |
|                                                                              |
|   RESULT SINK                                                                |
|                                                                              |
|   1:EXCHANGE                                                                 |
|                                                                              |
| PLAN FRAGMENT 1                                                              |
|  OUTPUT EXPRS:`k1` + `k2`                                                    |
|   PARTITION: HASH_PARTITIONED: `default_cluster:test`.`multi_tablet`.`k1`    |
|                                                                              |
|   RESULT FILE SINK                                                           |
|   FILE PATH: s3://ml-bd-repo/bpit_test/outfile_1951_                         |
|   STORAGE TYPE: S3                                                           |
|                                                                              |
|   0:OlapScanNode                                                             |
|      TABLE: multi_tablet                                                     |
+------------------------------------------------------------------------------+
```

Usage Examples

For details, please refer to OUTFILE Document.

Return Results

The command is a synchronization command. The command returns, which means the operation is over. At the same time, a row of results will be returned to show the exported execution result.

If it exports and returns normally, the result is as follows:

```
mysql> select * from tbl1 limit 10 into outfile "file:///home/work/path/result_";
+------------+-----------+----------+--------------------------------------------------------------------+
| FileNumber | TotalRows | FileSize | URL                                                                |
+------------+-----------+----------+--------------------------------------------------------------------+
|          1 |         2 |        8 | file:///192.168.1.10/home/work/path/result_{fragment_instance_id}_ |
+------------+-----------+----------+--------------------------------------------------------------------+
1 row in set (0.05 sec)
```

- FileNumber: The number of files finally generated.
- TotalRows: The number of rows in the result set.
- FileSize: The total size of the exported file. Unit byte.
- URL: If it is exported to a local disk, the Compute Node to which it is exported is displayed here.

If a concurrent export is performed, multiple rows of data will be returned.

```
+------------+-----------+----------+--------------------------------------------------------------------+
| FileNumber | TotalRows | FileSize | URL                                                                |
+------------+-----------+----------+--------------------------------------------------------------------+
|          1 |         3 |        7 | file:///192.168.1.10/home/work/path/result_{fragment_instance_id}_ |
|          1 |         2 |        4 | file:///192.168.1.11/home/work/path/result_{fragment_instance_id}_ |
+------------+-----------+----------+--------------------------------------------------------------------+
2 rows in set (2.218 sec)
```

If the execution is incorrect, an error message will be returned, such as:

```
mysql> SELECT * FROM tbl INTO OUTFILE ...
ERROR 1064 (HY000): errCode = 2, detailMessage = Open broker writer failed ...
```

Notice

- The current version of the Nereids optimizer does not support concurrent Outfile. If the Nereids optimizer is enabled, concurrent Outfile will directly revert to the old optimizer, which will handle the concurrent Outfile export. Similarly, the current version of the pipeline engine also does not support concurrent Outfile. If the pipeline engine is enabled, concurrent Outfile will revert to single-threaded export.
- The CSV format does not support exporting binary types, such as BITMAP and HLL types. These types will be output as `\N`, which is null.
- If you do not enable concurrent export, the query result is exported by a single BE node in a single thread. Therefore, the export time and the export result set size are positively correlated. Turning on concurrent export can reduce the export time.
- The export command does not check whether the file and file path exist. Whether the path will be automatically created or whether the existing file will be overwritten is entirely determined by the semantics of the remote storage system.
- If an error occurs during the export process, the exported file may remain on the remote storage system. Doris will not clean these files. The user needs to manually clean up.
- The timeout of the export command is the same as the timeout of the query. It can be set by `SET query_timeout = xxx`.
- For empty result query, there will be an empty file.
- File splitting will ensure that a row of data is stored in a single file. Therefore, the size of the file is not strictly equal to `max_file_size`.

- For functions whose output is invisible characters, such as BITMAP and HLL types, the output is `\N`, which is NULL.
- At present, the output type of some geo functions, such as `ST_Point` is VARCHAR, but the actual output value is an encoded binary character. Currently these functions will output garbled characters. For geo functions, use `ST_AsText` for output.

More Help

For more detailed syntax and best practices for using OUTFILE, please refer to the OUTFILE command manual, you can also More help information can be obtained by typing `HELP OUTFILE` at the command line of the MySql client.

Using MySQL Dump

Doris has supported exporting data or table structures through the `mysqldump` tool after version 0.15

Examples

Export

1. Export the table1 table in the test database: `mysqldump -h127.0.0.1 -P9030 -uroot --no-tablespaces --databases test --tables table1`
2. Export the table1 table structure in the test database: `mysqldump -h127.0.0.1 -P9030 -uroot --no-tablespaces --databases test --tables table1 --no-data`
3. Export all tables in the test1, test2 database: `mysqldump -h127.0.0.1 -P9030 -uroot --no-tablespaces --databases test1 test2`
4. Export all databases and tables `mysqldump -h127.0.0.1 -P9030 -uroot --no-tablespaces --all-databases` For more usage parameters, please refer to the manual of `mysqldump`

Import

The results exported by `mysqldump` can be redirected to a file, which can then be imported into Doris through the source command `source filename.sql`

Notice

1. Since there is no concept of tablespace in mysql in Doris, add the `--no-tablespaces` parameter when using `mysqldump`
2. Using mysqldump to export data and table structure is only used for development and testing or when the amount of data is small. Do not use it in a production environment with a large amount of data.

MySQL Compatibility

Doris is highly compatible with MySQL syntax and supports standard SQL. However, there are several differences between Doris and MySQL, as outlined below.

Data Types

Numeric Types

Type	MySQL	Doris
Boolean		
Bit		Not supported
Tinyint		
Smallint		
Mediumint		- Not supported
Int		
Bigint		
Largeint	- Not supported	
Decimal		
Float/Double		

Date Types

Type	MySQL	Doris
Date		
DateTime		
Timestamp		- Not supported
Time		- Not supported
Year		- Not supported

String Types

Type	MySQL	Doris
Char		

Type	MySQL	Doris
Varchar		
String	- Not supported	
Binary		- Not supported
Varbinary		
Blob		- Not supported
Text		- Not supported
Enum		- Not supported
Set		- Not supported

JSON Type

Type	MySQL	Doris
JSON	- Supported	Supported

Doris unique data type

Doris has several unique data types. Here are the details:

- **HyperLogLog**

  HLL (HyperLogLog) is a data type that cannot be used as a key column. It can be used in aggregate, duplicate, and unique models. In an aggregate model table, the corresponding aggregation type for HLL is HLL_UNION. The length and default value do not need to be specified. The length is controlled internally based on the data aggregation level. HLL columns can only be queried or used with `hll_union_agg`, `hll_raw_agg`, `hll_cardinality`, `hll_hash`, and other related functions.

  HLL is used for approximate fuzzy deduplication and performs better than count distinct when dealing with large amounts of data. The typical error rate of HLL is around 1%, sometimes reaching up to 2%.

- **Bitmap**

  Bitmap is another data type in Doris. It can be used in aggregate, unique, or duplicate models. In Unique or Duplicate models, it must be used as a non-key column. In aggregate models, it must also be used as a non-key column, and the corresponding aggregation type during table creation is BITMAP_UNION. Similar to HLL, the length and default values do not need to be specified, and the length is controlled internally based on the data aggregation level. Bitmap columns can only be queried or used with functions like `bitmap_union_count`, `bitmap_union`, `bitmap_hash`, `bitmap_hash64`, and others.

  Using BITMAP in traditional scenarios may impact loading speed, but it generally performs better than Count Distinct when dealing with large amounts of data. Please note that in real-time scenarios, using BITMAP without a global dictionary and with bitmap_hash() function may introduce an error of around 0.1%. If this error is not acceptable, you can use bitmap_hash64 instead.

- **QUANTILE_PERCENT**

  QUANTILE_STATE is another data type in Doris, which cannot be used as a key column. It can be used in aggregate, duplicate, and iuique models. In an aggregate model table, the corresponding aggregation type for QUANTILE_STATE is QUANTILE_UNION. The length and default value do not need to be specified, and the length is controlled internally based on the data aggregation level. QUANTILE_STATE columns can only be queried or used with functions like `QUANTILE_PERCENT`, `QUANTILE_UNION`, `TO_QUANTILE_STATE`, and others.

  QUANTILE_STATE is used for calculating approximate quantile values. During import, it performs pre-aggregation on the same key with different values. When the number of values does not exceed 2048, it stores all the data in detail. When the number of values exceeds 2048, it uses the TDigest algorithm to aggregate (cluster) the data and save the centroids of the clusters.

- **Array<T>**

  Array is a data type in Doris that represents an array composed of elements of type T. It cannot be used as a key column. Currently, it supports usage in duplicate models and non-key column usage in unique models.

  The supported types for T are `BOOLEAN`, `TINYINT`, `SMALLINT`, `INT`, `BIGINT`, `LARGEINT`, `FLOAT`, `DOUBLE`, `DECIMAL`, `DATE`, DATETIME, CHAR, VARCHAR, and STRING.

- **MAP<K, V>**

  MAP is a data type in Doris that represents a map composed of elements of types K and V. It cannot be used as a key column and can be used in both duplicate and unique models.

  The supported types for K and V are `BOOLEAN`, `TINYINT`, `SMALLINT`, `INT`, `BIGINT`, `LARGEINT`, `FLOAT`, `DOUBLE`, `DECIMAL`, `DATE`, `DATETIME`, `CHAR`, `VARCHAR`, and `STRING`.

- **STRUCT<field_name:field_type,...>**

  A structure (STRUCT) is composed of multiple fields. It can also be identified as a collection of multiple columns. It cannot be used as a key and is currently only supported in tables of the duplicate model.

  - field_name: The identifier of the field, which must be unique.
  - field_type: The type of field.

  The supported types for fields are `BOOLEAN`, `TINYINT`, `SMALLINT`, `INT`, `BIGINT`, `LARGEINT`, `FLOAT`, `DOUBLE`, `DECIMAL`, `DATE`, `DATETIME`, `CHAR`, `VARCHAR`, and `STRING`.

- Agg_State

AGG_STATE is a data type in Doris that cannot be used as a key column. During table creation, the signature of the aggregation function needs to be declared.

The length and default value do not need to be specified, and the actual storage size depends on the implementation of the function.

AGG_STATE can only be used in combination with state /merge/union functions from the SQL manual for aggregators.

Syntax

DDL

01 Create Table Syntax in Doris

```
CREATE TABLE [IF NOT EXISTS] [database.]table
(
    column_definition_list
    [, index_definition_list]
)
[engine_type]
[keys_type]
[table_comment]
[partition_info]
distribution_desc
[rollup_list]
[properties]
[extra_properties]
```

02 Differences with MySQL

- column_definition_list:
    - Defines the list of columns, and the basic syntax is similar to MySQL. However, there are additional operations for **aggregation types**.
    - The **aggregation types** mainly support data models such as Aggregate Key.
    - In MySQL, you can define constraints like Primary Key and Unique Key after each column in the column definition list. In Doris, these constraints are defined and calculated through data models.
- index_definition_list:
    - Defines the list of indexes, and the basic syntax is similar to MySQL. Doris supports bitmap indexes, inverted indexes, and N-Gram indexes, while Bloom filter indexes are set through properties.
    - MySQL supports B+Tree and Hash indexes.
- engine_type:
    - Specifies the table engine type, and it is optional.
    - Currently, Doris mainly supports native engines like OLAP.
    - MySQL supports storage engines like InnoDB and MyISAM.
- keys_type:
    - Specifies the data model, and it is optional.
    - Supported types are:

        - DUBLICATE KEY (default): The specified columns are sorting columns.
        - AGGREGATE KEY: The specified columns are dimension columns.
        - UNIQUE KEY: The specified columns are primary key columns.
    - MySQL does not have the concept of data models.
- table_comment:
    - Table comment or description.
- partition_info:
    - Specifies the partitioning algorithm, and it is optional.
    - Supported partitioning algorithms are:
        - LESS THAN: Defines only the upper bound of the partition. The lower bound is determined by the upper bound of the previous partition.
        - FIXED RANGE: Defines a closed-open interval for the partition.
        - MULTI RANGE: Creates multiple RANGE partitions in batches, defines closed-open intervals, sets time units and steps. Time units supported are year, month, day, week, and hour.
        - MULTI RANGE: Creates multiple RANGE partitions for numeric types, defines closed-open intervals, and sets steps.
    - MySQL supports algorithms like Hash, Range, and List. It also supports subpartitions, but only with the Hash algorithm.
- distribution_desc:
    - Specifies the bucketing algorithm, and it is mandatory.
    - Bucketing algorithms:
        - Hash bucketing syntax: DISTRIBUTED BY HASH (k1[,k2 …]) [BUCKETS num|auto]. It uses the specified key columns for hash bucketing.
        - Random bucketing syntax: DISTRIBUTED BY RANDOM [BUCKETS num|auto]. It uses random numbers for bucketing.
    - MySQL does not have bucketing algorithms.
- rollup_list:
    - Allows creating multiple materialized views while creating the table, and it is optional.
    - Syntax: rollup_name (col1[, col2, …]) [DUPLICATE KEY(col1[, col2, …])] [PROPERTIES(“key” = “value”)]
    - MySQL does not support this feature.

- properties:
    - Table properties.
    - Table properties in Doris are different from MySQL, and the syntax for defining table properties is also different from MySQL.

03 Create-Index

```
CREATE INDEX [IF NOT EXISTS] index_name ON table_name (column [, ...],) [USING BITMAP];
```

- Doris currently supports Bitmap index, Inverted index, and N-Gram index. BloomFilter index are supported as well, but they have a separate syntax for setting them.
- MySQL supports index algorithms such as B+Tree and Hash.

04 Create-View

```
CREATE VIEW [IF NOT EXISTS]
 [db_name.]view_name
 (column1[ COMMENT "col comment"][, column2, ...])
AS query_stmt

CREATE MATERIALIZED VIEW (IF NOT EXISTS)? mvName=multipartIdentifier
        (LEFT_PAREN cols=simpleColumnDefs RIGHT_PAREN)? buildMode?
        (REFRESH refreshMethod? refreshTrigger?)?
        (KEY keys=identifierList)?
        (COMMENT STRING_LITERAL)?
        (PARTITION BY LEFT_PAREN partitionKey = identifier RIGHT_PAREN)?
        (DISTRIBUTED BY (HASH hashKeys=identifierList | RANDOM) (BUCKETS (INTEGER_VALUE | AUTO))?)?
        propertyClause?
        AS query
```

- The basic syntax is consistent with MySQL.
- Doris supports two types of materialized views: synchronous materialized views and asynchronous materialized views (supported for v2.1). The asynchronous materialized views in Doris are more powerful.
- MySQL only supports asynchronous materialized views.

05 Alter-Table/Alter-Index

The syntax of Doris ALTER is basically the same as that of MySQL.

Drop-Table/Drop-Index

The syntax of Doris DROP is basically the same as MySQL.

DML

Insert

```
INSERT INTO table_name
    [ PARTITION (p1, ...) ]
    [ WITH LABEL label]
    [ (column [, ...]) ]
    [ [ hint [, ...] ] ]
    { VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | query }
```

The Doris INSERT syntax is basically the same as MySQL.

Update

```
UPDATE target_table [table_alias]
    SET assignment_list
    WHERE condition

assignment_list:
    assignment [, assignment] ...

assignment:
    col_name = value

value:
    {expr | DEFAULT}
```

The Doris UPDATE syntax is basically the same as MySQL, but it should be noted that the `WHERE` **condition must be added.**

Delete

```
DELETE FROM table_name [table_alias]
    [PARTITION partition_name | PARTITIONS (partition_name [, partition_name])]
    WHERE column_name op { value | value_list } [ AND column_name op { value | value_list } ...];
```

The syntax can only specify filter predicates

```
DELETE FROM table_name [table_alias]
    [PARTITION partition_name | PARTITIONS (partition_name [, partition_name])]
    [USING additional_tables]
    WHERE condition
```

This syntax can only be used on the UNIQUE KEY model table.

The DELETE syntax in Doris is basically the same as in MySQL. However, since Doris is an analytical database, deletions cannot be too frequent.

Select

```
SELECT
    [hint_statement, ...]
    [ALL | DISTINCT | DISTINCTROW | ALL EXCEPT ( col_name1 [, col_name2, col_name3, ...] )]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
      [TABLET tabletid_list]
      [TABLESAMPLE sample_value [ROWS | PERCENT]
        [REPEATABLE pos_seek]]
    [WHERE where_condition]
    [GROUP BY [GROUPING SETS | ROLLUP | CUBE] {col_name | expr | position}]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [INTO OUTFILE 'file_name']
```

The Doris SELECT syntax is basically the same as MySQL.

SQL Function

Doris Function covers most MySQL functions.

Select Manual

Select

Select syntax

```
SELECT
    [hint_statement, ...]
    [ALL | DISTINCT | DISTINCTROW | ALL EXCEPT ( col_name1 [, col_name2, col_name3, ...] )]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
      [TABLET tabletid_list]
      [TABLESAMPLE sample_value [ROWS | PERCENT]
        [REPEATABLE pos_seek]]
    [WHERE where_condition]
    [GROUP BY [GROUPING SETS | ROLLUP | CUBE] {col_name | expr | position}]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [INTO OUTFILE 'file_name']
```

Syntax explanation

- `select_expr, ...`Specifies the columns to retrieve and display in the result set. Aliases can be used, and `as` is optional.
- `table_references`Specifies the target tables for retrieval, which can be one or more tables (including temporary tables generated by subqueries).
- `where_definition`Specifies the retrieval conditions (expressions). If a WHERE clause exists, its conditions filter the row data. The `where_condition` is an expression that evaluates to true for each row to be selected. If there is no WHERE clause, the statement selects all rows. In a WHERE expression, you can use any MySQL-supported functions and operators except aggregate functions.
- `ALL | DISTINCT`Filters the result set. `ALL` selects all rows, while `DISTINCT` or `DISTINCTROW` filters out duplicate rows. The default is `ALL`.
- `ALL EXCEPT`Filters the result set from `ALL` by specifying one or more column names to exclude from the full result set. All matching column names will be ignored in the output.
- `INTO OUTFILE 'file_name'`Saves the result set to a new file (which must not exist beforehand), with differences in the saved format.
- `Group by having`Groups the result set by one or more columns. If `HAVING` is present, it filters the groups produced by `GROUP BY`. Extensions to `GROUP BY` such as `GROUPING SETS`, `ROLLUP`, and `CUBE` are available and detailed in the GROUPING SETS.
- `Order by`Sorts the final result set. `ORDER BY` sorts the result set by comparing values in one or more columns. Sorting operations can be time-consuming and resource-intensive because all data needs to be sent to a single node for sorting. Sorting requires more memory compared to non-sorted operations. If you need to return the top N sorted results, use the `LIMIT` clause.
- `Limit n`Limits the number of rows in the output result set. `LIMIT m,n` means to start outputting from the mth row and return n records. Using `LIMIT m,n` is meaningful only when combined with `ORDER BY`,

otherwise the data returned may be inconsistent each time the query is executed.

- `Having`The `HAVING` clause does not filter rows in the table but filters the results produced by aggregate functions. Typically, `HAVING` is used with aggregate functions (such as `COUNT()`, `SUM()`, `AVG()`, `MIN()`, `MAX()`) and the `GROUP BY` clause.
- `SELECT` supports explicit partition selection using `PARTITION`, which includes a list of partitions or subpartitions (or both) following the table name in `table_reference`.
- `[TABLET tids] TABLESAMPLE n [ROWS | PERCENT] [REPEATABLE seek]`Limits the number of rows read from a table in the `FROM` clause by pseudo-randomly selecting several tablets based on the specified number of rows or percentage. `REPEATABLE` with a specified seed allows the same sample to be returned again. Alternatively, Tablet IDs can be manually specified, but this is only applicable to OLAP tables.
- `hint_statement`Using hints before the select list can influence the optimizer's behavior to obtain a desired execution plan. For more information, refer to the joinHint Document.

Syntax constraints

- `SELECT` can also be used to retrieve calculated rows without referencing any tables.
- All clauses must strictly follow the above format. A `HAVING` clause must come after the `GROUP BY` clause and before the `ORDER BY` clause.
- The alias keyword `AS` is optional. Aliases can be used in `GROUP BY`, `ORDER BY`, and `HAVING`.
- `WHERE` clause: Executes the `WHERE` statement to determine which rows should be included in the `GROUP BY` section, while `HAVING` is used to determine which rows from the result set should be used.
- The `HAVING` clause can reference aggregate functions, such as `count, sum, max, min, avg`, while the `WHERE` clause cannot. However, the `WHERE` clause can reference other functions besides aggregate functions. Column aliases cannot be used in the `WHERE` clause to define conditions.
- Following `GROUP BY` with `WITH ROLLUP` allows for one or more aggregations of the results.

Join syntax

Doris supports the following JOIN syntax.

```
JOIN
table_references:
    table_reference [, table_reference] ...
table_reference:
    table_factor
  | join_table
table_factor:
    tbl_name [[AS] alias]
        [{USE|IGNORE|FORCE} INDEX (key_list)]
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
        ON conditional_expr }
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference LEFT [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [LEFT [OUTER]] JOIN table_factor
  | table_reference RIGHT [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [RIGHT [OUTER]] JOIN table_factor
join_condition:
    ON conditional_expr
```

UNION:

```
SELECT ...
UNION [ALL| DISTINCT] SELECT ......
[UNION [ALL| DISTINCT] SELECT ...]
```

`UNION` is used to combine the results of multiple `SELECT` statements into a single result set. The column names from the first `SELECT` statement are used as the column names for the returned result. The selected columns listed in the corresponding positions of each `SELECT` statement should have the same data type. (For example, the first column selected in the first statement should have the same type as the first column selected in the other statements.)

By default, `UNION` removes duplicate rows from the result. The optional `DISTINCT` keyword has no effect beyond the default, as it also specifies duplicate row removal. Using the optional `ALL` keyword, no duplicate row removal occurs, and the result includes all matching rows from all `SELECT` statements.

INTERSECT:

```
SELECT ...
INTERSECT [DISTINCT] SELECT ......
[INTERSECT [DISTINCT] SELECT ...]
```

`INTERSECT` is used to return the intersection of results from multiple `SELECT` statements, with duplicate results removed. The effect of `INTERSECT` is equivalent to `INTERSECT DISTINCT`. The `ALL` keyword is not supported. Each `SELECT` query must return the same number of columns, And when the column types are inconsistent, they will be `CAST` to the same type.

EXCEPT/MINUS:

```
SELECT ...
EXCEPT [DISTINCT] SELECT ......
[EXCEPT [DISTINCT] SELECT ...]
```

The `EXCEPT` clause is used to return the complement between the results of multiple queries, meaning it returns the data from the left query that does not exist in the right query, with duplicates removed. `EXCEPT` is functionally equivalent to `MINUS`. The effect of `EXCEPT` is the same as `EXCEPT DISTINCT`. The `ALL` keyword is not supported. Each `SELECT` query must return the same number of columns, And when the column types are inconsistent, they will be `CAST` to the same type.

WITH:

To specify a common table expression, use a `WITH` clause with one or more comma-separated subclauses. Each subclause provides a subquery that generates a result set and associates a name with the subquery. The following example defines CTEs named `cte1` and `cte2` in the `WITH` clause, and refers to them in the top-level `SELECT`following the WITH clause.

```
WITH
  cte1 AS (SELECT a, b FROM table1),
  cte2 AS (SELECT c, d FROM table2)
SELECT b, d FROM cte1 JOIN cte2
WHERE cte1.a = cte2.c;
```

In statements that include this `WITH` clause, each CTE name can be referenced to access the corresponding CTE result set. CTE names can be referenced in other CTEs, allowing CTEs to be defined based on other CTEs. Currently, recursive CTEs are not supported.

Example

- Query the names of students whose ages are 18, 20, and 25.

```
select Name from student where age in (18,20,25);
```

- ALL EXCEPT

```
-- Query all information except for the age of the students.
select * except(age) from student;
```

- GROUP BY

```
--Query the tb_book table, group by type, and calculate the average price for each category of books.
select type,avg(price) from tb_book group by type;
```

- DISTINCT

```
--Query the tb_book table and remove duplicate type data.
select distinct type from tb_book;
```

- ORDER BY

Sort the query results in ascending order (by default) or descending order (DESC). In ascending order, NULL values should appear at the beginning, and in descending order, NULL values should appear at the end.

```
--Query all records from the tb_book table, sort them in descending order by id, and display only the first three records.
select * from tb_book order by id desc limit 3;
```

- LIKE

`LIKE` can perform fuzzy queries with two wildcards: `%` and _. The `%` wildcard matches one or more characters, while the _ wildcard matches a single character.

```
-- Find all books where the second character is 'h'.
select * from tb_book where name like('_h%');
```

- LIMIT (Limit the number of result rows.)

```
-- Display 3 records in descending order.
select * from tb_book order by price desc limit 3;

Display 4 records starting from id=1
select * from tb_book where id limit 1,4;
```

- CONCAT (Concatenate multiple columns

```
--Concatenate 'name' and 'price' into a new string for output.
select id,concat(name,":",price) as info,type from tb_book;
```

- Functions and expressions

```
--Calculate the total price of each category of books in the tb_book table.
select sum(price) as total,type from tb_book group by type;
--20% off the price
select *,(price * 0.8) as "20% off" from tb_book;
```

- UNION

```
SELECT a FROM t1 WHERE a = 10 AND B = 1 ORDER by a LIMIT 10
UNION
SELECT a FROM t2 WHERE a = 11 AND B = 2 ORDER by a LIMIT 10;
```

- INTERSECT

```
SELECT a FROM t1 WHERE a = 10 AND B = 1 ORDER by a LIMIT 10
INTERSECT
SELECT a FROM t2 WHERE a = 11 AND B = 2 ORDER by a LIMIT 10;
```

- EXCEPT

```
SELECT a FROM t1 WHERE a = 10 AND B = 1 ORDER by a LIMIT 10
EXCEPT
SELECT a FROM t2 WHERE a = 11 AND B = 2 ORDER by a LIMIT 10;
```

- WITH clause

```
WITH cte AS
(
  SELECT 1 AS col1, 2 AS col2
  UNION ALL
  SELECT 3, 4
)
SELECT col1, col2 FROM cte;
```

- JOIN

```
SELECT * FROM t1 LEFT JOIN (t2, t3, t4)
                ON (t2.a = t1.a AND t3.b = t1.b AND t4.c = t1.c)
```

the same as

```
SELECT * FROM t1 LEFT JOIN (t2 CROSS JOIN t3 CROSS JOIN t4)
                ON (t2.a = t1.a AND t3.b = t1.b AND t4.c = t1.c)
```

- INNER JOIN

```
SELECT t1.name, t2.salary
  FROM employee AS t1 INNER JOIN info AS t2 ON t1.name = t2.name;

SELECT t1.name, t2.salary
  FROM employee t1 INNER JOIN info t2 ON t1.name = t2.name;
```

- LEFT JOIN

```
SELECT left_tbl.*
  FROM left_tbl LEFT JOIN right_tbl ON left_tbl.id = right_tbl.id
  WHERE right_tbl.id IS NULL;
```

- RIGHT JOIN

```
mysql SELECT * FROM t1 RIGHT JOIN t2 ON (t1.a = t2.a);
+------+------+------+------+
| a    | b    | a    | c    |
+------+------+------+------+
|    2 | y    |    2 | z    |
| NULL | NULL |    3 | w    |
+------+------+------+------+
```

- TABLESAMPLE

```
--Randomly sample 1000 rows in t1 using pseudo-random method. Note that the actual process is to select several Tablets based on the statistical information of the table, and the total number of rows in the selected Tablets may be greater than 1000. Therefore, if you want to return exactly 1000 rows, you need to add a Limit clause.
SELECT * FROM t1 TABLET(10001) TABLESAMPLE(1000 ROWS) REPEATABLE 2 limit 1000;
```

Best practice

- Additional explanation about the SELECT clause:
  - An alias can be specified for `select_expr` using `AS alias_name`. The alias serves as the column name for the expression and can be used in `GROUP BY`, `ORDER BY`, or `HAVING` clauses.
  - The `table_references` after `FROM` indicate one or multiple tables involved in the query. If multiple tables are listed, a `JOIN` operation will be performed. Each specified table can be assigned an alias.
  - The selected columns after `SELECT` can be referenced in `ORDER BY` and `GROUP BY` clauses using column names, column aliases, or integers representing the column position (starting from 1).

```
SELECT college, region, seed FROM tournament
  ORDER BY region, seed;

SELECT college, region AS r, seed AS s FROM tournament
  ORDER BY r, s;

SELECT college, region, seed FROM tournament
  ORDER BY 2, 3;
```

- If `ORDER BY` appears in a subquery and is also applied to the outer query, the outermost `ORDER BY` takes precedence.

- When using `GROUP BY`, the grouped columns are automatically sorted in ascending order (as if an `ORDER BY` clause followed with the same columns). To avoid the overhead caused by the automatic sorting of `GROUP BY`, adding `ORDER BY NULL` can solve the issue:

```
SELECT a, COUNT(b) FROM test_table GROUP BY a ORDER BY NULL;
```

- When sorting columns in a `SELECT` statement using `ORDER BY` or `GROUP BY`, the server only sorts the values using the initial number of bytes indicated by the `max_sort_length` system variable.
- The `HAVING` clause is typically applied at the end, just before the result set is returned to the client, and it is not optimized (whereas `LIMIT` is applied after `HAVING`).
- According to the SQL standard, `HAVING` must reference columns that are either in the `GROUP BY` list or used in aggregate functions. However, MySQL extends this by allowing `HAVING` to reference columns from the `SELECT` clause list and columns from outer subqueries.
- If a column referenced in `HAVING` is ambiguous, a warning will be generated. In the following statement, `col2` is ambiguous:

```
SELECT COUNT(col1) AS col2 FROM t GROUP BY col2 HAVING col2 = 2;
```

Do not use `HAVING` where `WHERE` should be used. `HAVING` is intended to be used with `GROUP BY`.

The `HAVING` clause can reference aggregate functions, whereas `WHERE` cannot.

```
SELECT user, MAX(salary) FROM users
  GROUP BY user HAVING MAX(salary)  10;
```

- The `LIMIT` clause can be used to restrict the number of rows returned by a `SELECT` statement. `LIMIT` can have one or two parameters, both of which must be non-negative integers.

```
-- Retrieve rows 6 to 15 from the result set.
SELECT * FROM tbl LIMIT 5,10;
-- If you want to retrieve all rows starting from a certain offset, you can set a very large constant as the second parameter. The
following query retrieves all data starting from the 96th row:
SELECT * FROM tbl LIMIT 95,18446744073709551615;
-- If LIMIT has only one parameter, then the parameter specifies the number of rows that should be retrieved, and the offset is
defaulted to 0, which means starting from the first row.
```

- `SELECT...INTO` allows the query results to be written to a file.
- Modifiers for the `SELECT` keyword:
    - Primarily used for removing duplicates.
    - The `ALL` and `DISTINCT` modifiers specify whether to remove duplicate rows (not a specific column) from the result set.
    - `ALL` is the default modifier, meaning all rows that meet the criteria will be retrieved.
    - `DISTINCT`removes duplicate rows.
- Key advantages of subqueries:
    - Subqueries enable structured queries, allowing each part of a statement to be isolated.
    - Some operations require complex joins and associations. Subqueries provide alternative methods to perform these operations.
- Accelerating queries:

    - Utilize Doris's partitioning and bucketing as data filtering conditions to reduce the data scanning range as much as possible.
    - Make full use of Doris's prefix index fields as data filtering conditions to speed up query performance.
- UNION:Using only the `union` keyword has the same effect as using `union distinct`. Since deduplication can be memory-intensive, using `union all` for queries can result in faster performance and reduced memory consumption. If users want to perform `order by` and `limit` operations on the returned result set, they should place the `union` operation within a subquery, then select from that subquery, and finally, place the subquery along with `order by` outside.

```
select * from (select age from student_01 union all select age from student_02) as t1
order by age limit 4;

+-------------+
|     age     |
+-------------+
|      18     |
|      19     |
|      20     |
|      21     |
+-------------+
4 rows in set (0.01 sec)
```

- JOIN
    - In addition to supporting equi-join in inner join conditions, non-equi-join is also supported. However, for performance considerations, it is recommended to use equi-join.
    - Other types of joins only support equi-join.

Complex Type

Doris supports complex types such as Array, Map, Struct, and JSON.

Doris provides various functions specifically designed for these complex types.

For detailed information on the supported functions, please refer to the SQL manual- Array functions, SQL manual-Struct functions, and SQL manual-JSON functions.

To check the support for Map functions, you can refer to the SQL manual docs under the section for the MAP data type.

Subquery

A subquery is a `SELECT` statement that is nested within another `SELECT` statement. The nested subquery is often referred to as the inner query, while the containing query is known as the outer query or the outer query block. The subquery returns data that is used as a condition by the outer query to determine which data needs to be retrieved. There is no limit to the number of nested subqueries you can create.

Like any query, a subquery can return records from a table as single-column single-record, single-column multiple-record, or multi-column multiple-record.

Subquery example in Where clause

```
SELECT * FROM sub_query_correlated_subquery1 WHERE k1 > (SELECT AVG(k1) FROM sub_query_correlated_subquery3) OR k1 < 10 order by k1,
k2;
```

```
select * from sub_query_correlated_subquery1 where sub_query_correlated_subquery1.k1 not in (select
sub_query_correlated_subquery3.k3 from sub_query_correlated_subquery3 where sub_query_correlated_subquery3.v2 =
sub_query_correlated_subquery1.k2) or k1 < 10 order by k1, k2
```

Subquery example in Join clause

```
select t1.* from t1 left join t2 on t1.k2 = t2.k3 and t1.k1 in ( select t3.k1 from t3 where t1.k2 = t3.k2 ) or t1.k1 < 10 order by
t1.k1, t1.k2;
select t1.* from t1 left join t2 on t1.k2 = t2.k3 and exists ( select t3.k1 from t3 where t1.k2 = t3.k2 ) or t1.k1 < 10 order by
t1.k1, t1.k2;
```

Common Table Expressions

Description

Common Table Expression (CTE) define a temporary result set that can be referenced multiple times within the scope of an SQL statement. CTE are primarily used in SELECT statements.

To specify a CTE, use the `WITH` clause with one or more comma-separated clauses. Each clause provides a subquery that generates a result set and associates a name with the subquery.

Within the statement that contains the `WITH` clause, you can reference each CTE name to access the corresponding CTE result set. CTE names can be referenced in other CTE, allowing you to define CTE based on other CTE.

Doris **DO NOT** support recursive CTE.

Example

The following example defines CTE named cte1 and cte2 within the WITH clause and refers to them in the top-level SELECT below the WITH clause:

```
WITH
  cte1 AS (SELECT a, b FROM table1),
  cte2 AS (SELECT c, d FROM table2)
SELECT b, d FROM cte1 JOIN cte2
WHERE cte1.a = cte2.c;
```

Column to Row (Lateral View)

Column to Row (Lateral View)

Used in conjunction with generator functions such as `EXPLODE`, will generate a virtual table containing one or more rows. `LATERAL VIEW` applies rows to each raw output row.

Grammar

```
LATERAL VIEW  generator_function ( expression [, ...] ) table_identifier AS column_identifier [, ...]
```

Parameters

- generator_function

  Generator functions (EXPLODE, EXPLODE_SPLIT, etc.).

- table_identifier

  Alias for `generator_function`.

- column_identifier

  List column alias `generator_function`, which can be used to output rows. The number of column identifiers must match the number of columns returned by the generator function.

Example

```
CREATE TABLE `person` (
  `id` int(11) NULL,
  `name` text NULL,
  `age` int(11) NULL,
  `class` int(11) NULL,
  `address` text NULL
) ENGINE=OLAP
UNIQUE KEY(`id`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
"in_memory" = "false",
"storage_format" = "V2",
"disable_auto_compaction" = "false"
);

INSERT INTO person VALUES
    (100, 'John', 30, 1, 'Street 1'),
    (200, 'Mary', NULL, 1, 'Street 2'),
    (300, 'Mike', 80, 3, 'Street 3'),
    (400, 'Dan', 50, 4, 'Street 4');
```

```
mysql> SELECT * FROM person
    ->     LATERAL VIEW EXPLODE(ARRAY(30, 60)) tableName AS c_age;
+------+------+------+-------+----------+-------+
| id   | name | age  | class | address  | c_age |
+------+------+------+-------+----------+-------+
|  100 | John |   30 |     1 | Street 1 |    30 |
|  100 | John |   30 |     1 | Street 1 |    60 |
|  200 | Mary | NULL |     1 | Street 2 |    30 |
|  200 | Mary | NULL |     1 | Street 2 |    60 |
|  300 | Mike |   80 |     3 | Street 3 |    30 |
|  300 | Mike |   80 |     3 | Street 3 |    60 |
|  400 | Dan  |   50 |     4 | Street 4 |    30 |
|  400 | Dan  |   50 |     4 | Street 4 |    60 |
+------+------+------+-------+----------+-------+
8 rows in set (0.12 sec)
```

Window Function

Window function

Window functions are a special type of built-in functions in databases. Similar to aggregate functions, window functions perform calculations on multiple input rows to obtain a single data value. However, the difference lies in the fact that window functions process the input data within a specific window, rather than grouping and calculating based on the `GROUP BY` clause. The data within each window can be sorted and grouped using the `OVER()` clause. Window functions calculate a separate value for each row of the result set, rather than a single value for each `GROUP BY`. This flexible approach allows users to add additional columns in the `SELECT` clause, providing more opportunities to reorganize and filter the result set. Window functions can only appear in the select list and the outermost `ORDER BY` clause. During the query process, window functions take effect at the end, meaning they are executed after operations such as `JOIN`, `WHERE`, and `GROUP BY`. Window functions are often used in finance and scientific computing to analyze trends, calculate outliers, and perform bucket analysis on large amounts of data.

The syntax of window functions as follows:

```
function(args) OVER(partition_by_clause order_by_clause [window_clause])
partition_by_clause ::= PARTITION BY expr [, expr ...]
order_by_clause ::= ORDER BY expr [ASC | DESC] [, expr [ASC | DESC] ...]
```

Function

The currently supported functions include AVG(), COUNT(), DENSE_RANK(), FIRST_VALUE(), LAG(), LAST_VALUE(), LEAD(), MAX(), MIN(), RANK(), ROW_NUMBER(), SUM() and all aggregate functions.

PARTITION BY clause

The `Partition By` clause is similar to `Group By`. It groups input rows based on the specified one or more columns, where rows with the same values are placed in the same group.

ORDER BY clause

The `Order By` clause within a window function behaves similarly to the outer-level `Order By`. It defines the arrangement of input rows, and when `Partition By` is specified, the `Order By` determines the order within each partition. The only difference from the outer `Order By` is that within the `OVER` clause, using `Order By n` (where n is a positive integer) effectively does nothing, whereas in the outer context, `Order By n` signifies sorting based on the nth column.

This example demonstrates adding an `id` column to the select list, where its values are 1, 2, 3, and so on, sorted according to the `date_and_time` column in the `events` table.

```
SELECT
row_number() OVER (ORDER BY date_and_time) AS id,
```

```
c1, c2, c3, c4
FROM events;
```

Window clause

The Window clause is used to specify a computational range for window functions. It considers the current row and a specified number of rows before and after it as the target for the window function's operation. The methods supported by the Window clause include: AVG(), COUNT(), FIRST_VALUE(), LAST_VALUE(), and SUM(). For MAX() and MIN(), the Window clause can specify a starting range of UNBOUNDED PRECEDING.

```
ROWS BETWEEN [ { m | UNBOUNDED } PRECEDING | CURRENT ROW] [ AND [CURRENT ROW | { UNBOUNDED | n } FOLLOWING] ]
```

Example

Taking the following stock data as an example, the stock code is JDR, and the "closing price" refers to the daily closing quotation.

```
create table stock_ticker (stock_symbol string, closing_price decimal(8,2), closing_date datetime);
...load some data...
select * from stock_ticker order by stock_symbol, closing_date
```

```
| stock_symbol | closing_price | closing_date        |
|--------------|---------------|---------------------|
| JDR          | 12.86         | 2014-10-02 00:00:00 |
| JDR          | 12.89         | 2014-10-03 00:00:00 |
| JDR          | 12.94         | 2014-10-04 00:00:00 |
| JDR          | 12.55         | 2014-10-05 00:00:00 |
| JDR          | 14.03         | 2014-10-06 00:00:00 |
| JDR          | 14.75         | 2014-10-07 00:00:00 |
| JDR          | 13.98         | 2014-10-08 00:00:00 |
```

This query utilizes a window function to generate the `moving_average` column, which calculates the average stock price over a three-day span, specifically the previous day, the current day, and the next day. Since there is no previous day's data for the first day and no next day's data for the last day, the average is only calculated based on two days for those rows. In this case, the `Partition By` clause is not relevant because all the data pertains to the stock JDR. However, if there was additional stock information, `Partition By` would ensure that the window function operates exclusively within each partition.

```
select stock_symbol, closing_date, closing_price,
avg(closing_price) over (partition by stock_symbol order by closing_date
rows between 1 preceding and 1 following) as moving_average
from stock_ticker;
```

```
| stock_symbol | closing_date        | closing_price | moving_average |
|--------------|---------------------|---------------|----------------|
| JDR          | 2014-10-02 00:00:00 | 12.86         | 12.87          |
| JDR          | 2014-10-03 00:00:00 | 12.89         | 12.89          |
| JDR          | 2014-10-04 00:00:00 | 12.94         | 12.79          |
| JDR          | 2014-10-05 00:00:00 | 12.55         | 13.17          |
| JDR          | 2014-10-06 00:00:00 | 14.03         | 13.77          |
```

```
| JDR          | 2014-10-07 00:00:00 | 14.75         | 14.25          |
| JDR          | 2014-10-08 00:00:00 | 13.98         | 14.36          |
```

See more

For more window functions, refer to Window Functions

Encrption and Masking Function

Doris provides the following built-in encryption and masking functions. For detailed usage, please refer to the SQL manual.

- AES_ENCRYPT
- AES_DECRYPT
- SM4_ENCRYPT
- SM4_DECRYPT
- MD5
- MD5SUM
- SM3
- SM3SUM
- SHA
- SHA2
- DIGITAL_MASKING

Variable

Variable

This document focuses on currently supported variables.

Variables in Doris refer to variable settings in MySQL. However, some of the variables are only used to be compatible with some MySQL client protocols, and do not produce their actual meaning in the MySQL database.

Variable setting and viewing

View

All or specified variables can be viewed via `SHOW VARIABLES [LIKE 'xxx'];`. Such as:

```
SHOW VARIABLES;
SHOW VARIABLES LIKE '%time_zone%';
```

Settings

Note that before version 1.1, after the setting takes effect globally, the setting value will be inherited in subsequent new session connections, but the value in the current session will remain unchanged. After version 1.1 (inclusive), after the setting takes effect globally, the setting value will be used in subsequent new session connections, and the value in the current session will also change.

For session-only, set by the `SET var_name=xxx;` statement. Such as:

```
SET exec_mem_limit = 137438953472;
SET forward_to_master = true;
SET time_zone = "Asia/Shanghai";
```

For global-level, set by `SET GLOBAL var_name=xxx;`. Such as:

```
SET GLOBAL exec_mem_limit = 137438953472
```

> Note 1: Only ADMIN users can set variable at global-level.

Variables that support both session-level and global-level setting include:

- `time_zone`
- `wait_timeout`
- `sql_mode`
- `enable_profile`

- query_timeout
- insert_timeout
- exec_mem_limit
- batch_size
- parallel_fragment_exec_instance_num
- parallel_exchange_instance_num
- allow_partition_column_nullable
- insert_visible_timeout_ms
- enable_fold_constant_by_be

Variables that support only global-level setting include:

- default_rowset_type
- default_password_lifetime
- password_history
- validate_password_policy

At the same time, variable settings also support constant expressions. Such as:

```
SET exec_mem_limit = 10 * 1024 * 1024 * 1024;
SET forward_to_master = concat('tr', 'u', 'e');
```

Set variables in the query statement

In some scenarios, we may need to set variables specifically for certain queries. The SET_VAR hint sets the session value of a system variable temporarily (for the duration of a single statement). Examples:

```
SELECT /*+ SET_VAR(exec_mem_limit = 8589934592) */ name FROM people ORDER BY name;
SELECT /*+ SET_VAR(query_timeout = 1, enable_partition_cache=true) */ sleep(3);
```

Note that the comment must start with /*+ and can only follow the SELECT.

Supported variables

- SQL_AUTO_IS_NULL

  Used for compatible JDBC connection pool C3P0. No practical effect.

- auto_increment_increment

  Used for compatibility with MySQL clients. No practical effect. Although Doris already has AUTO_INCREMENT feature, but this parameter does not affect the behavior of 'AUTO_INCREMENT'. Same as auto_increment_offset.

- autocommit

  Used for compatibility with MySQL clients. No practical effect.

- auto_broadcast_join_threshold

The maximum size in bytes of the table that will be broadcast to all nodes when a join is performed, broadcast can be disabled by setting this value to -1.

The system provides two join implementation methods, `broadcast join` and `shuffle join`.

`broadcast join` means that after conditional filtering the small table, broadcast it to each node where the large table is located to form an in-memory Hash table, and then stream the data of the large table for Hash Join.

`shuffle join` refers to hashing both small and large tables according to the join key, and then performing distributed join.

`broadcast join` has better performance when the data volume of the small table is small. On the contrary, shuffle join has better performance.

The system will automatically try to perform a Broadcast Join, or you can explicitly specify the implementation of each join operator. The system provides a configurable parameter `auto_broadcast_join_threshold`, which specifies the upper limit of the memory used by the hash table to the overall execution memory when `broadcast join` is used. The value ranges from 0 to 1, and the default value is 0.8. When the memory used by the system to calculate the hash table exceeds this limit, it will automatically switch to using `shuffle join`

The overall execution memory here is: a fraction of what the query optimizer estimates

> Note:
>
> It is not recommended to use this parameter to adjust, if you must use a certain join, it is recommended to use hint, such as join[shuffle]

- `batch_size`

  Used to specify the number of rows of a single packet transmitted by each node during query execution. By default, the number of rows of a packet is 1024 rows. That is, after the source node generates 1024 rows of data, it is packaged and sent to the destination node.

  A larger number of rows will increase the throughput of the query in the case of scanning large data volumes, but may increase the query delay in small query scenario. At the same time, it also increases the memory overhead of the query. The recommended setting range is 1024 to 4096.

- `character_set_client`

  Used for compatibility with MySQL clients. No practical effect.

- `character_set_connection`

  Used for compatibility with MySQL clients. No practical effect.

- `character_set_results`

  Used for compatibility with MySQL clients. No practical effect.

- `character_set_server`

  Used for compatibility with MySQL clients. No practical effect.

- `codegen_level`

  Used to set the level of LLVM codegen. (Not currently in effect).

- `collation_connection`

  Used for compatibility with MySQL clients. No practical effect.

- `collation_database`

  Used for compatibility with MySQL clients. No practical effect.

- `collation_server`

  Used for compatibility with MySQL clients. No practical effect.

- `have_query_cache`

  Used for compatibility with MySQL clients. No practical effect.

- `default_order_by_limit`

  Used to control the default number of items returned after OrderBy. The default value is -1, and the maximum number of records after the query is returned by default, and the upper limit is the MAX_VALUE of the long data type.

- `delete_without_partition`

  When set to true. When using the delete command to delete partition table data, no partition is required. The delete operation will be automatically applied to all partitions.

  Note, however, that the automatic application to all partitions may cause the delete command to take a long time to trigger a large number of subtasks and cause a long time. If it is not necessary, it is not recommended to turn it on.

- `disable_colocate_join`

  Controls whether the Colocation Join function is enabled. The default is false, which means that the feature is enabled. True means that the feature is disabled. When this feature is disabled, the query plan will not attempt to perform a Colocation Join.

- `enable_bucket_shuffle_join`

  Controls whether the Bucket Shuffle Join function is enabled. The default is true, which means that the feature is enabled. False means that the feature is disabled. When this feature is disabled, the query plan will not attempt to perform a Bucket Shuffle Join.

- `disable_streaming_preaggregations`

  Controls whether streaming pre-aggregation is turned on. The default is false, which is enabled. Currently not configurable and enabled by default.

- `enable_insert_strict`

  Used to set the `strict` mode when loading data via INSERT statement. The default is false, which means that the `strict` mode is not turned on. For an introduction to this mode, see here.

- `enable_spilling`

  Used to set whether to enable external sorting. The default is false, which turns off the feature. This feature is enabled when the user does not specify a LIMIT condition for the ORDER BY clause and also sets `enable_spilling` to true. When this feature is enabled, the temporary data is stored in the `doris-scratch/` directory of the BE data directory and the temporary data is cleared after the query is completed.

  This feature is mainly used for sorting operations with large amounts of data using limited memory.

  Note that this feature is experimental and does not guarantee stability. Please turn it on carefully.

- `exec_mem_limit`

  Used to set the memory limit for a single query. The default is 2GB, you can set it in B/K/KB/M/MB/G/GB/T/TB/P/PB, the default is B.

  This parameter is used to limit the memory that can be used by an instance of a single query fragment in a query plan. A query plan may have multiple instances, and a BE node may execute one or more instances. Therefore, this parameter does not accurately limit the memory usage of a query across the cluster, nor does it accurately limit the memory usage of a query on a single BE node. The specific needs need to be judged according to the generated query plan.

  Usually, only some blocking nodes (such as sorting node, aggregation node, and join node) consume more memory, while in other nodes (such as scan node), data is streamed and does not occupy much memory.

  When a `Memory Exceed Limit` error occurs, you can try to increase the parameter exponentially, such as 4G, 8G, 16G, and so on.

  It should be noted that this value may fluctuate by a few MB.

- `forward_to_master`

  The user sets whether to forward some commands to the Master FE node for execution. The default is `true`, which means forwarding. There are multiple FE nodes in Doris, one of which is the Master node. Usually users can connect to any FE node for full-featured operation. However, some of detail information can only be obtained from the Master FE node.

  For example, the `SHOW BACKENDS;` command, if not forwarded to the Master FE node, can only see some basic information such as whether the node is alive, and forwarded to the Master FE to obtain more detailed information including the node startup time and the last heartbeat time.

  The commands currently affected by this parameter are as follows:

  1. `SHOW FRONTEND;`

     Forward to Master to view the last heartbeat information.

2. `SHOW BACKENDS;`

   Forward to Master to view startup time, last heartbeat information, and disk capacity information.

3. `SHOW BROKERS;`

   Forward to Master to view the start time and last heartbeat information.

4. `SHOW TABLET;/ADMIN SHOW REPLICA DISTRIBUTION;/ADMIN SHOW REPLICA STATUS;`

   Forward to Master to view the tablet information stored in the Master FE metadata. Under normal circumstances, the tablet information in different FE metadata should be consistent. When a problem occurs, this method can be used to compare the difference between the current FE and Master FE metadata.

5. `SHOW PROC;`

   Forward to Master to view information about the relevant PROC stored in the Master FE metadata. Mainly used for metadata comparison.

- `init_connect`

  Used for compatibility with MySQL clients. No practical effect.

- `interactive_timeout`

  Used for compatibility with MySQL clients. No practical effect.

- `enable_profile`

  Used to set whether you need to view the profile of the query. The default is false, which means no profile is required.

  By default, the BE sends a profile to the FE for viewing errors only if an error occurs in the query. A successful query will not send a profile. Sending a profile will incur a certain amount of network overhead, which is detrimental to a high concurrent query scenario.

  When the user wants to analyze the profile of a query, the query can be sent after this variable is set to true. After the query is finished, you can view the profile on the web page of the currently connected FE:

  `fe_host:fe_http:port/query`

  It will display the most recent 100 queries which `enable_profile` is set to true.

- `language`

  Used for compatibility with MySQL clients. No practical effect.

- `license`

  Show Doris's license. No other effect.

- `lower_case_table_names`

  Used to control whether the user table name is case-sensitive.

A value of 0 makes the table name case-sensitive. The default is 0.

When the value is 1, the table name is case insensitive. Doris will convert the table name to lowercase when storing and querying.
The advantage is that any case of table name can be used in one statement. The following SQL is correct: `mysql> show tables; +-----------------+ | Tables_ in_testdb| +-----------------+ | cost | +-----------------+ mysql> select * from COST where COst.id < 100 order by cost.id;` The disadvantage is that the table name specified in the table creation statement cannot be obtained after table creation. The table name viewed by 'show tables' is lower case of the specified table name.

When the value is 2, the table name is case insensitive. Doris stores the table name specified in the table creation statement and converts it to lowercase for comparison during query.
The advantage is that the table name viewed by 'show tables' is the table name specified in the table creation statement;
The disadvantage is that only one case of table name can be used in the same statement. For example, the table name 'cost' can be used to query the 'cost' table: `mysql> select * from COST where COST.id < 100 order by COST.id;`

This variable is compatible with MySQL and must be configured at cluster initialization by specifying `lower_case_table_names=` in fe.conf. It cannot be modified by the `set` statement after cluster initialization is complete, nor can it be modified by restarting or upgrading the cluster.

The system view table names in information_schema are case-insensitive and behave as 2 when the value of `lower_case_table_names` is 0.

- `max_allowed_packet`

  For compatibility with JDBC connection pool C3P0. Has no real effect on Doris itself. If you encounter the error `Packet for query is too large (1,514,085 > 1,048,576). You can change this value on the server by setting the 'max_allowed_packet' variable.`, you can use `set GLOBAL max_allowed_packet = 1548576` to increase the value.

- `max_pushdown_conditions_per_column`

  For the specific meaning of this variable, please refer to the description of `max_pushdown_conditions_per_column` in BE Configuration. This variable is set to -1 by default, which means that the configuration value in `be.conf` is used. If the setting is greater than 0, the query in the current session will use the variable value, and ignore the configuration value in `be.conf`.

- `max_scan_key_num`

  For the specific meaning of this variable, please refer to the description of `doris_max_scan_key_num` in BE Configuration. This variable is set to -1 by default, which means that the configuration value in `be.conf` is used. If the setting is greater than 0, the query in the current session will use the variable value, and ignore the configuration value in `be.conf`.

- `net_buffer_length`

  Used for compatibility with MySQL clients. No practical effect.

- `net_read_timeout`

Used for compatibility with MySQL clients. No practical effect.

- `net_write_timeout`

  Used for compatibility with MySQL clients. No practical effect.

- `parallel_exchange_instance_num`

  Used to set the number of exchange nodes used by an upper node to receive data from the lower node in the execution plan. The default is -1, which means that the number of exchange nodes is equal to the number of execution instances of the lower nodes (default behavior). When the setting is greater than 0 and less than the number of execution instances of the lower node, the number of exchange nodes is equal to the set value.

  In a distributed query execution plan, the upper node usually has one or more exchange nodes for receiving data from the execution instances of the lower nodes on different BEs. Usually the number of exchange nodes is equal to the number of execution instances of the lower nodes.

  In some aggregate query scenarios, if the amount of data to be scanned at the bottom is large, but the amount of data after aggregation is small, you can try to modify this variable to a smaller value, which can reduce the resource overhead of such queries. Such as the scenario of aggregation query on the DUPLICATE KEY data model.

- `parallel_fragment_exec_instance_num`

  For the scan node, set its number of instances to execute on each BE node. The default is 1.

  A query plan typically produces a set of scan ranges, the range of data that needs to be scanned. These data are distributed across multiple BE nodes. A BE node will have one or more scan ranges. By default, a set of scan ranges for each BE node is processed by only one execution instance. When the machine resources are abundant, you can increase the variable and let more execution instances process a set of scan ranges at the same time, thus improving query efficiency.

  The number of scan instances determines the number of other execution nodes in the upper layer, such as aggregate nodes and join nodes. Therefore, it is equivalent to increasing the concurrency of the entire query plan execution. Modifying this parameter will help improve the efficiency of large queries, but larger values will consume more machine resources, such as CPU, memory, and disk IO.

- `query_cache_size`

  Used for compatibility with MySQL clients. No practical effect.

- `query_cache_type`

  Used for compatible JDBC connection pool C3P0. No practical effect.

- `query_timeout`

  Used to set the query timeout. This variable applies to all query statements in the current connection. Particularly, timeout of INSERT statements is recommended to be managed by the insert_timeout below. The default is 15 minutes, in seconds.

- `insert_timeout`

Used to set the insert timeout. This variable applies to INSERT statements particularly in the current connection, and is recommended to manage long-duration INSERT action. The default is 4 hours, in seconds. It will lose effect when query_timeout is greater than itself to make it compatible with the habits of older version users to use query_timeout to control the timeout of INSERT statements.

- `resource_group`

  Not used.

- `send_batch_parallelism`

  Used to set the default parallelism for sending batch when execute InsertStmt operation, if the value for parallelism exceed `max_send_batch_parallelism_per_job` in BE config, then the coordinator BE will use the value of `max_send_batch_parallelism_per_job`.

- `sql_mode`

  Used to specify SQL mode to accommodate certain SQL dialects. For the SQL mode, see here.

- `sql_safe_updates`

  Used for compatibility with MySQL clients. No practical effect.

- `sql_select_limit`

  Used to limit return rows of select stmt, including select clause of insert stmt.

- `system_time_zone`

  Set to the current system time zone when the cluster is initialised. It cannot be changed.

- `time_zone`

  Used to set the time zone for the current session. Defaults to the value of `system_time_zone`. It affects the results of certain time functions. For more information, see the time zone documentation.

- `tx_isolation`

  Used for compatibility with MySQL clients. No practical effect.

- `tx_read_only`

  Used for compatibility with MySQL clients. No practical effect.

- `transaction_read_only`

  Used for compatibility with MySQL clients. No practical effect.

- `transaction_isolation`

  Used for compatibility with MySQL clients. No practical effect.

- `version`

  Used for compatibility with MySQL clients. No practical effect.

- `performance_schema`

  Used for compatibility with MySQL JDBC 8.0.16 or later version. No practical effect.

- `version_comment`

  Used to display the version of Doris. Cannot be changed.

- `wait_timeout`

  The length of the connection used to set up an idle connection. When an idle connection does not interact with Doris for that length of time, Doris will actively disconnect the link. The default is 8 hours, in seconds.

- `default_rowset_type`

  Used for setting the default storage format of Backends storage engine. Valid options: alpha/beta

- `use_v2_rollup`

  Used to control the sql query to use segment v2 rollup index to get data. This variable is only used for validation when upgrading to segment v2 feature. Otherwise, not recommended to use.

- `rewrite_count_distinct_to_bitmap_hll`

  Whether to rewrite count distinct queries of bitmap and HLL types as bitmap_union_count and hll_union_agg.

- `prefer_join_method`

  When choosing the join method(broadcast join or shuffle join), if the broadcast join cost and shuffle join cost are equal, which join method should we prefer.

  Currently, the optional values for this variable are “broadcast” or “shuffle”.

- `allow_partition_column_nullable`

  Whether to allow the partition column to be NULL when creating the table. The default is true, which means NULL is allowed. false means the partition column must be defined as NOT NULL.

- `insert_visible_timeout_ms`

  When execute insert statement, doris will wait for the transaction to commit and visible after the import is completed. This parameter controls the timeout of waiting for transaction to be visible. The default value is 10000, and the minimum value is 1000.

- `enable_exchange_node_parallel_merge`

  In a sort query, when an upper level node receives the ordered data of the lower level node, it will sort the corresponding data on the exchange node to ensure that the final data is ordered. However, when a single thread merges multiple channels of data, if the amount of data is too large, it will lead to a single point of exchange node merge bottleneck.

Doris optimizes this part if there are too many data nodes in the lower layer. Exchange node will start multithreading for parallel merging to speed up the sorting process. This parameter is false by default, which means that exchange node does not adopt parallel merge sort to reduce the extra CPU and memory consumption.

- `extract_wide_range_expr`

  Used to control whether turn on the 'Wide Common Factors' rule. The value has two: true or false. On by default.

- `enable_fold_constant_by_be`

  Used to control the calculation method of constant folding. The default is `false`, that is, calculation is performed in `FE`; if it is set to `true`, it will be calculated by `BE` through `RPC` request.

- `cpu_resource_limit`

  Used to limit the resource overhead of a query. This is an experimental feature. The current implementation is to limit the number of scan threads for a query on a single node. The number of scan threads is limited, and the data returned from the bottom layer slows down, thereby limiting the overall computational resource overhead of the query. Assuming it is set to 2, a query can use up to 2 scan threads on a single node.

  This parameter will override the effect of `parallel_fragment_exec_instance_num`. That is, assuming that `parallel_fragment_exec_instance_num` is set to 4, and this parameter is set to 2. Then 4 execution instances on a single node will share up to 2 scanning threads.

  This parameter will be overridden by the `cpu_resource_limit` configuration in the user property.

  The default is -1, which means no limit.

- `disable_join_reorder`

  Used to turn off all automatic join reorder algorithms in the system. There are two values: true and false.It is closed by default, that is, the automatic join reorder algorithm of the system is adopted. After set to true, the system will close all automatic sorting algorithms, adopt the original SQL table order, and execute join

- `enable_infer_predicate`

  Used to control whether to perform predicate derivation. There are two values: true and false. It is turned off by default, that is, the system does not perform predicate derivation, and uses the original predicate to perform related operations. After it is set to true, predicate expansion is performed.

- `return_object_data_as_binary` Used to identify whether to return the bitmap/hll result in the select result. In the select into outfile statement, if the export file format is csv, the bimap/hll data will be base64-encoded, if it is the parquet file format, the data will be stored as a byte array. Below will be an example of Java, more examples can be found in samples.

```
try (Connection conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:9030/test?user=root");
            Statement stmt = conn.createStatement()
) {
    stmt.execute("set return_object_data_as_binary=true"); // IMPORTANT!!!
```

```
    ResultSet rs = stmt.executeQuery("select uids from t_bitmap");
    while(rs.next()){
        byte[] bytes = rs.getBytes(1);
        RoaringBitmap bitmap32 = new RoaringBitmap();
        switch(bytes[0]) {
            case 0: // for empty bitmap
                break;
            case 1: // for only 1 element in bitmap32
                bitmap32.add(ByteBuffer.wrap(bytes,1,bytes.length-1)
                        .order(ByteOrder.LITTLE_ENDIAN)
                        .getInt());
                break;
            case 2: // for more than 1 elements in bitmap32
                bitmap32.deserialize(ByteBuffer.wrap(bytes,1,bytes.length-1));
                break;
            // for more details, see https://github.com/apache/doris/tree/master/samples/read_bitmap
        }
    }
}
```

- `block_encryption_mode` The block encryption mode can be controlled by this parameter, the default value is empty. When empty, using the AES algorithm is equivalent to using `AES_128_ECB`, and when using the SM4 algorithm is equivalent to `SM4_128_ECB`.

Optional values: `AES_128_ECB, AES_192_ECB, AES_256_ECB, AES_128_CBC, AES_192_CBC, AES_256_CBC, AES_128_CFB, AES_192_CFB, AES_256_CFB, AES_128_CFB1, AES_192_CFB1, AES_256_CFB1, AES_128_CFB8, AES_192_CFB8, AES_256_CFB8, AES_128_CFB128, AES_192_CFB128, AES_256_CFB128, AES_128_CTR, AES_192_CTR, AES_256_CTR, AES_128_OFB, AES_192_OFB, AES_256_OFB, SM4_128_ECB, SM4_128_CBC, SM4_128_CFB128, SM4_128_OFB, SM4_128_CTR,`

- `enable_infer_predicate`

  Used to control whether predicate deduction is performed. There are two values: true and false. It is turned off by default, and the system does not perform predicate deduction, and uses the original predicate for related operations. When set to true, predicate expansion occurs.

- `trim_tailing_spaces_for_external_table_query`

  Used to control whether trim the tailing spaces while quering Hive external tables. The default is false.

- `skip_storage_engine_merge`

  For debugging purpose. In vectorized execution engine, in case of problems of reading data of Aggregate Key model and Unique Key model, setting value to `true` will read data as Duplicate Key model.

- `skip_delete_predicate`

  For debugging purpose. In vectorized execution engine, in case of problems of reading data, setting value to `true` will also read deleted data.

- `skip_delete_bitmap`

  For debugging purpose. In Unique Key MoW table, in case of problems of reading data, setting value to `true` will also read deleted data.

- `skip_missing_version`

In some scenarios, all replicas of tablet are having missing versions, and the tablet is unable to recover. This config can control the behavior of query. When it is opened, the query will ignore the visible version recorded in FE partition, use the replica version. If the replica on be has missing versions, the query will directly skip this missing version, and only return the data of the existing version, In addition, the query will always try to select the one with the highest lastSuccessVersion among all surviving BE replicas, so as to recover as much data as possible. You should only open it in the emergency scenarios mentioned above, only used for temporary recovery queries. Note that, this variable conflicts with the a variable, when the a variable is not -1, this variable will not work.

- `default_password_lifetime`

  Default password expiration time. The default value is 0, which means no expiration. The unit is days. This parameter is only enabled if the user's password expiration property has a value of DEFAULT. like:

  ```
  CREATE USER user1 IDENTIFIED BY "12345" PASSWORD_EXPIRE DEFAULT;
  ALTER USER user1 PASSWORD_EXPIRE DEFAULT;
  ```

- `password_history`

  The default number of historical passwords. The default value is 0, which means no limit. This parameter is enabled only when the user's password history attribute is the DEFAULT value. like:

  ```
  CREATE USER user1 IDENTIFIED BY "12345" PASSWORD_HISTORY DEFAULT;
  ALTER USER user1 PASSWORD_HISTORY DEFAULT;
  ```

- `validate_password_policy`

  Password strength verification policy. Defaults to `NONE` or `0`, i.e. no verification. Can be set to `STRONG` or `2`. When set to `STRONG` or `2`, when setting a password via the `ALTER USER` or `SET PASSWORD` commands, the password must contain any of "uppercase letters", "lowercase letters", "numbers" and "special characters". 3 items, and the length must be greater than or equal to 8. Special characters include: `~!@#$%^&*()_+|<>,.?/:;'[]{}"`.

- `group_concat_max_len`

  For compatible purpose. This variable has no effect, just enable some BI tools can query or set this session variable sucessfully.

- `rewrite_or_to_in_predicate_threshold`

  The default threshold of rewriting OR to IN. The default value is 2, which means that when there are 2 ORs, if they can be compact, they will be rewritten as IN predicate.

- `group_by_and_having_use_alias_first`

  Specifies whether group by and having clauses use column aliases rather than searching for column name in From clause. The default value is false.

- `enable_file_cache`

Set wether to use block file cache, default false. This variable takes effect only if the BE config enable_file_cache=true. The cache is not used when BE config enable_file_cache=false.

- `file_cache_base_path`

  Specify the storage path of the block file cache on BE, default 'random', and randomly select the storage path configured by BE.

- `enable_inverted_index_query`

  Set wether to use inverted index query, default true.

- `topn_opt_limit_threshold`

  Set threshold for limit of topn query (eg. SELECT * FROM t ORDER BY k LIMIT n). If n <= threshold, topn optimizations(runtime predicate pushdown, two phase result fetch and read order by key) will enable automatically, otherwise disable. Default value is 1024.

- `drop_table_if_ctas_failed`

  Controls whether create table as select deletes created tables when a insert error occurs, the default value is true.

- `show_user_default_role`

  Controls whether to show each user's implicit roles in the results of `show roles`. Default is false.

- `use_fix_replica`

  :::tip Tips This configuration is supported since the Apache Doris 1.2 version :::

  Use a fixed replica to query. replica starts with 0 and if use_fix_replica is 0, the smallest is used, if use_fix_replica is 1, the second smallest is used, and so on. The default value is -1, indicating that the function is disabled.

- `dry_run_query`

  If set to true, for query requests, the actual result set will no longer be returned, but only the number of rows, while for load and insert, the data is discarded by sink node, no writing happens. The default is false.

  This parameter can be used to avoid the time-consuming result set transmission when testing a large number of data sets, and focus on the time-consuming underlying query execution.

  ```
  mysql> select * from bigtable;
  +--------------+
  | ReturnedRows |
  +--------------+
  | 10000000     |
  +--------------+
  ```

- `enable_parquet_lazy_materialization`

  Controls whether to use lazy materialization technology in parquet reader. The default value is true.

- `enable_orc_lazy_materialization`

  Controls whether to use lazy materialization technology in orc reader. The default value is true.

- `enable_strong_consistency_read`

  Used to enable strong consistent reading. By default, Doris supports strong consistency within the same session, that is, changes to data within the same session are visible in real time. If you want strong consistent reads between sessions, set this variable to true.

- `truncate_char_or_varchar_columns`

  Whether to truncate char or varchar columns according to the table's schema. The default is false.

  Because the maximum length of the char or varchar column in the schema of the table is inconsistent with the schema in the underlying parquet or orc file. At this time, if the option is turned on, it will be truncated according to the maximum length in the schema of the table.

- `jdbc_clickhouse_query_final`

  Whether to add the final keyword when using the JDBC Catalog function to query ClickHouse,default is false.

  It is used for the ReplacingMergeTree table engine of ClickHouse to deduplicate queries.

- `enable_unique_key_partial_update`

> **Note**
>
> This field is supported since version 2.0.2.

Whether to enable partial columns update semantics for native insert into statement, default is false. Please note that the default value of the session variable `enable_insert_strict`, which controls whether the insert statement operates in strict mode, is true. In other words, the insert statement is in strict mode by default, and in this mode, updating non-existing keys in partial column updates is not allowed. Therefore, when using the insert statement for partial columns update and wishing to insert non-existing keys, you need to set `enable_unique_key_partial_update` to true and simultaneously set `enable_insert_strict` to false.

- `enable_cooldown_replica_affinity`

  Users can choose whether to use the cooled copy for scanning first, which defaults to true

Supplementary instructions on statement execution timeout control

- Means of control

  Currently doris supports timeout control through `variable` and `user property` two systems. Both include `qeury_timeout` and `insert_timeout`.

- Priority

The order of priority for timeout to take effect is: `session variable` > `user property` > `global variable` > `default value`

When a variable with a higher priority is not set, the value of the next priority is automatically adopted.

- Related semantics

  `query_timeout` is used to control the timeout of all statements, and `insert_timeout` is specifically used to control the timeout of the INSERT statement. When the INSERT statement is executed, the timeout time will take

  The maximum value of `query_timeout` and `insert_timeout`.

  `query_timeout` and `insert_timeout` in `user property` can only be specified by the ADMIN user for the target user, and its semantics is to change the default timeout time of the specified user, and it does not have `quota` semantics.

- Precautions

  The timeout set by `user property` needs to be triggered after the client reconnects.

SQL Mode

SQL Mode

The sql mode newly supported by Doris refers to the sql mode management mechanism of Mysql. Each client can set its own sql mode, and database administrators with Admin privileges can set the global sql mode. ### sql mode introduce

sql mode enables users to switch between different styles of sql syntax and data validation strictness, making Doris more compatible with other databases. For example, in some databases, the '||' symbol is a string concatenator, but in Doris it is equivalent to 'or', then users only need to use sql mode to switch to the style they want. Each client can set the sql mode, which is valid in the current session. Only users with Admin privileges can set the global sql mode. ### principle

The sql mode is stored in SessionVariables with a 64-bit Long type. Each bit of this address represents the enable/disable (1 means open, 0 means disable) state of a mode, as long as you know where each mode is. Bit, we can easily and quickly perform checksum operations on sql mode through bit operations.

Every time you query the sql mode, the Long type will be parsed and turned into a user-readable string. Similarly, the sql mode string sent by the user to the server will be parsed into a string that can be stored in SessionVariables. Long type.

The global sql mode that has been set will be persisted, so the operation on the global sql mode always only needs to be performed once, and the last global sql mode can be restored even after the program is restarted.

Operation method

1. Set sql mode

```
set global sql_mode = "DEFAULT"
set session sql_mode = "DEFAULT"
```

> Currently Doris's default sql mode is DEFAULT (but this will be changed in subsequent revisions soon). Setting global sql mode requires Admin privileges and will affect all clients connecting thereafter. Setting session sql mode will only affect the current dialog client, the default is session mode.

2. Query sql mode

```
select @@global.sql_mode
select @@session.sql_mode
```

> In addition to this way, you can also check the current sql mode by returning all session variables in the

> following way

```
show global variables
show session variables
```

mode is supported

1. PIPES_AS_CONCAT

In this mode, the ‘||’ symbol is a string concatenation symbol (same as the CONCAT() function), not a synonym for the ‘OR’ symbol. (e.g., `'a'||'b' = 'ab'`, `1||0 = '10'`) ### composite mode

(subsequent additions)

Nereids-the Brand New Planner

R&D background

Modern query optimizers face challenges such as more complex query statements and more diverse query scenarios. At the same time, users are more and more eager to obtain query results as soon as possible. The outdated architecture of the old optimizer is difficult to meet the needs of rapid iteration in the future. Based on this, we set out to develop a new query optimizer for modern architectures. While processing the query requests of the current Doris scene more efficiently, it provides better scalability and lays a good foundation for handling the more complex requirements that Doris will face in the future.

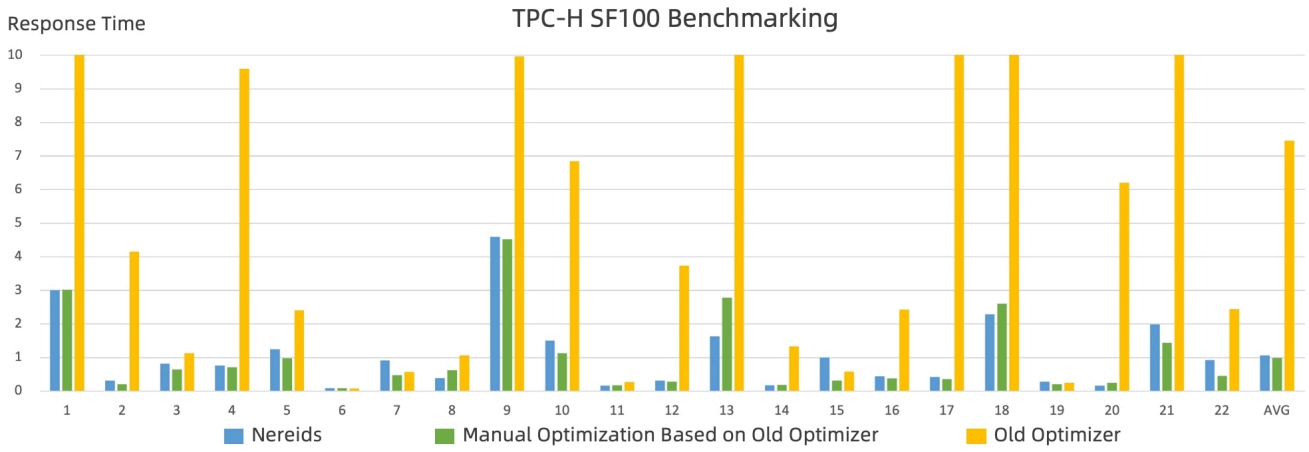
Advantages of the new optimizer

Smarter

The new optimizer presents the optimization points of each RBO and CBO in the form of rules. For each rule, the new optimizer provides a set of patterns used to describe the shape of the query plan, which can exactly match the query plan that can be optimized. Based on this, the new optimizer can better support more complex query statements such as multi-level subquery nesting.

At the same time, the CBO of the new optimizer is based on the advanced cascades framework, uses richer data statistics, and applies a cost model with more scientific dimensions. This makes the new optimizer more handy when faced with multi-table join queries.

TPC-H SF100 query speed comparison. The environment is 3BE, the new optimizer uses the original SQL, and the statistical information is collected before executing the SQL. Old optimizers use hand-tuned SQL. It can be seen that the new optimizer does not need to manually optimize the query, and the overall query time is similar to that of the old optimizer after manual optimization.



execution time comparison

More robust

All optimization rules of the new optimizer are completed on the logical execution plan tree. After the query syntax and semantic analysis is completed, it will be transformed into a tree structure. Compared with the internal data structure of the old optimizer, it is more reasonable and unified. Taking subquery processing as an example, the new optimizer is based on a new data structure, which avoids separate processing of subqueries by many rules in the old optimizer. In turn, the possibility of logic errors in optimization rules is reduced.

More flexible

The architectural design of the new optimizer is more reasonable and modern. Optimization rules and processing stages can be easily extended. Can more quickly respond to user needs.

How to use

Turn on Nereids

```
SET enable_nereids_planner=true;
```

Turn on auto fall back to legacy planner

```
SET enable_fallback_to_original_planner=true;
```

Recommand execute analyze on table before query on it to get the benefits of cbo

Known issues and temporarily unsupported features

Temporarily unsupported features

> **Note**
>
> If automatic fallback is enabled, it will automatically fall back to the old optimizer execution

- Json, Array, Map and Struct types: The table in the query contains the above types, or the expressions in the query outputs the above types
- DML: Only support below DML statements: Insert Into Select, Update and Delete
- Matrialized view with predicates
- Function alias
- Java UDF and HDFS UDF
- High concurrent point query optimize

Known issues

- Cannot use partition cache to accelarate query

Statistics

Statistics

Collecting statistics helps the optimizer understand data distribution characteristics. When performing Cost-Based Optimization (CBO), the optimizer uses these statistics to calculate the selectivity of predicates and estimate the cost of each execution plan. This allows for the selection of more optimal plans, significantly improving query efficiency.

Currently, the following information is collected for each column:

Information	Description
`row_count`	Total number of rows
`data_size`	Total data size
`avg_size_byte`	Average length of values
`ndv`	Number of distinct values
`min`	Minimum value
`max`	Maximum value
`null_count`	Number of null values

1. Collecting Statistics

1.1 Manual Collection Using ANALYZE Statement

Doris allows users to manually trigger the collection and update of statistics by submitting the ANALYZE statement.

Syntax:

```
ANALYZE < TABLE | DATABASE table_name | db_name >
    [ (column_name [, ...]) ]
    [ [ WITH SYNC ] [ WITH SAMPLE PERCENT | ROWS ] ];
```

Where:

- `table_name`: The specified target table. It can be in the format `db_name.table_name`.
- `column_name`: The specified target column. It must be an existing column in `table_name`. You can specify multiple column names separated by commas.
- `sync`: Collect statistics synchronously. Returns after collection. If not specified, it executes asynchronously and returns a JOB ID.
- `sample percent | rows`: Collect statistics with sampling. You can specify a sampling percentage or a number of sampling rows.

By default (WITH SAMPLE is not specified), a table will be fully analyzed. For relatively large tables (e.g. above 5GiB), from the perspective of system resources, we recommend sampling analyze, and the number of sampled rows is recommended to be no less than 4 million rows. Here are some examples:

Collect statistics for a table with full analyze:

```
ANALYZE TABLE lineitem;
```

Collect statistics for a table with a 10% sampling rate:

```
ANALYZE TABLE lineitem WITH SAMPLE PERCENT 10;
```

Collect statistics for a table with a sample of 100,000 rows:

```
ANALYZE TABLE lineitem WITH SAMPLE ROWS 100000;
```

1.2 Automatic Collection

This feature has been officially supported since 2.0.3 and is enabled by default. The basic operation logic is described below. After each import transaction commit, Doris records the number of rows updated by the import transaction to estimate the health of the existing table's statistics data (for tables that have not collected statistics, their health is 0). When the health of a table is below 60 (adjustable through the `table_stats_health_threshold` parameter), Doris considers the statistics for that table outdated and triggers statistics collection jobs for that table in subsequent operations. For tables with a health value above 60, no repeated collection is performed.

The collection jobs for statistics themselves consume a certain amount of system resources. To minimize the overhead, Doris automatically uses sampling to collect statistics. Automatic sampling defaults to sample 4,194,304 (2^22) rows to reduce the system's burden and complete the collection job as quickly as possible. If you want to sample more rows to obtain a more accurate data distribution, you can increase the sampling row count by adjusting the `huge_table_default_sample_rows` parameter. You can also control the full collection of small tables and the collection interval of large tables through session variables. For detailed configuration, please refer to 3.1.

If you are concerned about automatic collection jobs interfering with your business, you can specify a time frame for the automatic collection jobs to run during low business loads by setting the `auto_analyze_start_time` and `auto_analyze_end_time` parameters according to your needs. You can also completely disable this feature by setting the `enable_auto_analyze` parameter to `false`.

External catalogs do not participate in automatic collection by default. Because external catalogs often contain massive historical data, if they participate in automatic collection, it may occupy too many resources. You can turn on and off the automatic collection of external catalogs by setting the catalog's properties.

```
ALTER CATALOG external_catalog SET PROPERTIES ('enable.auto.analyze'='true'); // Turn on
ALTER CATALOG external_catalog SET PROPERTIES ('enable.auto.analyze'='false'); // Turn off
```

2. Job Management

2.1 View Analyze Jobs

Use `SHOW ANALYZE` to view information about statistics collection jobs.

Syntax:

```
SHOW [AUTO] ANALYZE < table_name | job_id >
    [ WHERE [ STATE = [ "PENDING" | "RUNNING" | "FINISHED" | "FAILED" ] ] ];
```

- AUTO: Show historical information for automatic collection jobs only. Note that, by default, the status of only the last 20,000 completed automatic collection jobs is retained.
- table_name: Table name, specify to view statistics job information for that table. It can be in the format `db_name.table_name`. When not specified, it returns information for all statistics jobs.
- job_id: Job ID for statistics collection, obtained when executing `ANALYZE`. When not specified, this command returns information for all statistics jobs.

Output:

Column Name	Description
job_id	Job ID
catalog_name	Catalog Name
db_name	Database Name
tbl_name	Table Name
col_name	Column Name List
job_type	Job Type
analysis_type	Analysis Type
message	Job Information
last_exec_time_in_ms	Last Execution Time
state	Job Status
schedule_type	Scheduling Method

Here's an example:

```
mysql> show analyze 245073\G;
*************************** 1. row ***************************
              job_id: 245073
        catalog_name: internal
             db_name: default_cluster:tpch
            tbl_name: lineitem
            col_name:
[l_returnflag,l_receiptdate,l_tax,l_shipmode,l_suppkey,l_shipdate,l_commitdate,l_partkey,l_orderkey,l_quantity,l_linestatus,l_commen
```

```
t,l_extendedprice,l_linenumber,l_discount,l_shipinstruct]
            job_type: MANUAL
       analysis_type: FUNDAMENTALS
             message:
last_exec_time_in_ms: 2023-11-07 11:00:52
               state: FINISHED
            progress: 16 Finished  |  0 Failed  |  0 In Progress  |  16 Total
       schedule_type: ONCE
```

2.2 View Column Statistics Collection Status

Each collection job can contain one or more tasks, with each task corresponding to the collection of a column. Users can use the following command to view the completion status of statistics collection for each column.

Syntax:

```
SHOW ANALYZE TASK STATUS [job_id]
```

Here's an example:

```
mysql> show analyze task status 20038 ;
+---------+----------+---------+----------------------+----------+
| task_id | col_name | message | last_exec_time_in_ms | state    |
+---------+----------+---------+----------------------+----------+
| 20039   | col4     |         | 2023-06-01 17:22:15  | FINISHED |
| 20040   | col2     |         | 2023-06-01 17:22:15  | FINISHED |
| 20041   | col3     |         | 2023-06-01 17:22:15  | FINISHED |
| 20042   | col1     |         | 2023-06-01 17:22:15  | FINISHED |
+---------+----------+---------+----------------------+----------+
```

2.3 View Column Statistics

Use `SHOW COLUMN STATS` to view various statistics data for columns.

Syntax:

```
SHOW COLUMN [cached] STATS table_name [ (column_name [, ...]) ];
```

Where:

- cached: Show statistics information in the current FE memory cache.
- table_name: The target table for collecting statistics. It can be in the format `db_name.table_name`.
- column_name: Specifies the target column, which must be an existing column in `table_name`. You can specify multiple column names separated by commas.

Here's an example:

```
mysql> show column stats lineitem(l_tax)\G;
*************************** 1. row ***************************
```

```
  column_name: l_tax
        count: 6001215.0
          ndv: 9.0
     num_null: 0.0
    data_size: 4.800972E7
avg_size_byte: 8.0
          min: 0.00
          max: 0.08
       method: FULL
         type: FUNDAMENTALS
      trigger: MANUAL
  query_times: 0
 updated_time: 2023-11-07 11:00:46
```

2.4 Table Collection Overview

Use `SHOW TABLE STATS` to view an overview of statistics collection for a table.

Syntax:

```
SHOW TABLE STATS table_name;
```

Where:

- table_name: The target table name. It can be in the format `db_name.table_name`.

Output:

Column Name	Description
updated_rows	Updated rows since the last ANALYZE
query_times	Reserved column for recording the number of times the table was queried in future versions
row_count	Number of rows (does not reflect the exact number of rows at the time of command execution)
updated_time	Last update time
columns	Columns for which statistics information has been collected

Here's an example:

```
mysql> show table stats lineitem \G;
*************************** 1. row ***************************
updated_rows: 0
 query_times: 0
   row_count: 6001215
updated_time: 2023-11-07
     columns: [l_returnflag, l_receiptdate, l_tax, l_shipmode, l_suppkey, l_shipdate, l_commitdate, l_partkey, l_orderkey,
l_quantity, l_linestatus, l_comment, l_extendedprice, l_linenumber, l_discount, l_shipinstruct]
     trigger: MANUAL
```

2.5 Terminate Statistics Jobs

Use `KILL ANALYZE` to terminate running statistics jobs.

Syntax:

```
KILL ANALYZE job_id;
```

Where:

- job_id: Job ID for statistics collection. Obtained when performing asynchronous statistics collection using the `ANALYZE` statement, and it can also be obtained through the `SHOW ANALYZE` statement.

Example:

- Terminate statistics job with ID 52357.

```
mysql> KILL ANALYZE 52357;
```

3. Session Variables and Configuration Options

3.1 Session Variables

Session Variable	Description	Default Value
auto_analyze_start_time	Start time for automatic statistics collection	00:00:00
auto_analyze_end_time	End time for automatic statistics collection	23:59:59
enable_auto_analyze	Enable automatic collection functionality	true
huge_table_default_sample_rows	Sampling rows for large tables	4194304
huge_table_lower_bound_size_in_bytes	Tables with size greater than this value will be automatically sampled during collection of statistics	0
huge_table_auto_analyze_interval_in_millis	Controls the minimum time interval for automatic ANALYZE on large tables. Tables with sizes greater than `huge_table_lower_bound_size_in_bytes * 5` will be ANALYZEed only once within this time interval.	0
table_stats_health_threshold	Ranges from 0 to 100. If data updates since the last statistics collection exceed `(100 - table_stats_health_threshold)%`, the table's statistics are considered outdated.	60

Session Variable	Description	Default Value
analyze_timeout	Controls the timeout for synchronous ANALYZE in seconds	43200
auto_analyze_table_width_threshold	Controls the maximum width of table that will be auto analyzed. Table with more columns than this value will not be auto analyzed.	100

3.2 FE Configuration Options

The following FE configuration options are typically not a major concern:

FE Configuration Option	Description	Default Value
analyze_record_limit	Controls the persistence of statistics job execution records	20000
stats_cache_size	FE-side statistics cache entries	500,000
statistics_simultaneously_running_task_num	Number of asynchronous jobs that can run simultaneously	3
statistics_sql_mem_limit_in_bytes	Controls the amount of BE memory each statistics SQL can use	2,147,483,648 bytes (2 GiB)

4. Common Issues

4.1 ANALYZE Submission Error: Stats table not available…

When ANALYZE is executed, statistics data is written to the internal table `__internal_schema.column_statistics`. FE checks the tablet status of this table before executing ANALYZE. If there are unavailable tablets, the task is rejected. Please check the BE cluster status if this error occurs.

Users can use `SHOW BACKENDS\G` to verify the BE (Backend) status. If the BE status is normal, you can use the command `ADMIN SHOW REPLICA STATUS FROM __internal_schema.[tbl_in_this_db]` to check the tablet status within this database, ensuring that the tablet status is also normal.

4.2 Failure of ANALYZE on Large Tables

Due to resource limitations, ANALYZE on some large tables may timeout or exceed BE memory limits. In such cases, it is recommended to use `ANALYZE ... WITH SAMPLE...`.

Pipeline Execution Engine

> **Note**
>
> This feature is supported since version 2.0.0.

Pipeline execution engine is an experimental feature added by Doris in version 2.0. The goal is to replace the current execution engine of Doris's volcano model, fully release the computing power of multi-core CPUs, and limit the number of Doris's query threads to solve the problem of Doris's execution thread bloat.

Its specific design, implementation and effects can be found in DSIP-027。

Principle

The current Doris SQL execution engine is designed based on the traditional volcano model, which has the following problems in a single multi-core scenario: * Inability to take full advantage of multi-core computing power to improve query performance,**most scenarios require manual setting of parallelism** for performance tuning, which is almost difficult to set in production environments.

- Each instance of a standalone query corresponds to one thread of the thread pool, which introduces two additional problems.
    - Once the thread pool is hit full. **Doris' query engine will enter a pseudo-deadlock** and will not respond to subsequent queries. **At the same time there is a certain probability of entering a logical deadlock** situation: for example, all threads are executing an instance's probe task.
    - Blocking arithmetic will take up thread resources,**blocking thread resources can not be yielded to instances that can be scheduled**, the overall resource utilization does not go up.
- Blocking arithmetic relies on the OS thread scheduling mechanism, **thread switching overhead (especially in the scenario of system mixing))**

The resulting set of problems drove Doris to implement an execution engine adapted to the architecture of modern multi-core CPUs.

And as shown in the figure below (quoted fromPush versus pull-based loop fusion in query engines)， The resulting set of problems drove Doris to implement an execution engine adapted to the architecture of modern multi-core CPUs.：

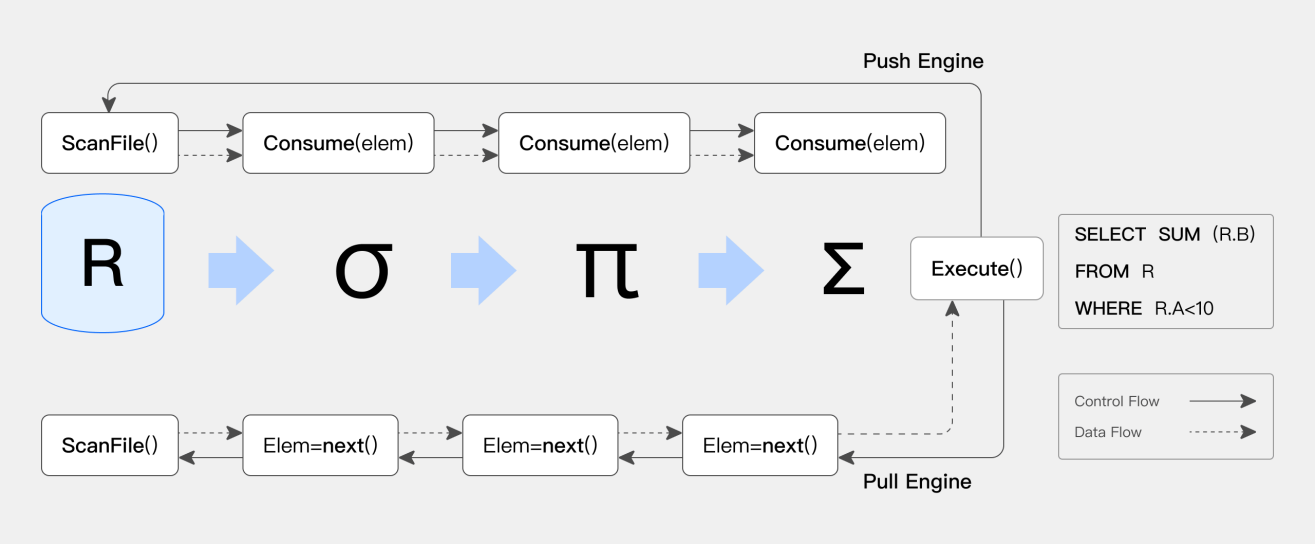


image.png

1. Transformation of the traditional pull pull logic-driven execution process into a data-driven execution engine for the push model
2. Blocking operations are asynchronous, reducing the execution overhead caused by thread switching and thread blocking and making more efficient use of the CPU
3. Controls the number of threads to be executed and reduces the resource congestion of large queries on small queries in mixed load scenarios by controlling time slice switching

This improves the efficiency of CPU execution on mixed-load SQL and enhances the performance of SQL queries.

Usage

Query

1. enable_pipeline_engine

Setting the session variable `enable_pipeline_engine` to `true` will make BE to use the Pipeline execution engine when performing query execution.

```
set enable_pipeline_engine = true;
```

2. parallel_pipeline_task_num

The `parallel_pipeline_task_num` represents the number of Pipeline Tasks for a SQL query to be queried concurrently.The default configuration of Doris is `0`, in which case the number of Pipeline Tasks will be automatically set to half of the minimum number of CPUs in the current cluster machine. You can also adjust it according to your own situation.

```
set parallel_pipeline_task_num = 0;
```

You can limit the automatically configured concurrency by setting `max_instance_num`（The default value is 64)

Load

The engine selected for import are detailed in the Import documentation.

Query Caches Overview

Demand scenario

Most data analysis scenarios are to write less and read more. The data is written once and read multiple times frequently. For example, the dimensions and indicators involved in a report are calculated once in the early morning, but hundreds or even thousands of times a day. Page access, so it is very suitable for caching the result set. In data analysis or BI applications, the following business scenarios exist:

- **High concurrency scenario**, Doris can better support high concurrency, but a single server cannot carry too high QPS
- **Complex Chart Kanban**, complex Dashboard or large-screen application, data comes from multiple tables, and each page has dozens of queries. Although each query only takes tens of milliseconds, the overall query time will be several seconds.
- **Trend Analysis**, for queries within a given date range, indicators are displayed on a daily basis, such as querying the trend of the number of users in the last 7 days. This type of query has a large amount of data, a wide query range, and the query time often takes tens of seconds.
- **User repeated query**, if the product does not have an anti-refresh mechanism, the user repeatedly refreshes the page due to manual error or other reasons, resulting in a large number of repeated SQL submissions.

In the above four scenarios, the solution at the application layer puts the query results into Redis and periodically updates the cache or the user manually refreshes the cache. However, this solution has the following problems:

- **Inconsistent data**, unable to detect data updates, causing users to often see old data
- **Low hit rate**, the entire query result is cached. If the data is written in real time, the cache fails frequently, the hit rate is low and the system load is heavy.
- **Additional Cost**, introducing external cache components will bring system complexity and increase additional costs.

solution

This partition cache strategy can solve the above problems, giving priority to ensuring data consistency, and on this basis, refining the cache granularity and improving the hit rate, so it has the following characteristics:

- Users do not need to worry about data consistency. Cache invalidation is controlled through versioning. The cached data is consistent with the data queried from BE.
- There are no additional components and costs, the cache results are stored in BE's memory, and users can adjust the cache memory size as needed
- Implemented two caching strategies, SQLCache and PartitionCache, the latter has a finer cache granularity
- Use consistent hashing to solve the problem of BE nodes going online and offline. The caching algorithm in BE is an improved LRU

scenes to be used

Currently, it supports two methods: SQL Cache and Partition Cache, and supports OlapTable internal table and Hive external table.

SQL Cache: Only SQL statements that are completely consistent will hit the cache. For details, see: sql-cache-manual.md

Partition Cache: Multiple SQLs can hit the cache using the same table partition, so it has a higher hit rate than SQL Cache. For details, see: partition-cache-manual.md

Monitoring

FE monitoring items:

```
query_table //The number of tables in Query
query_olap_table //The number of Olap tables in Query
cache_mode_sql //Identify the number of Query whose cache mode is sql
cache_hit_sql //The number of Query hits in Cache with mode sql
query_mode_partition //The number of queries that identify the cache mode as Partition
cache_hit_partition //The number of Query hits through Partition
partition_all //All partitions scanned in Query
partition_hit //Number of partitions hit through Cache

Cache hit rate = (cache_hit_sql + cache_hit_partition) / query_olap_table
Partition hit rate = partition_hit / partition_all
```

BE monitoring items:

```
query_cache_memory_total_byte //Cache memory size
query_query_cache_sql_total_count //The number of SQL cached
query_cache_partition_total_count //Number of Cache partitions

SQL average data size = cache_memory_total / cache_sql_total
Partition average data size = cache_memory_total / cache_partition_total
```

Other monitoring: You can view the CPU and memory indicators of the BE node, Query Percentile and other indicators in the Query statistics from Grafana, and adjust the Cache parameters to achieve business goals.

Related parameters

1. cache_result_max_row_count

The maximum number of rows that the query result set can put into the cache. The default is 3000.

```
vim fe/conf/fe.conf
cache_result_max_row_count=3000
```

2. cache_result_max_data_size

The maximum data size of the query result set placed in the cache is 30M by default. It can be adjusted according to the actual situation, but it is recommended not to set it too large to avoid excessive memory

usage. Result sets exceeding this size will not be cached.

```
vim fe/conf/fe.conf
cache_result_max_data_size=31457280
```

3. cache_last_version_interval_second

The minimum time interval between the latest version of the cached query partition and the current version. Only the query results of partitions that are larger than this interval and have not been updated will be cached. The default is 30, in seconds.

```
vim fe/conf/fe.conf
cache_last_version_interval_second=30
```

4. query_cache_max_size_mb and query_cache_elasticity_size

query_cache_max_size_mb is the upper memory limit of the cache, query_cache_elasticity_size is the memory size that the cache can stretch. When the total cache size on BE exceeds query_cache_max_size + cache_elasticity_size, it will start to be cleaned up and the memory will be controlled below query_cache_max_size.

These two parameters can be set according to the number of BE nodes, node memory size, and cache hit rate. Calculation method: If 10,000 Queries are cached, each Query caches 1,000 rows, each row is 128 bytes, and is distributed on 10 BEs, then each BE requires about 128M memory (10,000 * 1,000 * 128/10).

```
vim be/conf/be.conf
query_cache_max_size_mb=256
query_cache_elasticity_size_mb=128
```

5. cache_max_partition_count

Parameters unique to Partition Cache. The maximum number of BE partitions refers to the maximum number of partitions corresponding to each SQL. If it is partitioned by date, it can cache data for more than 2 years. If you want to keep the cache for a longer time, please set this parameter larger and modify the parameters at the same time. cache_result_max_row_count and cache_result_max_data_size.

```
vim be/conf/be.conf
cache_max_partition_count=1024
```

SQL Cache

SQL Cache

The SQL statement will hit the cache if it is completely consistent.

Demand scenarios & solutions

See Query Caches Overview

Design principles

SQLCache stores and obtains the cache based on the SQL signature, the partition ID of the queried table, and the latest version of the partition. The combination of the three determines a cached data set. If any one of them changes, such as the SQL changes, the query fields or conditions are different, or the version changes after the data is updated, the cache will not be hit.

If multiple tables are joined, the most recently updated partition ID and latest version number are used. If one of the tables is updated, the partition ID or version number will be different, and the cache will not be hit.

SQLCache is more suitable for T+1 update scenarios. Data is updated in the early morning. The first query obtains the results from BE and puts them into the cache. Subsequent queries of the same nature obtain the results from the cache. Real-time update data can also be used, but there may be a problem of low hit rate.

Currently supports OlapTable internal table and Hive external table.

Usage

Make sure cache_enable_sql_mode=true in fe.conf (default is true)

```
vim fe/conf/fe.conf
cache_enable_sql_mode=true
```

Set variables in MySQL command line

```
MySQL [(none)]> set [global] enable_sql_cache=true;
```

Note: global is a global variable and does not refer to the current session variable.

Cache conditions

After the first query, if the following three conditions are met, the query results will be cached.

1. (Current time - the last update time of the queried partition) is greater than cache_last_version_interval_second in fe.conf.

2. The number of query result rows is less than cache_result_max_row_count in fe.conf.

3. The query result bytes is less than cache_result_max_data_size in fe.conf.

For detailed parameter introduction and unfinished matters, see query-cache.md.

Unfinished business

- SQL contains functions that generate random values, such as random(). Using QueryCache will cause the query results to lose their randomness, and the same results will be obtained every time they are executed.
- Similar SQL, 2 indicators were queried before, and now 3 indicators are queried. Can the cache of 2 indicators be used? Not currently supported

Partition Cache

Cache hits can occur when multiple SQLs use the same table partition.

> **Caution**
>
> **Partition Cache is an experimental feature and is not well maintained, this feature will be removed in version 2.1. Please use it with caution**

Demand scenarios & solutions

See Query Caches Overview

Design principles

1. SQL can be split in parallel, Q = Q1 ∪ Q2 … ∪ Qn, R= R1 ∪ R2 … ∪ Rn, Q is the query statement, R is the result set
2. SQL only uses DATE, INT, and BIGINT types of partition field aggregation, and only scans one partition. Therefore, it does not support partitioning by day, but only supports partitioning by year and month.
3. Cache the results of some dates in the query result set, and then reduce the date range scanned in SQL. In essence, PartitionCache does not reduce the number of partitions scanned, but also reduces the date range scanned, thereby reducing the amount of scanned data.

In addition some restrictions:

- Only supports grouping by partition fields, not by other fields. Grouping by other fields may cause the group data to be updated, which will cause the cache to become invalid.
- Only the first half, the second half and all hits of the result set are supported in the cache. The result set is not supported to be divided into several parts by cached data, and the dates of the result set must be continuous. If there is no data in the result set on a certain day, then only this Dates one day older will be cached.
- If the predicate has columns outside the partition, you must add brackets to the partition predicate `where k1 = 1 and (key >= "2023-10-18" and key <= "2021-12-01")`
- The number of days in the query must be greater than 1 and less than cache_result_max_row_count, otherwise the partition cache cannot be used.
- The predicate of the partition field can only be key >= a and key <= b or key = a or key = b or key in (a,b,c).

Usage

Make sure cache_enable_partition_mode=true in fe.conf (default is true)

```
vim fe/conf/fe.conf
cache_enable_partition_mode=true
```

Set variables in MySQL command line

```
MySQL [(none)]> set [global] enable_partition_cache=true;
```

If two caching strategies are enabled at the same time, you need to pay attention to the following parameters:

```
cache_last_version_interval_second=30
```

If the interval between the latest version of the partition and the present is greater than cache_last_version_interval_second, the entire query result will be cached first. If it is less than this interval, if it meets the conditions of PartitionCache, the PartitionCache data will be pressed.

For detailed parameter introduction and unfinished matters, see query-cache.md.

Unfinished business

Split into read-only partitions and updateable partitions, read-only partitions are cached, update partitions are not cached

As above, query the number of daily users in the last 7 days. For example, if partitioned by date, the data will only be written to the partition of the current day. The data of other partitions other than the current day are fixed. Under the same query SQL, query a certain area that is not updated. The partition indicators are all fixed. As follows, the number of users in the previous 7 days is queried on 2020-03-09. The data from 2020-03-03 to 2020-03-07 comes from the cache. The first query on 2020-03-08 comes from the partition, and subsequent queries come from the cache. , 2020-03-09 because it was written continuously that day, so it comes from the partition.

Therefore, to query N days of data, the data is updated for the most recent D days. Similar queries with different date ranges every day only need to query D partitions. The other parts come from the cache, which can effectively reduce the cluster load and query time.

Implementation principle example:

```
MySQL [(none)]> SELECT eventdate,count(userid) FROM testdb.appevent WHERE eventdate>="2020-03-03" AND eventdate<="2020-03-09" GROUP
BY eventdate ORDER BY eventdate;
+----------------+-----------------+
| eventdate | count(`userid`) |
+----------------+-----------------+
| 2020-03-03 | 15 |
| 2020-03-04 | 20 |
| 2020-03-05 | 25 |
| 2020-03-06 | 30 |
| 2020-03-07 | 35 |
| 2020-03-08 | 40 | //The first time comes from the partition, and the subsequent ones come from the cache
```

```
| 2020-03-09 | 25 | //From partition
+----------------+-----------------+
7 rows in set (0.02 sec)
```

In PartitionCache, the cached first-level Key is the 128-bit MD5 signature of the SQL after removing the partition conditions. The following is the rewritten SQL to be signed:

```
SELECT eventdate,count(userid) FROM testdb.appevent GROUP BY eventdate ORDER BY eventdate;
```

The cached second-level key is the content of the partition field of the query result set, such as the content of the eventdate column of the query result above. The ancillary information of the second-level key is the version number and version update time of the partition.

The following demonstrates the process of executing the above SQL for the first time on 2020-03-09:

1. Get data from cache

```
+----------------+-----------------+
| 2020-03-03 | 15 |
| 2020-03-04 | 20 |
| 2020-03-05 | 25 |
| 2020-03-06 | 30 |
| 2020-03-07 | 35 |
+----------------+-----------------+
```

2. SQL and data to get data from BE

```
SELECT eventdate,count(userid) FROM testdb.appevent WHERE eventdate>="2020-03-08" AND eventdate<="2020-03-09" GROUP BY eventdate
ORDER BY eventdate;

+----------------+-----------------+
| 2020-03-08 | 40 |
+----------------+-----------------+
| 2020-03-09 | 25 |
+----------------+-----------------+
```

3. The last data sent to the terminal

```
+----------------+-----------------+
| eventdate | count(`userid`) |
+----------------+-----------------+
| 2020-03-03 | 15 |
| 2020-03-04 | 20 |
| 2020-03-05 | 25 |
| 2020-03-06 | 30 |
| 2020-03-07 | 35 |
| 2020-03-08 | 40 |
| 2020-03-09 | 25 |
+----------------+-----------------+
```

4. Data sent to cache

```
+----------------+-----------------+
| 2020-03-08 | 40 |
```

```
+----------------+-----------------+
```

Partition cache is suitable for partitioning by date, some partitions are updated in real time, and the query SQL is relatively fixed.

The partition field can also be other fields, but it needs to be ensured that only a small number of partitions are updated.

View

View

Views (logical views) are stored queries that encapsulate one or multiple SELECT statements. Views dynamically access and compute database data when executed. Views are read-only and can reference any combination of tables and other views.

Views can be used for the following purposes:

- To simplify access or provide secure access by hiding complex SELECT statements from users. For example, you can create a view that displays only the data users need from various tables while hiding sensitive data in those tables.
- To encapsulate details of table structures that may change over time behind a consistent user interface.

Unlike materialized views, logical views are not materialized, which means they do not store data on disk. Therefore, they have the following limitations:

- When the underlying table data changes, Doris does not need to refresh the view data. However, accessing and computing data through views can incur some overhead.
- Views do not support insert, delete, or update operations.

Creating View

The syntax for creating a logical view is as follows:

```
CREATE VIEW [IF NOT EXISTS]
 [db_name.]view_name
 (column1[ COMMENT "col comment"][, column2, ...])
AS query_stmt
```

Explanation:

- Views are logical and have no physical storage. All queries on views are equivalent to queries on the corresponding subquery of the view.
- query_stmt is any supported SQL statement.

Example:

- Creating a view named `example_view` in the `example_db` database:

```
CREATE VIEW example_db.example_view (k1, k2, k3, v1)
AS
SELECT c1 as k1, k2, k3, SUM(v1) FROM example_table
WHERE k1 = 20160112 GROUP BY k1,k2,k3;
```

- Creating a view with comments:

```
CREATE VIEW example_db.example_view
(
    k1 COMMENT "first key",
    k2 COMMENT "second key",
    k3 COMMENT "third key",
    v1 COMMENT "first value"
)
COMMENT "my first view"
AS
SELECT c1 as k1, k2, k3, SUM(v1) FROM example_table
WHERE k1 = 20160112 GROUP BY k1,k2,k3;
```

Materialized View

Materialized View

A materialized view is a data set that is pre-calculated (according to a defined SELECT statement) and stored in a special table in Doris.

The emergence of materialized views is mainly to satisfy users. It can analyze any dimension of the original detailed data, but also can quickly analyze and query fixed dimensions.

When to use materialized view

- Analyze requirements to cover both detailed data query and fixed-dimensional query.
- The query only involves a small part of the columns or rows in the table.
- The query contains some time-consuming processing operations, such as long-time aggregation operations.
- The query needs to match different prefix indexes.

Advantage

- For those queries that frequently use the same sub-query results repeatedly, the performance is greatly improved
- Doris automatically maintains the data of the materialized view, whether it is a new import or delete operation, it can ensure the data consistency of the base table and the materialized view table. No need for any additional labor maintenance costs.
- When querying, it will automatically match the optimal materialized view and read data directly from the materialized view.

Automatic maintenance of materialized view data will cause some maintenance overhead, which will be explained in the limitations of materialized views later.

Materialized View VS Rollup

Before the materialized view function, users generally used the Rollup function to improve query efficiency through pre-aggregation. However, Rollup has certain limitations. It cannot do pre-aggregation based on the detailed model.

Materialized views cover the functions of Rollup while also supporting richer aggregate functions. So the materialized view is actually a superset of Rollup.

In other words, the functions previously supported by the `ALTER TABLE ADD ROLLUP` syntax can now be implemented by `CREATE MATERIALIZED VIEW`.

Use materialized views

The Doris system provides a complete set of DDL syntax for materialized views, including creating, viewing, and deleting. The syntax of DDL is consistent with PostgreSQL and Oracle.

Create a materialized view

Here you must first decide what kind of materialized view to create based on the characteristics of your query statement. This is not to say that your materialized view definition is exactly the same as one of your query statements. There are two principles here:

1. **Abstract** from the query statement, the grouping and aggregation methods shared by multiple queries are used as the definition of the materialized view.
2. It is not necessary to create materialized views for all dimension combinations.

First of all, the first point, if a materialized view is abstracted, and multiple queries can be matched to this materialized view. This materialized view works best. Because the maintenance of the materialized view itself also consumes resources.

If the materialized view only fits a particular query, and other queries do not use this materialized view. As a result, the materialized view is not cost-effective, which not only occupies the storage resources of the cluster, but cannot serve more queries.

Therefore, users need to combine their own query statements and data dimension information to abstract the definition of some materialized views.

The second point is that in the actual analysis query, not all dimensional analysis will be covered. Therefore, it is enough to create a materialized view for the commonly used combination of dimensions, so as to achieve a space and time balance.

Creating a materialized view is an asynchronous operation, which means that after the user successfully submits the creation task, Doris will calculate the existing data in the background until the creation is successful.

The specific syntax can be viewed through the following command:

```
HELP CREATE MATERIALIZED VIEW
```

> **Tips**
>
> This feature is supported since the Apache Doris 2.0 version

In `Doris 2.0` we made some enhancements to materialized views (described in `Best Practice 4` of this article). We recommend that users check whether the expected query can hit the desired materialized view in the test environment before using the materialized view in the official production environment.

If you don’t know how to verify that a query hits a materialized view, you can read `Best Practice 1` of this article.

At the same time, we do not recommend that users create multiple materialized views with similar shapes on the same table, which may cause conflicts between multiple materialized views and cause query hit failures (this problem will be improved in the new optimizer). It is recommended that users first verify whether materialized views and queries meet the requirements and can be used normally in the test environment. #### Support aggregate functions

The aggregate functions currently supported by the materialized view function are:

- SUM, MIN, MAX (Version 0.12)
- COUNT, BITMAP_UNION, HLL_UNION (Version 0.13)

Update strategy

In order to ensure the data consistency between the materialized view table and the Base table, Doris will import, delete and other operations on the Base table are synchronized to the materialized view table. And through incremental update to improve update efficiency. To ensure atomicity through transaction.

For example, if the user inserts data into the base table through the INSERT command, this data will be inserted into the materialized view synchronously. When both the base table and the materialized view table are written successfully, the INSERT command will return successfully.

Query automatic matching

After the materialized view is successfully created, the user's query does not need to be changed, that is, it is still the base table of the query. Doris will automatically select an optimal materialized view based on the current query statement, read data from the materialized view and calculate it.

Users can use the EXPLAIN command to check whether the current query uses a materialized view.

The matching relationship between the aggregation in the materialized view and the aggregation in the query:

Materialized View Aggregation	Query Aggregation
sum	sum
min	min
max	max
count	count
bitmap_union	bitmap_union, bitmap_union_count, count(distinct)
hll_union	hll_raw_agg, hll_union_agg, ndv, approx_count_distinct

After the aggregation functions of bitmap and hll match the materialized view in the query, the aggregation operator of the query will be rewritten according to the table structure of the materialized view. See example 2 for details.

Query materialized views

Check what materialized views the current table has, and what their table structure is. Through the following command:

```
MySQL [test]> desc mv_test all;
+-----------+---------------+-----------------+----------+------+-------+---------+--------------+
| IndexName | IndexKeysType | Field           | Type     | Null | Key   | Default | Extra        |
+-----------+---------------+-----------------+----------+------+-------+---------+--------------+
| mv_test   | DUP_KEYS      | k1              | INT      | Yes  | true  | NULL    |              |
|           |               | k2              | BIGINT   | Yes  | true  | NULL    |              |
|           |               | k3              | LARGEINT | Yes  | true  | NULL    |              |
|           |               | k4              | SMALLINT | Yes  | false | NULL    | NONE         |
|           |               |                 |          |      |       |         |              |
| mv_2      | AGG_KEYS      | k2              | BIGINT   | Yes  | true  | NULL    |              |
|           |               | k4              | SMALLINT | Yes  | false | NULL    | MIN          |
|           |               | k1              | INT      | Yes  | false | NULL    | MAX          |
|           |               |                 |          |      |       |         |              |
| mv_3      | AGG_KEYS      | k1              | INT      | Yes  | true  | NULL    |              |
|           |               | to_bitmap(`k2`) | BITMAP   | No   | false |         | BITMAP_UNION |
|           |               |                 |          |      |       |         |              |
| mv_1      | AGG_KEYS      | k4              | SMALLINT | Yes  | true  | NULL    |              |
|           |               | k1              | BIGINT   | Yes  | false | NULL    | SUM          |
|           |               | k3              | LARGEINT | Yes  | false | NULL    | SUM          |
|           |               | k2              | BIGINT   | Yes  | false | NULL    | MIN          |
+-----------+---------------+-----------------+----------+------+-------+---------+--------------+
```

You can see that the current `mv_test` table has three materialized views: mv_1, mv_2 and mv_3, and their table structure.

Delete materialized view

If the user no longer needs the materialized view, you can delete the materialized view by ‘DROP’ commen.

You can view the specific syntaxSHOW CREATE MATERIALIZED VIEW

View the materialized view that has been created

Users can view the created materialized views by using commands

You can view the specific syntaxSHOW CREATE MATERIALIZED VIEW

Cancel Create materialized view

```
CANCEL ALTER TABLE MATERIALIZED VIEW FROM db_name.table_name
```

Best Practice 1

The use of materialized views is generally divided into the following steps:

1. Create a materialized view
2. Asynchronously check whether the materialized view has been constructed
3. Query and automatically match materialized views

First is the first step: Create a materialized view

Assume that the user has a sales record list, which stores the transaction id, salesperson, sales store, sales time, and amount of each transaction. The table building statement and insert data statement is:

```
create table sales_records(record_id int, seller_id int, store_id int, sale_date date, sale_amt bigint) distributed by
hash(record_id) properties("replication_num" = "1");
insert into sales_records values(1,1,1,"2020-02-02",1);
```

The table structure of this `sales_records` is as follows:

```
MySQL [test]> desc sales_records;
+-----------+--------+------+-------+---------+--- ----+
| Field | Type | Null | Key | Default | Extra |
+-----------+--------+------+-------+---------+--- ----+
| record_id | INT | Yes | true | NULL | |
| seller_id | INT | Yes | true | NULL | |
| store_id | INT | Yes | true | NULL | |
| sale_date | DATE | Yes | false | NULL | NONE |
| sale_amt | BIGINT | Yes | false | NULL | NONE |
+-----------+--------+------+-------+---------+--- ----+
```

At this time, if the user often performs an analysis query on the sales volume of different stores, you can create a materialized view for the `sales_records` table to group the sales stores and sum the sales of the same sales stores. The creation statement is as follows:

```
MySQL [test]> create materialized view store_amt as select store_id, sum(sale_amt) from sales_records group by store_id;
```

The backend returns to the following figure, indicating that the task of creating a materialized view is submitted successfully.

```
Query OK, 0 rows affected (0.012 sec)
```

Step 2: Check whether the materialized view has been built

Since the creation of a materialized view is an asynchronous operation, after the user submits the task of creating a materialized view, he needs to asynchronously check whether the materialized view has been constructed through a command. The command is as follows:

```
SHOW ALTER TABLE ROLLUP FROM db_name; (Version 0.12)
SHOW ALTER TABLE MATERIALIZED VIEW FROM db_name; (Version 0.13)
```

In this command, `db_name` is a parameter, you need to replace it with your real db name. The result of the command is to display all the tasks of creating a materialized view of this db. The results are as follows:

```
+-------+---------------+---------------------+--- ------------------+--------------+-------------- --+----------+--------------
+-----------+-------- ------------------------------------------------ ------------------------------------------------ --------
-----+----------+---------+
| JobId | TableName | CreateTime | FinishedTime | BaseIndexName | RollupIndexName | RollupId | TransactionId | State | Msg |
Progress | Timeout |
+-------+---------------+---------------------+--- ------------------+--------------+-------------- --+----------+--------------
+-----------+-------- ------------------------------------------------ ------------------------------------------------ --------
-----+----------+---------+
| 22036 | sales_records | 2020-07-30 20:04:28 | 2020-07-30 20:04:57 | sales_records | store_amt | 22037 | 5008 | FINISHED | | NULL |
86400 |
+-------+---------------+---------------------+--- ------------------+--------------+-------------- --+----------+--------------
+-----------+-------- ---------------------------------------
```

Among them, TableName refers to which table the data of the materialized view comes from, and RollupIndexName refers to the name of the materialized view. One of the more important indicators is State.

When the State of the task of creating a materialized view has become FINISHED, it means that the materialized view has been created successfully. This means that it is possible to automatically match this materialized view when querying.

Step 3: Query

After the materialized view is created, when users query the sales volume of different stores, they will directly read the aggregated data from the materialized view `store_amt` just created. To achieve the effect of improving query efficiency.

The user's query still specifies the query `sales_records` table, for example:

```
SELECT store_id, sum(sale_amt) FROM sales_records GROUP BY store_id;
```

The above query will automatically match `store_amt`. The user can use the following command to check whether the current query matches the appropriate materialized view.

```
EXPLAIN SELECT store_id, sum(sale_amt) FROM sales_records GROUP BY store_id;
+-----------------------------------------------------------------------------------------+
| Explain String                                                                          |
+-----------------------------------------------------------------------------------------+
| PLAN FRAGMENT 0                                                                         |
|   OUTPUT EXPRS:                                                                         |
|     <slot 4> `default_cluster:test`.`sales_records`.`mv_store_id`                       |
|     <slot 5> sum(`default_cluster:test`.`sales_records`.`mva_SUM__`sale_amt``)          |
|   PARTITION: UNPARTITIONED                                                              |
|                                                                                         |
|   VRESULT SINK                                                                          |
|                                                                                         |
|   4:VEXCHANGE                                                                           |
|      offset: 0                                                                          |
|                                                                                         |
| PLAN FRAGMENT 1                                                                         |
|                                                                                         |
|   PARTITION: HASH_PARTITIONED: <slot 4> `default_cluster:test`.`sales_records`.`mv_store_id` |
|                                                                                         |
|   STREAM DATA SINK                                                                      |
|     EXCHANGE ID: 04                                                                     |
|     UNPARTITIONED                                                                       |
|                                                                                         |
|   3:VAGGREGATE (merge finalize)                                                         |
|   |  output: sum(<slot 5> sum(`default_cluster:test`.`sales_records`.`mva_SUM__`sale_amt``)) |
|   |  group by: <slot 4> `default_cluster:test`.`sales_records`.`mv_store_id`           |
|   |  cardinality=-1                                                                     |
|   |                                                                                     |
|   2:VEXCHANGE                                                                           |
|      offset: 0                                                                          |
|                                                                                         |
| PLAN FRAGMENT 2                                                                         |
|                                                                                         |
|   PARTITION: HASH_PARTITIONED: `default_cluster:test`.`sales_records`.`record_id`       |
|                                                                                         |
|   STREAM DATA SINK                                                                      |
|     EXCHANGE ID: 02                                                                     |
|     HASH_PARTITIONED: <slot 4> `default_cluster:test`.`sales_records`.`mv_store_id`     |
|                                                                                         |
```

```
|   1:VAGGREGATE (update serialize)                                                                  |
|   |  STREAMING                                                                                     |
|   |  output: sum(`default_cluster:test`.`sales_records`.`mva_SUM__`sale_amt``)                     |
|   |  group by: `default_cluster:test`.`sales_records`.`mv_store_id`                                |
|   |  cardinality=-1                                                                                |
|   |                                                                                                |
|   0:VOlapScanNode                                                                                  |
|      TABLE: default_cluster:test.sales_records(store_amt), PREAGGREGATION: ON                      |
|      partitions=1/1, tablets=10/10, tabletList=50028,50030,50032 ...                               |
|      cardinality=1, avgRowSize=1520.0, numNodes=1                                                  |
+----------------------------------------------------------------------------------------------------+
```

From the bottom `test.sales_records(store_amt)`, it can be shown that this query hits the `store_amt` materialized view. It is worth noting that if there is no data in the table, then the materialized view may not be hit.

Best Practice 2 PV,UV

Business scenario: Calculate the UV and PV of advertising

Assuming that the user's original ad click data is stored in Doris, then for ad PV and UV queries, the query speed can be improved by creating a materialized view of `bitmap_union`.

Use the following statement to first create a table that stores the details of the advertisement click data, including the click event of each click, what advertisement was clicked, what channel clicked, and who was the user who clicked.

```
MySQL [test]> create table advertiser_view_record(time date, advertiser varchar(10), channel varchar(10), user_id int) distributed
by hash(time) properties("replication_num" = "1");
insert into advertiser_view_record values("2020-02-02",'a','a',1);
```

The original ad click data table structure is:

```
MySQL [test]> desc advertiser_view_record;
+------------+-------------+------+-------+---------+-------+
| Field      | Type        | Null | Key   | Default | Extra |
+------------+-------------+------+-------+---------+-------+
| time       | DATE        | Yes  | true  | NULL    |       |
| advertiser | VARCHAR(10) | Yes  | true  | NULL    |       |
| channel    | VARCHAR(10) | Yes  | false | NULL    | NONE  |
| user_id    | INT         | Yes  | false | NULL    | NONE  |
+------------+-------------+------+-------+---------+-------+
4 rows in set (0.001 sec)
```

1. Create a materialized view

   Since the user wants to query the UV value of the advertisement, that is, a precise de-duplication of users of the same advertisement is required, the user's query is generally:

   ```
   SELECT advertiser, channel, count(distinct user_id) FROM advertiser_view_record GROUP BY advertiser, channel;
   ```

   For this kind of UV-seeking scene, we can create a materialized view with `bitmap_union` to achieve a precise deduplication effect in advance.

In Doris, the result of `count(distinct)` aggregation is exactly the same as the result of `bitmap_union_count` aggregation. And `bitmap_union_count` is equal to the result of `bitmap_union` to calculate count, so if the query **involves** `count(distinct)`**, you can speed up the query by creating a materialized view with** `bitmap_union` **aggregation.**

For this case, you can create a materialized view that accurately deduplicate `user_id` based on advertising and channel grouping.

```
MySQL [test]> create materialized view advertiser_uv as select advertiser, channel, bitmap_union(to_bitmap(user_id)) from
advertiser_view_record group by advertiser, channel;
Query OK, 0 rows affected (0.012 sec)
```

Note: Because the user_id itself is an INT type, it is called `bitmap_union` *directly in Doris. The fields need to be converted to bitmap type through the function* `to_bitmap` *first, and then* `bitmap_union` *can be aggregated.*

After the creation is complete, the table structure of the advertisement click schedule and the materialized view table is as follows:

```
MySQL [test]> desc advertiser_view_record all;
+-----------------------+---------------+----------------------+-------------+------+-------+---------+--------------+
| IndexName             | IndexKeysType | Field                | Type        | Null | Key   | Default | Extra        |
+-----------------------+---------------+----------------------+-------------+------+-------+---------+--------------+
| advertiser_view_record | DUP_KEYS      | time                 | DATE        | Yes  | true  | NULL    |              |
|                        |               | advertiser           | VARCHAR(10) | Yes  | true  | NULL    |              |
|                        |               | channel              | VARCHAR(10) | Yes  | false | NULL    | NONE         |
|                        |               | user_id              | INT         | Yes  | false | NULL    | NONE         |
|                        |               |                      |             |      |       |         |              |
| advertiser_uv          | AGG_KEYS      | advertiser           | VARCHAR(10) | Yes  | true  | NULL    |              |
|                        |               | channel              | VARCHAR(10) | Yes  | true  | NULL    |              |
|                        |               | to_bitmap(`user_id`) | BITMAP      | No   | false |         | BITMAP_UNION |
+-----------------------+---------------+----------------------+-------------+------+-------+---------+--------------+
```

2. Automatic query matching

When the materialized view table is created, when querying the advertisement UV, Doris will automatically query the data from the materialized view `advertiser_uv` just created. For example, the original query statement is as follows:

```
SELECT advertiser, channel, count(distinct user_id) FROM advertiser_view_record GROUP BY advertiser, channel;
```

After the materialized view is selected, the actual query will be transformed into:

```
SELECT advertiser, channel, bitmap_union_count(to_bitmap(user_id)) FROM advertiser_uv GROUP BY advertiser, channel;
```

Through the EXPLAIN command, you can check whether Doris matches the materialized view:

```
mysql [test]>explain SELECT advertiser, channel, count(distinct user_id) FROM  advertiser_view_record GROUP BY advertiser,
channel;
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
| Explain String
|
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
| PLAN FRAGMENT 0
|
|   OUTPUT EXPRS:
|
|     <slot 9> `default_cluster:test`.`advertiser_view_record`.`mv_advertiser`
|
|     <slot 10> `default_cluster:test`.`advertiser_view_record`.`mv_channel`
|
|     <slot 11>
bitmap_union_count(`default_cluster:test`.`advertiser_view_record`.`mva_BITMAP_UNION__to_bitmap_with_check(`user_id`)`)
|
|   PARTITION: UNPARTITIONED
|
|
|   VRESULT SINK
|
|
|   4:VEXCHANGE
|
|      offset: 0
|
|
| PLAN FRAGMENT 1
|
|
|   PARTITION: HASH_PARTITIONED: <slot 6> `default_cluster:test`.`advertiser_view_record`.`mv_advertiser`, <slot 7>
`default_cluster:test`.`advertiser_view_record`.`mv_channel` |
|
|   STREAM DATA SINK
|
|     EXCHANGE ID: 04
|
|     UNPARTITIONED
|
|
|   3:VAGGREGATE (merge finalize)
|
|   |  output: bitmap_union_count(<slot 8>
bitmap_union_count(`default_cluster:test`.`advertiser_view_record`.`mva_BITMAP_UNION__to_bitmap_with_check(`user_id`)`))
|
|   |  group by: <slot 6> `default_cluster:test`.`advertiser_view_record`.`mv_advertiser`, <slot 7>
`default_cluster:test`.`advertiser_view_record`.`mv_channel`                 |
|   |  cardinality=-1
|
|   |
|
|   2:VEXCHANGE
|
|      offset: 0
|
|
| PLAN FRAGMENT 2
|
|
|   PARTITION: HASH_PARTITIONED: `default_cluster:test`.`advertiser_view_record`.`time`
```

```
|
|
|
|   STREAM DATA SINK
|
|     EXCHANGE ID: 02
|
|     HASH_PARTITIONED: <slot 6> `default_cluster:test`.`advertiser_view_record`.`mv_advertiser`, <slot 7>
`default_cluster:test`.`advertiser_view_record`.`mv_channel`          |
|
|
|   1:VAGGREGATE (update serialize)
|
|   |  STREAMING
|
|   |  output:
bitmap_union_count(`default_cluster:test`.`advertiser_view_record`.`mva_BITMAP_UNION__to_bitmap_with_check(`user_id`)`)
|
|   |  group by: `default_cluster:test`.`advertiser_view_record`.`mv_advertiser`,
`default_cluster:test`.`advertiser_view_record`.`mv_channel`                                    |
|   |  cardinality=-1
|
|   |
|
|   0:VOlapScanNode
|
|      TABLE: default_cluster:test.advertiser_view_record(advertiser_uv), PREAGGREGATION: ON
|
|      partitions=1/1, tablets=10/10, tabletList=50075,50077,50079 ...
|
|      cardinality=0, avgRowSize=48.0, numNodes=1
|
+----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
```

In the result of EXPLAIN, you can first see that `VOlapScanNode` hits `advertiser_uv`. That is, the query scans the materialized view's data directly. Indicates that the match is successful.

Secondly, the calculation of `count(distinct)` for the `user_id` field is rewritten as `bitmap_union_count`. That is to achieve the effect of precise deduplication through bitmap.

Best Practice 3

Business scenario: matching a richer prefix index

The user's original table has three columns (k1, k2, k3). Among them, k1, k2 are prefix index columns. At this time, if the user query condition contains `where k1=a and k2=b`, the query can be accelerated through the index.

But in some cases, the user's filter conditions cannot match the prefix index, such as `where k3=c`. Then the query speed cannot be improved through the index.

This problem can be solved by creating a materialized view with k3 as the first column.

1. Create a materialized view

```
CREATE MATERIALIZED VIEW mv_1 as SELECT k3, k2, k1 FROM tableA ORDER BY k3;
```

After the creation of the above grammar is completed, the complete detail data is retained in the materialized view, and the prefix index of the materialized view is the k3 column. The table structure is as follows:

```
MySQL [test]> desc tableA all;
+-----------+---------------+-------+------+------+-------+---------+-------+
| IndexName | IndexKeysType | Field | Type | Null | Key   | Default | Extra |
+-----------+---------------+-------+------+------+-------+---------+-------+
| tableA    | DUP_KEYS      | k1    | INT  | Yes  | true  | NULL    |       |
|           |               | k2    | INT  | Yes  | true  | NULL    |       |
|           |               | k3    | INT  | Yes  | true  | NULL    |       |
|           |               |       |      |      |       |         |       |
| mv_1      | DUP_KEYS      | k3    | INT  | Yes  | true  | NULL    |       |
|           |               | k2    | INT  | Yes  | false | NULL    | NONE  |
|           |               | k1    | INT  | Yes  | false | NULL    | NONE  |
+-----------+---------------+-------+------+------+-------+---------+-------+
```

2. Query matching

   At this time, if the user's query has k3 column, the filter condition is, for example:

   ```
   select k1, k2, k3 from table A where k3=1;
   ```

   At this time, the query will read data directly from the mv_1 materialized view just created. The materialized view has a prefix index on k3, and query efficiency will also be improved.

Best Practice 4

> **Tips**
>
> This feature is supported since the Apache Doris 2.0 version

In `Doris 2.0`, we have made some enhancements to the expressions supported by the materialized view. This example will mainly reflect the support and early filtering of the new version of the materialized view for various expressions.

1. Create a base table and insert some data. ```sql create table d_table (k1 int null, k2 int not null, k3 bigint null, k4 date null) duplicate key (k1,k2,k3) distributed BY hash(k1) buckets 3 properties("replication_num" = "1");

   insert into d_table select 1,1,1,'2020-02-20'; insert into d_table select 2,2,2,'2021-02-20'; insert into d_table select 3,-3,null,'2022-02-20'; ```

2. Create some materialized views. `sql create materialized view k1a2p2ap3ps as select abs(k1)+k2+1,sum(abs(k2+2)+k3+3) from d_table group by abs(k1)+k2+1; create materialized view kymd as select year(k4),month(k4) from d_table where year(k4) = 2020; // Filter with where expression in advance to reduce the amount of data in the materialized view.`

3. Use some queries to test if the materialized view was successfully hit. `sql select abs(k1)+k2+1, sum(abs(k2+2)+k3+3) from d_table group by abs(k1)+k2+1; // hit k1a2p2ap3ps select bin(abs(k1)+k2+1),`

```
sum(abs(k2+2)+k3+3) from d_table group by bin(abs(k1)+k2+1); // hit k1a2p2ap3ps select
year(k4),month(k4),day(k4) from d_table; // cannot hit the materialized view because the where
condition does not match select year(k4)+month(k4) from d_table where year(k4) = 2020; // hit kymd
```

Limitations

1. If the condition column of the delete statement exists in the materialized view, the delete operation cannot be performed. If you must delete the data, you need to delete the materialized view before deleting the data.
2. Too many materialized views on a single table will affect the efficiency of import: when importing data, the materialized view and Base table data are updated synchronously. If a table has more than 10 materialized views, the import speed may be slow. slow. This is the same as if a single import needs to import 10 table data at the same time.
3. For the Unique Key data model, the materialized view can only change the order of the columns and cannot perform aggregation. Therefore, it is not possible to perform coarse-grained aggregation operations on the data by creating a materialized view on the Unique Key model.
4. At present, the rewriting behavior of some optimizers to SQL may cause the materialized view to fail to be hit. For example, k1+1-1 is rewritten as k1, between is rewritten as <= and >=, and day is rewritten as dayofmonth. In this case, you need to manually adjust the statements of the query and materialized view.

Error

1. DATA_QUALITY_ERROR: “The data quality does not satisfy, please check your data” Materialized view creation failed due to data quality issues or Schema Change memory usage exceeding the limit. If it is a memory problem, increase the `memory_limitation_per_thread_for_schema_change_bytes` parameter. Note: The bitmap type only supports positive integers. If there are negative Numbers in the original data, the materialized view will fail to be created

More Help

For more detailed syntax and best practices for using materialized views, see CREATE MATERIALIZED VIEW and DROP MATERIALIZED VIEW command manual, you can also Enter `HELP CREATE MATERIALIZED VIEW` and `HELP DROP MATERIALIZED VIEW` at the command line of the MySql client for more help information.

Doris Join Optimization Principle

Doris Join optimization principle

Doris supports two physical operators, one is **Hash Join**, and the other is **Nest Loop Join**.

- Hash Join: Create a hash table on the right table based on the equivalent join column, and the left table uses the hash table to perform join calculations in a streaming manner. Its limitation is that it can only be applied to equivalent joins.
- Nest Loop Join: With two for loops, it is very intuitive. Then it is applicable to unequal-valued joins, such as: greater than or less than or the need to find a Cartesian product. It is a general join operator, but has poor performance.

As a distributed MPP database, data shuffle needs to be performed during the Join process. Data needs to be split and scheduled to ensure that the final Join result is correct. As a simple example, assume that the relationship S and R are joined, and N represents the number of nodes participating in the join calculation; T represents the number of tuples in the relationship.

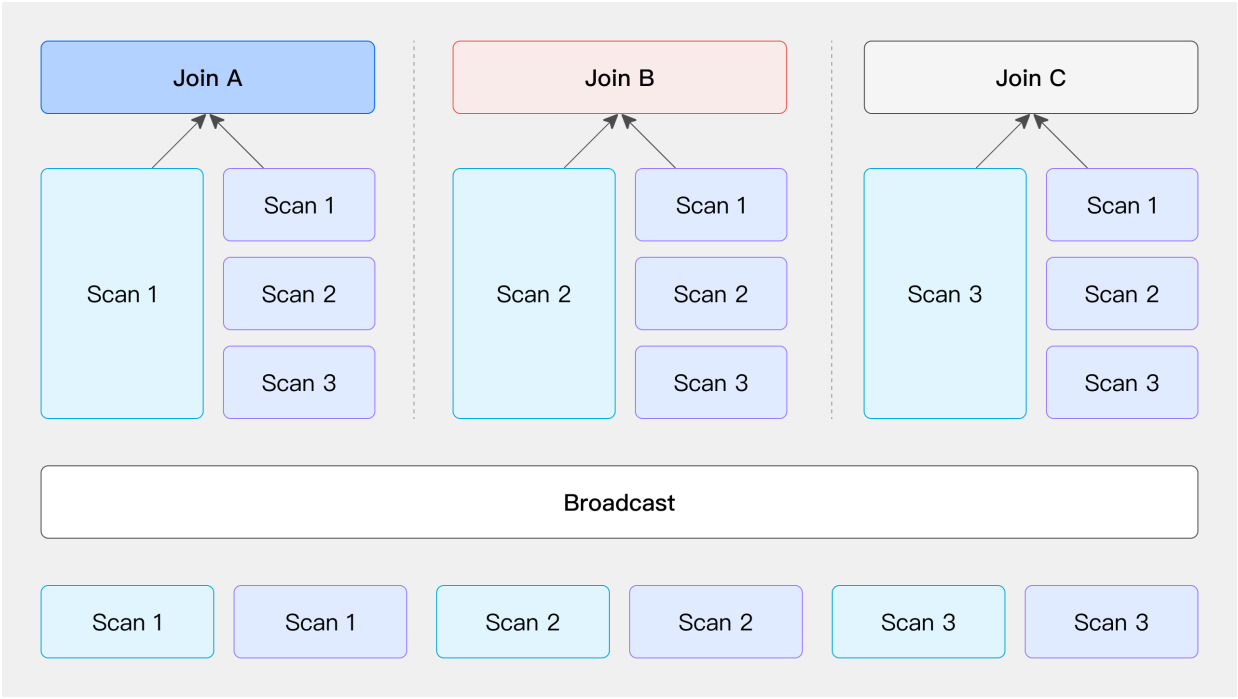
Doris Shuffle way

Doris supports 4 Shuffle methods

1. BroadCast Join

   It requires the full data of the right table to be sent to the left table, that is, each node participating in Join has the full data of the right table, that is, T(R).

   Its applicable scenarios are more general, and it can support Hash Join and Nest loop Join at the same time, and its network overhead is N * T(R).



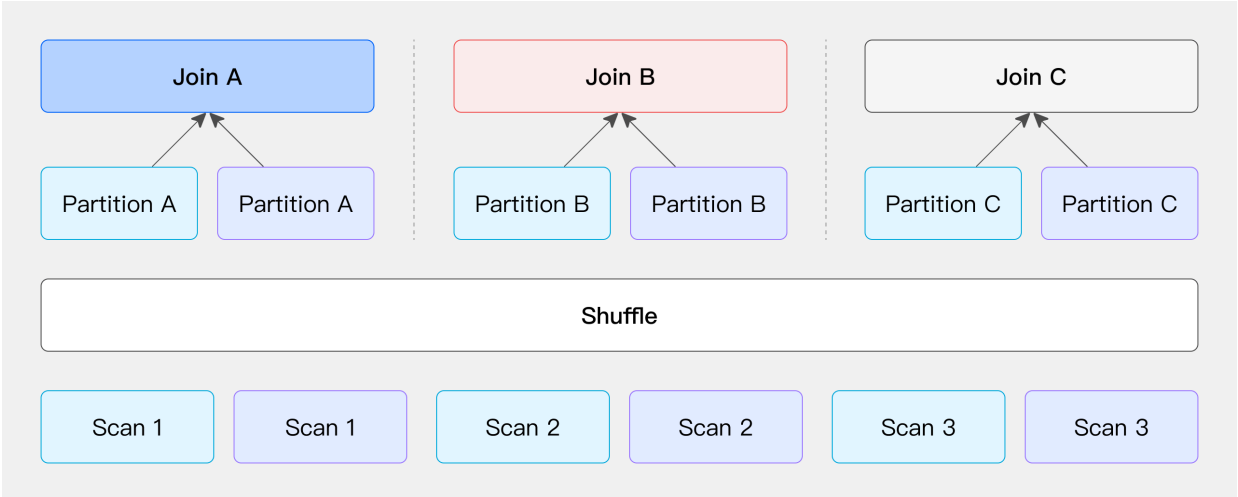
BroadCast Join

The data in the left table is not moved, and the data in the right table is sent to the scanning node of the data in the left table.

2. Shuffle Join

When Hash Join is performed, the corresponding Hash value can be calculated through the Join column, and Hash bucketing can be performed.

Its network overhead is: T(S) + T(R), but it can only support Hash Join, because it also calculates buckets according to the conditions of Join.



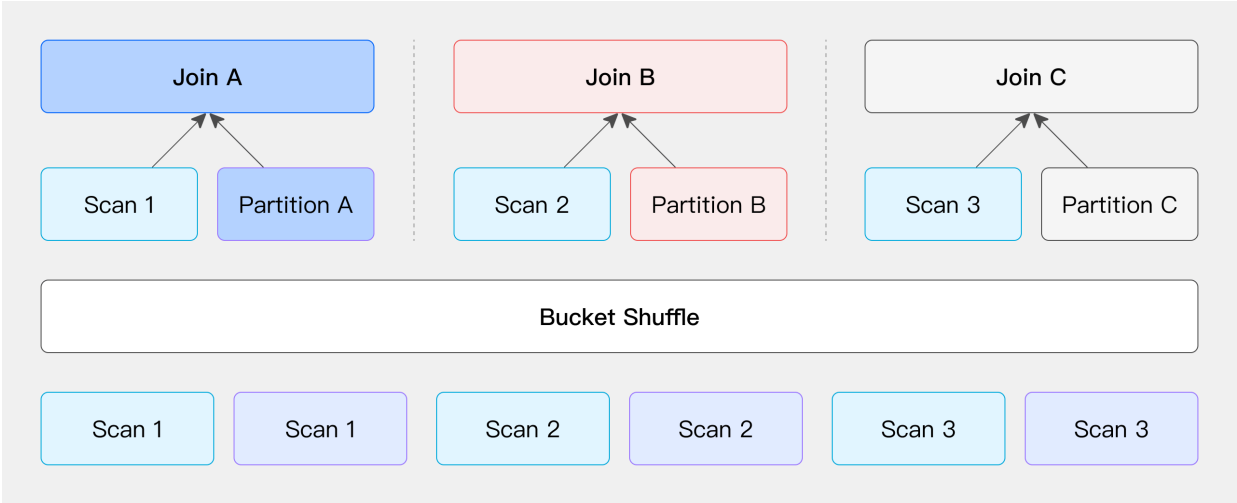
Shuffle Join

The left and right table data are sent to different partition nodes according to the partition, and the calculated demerits are sent.

3. Bucket Shuffle Join

Doris's table data itself is bucketed by Hash calculation, so you can use the properties of the bucketed columns of the table itself to shuffle the Join data. If two tables need to be joined, and the Join column is the bucket column of the left table, then the data in the left table can actually be calculated by sending the data into the buckets of the left table without moving the data in the right table.

Its network overhead is: T(R) is equivalent to only Shuffle the data in the right table.

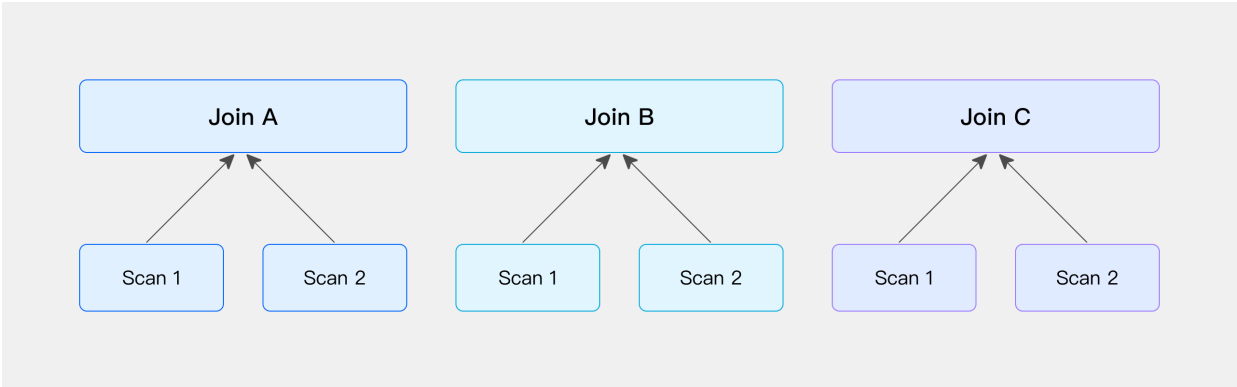


Bucket Shuffle Join

The data in the left table does not move, and the data in the right table is sent to the node that scans the table in the left table according to the result of the partition calculation.

4. Colocation

It is similar to Bucket Shuffle Join, which means that the data has been shuffled according to the preset Join column scenario when data is imported. Then the join calculation can be performed directly without considering the Shuffle problem of the data during the actual query.



Colocation Join

The data has been pre-partitioned, and the Join calculation is performed directly locally

Comparison of four Shuffle methods

Shuffle Mode	Network Overhead	Physical Operators	Applicable Scenarios
Broad Cast	N * T(R)	Hash Join / Nest Loop Join	Universal
Shuffle	T(S) + T(R)	Hash Join	General
Bucket Shuffle	T(R)	Hash Join	There are distributed columns in the left table in the join condition, and the left table is executed as a single partition
Colocate	0	Hash Join	There are distributed columns in the left table in the join condition, and the left and right tables belong to the same Colocate Group

N : The number of Instances participating in the Join calculation

T(relation) : Tuple number of relation

The flexibility of the above four methods is from high to low, and its requirements for this data distribution are becoming more and more strict, but the performance of Join calculation is also getting better and better.

Runtime Filter Join optimization

Doris will build a hash table in the right table when performing Hash Join calculation, and the left table will stream through the hash table of the right table to obtain the join result. The RuntimeFilter makes full use of the Hash table of the right table. When the right table generates a hash table, a filter condition based on the hash table data is generated at the same time, and then pushed down to the data scanning node of the left table. In this way, Doris can perform data filtering at runtime.

If the left table is a large table and the right table is a small table, then using the filter conditions generated by the left table, most of the data to be filtered in the Join layer can be filtered in advance when the data is read, so that a large amount of data can be filtered. Improve the performance of join queries.

Currently Doris supports three types of RuntimeFilter

- One is IN-IN, which is well understood, and pushes a hashset down to the data scanning node.
- The second is BloomFilter, which uses the data of the hash table to construct a BloomFilter, and then pushes the BloomFilter down to the scanning node that queries the data. .
- The last one is MinMax, which is a Range range. After the Range range is determined by the data in the right table, it is pushed down to the data scanning node.

There are two requirements for the applicable scenarios of Runtime Filter:

- The first requirement is that the right table is large and the left table is small, because building a Runtime Filter needs to bear the computational cost, including some memory overhead.

- The second requirement is that there are few results from the join of the left and right tables, indicating that this join can filter out most of the data in the left table.

When the above two conditions are met, turning on the Runtime Filter can achieve better results

When the Join column is the Key column of the left table, the RuntimeFilter will be pushed down to the storage engine. Doris itself supports delayed materialization,

Delayed materialization is simply like this: if you need to scan three columns A, B, and C, there is a filter condition on column A: A is equal to 2, if you want to scan 100 rows, you can scan 100 rows of column A first, Then filter through the filter condition A = 2. After filtering the results, read columns B and C, which can greatly reduce the data read IO. Therefore, if the Runtime Filter is generated on the Key column, and the delayed materialization of Doris itself is used to further improve the performance of the query.

Runtime Filter Type

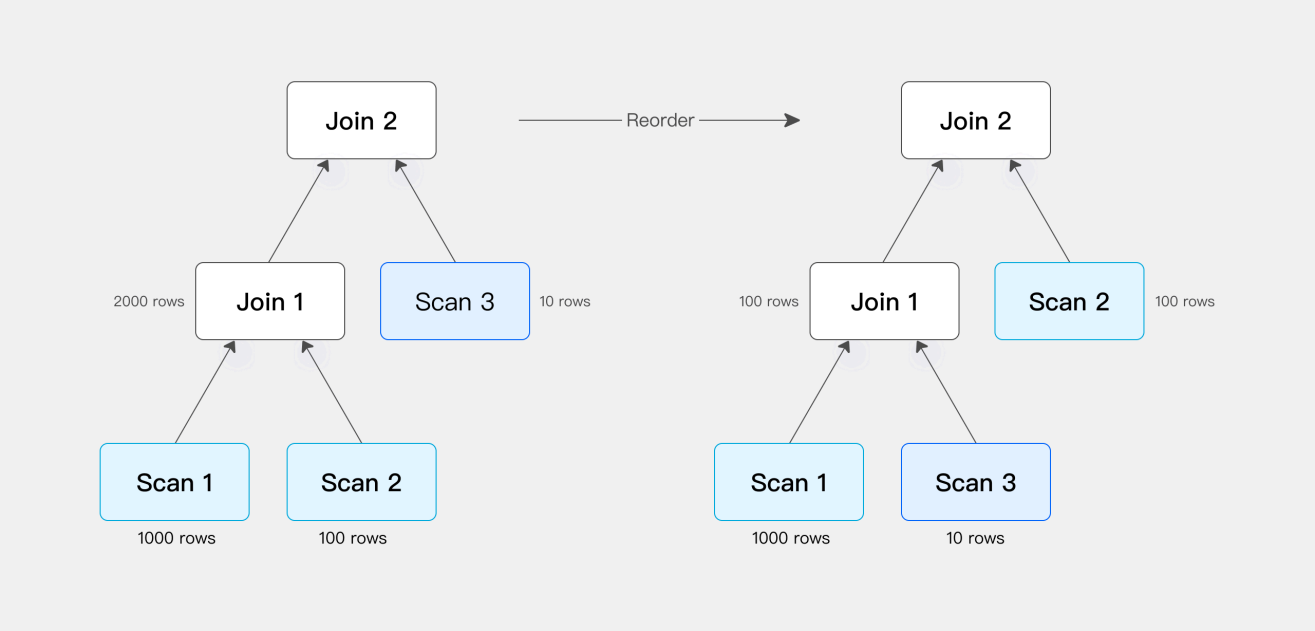
Doris provides three different Runtime Filter types:

- The advantage of **IN** is that the effect filtering effect is obvious and fast. Its shortcomings are: First, it only applies to BroadCast. Second, when the right table exceeds a certain amount of data, it will fail. The current Doris configuration is 1024, that is, if the right table is larger than 1024, the Runtime Filter of IN will directly failed.
- The advantage of **MinMax** is that the overhead is relatively small. Its disadvantage is that it has a relatively good effect on numeric columns, but basically no effect on non-numeric columns.
- The feature of **Bloom Filter** is that it is universal, suitable for various types, and the effect is better. The disadvantage is that its configuration is more complicated and the calculation is high.

Join Reader

Once the database involves multi-table Join, the order of Join has a great impact on the performance of the entire Join query. Assuming that there are three tables to join, refer to the following picture, the left is the a table and the b table to do the join first, the intermediate result has 2000 rows, and then the c table is joined.

Next, look at the picture on the right and adjust the order of Join. Join the a table with the c table first, the intermediate result generated is only 100, and then finally join with the b table for calculation. The final join result is the same, but the intermediate result it generates has a 20x difference, which results in a big performance diff.



Join Reorder

Doris currently supports the rule-based Join Reorder algorithm. Its logic is:

- Make joins with large tables and small tables as much as possible, and the intermediate results it generates are as small as possible.
- Put the conditional join table forward, that is to say, try to filter the conditional join table
- Hash Join has higher priority than Nest Loop Join, because Hash Join itself is much faster than Nest Loop Join.

Doris Join optimization method

Doris Join tuning method:

- Use the Profile provided by Doris itself to locate the bottleneck of the query. Profile records various information in Doris' entire query, which is first-hand information for performance tuning. .
- Understand the Join mechanism of Doris, which is also the content shared with you in the second part. Only by knowing why and understanding its mechanism can we analyze why it is slow.
- Use Session variables to change some behaviors of Join, so as to realize the tuning of Join.
- Check the Query Plan to analyze whether this tuning is effective.

The above 4 steps basically complete a standard Join tuning process, and then it is to actually query and verify it to see what the effect is.

If the first 4 methods are connected in series, it still does not work. At this time, it may be necessary to rewrite the Join statement, or to adjust the data distribution. It is necessary to recheck whether the entire data distribution is reasonable, including querying the Join statement, and some manual adjustments may be required. Of course, this method has a relatively high mental cost, which means that further analysis is required only when the previous method does not work.

Optimization case practice

Case 1

A four-table join query, through Profile, found that the second join took a long time, taking 14 seconds.

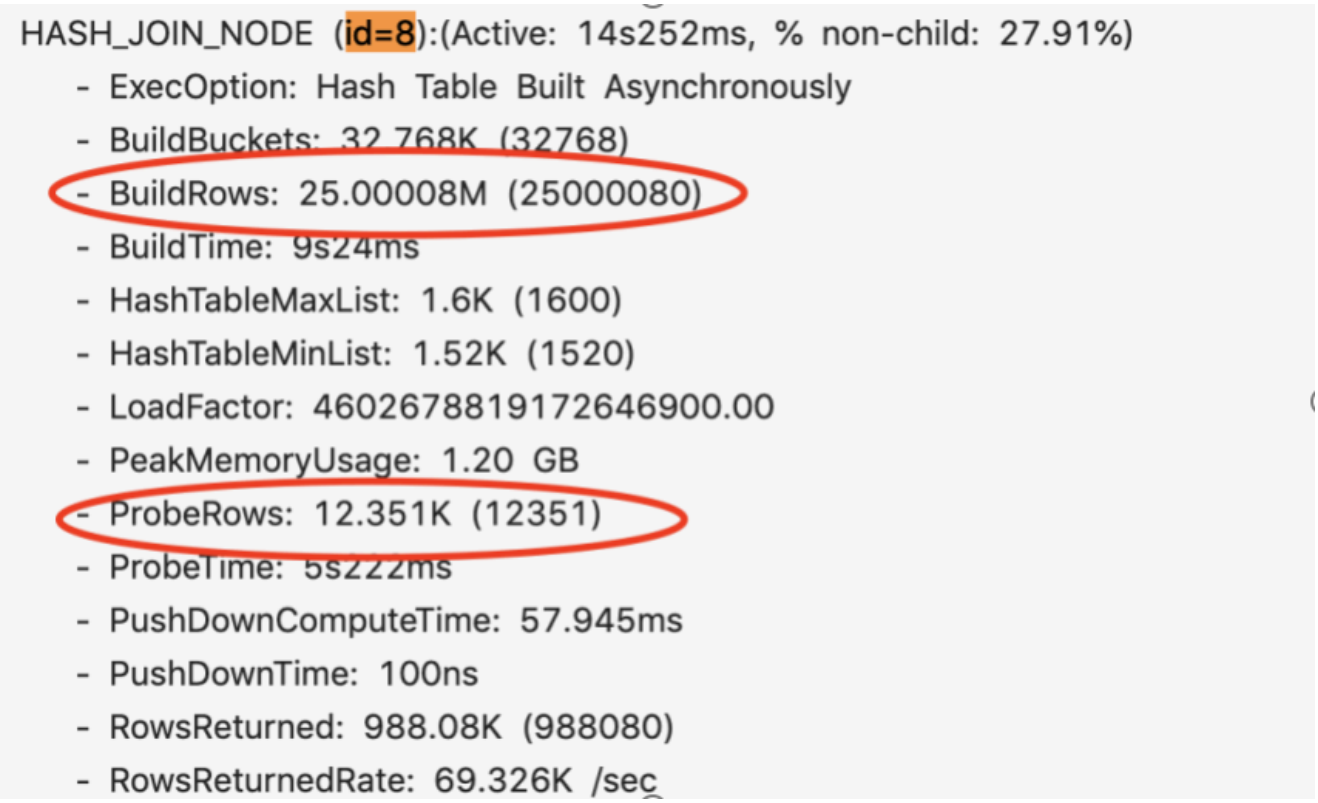


image-20220523153600797

After further analysis of Profile, it is found that BuildRows, that is, the data volume of the right table is about 25 million. And ProbeRows (ProbeRows is the amount of data in the left table) is only more than 10,000. In this scenario, the right table is much larger than the left table, which is obviously an unreasonable situation. This obviously shows that there is some problem with the order of Join. At this time, try to change the Session variable and enable Join Reorder.

```
set enable_cost_based_join_reorder = true
```

This time, the time has been reduced from 14 seconds to 4 seconds, and the performance has been improved by more than 3 times.

At this time, when checking the profile again, the order of the left and right tables has been adjusted correctly, that is, the right table is a large table, and the left table is a small table. The overhead of building a hash table based on a small table is very small. This is a typical scenario of using Join Reorder to improve Join performance.

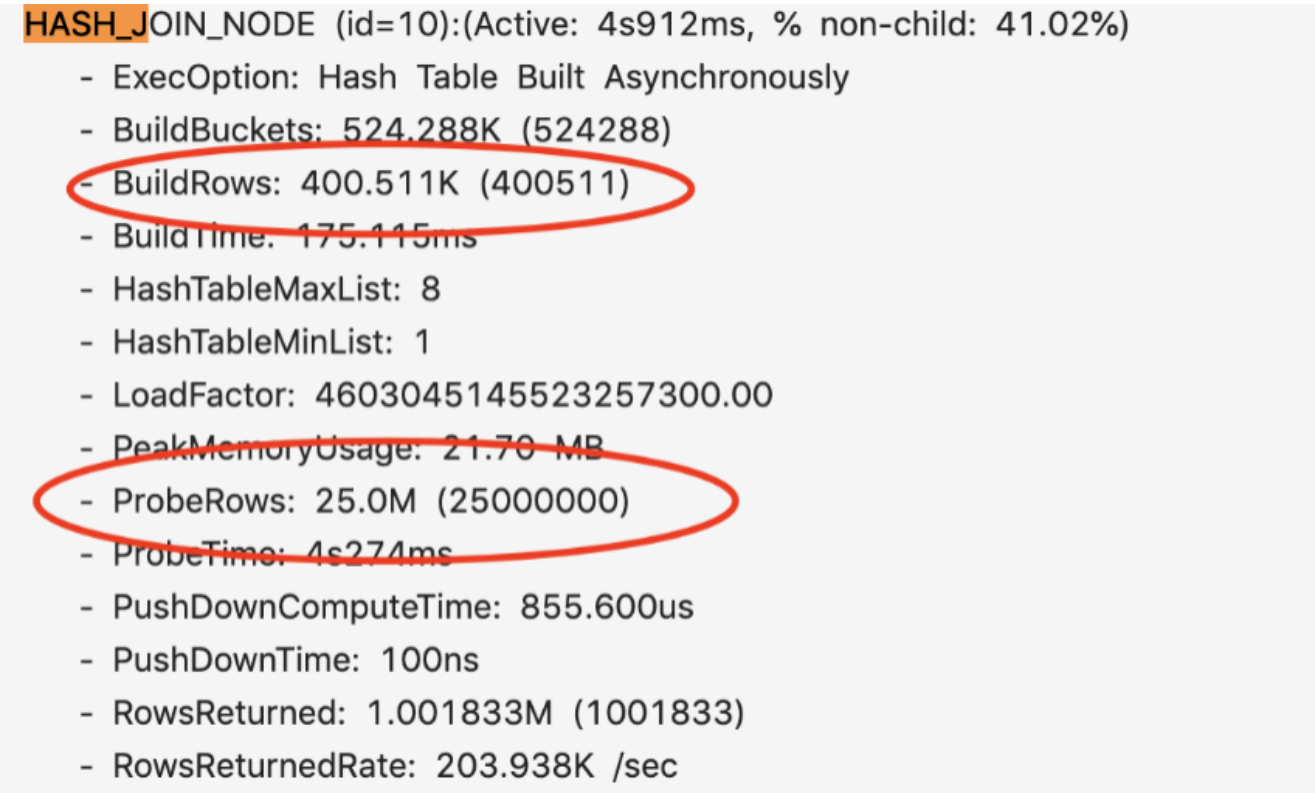


image-20220523153757607

Case 2

There is a slow query. After viewing the Profile, the entire Join node takes about 44 seconds. Its right table has 10 million, the left table has 60 million, and the final returned result is only 60 million.

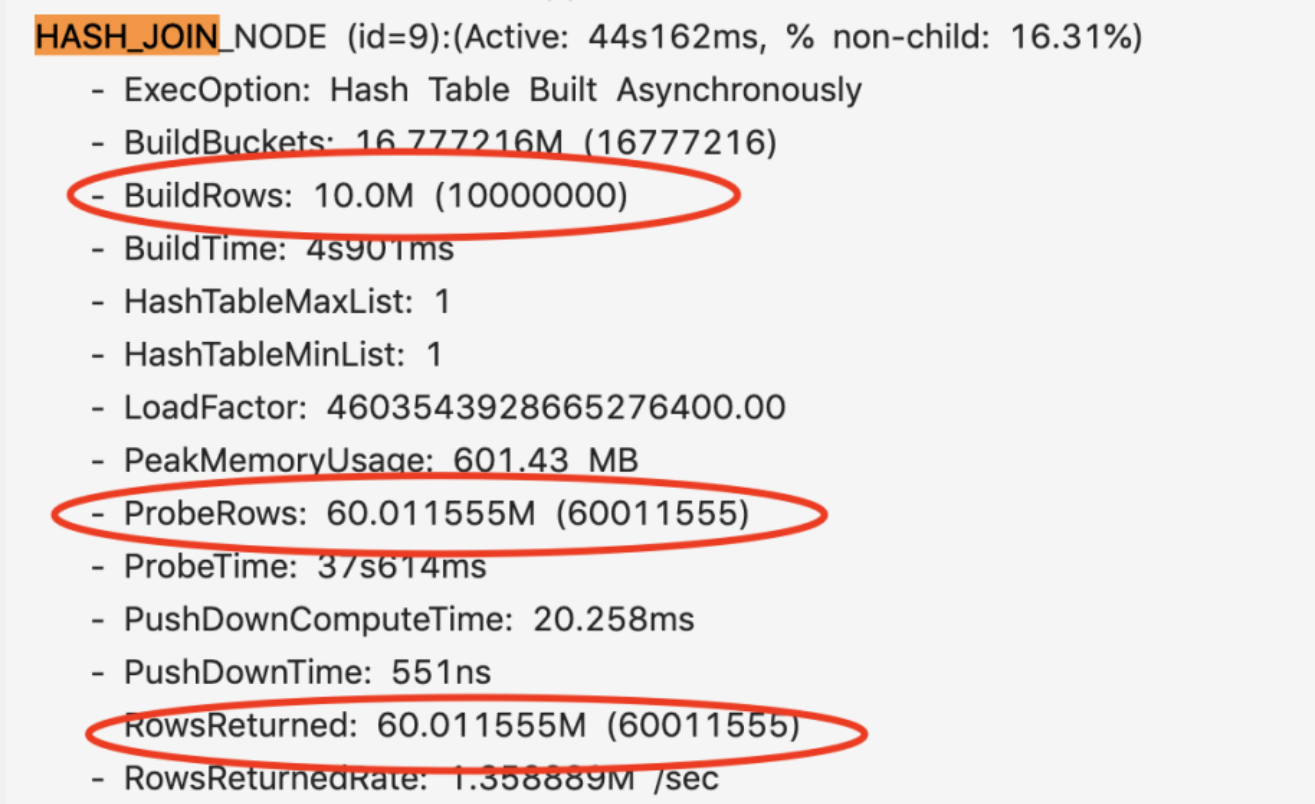


image-20220523153913059

It can be roughly estimated that the filtering rate is very high, so why does the Runtime Filter not take effect? Check it out through Query Plan and find that it only enables the Runtime Filter of IN.

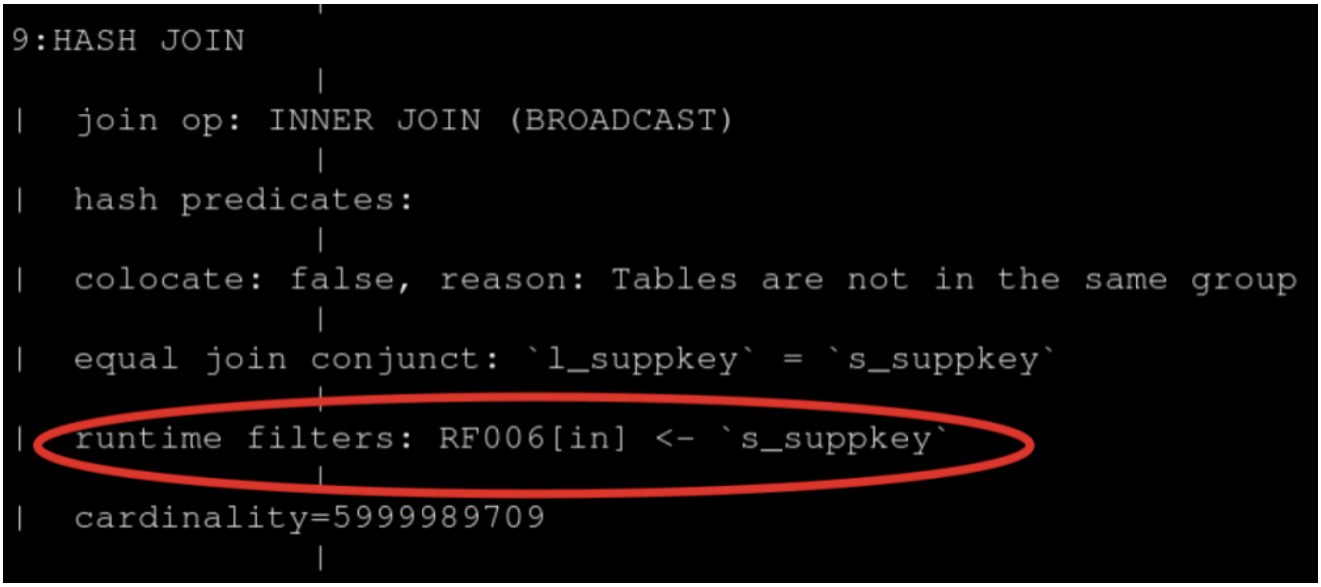


image-20220523153958828

When the right table exceeds 1024 rows, IN will not take effect, so there is no filtering effect at all, so try to adjust the type of RuntimeFilter.

This is changed to BloomFilter, and the 60 million pieces of data in the left table have filtered 59 million pieces. Basically, 99% of the data is filtered out, and this effect is very significant. The query has also dropped from the original 44 seconds to 13 seconds, and the performance has been improved by about three times.

Case 3

The following is a relatively extreme case, which cannot be solved by tuning some environment variables, because it involves SQL Rewrite, so the original SQL is listed here.

```
select 100.00 * sum (case
        when P_type like 'PROMOS'
        then 1 extendedprice * (1 - 1 discount)
        else 0
        end ) / sum(1 extendedprice * (1 - 1 discount)) as promo revenue
from lineitem, part
where
    l_partkey = p_partkey
    and l_shipdate >= date '1997-06-01'
    and l shipdate < date '1997-06-01' + interval '1' month
```

This Join query is very simple, a simple join of left and right tables. Of course, there are some filter conditions on it. When I opened the Profile, I found that the entire query Hash Join was executed for more than three minutes. It is a BroadCast Join, and its right table has 200 million entries, while the left table has only 700,000. In this case, it is unreasonable to choose Broadcast Join, which is equivalent to making a Hash Table of 200 million records, and then traversing the Hash Table of 200 million records with 700,000 records, which is obviously unreasonable.

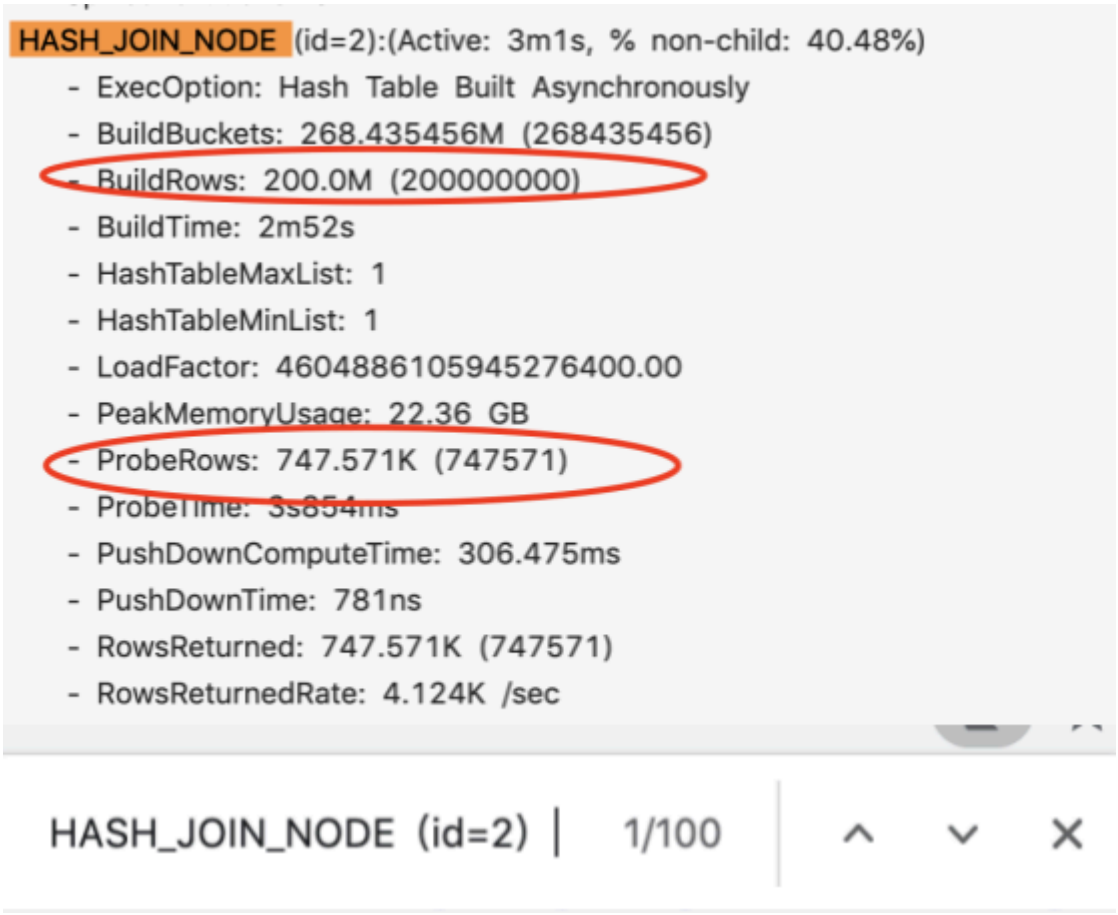


image-20220523154712519

Why is there an unreasonable Join order? In fact, the left table is a large table with a level of 1 billion records. Two filter conditions are added to it. After adding these two filter conditions, there are 700,000 records of 1 billion records. But Doris currently does not have a good framework for collecting statistics, so it does not know what the filtering rate of this filter condition is. Therefore, when the join order is arranged, the wrong left and right table order of the join is selected, resulting in extremely low performance.

The following figure is an SQL statement after the rewrite is completed. A Join Hint is added after the Join, a square bracket is added after the Join, and then the required Join method is written. Here, Shuffle Join is selected. You can see in the actual query plan on the right that the data is indeed Partitioned. The original 3-minute time-consuming is only 7 seconds after the rewriting, and the performance is improved significantly.

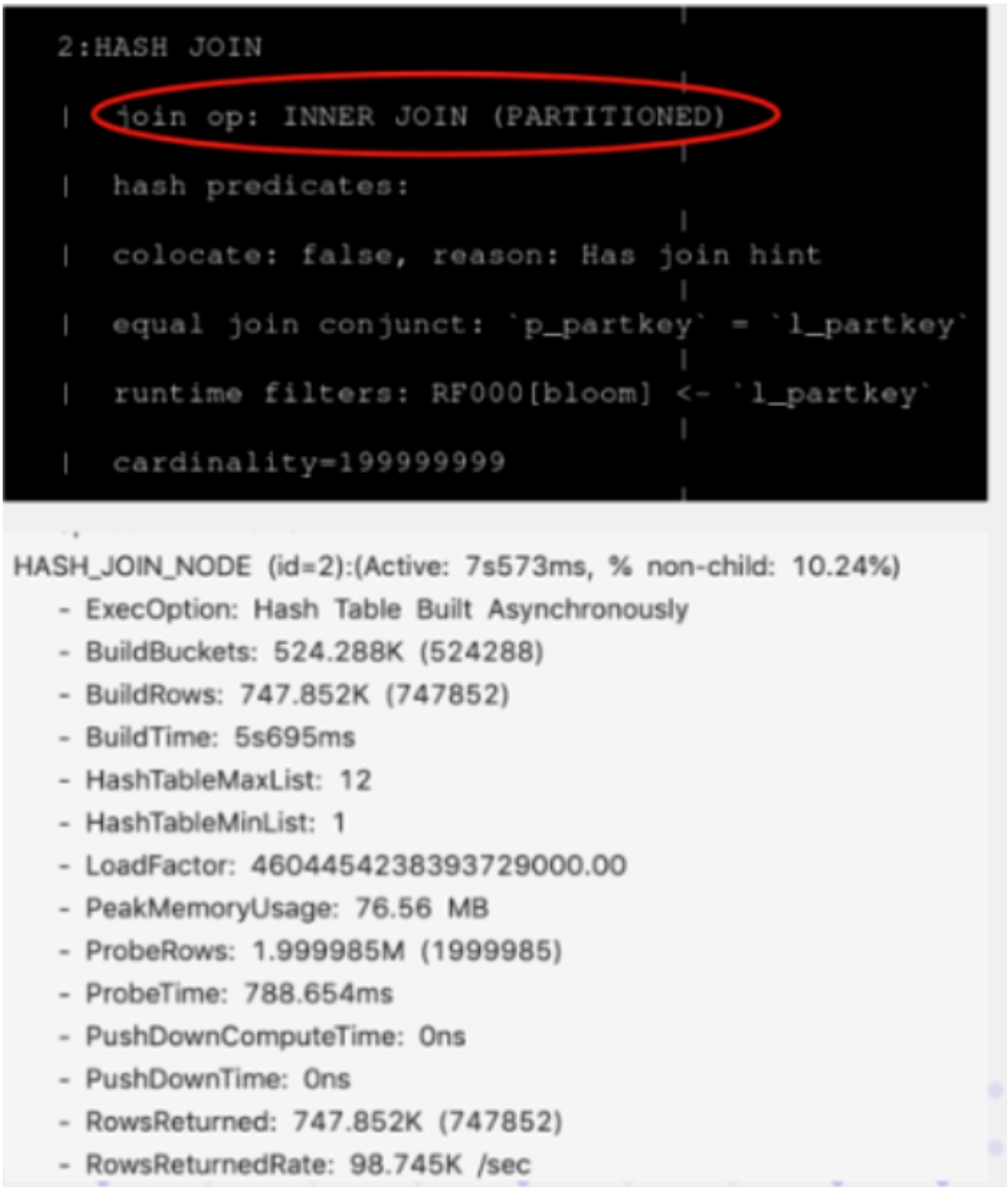


image-20220523160915229

Doris Join optimization suggestion

Finally, we summarize four suggestions for optimization and tuning of Doris Join:

- The first point: When doing Join, try to select columns of the same type or simple type. If the same type is used, reduce its data cast, and the simple type itself joins the calculation quickly.
- The second point: try to choose the Key column for Join. The reason is also introduced in the Runtime Filter. The Key column can play a better effect on delayed materialization.
- The third point: Join between large tables, try to make it Co-location, because the network overhead between large tables is very large, if you need to do Shuffle, the cost is very high.

- Fourth point: Use Runtime Filter reasonably, which is very effective in scenarios with high join filtering rate. But it is not a panacea, but has certain side effects, so it needs to be switched according to the granularity of specific SQL.
- Finally: When it comes to multi-table Join, it is necessary to judge the rationality of Join. Try to ensure that the left table is a large table and the right table is a small table, and then Hash Join will be better than Nest Loop Join. If necessary, you can use SQL Rewrite to adjust the order of Join using Hint.

Bucket Shuffle Join

Bucket Shuffle Join

Bucket Shuffle Join is a new function officially added in Doris 0.14. The purpose is to provide local optimization for some join queries to reduce the time-consuming of data transmission between nodes and speed up the query.

It's design, implementation can be referred to ISSUE 4394。

Noun Interpretation

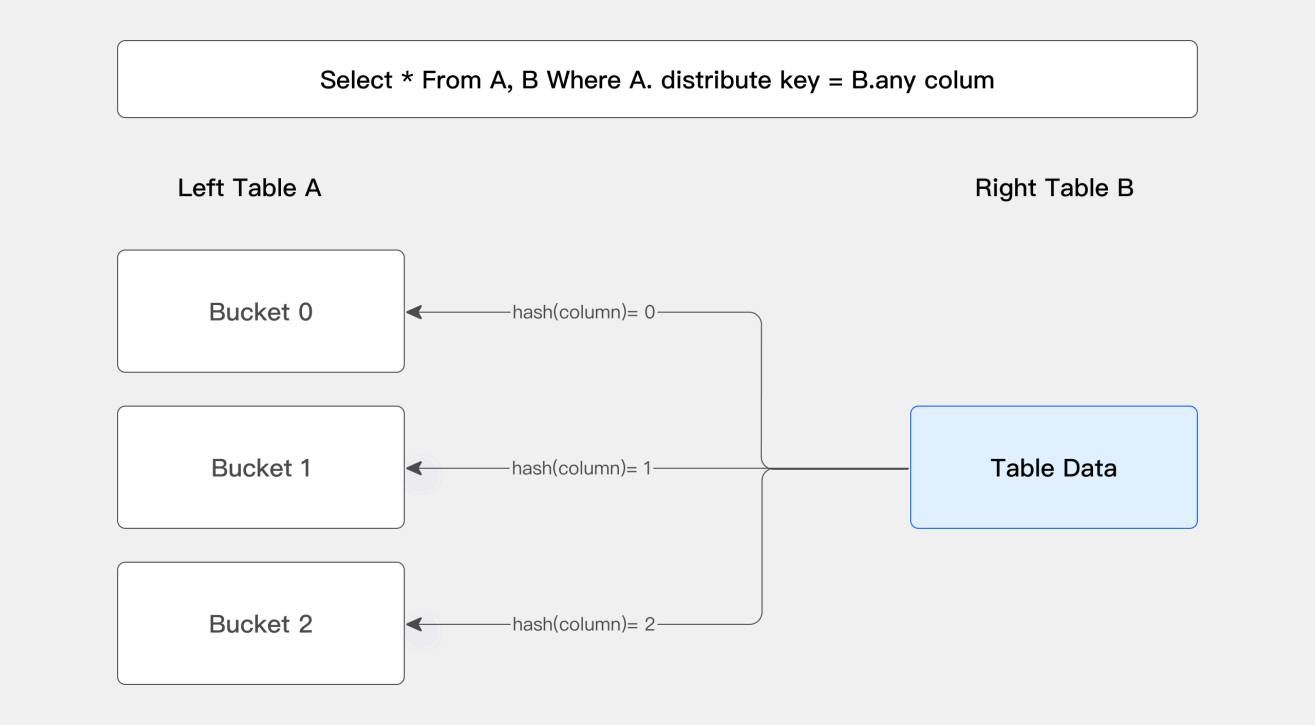
- FE: Frontend, the front-end node of Doris. Responsible for metadata management and request access.
- BE: Backend, Doris's back-end node. Responsible for query execution and data storage.
- Left table: the left table in join query. Perform probe expr. The order can be adjusted by join reorder.
- Right table: the right table in join query. Perform build expr The order can be adjusted by join reorder.

Principle

The conventional distributed join methods supported by Doris is: `Shuffle Join, Broadcast Join`. Both of these join will lead to some network overhead.

For example, there are join queries for table A and table B. the join method is hashjoin. The cost of different join types is as follows: * **Broadcast Join**: If table a has three executing hashjoinnodes according to the data distribution, table B needs to be sent to the three HashJoinNode. Its network overhead is `3B`, and its memory overhead is `3B`. * **Shuffle Join**: Shuffle join will distribute the data of tables A and B to the nodes of the cluster according to hash calculation, so its network overhead is `A + B` and memory overhead is `B`.

The data distribution information of each Doris table is saved in FE. If the join statement hits the data distribution column of the left table, we should use the data distribution information to reduce the network and memory overhead of the join query. This is the source of the idea of bucket shuffle join.



Shuffle Join.png

The picture above shows how the Bucket Shuffle Join works. The SQL query is A table join B table. The equivalent expression of join hits the data distribution column of A. According to the data distribution information of table A. Bucket Shuffle Join sends the data of table B to the corresponding data storage and calculation node of table A. The cost of Bucket Shuffle Join is as follows:

- network cost: `B < min(3B, A + B)`
- memory cost: `B <= min(3B, B)`

Therefore, compared with Broadcast Join and Shuffle Join, Bucket shuffle join has obvious performance advantages. It reduces the time-consuming of data transmission between nodes and the memory cost of join. Compared with Doris's original join method, it has the following advantages

- First of all, Bucket Shuffle Join reduces the network and memory cost which makes some join queries have better performance. Especially when FE can perform partition clipping and bucket clipping of the left table.
- Secondly, unlike Colocate Join, it is not intrusive to the data distribution of tables, which is transparent to users. There is no mandatory requirement for the data distribution of the table, which is not easy to lead to the problem of data skew.
- Finally, it can provide more optimization space for join reorder.

Usage

Set session variable

Set session variable `enable_bucket_shuffle_join` to `true`, FE will automatically plan queries that can be converted to Bucket Shuffle Join.

```
set enable_bucket_shuffle_join = true;
```

In FE's distributed query planning, the priority order is Colocate Join -> Bucket Shuffle Join -> Broadcast Join -> Shuffle Join. However, if the user explicitly hints the type of join, for example:

```
select * from test join [shuffle] baseall on test.k1 = baseall.k1;
```

the above order of preference will not take effect.

The session variable is set to `true` by default in version 0.14, while it needs to be set to `true` manually in version 0.13.

View the type of join

You can use the `explain` command to check whether the join is a Bucket Shuffle Join

```
|   2:HASH JOIN
|
|   |  join op: INNER JOIN (BUCKET_SHUFFLE)
|
|   |  hash predicates:
|
|   |  colocate: false, reason: table not in the same group
|
|   |  equal join conjunct: `test`.`k1` = `baseall`.`k1`
```

The join type indicates that the join method to be used is:`BUCKET_SHUFFLE`。

Planning rules of Bucket Shuffle Join

In most scenarios, users only need to turn on the session variable by default to transparently use the performance improvement brought by this join method. However, if we understand the planning rules of Bucket Shuffle Join, we can use it to write more efficient SQL.

- Bucket Shuffle Join only works when the join condition is equivalent. The reason is similar to Colocate Join. They all rely on hash to calculate the determined data distribution.
- The bucket column of two tables is included in the equivalent join condition. When the bucket column of the left table is an equivalent join condition, it has a high probability of being planned as a Bucket Shuffle Join.
- Because the hash values of different data types have different calculation results. Bucket Shuffle Join requires that the bucket column type of the left table and the equivalent join column type of the right table should be consistent, otherwise the corresponding planning cannot be carried out.
- Bucket Shuffle Join only works on Doris native OLAP tables. For ODBC, MySQL, ES External Table, when they are used as left tables, they cannot be planned as Bucket Shuffle Join.
- For partitioned tables, because the data distribution rules of each partition may be different, the Bucket Shuffle Join can only guarantee that the left table is a single partition. Therefore, in SQL execution, we need to use the `where` condition as far as possible to make the partition clipping policy effective.
- If the left table is a colocate table, the data distribution rules of each partition are determined. So the bucket shuffle join can perform better on the colocate table.

Colocation Join

Colocation Join

Colocation Join is a new feature introduced in Doris 0.9. The purpose of this paper is to provide local optimization for some Join queries to reduce data transmission time between nodes and speed up queries.

The original design, implementation and effect can be referred to ISSUE 245.

The Colocation Join function has undergone a revision, and its design and use are slightly different from the original design. This document mainly introduces Colocation Join's principle, implementation, usage and precautions.

Note: This property will not be synchronized by CCR. If this table is copied by CCR, that is, PROPERTIES contains `is_being_synced = true`, this property will be erased in this table.

Noun Interpretation

- FE: Frontend, the front-end node of Doris. Responsible for metadata management and request access.
- BE: Backend, Doris's back-end node. Responsible for query execution and data storage.
- Colocation Group (CG): A CG contains one or more tables. Tables within the same group have the same Colocation Group Schema and the same data fragmentation distribution.
- Colocation Group Schema (CGS): Used to describe table in a CG and general Schema information related to Colocation. Including bucket column type, bucket number and copy number.

Principle

The Colocation Join function is to make a CG of a set of tables with the same CGS. Ensure that the corresponding data fragments of these tables will fall on the same BE node. When tables in CG perform Join operations on bucket columns, local data Join can be directly performed to reduce data transmission time between nodes.

The data of a table will eventually fall into a barrel according to the barrel column value Hash and the number of barrels modeled. Assuming that the number of buckets in a table is 8, there are eight buckets `[0, 1, 2, 3, 4, 5, 6, 7]`Buckets'. We call such a sequence a `Buckets Sequence`. Each Bucket has one or more Tablets. When a table is a single partitioned table, there is only one Tablet in a Bucket. If it is a multi-partition table, there will be more than one.

In order for a table to have the same data distribution, the table in the same CG must ensure the following attributes are the same:

1. Barrel row and number of barrels

Bucket column, that is, the column specified in `DISTRIBUTED BY HASH (col1, col2,…)'in the table building statement. Bucket columns determine which column values are used to Hash data from a table into different Tablets. Tables in the same CG must ensure that the type and number of barrel columns are identical, and the number of barrels is identical, so that the data fragmentation of multiple tables can be controlled one by one.

2. Number of copies

The number of copies of all partitions of all tables in the same CG must be the same. If inconsistent, there may be a copy of a Tablet, and there is no corresponding copy of other table fragments on the same BE.

Tables in the same CG do not require consistency in the number, scope, and type of partition columns.

After fixing the number of bucket columns and buckets, the tables in the same CG will have the same Buckets Sequence. The number of replicas determines the number of replicas of Tablets in each bucket, which BE they are stored on. Suppose that Buckets Sequence is `[0, 1, 2, 3, 4, 5, 6, 7]`, and that BE nodes have `[A, B, C, D]`4. A possible distribution of data is as follows:

```
+---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 |
+---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
| A | | B | | C | | D | | A | | B | | C | | D |
|   | |   | |   | |   | |   | |   | |   | |   |
| B | | C | | D | | A | | B | | C | | D | | A |
|   | |   | |   | |   | |   | |   | |   | |   |
| C | | D | | A | | B | | C | | D | | A | | B |
+---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
```

The data of all tables in CG will be uniformly distributed according to the above rules, which ensures that the data with the same barrel column value are on the same BE node, and local data Join can be carried out.

Usage

Establishment of tables

When creating a table, you can specify the attribute `"colocate_with"="group_name"` in `PROPERTIES`, which means that the table is a Colocation Join table and belongs to a specified Colocation Group.

Examples:

```
CREATE TABLE tbl (k1 int, v1 int sum)
DISTRIBUTED BY HASH(k1)
BUCKETS 8
PROPERTIES(
  "colocate_with" = "group1"
);
```

If the specified group does not exist, Doris automatically creates a group that contains only the current table. If the Group already exists, Doris checks whether the current table satisfies the Colocation Group Schema. If satisfied, the table is created and added to the Group. At the same time, tables create fragments and replicas based on existing data distribution rules in Groups. Group belongs to a database, and its name is unique in a database. Internal storage is the full name of Group `dbId_groupName`, but users only perceive groupName.

In version 2.0, Doris supports cross-Database Group. When creating a table, you need to use the keyword `__global__` as a prefix of the Group name. like:

```
CREATE TABLE tbl (k1 int, v1 int sum)
DISTRIBUTED BY HASH(k1)
BUCKETS 8
```

```
PROPERTIES(
     "colocate_with" = "__global__group1"
);
```

The Group prefixed with `__global__` no longer belongs to a Database, and its name is also globally unique.

Cross-Database Colocate Join can be realized by creating a Global Group.

Delete table

When the last table in Group is deleted completely (deleting completely means deleting from the recycle bin). Usually, when a table is deleted by the `DROP TABLE` command, it will be deleted after the default one-day stay in the recycle bin, and the group will be deleted automatically.

View Group

The following command allows you to view the existing Group information in the cluster.

```
SHOW PROC '/colocation_group';

+-------------+--------------+--------------+------------+----------------+----------+----------+
| GroupId     | GroupName    | TableIds     | BucketsNum | ReplicationNum | DistCols | IsStable |
+-------------+--------------+--------------+------------+----------------+----------+----------+
| 10005.10008 | 10005_group1 | 10007, 10040 | 10         | 3              | int(11)  | true     |
+-------------+--------------+--------------+------------+----------------+----------+----------+
```

- GroupId: The unique identity of a group's entire cluster, with DB ID in the first half and group ID in the second half.
- GroupName: The full name of Group.
- Tablet Ids: The group contains a list of Tables'ID.
- Buckets Num: Number of barrels.
- Replication Num: Number of copies.
- DistCols: Distribution columns,
- IsStable: Is the group stable (for the definition of stability, see section `Collocation replica balancing and repair').

You can further view the data distribution of a group by following commands:

```
SHOW PROC '/colocation_group/10005.10008';

+-------------+---------------------+
| BucketIndex | BackendIds          |
+-------------+---------------------+
| 0           | 10004, 10002, 10001 |
| 1           | 10003, 10002, 10004 |
| 2           | 10002, 10004, 10001 |
| 3           | 10003, 10002, 10004 |
| 4           | 10002, 10004, 10003 |
| 5           | 10003, 10002, 10001 |
| 6           | 10003, 10004, 10001 |
| 7           | 10003, 10004, 10002 |
+-------------+---------------------+
```

- BucketIndex: Subscript to the bucket sequence.
- Backend Ids: A list of BE node IDs where data fragments are located in buckets.

> The above commands require ADMIN privileges. Normal user view is not supported at this time.

Modify Colocate Group

You can modify the Colocation Group property of a table that has been created. Examples:

```
ALTER TABLE tbl SET ("colocate_with" = "group2");
```

- If the table has not previously specified a Group, the command checks the Schema and adds the table to the Group (if the Group does not exist, it will be created).
- If other groups are specified before the table, the command first removes the table from the original group and adds a new group (if the group does not exist, it will be created).

You can also delete the Colocation attribute of a table by following commands:

```
ALTER TABLE tbl SET ("colocate_with" = "");
```

Other related operations

When an ADD PARTITION is added to a table with a Colocation attribute and the number of copies is modified, Doris checks whether the modification violates the Colocation Group Schema and rejects it if it does.

Colocation Duplicate Balancing and Repair

Copy distribution of Colocation tables needs to follow the distribution specified in Group, so it is different from common fragmentation in replica repair and balancing.

Group itself has a Stable attribute, when Stable is true, which indicates that all fragments of the table in the current Group are not changing, and the Colocation feature can be used normally. When Stable is false, it indicates that some tables in Group are being repaired or migrated. At this time, Colocation Join of related tables will degenerate into ordinary Join.

Replica Repair

Copies can only be stored on specified BE nodes. So when a BE is unavailable (downtime, Decommission, etc.), a new BE is needed to replace it. Doris will first look for the BE with the lowest load to replace it. After replacement, all data fragments on the old BE in the Bucket will be repaired. During the migration process, Group is marked Unstable.

Duplicate Equilibrium

Doris will try to distribute the fragments of the Collocation table evenly across all BE nodes. For the replica balancing of common tables, the granularity is single replica, that is to say, it is enough to find BE nodes with lower load for each replica alone. The equilibrium of the Colocation table is at the Bucket level, where all

replicas within a Bucket migrate together. We adopt a simple equalization algorithm, which distributes Buckets Sequence evenly on all BEs, regardless of the actual size of the replicas, but only according to the number of replicas. Specific algorithms can be referred to the code annotations in `ColocateTableBalancer.java`.

> Note 1: Current Colocation replica balancing and repair algorithms may not work well for heterogeneous deployed Oris clusters. The so-called heterogeneous deployment, that is, the BE node's disk capacity, number, disk type (SSD and HDD) is inconsistent. In the case of heterogeneous deployment, small BE nodes and large BE nodes may store the same number of replicas.
>
> Note 2: When a group is in an Unstable state, the Join of the table in it will degenerate into a normal Join. At this time, the query performance of the cluster may be greatly reduced. If you do not want the system to balance automatically, you can set the FE configuration item `disable_colocate_balance` to prohibit automatic balancing. Then open it at the right time. (See Section `Advanced Operations` for details)

Query

The Colocation table is queried in the same way as ordinary tables, and users do not need to perceive Colocation attributes. If the Group in which the Colocation table is located is in an Unstable state, it will automatically degenerate to a normal Join.

Examples are given to illustrate:

Table 1:

```
CREATE TABLE `tbl1` (
    `k1` date NOT NULL COMMENT "",
    `k2` int(11) NOT NULL COMMENT "",
    `v1` int(11) SUM NOT NULL COMMENT ""
) ENGINE=OLAP
AGGREGATE KEY(`k1`, `k2`)
PARTITION BY RANGE(`k1`)
(
    PARTITION p1 VALUES LESS THAN ('2019-05-31'),
    PARTITION p2 VALUES LESS THAN ('2019-06-30')
)
DISTRIBUTED BY HASH(`k2`) BUCKETS 8
PROPERTIES (
    "colocate_with" = "group1"
);
```

Table 2:

```
CREATE TABLE `tbl2` (
    `k1` datetime NOT NULL COMMENT "",
    `k2` int(11) NOT NULL COMMENT "",
    `v1` double SUM NOT NULL COMMENT ""
) ENGINE=OLAP
AGGREGATE KEY(`k1`, `k2`)
DISTRIBUTED BY HASH(`k2`) BUCKETS 8
PROPERTIES (
    "colocate_with" = "group1"
);
```

View the query plan:

```
DESC SELECT * FROM tbl1 INNER JOIN tbl2 ON (tbl1.k2 = tbl2.k2);

+-----------------------------------------------------+
| Explain String                                      |
+-----------------------------------------------------+
| PLAN FRAGMENT 0                                     |
|  OUTPUT EXPRS:`tbl1`.`k1` |                         |
|   PARTITION: RANDOM                                 |
|                                                     |
|   RESULT SINK                                       |
|                                                     |
|   2:HASH JOIN                                       |
|   |  join op: INNER JOIN                            |
|   |  hash predicates:                               |
|   |  colocate: true                                 |
|   |    `tbl1`.`k2` = `tbl2`.`k2`                    |
|   |  tuple ids: 0 1                                 |
|   |                                                 |
|   |----1:OlapScanNode                               |
|   |       TABLE: tbl2                               |
|   |       PREAGGREGATION: OFF. Reason: null         |
|   |       partitions=0/1                            |
|   |       rollup: null                              |
|   |       buckets=0/0                               |
|   |       cardinality=-1                            |
|   |       avgRowSize=0.0                            |
|   |       numNodes=0                                |
|   |       tuple ids: 1                              |
|   |                                                 |
|   0:OlapScanNode                                    |
|      TABLE: tbl1                                    |
|      PREAGGREGATION: OFF. Reason: No AggregateInfo  |
|      partitions=0/2                                 |
|      rollup: null                                   |
|      buckets=0/0                                    |
|      cardinality=-1                                 |
|      avgRowSize=0.0                                 |
|      numNodes=0                                     |
|      tuple ids: 0                                   |
+-----------------------------------------------------+
```

If Colocation Join works, the Hash Join Node will show `colocate: true`.

If not, the query plan is as follows:

```
+-----------------------------------------------------+
| Explain String                                      |
+-----------------------------------------------------+
| PLAN FRAGMENT 0                                     |
|  OUTPUT EXPRS:`tbl1`.`k1` |                         |
|   PARTITION: RANDOM                                 |
|                                                     |
|   RESULT SINK                                       |
|                                                     |
|   2:HASH JOIN                                       |
|   |  join op: INNER JOIN (BROADCAST)                |
|   |  hash predicates:                               |
|   |  colocate: false, reason: group is not stable   |
|   |    `tbl1`.`k2` = `tbl2`.`k2`                    |
|   |  tuple ids: 0 1                                 |
```

```
|   |                                               |
|   |----3:EXCHANGE                                 |
|   |       tuple ids: 1                            |
|   |                                               |
|   0:OlapScanNode                                  |
|      TABLE: tbl1                                  |
|      PREAGGREGATION: OFF. Reason: No AggregateInfo |
|      partitions=0/2                               |
|      rollup: null                                 |
|      buckets=0/0                                  |
|      cardinality=-1                               |
|      avgRowSize=0.0                               |
|      numNodes=0                                   |
|      tuple ids: 0                                 |
|                                                   |
| PLAN FRAGMENT 1                                   |
|  OUTPUT EXPRS:                                    |
|   PARTITION: RANDOM                               |
|                                                   |
|   STREAM DATA SINK                                |
|     EXCHANGE ID: 03                               |
|     UNPARTITIONED                                 |
|                                                   |
|   1:OlapScanNode                                  |
|      TABLE: tbl2                                  |
|      PREAGGREGATION: OFF. Reason: null            |
|      partitions=0/1                               |
|      rollup: null                                 |
|      buckets=0/0                                  |
|      cardinality=-1                               |
|      avgRowSize=0.0                               |
|      numNodes=0                                   |
|      tuple ids: 1                                 |
+---------------------------------------------------+
```

The HASH JOIN node displays the corresponding reason: `colocate: false, reason: group is not stable`. At the same time, an EXCHANGE node will be generated.

Advanced Operations

FE Configuration Item

- disable_colocate_relocate

Whether to close Doris's automatic Colocation replica repair. The default is false, i.e. not closed. This parameter only affects the replica repair of the Colocation table, but does not affect the normal table.

- disable_colocate_balance

Whether to turn off automatic Colocation replica balancing for Doris. The default is false, i.e. not closed. This parameter only affects the replica balance of the Collocation table, but does not affect the common table.

User can set these configurations at runtime. See `HELP ADMIN SHOW CONFIG;` and `HELP ADMIN SET CONFIG;`.

- disable_colocate_join

Whether to turn off the Colocation Join function or not. In 0.10 and previous versions, the default is true, that is, closed. In a later version, it will default to false, that is, open.

- use_new_tablet_scheduler

In 0.10 and previous versions, the new replica scheduling logic is incompatible with the Colocation Join function, so in 0.10 and previous versions, if `disable_colocate_join = false`, you need to set `use_new_tablet_scheduler = false`, that is, close the new replica scheduler. In later versions, `use_new_tablet_scheduler` will be equal to true.

###HTTP Restful API

Doris provides several HTTP Restful APIs related to Colocation Join for viewing and modifying Colocation Group.

The API is implemented on the FE side and accessed using `fe_host: fe_http_port`. ADMIN privileges are required.

1. View all Colocation information for the cluster

```
GET /api/colocate

Return the internal Colocation info in JSON format:

{
    "msg": "success",
  "code": 0,
  "data": {
    "infos": [
      ["10003.12002", "10003_group1", "10037, 10043", "1", "1", "int(11)", "true"]
    ],
    "unstableGroupIds": [],
    "allGroupIds": [{
      "dbId": 10003,
      "grpId": 12002
    }]
  },
  "count": 0
}
```

2. Mark Group as Stable or Unstable

- Mark as Stable

```
```
POST /api/colocate/group_stable?db_id=10005&group_id=10008

Returns: 200
```
```

- Mark as Unstable

```
```
DELETE /api/colocate/group_stable?db_id=10005&group_id=10008

Returns: 200
```
```

3. Setting Data Distribution for Group

The interface can force the number distribution of a group.

```
POST /api/colocate/bucketseq?db_id=10005&group_id=10008

Body:
[[10004,10002],[10003,10002],[10002,10004],[10003,10002],[10002,10004],[10003,10002],[10003,10004],[10003,10004],[10003,10004],
[10002,10004]]

Returns: 200
```

Body is a Buckets Sequence represented by a nested array and the ID of the BE where the fragments are distributed in each Bucket.

Note that using this command, you may need to set the FE configuration `disable_colocate_relocate` and `disable_colocate_balance` to true. That is to shut down the system for automatic Colocation replica repair and balancing. Otherwise, it may be automatically reset by the system after modification.

Runtime Filter

Runtime Filter

Runtime Filter is a new feature officially added in Doris 0.15. It is designed to dynamically generate filter conditions for certain Join queries at runtime to reduce the amount of scanned data, avoid unnecessary I/O and network transmission, and speed up the query.

It's design, implementation and effects, please refer to ISSUE 6116.

Noun Interpretation

- Left table: the table on the left during Join query. Perform Probe operation. The order can be adjusted by Join Reorder.
- Right table: the table on the right during Join query. Perform the Build operation. The order can be adjusted by Join Reorder.
- Fragment: FE will convert the execution of specific SQL statements into corresponding fragments and send them to BE for execution. The corresponding Fragment is executed on the BE, and the results are aggregated and returned to the FE.
- Join on clause: `Aa=Bb` in `A join B on Aa=Bb`, based on this to generate join conjuncts during query planning, including expr used by join Build and Probe, where Build expr is called in Runtime Filter src expr, Probe expr are called target expr in Runtime Filter.

Principle

Runtime Filter is generated during query planning, constructed in HashJoinNode, and applied in ScanNode.

For example, there is currently a Join query between the T1 table and the T2 table. Its Join mode is HashJoin. T1 is a fact table with 100,000 rows of data. T2 is a dimension table with 2000 rows of data. Doris join The actual situation is:

```
|          >      HashJoinNode     <
|         |                         |
|         | 100000                  | 2000
|         |                         |
|   OlapScanNode              OlapScanNode
|         ^                         ^
|         | 100000                  | 2000
|        T1                        T2
|
```

Obviously, scanning data for T2 is much faster than T1. If we take the initiative to wait for a while and then scan T1, after T2 sends the scanned data record to HashJoinNode, HashJoinNode calculates a filter condition based on the data of T2, such as the maximum value of T2 data And the minimum value, or build a Bloom Filter, and then send this filter condition to ScanNode waiting to scan T1, the latter applies this filter condition and delivers the filtered data to HashJoinNode, thereby reducing the number of probe hash tables and network overhead. This filter condition is Runtime Filter, and the effect is as follows:

```
|          >      HashJoinNode     <
|         |                         |
|         | 6000                    | 2000
|         |                         |
|   OlapScanNode              OlapScanNode
|         ^                         ^
|         | 100000                  | 2000
|        T1                        T2
|
```

If the filter condition (Runtime Filter) can be pushed down to the storage engine, in some cases, the index can be used to directly reduce the amount of scanned data, thereby greatly reducing the scanning time. The effect is as follows:

```
|          >      HashJoinNode     <
|         |                         |
|         | 6000                    | 2000
|         |                         |
|   OlapScanNode              OlapScanNode
|         ^                         ^
|         | 6000                    | 2000
|        T1                        T2
|
```

It can be seen that, unlike predicate push-down and partition cutting, Runtime Filter is a filter condition dynamically generated at runtime, that is, when the query is run, the join on clause is parsed to determine the filter expression, and the expression is broadcast to ScanNode that is reading the left table , Thereby reducing the amount of scanned data, thereby reducing the number of probe hash table, avoiding unnecessary I/O and network transmission.

Runtime Filter is mainly used to optimize joins for large tables. If the amount of data in the left table is too small, or the amount of data in the right table is too large, the Runtime Filter may not achieve the expected effect.

Usage

Runtime Filter query options

For query options related to Runtime Filter, please refer to the following sections:

- The first query option is to adjust the type of Runtime Filter used. In most cases, you only need to adjust this option, and keep the other options as default.
    - `runtime_filter_type`: Including Bloom Filter, MinMax Filter, IN predicate, IN_OR_BLOOM Filter and Bitmap_Filter. By default, only IN_OR_BLOOM Filter will be used. In some cases, the performance will be higher when both Bloom Filter, MinMax Filter and IN predicate are used at the same time.
- Other query options usually only need to be further adjusted in certain specific scenarios to achieve the best results. Usually only after performance testing, optimize for resource-intensive, long enough running time and high enough frequency queries.

- `runtime_filter_mode`: Used to adjust the push-down strategy of Runtime Filter, including three strategies of OFF, LOCAL, and GLOBAL. The default setting is the GLOBAL strategy
- `runtime_filter_wait_time_ms`: the time that ScanNode in the left table waits for each Runtime Filter, the default is 1000ms
- `runtime_filters_max_num`: The maximum number of Bloom Filters in the Runtime Filter that can be applied to each query, the default is 10
- `runtime_bloom_filter_min_size`: the minimum length of Bloom Filter in Runtime Filter, default 1048576 (1M)
- `runtime_bloom_filter_max_size`: the maximum length of Bloom Filter in Runtime Filter, the default is 16777216 (16M)
- `runtime_bloom_filter_size`: The default length of Bloom Filter in Runtime Filter, the default is 2097152 (2M)
- `runtime_filter_max_in_num`: If the number of rows in the right table of the join is greater than this value, we will not generate an IN predicate, the default is 1024

The query options are further explained below.

1.runtime_filter_type

Type of Runtime Filter used.

Type: Number (1, 2, 4, 8, 16) or the corresponding mnemonic string (IN, BLOOM_FILTER, MIN_MAX, IN_OR_BLOOM_FILTER, BITMAP_FILTER), the default is 8 (IN_OR_BLOOM FILTER), use multiple commas to separate, pay attention to the need to add quotation marks , Or add any number of types, for example:

```
set runtime_filter_type="BLOOM_FILTER,IN,MIN_MAX";
```

Equivalent to:

```
set runtime_filter_type=7;
```

Precautions for use

- **IN or Bloom Filter**: According to the actual number of rows in the right table during execution, the system automatically determines whether to use IN predicate or Bloom Filter.
  - By default, IN Predicate will be used when the number of data rows in the right table is less than 102400 (which can be adjusted by ` runtime_filter_max_in_num ’in the session variable). Otherwise, use bloom filter.
- **Bloom Filter**: There is a certain misjudgment rate, which results in the filtered data being a little less than expected, but it will not cause the final result to be inaccurate. In most cases, Bloom Filter can improve performance or has no significant impact on performance, but in some cases Under circumstances will cause performance degradation.
  - Bloom Filter construction and application overhead is high, so when the filtering rate is low, or the amount of data in the left table is small, Bloom Filter may cause performance degradation.

    - At present, only the Key column of the left table can be pushed down to the storage engine if the Bloom Filter is applied, and the test results show that the performance of the Bloom Filter is often reduced when the Bloom Filter is not pushed down to the storage engine.
    - Currently Bloom Filter only has short-circuit logic when using expression filtering on ScanNode, that is, when the false positive rate is too high, the Bloom Filter will not continue to be used, but there is no short-circuit logic when the Bloom Filter is pushed down to the storage engine , So when the filtration rate is low, it may cause performance degradation.
- **MinMax Filter**: Contains the maximum value and the minimum value, thereby filtering data smaller than the minimum value and greater than the maximum value. The filtering effect of the MinMax Filter is related to the type of the Key column in the join on clause and the data distribution of the left and right tables.
    - When the type of the Key column in the join on clause is int/bigint/double, etc., in extreme cases, if the maximum and minimum values of the left and right tables are the same, there is no effect, otherwise the maximum value of the right table is less than the minimum value of the left table, or the minimum of the right table The value is greater than the maximum value in the left table, the effect is best.
    - When the type of the Key column in the join on clause is varchar, etc., applying the MinMax Filter will often cause performance degradation.
- **IN predicate**: Construct IN predicate based on all the values of Key listed in the join on clause on the right table, and use the constructed IN predicate to filter on the left table. Compared with Bloom Filter, the cost of construction and application is lower. The amount of data in the right table is lower. When it is less, it tends to perform better.
    - Currently IN predicate already implement a merge method.
    - When IN predicate and other filters are specified at the same time, and the filtering value of IN predicate does not reach runtime_filter_max_in_num will try to remove other filters. The reason is that IN predicate is an accurate filtering condition. Even if there is no other filter, it can filter efficiently. If it is used at the same time, other filters will do useless work. Currently, only when the producer and consumer of the runtime filter are in the same fragment can there be logic to remove the Non-IN predicate.
- **Bitmap Filter**:
    - Currently, the bitmap filter is used only when the subquery in the in subquery operation returns a bitmap column.

2.runtime_filter_mode

Used to control the transmission range of Runtime Filter between instances.

Type: Number (0, 1, 2) or corresponding mnemonic string (OFF, LOCAL, GLOBAL), default 2 (GLOBAL).

Precautions for use

LOCAL: Relatively conservative, the constructed Runtime Filter can only be used in the same Fragment on the same instance (the smallest unit of query execution), that is, the Runtime Filter producer (the HashJoinNode that constructs the Filter) and the consumer (the ScanNode that uses the RuntimeFilter) The same Fragment, such as the general scene of broadcast join;

GLOBAL: Relatively radical. In addition to satisfying the scenario of the LOCAL strategy, the Runtime Filter can also be combined and transmitted to different Fragments on different instances via the network. For example,

the Runtime Filter producer and consumer are in different Fragments, such as shuffle join.

In most cases, the GLOBAL strategy can optimize queries in a wider range of scenarios, but in some shuffle joins, the cost of generating and merging Runtime Filters exceeds the performance advantage brought to the query, and you can consider changing to the LOCAL strategy.

If the join query involved in the cluster does not improve performance due to Runtime Filter, you can change the setting to OFF to completely turn off the function.

When building and applying Runtime Filters on different Fragments, the reasons and strategies for merging Runtime Filters can be found in ISSUE 6116

3.runtime_filter_wait_time_ms

Waiting for Runtime Filter is time consuming.

Type: integer, default 1000, unit ms

Precautions for use

After the Runtime Filter is turned on, the ScanNode in the table on the left will wait for a period of time for each Runtime Filter assigned to itself before scanning the data, that is, if the ScanNode is assigned 3 Runtime Filters, it will wait at most 3000ms.

Because it takes time to build and merge the Runtime Filter, ScanNode will try to push down the Runtime Filter that arrives within the waiting time to the storage engine. If the waiting time is exceeded, ScanNode will directly start scanning data using the Runtime Filter that has arrived.

If the Runtime Filter arrives after ScanNode starts scanning, ScanNode will not push the Runtime Filter down to the storage engine. Instead, it will use expression filtering on ScanNode based on the Runtime Filter for the data that has been scanned from the storage engine. The scanned data will not apply the Runtime Filter, so the intermediate data size obtained will be larger than the optimal solution, but serious cracking can be avoided.

If the cluster is busy and there are many resource-intensive or long-time-consuming queries on the cluster, consider increasing the waiting time to avoid missing optimization opportunities for complex queries. If the cluster load is light, and there are many small queries on the cluster that only take a few seconds, you can consider reducing the waiting time to avoid an increase of 1s for each query.

4.runtime_filters_max_num

The upper limit of the number of Bloom Filters in the Runtime Filter generated by each query.

Type: integer, default 10

Precautions for use Currently, only the number of Bloom Filters is limited, because the construction and application of Bloom Filters are more expensive than MinMax Filter and IN predicate.

If the number of Bloom Filters generated exceeds the maximum allowable number, then the Bloom Filter with a large selectivity is retained. A large selectivity means that more rows are expected to be filtered. This setting can prevent Bloom Filter from consuming too much memory overhead and causing potential problems.

```
Selectivity = (HashJoinNode Cardinality / HashJoinNode left child Cardinality)
- Because the cardinality of FE is currently inaccurate, the selectivity of Bloom Filter calculation here is inaccurate, so in the
end, it may only randomly reserve part of Bloom Filter.
```

This query option needs to be adjusted only when tuning some long-consuming queries involving joins between large tables.

5. Bloom Filter length related parameters

Including `runtime_bloom_filter_min_size`, `runtime_bloom_filter_max_size`, `runtime_bloom_filter_size`, used to determine the size (in bytes) of the Bloom Filter data structure used by the Runtime Filter.

Type: Integer

Precautions for use Because it is necessary to ensure that the length of the Bloom Filter constructed by each HashJoinNode is the same to be merged, the length of the Bloom Filter is currently calculated in the FE query planning.

If you can get the number of data rows (Cardinality) in the statistical information of the join right table, it will try to estimate the optimal size of the Bloom Filter based on Cardinality, and round to the nearest power of 2 (log value with the base 2). If the Cardinality of the table on the right cannot be obtained, the default Bloom Filter length `runtime_bloom_filter_size` will be used. `runtime_bloom_filter_min_size` and `runtime_bloom_filter_max_size` are used to limit the minimum and maximum length of the final Bloom Filter.

Larger Bloom Filters are more effective when processing high-cardinality input sets, but require more memory. If the query needs to filter high cardinality columns (for example, containing millions of different values), you can consider increasing the value of `runtime_bloom_filter_size` for some benchmark tests, which will help make the Bloom Filter filter more accurate, so as to obtain the expected Performance improvement.

The effectiveness of Bloom Filter depends on the data distribution of the query, so it is usually only for some specific queries to additionally adjust the length of the Bloom Filter, rather than global modification, generally only for some long time-consuming queries involving joins between large tables. Only when you need to adjust this query option.

View Runtime Filter generated by query

The query plan that can be displayed by the `explain` command includes the join on clause information used by each Fragment, as well as comments on the generation and use of the Runtime Filter by the Fragment, so as to confirm whether the Runtime Filter is applied to the desired join on clause. - The comment contained in the Fragment that generates the Runtime Filter, such as `runtime filters: filter_id[type] <- table.column`. - Use the comment contained in the fragment of Runtime Filter such as `runtime filters: filter_id[type] -> table.column`.

The query in the following example uses a Runtime Filter with ID RF000.

```
CREATE TABLE test (t1 INT) DISTRIBUTED BY HASH (t1) BUCKETS 2 PROPERTIES("replication_num" = "1");
INSERT INTO test VALUES (1), (2), (3), (4);

CREATE TABLE test2 (t2 INT) DISTRIBUTED BY HASH (t2) BUCKETS 2 PROPERTIES("replication_num" = "1");
INSERT INTO test2 VALUES (3), (4), (5);
```

```
EXPLAIN SELECT t1 FROM test JOIN test2 where test.t1 = test2.t2;
+-----------------------------------------------------------------------+
| Explain String                                                        |
+-----------------------------------------------------------------------+
| PLAN FRAGMENT 0                                                       |
|  OUTPUT EXPRS:`t1`                                                    |
|                                                                       |
|   4:EXCHANGE                                                          |
|                                                                       |
| PLAN FRAGMENT 1                                                       |
|  OUTPUT EXPRS:                                                        |
|   PARTITION: HASH_PARTITIONED: `default_cluster:ssb`.`test`.`t1`      |
|                                                                       |
|   2:HASH JOIN                                                         |
|   |  join op: INNER JOIN (BUCKET_SHUFFLE)                             |
|   |  equal join conjunct: `test`.`t1` = `test2`.`t2`                  |
|   |  runtime filters: RF000[in] <- `test2`.`t2`                       |
|   |                                                                   |
|   |----3:EXCHANGE                                                     |
|   |                                                                   |
|   0:OlapScanNode                                                      |
|      TABLE: test                                                      |
|      runtime filters: RF000[in] -> `test`.`t1`                        |
|                                                                       |
| PLAN FRAGMENT 2                                                       |
|  OUTPUT EXPRS:                                                        |
|   PARTITION: HASH_PARTITIONED: `default_cluster:ssb`.`test2`.`t2`     |
|                                                                       |
|   1:OlapScanNode                                                      |
|      TABLE: test2                                                     |
+-----------------------------------------------------------------------+
-- The line of `runtime filters` above shows that `2:HASH JOIN` of `PLAN FRAGMENT 1` generates IN predicate with ID RF000,
-- Among them, the key values of `test2`.`t2` are only known at runtime,
-- This IN predicate is used in `0:OlapScanNode` to filter unnecessary data when reading `test`.`t1`.

SELECT t1 FROM test JOIN test2 where test.t1 = test2.t2;
-- Return 2 rows of results [3, 4];

-- Through the query profile (set enable_profile=true;) you can view the detailed information of the internal work of the query,
-- Including whether each Runtime Filter is pushed down, waiting time,
-- and the total time from prepare to receiving Runtime Filter for OLAP_SCAN_NODE.
RuntimeFilter:in:
    -  HasPushDownToEngine:  true
    -  AWaitTimeCost:  0ns
    -  EffectTimeCost:  2.76ms

-- In addition, in the OLAP_SCAN_NODE of the profile, you can also view the filtering effect
-- and time consumption after the Runtime Filter is pushed down.
    -  RowsVectorPredFiltered:  9.320008M  (9320008)
    -  VectorPredEvalTime:  364.39ms
```

Runtime Filter planning rules

1. Only support the generation of Runtime Filter for the equivalent conditions in the join on clause, excluding the Null-safe condition, because it may filter out the null value of the join left table.
2. Does not support pushing down Runtime Filter to the left table of left outer, full outer, and anti join;
3. Does not support src expr or target expr is constant;
4. The equality of src expr and target expr is not supported;
5. The type of src expr is not supported to be equal to `HLL` or `BITMAP`;

6. Currently only supports pushing down Runtime Filter to OlapScanNode;
7. Target expr does not support NULL-checking expressions, such as `COALESCE/IFNULL/CASE`, because when the join on clause of other joins at the upper level of the outer join contains NULL-checking expressions and a Runtime Filter is generated, this Runtime Filter is downloaded Pushing to the left table of outer join may cause incorrect results;
8. The column (slot) in target expr is not supported, and an equivalent column cannot be found in the original table;
9. Column conduction is not supported. This includes two cases:
    - First, when the join on clause contains A.k = B.k and B.k = C.k, currently C.k can only be pushed down to B.k, but not to A.k;
    - Second, for example, the join on clause contains Aa + Bb = Cc. If Aa can be transmitted to Ba, that is, Aa and Ba are equivalent columns, then you can replace Aa with Ba, and then you can try to push the Runtime Filter down to B (If Aa and Ba are not equivalent columns, they cannot be pushed down to B, because target expr must be bound to the only join left table);
10. The types of Target expr and src expr must be equal, because Bloom Filter is based on hash, if the types are not equal, it will try to convert the type of target expr to the type of src expr;
11. The Runtime Filter generated by `PlanNode.Conjuncts` is not supported. Unlike HashJoinNode's `eqJoinConjuncts` and `otherJoinConjuncts`, the Runtime Filter generated by `PlanNode.Conjuncts` found in the test that it may cause incorrect results, such as `When an IN` subquery is converted to a join, the automatically generated join on clause will be stored in `PlanNode.Conjuncts`. At this time, applying Runtime Filter may result in missing some rows in the result.

Join Hint using Document

In the database, “Hint” is an instruction that instructs the query optimizer to execute a plan. By embedding hints in SQL statements, you can influence the optimizer’s decision to select the desired execution path. Here is a background example using Hint:

Suppose you have a table that contains a large amount of data, and you know that in some specific circumstances, the join order of the tables in a query may affect query performance. Leading Hint allows you to specify the order of table joins that you want the optimizer to follow.

For example, consider the following SQL query:

```
mysql> explain shape plan select * from t1 join t2 on t1.c1 = c2;
+----------------------------------------+
| Explain String                         |
+----------------------------------------+
| PhysicalResultSink                     |
| --PhysicalDistribute                   |
| ----PhysicalProject                    |
| ------hashJoin[INNER_JOIN](t1.c1 = t2.c2) |
| --------PhysicalOlapScan[t2]           |
| --------PhysicalDistribute             |
| ----------PhysicalOlapScan[t1]         |
+----------------------------------------+
7 rows in set (0.06 sec)
```

In the above example, when the execution efficiency is not ideal, we want to adjust the join order without changing the original sql so as to avoid affecting the user’s original scene and achieve the purpose of tuning. We can use leading to arbitrarily change the join order of tableA and tableB. For example, it could be written like:

```
mysql> explain shape plan select /*+ leading(t2 t1) */ * from t1 join t2 on c1 = c2;
+-----------------------------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                                                     |
+-----------------------------------------------------------------------------------------------------+
| PhysicalResultSink                                                                                  |
| --PhysicalDistribute                                                                                |
| ----PhysicalProject                                                                                 |
| ------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() build RFs:RF0 c1->[c2] |
| --------PhysicalOlapScan[t2] apply RFs: RF0                                                         |
| --------PhysicalDistribute                                                                          |
| ----------PhysicalOlapScan[t1]                                                                      |
|                                                                                                     |
| Hint log:                                                                                           |
| Used: leading(t2 t1)                                                                                |
| UnUsed:                                                                                             |
| SyntaxError:                                                                                        |
+-----------------------------------------------------------------------------------------------------+
12 rows in set (0.06 sec)
```

In this example, the Hint /+ *leading(t2 t1)* / is used. This Hint tells the optimizer to use the specified table (t2) as the driver table in the execution plan, before (t1).

This document mainly describes how to use join related hints in Doris: leading hint, ordered hint and distribute hint

Leading hint

Leading Hint is used to guide the optimizer in determining the join order of the query plan. In a query, the join order of the tables can affect query performance. Leading Hint allows you to specify the order of table joins that you want the optimizer to follow.

In doris, the syntax is /+*LEADING(tablespec [tablespec]…)* /,leading is surrounded by "/+*" and "*/" and placed directly behind the select in the select statement. Note that the '/' after leading and the selectlist need to be separated by at least one separator such as a space. At least two more tables need to be written before the leadinghint is justified. And any join can be bracketed to explicitly specify the shape of the joinTree. Example:

```
mysql> explain shape plan select /*+ leading(t2 t1) */ * from t1 join t2 on c1 = c2;
+------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                              |
+------------------------------------------------------------------------------+
| PhysicalResultSink                                                           |
| --PhysicalDistribute[DistributionSpecGather]                                 |
| ----PhysicalProject                                                          |
| ------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() |
| --------PhysicalOlapScan[t2]                                                 |
| --------PhysicalDistribute[DistributionSpecHash]                             |
| ----------PhysicalOlapScan[t1]                                               |
|                                                                              |
| Hint log:                                                                    |
| Used: leading(t2 t1)                                                         |
| UnUsed:                                                                      |
| SyntaxError:                                                                 |
+------------------------------------------------------------------------------+
12 rows in set (0.01 sec)
```

- If the leadinghint fails to take effect, the explain process will be used to generate the leadinghint. The EXPLAIN process will display whether the Leadinghint takes effect. There are three types of hints:
    - Used: leading hint takes effect normally
    - Unused: Unsupported cases include the feature that the leading specified join order is not equivalent to the original sql or is not supported in this version (see restrictions).
    - SyntaxError: indicates leading hint syntax errors, such as failure to find the corresponding table
- The leading hint syntax creates a left deep tree by default. For example, select /+ *leading(t1 t2 t3)* / * from t1 join t2 on… Specify by default

```
     join
    /    \
  join    t3
 /    \
t1    t2

mysql> explain shape plan select /*+ leading(t1 t2 t3) */ * from t1 join t2 on c1 = c2 join t3 on c2=c3;
+------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                              |
+------------------------------------------------------------------------------+
```

```
| PhysicalResultSink                                                                |
| --PhysicalDistribute[DistributionSpecGather]                                      |
| ----PhysicalProject                                                               |
| ------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()      |
| --------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=()    |
| ----------PhysicalOlapScan[t1]                                                    |
| ----------PhysicalDistribute[DistributionSpecHash]                               |
| ------------PhysicalOlapScan[t2]                                                  |
| --------PhysicalDistribute[DistributionSpecHash]                                  |
| ----------PhysicalOlapScan[t3]                                                    |
|                                                                                   |
| Hint log:                                                                         |
| Used: leading(t1 t2 t3)                                                           |
| UnUsed:                                                                           |
| SyntaxError:                                                                      |
+-----------------------------------------------------------------------------------+
15 rows in set (0.00 sec)
```

The join tree shape is also allowed to be specified using braces. For example: /+ *leading(t1 {t2 t3})* /

```
     join
    /    \
   t1    join
  /    \
  t2    t3

mysql> explain shape plan select /*+ leading(t1 {t2 t3}) */ * from t1 join t2 on c1 = c2 join t3 on c2=c3;
+-----------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                                   |
+-----------------------------------------------------------------------------------+
| PhysicalResultSink                                                                |
| --PhysicalDistribute[DistributionSpecGather]                                      |
| ----PhysicalProject                                                               |
| ------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=()      |
| --------PhysicalOlapScan[t1]                                                     |
| --------PhysicalDistribute[DistributionSpecHash]                                 |
| ----------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()  |
| ------------PhysicalOlapScan[t2]                                                  |
| ------------PhysicalDistribute[DistributionSpecHash]                             |
| --------------PhysicalOlapScan[t3]                                               |
|                                                                                   |
| Hint log:                                                                         |
| Used: leading(t1 { t2 t3 })                                                       |
| UnUsed:                                                                           |
| SyntaxError:                                                                      |
+-----------------------------------------------------------------------------------+
15 rows in set (0.02 sec)
```

- When a view is used as an alias in joinReorder, the corresponding view can be specified as the leading parameter. Example:

```
mysql> explain shape plan select /*+ leading(alias t1) */ count(*) from t1 join (select c2 from t2 join t3 on t2.c2 = t3.c3) as
alias on t1.c1 = alias.c2;
  +-----------------------------------------------------------------------------------+
  | Explain String(Nereids Planner)                                                   |
  +-----------------------------------------------------------------------------------+
  | PhysicalResultSink                                                                |
  | --hashAgg[GLOBAL]                                                                 |
  | ----PhysicalDistribute[DistributionSpecGather]                                    |
  | ------hashAgg[LOCAL]                                                              |
  | --------PhysicalProject                                                           |
```

```
  | ----------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = alias.c2)) otherCondition=()  |
  | ------------PhysicalProject                                                         |
  | --------------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=() |
  | ----------------PhysicalProject                                                     |
  | ------------------PhysicalOlapScan[t2]                                              |
  | ----------------PhysicalDistribute[DistributionSpecHash]                            |
  | ------------------PhysicalProject                                                   |
  | --------------------PhysicalOlapScan[t3]                                            |
  | ------------PhysicalDistribute[DistributionSpecHash]                                |
  | --------------PhysicalProject                                                       |
  | ----------------PhysicalOlapScan[t1]                                               |
  |                                                                                     |
  | Hint log:                                                                           |
  | Used: leading(alias t1)                                                             |
  | UnUsed:                                                                             |
  | SyntaxError:                                                                        |
  +-------------------------------------------------------------------------------------+
  21 rows in set (0.06 sec)
```

Basic cases

（Note that the column name is related to the table name, for example: only t1 has c1 field, the following example will write t1.c1 directly to c1 for simplicity）

```
CREATE DATABASE testleading;
USE testleading;

create table t1 (c1 int, c11 int) distributed by hash(c1) buckets 3 properties('replication_num' = '1');
create table t2 (c2 int, c22 int) distributed by hash(c2) buckets 3 properties('replication_num' = '1');
create table t3 (c3 int, c33 int) distributed by hash(c3) buckets 3 properties('replication_num' = '1');
create table t4 (c4 int, c44 int) distributed by hash(c4) buckets 3 properties('replication_num' = '1');
```

For a simple example, when we need to exchange the join order of t1 and t2, we only need to add leading(t2 t1) before it, which will be used in the explain Shows whether the hint is used. original plan

```
mysql> explain shape plan select * from t1 join t2 on t1.c1 = c2;
+-----------------------------------------+
| Explain String                          |
+-----------------------------------------+
| PhysicalResultSink                      |
| --PhysicalDistribute                    |
| ----PhysicalProject                     |
| ------hashJoin[INNER_JOIN](t1.c1 = t2.c2) |
| --------PhysicalOlapScan[t2]            |
| --------PhysicalDistribute              |
| ----------PhysicalOlapScan[t1]          |
+-----------------------------------------+
7 rows in set (0.06 sec)
```

Leading plan

```
mysql> explain shape plan select /*+ leading(t2 t1) */ * from t1 join t2 on c1 = c2;
+-------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                         |
+-------------------------------------------------------------------------+
| PhysicalResultSink                                                      |
| --PhysicalDistribute[DistributionSpecGather]                            |
| ----PhysicalProject                                                     |
```

```
| ------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() |
| --------PhysicalOlapScan[t2]                                                |
| --------PhysicalDistribute[DistributionSpecHash]                            |
| ----------PhysicalOlapScan[t1]                                              |
|                                                                             |
| Hint log:                                                                   |
| Used: leading(t2 t1)                                                        |
| UnUsed:                                                                     |
| SyntaxError:                                                                |
+-----------------------------------------------------------------------------+
12 rows in set (0.00 sec)
```

hint effect display （Used unused） If the leading hint has a syntax error, the corresponding information is displayed in the syntax Error when explaining, but the plan is generated as usual, just without leading

```
mysql> explain shape plan select /*+ leading(t2 t3) */ * from t1 join t2 on t1.c1 = c2;
+-----------------------------------------------------+
| Explain String                                      |
+-----------------------------------------------------+
| PhysicalResultSink                                  |
| --PhysicalDistribute                                |
| ----PhysicalProject                                 |
| ------hashJoin[INNER_JOIN](t1.c1 = t2.c2)           |
| --------PhysicalOlapScan[t1]                        |
| --------PhysicalDistribute                          |
| ----------PhysicalOlapScan[t2]                      |
|                                                     |
| Used:                                               |
| UnUsed:                                             |
| SyntaxError: leading(t2 t3) Msg:can not find table: t3 |
+-----------------------------------------------------+
11 rows in set (0.01 sec)
```

more cases

Left deep tree

leading generates a left deep tree by default when we don’t use any parentheses

```
mysql> explain shape plan select /*+ leading(t1 t2 t3) */ * from t1 join t2 on t1.c1 = c2 join t3 on c2 = c3;
+-------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                               |
+-------------------------------------------------------------------------------+
| PhysicalResultSink                                                            |
| --PhysicalDistribute[DistributionSpecGather]                                  |
| ----PhysicalProject                                                           |
| ------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()   |
| --------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() |
| ----------PhysicalOlapScan[t1]                                                |
| ----------PhysicalDistribute[DistributionSpecHash]                            |
| ------------PhysicalOlapScan[t2]                                              |
| --------PhysicalDistribute[DistributionSpecHash]                              |
| ----------PhysicalOlapScan[t3]                                                |
|                                                                               |
| Hint log:                                                                     |
| Used: leading(t1 t2 t3)                                                       |
| UnUsed:                                                                       |
| SyntaxError:                                                                  |
```

```
+--------------------------------------------------------------------------------+
15 rows in set (0.10 sec)
```

Right deep tree

When we want to make the shape of the plan into a right deep tree, bushy tree or zigzag tree, we only need to add curly brackets to limit the shape of the plan, instead of using swap to adjust from the left deep tree step by step like oracle.

```
mysql> explain shape plan select /*+ leading(t1 {t2 t3}) */ * from t1 join t2 on t1.c1 = c2 join t3 on c2 = c3;
+------------------------------------------------+
| Explain String                                 |
+------------------------------------------------+
| PhysicalResultSink                             |
| --PhysicalDistribute                           |
| ----PhysicalProject                            |
| ------hashJoin[INNER_JOIN](t1.c1 = t2.c2)      |
| --------PhysicalOlapScan[t1]                   |
| --------PhysicalDistribute                     |
| ----------hashJoin[INNER_JOIN](t2.c2 = t3.c3)  |
| ------------PhysicalOlapScan[t2]               |
| ------------PhysicalDistribute                 |
| --------------PhysicalOlapScan[t3]             |
|                                                |
| Used: leading(t1 { t2 t3 })                    |
| UnUsed:                                        |
| SyntaxError:                                   |
+------------------------------------------------+
14 rows in set (0.02 sec)
```

Bushy tree

```
mysql> explain shape plan select /*+ leading({t1 t2} {t3 t4}) */ * from t1 join t2 on t1.c1 = c2 join t3 on c2 = c3 join t4 on c3 =
c4;
+------------------------------------------------+
| Explain String                                 |
+------------------------------------------------+
| PhysicalResultSink                             |
| --PhysicalDistribute                           |
| ----PhysicalProject                            |
| ------hashJoin[INNER_JOIN](t2.c2 = t3.c3)      |
| --------hashJoin[INNER_JOIN](t1.c1 = t2.c2)    |
| ----------PhysicalOlapScan[t1]                 |
| ----------PhysicalDistribute                   |
| ------------PhysicalOlapScan[t2]               |
| --------PhysicalDistribute                     |
| ----------hashJoin[INNER_JOIN](t3.c3 = t4.c4)  |
| ------------PhysicalOlapScan[t3]               |
| ------------PhysicalDistribute                 |
| --------------PhysicalOlapScan[t4]             |
|                                                |
| Used: leading({ t1 t2 } { t3 t4 })             |
| UnUsed:                                        |
| SyntaxError:                                   |
+------------------------------------------------+
17 rows in set (0.02 sec)
```

Zig-Zag Tree

```
mysql> explain shape plan select /*+ leading(t1 {t2 t3} t4) */ * from t1 join t2 on t1.c1 = c2 join t3 on c2 = c3 join t4 on c3 =
c4;
+-----------------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                                         |
+-----------------------------------------------------------------------------------------+
| PhysicalResultSink                                                                      |
| --PhysicalDistribute[DistributionSpecGather]                                            |
| ----PhysicalProject                                                                     |
| ------hashJoin[INNER_JOIN] hashCondition=((t3.c3 = t4.c4)) otherCondition=()            |
| --------PhysicalDistribute[DistributionSpecHash]                                       |
| ----------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=()        |
| ------------PhysicalOlapScan[t1]                                                        |
| ------------PhysicalDistribute[DistributionSpecHash]                                   |
| --------------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()    |
| ----------------PhysicalOlapScan[t2]                                                   |
| ----------------PhysicalDistribute[DistributionSpecHash]                               |
| ------------------PhysicalOlapScan[t3]                                                 |
| --------PhysicalDistribute[DistributionSpecHash]                                       |
| ----------PhysicalOlapScan[t4]                                                          |
|                                                                                         |
| Hint log:                                                                               |
| Used: leading(t1 { t2 t3 } t4)                                                          |
| UnUsed:                                                                                 |
| SyntaxError:                                                                            |
+-----------------------------------------------------------------------------------------+
19 rows in set (0.02 sec)
```

Non-inner join:

When non-inner-joins are encountered, such as Outer join or semi/anti join, leading hint automatically deduces the join mode of each join according to the original sql semantics. If leading hints that differ from the original sql semantics or cannot be generated, they are placed in unused, but do not affect the generation of the normal flow of the plan.

Here are examples of things that can’t be exchanged:
——— test outer join which can not swap
– t1 leftjoin (t2 join t3 on (P23)) on (P12) != (t1 leftjoin t2 on (P12)) join t3 on (P23)

```
mysql> explain shape plan select /*+ leading(t1 {t2 t3}) */ * from t1 left join t2 on c1 = c2 join t3 on c2 = c3;
+-------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                               |
+-------------------------------------------------------------------------------+
| PhysicalResultSink                                                            |
| --PhysicalDistribute[DistributionSpecGather]                                  |
| ----PhysicalProject                                                           |
| ------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()   |
| --------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() |
| ----------PhysicalOlapScan[t1]                                                |
| ----------PhysicalDistribute[DistributionSpecHash]                            |
| ------------PhysicalOlapScan[t2]                                              |
| --------PhysicalDistribute[DistributionSpecHash]                              |
| ----------PhysicalOlapScan[t3]                                                |
|                                                                               |
| Hint log:                                                                     |
| Used:                                                                         |
| UnUsed: leading(t1 { t2 t3 })                                                 |
| SyntaxError:                                                                  |
```

```
+------------------------------------------------------------------------------+
15 rows in set (0.01 sec)
```

Below are some examples that can be exchanged and examples that cannot be exchanged, which readers can verify for themselves

```
-------- test outer join which can swap
-- (t1 leftjoin t2  on (P12)) innerjoin t3 on (P13) = (t1 innerjoin t3 on (P13)) leftjoin t2  on (P12)
explain shape plan select * from t1 left join t2 on c1 = c2 join t3 on c1 = c3;
explain shape plan select /*+ leading(t1 t3 t2) */ * from t1 left join t2 on c1 = c2 join t3 on c1 = c3;

-- (t1 leftjoin t2  on (P12)) leftjoin t3 on (P13) = (t1 leftjoin t3 on (P13)) leftjoin t2  on (P12)
explain shape plan select * from t1 left join t2 on c1 = c2 left join t3 on c1 = c3;
explain shape plan select /*+ leading(t1 t3 t2) */ * from t1 left join t2 on c1 = c2 left join t3 on c1 = c3;

-- (t1 leftjoin t2  on (P12)) leftjoin t3 on (P23) = t1 leftjoin (t2  leftjoin t3 on (P23)) on (P12)
select /*+ leading(t2 t3 t1) SWAP_INPUT(t1) */ * from t1 left join t2 on c1 = c2 left join t3 on c2 = c3;
explain shape plan select /*+ leading(t1 {t2 t3}) */ * from t1 left join t2 on c1 = c2 left join t3 on c2 = c3;
explain shape plan select /*+ leading(t1 {t2 t3}) */ * from t1 left join t2 on c1 = c2 left join t3 on c2 = c3;

-------- test outer join which can not swap
--  t1 leftjoin (t2  join t3 on (P23)) on (P12) != (t1 leftjoin t2  on (P12)) join t3 on (P23)
-- eliminated to inner join
explain shape plan select /*+ leading(t1 {t2 t3}) */ * from t1 left join t2 on c1 = c2 join t3 on c2 = c3;
explain graph select /*+ leading(t1 t2 t3) */ * from t1 left join (select * from t2 join t3 on c2 = c3) on c1 = c2;

-- test semi join
explain shape plan select * from t1 where c1 in (select c2 from t2);
explain shape plan select /*+ leading(t2 t1) */ * from t1 where c1 in (select c2 from t2);

-- test anti join
explain shape plan select * from t1 where exists (select c2 from t2);
```

View

In the case of aliases, you can specify the alias as a complete subtree with joinOrder generated from text order.

```
mysql>  explain shape plan select /*+ leading(alias t1) */ count(*) from t1 join (select c2 from t2 join t3 on t2.c2 = t3.c3) as
alias on t1.c1 = alias.c2;
+-----------------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                                         |
+-----------------------------------------------------------------------------------------+
| PhysicalResultSink                                                                      |
| --hashAgg[GLOBAL]                                                                       |
| ----PhysicalDistribute[DistributionSpecGather]                                          |
| ------hashAgg[LOCAL]                                                                    |
| --------PhysicalProject                                                                 |
| ----------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = alias.c2)) otherCondition=()  |
| ------------PhysicalProject                                                             |
| --------------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=() |
| ----------------PhysicalProject                                                        |
| ------------------PhysicalOlapScan[t2]                                                 |
| ----------------PhysicalDistribute[DistributionSpecHash]                               |
| ------------------PhysicalProject                                                      |
| --------------------PhysicalOlapScan[t3]                                               |
| ------------PhysicalDistribute[DistributionSpecHash]                                   |
| --------------PhysicalProject                                                          |
| ----------------PhysicalOlapScan[t1]                                                   |
|                                                                                         |
| Hint log:                                                                               |
```

```
| Used: leading(alias t1)                                                                |
| UnUsed:                                                                                |
| SyntaxError:                                                                           |
+----------------------------------------------------------------------------------------+
21 rows in set (0.02 sec)
```

mixed with ordered hint

When ordered hint is used together with ordered hint, ordered hint takes effect with a higher priority than leading hint. Example:

```
mysql>  explain shape plan select /*+ ORDERED LEADING(t1 t2 t3) */ t1.c1 from t2 join t1 on t1.c1 = t2.c2 join t3 on c2 = c3;
+----------------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                                        |
+----------------------------------------------------------------------------------------+
| PhysicalResultSink                                                                     |
| --PhysicalDistribute[DistributionSpecGather]                                           |
| ----PhysicalProject                                                                    |
| ------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()           |
| --------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=()         |
| ----------PhysicalProject                                                              |
| ------------PhysicalOlapScan[t2]                                                       |
| ----------PhysicalDistribute[DistributionSpecHash]                                     |
| ------------PhysicalProject                                                            |
| --------------PhysicalOlapScan[t1]                                                     |
| --------PhysicalDistribute[DistributionSpecHash]                                       |
| ----------PhysicalProject                                                              |
| ------------PhysicalOlapScan[t3]                                                       |
|                                                                                        |
| Hint log:                                                                              |
| Used: ORDERED                                                                          |
| UnUsed: leading(t1 t2 t3)                                                              |
| SyntaxError:                                                                           |
+----------------------------------------------------------------------------------------+
18 rows in set (0.02 sec)
```

Limitation

- The current version supports only one leadingHint. If leadinghint is used with a subquery, the query will report an error. Example (This example explain will report an error, but will follow the normal path generation plan) :

```
mysql>  explain shape plan select /*+ leading(alias t1) */ count(*) from t1 join (select /*+ leading(t3 t2) */ c2 from t2 join t3 on
t2.c2 = t3.c3) as alias on t1.c1 = alias.c2;
  +----------------------------------------------------------------------------------------+
  | Explain String(Nereids Planner)                                                        |
  +----------------------------------------------------------------------------------------+
  | PhysicalResultSink                                                                     |
  | --hashAgg[GLOBAL]                                                                      |
  | ----PhysicalDistribute[DistributionSpecGather]                                         |
  | ------hashAgg[LOCAL]                                                                   |
  | --------PhysicalProject                                                                |
  | ----------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = alias.c2)) otherCondition=()    |
  | ------------PhysicalProject                                                            |
  | --------------PhysicalOlapScan[t1]                                                     |
  | ------------PhysicalDistribute[DistributionSpecHash]                                   |
  | --------------PhysicalProject                                                          |
  | ----------------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=() |
```

```
  | ------------------PhysicalProject                                                  |
  | --------------------PhysicalOlapScan[t2]                                         |
  | ------------------PhysicalDistribute[DistributionSpecHash]                       |
  | --------------------PhysicalProject                                              |
  | ----------------------PhysicalOlapScan[t3]                                       |
  |                                                                                  |
  | Hint log:                                                                        |
  | Used:                                                                            |
  | UnUsed: leading(alias t1)                                                        |
  | SyntaxError: leading(t3 t2) Msg:one query block can only have one leading clause |
  +----------------------------------------------------------------------------------+
  21 rows in set (0.01 sec)
```

OrderedHint - Using ordered hint causes the join tree to be fixed in shape and displayed in text order - The syntax is /+ *ORDERED* /,leading is surrounded by “/+” *and* ”/” and placed directly behind the select in the select statement, for example:

```
  explain shape plan select /*+ ORDERED */ t1.c1 from t2 join t1 on t1.c1 = t2.c2 join t3 on c2 = c3;
     join
    /    \
   join    t3
  /    \
  t2    t1

mysql> explain shape plan select /*+ ORDERED */ t1.c1 from t2 join t1 on t1.c1 = t2.c2 join t3 on c2 = c3;
+------------------------------------------------------------------------------+
| Explain String(Nereids Planner)                                              |
+------------------------------------------------------------------------------+
| PhysicalResultSink                                                           |
| --PhysicalDistribute[DistributionSpecGather]                                 |
| ----PhysicalProject                                                          |
| ------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()  |
| --------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() |
| ----------PhysicalProject                                                    |
| ------------PhysicalOlapScan[t2]                                            |
| ----------PhysicalDistribute[DistributionSpecHash]                          |
| ------------PhysicalProject                                                  |
| --------------PhysicalOlapScan[t1]                                          |
| --------PhysicalDistribute[DistributionSpecHash]                            |
| ----------PhysicalProject                                                    |
| ------------PhysicalOlapScan[t3]                                            |
|                                                                              |
| Hint log:                                                                    |
| Used: ORDERED                                                                |
| UnUsed:                                                                      |
| SyntaxError:                                                                 |
+------------------------------------------------------------------------------+
18 rows in set (0.02 sec)
```

- When ordered hint and leading hint are used at the same time, the ordered hint prevails and the leading hint becomes invalid

```
mysql> explain shape plan select /*+ ORDERED LEADING(t1 t2 t3) */ t1.c1 from t2 join t1 on t1.c1 = t2.c2 join t3 on c2 = c3;
  +------------------------------------------------------------------------------+
  | Explain String(Nereids Planner)                                              |
  +------------------------------------------------------------------------------+
  | PhysicalResultSink                                                           |
  | --PhysicalDistribute[DistributionSpecGather]                                 |
  | ----PhysicalProject                                                          |
  | ------hashJoin[INNER_JOIN] hashCondition=((t2.c2 = t3.c3)) otherCondition=()  |
```

```
| --------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() |
| ----------PhysicalProject                                                      |
| ------------PhysicalOlapScan[t2]                                              |
| ----------PhysicalDistribute[DistributionSpecHash]                            |
| ------------PhysicalProject                                                    |
| --------------PhysicalOlapScan[t1]                                            |
| --------PhysicalDistribute[DistributionSpecHash]                              |
| ----------PhysicalProject                                                      |
| ------------PhysicalOlapScan[t3]                                              |
|                                                                                |
| Hint log:                                                                      |
| Used: ORDERED                                                                  |
| UnUsed: leading(t1 t2 t3)                                                      |
| SyntaxError:                                                                   |
+--------------------------------------------------------------------------------+
18 rows in set (0.02 sec)
```

DistributeHint

- Currently, only the distribute Type of the right table can be specified, and only two types are available: [shuffle] and [broadcast]. They are written before the right join table. Brackets and /+ / are allowed
- Currently you can use any DistributeHint
- When a DistributeHint whose plan cannot be correctly generated is not displayed, it takes effect based on maximum effort. Finally, the distribute mode that is displayed is mainly explained
- The current distribute version is not combined with leading. It takes effect only when the table specified in DISTRIBUTE is on the right of join.
- Mixed with ordered, the text order is used to set the join order and then specify the expected distribute mode in the corresponding join. Example:

Before use distribute hint:

```
mysql> explain shape plan select count(*) from t1 join t2 on t1.c1 = t2.c2;
  +--------------------------------------------------------------------------------+
  | Explain String(Nereids Planner)                                                |
  +--------------------------------------------------------------------------------+
  | PhysicalResultSink                                                             |
  | --hashAgg[GLOBAL]                                                              |
  | ----PhysicalDistribute[DistributionSpecGather]                                 |
  | ------hashAgg[LOCAL]                                                           |
  | --------PhysicalProject                                                        |
  | ----------hashJoin[INNER_JOIN] hashCondition=((t1.c1 = t2.c2)) otherCondition=() |
  | ------------PhysicalProject                                                    |
  | --------------PhysicalOlapScan[t1]                                            |
  | ------------PhysicalDistribute[DistributionSpecHash]                          |
  | --------------PhysicalProject                                                  |
  | ----------------PhysicalOlapScan[t2]                                          |
  +--------------------------------------------------------------------------------+
  11 rows in set (0.01 sec)
```

after use distribute hint: sql mysql> explain shape plan select /*+ ordered */ count(*) from t2 join[broadcast] t1 on t1.c1 = t2.c2; +--+ | Explain String(Nereids Planner) | +--+ | PhysicalResultSink | | --hashAgg[GLOBAL] | | ----PhysicalDistribute[DistributionSpecGather] | | -

```
-----hashAgg[LOCAL] | | --------PhysicalProject | | ----------hashJoin[INNER_JOIN] hashCondition=((t1.c1 =
t2.c2)) otherCondition=() | | ------------PhysicalProject | | --------------PhysicalOlapScan[t2] | | --------
----PhysicalDistribute[DistributionSpecReplicated] | | --------------PhysicalProject | | ---------------
PhysicalOlapScan[t1] | | | | Hint log: | | Used: ORDERED | | UnUsed: | | SyntaxError: | +--------------------
------------------------------------------------------------+ 16 rows in set (0.01 sec)
```

- the Explain shape plan will display inside distribute operator related information, including DistributionSpecReplicated said the operator will be corresponding data into a copy of all be node, DistributionSpecGather indicates that data is gathered to fe nodes, and DistributionSpecHash indicates that data is dispersed to different be nodes according to a specific hashKey and algorithm. ## To be supported - leadingHint indicates that the current query cannot be mixed with the subquery after it is promoted. A hint is required to control whether the subquery can be unnested

- A new distributeHint is needed for better and more complete control of the distribute operator
- Use leadingHint and distributeHint together to determine the shape of a join

Orthogonal BITMAP Calculation

Orthogonal BITMAP Calculation

Background

The original bitmap aggregate function designed by Doris is more general, but it has poor performance for the intersection and union of bitmap large cardinality above 100 million level. There are two main reasons for checking the bitmap aggregate function logic of the back-end be. First, when the bitmap cardinality is large, if the bitmap data size exceeds 1g, the network / disk IO processing time is relatively long; second, after the scan data, all the back-end be instances are transmitted to the top-level node for intersection and union operation, which brings pressure on the top-level single node and becomes the processing bottleneck.

The solution is to divide the bitmap column values according to the range, and the values of different ranges are stored in different buckets, so as to ensure that the bitmap values of different buckets are orthogonal and the data distribution is more uniform. In the case of query, the orthogonal bitmap in different buckets is firstly aggregated and calculated, and then the top-level node directly combines and summarizes the aggregated calculated values and outputs them. This will greatly improve the computing efficiency and solve the bottleneck problem of the top single node computing.

User guide

1. Create a table and add hid column to represent bitmap column value ID range as hash bucket column
2. Usage scenarios

Create table

We need to use the aggregation model when building tables. The data type is bitmap, and the aggregation function is bitmap_ union

```
CREATE TABLE `user_tag_bitmap` (
  `tag` bigint(20) NULL COMMENT "user tag",
  `hid` smallint(6) NULL COMMENT "Bucket ID",
  `user_id` bitmap BITMAP_UNION NULL COMMENT ""
) ENGINE=OLAP
AGGREGATE KEY(`tag`, `hid`)
COMMENT "OLAP"
DISTRIBUTED BY HASH(`hid`) BUCKETS 3
```

The HID column is added to the table schema to indicate the ID range as a hash bucket column.

Note: the HID number and buckets should be set reasonably, and the HID number should be set at least 5 times of buckets, so as to make the data hash bucket division as balanced as possible

Data Load

```
LOAD LABEL user_tag_bitmap_test
(
DATA INFILE('hdfs://abc')
INTO TABLE user_tag_bitmap
COLUMNS TERMINATED BY ','
(tmp_tag, tmp_user_id)
SET (
tag = tmp_tag,
hid = ceil(tmp_user_id/5000000),
user_id = to_bitmap(tmp_user_id)
)
)
...
```

Data format:

```
11111111,1
11111112,2
11111113,3
11111114,4
...
```

Note: the first column represents the user tags, which have been converted from Chinese into numbers

When loading data, vertically cut the bitmap value range of the user. For example, the hid value of the user ID in the range of 1-5000000 is the same, and the row with the same HID value will be allocated into a sub-bucket, so that the bitmap value in each sub-bucket is orthogonal. On the UDAF implementation of bitmap, the orthogonal feature of bitmap value in the bucket can be used to perform intersection union calculation, and the calculation results will be shuffled to the top node for aggregation.

Note: The orthogonal bitmap function cannot be used in the partitioned table. Because the partitions of the partitioned table are orthogonal, the data between partitions cannot be guaranteed to be orthogonal, so the calculation result cannot be estimated.

orthogonal_bitmap_intersect

The bitmap intersection function

Syntax:

orthogonal_bitmap_intersect(bitmap_column, column_to_filter, filter_values)

Parameters:

the first parameter is the bitmap column, the second parameter is the dimension column for filtering, and the third parameter is the variable length parameter, which means different values of the filter dimension column

Explain:

on the basis of this table schema, this function has two levels of aggregation in query planning. In the first layer, be nodes (update and serialize) first press filter_ Values are used to hash aggregate the keys, and then the bitmaps of all keys are intersected. The results are serialized and sent to the second level be nodes (merge

and finalize). In the second level be nodes, all the bitmap values from the first level nodes are combined circularly

Example:

```
select BITMAP_COUNT(orthogonal_bitmap_intersect(user_id, tag, 13080800, 11110200)) from user_tag_bitmap  where tag in (13080800, 11110200);
```

orthogonal_bitmap_intersect_count

To calculate the bitmap intersection count function, the syntax is the same as the original Intersect_Count, but the implementation is different

Syntax:

orthogonal_bitmap_intersect_count(bitmap_column, column_to_filter, filter_values)

Parameters:

The first parameter is the bitmap column, the second parameter is the dimension column for filtering, and the third parameter is the variable length parameter, which means different values of the filter dimension column

Explain:

on the basis of this table schema, the query planning aggregation is divided into two layers. In the first layer, be nodes (update and serialize) first press filter_ Values are used to hash aggregate the keys, and then the intersection of bitmaps of all keys is performed, and then the intersection results are counted. The count values are serialized and sent to the second level be nodes (merge and finalize). In the second level be nodes, the sum of all the count values from the first level nodes is calculated circularly

Example:

```
    select orthogonal_bitmap_intersect_count(members, tag_group, 1150000, 1150001, 390006) from tag_map where  tag_group in ( 1150000, 1150001, 390006);
```

orthogonal_bitmap_union_count

Figure out the bitmap union count function, syntax with the original bitmap_union_count, but the implementation is different.

Syntax:

orthogonal_bitmap_union_count(bitmap_column)

Explain:

on the basis of this table schema, this function is divided into two layers. In the first layer, be nodes (update and serialize) merge all the bitmaps, and then count the resulting bitmaps. The count values are serialized and sent to the second level be nodes (merge and finalize). In the second layer, the be nodes are used to calculate the sum of all the count values from the first level nodes

Example:

```
select ORTHOGONAL_BITMAP_UNION_COUNT(members) from tag_map where  tag_group in ( 1150000, 1150001, 390006);
```

orthogonal_bitmap_expr_calculate

Compute the function by computing the intersection, union and difference set of the expression bitmap.

Syntax:

orthogonal_bitmap_expr_calculate(bitmap_column, filter_column, input_string)

Parameters:

the first parameter is the Bitmap column, the second parameter is the dimension column used for filtering, that is, the calculated key column, and the third parameter is the calculation expression string, which means that the bitmap intersection and union difference expression is calculated according to the key column

the calculators supported by the expression: & represents intersection calculation, | represents union calculation, - represents difference calculation, ^ represents XOR calculation, and represents escape characters

Explain:

the aggregation of query planning is divided into two layers. The first layer of be aggregation node calculation includes init, update, and serialize steps. The second layer of be aggregation node calculation includes merge and finalize steps. In the first layer of be node, the input string is parsed in the init phase, which is converted into a suffix expression (inverse Polish), parses the calculated key value, and initializes it in the map<key, bitmap>structure; In the update phase, the underlying kernel scan dimension column (filter_column) calls back the update function, and then aggregates the bitmap in the map structure of the previous step in the unit of computing key; In the serialize stage, the bitmap of the key column is parsed according to the suffix expression, and the bitmap intersection, merge and difference set is calculated using the first in, last out principle of the stack structure. Then the final bitmap is serialized and sent to the aggregation be node in the second layer. Aggregates be nodes in the second layer, finds the union set of all bitmap values from the first layer nodes, and returns the final bitmap results

orthogonal_bitmap_expr_calculate_count

Compute the count function by computing the intersection, union and difference set of the expression bitmap. The syntax and parameters is the same as orthogonal_bitmap_expr_calculate

Syntax:

orthogonal_bitmap_expr_calculate_count(bitmap_column, filter_column, input_string)

Explain:

the aggregation of query planning is divided into two layers. The first layer of be aggregation node calculation includes init, update, and serialize steps. The second layer of be aggregation node calculation includes merge and finalize steps. In the first layer of be node, the input string is parsed in the init phase, converted to suffix

expression Formula (inverse Polish formula), parse the calculated key value and initialize it in the map<key, bitmap>structure; In the update phase, the underlying kernel scan dimension column (filter_column) calls back the update function, and then aggregates the bitmap in the map structure of the previous step in the unit of computing key; In the serialize stage, the bitmap of the key column is parsed according to the suffix expression, and the bitmap intersection, merge and difference set is calculated using the first in, last out principle of the stack structure. Then the count value of the final bitmap is serialized and sent to the aggregation be node in the second layer.> Aggregates be nodes in the second layer, adds and sums all count values from the first layer nodes, and returns the final count result.

Suitable for the scene

It is consistent with the scenario of orthogonal calculation of bitmap, such as calculation retention, funnel, user portrait, etc.

Crowd selection:

```
select orthogonal_bitmap_intersect_count(user_id, tag, 13080800, 11110200) from user_tag_bitmap where tag in (13080800, 11110200);

Note: 13080800 and 11110200 represent user labels
```

Calculate the deduplication value for user_id:

```
select orthogonal_bitmap_union_count(user_id) from user_tag_bitmap where tag in (13080800, 11110200);
```

Bitmap cross merge difference set hybrid computing:

```
select orthogonal_bitmap_expr_calculate_count(user_id, tag, '(833736|999777)&(1308083|231207)&(1000|20000-30000)') from
user_tag_bitmap where tag in (833736,999777,130808,231207,1000,20000,30000);
Note: 1000, 20000, 30000 plastic tags represent different labels of users
```

```
select orthogonal_bitmap_expr_calculate_count(user_id, tag, '(A:a/b|B:2\\-4)&(C:1-D:12)&E:23') from user_str_tag_bitmap where tag in
('A:a/b', 'B:2-4', 'C:1', 'D:12', 'E:23');
Note: 'A:a/b', 'B:2-4', etc. are string types tag, representing different labels of users, where 'B:2-4' needs to be escaped
as'B:2\\-4'
```

Approximate Deduplication Using HLL

HLL approximate deduplication

In actual business scenarios, with the increasing amount of business data, the pressure on data deduplication is also increasing. When the data reaches a certain scale, the cost of using accurate deduplication is also increasing. If it is acceptable, it is a very good way to achieve fast deduplication and reduce computational pressure through approximate algorithms. This article mainly introduces HyperLogLog (HLL for short) provided by Doris as an approximate deduplication algorithm.

The characteristic of HLL is that it has excellent space complexity O(mloglogn), time complexity is O(n), and the error of the calculation result can be controlled at about 1%-2%. The error is related to the size of the data set and the ha related to the Hierarchy function.

What is HyperLogLog

It is an upgraded version of the LogLog algorithm, and its role is to provide imprecise deduplication counts. Its mathematical basis is the **Bernoulli test**.

Assuming that the coin has both heads and tails, the probability of the coin being flipped up and down is 50%. Keep flipping the coin until it comes up heads, which we record as a full trial.

Then for multiple Bernoulli trials, assume that the number of times is n. It means that there are n times of heads. Suppose the number of tosses experienced per Bernoulli trial is k. For the first Bernoulli trial, the number of times is set to k1, and so on, the nth time corresponds to kn.

Among them, for these n Bernoulli trials, there must be a maximum number of tosses k. For example, after 12 tosses, a head will appear, then this is called k_max, which represents the maximum number of tosses.

Bernoulli's experiment can easily draw the following conclusions:

- No number of throws for n Bernoulli processes is greater than k_max.
- n Bernoulli processes with at least one throw equal to k_max

Finally, combined with the method of maximum likelihood estimation, it is found that there is an estimated correlation between n and k_max: n = 2^k_max. **When we only record k_max, we can estimate how many pieces of data there are in total, that is, the cardinality.**

Suppose the test results are as follows:

- 1st trial: 3 tosses before it turns heads, at this time k=3, n=1
- 2nd trial: Heads appear after 2 tosses, at this time k=2, n=2
- The 3rd trial: 6 tosses before the heads appear, at this time k=6, n=3
- The nth trial: it took 12 tosses to get heads, at this point we estimate, n = 2^12

Take the first three groups of experiments in the above example, then k_max = 6, and finally n=3, we put it into the estimation formula, obviously: 3 ≠ 2^6 . That is to say, when the number of trials is small, the error of this estimation method is very large.

These three sets of trials, we call one round of estimation. If only one round is performed, when n is large enough, the estimated error rate will be relatively reduced, but still not small enough.

Doris HLL Function

HLL is an engineering implementation based on the HyperLogLog algorithm. It is used to save the intermediate results of the HyperLogLog calculation process. It can only be used as the value column type of the table to continuously reduce the amount of data through aggregation.

To achieve the purpose of speeding up the query, based on it is an estimated result, the error is about 1%, the hll column is generated by other columns or the data in the imported data, and the hll_hash function is used when importing

To specify which column in the data is used to generate the hll column, it is often used to replace count distinct, and is used to quickly calculate uv in business by combining rollup

HLL_UNION_AGG(hll)

This function is an aggregate function that computes a cardinality estimate for all data that satisfies the condition.

HLL_CARDINALITY(hll)

This function is used to calculate the cardinality estimate for a single hll column

HLL_HASH(column_name)

Generate HLL column type for insert or import, see related instructions for import usage

How to use Doris HLL

1. When using HLL to deduplicate, you need to set the target column type to HLL and the aggregate function to HLL_UNION in the table creation statement
2. Columns of HLL type cannot be used as Key columns
3. The user does not need to specify the length and default value, the length is controlled within the system according to the degree of data aggregation

Create a table with hll column

```
create table test_hll(
    dt date,
    id int,
    name char(10),
    province char(10),
    os char(10),
    pv hll hll_union
)
Aggregate KEY (dt,id,name,province,os)
distributed by hash(id) buckets 10
PROPERTIES(
    "replication_num" = "1",
    "in_memory"="false"
);
```

Import Data

1. Stream load Import

```
curl --location-trusted -u root: -H "label:label_test_hll_load" \
    -H "column_separator:," \
    -H "columns:dt,id,name,province,os, pv=hll_hash(id)" -T test_hll.csv http://fe_IP:8030/api/demo/test_hll/_stream_load
```

The sample data is as follows（test_hll.csv）:

```
  2022-05-05,10001,Testing01,Beijing,Windows
  2022-05-05,10002,Testing01,Beijing,Linux
  2022-05-05,10003,Testing01,Beijing,MacOS
  2022-05-05,10004,Testing01,Hebei,Windows
  2022-05-06,10001,Testing01,Shanghai,Windows
  2022-05-06,10002,Testing01,Shanghai,Linux
  2022-05-06,10003,Testing01,Jiangsu,MacOS
  2022-05-06,10004,Testing01,Shaanxi,Windows
```

The import result is as follows:

```
# curl --location-trusted -u root: -H "label:label_test_hll_load"     -H "column_separator:,"     -H
"columns:dt,id,name,province,os, pv=hll_hash(id)" -T test_hll.csv http://127.0.0.1:8030/api/demo/test_hll/_stream_load

{
    "TxnId": 693,
    "Label": "label_test_hll_load",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 8,
    "NumberLoadedRows": 8,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 320,
    "LoadTimeMs": 23,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 1,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 9,
    "CommitAndPublishTimeMs": 11
}
```

2. Broker Load

```
LOAD LABEL demo.test_hlllabel
 (
    DATA INFILE("hdfs://hdfs_host:hdfs_port/user/doris_test_hll/data/input/file")
    INTO TABLE `test_hll`
    COLUMNS TERMINATED BY ","
    (dt,id,name,province,os)
    SET (
      pv = HLL_HASH(id)
    )
 );
```

Query data

HLL columns do not allow direct query of the original value, but can only be queried through the HLL aggregate function.

1. Find the total PV

```
mysql> select HLL_UNION_AGG(pv) from test_hll;
+---------------------+
| hll_union_agg(`pv`) |
+---------------------+
|                   4 |
+---------------------+
1 row in set (0.00 sec)
```

Equivalent to:

```
mysql> SELECT COUNT(DISTINCT pv) FROM test_hll;
+----------------------+
| count(DISTINCT `pv`) |
+----------------------+
|                    4 |
+----------------------+
1 row in set (0.01 sec)
```

2. Find the PV of each day

```
mysql> select HLL_UNION_AGG(pv) from test_hll group by dt;
+---------------------+
| hll_union_agg(`pv`) |
+---------------------+
|                   4 |
|                   4 |
+---------------------+
2 rows in set (0.01 sec)
```

High-Concurrency Point Query

High-Concurrency Point Query

> **Note**
>
> This feature is supported since version 2.0.0.

Background

Doris is built on a columnar storage format engine. In high-concurrency service scenarios, users always want to retrieve entire rows of data from the system. However, when tables are wide, the columnar format greatly amplifies random read IO. Doris query engine and planner are too heavy for some simple queries, such as point queries. A short path needs to be planned in the FE's query plan to handle such queries. FE is the access layer service for SQL queries, written in Java. Parsing and analyzing SQL also leads to high CPU overhead for high-concurrency queries. To solve these problems, we have introduced row storage, short query path, and PreparedStatement in Doris. Below is a guide to enable these optimizations.

Row Store Format

We support a row format for olap table to reduce point lookup io cost, but to enable this format, you need to spend more disk space for row format store. Currently, we store row in an extra column called `row column` for simplicity. The Row Storage mode can only be turned on when creating a table. You need to specify the following properties in the property of the table creation statement:

```
"store_row_column" = "true"
```

Accelerate point query for unique model

The above row storage is used to enable the Merge-On-Write strategy under the Unique model to reduce the IO overhead during enumeration. When `enable_unique_key_merge_on_write` and `store_row_column` are enabled when creating a Unique table, the query of the primary key will take a short path to optimize SQL execution, and only one RPC is required to complete the query. The following is an example of enabling the Merge-On-Write strategy under the Unique model by combining the query and row existence:

```
CREATE TABLE `tbl_point_query` (
    `k1` int(11) NULL,
    `v1` decimal(27, 9) NULL,
    `v2` varchar(30) NULL,
    `v3` varchar(30) NULL,
    `v4` date NULL,
    `v5` datetime NULL,
    `v6` float NULL,
    `v7` datev2 NULL
) ENGINE=OLAP
```

```
UNIQUE KEY(`k1`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`k1)` BUCKETS 1
PROPERTIES (
    "replication_allocation" = "tag.location.default: 1",
    "enable_unique_key_merge_on_write" = "true",
    "light_schema_change" = "true",
    "store_row_column" = "true"
);
```

Note: 1. `enable_unique_key_merge_on_write` should be enabled, since we need primary key for quick point lookup in storage engine 2. when condition only contains primary key like `select * from tbl_point_query where k1 = 123`, such query will go through the short fast path 3. `light_schema_change` should also been enabled since we rely on `column unique id` of each column when doing a point query. 4. It only supports equality queries on the key column of a single table and does not support joins or nested subqueries. The WHERE condition should consist of the key column alone and be an equality comparison. It can be considered as a type of key-value query.

Using `PreparedStatement`

In order to reduce CPU cost for parsing query SQL and SQL expressions, we provide `PreparedStatement` feature in FE fully compatible with mysql protocol (currently only support point queries like above mentioned).Enable it will pre caculate PreparedStatement SQL and expresions and caches it in a session level memory buffer and will be reused later on.We could improve 4x+ performance by using `PreparedStatement` when CPU became hotspot doing such queries.Bellow is an JDBC example of using `PreparedStatement`.

1. Setup JDBC url and enable server side prepared statement

```
url = jdbc:mysql://127.0.0.1:9030/ycsb?useServerPrepStmts=true
```

2. Using `PreparedStatement`

```
// use `?` for placement holders, readStatement should be reused
PreparedStatement readStatement = conn.prepareStatement("select * from tbl_point_query where k1 = ?");
...
readStatement.setInt(1,1234);
ResultSet resultSet = readStatement.executeQuery();
...
readStatement.setInt(1,1235);
resultSet = readStatement.executeQuery();
...
```

Enable row cache

Doris has a page-level cache that stores data for a specific column in each page. Therefore, the page cache is a column-based cache. For the row storage mentioned earlier, a row contains data for multiple columns, and the cache may be evicted by large queries, which can reduce the hit rate. To increase the hit rate of the row cache, a separate row cache is introduced, which reuses the LRU cache mechanism in Doris to ensure memory usage. You can enable it by specifying the following BE configuration:

- `disable_storage_row_cache` : Whether to enable the row cache. It is not enabled by default.

- `row_cache_mem_limit` : Specifies the percentage of memory occupied by the row cache. The default is 20% of memory.

Performance optimization

1. Generally, it is effective to improve query processing capabilities by increasing the number of Observers.
2. Query load balancing: During the enumeration, if it is found that the FE CPU that accepts enumeration requests is used too high, or the request response becomes slow, you can use jdbc load balance for load balancing, and distribute the requests to multiple nodes to share the pressure (and also You can use other methods for query load balancing configuration, such as Nginx, proxySQL)
3. By directing the query requests to the Observer role to share the request pressure of high-concurrency queries and reducing the number of query requests sent to the fe master, it can usually solve the problem of the time-consuming fluctuation of the Fe Master node query to obtain better performance and stability

QA

1. How to confirm that the configuration is correct and short path optimization using concurrent enumeration is used A: explain sql, when SHORT-CIRCUIT appears in the execution plan, it proves that short path optimization is used

```
mysql> explain select * from tbl_point_query where k1 = -2147481418 ;
      +-----------------------------------------------------------------------------------------------+
      | Explain String(Old Planner)                                                                   |
      +-----------------------------------------------------------------------------------------------+
      | PLAN FRAGMENT 0                                                                               |
      |   OUTPUT EXPRS:                                                                               |
      |     `test`.`tbl_point_query`.`k1`                                                             |
      |     `test`.`tbl_point_query`.`v1`                                                             |
      |     `test`.`tbl_point_query`.`v2`                                                             |
      |     `test`.`tbl_point_query`.`v3`                                                             |
      |     `test`.`tbl_point_query`.`v4`                                                             |
      |     `test`.`tbl_point_query`.`v5`                                                             |
      |     `test`.`tbl_point_query`.`v6`                                                             |
      |     `test`.`tbl_point_query`.`v7`                                                             |
      |   PARTITION: UNPARTITIONED                                                                    |
      |                                                                                               |
      |   HAS_COLO_PLAN_NODE: false                                                                   |
      |                                                                                               |
      |   VRESULT SINK                                                                                |
      |      MYSQL_PROTOCAL                                                                           |
      |                                                                                               |
      |   0:VOlapScanNode                                                                             |
      |      TABLE: test.tbl_point_query(tbl_point_query), PREAGGREGATION: ON                         |
      |      PREDICATES: `k1` = -2147481418 AND `test`.`tbl_point_query`.`__DORIS_DELETE_SIGN__` = 0 |
      |      partitions=1/1 (tbl_point_query), tablets=1/1, tabletList=360065                         |
      |      cardinality=9452868, avgRowSize=833.31323, numNodes=1                                    |
      |      pushAggOp=NONE                                                                           |
      |      SHORT-CIRCUIT                                                                            |
      +-----------------------------------------------------------------------------------------------+
```

2. How to confirm that prepared statement is effective A: After sending the request to Doris, find the corresponding query request in fe.audit.log and find Stmt=EXECUTE(), indicating that prepared statement is effective

```
2024-01-02 11:15:51,248 [query]
|Client=192.168.1.82:53450|User=root|Db=test|State=EOF|ErrorCode=0|ErrorMessage=|Time(ms)=49|ScanBytes=0|ScanRows=0|ReturnR
ows=1|StmtId=51|QueryId=b63d30b908f04dad-ab4a

3ba21d2c776b|IsQuery=true|isNereids=false|feIp=10.16.10.6|Stmt=EXECUTE(-2147481418)|CpuTimeMS=0|SqlHash=eee20fa2ac13a4f93bd
4503a87921024|peakMemoryBytes=0|SqlDigest=|TraceId=|WorkloadGroup=|FuzzyVaria
   bles=
```

3. Can non-primary key queries use special optimization of high-concurrency point lookups? A: No, high-concurrency query only targets the equivalent query of the key column, and the query cannot contain join or nested subqueries.

4. Is useServerPrepStmts useful in ordinary queries? A: Prepared Statement currently only takes effect when primary key is checked.

5. Does optimizer selection require global settings? A: When using prepared statement for query, Doris will choose the query method with the best performance, and there is no need to manually set the optimizer.

6. What should we do when the FE becomes a bottleneck?**

   A: If the FE is consuming too much CPU (i.e., high %CPU usage), enable the following configuration in the JDBC URL:

```
jdbc:mysql:loadbalance://[host1][:port],[host2][:port][,[host3][:port]]/${tbl_name}?
useServerPrepStmts=true&cachePrepStmts=true&prepStmtCacheSize=500&prepStmtCacheSqlLimit=1024
```

- Enable loadbalance to ensure multiple FEs can serve requests, and the more FE instances, the better (deploy one per instance).
- Enable useServerPrepStmts to reduce parsing and planning overhead on the FE.
- Enable cachePrepStmts so the client caches prepared statements, reducing the need to frequently send prepare requests to the FE.
- Adjust prepStmtCacheSize to set the maximum number of cached query templates.
- Adjust prepStmtCacheSqlLimit to set the maximum length of a single cached SQL template.

7. How to optimize query performance under a compute-storage separation architecture?**

   A:

- `set global enable_snapshot_point_query = false`. Point queries require an additional RPC to the meta service to obtain the version, which can easily become a bottleneck under high QPS. Setting it to false can speed up queries but reduces data visibility (requires a trade-off between performance and consistency).

- Configure the BE parameter enable_file_cache_keep_base_compaction_output=1 so that the result data after base compaction is stored in the cache, avoiding query jitter caused by remote access.

TOPN Query Optimization

TOPN queries refer to queries that involve ORDER BY LIMIT operations, which are common in log retrieval and other detailed query scenarios. Doris automatically optimizes this type of query.

```
SELECT * FROM tablex WHERE xxx ORDER BY c1,c2 ... LIMIT n
```

Optimization Points

1. During execution, dynamic range filters are built for the sorting columns (e.g., c1 >= 1000), which automatically apply the preceding conditions when reading data, leveraging zonemap indexes to filter out some rows or even entire files.
2. If the sorting fields c1, c2 are exactly the prefix of the table key, further optimization is applied. When reading data, only the header or tail of the data files is read, reducing the amount of data read to just the n rows needed.
3. SELECT * deferred materialization, during the data reading and sorting process, only the sorting columns are read, not the other columns. After obtaining the row numbers that meet the conditions, the entire data of those n rows needed is read, significantly reducing the amount of data read and sorted.

Limitations

1. It only applies to DUP and MOW tables, not to MOR and AGG tables.
2. Due to the high memory consumption on very large `n`, it will not take effect if n is greater than `topn_opt_limit_threshold`.

Configuration and Query Analysis

The following two parameters are session variables that can be set for a specific SQL or globally.

1. `topn_opt_limit_threshold`: This session variable determines whether TOPN optimization is applied. It defaults to 1024, and setting it to 0 disables the optimization.

2. `enable_two_phase_read_optimization`: This session variable determines whether to enable this optimization. It defaults to true, and setting it to false disables the optimization.

Checking if TOPN Query Optimization is Enabled

To confirm if TOPN query optimization is enabled for a particular SQL, you can use the `EXPLAIN` statement to get the query plan. An example is as follows:

- `TOPN OPT` indicates that optimization point 1 is applied.
- `VOlapScanNode` with `SORT LIMIT` indicates optimization point 2 is applied.
- `OPT TWO PHASE` indicates optimization point 3 is applied.

```
1:VTOP-N(137)
|   order by: @timestamp18 DESC
```

```
|   TOPN OPT
|   OPT TWO PHASE
|   offset: 0
|   limit: 10
|   distribute expr lists: applicationName5
|
0:VOlapScanNode(106)
   TABLE: log_db.log_core_all_no_index(log_core_all_no_index), PREAGGREGATION: ON
   SORT INFO:
        @timestamp18
   SORT LIMIT: 10
   TOPN OPT:1
   PREDICATES: ZYCFC-TRACE-ID4 like '%flowId-1720055220933%'
   partitions=1/8 (p20240704), tablets=250/250, tabletList=1727094,1727096,1727098 ...
   cardinality=345472780, avgRowSize=0.0, numNodes=1
   pushAggOp=NONE
```

Checking the Effectiveness of TOPN Query Optimization During Execution

First, set `topn_opt_limit_threshold` to 0 to disable TOPN query optimization and compare the execution time of the SQL with and without optimization enabled.

After enabling TOPN query optimization, search for `RuntimePredicate` in the query profile and focus on the following metrics:

- `RowsZonemapRuntimePredicateFiltered`: The number of rows filtered out, the higher the better.
- `NumSegmentFiltered`: The number of data files filtered out, the higher the better.
- `BlockConditionsFilteredZonemapRuntimePredicateTime`: The time taken to filter data, the lower the better.

Before version 2.0.3, the `RuntimePredicate` metrics were not separated out, and the `Zonemap` metrics can be used as a rough guide.

```
 SegmentIterator:
       -  BitmapIndexFilterTimer:  46.54us
       -  BlockConditionsFilteredBloomFilterTime:  10.352us
       -  BlockConditionsFilteredDictTime:  7.299us
       -  BlockConditionsFilteredTime:  202.23ms
       -  BlockConditionsFilteredZonemapRuntimePredicateTime:  0ns
       -  BlockConditionsFilteredZonemapTime:  402.917ms
       -  BlockInitSeekCount:  399
       -  BlockInitSeekTime:  11.309ms
       -  BlockInitTime:  215.59ms
       -  BlockLoadTime:  7s567ms
       -  BlocksLoad:  392.97K  (392970)
       -  CachedPagesNum:  0
       -  CollectIteratorMergeTime:  0ns
       -  CollectIteratorNormalTime:  0ns
       -  CompressedBytesRead:  29.76  MB
       -  DecompressorTimer:  427.713ms
       -  ExprFilterEvalTime:  3s930ms
       -  FirstReadSeekCount:  392.921K  (392921)
       -  FirstReadSeekTime:  528.287ms
       -  FirstReadTime:  1s134ms
       -  IOTimer:  51.286ms
       -  InvertedIndexFilterTime:  49.457us
       -  InvertedIndexQueryBitmapCopyTime:  0ns
       -  InvertedIndexQueryBitmapOpTime:  0ns
       -  InvertedIndexQueryCacheHit:  0
       -  InvertedIndexQueryCacheMiss:  0
```

```
          -  InvertedIndexQueryTime:  0ns
          -  InvertedIndexSearcherOpenTime:  0ns
          -  InvertedIndexSearcherSearchTime:  0ns
          -  LazyReadSeekCount:  0
          -  LazyReadSeekTime:  0ns
          -  LazyReadTime:  106.952us
          -  NumSegmentFiltered:  0
          -  NumSegmentTotal:  50
          -  OutputColumnTime:  61.987ms
          -  OutputIndexResultColumnTimer:  12.345ms
          -  RawRowsRead:  3.929151M  (3929151)
          -  RowsBitmapIndexFiltered:  0
          -  RowsBloomFilterFiltered:  0
          -  RowsConditionsFiltered:  6.38976M  (6389760)
          -  RowsDictFiltered:  0
          -  RowsInvertedIndexFiltered:  0
          -  RowsKeyRangeFiltered:  0
          -  RowsShortCircuitPredFiltered:  0
          -  RowsShortCircuitPredInput:  0
          -  RowsStatsFiltered:  6.38976M  (6389760)
          -  RowsVectorPredFiltered:  0
          -  RowsVectorPredInput:  0
          -  RowsZonemapRuntimePredicateFiltered:  6.38976M  (6389760)
          -  SecondReadTime:  0ns
          -  ShortPredEvalTime:  0ns
          -  TotalPagesNum:  2.301K  (2301)
          -  UncompressedBytesRead:  137.99  MB
          -  VectorPredEvalTime:  0ns
```

Statistics of query execution

Statistics of query execution

This document focuses on introducing the **Running Profile** which recorded runtime status of Doris in query execution. Using these statistical information, we can understand the execution of fragment to become a expert of Doris's **debugging and tuning**.

You can also refer to following statements to view profile in command line:

- SHOW QUERY PROFILE
- SHOW LOAD PROFILE

Noun Interpretation

- **FE**: Frontend, frontend node of Doris. Responsible for metadata management and request access.
- **BE**: Backend, backend node of Doris. Responsible for query execution and data storage.
- **Fragment**: FE will convert the execution of specific SQL statements into corresponding fragments and distribute them to BE for execution. BE will execute corresponding fragments and gather the result of RunningProfile to send back FE.

Basic concepts

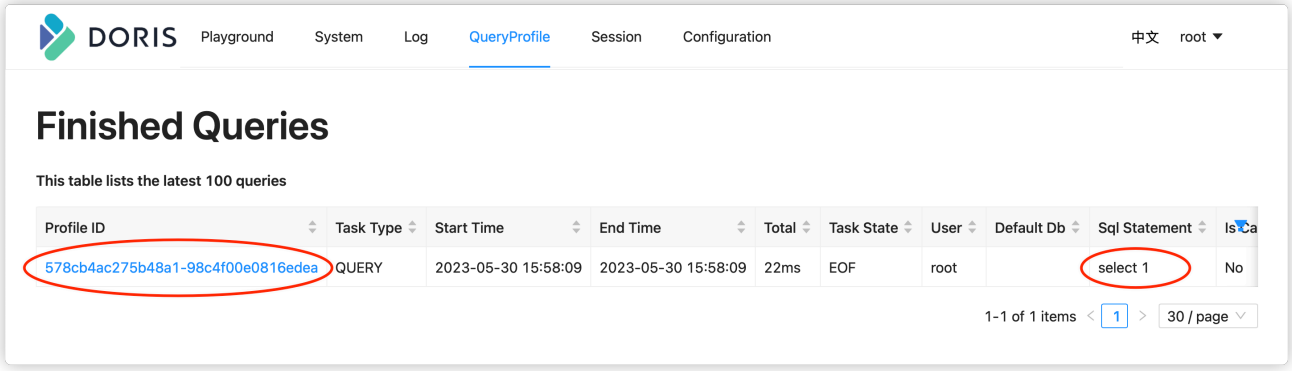
FE splits the query plan into fragments and distributes them to BE for task execution. BE records the statistics of **Running State** when executing fragment. BE print the outputs statistics of fragment execution into the log. FE can also collect these statistics recorded by each fragment and print the results on FE's web page.

Specific operation

Turn on the report switch on FE through MySQL command

```
mysql> set enable_profile=true;
```

After executing the corresponding SQL statement(`is_report_success` in old versions), we can see the report information of the corresponding SQL statement on the FE web page like the picture below.



The latest **100 statements** executed will be listed here. We can view detailed statistics of RunningProfile.

```
Query:
  Summary:
    Query ID: 9664061c57e84404-85ae111b8ba7e83a
    Start Time: 2020-05-02 10:34:57
    End Time: 2020-05-02 10:35:08
    Total: 10s323ms
    Query Type: Query
    Query State: EOF
    Doris Version: trunk
    User: root
    Default Db: default_cluster:test
    Sql Statement: select max(Bid_Price) from quotes group by Symbol
```

Here is a detailed list of `query ID`, `execution time`, `execution statement` and other summary information. The next step is to print the details of each fragment collected from be. `Fragment 0: Instance 9664061c57e84404-85ae111b8ba7e83d (host=TNetworkAddress(hostname:192.168.0.1, port:9060)):(Active: 10s270ms, % non-child: 0.14%) - MemoryLimit: 2.00 GB - BytesReceived: 168.08 KB - PeakUsedReservation: 0.00 - SendersBlockedTimer: 0ns - DeserializeRowBatchTimer: 501.975us - PeakMemoryUsage: 577.04 KB - RowsProduced: 8.322K (8322) EXCHANGE_NODE (id=4):(Active: 10s256ms, % non-child: 99.35%) - ConvertRowBatchTime: 180.171us - PeakMemoryUsage: 0.00 - RowsReturned: 8.322K (8322) - MemoryUsed: 0.00 - RowsReturnedRate: 811` The fragment ID is listed here; `hostname` show the be node executing the fragment; `active: 10s270ms`show the total execution time of the node; `non child: 0.14%` means the execution time of the execution node itself (not including the execution time of child nodes) as a percentage of the total time.

`PeakMemoryUsage` indicates the peak memory usage of `EXCHANGE_NODE`; `RowsReturned` indicates the number of rows returned by `EXCHANGE_NODE`; `RowsReturnedRate=RowsReturned/ActiveTime`; the meaning of these three statistics in other `NODE` the same.

Subsequently, the statistics of the child nodes will be printed in turn. **here you can distinguish the parent-child relationship by intent**.

Profile statistic analysis

There are many statistical information collected at BE. so we list the corresponding meanings of profile are below:

`Fragment`

- AverageThreadTokens: Number of threads used to execute fragment, excluding the usage of thread pool
- PeakReservation: Peak memory used by buffer pool
- MemoryLimit: Memory limit at query
- PeakMemoryUsage: Peak memory usage of instance
- RowsProduced: Number of rows that process

`BlockMgr`

- BlocksCreated: Number of Block be created by BlockMgr
- BlocksRecycled: Number of Block be recycled by BlockMgr
- BytesWritten: How many bytes be written to spill to disk
- MaxBlockSize: Max size of one Block
- TotalReadBlockTime: Total time read block from disk

`DataStreamSender`

- BytesSent: Total bytes data sent
- IgnoreRows: Rows filtered
- LocalBytesSent: The amount bytes of local node send to it's self during Exchange
- OverallThroughput: Total throughput = BytesSent / Time
- SerializeBatchTime: Sending data serialization time
- UncompressedRowBatchSize: Size of rowbatch before sending data compression

`ODBC_TABLE_SINK`

- NumSentRows: Total number of rows written to ODBC table
- TupleConvertTime: Time consuming of sending data serialization to insert statement
- ResultSendTime: Time consuming of writing through ODBC driver

`EXCHANGE_NODE`

- BytesReceived: Size of bytes received by network
- DataArrivalWaitTime: Total waiting time of sender to push data
- MergeGetNext: When there is a sort in the lower level node, exchange node will perform a unified merge sort and output an ordered result. This indicator records the total time consumption of merge sorting, including the time consumption of MergeGetNextBatch.
- MergeGetNextBatch: It takes time for merge node to get data. If it is single-layer merge sort, the object to get data is network queue. For multi-level merge sorting, the data object is child merger.
- ChildMergeGetNext: When there are too many senders in the lower layer to send data, single thread merge will become a performance bottleneck. Doris will start multiple child merge threads to do merge sort in parallel. The sorting time of child merge is recorded, which is the cumulative value of multiple threads.
- ChildMergeGetNextBatch: It takes time for child merge to get data, If the time consumption is too large, the bottleneck may be the lower level data sending node.

- FirstBatchArrivalWaitTime: The time waiting for the first batch come from sender
- DeserializeRowBatchTimer: Time consuming to receive data deserialization
- SendersBlockedTotalTimer(*): When the DataStreamRecv's queue buffer is full, wait time of sender
- ConvertRowBatchTime: Time taken to transfer received data to RowBatch
- RowsReturned: Number of receiving rows
- RowsReturnedRate: Rate of rows received

`SORT_NODE`

- InMemorySortTime: In memory sort time
- InitialRunsCreated: Number of initialize sort run
- MergeGetNext: Time cost of MergeSort from multiple sort_run to get the next batch (only show spilled disk)
- MergeGetNextBatch: Time cost MergeSort one sort_run to get the next batch (only show spilled disk)
- SortDataSize: Total sorted data
- TotalMergesPerformed: Number of external sort merges

`AGGREGATION_NODE`

- PartitionsCreated: Number of partition split by aggregate
- GetResultsTime: Time to get aggregate results from each partition
- HTResizeTime: Time spent in resizing hashtable
- HTResize: Number of times hashtable resizes
- HashBuckets: Number of buckets in hashtable
- HashBucketsWithDuplicate: Number of buckets with duplicate node in hashtable
- HashCollisions: Number of hash conflicts generated
- HashDuplicateNodes: Number of duplicate nodes with the same buckets in hashtable
- HashFailedProbe: Number of failed probe operations
- HashFilledBuckets: Number of buckets filled data
- HashProbe: Number of hashtable probe
- HashTravelLength: The number of steps moved when hashtable queries

`HASH_JOIN_NODE`

- ExecOption: The way to construct a HashTable for the right child (synchronous or asynchronous), the right child in Join may be a table or a subquery, the same is true for the left child
- BuildBuckets: The number of Buckets in HashTable
- BuildRows: the number of rows of HashTable
- BuildTime: Time-consuming to construct HashTable
- LoadFactor: Load factor of HashTable (ie the number of non-empty buckets)
- ProbeRows: Traverse the number of rows of the left child for Hash Probe
- ProbeTime: Time consuming to traverse the left child for Hash Probe, excluding the time consuming to call GetNext on the left child RowBatch
- PushDownComputeTime: The calculation time of the predicate pushdown condition
- PushDownTime: The total time consumed by the predicate push down. When Join, the right child who meets the requirements is converted to the left child's in query

`CROSS_JOIN_NODE`

- ExecOption: The way to construct RowBatchList for the right child (synchronous or asynchronous)
- BuildRows: The number of rows of RowBatchList (ie the number of rows of the right child)
- BuildTime: Time-consuming to construct RowBatchList
- LeftChildRows: the number of rows of the left child
- LeftChildTime: The time it takes to traverse the left child and find the Cartesian product with the right child, not including the time it takes to call GetNext on the left child RowBatch

`UNION_NODE`

- MaterializeExprsEvaluateTime: When the field types at both ends of the Union are inconsistent, the time spent to evaluates type conversion exprs and materializes the results

`ANALYTIC_EVAL_NODE`

- EvaluationTime: Analysis function (window function) calculation total time
- GetNewBlockTime: It takes time to apply for a new block during initialization. Block saves the cache line window or the entire partition for analysis function calculation
- PinTime: the time it takes to apply for a new block later or reread the block written to the disk back to the memory
- UnpinTime: the time it takes to flush the data of the block to the disk when the memory pressure of the block that is not in use or the current operator is high

`OLAP_SCAN_NODE`

The `OLAP_SCAN_NODE` is responsible for specific data scanning tasks. One `OLAP_SCAN_NODE` will generate one or more `OlapScanner`. Each Scanner thread is responsible for scanning part of the data.

Some or all of the predicate conditions in the query will be pushed to `OLAP_SCAN_NODE`. Some of these predicate conditions will continue to be pushed down to the storage engine in order to use the storage engine's index for data filtering. The other part will be kept in `OLAP_SCAN_NODE` to filter the data returned from the storage engine.

The profile of the `OLAP_SCAN_NODE` node is usually used to analyze the efficiency of data scanning. It is divided into three layers: `OLAP_SCAN_NODE`, `OlapScanner`, and `SegmentIterator` according to the calling relationship.

The profile of a typical `OLAP_SCAN_NODE` is as follows. Some indicators will have different meanings depending on the storage format (V1 or V2).

```
OLAP_SCAN_NODE (id=0):(Active: 1.2ms,% non-child: 0.00%)
  - BytesRead: 265.00 B                 # The amount of data read from the data file. Assuming that 10 32-bit integers are read, the
amount of data is 10 * 4B = 40 Bytes. This data only represents the fully expanded size of the data in memory, and does not
represent the actual IO size.
  - NumDiskAccess: 1                    # The number of disks involved in this ScanNode node.
  - NumScanners: 20                     # The number of Scanners generated by this ScanNode.
  - PeakMemoryUsage: 0.00               # Peak memory usage during query, not used yet
  - RowsRead: 7                         # The number of rows returned from the storage engine to the Scanner, excluding the number
of rows filtered by the Scanner.
  - RowsReturned: 7                     # The number of rows returned from ScanNode to the upper node.
  - RowsReturnedRate: 6.979K /sec       # RowsReturned/ActiveTime
```

```
  - TabletCount: 20                   # The number of Tablets involved in this ScanNode.
  - TotalReadThroughput: 74.70 KB/sec   # BytesRead divided by the total time spent in this node (from Open to Close). For IO
bounded queries, this should be very close to the total throughput of all the disks
  - ScannerBatchWaitTime: 426.886us     # To count the time the transfer thread waits for the scanner thread to return rowbatch. In
pipeline, this value is always 0.
  - ScannerWorkerWaitTime: 17.745us     # To count the time that the scanner thread waits for the available worker threads in the
thread pool.
  OlapScanner:
    - BlockConvertTime: 8.941us         # The time it takes to convert a vectorized Block into a RowBlock with a row structure. The
vectorized Block is VectorizedRowBatch in V1 and RowBlockV2 in V2.
    - BlockFetchTime: 468.974us         # Rowset Reader gets the time of the Block.
    - ReaderInitTime: 5.475ms           # The time when OlapScanner initializes Reader. V1 includes the time to form MergeHeap. V2
includes the time to generate various Iterators and read the first group of blocks.
    - RowsDelFiltered: 0                # Including the number of rows filtered out according to the Delete information in the
Tablet, and the number of rows filtered for marked deleted rows under the unique key model.
    - RowsPushedCondFiltered: 0         # Filter conditions based on the predicates passed down, such as the conditions passed from
BuildTable to ProbeTable in Join calculation. This value is not accurate, because if the filtering effect is poor, it will no longer
be filtered.
    - ScanTime: 39.24us                 # The time returned from ScanNode to the upper node.
    - ShowHintsTime_V1: 0ns             # V2 has no meaning. Read part of the data in V1 to perform ScanRange segmentation.
    SegmentIterator:
      - InvertedIndexFilterTimer: 779ns # Use inverted index to filter data time-consuming.
      - BlockLoadTime: 415.925us        # SegmentReader(V1) or SegmentIterator(V2) gets the time of the block.
      - BlockSeekCount: 12              # The number of block seeks when reading Segment.
      - BlockSeekTime: 222.556us        # It takes time to block seek when reading Segment.
      - BlocksLoad: 6                   # read the number of blocks
      - CachedPagesNum: 30              # In V2 only, when PageCache is enabled, the number of Pages that hit the Cache.
      - CompressedBytesRead: 0.00       # In V1, the size of the data read from the file before decompression. In V2, the pre-
compressed size of the read page that did not hit the PageCache.
      - DecompressorTimer: 0ns          # Data decompression takes time.
      - IOTimer: 0ns                    # IO time for actually reading data from the operating system.
      - IndexLoadTime_V1: 0ns           # Only in V1, it takes time to read Index Stream.
      - NumSegmentFiltered: 0           # When generating Segment Iterator, the number of Segments that are completely filtered out
through column statistics and query conditions.
      - NumSegmentTotal: 6              # Query the number of all segments involved.
      - RawRowsRead: 7                  # The number of raw rows read in the storage engine. See below for details.
      - RowsInvertedIndexFiltered: 0    # Only in V2, the number of rows filtered by the Inverted index.
      - RowsBloomFilterFiltered: 0      # Only in V2, the number of rows filtered by BloomFilter index.
      - RowsKeyRangeFiltered: 0         # In V2 only, the number of rows filtered out by SortkeyIndex index.
      - RowsStatsFiltered: 0            # In V2, the number of rows filtered by the ZoneMap index, including the deletion condition.
V1 also contains the number of rows filtered by BloomFilter.
      - RowsConditionsFiltered: 0       # Only in V2, the number of rows filtered by various column indexes.
      - RowsVectorPredFiltered: 0       # The number of rows filtered by the vectorized condition filtering operation.
      - TotalPagesNum: 30               # Only in V2, the total number of pages read.
      - UncompressedBytesRead: 0.00     # V1 is the decompressed size of the read data file (if the file does not need to be
decompressed, the file size is directly counted). In V2, only the decompressed size of the Page that missed PageCache is counted (if
the Page does not need to be decompressed, the Page size is directly counted)
      - VectorPredEvalTime: 0ns         # Time-consuming of vectorized condition filtering operation.
```

The predicate push down and index usage can be inferred from the related indicators of the number of data rows in the profile. The following only describes the profile in the reading process of segment V2 format data. In segment V1 format, the meaning of these indicators is slightly different.

- When reading a segment V2, if the query has key_ranges (the query range composed of prefix keys), first filter the data through the SortkeyIndex index, and the number of filtered rows is recorded in `RowsKeyRangeFiltered`.
- After that, use the Inverted index to perform precise filtering on the columns containing the inverted index in the query condition, and the number of filtered rows is recorded in `RowsInvertedIndexFiltered`.
- After that, according to the equivalent (eq, in, is) condition in the query condition, use the BloomFilter index to filter the data and record it in `RowsBloomFilterFiltered`. The value of `RowsBloomFilterFiltered` is

the difference between the total number of rows of the Segment (not the number of rows filtered by the Inverted index) and the number of remaining rows after BloomFilter, so the data filtered by BloomFilter may overlap with the data filtered by inverted index.
- After that, use the ZoneMap index to filter the data according to the query conditions and delete conditions and record it in `RowsStatsFiltered`.
- `RowsConditionsFiltered` is the number of rows filtered by various indexes, including the values of `RowsBloomFilterFiltered` and `RowsStatsFiltered`.
- So far, the Init phase is completed, and the number of rows filtered by the condition to be deleted in the Next phase is recorded in `RowsDelFiltered`. Therefore, the number of rows actually filtered by the delete condition are recorded in `RowsStatsFiltered` and `RowsDelFiltered` respectively.
- `RawRowsRead` is the final number of rows to be read after the above filtering.
- `RowsRead` is the number of rows finally returned to Scanner. `RowsRead` is usually smaller than `RawRowsRead`, because returning from the storage engine to the Scanner may go through a data aggregation. If the difference between `RawRowsRead` and `RowsRead` is large, it means that a large number of rows are aggregated, and aggregation may be time-consuming.
- `RowsReturned` is the number of rows finally returned by ScanNode to the upper node. `RowsReturned` is usually smaller than `RowsRead`. Because there will be some predicate conditions on the Scanner that are not pushed down to the storage engine, filtering will be performed once. If the difference between `RowsRead` and `RowsReturned` is large, it means that many rows are filtered in the Scanner. This shows that many highly selective predicate conditions are not pushed to the storage engine. The filtering efficiency in Scanner is worse than that in storage engine.

Through the above indicators, you can roughly analyze the number of rows processed by the storage engine and the size of the final filtered result row. Through the `Rows***Filtered` group of indicators, it is also possible to analyze whether the query conditions are pushed down to the storage engine, and the filtering effects of different indexes. In addition, a simple analysis can be made through the following aspects.

- Many indicators under `OlapScanner`, such as `IOTimer`, `BlockFetchTime`, etc., are the accumulation of all Scanner thread indicators, so the value may be relatively large. And because the Scanner thread reads data asynchronously, these cumulative indicators can only reflect the cumulative working time of the Scanner, and do not directly represent the time consumption of the ScanNode. The time-consuming ratio of ScanNode in the entire query plan is the value recorded in the `Active` field. Sometimes it appears that `IOTimer` has tens of seconds, but `Active` is actually only a few seconds. This situation is usually due to:
    - `IOTimer` is the accumulated time of multiple Scanners, and there are more Scanners.
    - The upper node is time-consuming. For example, the upper node takes 100 seconds, while the lower ScanNode only takes 10 seconds. The field reflected in `Active` may be only a few milliseconds. Because while the upper layer is processing data, ScanNode has performed data scanning asynchronously and prepared the data. When the upper node obtains data from ScanNode, it can obtain the prepared data, so the Active time is very short.
- `NumScanners` represents the number of Tasks submitted by the Scanner to the thread pool. It is scheduled by the thread pool in `RuntimeState`. The two parameters `doris_scanner_thread_pool_thread_num` and `doris_scanner_thread_pool_queue_size` control the size of the thread pool and the queue length respectively. Too many or too few threads will affect query efficiency. At the same time, some summary indicators can be divided by the number of threads to roughly estimate the time consumption of each thread.
- `TabletCount` indicates the number of tablets to be scanned. Too many may mean a lot of random read and data merge operations.

- `UncompressedBytesRead` indirectly reflects the amount of data read. If the value is large, it means that there may be a lot of IO operations.
- `CachedPagesNum` and `TotalPagesNum` can check the hitting status of PageCache. The higher the hit rate, the less time-consuming IO and decompression operations.

`Buffer pool`

- AllocTime: Memory allocation time
- CumulativeAllocationBytes: Cumulative amount of memory allocated
- CumulativeAllocations: Cumulative number of memory allocations
- PeakReservation: Peak of reservation
- PeakUnpinnedBytes: Amount of memory data of unpin
- PeakUsedReservation: Peak usage of reservation
- ReservationLimit: Limit of reservation of bufferpool

Query Analysis

Query Analysis

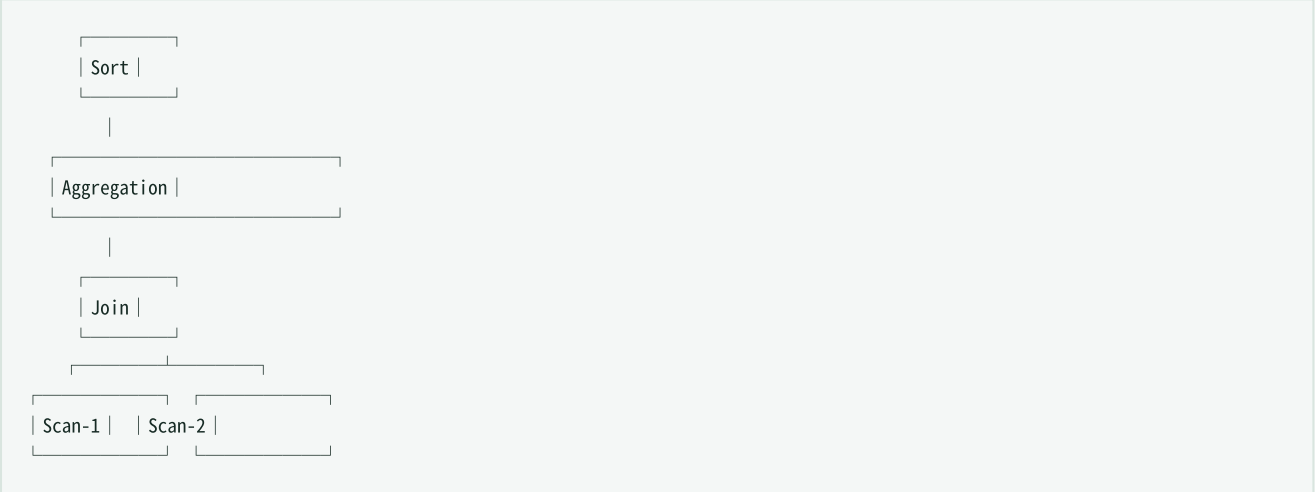
Doris provides a graphical command to help users analyze a specific query or import more easily. This article describes how to use this feature.

Query Plan Tree

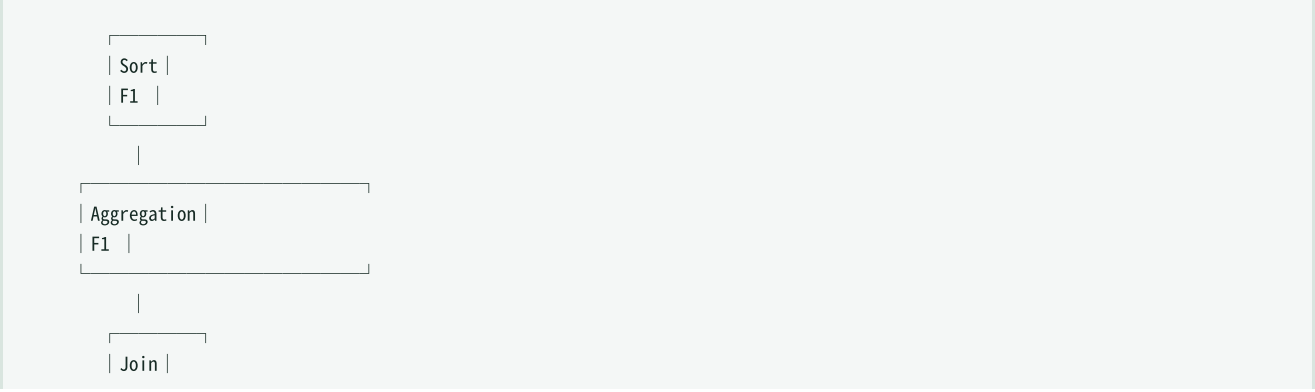
SQL is a descriptive language, and users describe the data they want to get through a SQL. The specific execution mode of a SQL depends on the implementation of the database. The query planner is used to determine how the database executes a SQL.

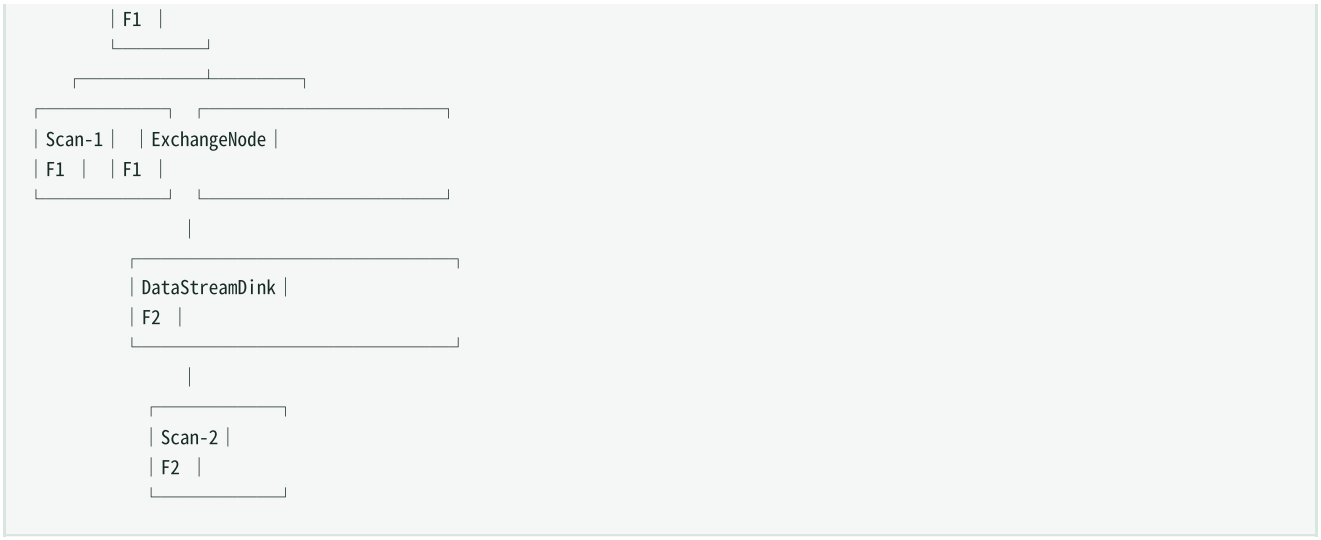
For example, if the user specifies a Join operator, the query planner needs to decide the specific Join algorithm, such as Hash Join or Merge Sort Join; whether to use Shuffle or Broadcast; whether the Join order needs to be adjusted to avoid Cartesian product; on which nodes to execute and so on.

Doris' query planning process is to first convert an SQL statement into a single-machine execution plan tree.



After that, the query planner will convert the single-machine query plan into a distributed query plan according to the specific operator execution mode and the specific distribution of data. The distributed query plan is composed of multiple fragments, each fragment is responsible for a part of the query plan, and the data is transmitted between the fragments through the ExchangeNode operator.





As shown above, we divided the stand-alone plan into two Fragments: F1 and F2. Data is transmitted between two Fragments through an ExchangeNode.

And a Fragment will be further divided into multiple Instances. Instance is the final concrete execution instance. Dividing into multiple Instances helps to make full use of machine resources and improve the execution concurrency of a Fragment.

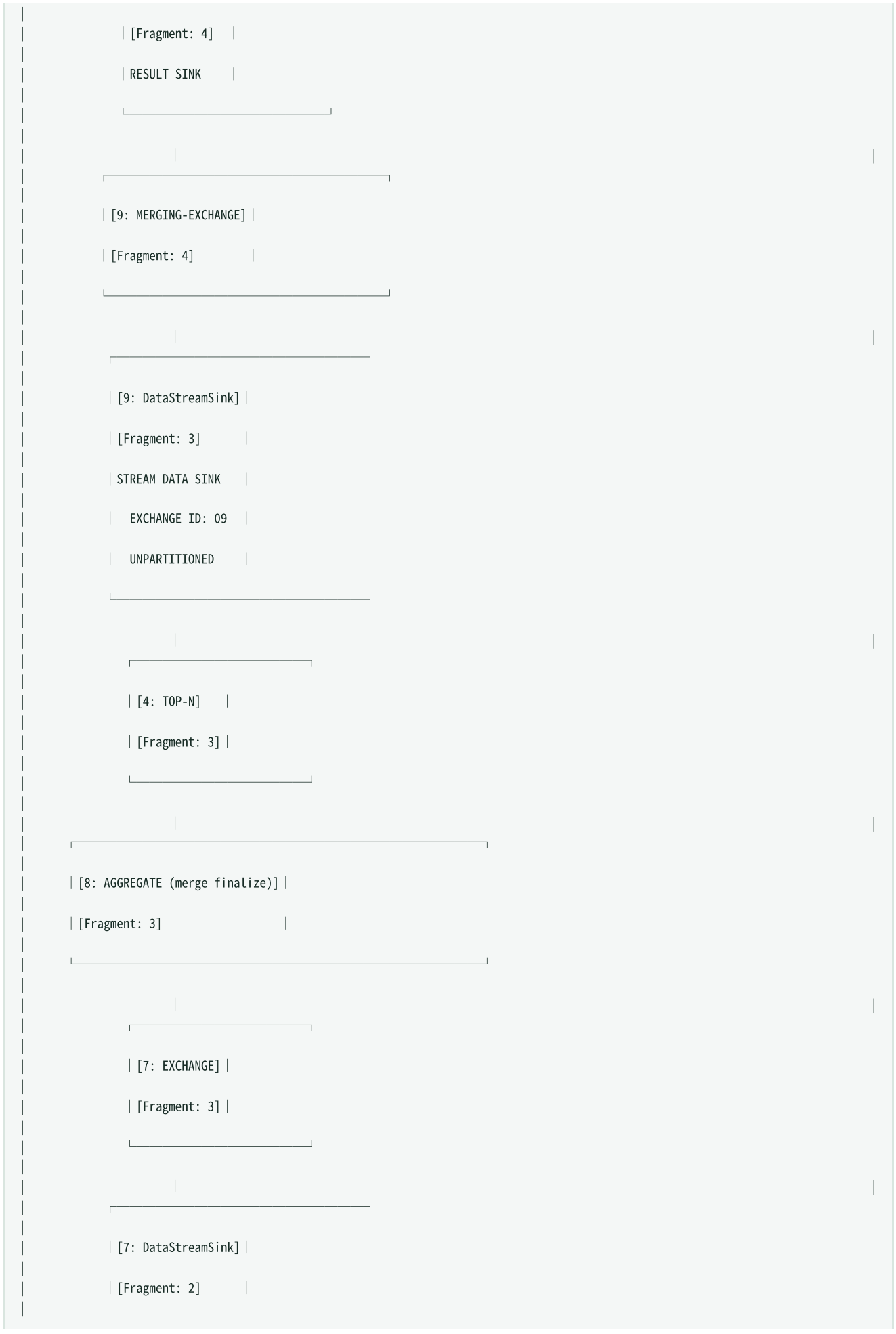
View Query Plan

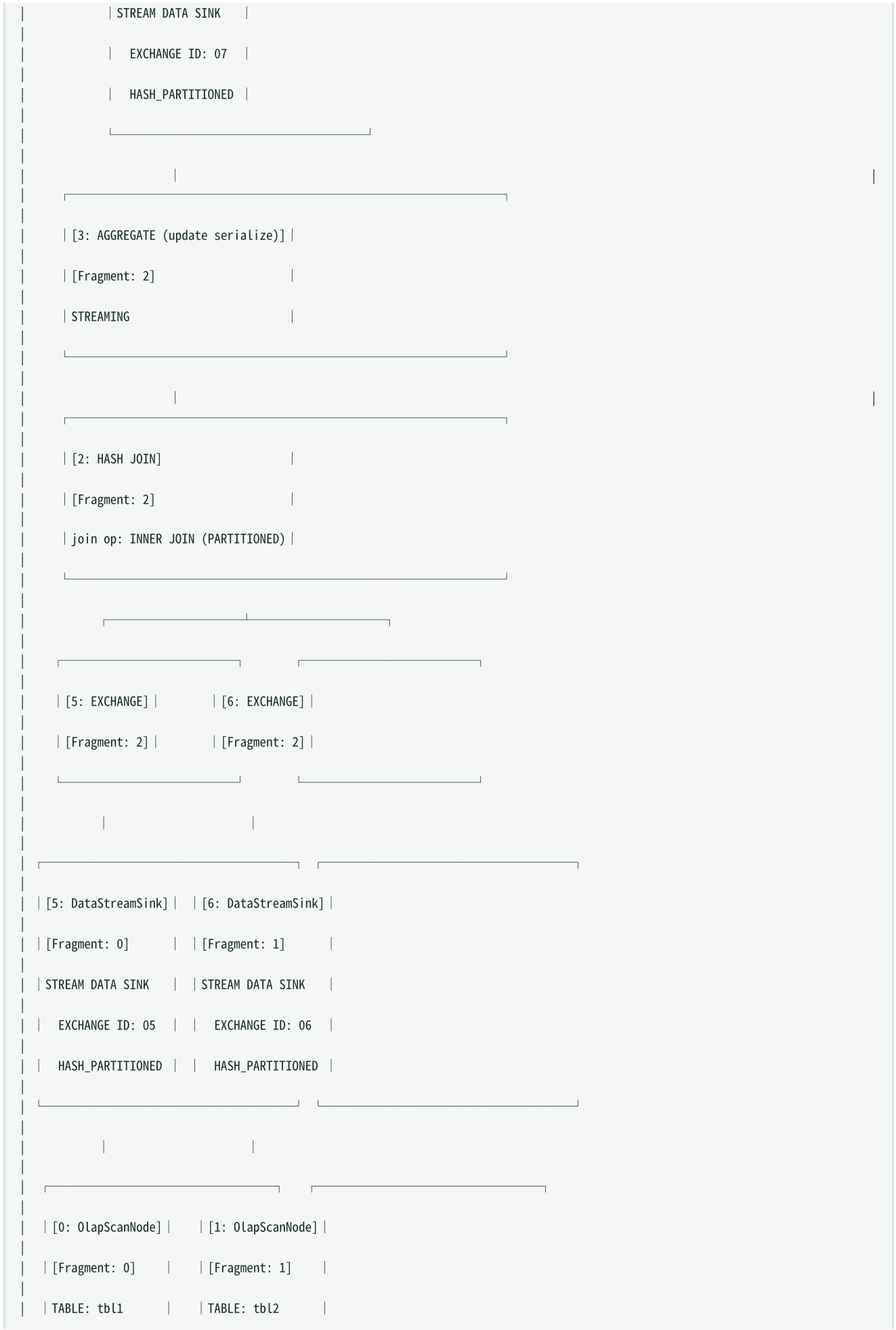
You can view the execution plan of a SQL through the following three commands.

- `EXPLAIN GRAPH select ...;` OR `DESC GRAPH select ...;` These commands provide a graphical representation of the execution plan.
- `EXPLAIN select ...;` This command displays a textual representation of the execution plan for the specified SQL query.
- `EXPLAIN VERBOSE select ...;` Similar to the previous command, this command provides a more detailed output.
- `EXPLAIN PARSED PLAN select ...;` This command returns the parsed execution plan of the SQL query. It displays the plan trees and information about the logical operators involved in query processing.
- `EXPLAIN ANALYZED PLAN select ...;` This command returns the analyzed execution plan for the SQL query.
- `EXPLAIN REWRITTEN PLAN select ...;` This command shows the rewritten execution plan after applying any query transformations or optimizations performed by the database engine.
- `EXPLAIN OPTIMIZED PLAN select ...;` This command shows the best execution plan after CBO.
- `EXPLAIN SHAPE PLAN select ...;` This command presents the simplified execution plan with a focus on how the query is shaped and structured.

The first command displays a query plan graphically. This command can more intuitively display the tree structure of the query plan and the division of Fragments:

```
mysql> explain graph select tbl1.k1, sum(tbl1.k2) from tbl1 join tbl2 on tbl1.k1 = tbl2.k1 group by tbl1.k1 order by tbl1.k1;
+------------------------------------------------------------------------------------------------------------------------------------------------+
| Explain String                                                                                                                                 |
+------------------------------------------------------------------------------------------------------------------------------------------------+
|                                                                                                                                                |
|              ┌-----------------------------┐
|
|              |[9: ResultSink]|
```


[Fragment: 4]
RESULT SINK
[9: MERGING-EXCHANGE]
[Fragment: 4]
[9: DataStreamSink]
[Fragment: 3]
STREAM DATA SINK
EXCHANGE ID: 09
UNPARTITIONED
[4: TOP-N]
[Fragment: 3]
[8: AGGREGATE (merge finalize)]
[Fragment: 3]
[7: EXCHANGE]
[Fragment: 3]
[7: DataStreamSink]
[Fragment: 2]


STREAM DATA SINK
EXCHANGE ID: 07
HASH_PARTITIONED
[3: AGGREGATE (update serialize)]
[Fragment: 2]
STREAMING
[2: HASH JOIN]
[Fragment: 2]
join op: INNER JOIN (PARTITIONED)
[5: EXCHANGE]
[Fragment: 2]
[6: EXCHANGE]
[Fragment: 2]
[5: DataStreamSink]
[Fragment: 0]
STREAM DATA SINK
EXCHANGE ID: 05
HASH_PARTITIONED
[6: DataStreamSink]
[Fragment: 1]
STREAM DATA SINK
EXCHANGE ID: 06
HASH_PARTITIONED
[0: OlapScanNode]
[Fragment: 0]
TABLE: tbl1
[1: OlapScanNode]
[Fragment: 1]
TABLE: tbl2

```
|
|   └─────────────────────────────────┘    └─────────────────────────────────┘
|
+-----------------------------------------------------------------------------------------------------------------------+
```

As can be seen from the figure, the query plan tree is divided into 5 fragments: 0, 1, 2, 3, and 4. For example, `[Fragment: 0]` on the `OlapScanNode` node indicates that this node belongs to Fragment 0. Data is transferred between each Fragment through DataStreamSink and ExchangeNode.

The graphics command only displays the simplified node information. If you need to view more specific node information, such as the filter conditions pushed to the node as follows, you need to view the more detailed text version information through the second command:

```
mysql> explain select tbl1.k1, sum(tbl1.k2) from tbl1 join tbl2 on tbl1.k1 = tbl2.k1 group by tbl1.k1 order by tbl1.k1;
+-----------------------------------------------------------------------------+
| Explain String                                                              |
+-----------------------------------------------------------------------------+
| PLAN FRAGMENT 0                                                             |
|  OUTPUT EXPRS:<slot 5> <slot 3> `tbl1`.`k1` | <slot 6> <slot 4> sum(`tbl1`.`k2`) |
|   PARTITION: UNPARTITIONED                                                  |
|                                                                             |
|   RESULT SINK                                                               |
|                                                                             |
|   9:MERGING-EXCHANGE                                                        |
|      limit: 65535                                                           |
|                                                                             |
| PLAN FRAGMENT 1                                                             |
|  OUTPUT EXPRS:                                                              |
|   PARTITION: HASH_PARTITIONED: <slot 3> `tbl1`.`k1`                         |
|                                                                             |
|   STREAM DATA SINK                                                          |
|     EXCHANGE ID: 09                                                         |
|     UNPARTITIONED                                                           |
|                                                                             |
|   4:TOP-N                                                                   |
|   |  order by: <slot 5> <slot 3> `tbl1`.`k1` ASC                            |
|   |  offset: 0                                                              |
|   |  limit: 65535                                                           |
|   |                                                                         |
|   8:AGGREGATE (merge finalize)                                              |
|   |  output: sum(<slot 4> sum(`tbl1`.`k2`))                                 |
|   |  group by: <slot 3> `tbl1`.`k1`                                         |
|   |  cardinality=-1                                                         |
|   |                                                                         |
|   7:EXCHANGE                                                                |
|                                                                             |
| PLAN FRAGMENT 2                                                             |
|  OUTPUT EXPRS:                                                              |
|   PARTITION: HASH_PARTITIONED: `tbl1`.`k1`                                  |
|                                                                             |
|   STREAM DATA SINK                                                          |
|     EXCHANGE ID: 07                                                         |
|     HASH_PARTITIONED: <slot 3> `tbl1`.`k1`                                  |
|                                                                             |
|   3:AGGREGATE (update serialize)                                            |
|   |  STREAMING                                                              |
|   |  output: sum(`tbl1`.`k2`)                                               |
|   |  group by: `tbl1`.`k1`                                                  |
|   |  cardinality=-1                                                         |
|   |                                                                         |
|   2:HASH JOIN                                                               |
|   |  join op: INNER JOIN (PARTITIONED)                                      |
```

```
|   |  runtime filter: false                                                        |
|   |  hash predicates:                                                             |
|   |  colocate: false, reason: table not in the same group                         |
|   |  equal join conjunct: `tbl1`.`k1` = `tbl2`.`k1`                               |
|   |  cardinality=2                                                                |
|   |                                                                               |
|   |----6:EXCHANGE                                                                 |
|   |                                                                               |
|   5:EXCHANGE                                                                      |
|                                                                                   |
| PLAN FRAGMENT 3                                                                   |
|  OUTPUT EXPRS:                                                                    |
|   PARTITION: RANDOM                                                               |
|                                                                                   |
|   STREAM DATA SINK                                                                |
|     EXCHANGE ID: 06                                                               |
|     HASH_PARTITIONED: `tbl2`.`k1`                                                 |
|                                                                                   |
|   1:OlapScanNode                                                                  |
|      TABLE: tbl2                                                                  |
|      PREAGGREGATION: ON                                                           |
|      partitions=1/1                                                               |
|      rollup: tbl2                                                                 |
|      tabletRatio=3/3                                                              |
|      tabletList=105104776,105104780,105104784                                     |
|      cardinality=1                                                                |
|      avgRowSize=4.0                                                               |
|      numNodes=6                                                                   |
|                                                                                   |
| PLAN FRAGMENT 4                                                                   |
|  OUTPUT EXPRS:                                                                    |
|   PARTITION: RANDOM                                                               |
|                                                                                   |
|   STREAM DATA SINK                                                                |
|     EXCHANGE ID: 05                                                               |
|     HASH_PARTITIONED: `tbl1`.`k1`                                                 |
|                                                                                   |
|   0:OlapScanNode                                                                  |
|      TABLE: tbl1                                                                  |
|      PREAGGREGATION: ON                                                           |
|      partitions=1/1                                                               |
|      rollup: tbl1                                                                 |
|      tabletRatio=3/3                                                              |
|      tabletList=105104752,105104763,105104767                                     |
|      cardinality=2                                                                |
|      avgRowSize=8.0                                                               |
|      numNodes=6                                                                   |
+-----------------------------------------------------------------------------------+
```

The third command `explain verbose select ...;` gives you more details than the second command.

```
mysql> explain verbose select tbl1.k1, sum(tbl1.k2) from tbl1 join tbl2 on tbl1.k1 = tbl2.k1 group by tbl1.k1 order by tbl1.k1;
+-------------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
| Explain String
|
+-------------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
| PLAN FRAGMENT 0
|
|   OUTPUT EXPRS:<slot 5> <slot 3> `tbl1`.`k1` | <slot 6> <slot 4> sum(`tbl1`.`k2`)
|
|   PARTITION: UNPARTITIONED
```

```
|
|
|
|   VRESULT SINK
|
|
|
|   6:VMERGING-EXCHANGE
|
|      limit: 65535
|
|      tuple ids: 3
|
|
| PLAN FRAGMENT 1
|
|
|
|   PARTITION: HASH_PARTITIONED: `default_cluster:test`.`tbl1`.`k2`
|
|
|
|   STREAM DATA SINK
|
|     EXCHANGE ID: 06
|
|     UNPARTITIONED
|
|
|
|   4:VTOP-N
|
|   |  order by: <slot 5> <slot 3> `tbl1`.`k1` ASC
|
|   |  offset: 0
|
|   |  limit: 65535
|
|   |  tuple ids: 3
|
|   |
|
|   3:VAGGREGATE (update finalize)
|
|   |  output: sum(<slot 8>)
|
|   |  group by: <slot 7>
|
|   |  cardinality=-1
|
|   |  tuple ids: 2
|
|   |
|
|   2:VHASH JOIN
|
|   |  join op: INNER JOIN(BROADCAST)[Tables are not in the same group]
|
|   |  equal join conjunct: CAST(`tbl1`.`k1` AS DATETIME) = `tbl2`.`k1`
|
|   |  runtime filters: RF000[in_or_bloom] <- `tbl2`.`k1`
|
|   |  cardinality=0
|
```

```
|   |  vec output tuple id: 4  |  tuple ids: 0 1
|
|   |
|
|   |----5:VEXCHANGE
|
|   |       tuple ids: 1
|
|   |
|
|   0:VOlapScanNode
|
|      TABLE: tbl1(null), PREAGGREGATION: OFF. Reason: the type of agg on StorageEngine's Key column should only be MAX or MIN.agg
expr: sum(`tbl1`.`k2`) |
|      runtime filters: RF000[in_or_bloom] -> CAST(`tbl1`.`k1` AS DATETIME)
|
|      partitions=0/1, tablets=0/0, tabletList=
|
|      cardinality=0, avgRowSize=20.0, numNodes=1
|
|      tuple ids: 0
|
|
|
| PLAN FRAGMENT 2
|
|
|
|   PARTITION: HASH_PARTITIONED: `default_cluster:test`.`tbl2`.`k2`
|
|
|
|   STREAM DATA SINK
|
|     EXCHANGE ID: 05
|
|     UNPARTITIONED
|
|
|
|   1:VOlapScanNode
|
|      TABLE: tbl2(null), PREAGGREGATION: OFF. Reason: null
|
|      partitions=0/1, tablets=0/0, tabletList=
|
|      cardinality=0, avgRowSize=16.0, numNodes=1
|
|      tuple ids: 1
|
|
|
| Tuples:
|
| TupleDescriptor{id=0, tbl=tbl1, byteSize=32, materialized=true}
|
|   SlotDescriptor{id=0, col=k1, type=DATE}
|
|     parent=0
|
|     materialized=true
|
|     byteSize=16
|
|     byteOffset=16
```

```
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=1
|
|
|
|   SlotDescriptor{id=2, col=k2, type=INT}
|
|     parent=0
|
|     materialized=true
|
|     byteSize=4
|
|     byteOffset=0
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=0
|
|
|
|
|
| TupleDescriptor{id=1, tbl=tbl2, byteSize=16, materialized=true}
|
|   SlotDescriptor{id=1, col=k1, type=DATETIME}
|
|     parent=1
|
|     materialized=true
|
|     byteSize=16
|
|     byteOffset=0
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=0
|
|
|
|
|
| TupleDescriptor{id=2, tbl=null, byteSize=32, materialized=true}
|
|   SlotDescriptor{id=3, col=null, type=DATE}
|
|     parent=2
|
|     materialized=true
|
|     byteSize=16
|
|     byteOffset=16
|
|     nullIndicatorByte=0
|
```

```
|     nullIndicatorBit=-1
|
|     slotIdx=1
|
|
|
|   SlotDescriptor{id=4, col=null, type=BIGINT}
|
|     parent=2
|
|     materialized=true
|
|     byteSize=8
|
|     byteOffset=0
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=0
|
|
|
|
|
| TupleDescriptor{id=3, tbl=null, byteSize=32, materialized=true}
|
|   SlotDescriptor{id=5, col=null, type=DATE}
|
|     parent=3
|
|     materialized=true
|
|     byteSize=16
|
|     byteOffset=16
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=1
|
|
|
|   SlotDescriptor{id=6, col=null, type=BIGINT}
|
|     parent=3
|
|     materialized=true
|
|     byteSize=8
|
|     byteOffset=0
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=0
|
|
|
|
```

```
|
| TupleDescriptor{id=4, tbl=null, byteSize=48, materialized=true}
|
|   SlotDescriptor{id=7, col=k1, type=DATE}
|
|     parent=4
|
|     materialized=true
|
|     byteSize=16
|
|     byteOffset=16
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=1
|
|
|
|   SlotDescriptor{id=8, col=k2, type=INT}
|
|     parent=4
|
|     materialized=true
|
|     byteSize=4
|
|     byteOffset=0
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=0
|
|
|
|   SlotDescriptor{id=9, col=k1, type=DATETIME}
|
|     parent=4
|
|     materialized=true
|
|     byteSize=16
|
|     byteOffset=32
|
|     nullIndicatorByte=0
|
|     nullIndicatorBit=-1
|
|     slotIdx=2
|
+----------------------------------------------------------------------------------------------------------------------------------
----------------------+
160 rows in set (0.00 sec)
```

> The information displayed in the query plan is still being standardized and improved, and we will introduce it in detail in subsequent articles.

View Query Profile

The user can open the session variable `is_report_success` with the following command:

```
SET is_report_success=true;
```

Then execute the query, and Doris will generate a Profile of the query. Profile contains the specific execution of a query for each node, which helps us analyze query bottlenecks.

After executing the query, we can first get the Profile list with the following command:

```
mysql> show query profile "/"\G
*************************** 1. row ******************** ******
   QueryId: c257c52f93e149ee-ace8ac14e8c9fef9
      User: root
 DefaultDb: default_cluster:db1
       SQL: select tbl1.k1, sum(tbl1.k2) from tbl1 join tbl2 on tbl1.k1 = tbl2.k1 group by tbl1.k1 order by tbl1.k1
 QueryType: Query
 StartTime: 2021-04-08 11:30:50
   EndTime: 2021-04-08 11:30:50
 TotalTime: 9ms
QueryState: EOF
```

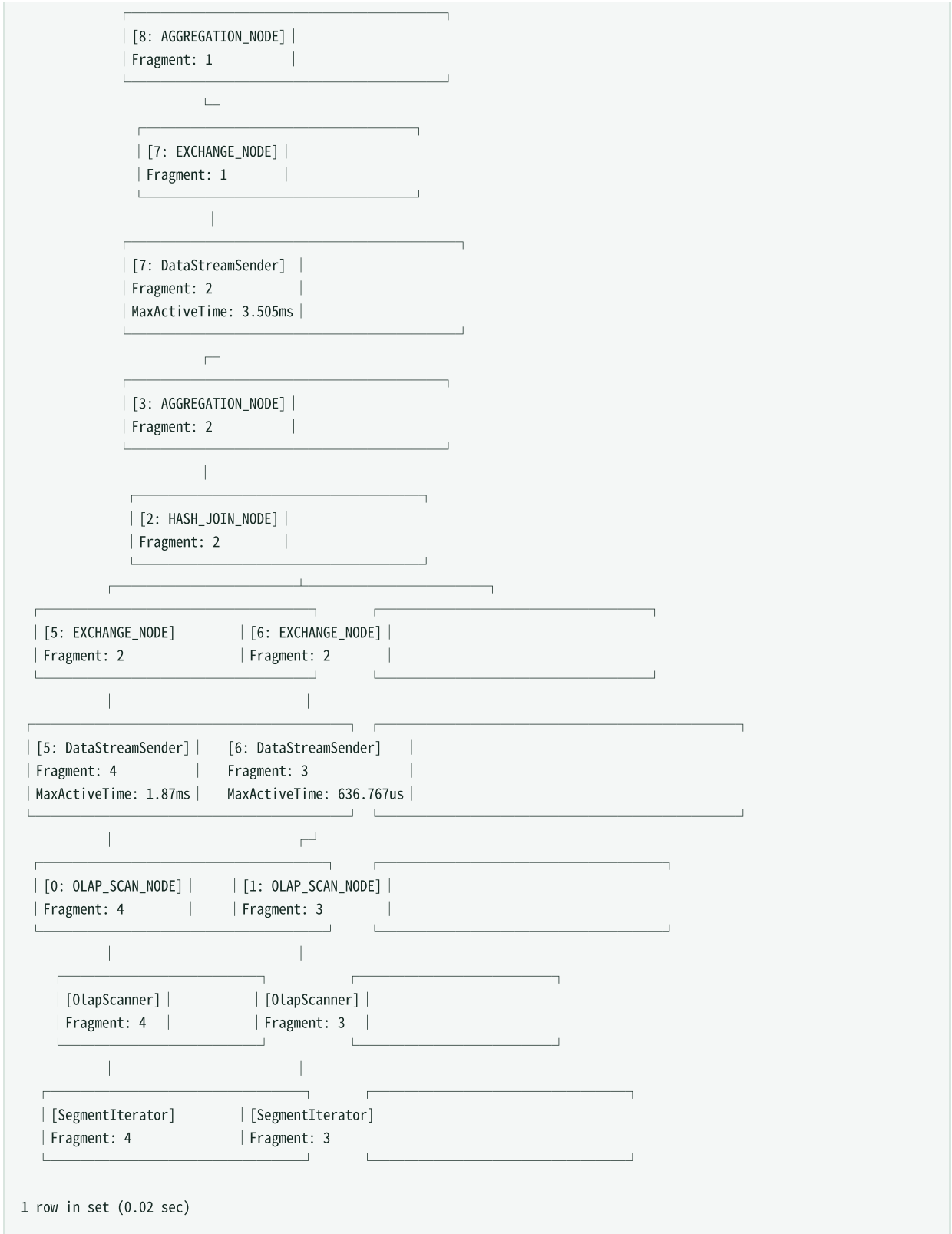
This command will list all currently saved profiles. Each row corresponds to a query. We can select the QueryId corresponding to the Profile we want to see to see the specific situation.

Viewing a Profile is divided into 3 steps:

1. View the overall execution plan tree

   This step is mainly used to analyze the execution plan as a whole and view the execution time of each Fragment.

   ```
   mysql> show query profile "/c257c52f93e149ee-ace8ac14e8c9fef9"\G
   *************************** 1. row ***************************
   Fragments:
                ┌──────────────────────────────────────────────────┐
                │[-1: DataBufferSender]                            │
                │Fragment: 0                                       │
                │MaxActiveTime: 6.626ms                            │
                └──────────────────────────────────────────────────┘
                           │
                  ┌─────────────────────────────────────┐
                  │[9: EXCHANGE_NODE]                   │
                  │Fragment: 0                          │
                  └─────────────────────────────────────┘
                           │
                ┌──────────────────────────────────────────────────┐
                │[9: DataStreamSender]                             │
                │Fragment: 1                                       │
                │MaxActiveTime: 5.449ms                            │
                └──────────────────────────────────────────────────┘
                           │
                     ┌──────────────────────────┐
                     │[4: SORT_NODE]            │
                     │Fragment: 1               │
                     └──────────────────────────┘
                           ┌┘
   ```



1 row in set (0.02 sec)

As shown in the figure above, each node is marked with the Fragment to which it belongs, and at the Sender node of each Fragment, the execution time of the Fragment is marked. This time-consuming is the longest of all Instance execution times under Fragment. This helps us find the most time-consuming Fragment from an overall perspective.

2. View the Instance list under the specific Fragment

For example, if we find that Fragment 1 takes the longest time, we can continue to view the Instance list of Fragment 1:

```
mysql> show query profile "/c257c52f93e149ee-ace8ac14e8c9fef9/1";
+----------------------------------+------------------+------------+
| Instances                        | Host             | ActiveTime |
+----------------------------------+------------------+------------+
| c257c52f93e149ee-ace8ac14e8c9ff03 | 10.200.00.01:9060 | 5.449ms    |
| c257c52f93e149ee-ace8ac14e8c9ff05 | 10.200.00.02:9060 | 5.367ms    |
| c257c52f93e149ee-ace8ac14e8c9ff04 | 10.200.00.03:9060 | 5.358ms    |
+----------------------------------+------------------+------------+
```

This shows the execution nodes and time consumption of all three Instances on Fragment 1.

3. View the specific Instance

   We can continue to view the detailed profile of each operator on a specific Instance:

```
mysql> show query profile "/c257c52f93e149ee-ace8ac14e8c9fef9/1/c257c52f93e149ee-ace8ac14e8c9ff03"\G
*************************** 1. row ******************* ******
Instance:
  ┌-----------------------------------------------------------------------------------------┐
  |[9: DataStreamSender]  |
  |(Active: 37.222us, non-child: 0.40)  |
  |  - Counters:  |
  |  - BytesSent: 0.00  |
  |  - IgnoreRows: 0  |
  |  - OverallThroughput: 0.0 /sec  |
  |  - PeakMemoryUsage: 8.00 KB  |
  |  - SerializeBatchTime: 0ns  |
  |  - UncompressedRowBatchSize: 0.00  |
  └-----------------------------------------------------------------------------------------┘
                     └-┐
                       |
    ┌-------------------------------------------------------------------------┐
    |[4: SORT_NODE]  |
    |(Active: 5.421ms, non-child: 0.71)|
    |  - Counters:  |
    |  - PeakMemoryUsage: 12.00 KB  |
    |  - RowsReturned: 0  |
    |  - RowsReturnedRate: 0  |
    └-------------------------------------------------------------------------┘
                     ┌-┘
                     |
   ┌-------------------------------------------------------------------------┐
   |[8: AGGREGATION_NODE]  |
   |(Active: 5.355ms, non-child: 10.68)|
   |  - Counters:  |
   |  - BuildTime: 3.701us  |
   |  - GetResultsTime: 0ns  |
   |  - HTResize: 0  |
   |  - HTResizeTime: 1.211us  |
   |  - HashBuckets: 0  |
   |  - HashCollisions: 0  |
   |  - HashFailedProbe: 0  |
   |  - HashFilledBuckets: 0  |
   |  - HashProbe: 0  |
   |  - HashTravelLength: 0  |
   |  - LargestPartitionPercent: 0  |
   |  - MaxPartitionLevel: 0  |
   |  - NumRepartitions: 0  |
```

```
    | - PartitionsCreated: 16 |
    | - PeakMemoryUsage: 34.02 MB |
    | - RowsProcessed: 0 |
    | - RowsRepartitioned: 0 |
    | - RowsReturned: 0 |
    | - RowsReturnedRate: 0 |
    | - SpilledPartitions: 0 |
    └─────────────────────────────────────────────────────────────────────────┘
                      └┐
                       |
┌──────────────────────────────────────────────────────────────────────────────────────────────────────┐
|[7: EXCHANGE_NODE] |
|(Active: 4.360ms, non-child: 46.84) |
| - Counters: |
| - BytesReceived: 0.00 |
| - ConvertRowBatchTime: 387ns |
| - DataArrivalWaitTime: 4.357ms |
| - DeserializeRowBatchTimer: 0ns |
| - FirstBatchArrivalWaitTime: 4.356ms|
| - PeakMemoryUsage: 0.00 |
| - RowsReturned: 0 |
| - RowsReturnedRate: 0 |
| - SendersBlockedTotalTimer(*): 0ns |
└──────────────────────────────────────────────────────────────────────────────────────────────────────┘
```

The above figure shows the specific profiles of each operator of Instance c257c52f93e149ee-ace8ac14e8c9ff03 in Fragment 1.

Through the above three steps, we can gradually check the performance bottleneck of a SQL.

Import Analysis

Import Analysis

Doris provides a graphical command to help users analyze a specific import more easily. This article describes how to use this feature.

> This function is currently only for Broker Load analysis.

Import Plan Tree

If you don't know much about Doris' query plan tree, please read the previous article DORIS/best practices/query analysis.

The execution process of a Broker Load request is also based on Doris' query framework. A Broker Load job will be split into multiple subtasks based on the number of DATA INFILE clauses in the import request. Each subtask can be regarded as an independent import execution plan. An import plan consists of only one Fragment, which is composed as follows:

```
┌─────────────┐
│OlapTableSink│
└─────────────┘
       │
┌──────────────┐
│BrokerScanNode│
└──────────────┘
```

BrokerScanNode is mainly responsible for reading the source data and sending it to OlapTableSink, and OlapTableSink is responsible for sending data to the corresponding node according to the partition and bucketing rules, and the corresponding node is responsible for the actual data writing.

The Fragment of the import execution plan will be divided into one or more Instances according to the number of imported source files, the number of BE nodes and other parameters. Each Instance is responsible for part of the data import.

The execution plans of multiple subtasks are executed concurrently, and multiple instances of an execution plan are also executed in parallel.

View import Profile

The user can open the session variable `is_report_success` with the following command:

```
SET is_report_success=true;
```

Then submit a Broker Load import request and wait until the import execution completes. Doris will generate a Profile for this import. Profile contains the execution details of importing each subtask and Instance, which helps us analyze import bottlenecks.

> Viewing profiles of unsuccessful import jobs is currently not supported.

We can get the Profile list first with the following command:

```
mysql> show load profile "/"\G
*************************** 1. row ***************************
                 JobId: 20010
               QueryId: 980014623046410a-af5d36f23381017f
                  User: root
             DefaultDb: default_cluster:test
                   SQL: LOAD LABEL xxx
             QueryType: Load
             StartTime: 2023-03-07 19:48:24
               EndTime: 2023-03-07 19:50:45
             TotalTime: 2m21s
            QueryState: N/A
               TraceId:
          AnalysisTime: NULL
              PlanTime: NULL
          ScheduleTime: NULL
       FetchResultTime: NULL
       WriteResultTime: NULL
WaitAndFetchResultTime: NULL
*************************** 2. row ***************************
                 JobId: N/A
               QueryId: 7cc2d0282a7a4391-8dd75030185134d8
                  User: root
             DefaultDb: default_cluster:test
                   SQL: insert into xxx
             QueryType: Load
             StartTime: 2023-03-07 19:49:15
               EndTime: 2023-03-07 19:49:15
             TotalTime: 102ms
            QueryState: OK
               TraceId:
          AnalysisTime: 825.277us
              PlanTime: 4.126ms
          ScheduleTime: N/A
       FetchResultTime: 0ns
       WriteResultTime: 0ns
WaitAndFetchResultTime: N/A
```

This command will list all currently saved import profiles. Each line corresponds to one import. where the QueryId column is the ID of the import job. This ID can also be viewed through the SHOW LOAD statement. We can select the QueryId corresponding to the Profile we want to see to see the specific situation.

Viewing a Profile is divided into 3 steps:

1. View the subtask overview

   View an overview of subtasks with imported jobs by running the following command:

```
mysql> show load profile "/980014623046410a-af5d36f23381017f";
+----------------------------------+------------+
| TaskId                           | ActiveTime |
+----------------------------------+------------+
| 980014623046410a-af5d36f23381017f | 3m14s      |
+----------------------------------+------------+
```

As shown in the figure above, it means that the import job `980014623046410a-af5d36f23381017f` has a total of one subtask, in which ActiveTime indicates the execution time of the longest instance in this subtask.

2. View the Instance overview of the specified subtask

   When we find that a subtask takes a long time, we can further check the execution time of each instance of the subtask:

```
mysql> show load profile "/980014623046410a-af5d36f23381017f/980014623046410a-af5d36f23381017f";
+----------------------------------+------------------+------------+
| Instances                        | Host             | ActiveTime |
+----------------------------------+------------------+------------+
| 980014623046410a-88e260f0c43031f2 | 10.81.85.89:9067 | 3m7s       |
| 980014623046410a-88e260f0c43031f3 | 10.81.85.89:9067 | 3m6s       |
| 980014623046410a-88e260f0c43031f4 | 10.81.85.89:9067 | 3m10s      |
| 980014623046410a-88e260f0c43031f5 | 10.81.85.89:9067 | 3m14s      |
+----------------------------------+------------------+------------+
```

This shows the time-consuming of four instances of the subtask 980014623046410a-af5d36f23381017f, and also shows the execution node where the instance is located.

3. View the specific Instance

   We can continue to view the detailed profile of each operator on a specific Instance:

```
mysql> show load profile "/980014623046410a-af5d36f23381017f/980014623046410a-af5d36f23381017f/980014623046410a-
88e260f0c43031f5"\G
*************************** 1. row ******************** ******
Instance:
       ┌----------------------------------------┐
       |[-1: OlapTableSink] |
       |(Active: 2m17s, non-child: 70.91) |
       | - Counters: |
       | - CloseWaitTime: 1m53s |
       | - ConvertBatchTime: 0ns |
       | - MaxAddBatchExecTime: 1m46s |
       | - NonBlockingSendTime: 3m11s |
       | - NumberBatchAdded: 782 |
       | - NumberNodeChannels: 1 |
       | - OpenTime: 743.822us |
       | - RowsFiltered: 0 |
       | - RowsRead: 1.599729M (1599729) |
       | - RowsReturned: 1.599729M (1599729)|
       | - SendDataTime: 11s761ms |
       | - TotalAddBatchExecTime: 1m46s |
       | - ValidateDataTime: 9s802ms |
       └----------------------------------------┘
                           |
 ┌------------------------------------------------ ----┐
 |[0: BROKER_SCAN_NODE] |
 |(Active: 56s537ms, non-child: 29.06) |
 | - Counters: |
```

```
| - BytesDecompressed: 0.00 |
| - BytesRead: 5.77 GB |
| - DecompressTime: 0ns |
| - FileReadTime: 34s263ms |
| - MaterializeTupleTime(*): 45s54ms |
| - NumDiskAccess: 0 |
| - PeakMemoryUsage: 33.03 MB |
| - RowsRead: 1.599729M (1599729) |
| - RowsReturned: 1.599729M (1599729) |
| - RowsReturnedRate: 28.295K /sec |
| - TotalRawReadTime(*): 1m20s |
| - TotalReadThroughput: 30.39858627319336 MB/sec|
| - WaitScannerTime: 56s528ms |
└------------------------------------------------ ----┘
```

The figure above shows the specific profiles of each operator of Instance 980014623046410a-af5d36f23381017f in subtask 980014623046410a-88e260f0c43031f5.

Through the above three steps, we can gradually check the execution bottleneck of an import task.

Java UDF

> **Java UDF is supported starting from Doris version 1.2.**

Introduction to Java UDF

Java UDF provides users with a Java interface for writing UDFs, making it convenient for users to execute custom functions using the Java language.

Doris supports writing UDFs, UDAFs, and UDTFs using JAVA. Unless otherwise specified, UDF is used as a general term for all user-defined functions in the following text.

Creating UDF

The implemented jar package can be placed locally or stored on a remote server for download via HTTP, but each FE and BE node must be able to access the jar package.

Otherwise, an error message `Couldn't open file ......` will be returned.

For more syntax help, refer to CREATE FUNCTION.

UDF

```
CREATE FUNCTION java_udf_add_one(int) RETURNS int PROPERTIES (
    "file"="file:///path/to/java-udf-demo-jar-with-dependencies.jar",
    "symbol"="org.apache.doris.udf.AddOne",
    "always_nullable"="true",
    "type"="JAVA_UDF"
);
```

UDAF

```
CREATE AGGREGATE FUNCTION middle_quantiles(DOUBLE,INT) RETURNS DOUBLE PROPERTIES (
    "file"="file:///pathTo/java-udaf.jar",
    "symbol"="org.apache.doris.udf.demo.MiddleNumberUDAF",
    "always_nullable"="true",
    "type"="JAVA_UDF"
);
```

UDTF

> **UDTF is supported starting from Doris version 3.0.**

Using UDF

To utilize UDFs, users must possess the `SELECT` privilege for the corresponding database.

The usage of UDFs is identical to standard functions, with the primary distinction being that built-in functions have a global scope, while UDFs are scoped within the DB.

When the session is linked within the database, directly using the UDF name will search for the corresponding UDF within the current DB. Otherwise, users must explicitly specify the UDF's database name, for example, `dbName.funcName`.

Dropping UDF

If a UDF is no longer needed, it can be dropped using the following command, as detailed in DROP FUNCTION.

Type Correspondence

Type	UDF Argument Type
Bool	Boolean
TinyInt	Byte
SmallInt	Short
Int	Integer
BigInt	Long
LargeInt	BigInteger
Float	Float
Double	Double
Date	LocalDate
Datetime	LocalDateTime
String	String
Decimal	BigDecimal
`array<Type>`	`ArrayList<Type>`
`map<Type1,Type2>`	`HashMap<Type1,Type2>`
`struct<Type...>`	`ArrayList<Object>` (from version 3.0.0)

> `array/map/struct` **types can be nested with other types. For instance, Doris:** `array<array<int>>` **corresponds to JAVA UDF Argument Type:** `ArrayList<ArrayList<Integer>>`**. Other types follow the same pattern.**

> **Warning**
>
> When creating functions, avoid using `varchar` in place of `string`, as this may cause the function to fail.

Writing UDF

This section mainly introduces how to develop a Java UDF. Examples are provided in `samples/doris-demo/java-udf-demo/` for reference. Click here to view them.

When writing a UDF in Java, the main entry point must be the `evaluate` function. This is consistent with other engines like Hive. In this example, we write an `AddOne` UDF to perform an increment operation on integer inputs.

It is worth mentioning that this example not only supports Java UDFs in Doris but is also a UDF supported by Hive. This means that Hive UDFs can be directly migrated to Doris.

Additionally, if the UDF being defined needs to load large resource files or if you want to define global static variables, you can refer to the static variable loading method described at the bottom of the document.

UDF

```
public class AddOne extends UDF {
    public Integer evaluate(Integer value) {
        return value == null ? null : value + 1;
    }
}
```

UDAF

When writing a UDAF using Java code, there are some required functions (marked as required) and an inner class State that must be implemented. Below is a specific example to illustrate.

Example 1

The following SimpleDemo will implement a simple function similar to sum, with the input parameter being INT and the output parameter being INT.

```
package org.apache.doris.udf.demo;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.logging.Logger;

public class SimpleDemo  {

    Logger log = Logger.getLogger("SimpleDemo");

    //Need an inner class to store data
    /*required*/
    public static class State {
        /*some variables if you need */
        public int sum = 0;
```

```
    }

    /*required*/
    public State create() {
        /* here could do some init work if needed */
        return new State();
    }

    /*required*/
    public void destroy(State state) {
        /* here could do some destroy work if needed */
    }

    /*Not Required*/
    public void reset(State state) {
        /*if you want this udaf function can work with window function.*/
        /*Must impl this, it will be reset to init state after calculate every window frame*/
        state.sum = 0;
    }

    /*required*/
    //first argument is State, then other types your input
    public void add(State state, Integer val) throws Exception {
        /* here doing update work when input data*/
        if (val != null) {
            state.sum += val;
        }
    }

    /*required*/
    public void serialize(State state, DataOutputStream out)  {
        /* serialize some data into buffer */
        try {
            out.writeInt(state.sum);
        } catch (Exception e) {
            /* Do not throw exceptions */
            log.info(e.getMessage());
        }
    }

    /*required*/
    public void deserialize(State state, DataInputStream in)  {
        /* deserialize get data from buffer before you put */
        int val = 0;
        try {
            val = in.readInt();
        } catch (Exception e) {
            /* Do not throw exceptions */
            log.info(e.getMessage());
        }
        state.sum = val;
    }

    /*required*/
    public void merge(State state, State rhs) throws Exception {
        /* merge data from state */
        state.sum += rhs.sum;
    }

    /*required*/
    //return Type you defined
    public Integer getValue(State state) throws Exception {
        /* return finally result */
        return state.sum;
```

```
    }
}
```

Example 2

```
package org.apache.doris.udf.demo;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.math.BigDecimal;
import java.util.Arrays;
import java.util.logging.Logger;

/* UDAF to calculate the median */
public class MedianUDAF {
    Logger log = Logger.getLogger("MedianUDAF");

    // State storage
    public static class State {
        // Precision of the return result
        int scale = 0;
        // Whether it is the first time to execute the add method for a certain aggregation condition under a certain tablet
        boolean isFirst = true;
        // Data storage
        public StringBuilder stringBuilder;
    }

    // Initialize the state
    public State create() {
        State state = new State();
        // Pre-initialize based on the amount of data that needs to be aggregated under each aggregation condition of each tablet to
increase performance
        state.stringBuilder = new StringBuilder(1000);
        return state;
    }

    // Process each data under respective aggregation conditions for each tablet
    public void add(State state, Double val, int scale) {
        try {
            if (val != null && state.isFirst) {
                state.stringBuilder.append(scale).append(",").append(val).append(",");
                state.isFirst = false;
            } else if (val != null) {
                state.stringBuilder.append(val).append(",");
            }
        } catch (Exception e) {
            // If it cannot be guaranteed that there will be no exceptions, it is recommended to maximize exception capture in each
method, as handling of exceptions thrown by Java is currently not supported
            log.info("Data acquisition exception: " + e.getMessage());
        }
    }

    // Data needs to be output for aggregation after processing
    public void serialize(State state, DataOutputStream out) {
        try {
            // Currently, only DataOutputStream is provided. If serialization of objects is required, methods such as concatenating
strings, converting to JSON, or serializing into byte arrays can be considered
            // If the State object needs to be serialized, it may be necessary to implement a serialization interface for the State
inner class
            // Ultimately, everything needs to be transmitted via DataOutputStream
            out.writeUTF(state.stringBuilder.toString());
        } catch (Exception e) {
            log.info("Serialization exception: " + e.getMessage());
```

```
        }
    }

    // Obtain the output data from the data processing execution unit
    public void deserialize(State state, DataInputStream in) {
        try {
            String string = in.readUTF();
            state.scale = Integer.parseInt(String.valueOf(string.charAt(0)));
            StringBuilder stringBuilder = new StringBuilder(string.substring(2));
            state.stringBuilder = stringBuilder;
        } catch (Exception e) {
            log.info("Deserialization exception: " + e.getMessage());
        }
    }

    // The aggregation execution unit merges the processing results of data under certain aggregation conditions for a given key.
The state1 parameter is the initialized instance during the first merge of each key
    public void merge(State state1, State state2) {
        try {
            state1.scale = state2.scale;
            state1.stringBuilder.append(state2.stringBuilder.toString());
        } catch (Exception e) {
            log.info("Merge result exception: " + e.getMessage());
        }
    }

    // Output the final result after merging the data for each key
    public Double getValue(State state) {
        try {
            String[] strings = state.stringBuilder.toString().split(",");
            double[] doubles = new double[strings.length];
            for (int i = 0; i < strings.length - 1; i++) {
                doubles[i] = Double.parseDouble(strings[i + 1]);
            }

            Arrays.sort(doubles);
            double n = doubles.length;
            if (n == 0) {
                return 0.0;
            }
            double index = (n - 1) / 2.0;

            int low = (int) Math.floor(index);
            int high = (int) Math.ceil(index);

            double value = low == high ? (doubles[low] + doubles[high]) / 2 : doubles[high];

            BigDecimal decimal = new BigDecimal(value);
            return decimal.setScale(state.scale, BigDecimal.ROUND_HALF_UP).doubleValue();
        } catch (Exception e) {
            log.info("Calculation exception: " + e.getMessage());
        }
        return 0.0;
    }

    // Executed after each execution unit completes
    public void destroy(State state) {
    }
}
```

Best Practices

Loading static variables

Currently, in Doris, executing a UDF function, e.g., `select udf(col) from table`, will load the udf.jar package for each concurrent instance, and unload the udf.jar package when the instance ends. If the udf.jar file needs to load a file of several hundred MBs, the memory usage will increase sharply due to concurrency, potentially leading to OOM (Out of Memory).

The solution is to split the resource loading code, generate a separate jar package, and have other packages directly reference this resource jar package.

Assume the files have been split into DictLibrary and FunctionUdf.

1. Compile the DictLibrary file separately to generate an independent jar package, resulting in a resource file DictLibrary.jar:

```
javac ./DictLibrary.java
jar -cf ./DictLibrary.jar ./DictLibrary.class
```

2. Then compile the FunctionUdf file, directly referencing the resource package from the previous step, resulting in the FunctionUdf.jar package:

```
javac -cp ./DictLibrary.jar ./FunctionUdf.java
jar -cvf ./FunctionUdf.jar ./FunctionUdf.class
```

3. After the above two steps, you will get two jar packages. To allow the resource jar package to be referenced by all concurrent instances, place it in the BE deployment path `be/lib/java_extensions/java-udf`. After restarting BE, it will be loaded with the JVM startup.

4. Finally, use the `create function` statement to create a UDF function, with the file path pointing to the FunctionUdf.jar package. This way, the resource package will be loaded and released with the BE startup and shutdown. The FunctionUdf.jar will be loaded and released with the SQL execution cycle.

```
public class DictLibrary {
    private static HashMap<String, String> res = new HashMap<>();

    static {
        // suppose we built this dictionary from a certain local file.
        res.put("key1", "value1");
        res.put("key2", "value2");
        res.put("key3", "value3");
        res.put("0", "value4");
        res.put("1", "value5");
        res.put("2", "value6");
    }

    public static String evaluate(String key) {
        if (key == null) {
            return null;
        }
        return res.get(key);
```

```
    }
}
```

```
public class FunctionUdf {
    public String evaluate(String key) {
        String value = DictLibrary.evaluate(key);
        return value;
    }
}
```

Usage Notes

1. Complex data types (HLL, Bitmap) are not supported.
2. Users are currently allowed to specify the maximum JVM heap size. The configuration item is the `-Xmx` part of `JAVA_OPTS` in `be.conf`. The default is 1024m. If you need to aggregate data, it is recommended to increase this value to enhance performance and reduce the risk of memory overflow.
3. For Char type UDFs, use the String type when creating the function.
4. Due to issues with JVM loading classes with the same name, do not use multiple classes with the same name as UDF implementations simultaneously. If you want to update a UDF with a class of the same name, you need to restart BE to reload the classpath.

Remote User Defined Function Service

Remote UDF

Remote UDF Service supports accessing user-provided UDF Services via RPC to execute user-defined functions. Compared to native UDF implementation, Remote UDF Service has the following advantages and limitations:

1. Advantages

- Cross-language: UDF Services can be written in various languages supported by Protobuf.
- Security: UDF failures or crashes only affect the UDF Service itself and do not cause Doris process crashes.
- Flexibility: UDF Services can invoke any other services or library classes to meet diverse business requirements.

2. Usage Limitations

- Performance: Compared to native UDFs, UDF Service introduces additional network overhead, resulting in lower performance. Additionally, the UDF Service implementation itself can impact function execution efficiency, and users need to handle issues like high concurrency and thread safety.
- Single-row mode and batch processing mode: In Doris' original row-based query execution framework, UDF RPC calls are made for each row of data, resulting in poor performance. However, in the new vectorized execution framework, UDF RPC calls are made for each batch of data (default: 2048 rows), leading to significant performance improvements. In actual testing, the performance of Remote UDF based on vectorization and batch processing is comparable to that of native UDF based on row storage.

Writing UDF Functions

This section provides instructions on how to develop a Remote RPC service. A Java version example is provided in `samples/doris-demo/udf-demo/` for reference.

Copying the Proto Files

Copy `gensrc/proto/function_service.proto` and `gensrc/proto/types.proto` to the RPC service.

function_service.proto

- PFunctionCallRequest
    - function_name: Function name, corresponding to the symbol specified during function creation.
    - args: Arguments passed to the method.
    - context: Query context information.

- PFunctionCallResponse
    - result: Result.
    - status: Status, where 0 represents normal.
- PCheckFunctionRequest
    - function: Function-related information.
    - match_type: Matching type.
- PCheckFunctionResponse
    - status: Status, where 0 represents normal.

Generating Interfaces

Generate code using protoc. Refer to `protoc -h` for specific parameters.

Implementing Interfaces

The following three methods need to be implemented:

- fnCall: Used to write the calculation logic.
- checkFn: Used for UDF creation validation, checking if the function name, parameters, return values, etc., are valid.
- handShake: Used for interface probing.

Creating UDF

Currently, UDTF is not supported.

```
CREATE FUNCTION
name ([,...])
[RETURNS] rettype
PROPERTIES (["key"="value"][,...])
```

Note:

1. The `symbol` in the PROPERTIES represents the method name passed in the RPC call, and this parameter must be set.
2. The `object_file` in the PROPERTIES represents the RPC service address. Currently, it supports a single address and cluster addresses in the brpc-compatible format.
3. The `type` in the PROPERTIES represents the UDF invocation type, which is set to Native by default. Use RPC to pass when using RPC UDF.

4. `name`: A function belongs to a specific database. The name is in the form of `dbName.funcName`. When `dbName` is not explicitly specified, the current session's database is used as `dbName`.

Example:

```
CREATE FUNCTION rpc_add_two(INT,INT) RETURNS INT PROPERTIES (
  "SYMBOL"="add_int_two",
  "OBJECT_FILE"="127.0.0.1:9114",
  "TYPE"="RPC"
);
CREATE FUNCTION rpc_add_one(INT) RETURNS INT PROPERTIES (
  "SYMBOL"="add_int_one",
  "OBJECT_FILE"="127.0.0.1:9114",
  "TYPE"="RPC"
);
CREATE FUNCTION rpc_add_string(varchar(30)) RETURNS varchar(30) PROPERTIES (
  "SYMBOL"="add_string",
  "OBJECT_FILE"="127.0.0.1:9114",
  "TYPE"="RPC"
);
```

Using UDF

Users must have the `SELECT` privilege on the corresponding database to use UDF.

The usage of UDF is similar to regular functions, with the only difference being that the scope of built-in functions is global, while the scope of UDF is within the database. When the session is connected to a database, simply use the UDF name to search for the corresponding UDF within the current database. Otherwise, the user needs to explicitly specify the database name of the UDF, such as `dbName.funcName`.

Deleting UDF

When you no longer need a UDF function, you can delete it using the `DROP FUNCTION` command.

Example

The `samples/doris-demo/` directory provides examples of RPC server implementations in CPP, Java, and Python languages. Please refer to the `README.md` file in each directory for specific usage instructions. For example, `rpc_add_string`:

```
mysql >select rpc_add_string('doris');
+-------------------------+
| rpc_add_string('doris') |
+-------------------------+
| doris_rpc_test          |
+-------------------------+
```

The log will display:

```
INFO: fnCall request=function_name: "add_string"
args {
  type {
    id: STRING
  }
```

```
  has_null: false
  string_value: "doris"
}
INFO: fnCall res=result {
  type {
    id: STRING
  }
  has_null: false
  string_value: "doris_rpc_test"
}
status {
  status_code: 0
}
```

Lakehouse Overview

Before the integration of data lake and data warehouse, the history of data analysis went through three eras: database, data warehouse, and data lake analytics.

- Database, the fundamental concept, was primarily responsible for online transaction processing and providing basic data analysis capabilities.
- As data volumes grew, data warehouses emerged. They store valuable data that has been cleansed, processed, and modeled, providing analytics capabilities for business.
- The advent of data lakes was to serve the needs of enterprises for storing, managing, and reprocessing raw data. They required low-cost storage for structured, semi-structured, and even unstructured data, and they also needed an integrated solution encompassing data processing, data management, and data governance.

Data warehouses addresses the need for fast data analysis, while data lakes are good at data storage and management. The integration of them, known as “lakehouse”, is to facilitate the seamless integration and free flow of data between the data lake and data warehouse. It enables users to leverage the analytic capabilities of the data warehouse while harnessing the data management power of the data lake.

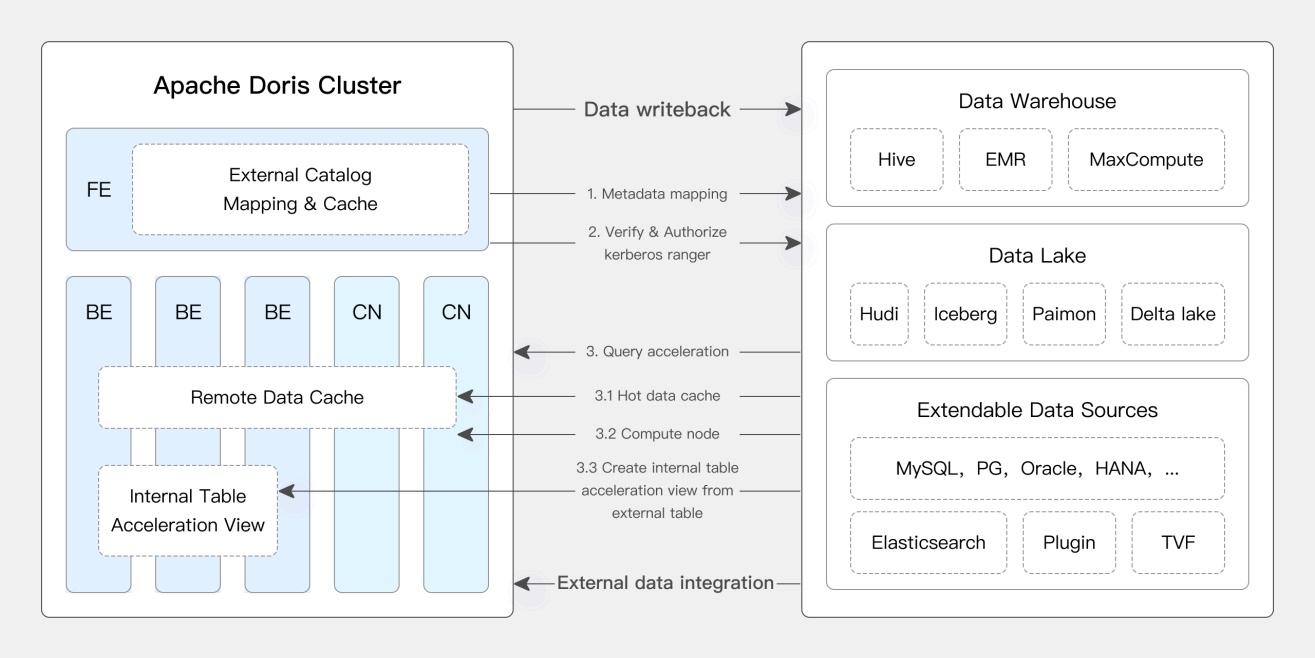
Applicable scenarios

We design the Doris lakehouse solution for the following four applicable scenarios:

- Lakehouse query acceleration: As a highly efficient OLAP query engine, Doris has excellent MPP-based vectorized distributed query capabilities. Data lake analysis with Doris will benefit from the efficient query engine.
- Unified data analysis gateway: Doris provides data query and writing capabilities for various and heterogeneous data sources. Users can unify these external data sources onto Doris’ data mapping structure, allowing for a consistent query experience when accessing external data sources through Doris.
- Unified data integration: Doris, through its data lake integration capabilities, enables incremental or full data synchronization from multiple data sources to Doris. Doris processes the synchronized data and makes it available for querying. The processed data can also be exported to downstream systems as full or incremental data services. Using Doris, you can have less reliance on external tools and enable end-to-end connectivity from ata synchronization to data processing.
- More open data platform: Many data warehouses have their own storage formats. They require external data to be imported into themselve before the data queryable. This creates a closed ecosystem where data in the data warehouse can only be accessed by the data warehouse itself. In this case, users might concern if data will be locked into a specific data warehouse or wonder if there are any other easy way for data export. The Doris lakehouse solution provides open data formats, such as Parquet/ORC, to allow data access by various external systems. Additionally, just like Iceberg and Hudi providing open metadata management capabilities, metadata, whether stored in Doris, Hive Metastore, or other unified metadata centers, can be accessed through publicly available APIs. An open data ecosystem makes it easy for enterprises to migrate to new data management systems and reduces the costs and risks in this process.

Doris-based data lakehouse architecture

Apache Doris can work as a data lakehouse with its Multi Catalog feature. It can access databases and data lakes including Apache Hive, Apache Iceberg, Apache Hudi, Apache Paimon (incubating), Elasticsearch, MySQL, Oracle, and SQLServer. It also supports Apache Ranger for privilege management.



doris-based-data-lakehouse-architecture

Data access steps:

1. Create metadata mapping: Apache Doris fetches metadata via Catalog and caches it for metadata management. For metadata mapping, it supports JDBC username-password authentication, Kerberos/Ranger-based authentication, and KMS-based data encryption.
2. Launch query request: When the user launches a query request from the Doris frontend (FE), Doris generates a query plan based on its cached metadata. Then, it utilizes the Native Reader to fetch data from external storage (HDFS, S3) for data computation and analysis. During query execution, it caches the hot data to prepare for similar queries in the future.
3. Return result: When a query is finished, it returns the query result on the frontend.
4. Write result to data lake: For users who need to write the result back to the data lake instead of returning it on the frontend, Doris supports result writeback in CSV, Parquet, and ORC formats via the export method to the data lake.

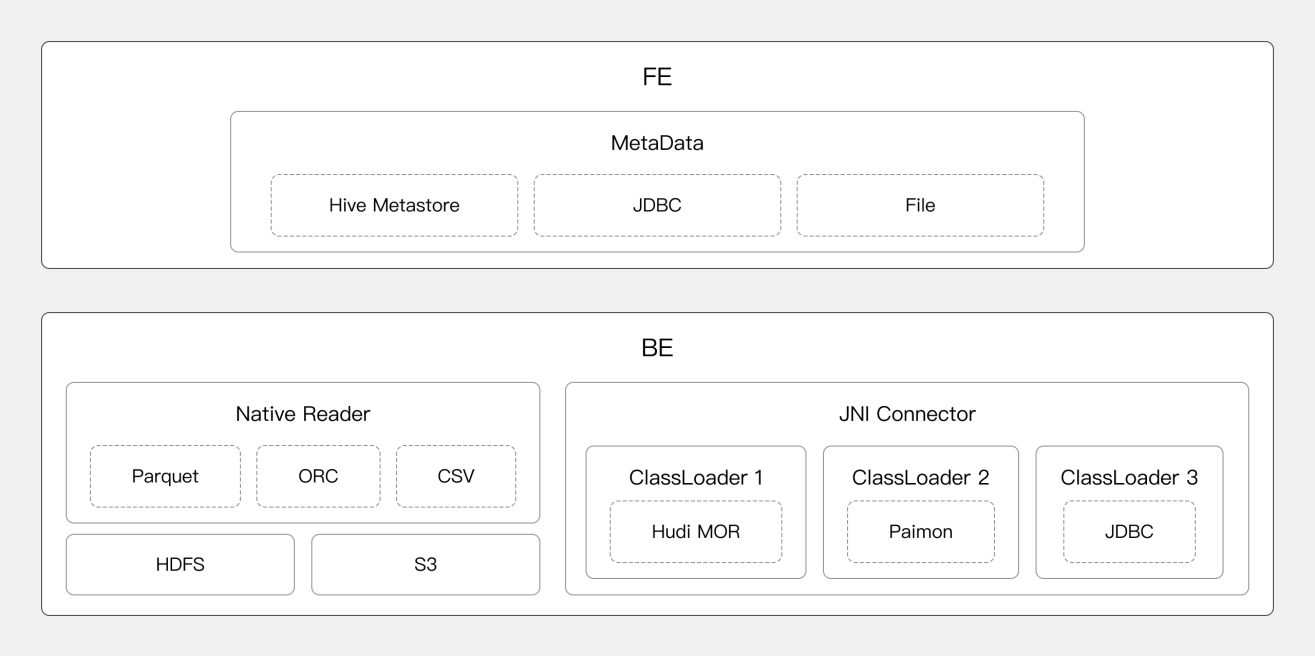
Core technologies

Apache Doris empowers data lake analytics with its extensible connection framework, metadata caching, data caching, NativeReader, I/O optimization, and statistics collection capabilities.

Extensible connection framework

The data connection framework in Apache Doris includes metadata connection and data reading.

- Metadata connection: Metadata connection is conducted in the frontend of Doris. The MetaData Manager in the frontend can access and manage metadata from Hive Metastore, JDBC, and data files.
- Data reading: Apache Doris has a NativeReader for efficient data reading from HDFS and object storage. It supports Parquet, ORC, and text data. You can also connect Apache Doris to the Java big data ecosystem via its JNI Connector.



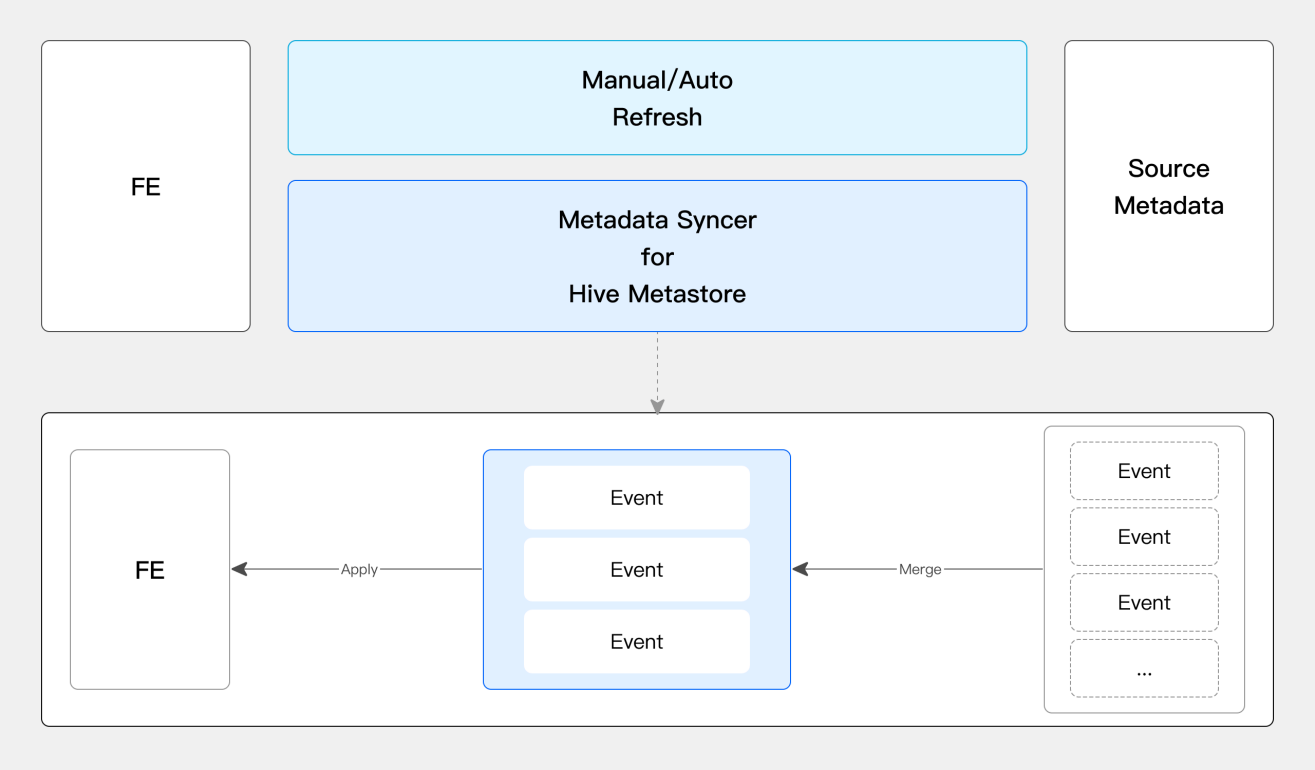
extensible-connection-framework

Efficient caching strategy

Apache Doris caches metadata, data, and query results to improve query performance.

Metadata caching

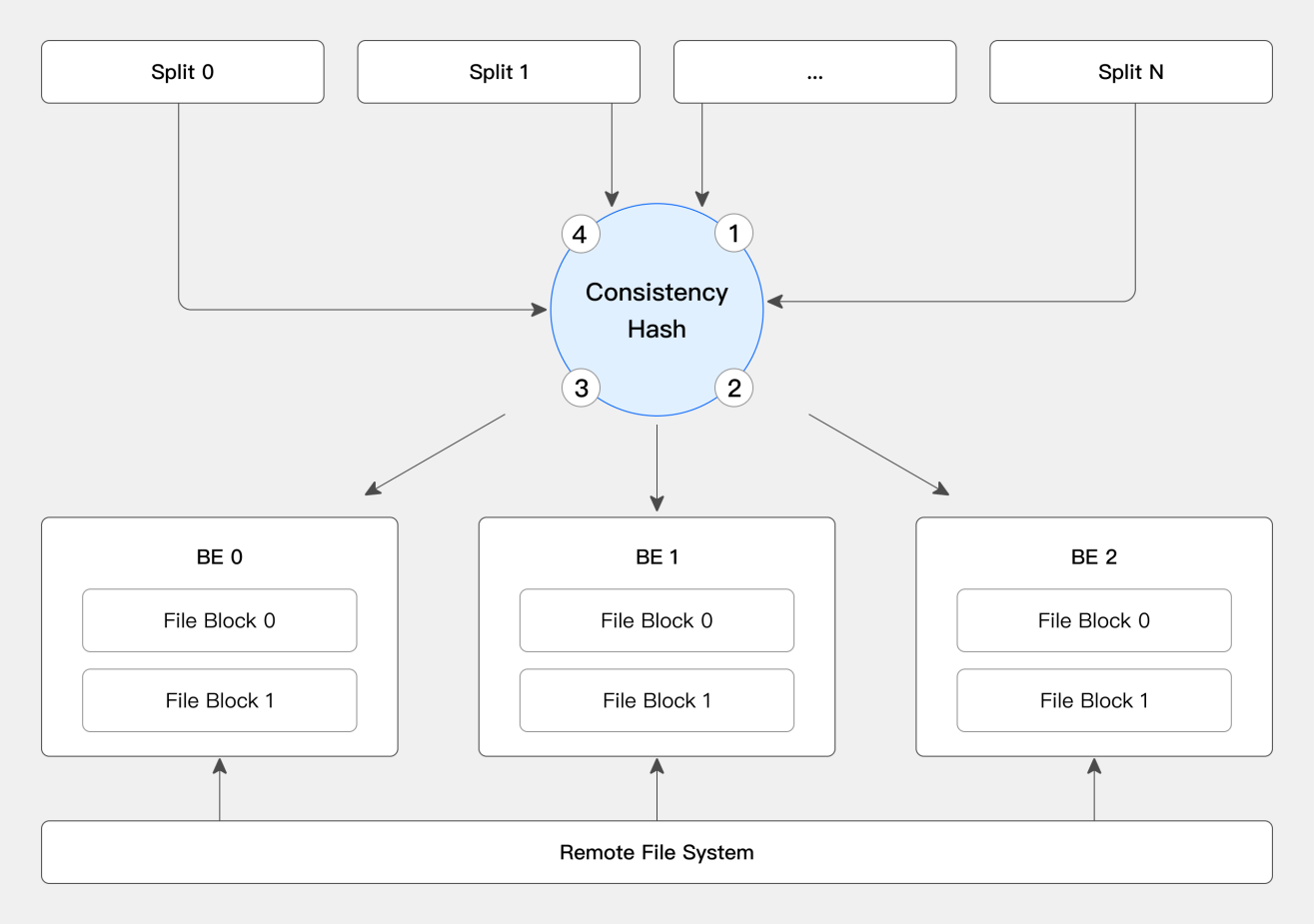
Apache Doris supports metadata synchronization by three methods: auto synchronization, periodic synchronization, and metadata subscription (for Hive Metastore only). It accesses metadata from the data lake and stores it in the memory of its frontend. When a user issues a query request, Doris can quickly fetch metadata from its own memory and generate a query plan accordingly. Apache Doris improves synchronization efficiency by merging the concurrent metadata events. It can process over 100 metadata events per second.



metadata-caching

Data caching

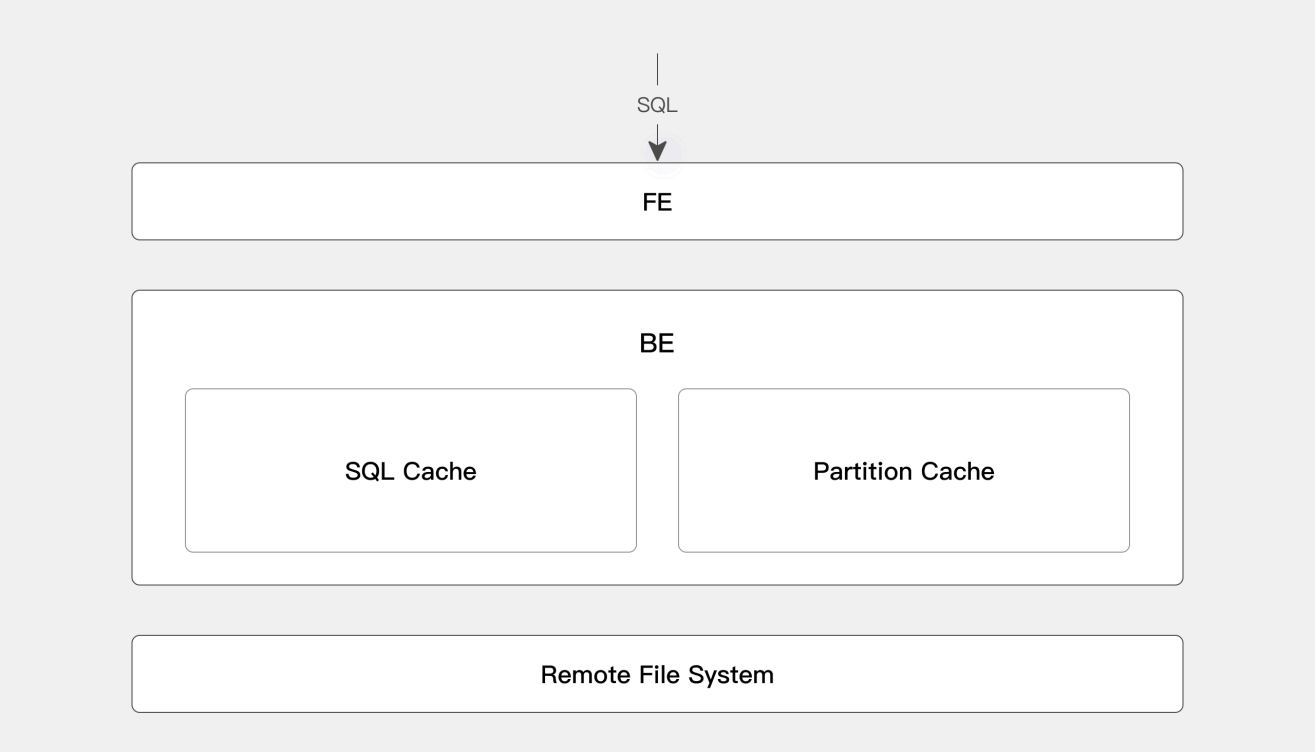
- File caching: Apache Doris caches hot data from the data lake onto its local disks to reduce data transfer via the network and increase data access efficiency.
- Cache distribution: Apache Doris distributes the cached data across all backend nodes via consistent hashing to avoid cache expiration caused by cluster scaling.
- Cache eviction(update): When Apache Doris detects changes in the metadata of a data file, it promptly updates its cached data to ensure data consistency.



data-caching

Query result caching & partition caching

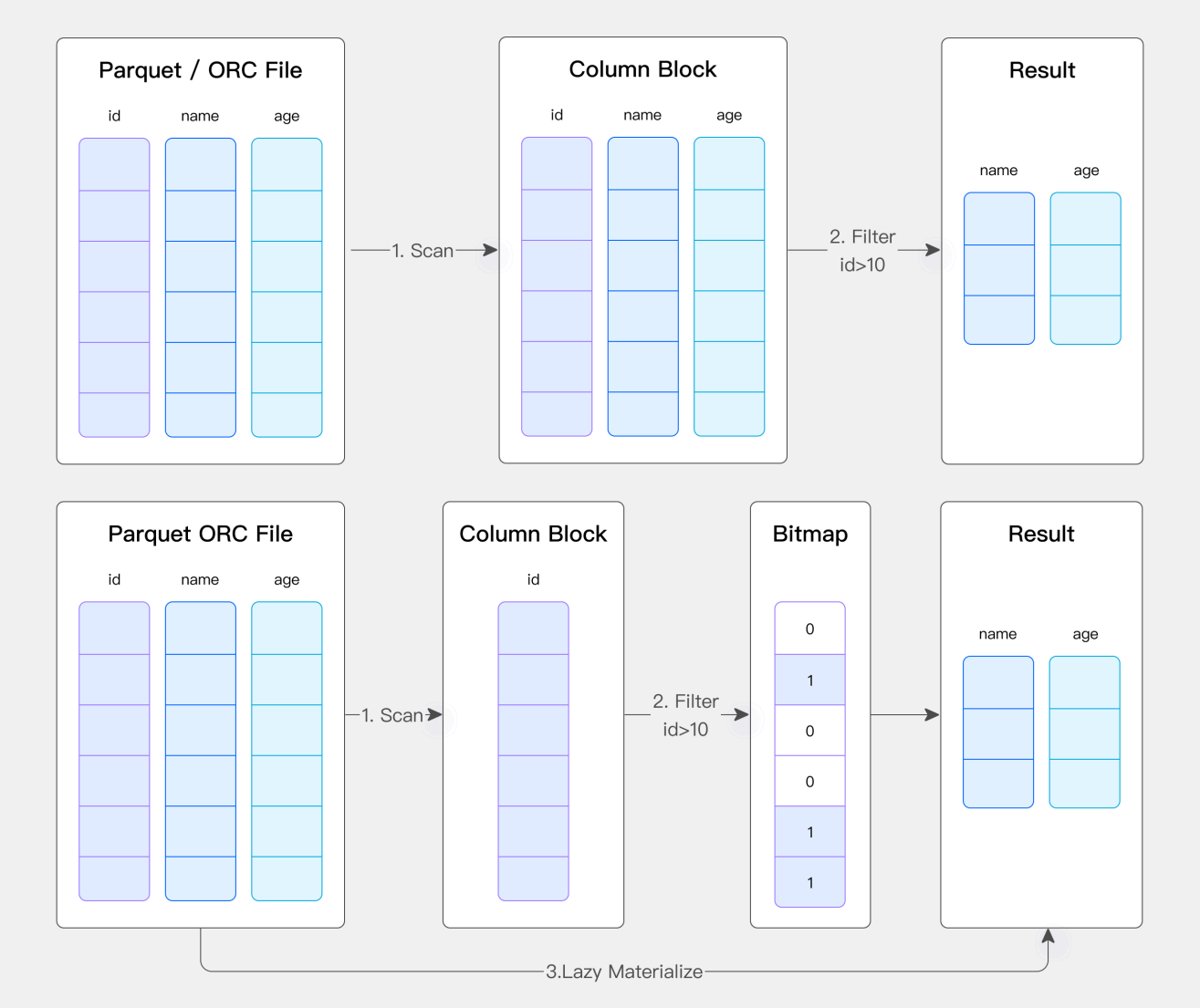
- Query result caching: Apache Doris caches the results of previous SQL queries, so it can reuse them when similar queries are launched. It will read the corresponding result from the cache directly and return it to the client. This increases query efficiency and concurrency.
- Partition caching: Apache Doris allows you to cache part of your data partitions in the backend to increase query efficiency. For example, if you need the data from the past 7 days (counting today), you can cache the data from the previous 6 days and merge it with today's data. This can largely reduce real-time computation burden and increase speed.



query-result-caching-and-partition-caching

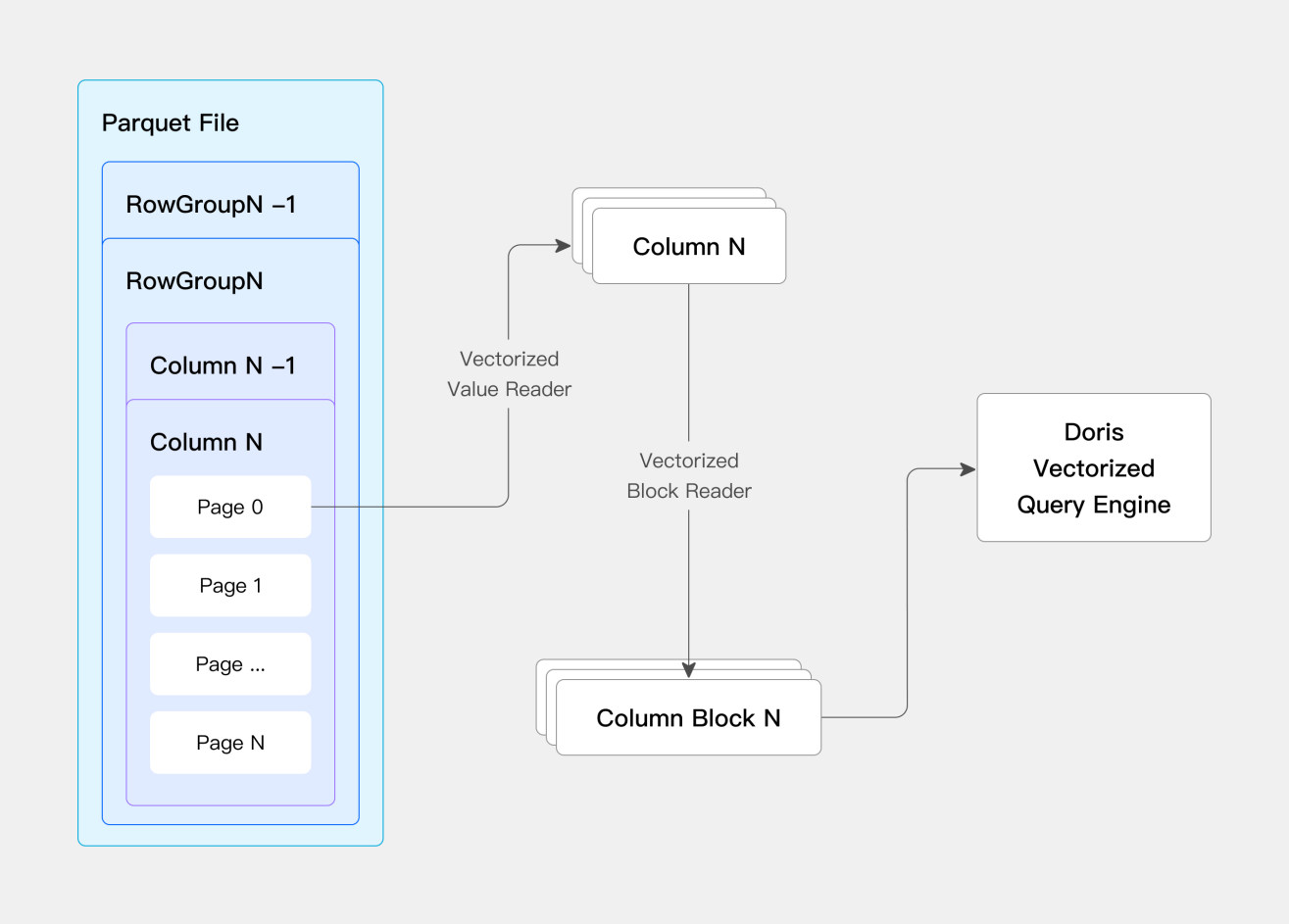
Native Reader

- Self-developed Native Reader to avoid data conversion: Apache Doris has its own columnar storage format, which is different from Parquet and ORC. To avoid overheads caused by data format conversion, we have built our own Native Reader for Parquet and ORC files.
- Lazy materialization: The Native Reader can utilize indexes and filters to improve data reading. For example, when the user needs to do filtering based on the ID column, what the Native Reader does is to read and filter the ID column, take note of the relevant row numbers, and then read the corresponding rows of the other columns based on the row numbers recorded. This reduces data scanning and speeds up data reading.



native-reader

- Vectorized data reading: We have vectorized data file reading in Apache Doris to increase reading speed.



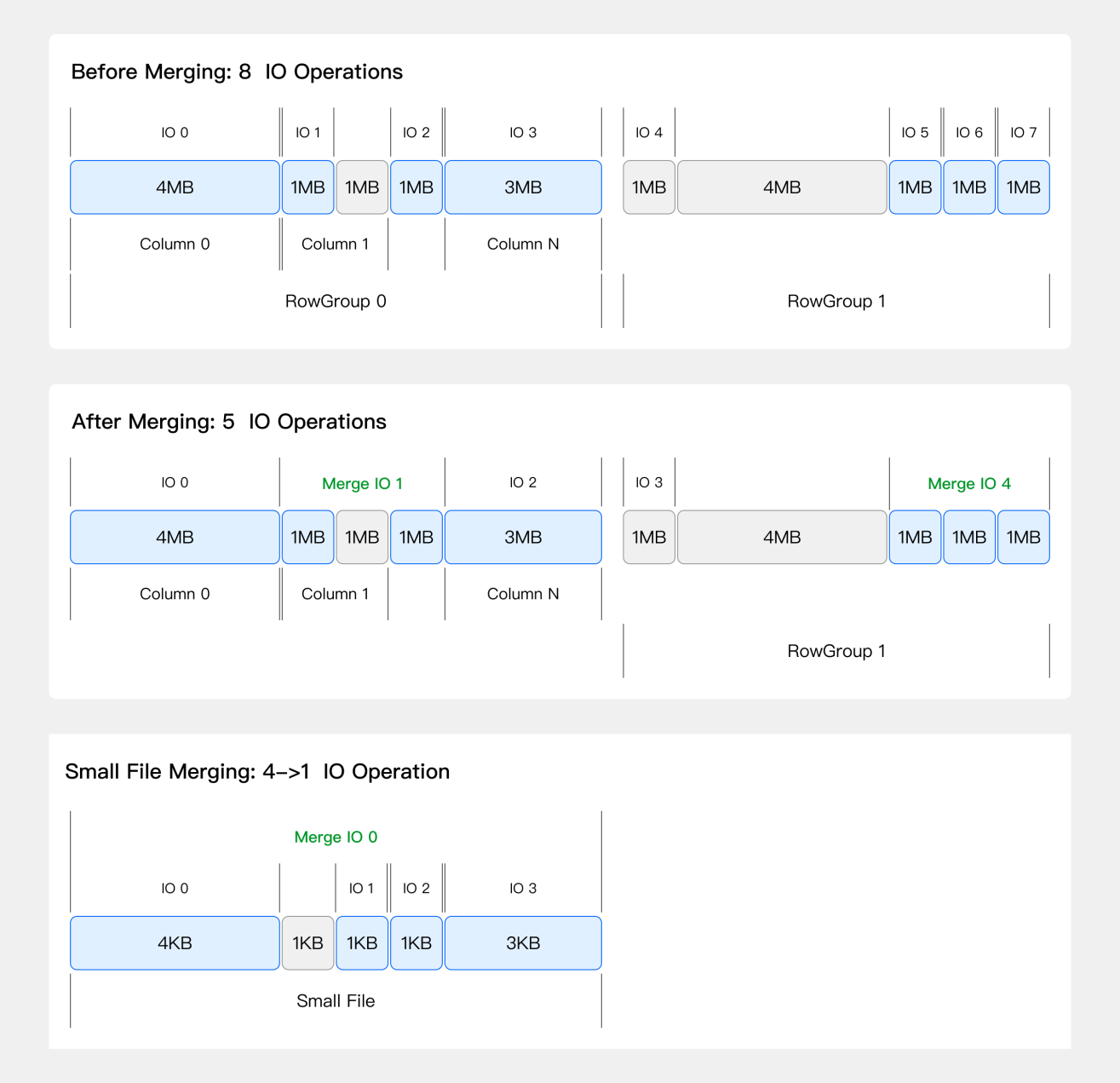
vectorized-data-reading

Merge I/O

A large number of small file network I/O requests can reduce I/O performance. To solve that, we introduce an I/O merging mechanism.

For example, if you decide to merge any I/O requests smaller than 3MB, then instead of dealing with all 8 small I/O requests separately, the system only needs to read 5 timess. That's how it improves data access efficiency.

The downside of I/O merging is that it might read some irrelevant data, because it merges and reads together some intermediate data that might not be needed. Despite that, I/O merging can still bring a significant increase in overall throughput, especially for use cases with a large number of small files (1KB~1MB). Besides, users can mitigate the downside by adjusting the size of I/O merging.



merge-io

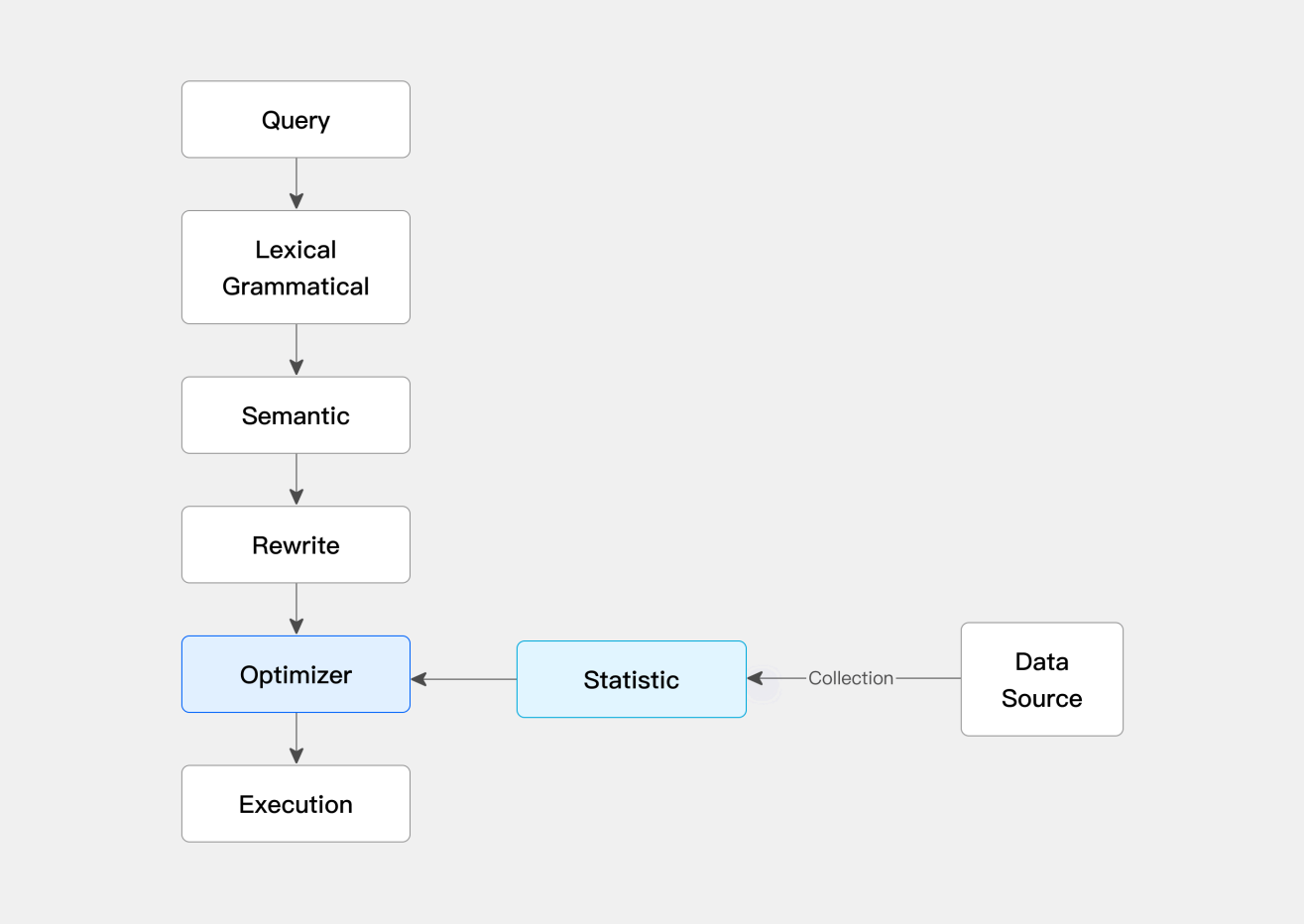
Statistics collection

Apache Doris collects statistical information to help the CBO understand data distribution, so that the CBO can evaluate the feasibility of each predicate and the cost of each execution plan. This helps the CBO figure out the most efficient query plan. It is also applicable in data lake analytics scenarios.

Methods of statistics collection include manual collection and auto collection.

Meanwhile, to avoid pressure on the Doris backend, we have enabled sampling collection of statistics.

In some cases, some data is frequently accessed while others are not, so Apache Doris allows partition-based statistics collection, so it can ensure high efficiency in hot data queries while mitigating the impact of statistics collection on the backend.



statistics-collection

Multi-Catalog

Multi-Catalog is designed to facilitate connection to external data catalogs and enhance the data lake analysis and federated data query capabilities of Doris.

In older versions of Doris, user data is in a two-tiered structure: database and table. Thus, connections to external catalogs could only be done at the database or table level. For example, users could create a mapping to a table in an external catalog via `create external table`, or to a database via `create external database`. If there are large amounts of databases or tables in the external catalog, users will need to create mappings to them one by one, which could be tedious.

With Multi-Catalog, Doris now has a new three-tiered metadata hierarchy (catalog -> database -> table), which means users can connect to external data at the catalog level directly.

Multi-Catalog works as an additional and enhanced external table connection method. It helps users conduct multi-catalog federated queries quickly.

Basic concepts

- Internal Catalog

Existing databases and tables in Doris are all under the Internal Catalog, which is the default catalog in Doris and cannot be modified or deleted.

- External Catalog

Users can create an External Catalog using the CREATE CATALOG command, and view the existing Catalogs via the SHOW CATALOGS command.

- Switch Catalog

After logging in to Doris, you will enter the Internal Catalog by default. Then, you can view or switch to your target database via `SHOW DATABASES` and `USE DB` .

Example:

```
SWITCH internal;
SWITCH hive_catalog;
```

After switching catalog, you can view or switch to your target database in that catalog via `SHOW DATABASES` and `USE DB`. You can view and access data in External Catalogs the same way as doing that in the Internal Catalog.

- Delete Catalog

You cand delete an External Catalog via the DROP CATALOG command. (The Internal Catalog cannot be deleted.) The deletion only removes the mapping in Doris to the corresponding catalog. It doesn't change the external catalog in external data sources by all means.

Examples

Connect to Hive

The following is the instruction on how to connect to a Hive catalog using the Catalog feature.

For more information about connecting to Hive, please see Hive Catalog.

1. Create Catalog

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004'
);
```

Syntax help: CREATE CATALOG

1. View Catalog
2. View existing Catalogs via the `SHOW CATALOGS` command:

```
mysql> SHOW CATALOGS;
+-----------+-------------+----------+-----------+-------------------------+---------------------+-------------------------+
| CatalogId | CatalogName | Type     | IsCurrent | CreateTime              | LastUpdateTime      | Comment                 |
+-----------+-------------+----------+-----------+-------------------------+---------------------+-------------------------+
|     10024 | hive        | hms      | yes       | 2023-12-25 16:11:41.687 | 2023-12-25 20:43:18 | NULL                    |
|         0 | internal    | internal |           | UNRECORDED              | NULL                | Doris internal catalog |
+-----------+-------------+----------+-----------+-------------------------+---------------------+-------------------------+
```

- SHOW CATALOGS

- You can view the CREATE CATALOG statement via SHOW CREATE CATALOG.
- You can modify the Catalog PROPERTIES via ALTER CATALOG.

1. Switch Catalog

Switch to the Hive Catalog using the `SWITCH` command, and view the databases in it:

```
mysql> SWITCH hive;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW DATABASES;
+-----------+
| Database  |
+-----------+
| default   |
| random    |
| ssb100    |
| tpch1     |
| tpch100   |
| tpch1_orc |
+-----------+
```

Syntax help: SWITCH

1. Use the Catalog

After switching to the Hive Catalog, you can use the relevant features.

For example, you can switch to Database tpch100, and view the tables in it:

```
mysql> USE tpch100;
Database changed

mysql> SHOW TABLES;
+-------------------+
| Tables_in_tpch100 |
+-------------------+
| customer          |
| lineitem          |
| nation            |
| orders            |
| part              |
| partsupp          |
| region            |
| supplier          |
+-------------------+
```

You can view the schema of Table lineitem:

```
mysql> DESC lineitem;
+-----------------+---------------+------+------+---------+-------+
| Field           | Type          | Null | Key  | Default | Extra |
+-----------------+---------------+------+------+---------+-------+
| l_shipdate      | DATE          | Yes  | true | NULL    |       |
| l_orderkey      | BIGINT        | Yes  | true | NULL    |       |
| l_linenumber    | INT           | Yes  | true | NULL    |       |
| l_partkey       | INT           | Yes  | true | NULL    |       |
| l_suppkey       | INT           | Yes  | true | NULL    |       |
| l_quantity      | DECIMAL(15,2) | Yes  | true | NULL    |       |
```

```
| l_extendedprice | DECIMAL(15,2) | Yes  | true | NULL    |       |
| l_discount      | DECIMAL(15,2) | Yes  | true | NULL    |       |
| l_tax           | DECIMAL(15,2) | Yes  | true | NULL    |       |
| l_returnflag    | TEXT          | Yes  | true | NULL    |       |
| l_linestatus    | TEXT          | Yes  | true | NULL    |       |
| l_commitdate    | DATE          | Yes  | true | NULL    |       |
| l_receiptdate   | DATE          | Yes  | true | NULL    |       |
| l_shipinstruct  | TEXT          | Yes  | true | NULL    |       |
| l_shipmode      | TEXT          | Yes  | true | NULL    |       |
| l_comment       | TEXT          | Yes  | true | NULL    |       |
+-----------------+---------------+------+------+---------+-------+
```

You can perform a query:

```
mysql> SELECT l_shipdate, l_orderkey, l_partkey FROM lineitem limit 10;
+------------+------------+-----------+
| l_shipdate | l_orderkey | l_partkey |
+------------+------------+-----------+
| 1998-01-21 |   66374304 |    270146 |
| 1997-11-17 |   66374304 |    340557 |
| 1997-06-17 |   66374400 |   6839498 |
| 1997-08-21 |   66374400 |  11436870 |
| 1997-08-07 |   66374400 |  19473325 |
| 1997-06-16 |   66374400 |   8157699 |
| 1998-09-21 |   66374496 |  19892278 |
| 1998-08-07 |   66374496 |   9509408 |
| 1998-10-27 |   66374496 |   4608731 |
| 1998-07-14 |   66374592 |  13555929 |
+------------+------------+-----------+
```

Or you can conduct a join query:

```
mysql> SELECT l.l_shipdate FROM hive.tpch100.lineitem l WHERE l.l_partkey IN (SELECT p_partkey FROM internal.db1.part) LIMIT 10;
+------------+
| l_shipdate |
+------------+
| 1993-02-16 |
| 1995-06-26 |
| 1995-08-19 |
| 1992-07-23 |
| 1998-05-23 |
| 1997-07-12 |
| 1994-03-06 |
| 1996-02-07 |
| 1997-06-01 |
| 1996-08-23 |
+------------+
```

- The table is identified in the format of `catalog.database.table`. For example, `internal.db1.part` in the above snippet.
- If the target table is in the current Database of the current Catalog, `catalog` and `database` in the format can be omitted.
- You can use the `INSERT INTO` command to insert table data from the Hive Catalog into a table in the Internal Catalog. This is how you can import data from External Catalogs to the Internal Catalog:

```
mysql> SWITCH internal;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> USE db1;
Database changed

mysql> INSERT INTO part SELECT * FROM hive.tpch100.part limit 1000;
Query OK, 1000 rows affected (0.28 sec)
{'label':'insert_212f67420c6444d5_9bfc184bf2e7edb8', 'status':'VISIBLE', 'txnId':'4'}
```

Column type mapping

After you create a Catalog, Doris will automatically synchronize the databases and tables from the corresponding external catalog to it. The following shows how Doris maps different types of catalogs and tables.

As for types that cannot be mapped to a Doris column type, such as `UNION` and `INTERVAL` , Doris will map them as the UNSUPPORTED type. Here are examples of queries in a table containing UNSUPPORTED types:

Suppose the table is of the following schema:

```
k1 INT,
k2 INT,
k3 UNSUPPORTED,
k4 INT
select * from table;                // Error: Unsupported type 'UNSUPPORTED_TYPE' in 'k3
select * except(k3) from table;     // Query OK.
select k1, k3 from table;           // Error: Unsupported type 'UNSUPPORTED_TYPE' in 'k3
select k1, k4 from table;           // Query OK.
```

You can find more details of the mapping of various data sources (Hive, Iceberg, Hudi, Elasticsearch, and JDBC) in the corresponding pages.

Privilege management

When using Doris to access the data in the External Catalog, by default, it relies on Doris's own permission access management function.

Along with the new Multi-Catalog feature, we also added privilege management at the Catalog level (See Privilege Management for details).

Users can also specify a custom authentication class through `access_controller.class`. For example, if you specify it as

`"access_controller.class" = "org.apache.doris.catalog.authorizer.RangerHiveAccessControllerFactory"`, then you can use Apache Ranger to perform authentication management on Hive Catalog. For more information see: Hive

Database synchronization management

Set `include_database_list` and `exclude_database_list` in Catalog properties to specify the databases to synchronize.

`include_database_list`: only synchronize the specified databases, split with `,`. The default setting is to synchronize all databases. The database names are case-sensitive.

`exclude_database_list`: databases that do not need to synchronize, split with `,`. The default setting is to exclude no databases. The database names are case-sensitive.

> If there is overlap between `include_database_list` and `exclude_database_list`, `exclude_database_list`will take precedence over `include_database_list`.
>
> To connect to JDBC data sources, these two properties should work with `only_specified_database`. See JDBC for more details.

Metadata refresh

By default, metadata changes in external catalogs, such as creating and dropping tables, adding and dropping columns, etc., will not be synchronized to Doris.

Users can refresh metadata in the following ways.

Manual refresh

Users need to manually refresh the metadata through the REFRESH command.

Regular refresh

When creating the catalog, specify the refresh time parameter `metadata_refresh_interval_sec` in the properties. It is measured in seconds. If this is set when creating the catalog, the FE master node will refresh the catalog regularly accordingly. Currently, Doris supports regular refresh for three types of catalogs:

- hms: Hive MetaStore
- es: Elasticsearch
- Jdbc: standard interface for database access (JDBC)

```
-- Set the catalog refresh interval to 20 seconds
CREATE CATALOG es PROPERTIES (
    "type"="es",
    "hosts"="http://127.0.0.1:9200",
    "metadata_refresh_interval_sec"="20"
);
```

Auto Refresh

Currently Doris supports auto-refresh for Hive Catalog.

Hive Catalog

By connecting to Hive Metastore, Doris can automatically obtain database and table information from Hive and perform data queries.

In addition to Hive, systems like Iceberg and Hudi also use Hive Metastore for metadata storage. Hive Catalog in Doris allows users to easily integrate with not only Hive but also systems that use Hive Metastore as metadata storage.

Note

- Hive 1, Hive 2, and Hive 3 are supported.
- Managed Tables, External Tables, and part of the Hive Views are supported.
- It can recognize Hive, Iceberg, and Hudi metadata stored in the Hive Metastore.
- You should place core-site.xml, hdfs-site.xml, and hive-site.xml in the conf directory of both FE and BE. During reading, the hadoop configuration files in the conf directory are read first, followed by the configuration files related to the HADOOP_CONF_DIR environment variable.
- If the Hadoop node is configured with hostname, please ensure to add the corresponding mapping relationship to the /etc/hosts file.

Create Catalog

Hive on HDFS

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hadoop.username' = 'hive',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:8088',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:8088',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

In addition to the two required parameters of `type` and `hive.metastore.uris`, you can specify more parameters to pass the required information for the connection.

Example to provide HDFS HA information:

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hadoop.username' = 'hive',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:8088',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:8088',
    'dfs.client.failover.proxy.provider.your-
```

```
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

Hive on VIEWFS

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hadoop.username' = 'hive',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:8088',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:8088',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider',
    'fs.defaultFS' = 'viewfs://your-cluster',
    'fs.viewfs.mounttable.your-cluster.link./ns1' = 'hdfs://your-nameservice/',
    'fs.viewfs.mounttable.your-cluster.homedir' = '/ns1'
);
```

You can add viewfs-related parameters to the catalog configuration as above, or add them to `conf/core-site.xml`.

For details about how viewfs works and the parameter configurations, please refer to relevant hadoop documents, such as https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/ViewFs.html

Hive on JuiceFS

Example for data stored on JuiceFS:

(You should put `juicefs-hadoop-x.x.x.jar` under `fe/lib/` and `apache_hdfs_broker/lib/`.)

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hadoop.username' = 'root',
    'fs.jfs.impl' = 'io.juicefs.JuiceFileSystem',
    'fs.AbstractFileSystem.jfs.impl' = 'io.juicefs.JuiceFS',
    'juicefs.meta' = 'xxx'
);
```

Doris query DLC

> **Supported since Doris version 2.0.13**

DLC of Tencent Cloud use HMS to manage its metadata, so Hive catalog can be used。 DLC stores data on LakeFS or COSN, below catalog is compatible with both file system.

```
CREATE CATALOG dlc PROPERTIES (
    'type' = 'hms',
    'hive.metastore.uris' = 'thrift://<dlc_metastore_ip>:<dlc_metastore_port>',
    's3.access_key' = 'xxxxx',
```

```
    's3.secret_key' = 'xxxxx',
    's3.region' = 'ap-xxx',
    's3.endpoint' = 'cos.ap-xxx.myqcloud.com',
    'fs.cosn.bucket.region' = 'ap-xxx',
    'fs.ofs.user.appid' = '<your_tencent_cloud_appid>',
    'fs.ofs.tmp.cache.dir' = '<tmp_cache_dir>'
);
```

After create catalog, you need to grant essential privilege to the appid your dlc catalog is using.

Hive on S3

```
CREATE CATALOG hive PROPERTIES (
    "type"="hms",
    "hive.metastore.uris" = "thrift://172.0.0.1:9083",
    "s3.endpoint" = "s3.us-east-1.amazonaws.com",
    "s3.region" = "us-east-1",
    "s3.access_key" = "ak",
    "s3.secret_key" = "sk",
    "use_path_style" = "true"
);
```

Optional properties:

- s3.connection.maximum: defaults to 50
- s3.connection.request.timeout: defaults to 3000ms
- s3.connection.timeout: defaults to 1000ms

Hive on OSS

```
CREATE CATALOG hive PROPERTIES (
    "type"="hms",
    "hive.metastore.uris" = "thrift://172.0.0.1:9083",
    "oss.endpoint" = "oss.oss-cn-beijing.aliyuncs.com",
    "oss.access_key" = "ak",
    "oss.secret_key" = "sk"
);
```

Hive on OBS

```
CREATE CATALOG hive PROPERTIES (
    "type"="hms",
    "hive.metastore.uris" = "thrift://172.0.0.1:9083",
    "obs.endpoint" = "obs.cn-north-4.myhuaweicloud.com",
    "obs.access_key" = "ak",
    "obs.secret_key" = "sk"
);
```

Hive on COS

```
CREATE CATALOG hive PROPERTIES (
    "type"="hms",
    "hive.metastore.uris" = "thrift://172.0.0.1:9083",
```

```
    "cos.endpoint" = "cos.ap-beijing.myqcloud.com",
    "cos.access_key" = "ak",
    "cos.secret_key" = "sk"
);
```

Hive with AWS Glue

> **When connecting to Glue, if you are in a non-EC2 environment, you need to copy the `~/.aws` directory from the EC2 environment to your current environment. Alternatively, you can download the AWS CLI tool for configuration, which will also create the `.aws` directory in the current user's home directory.**

Access to AWS services with Doris supports two authentication methods.

Authentication using Catalog properties

You can fill in the basic properties of credentials, such as:

- When accessing S3, you can use `s3.endpoint`, `s3.access_key`, `s3.secret_key`.
- When accessing Glue, you can use `glue.endpoint`, `glue.access_key`, `glue.secret_key`.

For example, if you are accessing Glue using Iceberg Catalog, you can fill in the following properties to access tables hosted on Glue:

```
CREATE CATALOG hive PROPERTIES (
    "type"="hms",
    "hive.metastore.type" = "glue",
    "glue.endpoint" = "https://glue.us-east-1.amazonaws.com",
    "glue.access_key" = "ak",
    "glue.secret_key" = "sk"
);
```

Authentication using system properties

This authentication method is used for applications running on AWS resources such as EC2 instances. It avoids hard-coding credentials and enhances data security.

If you create a Catalog without filling in the Credentials properties, the DefaultAWSCredentialsProviderChain will be used by default. It can read properties configured in system environment variables or instance profiles.

For configurations of environment variables and system properties, refer to AWS CLI.

- Optional environment variables include: `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, `AWS_SESSION_TOKEN`, `AWS_ROLE_ARN`, `AWS_WEB_IDENTITY_TOKEN_FILE`, etc.
- You can follow AWS Configure to directly configure the Credentials information, which will generate a credentials file in the `~/.aws` directory.

Metadata Cache & Refresh

For Hive Catalog, Doris caches 4 types of metadata:

- Table schema: table column information, etc.
- Partition value: partition value information of all partitions in a table.
- Partition information: information of each partition, such as partition data format, partition storage location, partition value, etc.
- File information: file information corresponding to each partition, such as file path location, etc.

The above cached information will not be persisted to Doris, so operations such as restarting Doris FE node, switching to masters may invalidate the cache. If the cache expires, Doris will directly access the Hive MetaStore to obtain information and refill the cache.

Metadata cache can be updated automatically, manually, or configured with TTL (Time-to-Live).

Default behavior and TTL

By default, the metadata cache expires 10 minutes after it is first accessed. The TTL is determined by the configuration parameter `external_cache_expire_time_minutes_after_access` in fe.conf. (Note that in versions 2.0.1 and earlier, the default value for this parameter was 1 day).

For example, if the user accesses the metadata of table A for the first time at 10:00, then the metadata will be cached and will automatically expire after 10:10. If the user accesses the same metadata again at 10:11, Doris will directly access the Hive MetaStore to obtain information and refill the cache.

`external_cache_expire_time_minutes_after_access` affects all 4 caches under Catalog.

For the `INSERT INTO OVERWRITE PARTITION` operation commonly used in Hive, you can also timely update the `File Information Cache` by configuring the TTL of the `File Information Cache`:

```
CREATE CATALOG hive PROPERTIES (
     'type'='hms',
     'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
     'file.meta.cache.ttl-second' = '60'
);
```

In the above example, `file.meta.cache.ttl-second` is set to 60 seconds, and the cache will expire after 60 seconds. This parameter will only affect the `file information cache`.

You can also set this value to 0 to disable file caching, which will fetch file information directly from the Hive MetaStore every time.

Manual refresh

Users need to manually refresh the metadata through the REFRESH command.

REFRESH CATALOG: Refresh the specified Catalog.

```
REFRESH CATALOG ctl1 PROPERTIES("invalid_cache" = "true");
```

- This command refreshes the database list, table list, and all cached information of the specified Catalog.
- `invalid_cache` indicates whether to refresh the cache. It defaults to true. If it is false, it will only refresh the database and table list of the catalog, but not the cached information. This parameter is applicable when

the user only wants to synchronize newly added or deleted database/table information.

REFRESH DATABASE: Refresh the specified Database.

```
REFRESH DATABASE [ctl.]db1 PROPERTIES("invalid_cache" = "true");
```

- This command refreshes the table list of the specified database and all cached information under the database.
- `invalid_cache` indicates the same as above. It defaults to true. If false, it will only refresh the table list of the database, but not the cached information. This parameter is applicable when the user only wants to synchronize newly added or deleted table information.

REFRESH TABLE: Refresh the specified Table.

```
REFRESH TABLE [ctl.][db.]tbl1;
```

- This command refreshes all cached information under the specified table.

Regular refresh

Users can schedule the refresh for a Catalog when creating the Catalog.

```
CREATE CATALOG hive PROPERTIES (
     'type'='hms',
     'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
     'metadata_refresh_interval_sec' = '600'
);
```

In the above example, `metadata_refresh_interval_sec` means refreshing the Catalog every 600 seconds, which means the following command will be executed automatically every 600 seconds:

```
REFRESH CATALOG ctl1 PROPERTIES("invalid_cache" = "true");
```

Note that the scheduled refresh interval must not be less than 5 seconds.

Auto refresh

Currently, Doris only supports automatic update of metadata in Hive Metastore (HMS). It perceives changes in metadata by the FE node which regularly reads the notification events from HMS. The supported events are as follows:

Event	Corresponding Update Operation
CREATE DATABASE	Create a database in the corresponding catalog.
DROP DATABASE	Delete a database in the corresponding catalog.
ALTER DATABASE	Such alterations mainly include changes in properties, comments, or storage location of databases. They do not affect Doris' queries in External Catalogs.
CREATE TABLE	Create a table in the corresponding database.

Event	Corresponding Update Operation
DROP TABLE	Delete a table in the corresponding database, and invalidate the cache of that table.
ALTER TABLE	If it is renaming, delete the table of the old name, and then create a new table with the new name; otherwise, invalidate the cache of that table.
ADD PARTITION	Add a partition to the cached partition list of the corresponding table.
DROP PARTITION	Delete a partition from the cached partition list of the corresponding table, and invalidate the cache of that partition.
ALTER PARTITION	If it is a renaming, delete the partition of the old name, and then create a new partition with the new name; otherwise, invalidate the cache of that partition.

After data ingestion, changes in partition tables will follow the `ALTER PARTITION` logic, while those in non-partition tables will follow the `ALTER TABLE` logic.

If changes are conducted on the file system directly instead of through the HMS, the HMS will not generate an event. As a result, such changes will not be perceived by Doris.

The automatic update feature involves the following parameters in fe.conf:

- `enable_hms_events_incremental_sync`: This specifies whether to enable automatic incremental synchronization for metadata, which is disabled by default.
- `hms_events_polling_interval_ms`: This specifies the interval between two readings, which is set to 10000 by default. (Unit: millisecond)
- `hms_events_batch_size_per_rpc`: This specifies the maximum number of events that are read at a time, which is set to 500 by default.

To enable automatic update(Excluding Huawei MRS), you need to modify the hive-site.xml of HMS and then restart HMS and HiveServer2:

```
<property>
    <name>hive.metastore.event.db.notification.api.auth</name>
    <value>false</value>
</property>
<property>
    <name>hive.metastore.dml.events</name>
    <value>true</value>
</property>
<property>
    <name>hive.metastore.transactional.event.listeners</name>
    <value>org.apache.hive.hcatalog.listener.DbNotificationListener</value>
</property>
```

For Huawei MRS, you need to change hivemetastore-site.xml and restart HMS and HiveServer2:

```
<property>
    <name>metastore.transactional.event.listeners</name>
    <value>org.apache.hive.hcatalog.listener.DbNotificationListener</value>
</property>
```

Hive Version

Doris can correctly access the Hive Metastore in different Hive versions. By default, Doris will access the Hive Metastore with a Hive 2.3 compatible interface. If you encounter the `Invalid method name: 'get_table_req'` error during queries, the root cause is Hive version incompatibility.

You can specify the Hive version when creating the Catalog. For example, to access Hive 1.1.0 version:

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hive.version' = '1.1.0'
);
```

Column type mapping

For Hive/Iceberge/Hudi

HMS Type	Doris Type	Comment
boolean	boolean	
tinyint	tinyint	
smallint	smallint	
int	int	
bigint	bigint	
date	date	
timestamp	datetime	
float	float	
double	double	
char	char	
varchar	varchar	
decimal	decimal	
`array<type>`	`array<type>`	Supports nested structure, such as `array<map<string, int>>`
`map<KeyType, ValueType>`	`map<KeyType, ValueType>`	Supports nested structure, such as `map<string, array<int>>`
`struct<col1: Type1, col2: Type2, ...>`	`struct<col1: Type1, col2: Type2, ...>`	Supports nested structure, such as `struct<col1: array<int>, col2: map<int, date>>`
other	unsupported	

> ****Truncate char or varchar columns****
>
> If `truncate_char_or_varchar_columns` is enabled, and the maximum length of the char or varchar column in the schema of the hive table is inconsistent with the schema in the underlying Parquet or ORC file, the columns will be truncated according to the maximum length of the hive table column.

> The variable defaults to false.

Access HMS with broker

Add the following setting when creating an HMS catalog, file splitting and scanning for Hive external table will be completed by the broker named `test_broker`.

```
"broker.name" = "test_broker"
```

Doris supports querying HMS Catalog Iceberg by implementing the Iceberg FileIO interface in the Broker. If needed, you can add the following configuration when creating an HMS Catalog.

```
"io-impl" = "org.apache.doris.datasource.iceberg.broker.IcebergBrokerIO"
```

Integrate with Apache Ranger

Apache Ranger is a security framework for monitoring, enabling services, and comprehensive data security access management on the Hadoop platform.

Doris supports authentication on specified External Hive Catalog using Apache Ranger.

Currently Doris supports authentication of Ranger libraries, tables, and columns, but does not support encryption, row privileges, and Data Mask.

If you need to use Apache Ranger for authentication of the entire Doris cluster, refer to Integrate with Apache Ranger.

Environment settings

To connect to the Hive Metastore with Ranger authentication enabled, you need to add configurations and environment settings as follows:

1. When creating a Catalog, add:

```
"access_controller.properties.ranger.service.name" = "hive",
"access_controller.class" = "org.apache.doris.catalog.authorizer.RangerHiveAccessControllerFactory",
```

> `access_controller.properties.ranger.service.name` **refers to the type of service, such as** `hive`**,** `hdfs`**, etc. It does not correspond to the value of** `ranger.plugin.hive.service.name` **in the configuration file.**

2. Configure all FE environments:

a. Copy the configuration files ranger-hive-audit.xml, ranger-hive-security.xml, and ranger-policymgr-ssl.xml under the HMS conf directory to the FE conf directory.

b. Modify the properties in ranger-hive-security.xml, an example is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    #The directory for caching permission data, needs to be writable
    <property>
        <name>ranger.plugin.hive.policy.cache.dir</name>
        <value>/mnt/datadisk0/zhangdong/rangerdata</value>
    </property>
    #The time interval for periodically pulling permission data
    <property>
        <name>ranger.plugin.hive.policy.pollIntervalMs</name>
        <value>30000</value>
    </property>

    <property>
        <name>ranger.plugin.hive.policy.rest.client.connection.timeoutMs</name>
        <value>60000</value>
    </property>

    <property>
        <name>ranger.plugin.hive.policy.rest.client.read.timeoutMs</name>
        <value>60000</value>
    </property>

    <property>
        <name>ranger.plugin.hive.policy.rest.ssl.config.file</name>
        <value></value>
    </property>

    <property>
        <name>ranger.plugin.hive.policy.rest.url</name>
        <value>http://172.21.0.32:6080</value>
    </property>

    <property>
        <name>ranger.plugin.hive.policy.source.impl</name>
        <value>org.apache.ranger.admin.client.RangerAdminRESTClient</value>
    </property>

    <property>
        <name>ranger.plugin.hive.service.name</name>
        <value>hive</value>
    </property>

    <property>
        <name>xasecure.hive.update.xapolicies.on.grant.revoke</name>
        <value>true</value>
    </property>

</configuration>
```

c. In order to obtain the logs of Ranger authentication, add the configuration file log4j.properties in the `<doris_home>/conf` directory.

d. Restart FE.

Best Practice

1. Create `user1` on the Ranger side and authorize the query permission of db1.table1.col1
2. Create `role1` on the Ranger side and authorize the query permission of db1.table1.col2

3. Create the same `user1` in Doris, `user1` will be directly granted with the query permission of db1.table1.col1
4. Create the same `role1` in Doris, and assign `role1` to `user1`. `user1` will be directly granted with the query permission of db1.table1.col1 and col2 at the same time.
5. The Admin and Root privileges of Doris are not impacted by Apache Ranger.

Authentication with Kerberos

Kerberos is an authentication protocol designed to provide strong identity verification for applications through the use of encryption with private keys.

Environment settings

1. When the services in the cluster are configured with Kerberos authentication, the relevant authentication information is needed in Hive Catalog configuration.

- `hadoop.kerberos.keytab`: the principal required for authentication. It should be the same as the keytab in the Doris cluster.
- `hadoop.kerberos.principal`: Find the corresponding principal for the hostname in the Doris cluster, such as `doris/``hostname@HADOOP.COM`, and verify with `klist -kt` using the keytab.
- `yarn.resourcemanager.principal`: Go to the Yarn Resource Manager node and retrieve this from `yarn-site.xml`. Verify the keytab of Yarn with `klist -kt`.
- `hive.metastore.kerberos.principal`: Go to the Hive metadata service node and retrieve this from `hive-site.xml`. Verify the keytab of Hive with `klist -kt`.
- `hadoop.security.authentication`: Enable Hadoop Kerberos authentication.

Place the `krb5.conf` file and `keytab` authentication file on all BE and FE nodes. Ensure consistencyt of the path of the `keytab` authentication file. By default, the `krb5.conf` file is placed in the `/etc/krb5.conf` path. Also, confirm that the JVM parameter `-Djava.security.krb5.conf` and the environment variable `KRB5_CONFIG`point to the correct path of the `krb5.conf` file.

2. After the configuration is completed, if you cannot locate the issue in the FE and BE logs, you can enable Kerberos debugging.

- On all FE and BE nodes, locate the `conf/fe.conf` and `conf/be.conf` files in the deployment path.
- After finding the configuration files, set the JVM parameter `-Dsun.security.krb5.debug=true` in the `JAVA_OPTS` variable to enable Kerberos debugging.
- The FE Kerberos authentication debug information can be found in the FE log path `log/fe.out`, and the BE Kerberos authentication debug information can be found in the BE log path `log/be.out`.
- For solutions to more common issues, refer to FAQ.

Best practice

Example:

```
CREATE CATALOG hive_krb PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hive.metastore.sasl.enabled' = 'true',
    'hive.metastore.kerberos.principal' = 'your-hms-principal',
```

```
    'hadoop.security.authentication' = 'kerberos',
    'hadoop.kerberos.keytab' = '/your-keytab-filepath/your.keytab',
    'hadoop.kerberos.principal' = 'your-principal@YOUR.COM',
    'yarn.resourcemanager.principal' = 'your-rm-principal'
);
```

Example of HDFS HA information and Kerberos authentication information:

```
CREATE CATALOG hive_krb_ha PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.0.0.1:9083',
    'hive.metastore.sasl.enabled' = 'true',
    'hive.metastore.kerberos.principal' = 'your-hms-principal',
    'hadoop.security.authentication' = 'kerberos',
    'hadoop.kerberos.keytab' = '/your-keytab-filepath/your.keytab',
    'hadoop.kerberos.principal' = 'your-principal@YOUR.COM',
    'yarn.resourcemanager.principal' = 'your-rm-principal',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:8088',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:8088',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

Hive transactional table

Hive transactional tables are tables that support ACID semantics in Hive. For more details, see: https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions

Support for Hive transactional tables

Table Type	Supported Operations in Hive	Hive Table Properties	Supported Hive Versions
Full-ACID Transactional Table	Supports Insert, Update, Delete operations	‘transactional’=‘true’, ‘transactional_properties’=‘insert_only’	3.x, 2.x. In 2.x, loading can only be performed after major compaction is executed in Hive.
Insert-Only Transactional Table	Only supports Insert operations	‘transactional’=‘true’	3.x，2.x

Current restrictions

Currently, scenarios involving Original Files are not supported. After a table is converted into a transactional table, subsequent newly written data files will follow the schema of Hive Transactional table. However, existing data files will not be converted to the schema of the Transactional table. These files are referred to as Original Files.

Hudi Catalog

Usage

1. The query types supported by the Hudi table are as follows, and the Incremental Query will be supported in the future.

Table Type	Supported Query types
Copy On Write	Snapshot Query + Time Travel
Merge On Read	Snapshot Queries + Read Optimized Queries + Time Travel

2. Doris supports Hive Metastore(Including catalogs compatible with Hive MetaStore, like AWS Glue/Alibaba DLF) Catalogs.

Create Catalog

Same as creating Hive Catalogs. A simple example is provided here. See Hive for more information.

```
CREATE CATALOG hudi PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004',
    'hadoop.username' = 'hive',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:4007',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:4007',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

Optional configuration parameters:

name	description	default
use_hive_sync_partition	Use hms synchronized partition data	false

Column Type Mapping

Same as that in Hive Catalogs. See the relevant section in Hive.

Skip Merge

Spark will create the read optimize table with `_ro` suffix when generating hudi mor table. Doris will skip the log files when reading optimize table. Doris does not determine whether a table is read optimize by the `_ro` suffix instead of the hive inputformat. Users can observe whether the inputformat of the ‘cow/mor/read optimize’ table is the same through the `SHOW CREATE TABLE` command. In addition, Doris supports adding hoodie related

configurations to catalog properties, which are compatible with Spark Datasource Configs, so users can add `hoodie.datasource.merge.type=skip_merge` in catalog properties to skip merge logs files.

Query Optimization

Doris uses the parquet native reader to read the data files of the COW table, and uses the Java SDK (By calling hudi-bundle through JNI) to read the data files of the MOR table. In `upsert` scenario, there may still remains base files that have not been updated in the MOR table, which can be read through the parquet native reader. Users can view the execution plan of hudi scan through the explain command, where `hudiNativeReadSplits` indicates how many split files are read through the parquet native reader.

```
|0:VHUDI_SCAN_NODE                                                                |
|      table: minbatch_mor_rt                                                     |
|      predicates: `o_orderkey` = 100030752                                       |
|      inputSplitNum=810, totalFileSize=5645053056, scanRanges=810                |
|      partition=80/80                                                            |
|      numNodes=6                                                                 |
|      hudiNativeReadSplits=717/810                                               |
```

Users can view the perfomace of Java SDK through profile, for example:

```
-  HudiJniScanner:  0ns
  -  FillBlockTime:  31.29ms
  -  GetRecordReaderTime:  1m5s
  -  JavaScanTime:  35s991ms
  -  OpenScannerTime:  1m6s
```

1. `OpenScannerTime`: Time to create and initialize JNI reader
2. `JavaScanTime`: Time to read data by Java SDK
3. `FillBlockTime`: Time co convert Java column data into C++ column data
4. `GetRecordReaderTime`: Time to create and initialize Hudi Record Reader

Time Travel

Supports reading snapshots specified in Hudi table.

Every write operation to the Hudi table will generate a new snapshot.

By default, query requests will only read the latest version of the snapshot.

You can use the `FOR TIME AS OF` statement, based on the time of the snapshot to read historical version data. Examples are as follows:

```
SELECT * FROM hudi_tbl FOR TIME AS OF "2022-10-07 17:20:37";

SELECT * FROM hudi_tbl FOR TIME AS OF "20221007172037";
```

Hudi table does not support the `FOR VERSION AS OF` statement. Using this syntax to query the Hudi table will throw an error.

Iceberg Catalog

Limitations

1. Support Iceberg V1/V2.
2. The V2 format only supports Position Delete, not Equality Delete.
3. Supports Parquet format.

Create Catalog

Create Catalog Based on Hive Metastore

It is basically the same as Hive Catalog, and only a simple example is given here. See Hive Catalog for other examples.

```
CREATE CATALOG iceberg PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004',
    'hadoop.username' = 'hive',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:4007',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:4007',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

Create Catalog based on Iceberg API

Use the Iceberg API to access metadata, and support services such as Hadoop File System, Hive, REST, DLF and Glue as Iceberg's Catalog.

Hadoop Catalog

```
CREATE CATALOG iceberg_hadoop PROPERTIES (
    'type'='iceberg',
    'iceberg.catalog.type' = 'hadoop',
    'warehouse' = 'hdfs://your-host:8020/dir/key'
);
```

```
CREATE CATALOG iceberg_hadoop_ha PROPERTIES (
    'type'='iceberg',
    'iceberg.catalog.type' = 'hadoop',
    'warehouse' = 'hdfs://your-nameservice/dir/key',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:4007',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:4007',
    'dfs.client.failover.proxy.provider.your-
```

```
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

Hive Metastore

```
CREATE CATALOG iceberg PROPERTIES (
    'type'='iceberg',
    'iceberg.catalog.type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004',
    'hadoop.username' = 'hive',
    'dfs.nameservices'='your-nameservice',
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.2:4007',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.3:4007',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

AWS Glue

```
CREATE CATALOG glue PROPERTIES (
    "type"="iceberg",
    "iceberg.catalog.type" = "glue",
    "glue.endpoint" = "https://glue.us-east-1.amazonaws.com",
    "glue.access_key" = "ak",
    "glue.secret_key" = "sk"
);
```

For Iceberg properties, see Iceberg Glue Catalog

Alibaba Cloud DLF

see Alibaba Cloud DLF Catalog

REST Catalog

This method needs to provide REST services in advance, and users need to implement the REST interface for obtaining Iceberg metadata.

```
CREATE CATALOG iceberg PROPERTIES (
    'type'='iceberg',
    'iceberg.catalog.type'='rest',
    'uri' = 'http://172.21.0.1:8181'
);
```

If the data is on HDFS and High Availability (HA) is set up, need to add HA configuration to the Catalog.

```
CREATE CATALOG iceberg PROPERTIES (
    'type'='iceberg',
    'iceberg.catalog.type'='rest',
    'uri' = 'http://172.21.0.1:8181',
    'dfs.nameservices'='your-nameservice',
```

```
    'dfs.ha.namenodes.your-nameservice'='nn1,nn2',
    'dfs.namenode.rpc-address.your-nameservice.nn1'='172.21.0.1:8020',
    'dfs.namenode.rpc-address.your-nameservice.nn2'='172.21.0.2:8020',
    'dfs.client.failover.proxy.provider.your-
nameservice'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

Google Dataproc Metastore

```
CREATE CATALOG iceberg PROPERTIES (
    "type"="iceberg",
    "iceberg.catalog.type"="hms",
    "hive.metastore.uris" = "thrift://172.21.0.1:9083",
    "gs.endpoint" = "https://storage.googleapis.com",
    "gs.region" = "us-east-1",
    "gs.access_key" = "ak",
    "gs.secret_key" = "sk",
    "use_path_style" = "true"
);
```

`hive.metastore.uris`: Dataproc Metastore URI，See in Metastore Services： Dataproc Metastore Services.

Iceberg On Object Storage

If the data is stored on S3, the following parameters can be used in properties:

```
"s3.access_key" = "ak"
"s3.secret_key" = "sk"
"s3.endpoint" = "s3.us-east-1.amazonaws.com"
"s3.region" = "us-east-1"
```

The data is stored on Alibaba Cloud OSS:

```
"oss.access_key" = "ak"
"oss.secret_key" = "sk"
"oss.endpoint" = "oss-cn-beijing-internal.aliyuncs.com"
"oss.region" = "oss-cn-beijing"
```

The data is stored on Tencent Cloud COS:

```
"cos.access_key" = "ak"
"cos.secret_key" = "sk"
"cos.endpoint" = "cos.ap-beijing.myqcloud.com"
"cos.region" = "ap-beijing"
```

The data is stored on Huawei Cloud OBS:

```
"obs.access_key" = "ak"
"obs.secret_key" = "sk"
"obs.endpoint" = "obs.cn-north-4.myhuaweicloud.com"
"obs.region" = "cn-north-4"
```

Column type mapping

Iceberg Type	Doris Type
boolean	boolean
int	int
long	bigint
float	float
double	double
decimal(p,s)	decimal(p,s)
date	date
uuid	string
timestamp (Timestamp without timezone)	datetime(6)
timestamptz (Timestamp with timezone)	datetime(6)
string	string
fixed(L)	char(L)
binary	string
list	array
struct	unsupported
map	unsupported
time	unsupported

Time Travel

Supports reading the snapshot specified by the Iceberg table.

Every write operation to the iceberg table will generate a new snapshot.

By default, read requests will only read the latest version of the snapshot.

You can use the `FOR TIME AS OF` and `FOR VERSION AS OF` statements to read historical versions of data based on the snapshot ID or the time when the snapshot was generated. Examples are as follows:

```
SELECT * FROM iceberg_tbl FOR TIME AS OF "2022-10-07 17:20:37";

SELECT * FROM iceberg_tbl FOR VERSION AS OF 868895038966572;
```

In addition, you can use the iceberg_meta table function to query the snapshot information of the specified table.

Paimon Catalog

Instructions for use

1. When data in hdfs,need to put core-site.xml, hdfs-site.xml and hive-site.xml in the conf directory of FE and BE. First read the hadoop configuration file in the conf directory, and then read the related to the environment variable `HADOOP_CONF_DIR` configuration file.
2. The currently adapted version of the Paimon is 0.7.

Create Catalog

Paimon Catalog Currently supports two types of Metastore creation catalogs: * filesystem(default),Store both metadata and data in the file system. * hive metastore,It also stores metadata in Hive metastore. Users can access these tables directly from Hive.

Creating a Catalog Based on FileSystem

HDFS

```
CREATE CATALOG `paimon_hdfs` PROPERTIES (
    "type" = "paimon",
    "warehouse" = "hdfs://HDFS8000871/user/paimon",
    "dfs.nameservices" = "HDFS8000871",
    "dfs.ha.namenodes.HDFS8000871" = "nn1,nn2",
    "dfs.namenode.rpc-address.HDFS8000871.nn1" = "172.21.0.1:4007",
    "dfs.namenode.rpc-address.HDFS8000871.nn2" = "172.21.0.2:4007",
    "dfs.client.failover.proxy.provider.HDFS8000871" = "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider",
    "hadoop.username" = "hadoop"
);
```

MINIO

> Note that.
>
> user need download paimon-s3-0.6.0-incubating.jar
>
> Place it in directory ${DORIS_HOME}/be/lib/java_extensions/preload-extensions and restart be
>
> Starting from version 2.0.2, this file can be placed in BE's `custom_lib/` directory (if it does not exist, just create it manually) to prevent the file from being lost due to the replacement of the lib directory when upgrading the cluster.

```
CREATE CATALOG `paimon_s3` PROPERTIES (
    "type" = "paimon",
    "warehouse" = "s3://bucket_name/paimons3",
    "s3.endpoint" = "http://<ip>:<port>",
```

```
    "s3.access_key" = "ak",
    "s3.secret_key" = "sk"
);
```

OBS

> Note that.
>
> user need download paimon-s3-0.6.0-incubating.jar
>
> Place it in directory `${DORIS_HOME}/be/lib/java_extensions/preload-extensions` and restart be
>
> Starting from version 2.0.2, this file can be placed in BE's `custom_lib/` directory (if it does not exist, just create it manually) to prevent the file from being lost due to the replacement of the lib directory when upgrading the cluster.

```
CREATE CATALOG `paimon_obs` PROPERTIES (
    "type" = "paimon",
    "warehouse" = "obs://bucket_name/paimon",
    "obs.endpoint"="obs.cn-north-4.myhuaweicloud.com",
    "obs.access_key"="ak",
    "obs.secret_key"="sk"
);
```

COS

```
CREATE CATALOG `paimon_cos` PROPERTIES (
    "type" = "paimon",
    "warehouse" = "cosn://paimon-1308700295/paimoncos",
    "cos.endpoint" = "cos.ap-beijing.myqcloud.com",
    "cos.access_key" = "ak",
    "cos.secret_key" = "sk"
);
```

OSS

```
CREATE CATALOG `paimon_oss` PROPERTIES (
    "type" = "paimon",
    "warehouse" = "oss://paimon-zd/paimonoss",
    "oss.endpoint" = "oss-cn-beijing.aliyuncs.com",
    "oss.access_key" = "ak",
    "oss.secret_key" = "sk"
);
```

Creating a Catalog Based on Hive Metastore

```
CREATE CATALOG `paimon_hms` PROPERTIES (
    "type" = "paimon",
    "paimon.catalog.type" = "hms",
```

```
    "warehouse" = "hdfs://HDFS8000871/user/zhangdong/paimon2",
    "hive.metastore.uris" = "thrift://172.21.0.44:7004",
    "dfs.nameservices" = "HDFS8000871",
    "dfs.ha.namenodes.HDFS8000871" = "nn1,nn2",
    "dfs.namenode.rpc-address.HDFS8000871.nn1" = "172.21.0.1:4007",
    "dfs.namenode.rpc-address.HDFS8000871.nn2" = "172.21.0.2:4007",
    "dfs.client.failover.proxy.provider.HDFS8000871" = "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider",
    "hadoop.username" = "hadoop"
);
```

Column Type Mapping

Paimon Data Type	Doris Data Type	Comment
BooleanType	Boolean	
TinyIntType	TinyInt	
SmallIntType	SmallInt	
IntType	Int	
FloatType	Float	
BigIntType	BigInt	
DoubleType	Double	
VarCharType	VarChar	
CharType	Char	
VarBinaryType, BinaryType	Binary	
DecimalType(precision, scale)	Decimal(precision, scale)	
TimestampType,LocalZonedTimestampType	DateTime	
DateType	Date	
ArrayType	Array	Support Array nesting
MapType	Map	Support Map nesting
RowType	Struct	Support Struct nesting (since 2.0.10 & 2.1.3)

FAQ

1. Kerberos

    - Make sure principal and keytab are correct.
    - You need to start a scheduled task (such as crontab) on the BE node, and execute the `kinit -kt your_principal your_keytab` command every certain time (such as 12 hours).

2. Unknown type value: UNSUPPORTED

This is a compatible issue exist in 2.0.2 with Paimon 0.5, you need to upgrade to 2.0.3 or higher to solve this problem. Or patch yourself.

3. When accessing object storage (OSS, S3, etc.), encounter “file system does not support”.

   In versions before 2.0.5 (inclusive), users need to manually download the following jar package and place it in the `${DORIS_HOME}/be/lib/java_extensions/preload-extensions` directory, and restart BE.

   - OSS: paimon-oss-0.6.0-incubating.jar
   - Other Object Storage: paimon-s3-0.6.0-incubating.jar

   No need to download these jars since 2.0.6.

Cloud Service Authentication

Overview

When accessing a service on the cloud, we need to provide the credentials needed to access the service so that the service can be authenticated by IAM of cloud vendors.

AWS

Now Doris support two types of authentication to access AWS service.

Catalog Credentials

The Catalog supports filling in basic Credentials properties, such as: 1. For S3: `s3.endpoint`, `s3.access_key`, `s3.secret_key`。 2. For Glue: `glue.endpoint`, `glue.access_key`, `glue.secret_key`。

When access Glue though Iceberg Catalog, we can access tables on Glue by filling in the following properties:

```
CREATE CATALOG glue PROPERTIES (
    "type"="iceberg",
    "iceberg.catalog.type" = "glue",
    "glue.endpoint" = "https://glue.us-east-1.amazonaws.com",
    "glue.access_key" = "ak",
    "glue.secret_key" = "sk"
);
```

System Credentials

For applications running on AWS resources, such as EC2 instances, this approach enhances security by avoiding hardcoded credentials.

If we create the Catalog but not fill any Credentials in properties, the `DefaultAWSCredentialsProviderChain` will be used to read in the system environment variables or instance profile.

For details about how to configure environment variables and system properties, see: AWS CLI . - The configurable environment variables are: `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、`AWS_SESSION_TOKEN`、`AWS_ROLE_ARN`、`AWS_WEB_IDENTITY_TOKEN_FILE` and so on. - In addition, you can also use aws configure to configure Credentials, the Credentials file will be written to the `~/.aws` directory.

JDBC Catalog

Doris JDBC Catalog supports connecting to different databases that support the JDBC protocol through the standard JDBC interface. This document introduces the general configuration and usage of JDBC Catalog.

Supported databases

Doris JDBC Catalog supports connecting to the following databases:

Database	Description
MySQL	
PostgreSQL	
Oracle	
SQL Server	
ClickHouse	
SAP HANA	
OceanBase	

Configuration

Basic properties

Parameters	Description
`type`	Fixed to `jdbc`
`user`	Data source user name
`password`	Data source password
`jdbc_url`	Data source connection URL
`driver_url`	Path to the data source JDBC driver
`driver_class`	The class name of the data source JDBC driver

Optional properties

Parameters	Default value	Description
`lower_case_table_names`	“false”	Whether to synchronize the library name and table name of jdbc external data source in lowercase
`only_specified_database`	“false”	Whether to synchronize only the Database of the data source specified in the JDBC URL (Database here is the Database level mapped to Doris)

Parameters	Default value	Description
include_database_list	“”	When only_specified_database=true, specify to synchronize multiple Databases, separated by ‘,’. Database names are case-sensitive.
exclude_database_list	“”	When only_specified_database=true, specify multiple Databases that do not need to be synchronized, separated by ‘,’. Database names are case-sensitive.

Connection pool properties

Parameter	Default value	Description
connection_pool_min_size	1	Defines the minimum number of connections in the connection pool, which is used to initialize the connection pool and ensure that at least this number of connections are active when the keep-alive mechanism is enabled.
connection_pool_max_size	30	Defines the maximum number of connections in the connection pool. Each FE or BE node corresponding to each Catalog can hold up to this number of connections.
connection_pool_max_wait_time	5000	Defines the maximum number of milliseconds the client will wait for a connection if there is no available connection in the connection pool.
connection_pool_max_life_time	1800000	Set the maximum time (in milliseconds) that a connection remains active in the connection pool. Timed out connections will be recycled. At the same time, half of this value will serve as the minimum eviction idle time of the connection pool, and connections that reach this time will become eviction candidates.
connection_pool_keep_alive	false	Only valid on BE nodes, used to decide whether to keep connections that have reached the minimum eviction idle time but have not reached the maximum lifetime active. Turned off by default to reduce unnecessary resource usage.

Property Notes

Driver package path and security

driver_url can be specified in the following three ways:

1. File name. Such as mysql-connector-j-8.3.0.jar. The Jar package needs to be pre-stored in jdbc_drivers/ under the FE and BE deployment directories. Under contents. The system will automatically search in this directory. The location of this directory can also be modified by the jdbc_drivers_dir configuration in fe.conf and be.conf.
2. Local absolute path. Such as file:///path/to/mysql-connector-j-8.3.0.jar. Jar packages need to be stored in the paths specified by all FE/BE nodes in advance.
3. HTTP address. For example: http://repo1.maven.org/maven2/com/mysql/mysql-connector-j/8.3.0/mysql-connector-j-8.3.0.jar The system will download the Driver file from this Http address. Only HTTP services without authentication are supported.

Driver package security

In order to prevent the use of a Driver Jar package with an unallowed path when creating the Catalog, Doris will perform path management and checksum checking on the Jar package.

1. For the above method 1, the `jdbc_drivers_dir` configured by the Doris default user and all Jar packages in its directory are safe and will not be path checked.

2. For the above methods 2 and 3, Doris will check the source of the Jar package. The checking rules are as follows:

    - Control the allowed driver package paths through the FE configuration item `jdbc_driver_secure_path`. This configuration item can configure multiple paths, separated by semicolons. When this option is configured, Doris It will check whether the partial prefix of the path of driver_url in Catalog properties is in `jdbc_driver_secure_path`. If it is not in it, the creation will be refused. Catalog.
    - This parameter defaults to *, which means Jar packages of all paths are allowed.
    - If the configuration `jdbc_driver_secure_path` is empty, it also means that Jar packages of all paths are allowed.

    :::info remarks For example, configure `jdbc_driver_secure_path = "file:///path/to/jdbc_drivers;http://path/to/jdbc_drivers"`:

    Then only driver package paths starting with `file:///path/to/jdbc_drivers` or `http://path/to/jdbc_drivers` are allowed. :::

3. When creating a Catalog, you can specify the checksum of the driver package through the `checksum` parameter. Doris will verify the driver package after loading the driver package. If the verification fails, the creation will be rejected. Catalog.

> **remarks**
>
> The above verification will only be performed when the catalog is created, and the already created catalog will not be verified again.

Lowercase name synchronization

When `lower_case_table_names` is set to `true`, Doris is able to query non-lowercase databases and tables by maintaining a mapping of lowercase names to actual names on the remote system

Notice:

1. In versions before Doris 2.0.3, it is only valid for Oracle database. When querying, all library names and table names will be converted to uppercase before querying Oracle, for example:

    Oracle has the TEST table in the TEST space. When Doris creates the Catalog, set `lower_case_table_names` to `true`, then Doris can query the TEST table through `select * from oracle_catalog.test.test`, and Doris will automatically format test.test into TEST.TEST is sent to Oracle.

It should be noted that this is the default behavior, which also means that lowercase table names in Oracle cannot be queried.

For other databases, you still need to specify the real library name and table name when querying.

2. In Doris 2.0.3 and later 2.0.x versions, it is valid for all databases. When querying, all library names and table names will be converted into real names and then queried. If you upgrade from an old version To 2.0.3, `Refresh <catalog_name>` is required to take effect.

   However, if the database or table names differ only in case, such as `Doris` and `doris`, Doris cannot query them due to ambiguity.

3. When the `lower_case_table_names` parameter of the FE parameter is set to `1` or `2`, the `lower_case_table_names` parameter of the JDBC Catalog must be set to `true`. If the `lower_case_table_names` of the FE parameter is set to `0`, the JDBC Catalog parameter can be `true` or `false`, defaulting to `false`. This ensures consistency and predictability in how Doris handles internal and external table configurations.

Specify synchronization database

`only_specified_database`: Whether to synchronize only the Database of the data source specified in the JDBC URL. The default value is `false`, which means synchronizing all Databases in the JDBC URL.

`include_database_list`: Only effective when `only_specified_database=true`, specify the Schema of PostgreSQL that needs to be synchronized, separated by ‘,’. Schema names are case-sensitive.

`exclude_database_list`: Only effective when `only_specified_database=true`, specify the Schema of PostgreSQL that does not need to be synchronized, separated by ‘,’. Schema names are case-sensitive.

> **remarks**
>
> - The Database mentioned in the above three parameters refers to the Database level in Doris, not the Database level of external data sources. For specific mapping relationships, please refer to each data source document.
> - When `include_database_list` and `exclude_database_list` have overlapping database configurations, `exclude_database_list` will take effect first.

Connection pool configuration

In Doris, each FE and BE node maintains a connection pool, which avoids frequently opening and closing separate data source connections. Each connection in the connection pool can be used to establish a connection with the data source and execute queries. When the task is completed, these connections are returned to the pool for reuse, which not only improves performance, but also reduces the overhead of establishing connections and helps prevent the data source’s connection limit from being reached.

The connection pool size can be adjusted to better suit your workload. Typically, the minimum number of connections in a connection pool should be set to 1 to ensure that at least one connection is active when the

keepalive mechanism is enabled. The maximum number of connections in the connection pool should be set to a reasonable value to avoid too many connections occupying resources.

At the same time, in order to avoid accumulating too many unused connection pool caches on BE, you can specify the time interval for clearing the cache by setting the `jdbc_connection_pool_cache_clear_time_sec` parameter of BE. The default value is 28800 seconds (8 hours). After this interval, BE will forcefully clear all connection pool caches that have not been used for more than this time.

> **When using Doris JDBC Catalog to connect to external data sources, you need to be careful when updating database credentials.**
>
> Doris maintains active connections through a connection pool to respond to queries quickly. However, after the credentials are changed, the connection pool may continue to use the old credentials to try to establish new connections and fail. Such erroneous attempts are repeated as the system attempts to maintain a certain number of active connections, and in some database systems, frequent failures may result in account lockout. It is recommended that when credentials must be changed, the Doris JDBC Catalog configuration is updated synchronously and the Doris cluster is restarted to ensure that all nodes use the latest credentials to prevent connection failures and potential account lockouts.
>
> Account lockouts you may encounter are as follows:
>
> MySQL: account is locked
>
> Oracle: ORA-28000: the account is locked
>
> SQL Server: Login is locked out

Insert transaction

Doris' data is written to the JDBC Catalog in a batch manner. If the import is interrupted midway, the previously written data may need to be rolled back. Therefore, JDBC Catalog supports transactions when data is written. Transaction support needs to be set by setting session variable: `enable_odbc_transcation`.

```
set enable_odbc_transcation = true;
```

Transactions ensure the atomicity of JDBC Catalog data writing, but will reduce the performance of data writing to a certain extent. You can consider turning on this function as appropriate.

Example

Here, MySQL is used as an example to show how to create a MySQL Catalog and query the data in it.

Create a Catalog named `mysql`:

```
CREATE CATALOG mysql PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="secret",
    "jdbc_url" = "jdbc:mysql://example.net:3306",
```

```
    "driver_url" = "mysql-connector-j-8.3.0.jar",
    "driver_class" = "com.mysql.cj.jdbc.Driver"
)
```

View all databases in this catalog by running SHOW DATABASES:

```
SHOW DATABASES FROM mysql;
```

If you have a MySQL database named test, you can view the tables in that database by running SHOW TABLES:

```
SHOW TABLES FROM mysql.test;
```

Finally, you can access the table in the MySQL database:

```
SELECT * FROM mysql.test.table;
```

Statement transparent transmission

Doris supports direct execution of DDL, DML statements and query statements of JDBC data sources through transparent transmission.

Transparent transmission of DDL and DML

```
CALL EXECUTE_STMT("catalog_name", "raw_stmt_string");
```

The `EXECUTE_STMT()` procedure takes two parameters:

- Catalog Name: Currently only JDBC type Catalog is supported.
- Execution statements: Currently only DDL and DML statements are supported, and the syntax corresponding to the data source needs to be used directly.

```
CALL EXECUTE_STMT("jdbc_catalog", "insert into db1.tbl1 values(1,2), (3, 4)");

CALL EXECUTE_STMT("jdbc_catalog", "delete from db1.tbl1 where k1 = 2");

CALL EXECUTE_STMT("jdbc_catalog", "create table dbl1.tbl2 (k1 int)");
```

Transparent query

```
query(
  "catalog" = "catalog_name",
  "query" = "select * from db_name.table_name where condition"
  );
```

The `query` table function takes two parameters:

- `catalog`: Catalog name, which needs to be filled in according to the name of the Catalog.

- `query`: The query statement that needs to be executed, and the syntax corresponding to the data source needs to be used directly.

```
select * from query("catalog" = "jdbc_catalog", "query" = "select * from db_name.table_name where condition");
```

Principles and Limitations

Through the `CALL EXECUTE_STMT()` command, Doris will directly send the SQL statement written by the user to the JDBC data source corresponding to the Catalog for execution. Therefore, this operation has the following limitations:

- The SQL statement must be the syntax corresponding to the data source. Doris will not perform syntax and semantic checks.
- It is recommended that the table name referenced in the SQL statement be a fully qualified name, that is, in the format of `db.tbl`. If db is not specified, the db name specified in the JDBC URL of the JDBC Catalog is used.
- SQL statements cannot reference library tables other than JDBC data sources, nor can they reference Doris library tables. However, you can reference library tables in the JDBC data source but not synchronized to the Doris JDBC Catalog.
- When executing a DML statement, the number of rows inserted, updated, or deleted cannot be obtained, but only whether the command was successfully executed.
- Only users with LOAD permissions on the Catalog can execute the `CALL EXECUTE_STMT()` command.
- Only users with SELECT permissions on Catalog can execute the `query()` table function.
- The supported data types of the data read by the `query` table function are consistent with the data types supported by the queried catalog type.

Troubleshooting connection pool issues

1. In versions less than 2.0.5, the connection pool related configuration can only be configured in JAVA_OPTS of BE conf, refer to version 2.0.4 be.conf.
2. In versions 2.0.5 and later, connection pool related configurations can be configured in the Catalog properties, refer to Connection Pool Properties.
3. The connection pool used by Doris was changed from Druid to HikariCP starting from 2.0.10 (2.0 Release) and 2.1.3 (2.1 Release), so the connection pool related errors and troubleshooting methods are different, refer to the following

Druid connection pool version

Initialize datasource failed: CAUSED BY: GetConnectionTimeoutException: wait millis 5006, active 10, maxActive 10, creating 1 * Reason 1: Too many queries cause the number of connections to exceed the configuration * Reason 2: The connection pool count is abnormal and the active count does not decrease. * Solution * alter catalog set properties ('connection_pool_max_size' = '100'); Temporarily increase the connection pool capacity by adjusting the number of connections, and the connection pool cache can be refreshed in this way * Upgrade to replace the connection pool to Hikari version

Initialize datasource failed: CAUSED BY: GetConnectionTimeoutException: wait millis 5006, active 10, maxActive 0, creating 1 * Reason 1: Network failure * Reason 2: The network delay is high, causing the

connection creation to take more than 5s * Solution * Check network * alter catalog set properties ('connection_pool_max_wait' = '10000'); Increase the timeout

HikariCP connection pool version

```
Connection is not available, request timed out after 5000ms
```

Possible reasons:

- **Cause 1**: Network problem (e.g. server unreachable)
- **Cause 2**: Authentication issues, such as invalid username or password
- **Cause 3**: The network latency is too high, causing the connection creation to exceed the 5 second timeout
- **Cause 4**: Too many concurrent queries, exceeding the maximum number of connections configured in the connection pool

Solution:

- **If there are only "Connection is not available, request timed out after 5000ms" errors**, please check **Cause 3** and **Cause 4**:
    - Check for high network latency or resource exhaustion.
    - Increase the maximum number of connections in the connection pool:

      ```
      ALTER CATALOG <catalog_name> SET PROPERTIES ('connection_pool_max_size' = '100');
      ```

    - Increase the connection timeout:

      ```
      ALTER CATALOG <catalog_name> SET PROPERTIES ('connection_pool_max_wait_time' = '10000');
      ```

- **If there are other error messages besides "Connection is not available, request timed out after 5000ms"**, please check these additional errors:
    - **Network problems** (e.g. server unreachable) may cause connection failure. Please check whether the network connection is normal.
    - **Authentication issues** (e.g. invalid username or password) may also cause connection failure. Please check the database credentials used in the configuration to ensure that the username and password are correct.
    - Based on specific error messages, investigate network, database, or authentication-related issues to identify the root cause.

MySQL

Doris JDBC Catalog supports connecting to MySQL database through the standard JDBC interface. This document describes how to configure a MySQL database connection.

Terms and Conditions

To connect to a MySQL database you need

- MySQL 5.7, 8.0 or higher
- JDBC driver for MySQL database, you can download it from Maven repository Download the latest or specified version of the MySQL JDBC driver. **It is recommended to use MySQL Connector/J 8.0.31 and above.**
- Doris Network connection between each FE and BE node and the MySQL server, the default port is 3306.

Connect to MySQL

```
CREATE CATALOG mysql PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="secret",
    "jdbc_url" = "jdbc:mysql://example.net:3306",
    "driver_url" = "mysql-connector-j-8.3.0.jar",
    "driver_class" = "com.mysql.cj.jdbc.Driver"
)
```

> **remarks**
>
> `jdbc_url` defines the connection information and parameters to be passed to the MySQL JDBC driver. The parameters for supported URLs can be found in the MySQL Development Guide.

Connection security

If you configured TLS with a globally trusted certificate installed on the data source, you can enable TLS between the cluster and the data source by appending the parameter to the JDBC connection string set in the jdbc_url property.

For example, with MySQL Connector/J version 8.0, use the sslMode parameter to secure the connection over TLS. By default, this parameter is set to PREFERRED, which protects the connection if the server is enabled. You can also set this parameter to REQUIRED, which will cause the connection to fail if TLS is not established.

You can configure this by adding the sslMode parameter to `jdbc_url`:

```
"jdbc_url" = "jdbc:mysql://example.net:3306/?sslMode=REQUIRED"
```

For more information about TLS configuration options, see the MySQL JDBC security documentation.

Hierarchical mapping

When mapping MySQL, a Database in Doris corresponds to a Database in MySQL. The Table under Doris' Database corresponds to the Tables under the Database in MySQL. That is, the mapping relationship is as follows:

Doris	MySQL
Catalog	MySQL Server
Database	Database
Table	Table

Type mapping

MySQL to Doris type mapping

MYSQL Type	Doris Type	Comment
BOOLEAN	TINYINT	
TINYINT	TINYINT	
SMALLINT	SMALLINT	
MEDIUMINT	INT	
INT	INT	
BIGINT	BIGINT	
UNSIGNED TINYINT	SMALLINT	
UNSIGNED MEDIUMINT	INT	
UNSIGNED INT	BIGINT	
UNSIGNED BIGINT	LARGEINT	
FLOAT	FLOAT	
DOUBLE	DOUBLE	
DECIMAL	DECIMAL	
UNSIGNED DECIMAL(p,s)	DECIMAL(p+1,s) / STRING	
DATE	DATE	
TIMESTAMP	DATETIME	
DATETIME	DATETIME	
YEAR	SMALLINT	

MYSQL Type	Doris Type	Comment
TIME	STRING	
CHAR	CHAR	
VARCHAR	VARCHAR	
JSON	STRING	
SET	STRING	
ENUM	STRING	
BIT	BOOLEAN/STRING	
TINYTEXT,TEXT,MEDIUMTEXT,LONGTEXT	STRING	
BLOB,MEDIUMBLOB,LONGBLOB,TINYBLOB	STRING	
BINARY,VARBINARY	STRING	
Other	UNSUPPORTED	

- Doris does not support UNSIGNED data types, so UNSIGNED data types will be mapped to Doris corresponding data types that are an order of magnitude larger.

- UNSIGNED DECIMAL(p,s) will be mapped to DECIMAL(p+1,s) or STRING. Note that when this type is mapped to String, it can only support queries and cannot write to MySQL.
- In order to better balance reading and computing performance, Doris will map the JSON type to the STRING type.
- Doris does not support the BIT type. The BIT type will be mapped to BOOLEAN when BIT(1) is used, and to STRING in other cases.
- Doris does not support YEAR type, YEAR type will be mapped to SMALLINT.
- Doris does not support the TIME type, and the TIME type will be mapped to STRING.

Timestamp type processing

The Java part of BE uses the JVM time zone when reading data from the JDBC type catalog. The JVM time zone defaults to the time zone of the BE deployment machine, which affects the time zone conversion when JDBC reads data.

In order to ensure time zone consistency, it is recommended to set the JVM time zone in JAVA_OPTS of be.conf to be consistent with the `time_zone` of the Doris session.

When reading MySQL's TIMESTAMP type, add parameters to the JDBC URL: `connectionTimeZone=LOCAL` and `forceConnectionTimeZoneToSession=true`. These parameters are applicable to MySQL Connector/J 8 and above and ensure that the time read is the Doris BE JVM time zone, not the MySQL session time zone.

Query optimization

Predicate pushdown

1. When executing a query like `where dt = '2022-01-01'`, Doris can push these filtering conditions down to the external data source, thereby directly excluding data that does not meet the conditions at the data source level, reducing unnecessary data acquisition and transmission. This greatly improves query performance while also reducing the load on external data sources.

2. When FE conf `enable_func_pushdown` is set to true, the function conditions after where will also be pushed down to the external data source. Doris will automatically identify some functions that are not supported by MySQL, which can be viewed through explain sql.

   Currently, the functions that Doris will not push down to MySQL by default are as follows:

Function
DATE_TRUNC
MONEY_FORMAT

Row limit

If you have the limit keyword in the query, Doris will push the limit down to MySQL to reduce the amount of data transfer.

Escape characters

Doris will automatically add the escape character (``) to the field names and table names in the query statements sent to MySQL to avoid conflicts between field names, table names and MySQL internal keywords.

Troubleshoot connection exceptions

- Communications link failure The last packet successfully received from the server was 7 milliseconds ago.
    - reason:
        - Internet problem:
            - The network is unstable or the connection is interrupted.
            - Network latency between client and server is too high.
        - MySQL server settings
            - The MySQL server may be configured with connection timeout parameters, such as wait_timeout or interactive_timeout, causing the connection to timeout and be closed.
        - Firewall settings
            - Firewall rules may be blocking communication between the client and the server.
        - Connection pool settings
            - The configuration connection_pool_max_life_time in the connection pool may cause the connection to be closed or recycled, or the connection may not be detected in time.

        - Server resource issues
            - The MySQL server may have insufficient resources to handle new connection requests.
        - Client configuration
            - Client JDBC driver configuration error, for example, the autoReconnect parameter is not set or set improperly.
    - solve
        - Check network connection:
            - Confirm that the network connection between the client and the server is stable to avoid excessive network latency.
        - Check MySQL server configuration:
            - Review and adjust the MySQL server's wait_timeout and interactive_timeout parameters to ensure they are set appropriately.
        - Check firewall configuration:
            - Verify that firewall rules allow communication between client and server.
        - Adjust connection pool settings:
            - Check and adjust the connection pool configuration parameter connection_pool_max_life_time to ensure it is smaller than MySQL's wait_timeout and interactive_timeout parameters and larger than the SQL with the longest execution time
        - Monitor server resources:
            - Monitor the resource usage of the MySQL server to ensure that there are sufficient resources to handle connection requests.
        - Optimize client configuration:
            - Confirm that the configuration parameters of the JDBC driver are correct, such as autoReconnect=true, to ensure that the connection can automatically reconnect after being interrupted.
- java.io.EOFException MESSAGE: Can not read response from server. Expected to read 819 bytes, read 686 bytes before connection was unexpectedly lost.
    - Reason: The connection was killed by MySQL or MySQL crashed
    - Solution: Check whether MySQL has a mechanism to actively kill connections, or whether MySQL crashes because of too large a query.

FAQ

1. The emoji expressions read and written in MySQL are garbled.

   When Doris performs MySQL Catalog query, because the default utf8 encoding in MySQL is utf8mb3, it cannot express emoji expressions that require 4-byte encoding. Here you need to modify the encoding of MySQL to utf8mb4 to support 4-byte encoding.

   Configuration items can be modified globally

   ```
   Modify the my.ini file in the mysql directory (for Linux systems, it is the my.cnf file in the etc directory)
   [client]
   default-character-set=utf8mb4

   [mysql]
   Set mysql default character set
   ```

```
default-character-set=utf8mb4

[mysqld]
Set mysql character set server
character-set-server=utf8mb4
collation-server=utf8mb4_unicode_ci
init_connect='SET NAMES utf8mb4

Modify the types of corresponding tables and columns
ALTER TABLE table_name MODIFY colum_name VARCHAR(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
ALTER TABLE table_name CHARSET=utf8mb4;
SET NAMES utf8mb4
```

2. An exception occurred while reading the MySQL DATE/DATETIME type.

```
ERROR 1105 (HY000): errCode = 2, detailMessage = (10.16.10.6)[INTERNAL_ERROR]UdfRuntimeException: get next block failed:
CAUSED BY: SQLException: Zero date value prohibited
CAUSED BY: DataReadException: Zero date value prohibited
```

   This is because the default handling of illegal DATE/DATETIME in JDBC is to throw an exception, and this behavior can be controlled through the parameter `zeroDateTimeBehavior`.

   Optional parameters are: `exception`, `convertToNull`, `round`, respectively: exception error report, converted to NULL value, converted to “0001-01-01 00:00:00”;

   You need to add `zeroDateTimeBehavior=convertToNull` to the end of the JDBC connection string when creating the Catalog `jdbc_url`, such as `"jdbc_url" = "jdbc:mysql://127.0.0.1:3306/test?zeroDateTimeBehavior=convertToNull"` In this case, JDBC will convert 0000-00-00 or 0000-00-00 00:00:00 into null, and then Doris will process all Date/DateTime type columns in the current Catalog as nullable types, so that It can be read normally.

3. When reading MySQL Catalog or other JDBC Catalog, a class loading failure occurs

   Such as the following exception:

```
failed to load driver class com.mysql.cj.jdbc.driver in either of hikariconfig class loader
```

   This is because when creating the Catalog, the driver_class filled in is incorrect and needs to be filled in correctly. For example, the above example has a case problem and should be filled in as `"driver_class" = "com.mysql.cj.jdbc.Driver"`

4. Communication link abnormality occurs when reading MySQL

   If the following error occurs:

```
ERROR 1105 (HY000): errCode = 2, detailMessage = PoolInitializationException: Failed to initialize pool: Communications
link failure

The last packet successfully received from the server was 7 milliseconds ago. The last packet sent successfully to the
server was 4 milliseconds ago.
CAUSED BY: CommunicationsException: Communications link failure

The last packet successfully received from the server was 7 milliseconds ago. The last packet sent successfully to the
```

```
server was 4 milliseconds ago.
CAUSED BY: SSLHandshakeExcepti
```

You can view the be.out log of be

If the following information is included:

```
WARN: Establishing SSL connection without server's identity verification is not recommended.
According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit
option isn't set.
For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'.
You need either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server
certificate verification.
```

You can add `?useSSL=false` to the end of the JDBC connection string when creating the Catalog, such as `"jdbc_url" = "jdbc:mysql://127.0.0.1:3306/test?useSSL=false"`

5. When querying a large amount of MySQL data, if the query is occasionally successful, the following error will occasionally be reported. When this error occurs, all MySQL connections are disconnected and cannot be connected to the MySQL Server. After a while, MySQL returns to normal, but All previous connections are gone:

   ```
   ERROR 1105 (HY000): errCode = 2, detailMessage = [INTERNAL_ERROR]UdfRuntimeException: JDBC executor sql has error:
   CAUSED BY: CommunicationsException: Communications link failure
   The last packet successfully received from the server was 4,446 milliseconds ago. The last packet sent successfully to the
   server was 4,446 milliseconds ago.
   ```

   When the above phenomenon occurs, it may be that MySQL Server's own memory or CPU resources are exhausted, causing the MySQL service to be unavailable. You can try to increase the memory or CPU configuration of MySQL Server.

6. During the process of querying MySQL, if it is found that the query results are inconsistent with the query results in the MySQL library

   First, check whether the string in the query field is case-sensitive. For example, there is a field c_1 in Table that contains two pieces of data: "aaa" and "AAA". If case-sensitivity of strings is not specified when initializing the MySQL database, then MySQL does not case-sensitive strings by default, but in Doris is strictly case-sensitive, so the following situations may occur:

   ```
   MySQL behavior:
   select count(c_1) from table where c_1 = "aaa"; The string size is not distinguished, so the result is: 2

   Doris behavior:
   select count(c_1) from table where c_1 = "aaa"; strictly distinguishes the string size, so the result is: 1
   ```

   If the above phenomenon occurs, it needs to be adjusted according to needs, as follows:

   Add the "BINARY" keyword to force case sensitivity when querying in MySQL: `select count(c_1) from table where BINARY c_1 = "aaa";`

   Or specify when creating a table in MySQL: `CREATE TABLE table (c_1 VARCHAR(255) CHARACTER SET binary);`

Or specify collation rules to make case sensitive when initializing the MySQL database: `[mysqld] character-set-server=utf8 collation-server=utf8_bin [client] default-character-set=utf8 [mysql] default-character-set=utf8`

7. When querying MySQL, if you are stuck for a long time and no results are returned, or if you are stuck for a long time and a large number of write lock logs appear in fe.warn.log, you can try adding socketTimeout to the URL, for example: `jdbc:mysql ://host:port/database?socketTimeout=30000`, prevents the JDBC client from waiting indefinitely after the connection is closed by MySQL.

8. In the process of using MySQL Catalog, it is found that the JVM memory or the number of Threads in FE continues to grow and does not decrease, and a Forward to master connection timed out error may appear at the same time.

   Print the FE thread stack `jstack fe_pid > fe.js`. If a large number of `mysql-cj-abandoned-connection-cleanup` threads appear, it means there is a problem with the MySQL JDBC driver.

   Proceed as follows:

   1. Upgrade the MySQL JDBC driver to version 8.0.31 and above
   2. Add the `-Dcom.mysql.cj.disableAbandonedConnectionCleanup=true` parameter to JAVA_OPTS in the FE and BE conf files, disable the connection cleanup function of the MySQL JDBC driver, and restart the cluster

   Note: If the version of Doris is 2.0.13 and above (2.0 Release), or 2.1.5 and above (2.1 Release), there is no need to increase this parameter, because Doris has disabled the connection cleaning function of the MySQL JDBC driver by default. . Just change the MySQL JDBC driver version. However, the Doris cluster needs to be restarted to clean up the previous Threads.

PostgreSQL

Doris JDBC Catalog supports connecting to PostgreSQL databases through the standard JDBC interface. This document describes how to configure a PostgreSQL database connection.

Terms and Conditions

To connect to a PostgreSQL database, you need

- PostgreSQL 11.x or higher
- JDBC driver for PostgreSQL database, which you can download from the Maven repository Download the latest or specified version of the PostgreSQL JDBC driver. **It is recommended to use PostgreSQL JDBC Driver 42.5.x and above.**
- Doris Network connection between each FE and BE node and the PostgreSQL server, default port is 5432.

Connect to PostgreSQL

```
CREATE CATALOG postgresql PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="secret",
    "jdbc_url" = "jdbc:postgresql://example.net:5432/postgres",
    "driver_url" = "postgresql-42.5.6.jar",
    "driver_class" = "org.postgresql.Driver"
)
```

> **remarks**
>
> `jdbc_url` defines the connection information and parameters to be passed to the PostgreSQL JDBC driver. Parameters for supported URLs can be found in the PostgreSQL JDBC Driver Documentation.

Connection security

If you configured TLS with a globally trusted certificate installed on the data source, you can enable TLS between the cluster and the data source by appending the parameter to the JDBC connection string set in the jdbc_url property.

For example, for version 42 of the PostgreSQL JDBC driver, enable TLS by adding the ssl=true parameter to the jdbc_url configuration property:

```
"jdbc_url"="jdbc:postgresql://example.net:5432/database?ssl=true"
```

For more information about TLS configuration options, see the PostgreSQL JDBC Driver documentation.

Hierarchical mapping

When mapping PostgreSQL, a Database in Doris corresponds to a Schema under the specified database in PostgreSQL (such as the schemas under `postgres` in the `jdbc_url` parameter in the example). The Table under Doris' Database corresponds to the Tables under the Schema in PostgreSQL. That is, the mapping relationship is as follows:

Doris	PostgreSQL
Catalog	Database
Database	Schema
Table	Table

Type mapping

PostgreSQL to Doris type mapping

PostgreSQL Type	Doris Type	Comment
boolean	BOOLEAN	
smallint/int2	SMALLINT	
integer/int4	INT	
bigint/int8	BIGINT	
decimal/numeric	DECIMAL	
real/float4	FLOAT	
double precision	DOUBLE	
smallserial	SMALLINT	
serial	INT	
bigserial	BIGINT	
char	CHAR	
varchar/text	STRING	
timestamp/timestampz	DATETIME	
date	DATE	
json/jsonb	STRING	
time	STRING	
interval	STRING	
point/line/lseg/box/path/polygon/circle	STRING	
cidr/inet/macaddr	STRING	
bit	BOOLEAN/STRING	

PostgreSQL Type	Doris Type	Comment
uuid	STRING	
Other	UNSUPPORTED	

- No-precision numeric will be mapped to the String type. When performing numerical calculations, it needs to be converted to the DECIMAL type first, and write-back is not supported.

- In order to better balance reading and computing performance, Doris will map the JSON type to the STRING type.
- Doris does not support the BIT type. The BIT type will be mapped to BOOLEAN when BIT(1) is used, and to STRING in other cases.
- Doris does not support the time type, and the TIME type will be mapped to STRING.

Timestamp type processing

Since Doris does not support timestamp types with time zones, when reading the timestampz type of PostgreSQL, Doris will map it to the DATETIME type and convert it to the time in the local time zone when reading.

And since the Java part of BE uses the JVM time zone when reading data from the JDBC type Catalog. The JVM time zone defaults to the time zone of the BE deployment machine, which affects the time zone conversion when JDBC reads data.

In order to ensure time zone consistency, it is recommended to set the JVM time zone in JAVA_OPTS of be.conf to be consistent with the `time_zone` of the Doris session.

Query optimization

Predicate pushdown

When executing a query like `where dt = '2022-01-01'`, Doris can push these filtering conditions down to the external data source, thereby directly excluding data that does not meet the conditions at the data source level, reducing inaccuracies. Necessary data acquisition and transmission. This greatly improves query performance while also reducing the load on external data sources.

Row limit

If you have the limit keyword in the query, Doris will push the limit down to PostgreSQL to reduce the amount of data transfer.

Escape characters

Doris will automatically add the escape character (“) to the field names and table names in the query statements sent to PostgreSQL to avoid conflicts between the field names and table names and PostgreSQL internal keywords.

Oracle

Doris JDBC Catalog supports connecting to Oracle databases through the standard JDBC interface. This document describes how to configure an Oracle database connection.

Terms and Conditions

To connect to an Oracle database, you need

- Oracle 19c, 18c, 12c, 11g or 10g.
- JDBC driver for Oracle database, you can download the latest or specified version of Oracle JDBC driver from Maven repository.
- Doris Network connection between each FE and BE node and Oracle server, default port is 1521; if Oracle RAC is ONS-enabled, port 6200 is also required.

Connect to Oracle

```
CREATE CATALOG oracle PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="secret",
    "jdbc_url" = "jdbc:oracle:thin:@example.net:1521:orcl",
    "driver_url" = "ojdbc8.jar",
    "driver_class" = "oracle.jdbc.driver.OracleDriver"
)
```

> **remarks**
>
> `jdbc_url` defines the connection information and parameters to be passed to the JDBC driver. When using the Oracle JDBC Thin driver, the syntax of the URL may vary depending on your Oracle configuration. For example, if you are connecting to an Oracle SID or Oracle service name, the connection URL will be different. For more information, see Oracle Database JDBC Driver Documentation . The above example URL connects to an Oracle SID named `orcl`.

Hierarchical mapping

When mapping Oracle, a Database in Doris corresponds to a User in Oracle. The Table under Doris's Database corresponds to the Table that the User has permission to access in Oracle. That is, the mapping relationship is as follows:

Doris	Oracle
Catalog	Database
Database	User

Doris	Oracle
Table	Table

Type mapping

Oracle to Doris type mapping

Oracle Type	Doris Type	Comment
number(p) / number(p,0)	TINYINT/SMALLINT/INT/BIGINT/LARGEINT	Doris will select the corresponding type according to the size of p: `p < 3` -> `TINYINT`; `p < 5` -> `SMALLINT`; `p < 10` -> `INT`; `p < 19` -> `BIGINT`; `p > 19` -> `LARGEINT`
number(p,s), [if(s>0 && p>s)]	DECIMAL(p,s)	
number(p,s), [if(s>0 && p < s)]	DECIMAL(s,s)	
number(p,s), [if(s<0)]	TINYINT/SMALLINT/INT/BIGINT/LARGEINT	If s<0, Doris will set p to p+|s|, and perform the sum The same mapping as number(p) / number(p,0)
number		Doris currently does not support oracle types that do not specify p and s
decimal	DECIMAL	
float/real	DOUBLE	
DATE	DATETIME	
TIMESTAMP	DATETIME	
CHAR/NCHAR	STRING	
VARCHAR2/NVARCHAR2	STRING	
LONG/ RAW/ LONG RAW/ INTERVAL	STRING	
Other	UNSUPPORTED	

Query optimization

Predicate pushdown

1. When executing a query like `where dt = '2022-01-01'`, Doris can push these filtering conditions down to the external data source, thereby directly excluding data that does not meet the conditions at the data source level, reducing unnecessary data acquisition and transmission. This greatly improves query performance while also reducing the load on external data sources.

2. When FE conf `enable_func_pushdown` is set to true, the function conditions after where will also be pushed down to the external data source.

The functions that currently support push down to Oracle include:

Function
NVL

Row limit

If you have the limit keyword in the query, Doris will escape the limit to Oracle's `rownum` syntax to reduce the amount of data transfer.

Escape characters

Doris will automatically add the escape character (“) to the field names and table names in the query statements sent to Oracle to avoid conflicts between the field names and table names and Oracle's internal keywords.

FAQ

1. `ONS configuration failed` occurs when creating or querying Oracle Catalog

   Add -Doracle.jdbc.fanEnabled=false to JAVA_OPTS in be.conf and upgrade the driver to https://repo1.maven.org/maven2/com/oracle/database/jdbc/ojdbc8/19.23.0.0/ojdbc8-19.23.0.0.jar

2. `Non supported character set (add orai18n.jar in your classpath): ZHS16GBK` exception occurs when creating or querying Oracle Catalog

   Download orai18n.jar and put it in the `custom_lib/` directory under each FE and BE directory (if not exists, just create it manually) and restart each FE and BE.

SQL Server

Doris JDBC Catalog supports connecting to SQL Server databases through the standard JDBC interface. This document describes how to configure a SQL Server database connection.

Terms and Conditions

To connect to a SQL Server database you need

- SQL Server 2012 or later, or Azure SQL Database.
- JDBC driver for SQL Server database, you can download the latest or specified version of SQL Server JDBC driver from Maven repository. **It is recommended to use SQL Server JDBC Driver 11.2.x and above.**
- Doris Network connection between each FE and BE node and the SQL Server server, default port is 1433.

Connect to SQL Server

```
CREATE CATALOG sqlserver PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="secret",
    "jdbc_url" = "jdbc:sqlserver://<host>:<port>;databaseName=<databaseName>;encrypt=false",
    "driver_url" = "mssql-jdbc-11.2.3.jre8.jar",
    "driver_class" = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
)
```

> **remarks**
>
> `jdbc_url` defines the connection information and parameters to be passed to the SQL Server JDBC driver. URL support is provided in SQL Server JDBC Driver Documentation parameters.

Connection security

The JDBC driver and connector automatically use Transport Layer Security (TLS) encryption and certificate verification. This requires configuring the appropriate TLS certificate on the SQL Server database host.

If you have not set up the necessary configuration, you can disable encryption in the connection string using the encrypt attribute:

```
"jdbc_url"="jdbc:sqlserver://<host>:<port>;databaseName=<databaseName>;encrypt=false"
```

The TLS section of the SQL Server JDBC driver documentation describes trustServerCertificate in detail , hostNameInCertificate, trustStore and trustStorePassword and other parameters.

Hierarchical mapping

When mapping SQLServer, a Database in Doris corresponds to a Schema under the specified Database (`<databaseName>` in the `jdbc_url` parameter) in SQL Server. The Table under Doris' Database corresponds to the Tables under Schema in SQL Server. That is, the mapping relationship is as follows:

Doris	SQLServer
Catalog	Database
Database	Schema
Table	Table

Type mapping

SQL Server to Doris type mapping

SQL Server Type	Doris Type	Comment
bit	BOOLEAN	
tinyint	SMALLINT	SQLServer's tinyint is an unsigned number, so it is mapped to Doris's SMALLINT
smallint	SMALLINT	
int	INT	
bigint	BIGINT	
real	FLOAT	
float	DOUBLE	
money	DECIMAL(19,4)	
smallmoney	DECIMAL(10,4)	
decimal/numeric	DECIMAL	
date	DATE	
datetime/datetime2/smalldatetime	DATETIMEV2	
char/varchar/text/nchar/nvarchar/ntext	STRING	
time/datetimeoffset	STRING	
timestamp	STRING	Read the hexadecimal display of binary data, no practical significance
Other	UNSUPPORTED	

Query optimization

Predicate pushdown

When executing a query like `where dt = '2022-01-01'`, Doris can push these filtering conditions down to the external data source, thereby directly excluding data that does not meet the conditions at the data source level, reducing inaccuracies. Necessary data acquisition and transmission. This greatly improves query performance while also reducing the load on external data sources.

Row limit

If you have the limit keyword in the query, Doris will escape the limit into SQL Server's `TOP` syntax to reduce the amount of data transfer.

Escape characters

Doris will automatically add the escape character ([]) to the field names and table names in the query statements sent to SQL Server to avoid conflicts between field names, table names and SQL Server internal keywords.

FAQ

1. Certificate authentication exception occurs when connecting to SQL Server

   ```
   SQLServerException: The driver could not establish a secure connection to SQL Server by using Secure Sockets Layer (SSL) encryption.
   Error: "sun.security.validator.ValidatorException: PKIX path building failed:
   sun.security.provider.certpath.SunCertPathBuilderException:
   unable to find valid certification path to requested target". ClientConnectionId:a92f3817-e8e6-4311-bc21-7c66
   ```

   You can add `encrypt=false` to the end of the JDBC connection string when creating the Catalog, such as `"jdbc_url" = "jdbc:sqlserver://127.0.0.1:1433;DataBaseName=doris_test;encrypt=false"`

2. TLS exception occurs when connecting to SQL Server

   ```
   The server selected protocol version TLS10 is not accepted by client preferences [TLS13, TLS12]
   ```

   This is due to a TLS protocol version mismatch between SQL Server and the JDBC client. The connected SQL Server only supports TLS 1.0, and the JAVA environment where the JDBC client is located has TLS 1.0 disabled by default.

   The solution is as follows:

   1. Enable TLS 1.2 on SQL Server. Reference: SQL Server TLS 1.2 support

   2. Enable TLS 1.0 for the JDK.

      ```
      vim ${JAVA_HOME}/lib/security/java.security
      #find this paragraph
      jdk.tls.disabledAlgorithms=SSLv3, TLSv1, TLSv1.1, RC4, DES, MD5withRSA, \
      ```

```
DH keySize < 1024, EC keySize < 224, 3DES_EDE_CBC, anon, NULL, \
include jdk.disabled.namedCurves

#Delete TLSv1, TLSv1.1 and change it to the following
jdk.tls.disabledAlgorithms=SSLv3, RC4, DES, MD5withRSA, \
DH keySize < 1024, EC keySize < 224, anon, NULL, \
include jdk.disabled.namedCurves
```

ClickHouse

Doris JDBC Catalog supports connection to the ClickHouse database through the standard JDBC interface. This document describes how to configure a ClickHouse database connection.

Terms and Conditions

To connect to the ClickHouse database you need

- ClickHouse 23.x or higher (versions below this are not fully tested).
- JDBC driver for ClickHouse database, you can download the latest or specified version of ClickHouse JDBC driver from Maven repository. **ClickHouse JDBC Driver version 0.4.6 is recommended.**
- Doris Network connection between each FE and BE node and the ClickHouse server, default port is 8123.

Connect to ClickHouse

```
CREATE CATALOG clickhouse PROPERTIES (
    "type"="jdbc",
    "user"="default",
    "password"="password",
    "jdbc_url" = "jdbc:clickhouse://example.net:8123/",
    "driver_url" = "clickhouse-jdbc-0.4.6-all.jar",
    "driver_class" = "com.clickhouse.jdbc.ClickHouseDriver"
)
```

> **remarks**
>
> `jdbc_url` defines the connection information and parameters to be passed to the ClickHouse JDBC driver. Parameters for supported URLs can be found in ClickHouse JDBC Driver Configuration.

Connection security

If you configured TLS with a global trust certificate installed on the data source, you can enable TLS between the cluster and the data source by appending parameters to the JDBC connection string set in the jdbc_url property.

For example, enable TLS by adding the ssl=true parameter to the jdbc_url configuration property:

```
"jdbc_url"="jdbc:clickhouse://example.net:8123/db?ssl=true"
```

For more information about TLS configuration options, see Clickhouse JDBC Driver Documentation SSL Configuration Section

Hierarchical mapping

When mapping ClickHouse, a Database in Doris corresponds to a Database in ClickHouse. The Table under Doris' Database corresponds to the Tables under the Database in ClickHouse. That is, the mapping relationship is as follows:

Doris	ClickHouse
Catalog	ClickHouse Server
Database	Database
Table	Table

Type mapping

ClickHouse to Doris type mapping

ClickHouse Type	Doris Type	Comment
Bool	BOOLEAN	
String	STRING	
Date/Date32	DATE	
DateTime/DateTime64	DATETIME	
Float32	FLOAT	
Float64	DOUBLE	
Int8	TINYINT	
Int16/UInt8	SMALLINT	Doris does not have UNSIGNED data type, so expand by an order of magnitude
Int32/UInt16	INT	Doris does not have UNSIGNED data type, so expand by an order of magnitude
Int64/Uint32	BIGINT	Doris does not have UNSIGNED data type, so it is expanded by an order of magnitude
Int128/UInt64	LARGEINT	Doris does not have UNSIGNED data type, so expand it by an order of magnitude
Int256/UInt128/UInt256	STRING	Doris does not have a data type of this magnitude and uses STRING for processing
DECIMAL	DECIMALV3/STRING	Which type will be selected based on the (precision, scale) of the DECIMAL field
Enum/IPv4/IPv6/UUID	STRING	
Array	ARRAY	Array's internal type adaptation logic refers to the above types, nested array is not supported
Other	UNSUPPORTED	

Query optimization

Predicate pushdown

1. When executing a query like `where dt = '2022-01-01'`, Doris can push these filtering conditions down to the external data source, thereby directly excluding data that does not meet the conditions at the data source level, reducing unnecessary data acquisition and transmission. This greatly improves query performance while also reducing the load on external data sources.

2. When FE conf `enable_func_pushdown` is set to true, the function conditions after where will also be pushed down to the external data source.

   The functions that currently support push down to ClickHouse are:

Function
FROM_UNIXTIME
UNIX_TIMESTAMP

Row limit

If you have the limit keyword in the query, Doris will push the limit down to ClickHouse to reduce the amount of data transfer.

Escape characters

Doris will automatically add the escape character (“`”) to the field names and table names in the query statements sent to ClickHouse to avoid conflicts between the field names and table names and ClickHouse internal keywords.

FAQ

1. `NoClassDefFoundError: net/jpountz/lz4/LZ4Factory` error message appears when reading Clickhouse data through ClickHouse Catalog

   You can first download the lz4-1.3.0.jar package and put it in every `custom_lib/` directory under the FE and BE directories (if it does not exist, just create it manually).

SAP HANA

Doris JDBC Catalog supports connection to the SAP HANA database through the standard JDBC interface. This document describes how to configure a SAP HANA database connection.

Terms and Conditions

To connect to the SAP HANA database you need

- SAP HANA 2.0 or higher.
- JDBC driver for SAP HANA database, you can download the latest or specified version of SAP HANA JDBC driver from Maven repository . **It is recommended to use ngdbc version 2.4.51 or above.**
- Doris Network connection between each FE and BE node and the SAP HANA server, default port is 30015.

Connect to SAP HANA

```
CREATE CATALOG saphana PROPERTIES (
    "type"="jdbc",
    "user"="USERNAME",
    "password"="PASSWORD",
    "jdbc_url" = "jdbc:sap://Hostname:Port/?optionalparameters",
    "driver_url" = "ngdbc-2.4.51.jar",
    "driver_class" = "com.sap.db.jdbc.Driver"
)
```

> **remarks**
>
> For more information about the JDBC URL formats and parameters supported by the SAP HANA JDBC driver, see SAP HANA.

Hierarchical mapping

When mapping SAP HANA, Doris' Database corresponds to a Schema under the specified DataBase ("DATABASE" in the `jdbc_url` parameter) in SAP HANA. The Table under Doris' Database corresponds to the Tables under Schema in SAP HANA. That is, the mapping relationship is as follows:

Doris	SAP HANA
Catalog	Database
Database	Schema
Table	Table

Type mapping

SAP HANA to Doris type mapping

SAP HANA Type	Doris Type	Comment
BOOLEAN	BOOLEAN	
TINYINT	TINYINT	
SMALLINT	SMALLINT	
INTERGER	INT	
BIGINT	BIGINT	
SMALLDECIMAL	DECIMAL	
DECIMAL	DECIMAL/STRING	Which type will be selected based on the (precision, scale) of the Doris DECIMAL field
REAL	FLOAT	
DOUBLE	DOUBLE	
DATE	DATE	
TIME	STRING	
TIMESTAMP	DATETIME	
SECONDDATE	DATETIME	
VARCHAR	STRING	
NVARCHAR	STRING	
ALPHANUM	STRING	
SHORTTEXT	STRING	
CHAR	CHAR	
NCHAR	CHAR	

Query optimization

Predicate pushdown

When executing a query like `where dt = '2022-01-01'`, Doris can push these filtering conditions down to the external data source, thereby directly excluding data that does not meet the conditions at the data source level, reducing inaccuracies. Necessary data acquisition and transmission. This greatly improves query performance while also reducing the load on external data sources.

Row limit

If you have the limit keyword in the query, Doris will push the limit down to the SAP HANA database to reduce the amount of data transfer.

Escape characters

Doris will automatically add the escape character (“) to the field names and table names in the query statements sent to SAP HANA to avoid conflicts between the field names and table names and SAP HANA internal keywords.

OceanBase

Doris JDBC Catalog supports connecting to the OceanBase database through the standard JDBC interface. This document describes how to configure OceanBase database connection.

Terms and Conditions

To connect to the OceanBase database you need

- OceanBase 3.1.0 or higher
- JDBC driver for OceanBase database, you can download the latest or specified version of OceanBase JDBC driver from Maven repository. **It is recommended to use OceanBase Connector/J 2.4.8 or above.**
- Doris Network connection between each FE and BE node and OceanBase server.

Connect to OceanBase

```
CREATE CATALOG oceanbase PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="password",
    "jdbc_url" = "jdbc:oceanbase://host:port/db",
    "driver_url" = "oceanbase-client-2.4.8.jar",
    "driver_class" = "com.oceanbase.jdbc.Driver"
)
```

> **remarks**
>
> `jdbc_url` defines the connection information and parameters to be passed to the OceanBase JDBC driver. Parameters for supported URLs can be found in OceanBase JDBC Driver Configuration.

Mode compatible

Doris will automatically recognize that OceanBase is in MySQL or Oracle mode when creating the OceanBase Catalog so that metadata can be parsed correctly.

Hierarchical mapping, type mapping, and query optimization in different modes are the same as the Catalog processing methods of MySQL or Oracle databases. Please refer to the documentation.

- MySQL Catalog
- Oracle Catalog

Elasticsearch

Elasticsearch

Elasticsearch (ES) Catalogs in Doris support auto-mapping of ES metadata. Users can utilize the full-text search capability of ES in combination of the distributed query planning capability of Doris to provide a full-fledged OLAP solution that is able to perform:

1. Multi-index distributed Join queries in ES;
2. Join queries across Doris and ES as well as full-text search and filter.

Usage

1. Doris supports Elasticsearch 5.x and newer versions.

Create Catalog

```
CREATE CATALOG es PROPERTIES (
    "type"="es",
    "hosts"="http://127.0.0.1:9200"
);
```

Since there is no concept of “database” in ES, after connecting to ES, Doris will automatically generate a unique database: `default_db`.

After switching to the ES Catalog, you will be in the `dafault_db` so you don’t need to execute the `USE default_db` command.

Parameter Description

Parameter	Required or Not	Default Value	Description
`hosts`	Yes		ES address, can be one or multiple addresses, or the load balancer address of ES
`user`	No	Empty	ES username
`password`	No	Empty	Password of the corresponding user
`doc_value_scan`	No	true	Whether to obtain value of the target field by ES/Lucene columnar storage
`keyword_sniff`	No	true	Whether to sniff the text.fields in ES based on keyword; If this is set to false, the system will perform matching after tokenization.
`nodes_discovery`	No	true	Whether to enable ES node discovery, set to true by default; set to false in network isolation environments and only connected to specified nodes
`ssl`	No	false	Whether to enable HTTPS access mode for ES, currently follows a “Trust All” method in FE/BE

Parameter	Required or Not	Default Value	Description
`mapping_es_id`	No	false	Whether to map the `_id` field in the ES index
`like_push_down`	No	true	Whether to transform like to wildcard push down to es, this increases the cpu consumption of the es.
`include_hidden_index`	No	false	Whether to include hidden index, default to false.

1. In terms of authentication, only HTTP Basic authentication is supported and it requires the user to have read privilege for the index and paths including `/_cluster/state/` and `_nodes/http` ; if you have not enabled security authentication for the cluster, you don't need to set the `user` and `password`.
2. If there are multiple types in the index in 5.x and 6.x, the first type is taken by default.

Column Type Mapping

ES Type	Doris Type	Comment
null	null	
boolean	boolean	
byte	tinyint	
short	smallint	
integer	int	
long	bigint	
unsigned_long	largeint	
float	float	
half_float	float	
double	double	
scaled_float	double	
date	date	Only support default/yyyy-MM-dd HH:mm:ss/yyyy-MM-dd/epoch_millis format
keyword	string	
text	string	
ip	string	
constant_keyword	string	
wildcard	string	
nested	json	
object	json	

ES Type	Doris Type	Comment
other	unsupported	

Array Type

Elasticsearch does not have an explicit array type, but one of its fields can contain 0 or more values. To indicate that a field is an array type, a specific `doris` structural annotation can be added to the _meta section of the index mapping. For Elasticsearch 6.x and before release, please refer _meta.

For example, suppose there is an index `doc` containing the following data structure.

```
{
  "array_int_field": [1, 2, 3, 4],
  "array_string_field": ["doris", "is", "the", "best"],
  "id_field": "id-xxx-xxx",
  "timestamp_field": "2022-11-12T12:08:56Z",
  "array_object_field": [
    {
      "name": "xxx",
      "age": 18
    }
  ]
}
```

The array fields of this structure can be defined by using the following command to add the field property definition to the `_meta.doris` property of the target index mapping.

```
# ES 7.x and above
curl -X PUT "localhost:9200/doc/_mapping?pretty" -H 'Content-Type:application/json' -d '
{
    "_meta": {
        "doris":{
            "array_fields":[
                "array_int_field",
                "array_string_field",
                "array_object_field"
            ]
        }
    }
}'

# ES 6.x and before
curl -X PUT "localhost:9200/doc/_mapping/_doc?pretty" -H 'Content-Type: application/json' -d '
{
    "_meta": {
        "doris":{
            "array_fields":[
                "array_int_field",
                "array_string_field",
                "array_object_field"
            ]
        }
    }
}
'
```

`array_fields`：Used to indicate a field that is an array type.

Best Practice

Predicate Pushdown

ES Catalogs support predicate pushdown to ES, which means only the filtered data will be returned. This can markedly improve query performance and reduce usage of CPU, memory, and IO in both Doris and ES.

For the sake of optimization, operators will be converted into the following ES queries:

SQL syntax	ES 5.x+ syntax
=	term query
in	terms query
> , < , >= , ⇐	range query
and	bool.filter
or	bool.should
not	bool.must_not
not in	bool.must_not + terms query
is_not_null	exists query
is_null	bool.must_not + exists query
esquery	ES-native JSON QueryDSL

Columnar Scan for Faster Queries (enable_docvalue_scan=true)

Set `"enable_docvalue_scan" = "true"`.

After this, when obtaining data from ES, Doris will follow these rules:

- **Try and see**: Doris will automatically check if columnar storage is enabled for the target fields (doc_value: true), if it is, Doris will obtain all values in the fields from the columnar storage.
- **Auto-downgrading**: If any one of the target fields is not available in columnar storage, Doris will parse and obtain all target data from row storage (`_source`).

Benefits

By default, Doris On ES obtains all target columns from `_source`, which is in row storage and JSON format. Compared to columnar storage, `_source` is slow in batch read. In particular, when the system only needs to read small number of columns, the performance of `docvalue` can be about a dozen times faster than that of `_source`.

Note

1. Columnar storage is not available for `text` fields in ES. Thus, if you need to obtain fields containing `text` values, you will need to obtain them from `_source`.
2. When obtaining large numbers of fields (`>= 25`), the performances of `docvalue` and `_source` are basically equivalent.
3. The `keyword` type field, due to the `ignore_above` parameter's limit, long text fields exceeding this limit will be ignored, so the result may be empty. In this case, you need to turn off `enable_docvalue_scan` and get

the result from `_source`.

Sniff Keyword Fields

Set `"enable_keyword_sniff" = "true"`.

ES allows direct data ingestion without an index since it will automatically create an index after ingestion. For string fields, ES will create a field with both `text` and `keyword` types. This is how the Multi-Field feature of ES works. The mapping is as follows:

```
"k4": {
   "type": "text",
   "fields": {
      "keyword": {
         "type": "keyword",
         "ignore_above": 256
      }
   }
}
```

For example, to conduct "=" filtering on `k4`, Doris on ES will convert the filtering operation into an ES TermQuery.

The original SQL filter:

```
k4 = "Doris On ES"
```

The converted ES query DSL:

```
"term" : {
    "k4": "Doris On ES"

}
```

Since the first field of `k4` is `text`, it will be tokenized by the analyzer set for `k4` (or by the standard analyzer if no analyzer has been set for `k4`) after data ingestion. As a result, it will be tokenized into three terms: "Doris", "on", and "ES".

The details are as follows:

```
POST /_analyze
{
  "analyzer": "standard",
  "text": "Doris On ES"
}
```

The tokenization results:

```
{
   "tokens": [
      {
         "token": "doris",
         "start_offset": 0,
```

```
         "end_offset": 5,
         "type": "<ALPHANUM>",
         "position": 0
      },
      {
         "token": "on",
         "start_offset": 6,
         "end_offset": 8,
         "type": "<ALPHANUM>",
         "position": 1
      },
      {
         "token": "es",
         "start_offset": 9,
         "end_offset": 11,
         "type": "<ALPHANUM>",
         "position": 2
      }
   ]
}
```

If you conduct a query as follows:

```
"term" : {
    "k4": "Doris On ES"
}
```

Since there is no term in the dictionary that matches the term `Doris On ES`, no result will be returned.

However, if you have set `enable_keyword_sniff: true`, the system will convert `k4 = "Doris On ES"` to `k4.keyword = "Doris On ES"` to match the SQL semantics. The converted ES query DSL will be:

```
"term" : {
    "k4.keyword": "Doris On ES"
}
```

`k4.keyword` is of `keyword` type, so the data is written in ES as a complete term, allowing for successful matching.

Auto Node Discovery, Set to True by Default (nodes_discovery=true)

Set `"nodes_discovery" = "true"`.

Then, Doris will discover all available data nodes (the allocated shards) in ES. If Doris BE hasn't accessed the ES data node addresses, then set `"nodes_discovery" = "false"` . ES clusters are deployed in private networks that are isolated from public Internet, so users will need proxy access.

HTTPS Access Mode for ES Clusters

Set `"ssl" = "true"`.

A temporary solution is to implement a "Trust All" method in FE/BE. In the future, the real user configuration certificates will be used.

Query Usage

You can use the ES external tables in Doris the same way as using Doris internal tables, except that the Doris data models (Rollup, Pre-Aggregation, and Materialized Views) are unavailable.

Basic Query

```
select * from es_table where k1 > 1000 and k3 ='term' or k4 like 'fu*z_'
```

Extended esquery(field, QueryDSL)

The `esquery(field, QueryDSL)` function can be used to push queries that cannot be expressed in SQL, such as `match_phrase` and `geoshape` , to ES for filtering.

In `esquery`, the first parameter (the column name) is used to associate with `index`, while the second parameter is the JSON expression of basic `Query DSL` in ES, which is surrounded by `{}`. The `root key` in JSON is unique, which can be `match_phrase`, `geo_shape` or `bool` , etc.

A `match_phrase` query:

```
select * from es_table where esquery(k4, '{
        "match_phrase": {
           "k4": "doris on es"
        }
    }');
```

A `geo` query:

```
select * from es_table where esquery(k4, '{
      "geo_shape": {
         "location": {
            "shape": {
               "type": "envelope",
               "coordinates": [
                  [
                     13,
                     53
                  ],
                  [
                     14,
                     52
                  ]
               ]
            },
            "relation": "within"
         }
      }
   }');
```

A `bool` query:

```
select * from es_table where esquery(k4, ' {
        "bool": {
```

```
            "must": [
               {
                  "terms": {
                     "k1": [
                        11,
                        12
                     ]
                  }
               },
               {
                  "terms": {
                     "k2": [
                        100
                     ]
                  }
               }
            ]
         }
      }');
```

Suggestions for Time Fields

> These are only applicable for ES external tables. Time fields will be automatically mapped to Date or Datetime type in ES Catalogs.

ES boasts flexible usage of time fields, but in ES external tables, improper type setting of time fields will result in predicate pushdown failures.

It is recommended to allow the highest level of format compatibility for time fields when creating an index:

```
"dt": {
    "type": "date",
    "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
}
```

When creating this field in Doris, it is recommended to set its type to `date` or `datetime` (or `varchar`) . You can use the following SQL statements to push the filters down to ES.

```
select * from doe where k2 > '2020-06-21';

select * from doe where k2 < '2020-06-21 12:00:00';

select * from doe where k2 < 1593497011;

select * from doe where k2 < now();

select * from doe where k2 < date_format(now(), '%Y-%m-%d');
```

Note:

- The default format of time fields in ES is:

```
strict_date_optional_time||epoch_millis
```

- Timestamps ingested into ES need to be converted into `ms`, which is the internal processing format in ES; otherwise errors will occur in ES external tables.

Obtain ES Metadata Field `_id`

Each ingested files, if not specified with an `_id` , will be given a globally unique `_id`, which is the primary key. Users can assign an `_id` with unique business meanings to the files during ingestion.

To obtain such field values from ES external tables, you can add an `_id` field of `varchar` type when creating tables.

```
CREATE EXTERNAL TABLE `doe` (
  `_id` varchar COMMENT "",
  `city`  varchar COMMENT ""
) ENGINE=ELASTICSEARCH
PROPERTIES (
"hosts" = "http://127.0.0.1:8200",
"user" = "root",
"password" = "root",
"index" = "doe"
}
```

To obtain such field values from ES Catalogs, please set `"mapping_es_id" = "true"`.

Note:

1. The `_id` field only supports `=` and `in` filtering.
2. The`_id` field must be of `varchar` type.

Getting globally ordered query results

ES query results sorted by scores are useful in scenarios such as relevance sorting, prioritizing important content, etc. Doris querying ES pulls data according to the distribution of shards in the ES index in order to take full advantage of the MPP architecture.
In order to get globally ordered sorting results, you need to do a single point query on ES. This can be controlled by the session variable `enable_es_parallel_scroll` (default true).
When `enable_es_parallel_scroll=false` is set, Doris will send a `scroll` query without `shard_preference` and `sort` information to the ES cluster to get globally ordered results.
Note: Use with caution when the query result set is large.

Modify the batch size for scroll requests.

The `batch` of a `scroll` request is 4064 by default, and can be changed with the session variable `batch_size`.

FAQ

1. Are X-Pack authenticated ES clusters supported?

   All ES clusters with HTTP Basic authentications are supported.

2. Why are some queries require longer response time than those in ES?

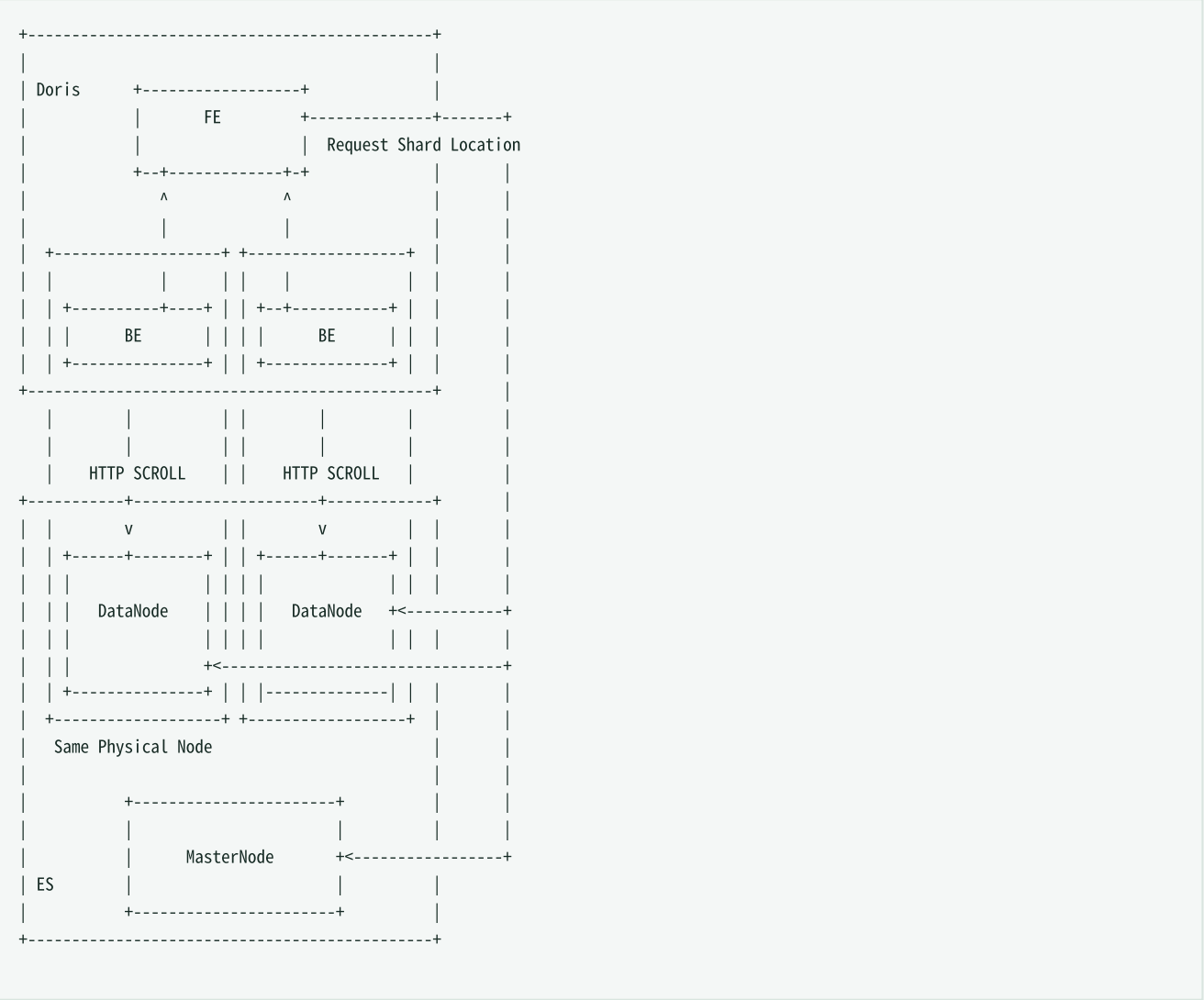
For `_count` queries, ES can directly read the metadata regarding the number of the specified files instead of filtering the original data. This is a huge time saver.

3. Can aggregation operations be pushed down?

   Currently, Doris On ES does not support pushdown for aggregations such as sum, avg, and min/max. In such operations, Doris obtains all files that met the specified conditions from ES and then conducts computing internally.

Appendix

How Doris Conducts Queries in ES



1. Doris FE sends a request to the specified host for table creation in order to obtain information about the HTTP port and the index shard allocation.

2. Based on the information about node and index metadata from FE, Doris generates a query plan and send it to the corresponding BE node.

3. Following the principle of proximity, the BE node sends request to the locally deployed ES node, and obtain data from `_source` or `docvalue` from each shard of ES index concurrently by way of `HTTP Scroll`.

4. Doris returns the computing results to the user.

MaxCompute

MaxCompute (previously known as ODPS) is a data warehousing solution that can process terabytes or petabytes of data.

> What is MaxCompute

Connect to MaxCompute

```
CREATE CATALOG mc PROPERTIES (
  "type" = "max_compute",
  "mc.region" = "cn-beijing",
  "mc.default.project" = "your-project",
  "mc.access_key" = "ak",
  "mc.secret_key" = "sk"
);
```

- `mc.region`: MaxCompute Region. Can Get the Region From Endpoints.
- `mc.default.project`: MaxCompute Project. See Your MaxCompute Projects.
- `mc.access_key`: AccessKey, which you can create and manage on the Alibaba Cloud console.
- `mc.secret_key`: SecretKey, which you can create and manage on the Alibaba Cloud console.
- `mc.public_access`: You can enable public network access for test, when set `"mc.public_access"="true"`.

Quotas

Pay-as-you-go quota has limited concurrency and usage. For additional resources, please refer to the documentation. See Manage quotas.

Column type mapping

Consistent with Hive Catalog, please refer to the **column type mapping** section in Hive Catalog.

User-defined service address

The region property is specified to generate a default endpoint of public network.

In addition to default endpoint addresses, MaxCompute Catalog also supports custom service addresses in properties.

Use the following properties: * `mc.odps_endpoint`： MaxCompute Endpoint。 * `mc.tunnel_endpoint`: Tunnel Endpoint，MaxCompute Catalog uses the Tunnel SDK to obtain data.

For more information about MaxCompute Endpoint and Tunnel Endpoint that are used in different regions and network connection modes, see Endpoint

For example:

```
CREATE CATALOG mc PROPERTIES (
  "type" = "max_compute",
  "mc.region" = "cn-beijing",
  "mc.default.project" = "your-project",
  "mc.access_key" = "ak",
  "mc.secret_key" = "sk"
  "mc.odps_endpoint" = "http://service.cn-beijing.maxcompute.aliyun-inc.com/api",
  "mc.tunnel_endpoint" = "http://dt.cn-beijing.maxcompute.aliyun-inc.com"
);
```

Querying Files on S3/HDFS

With the Table Value Function feature, Doris is able to query files in object storage or HDFS as simply as querying Tables. In addition, it supports automatic column type inference.

Usage

For more usage details, please see the documentation:

- S3: supports file analysis on object storage compatible with S3
- HDFS: supports file analysis on HDFS
- LOCAL: supports file analysis on local file system

The followings illustrate how file analysis is conducted with the example of S3 Table Value Function.

Automatic Column Type Inference

```
> DESC FUNCTION s3 (
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "parquet",
    "use_path_style"="true"
);
+---------------+--------------+------+-------+---------+-------+
| Field         | Type         | Null | Key   | Default | Extra |
+---------------+--------------+------+-------+---------+-------+
| p_partkey     | INT          | Yes  | false | NULL    | NONE  |
| p_name        | TEXT         | Yes  | false | NULL    | NONE  |
| p_mfgr        | TEXT         | Yes  | false | NULL    | NONE  |
| p_brand       | TEXT         | Yes  | false | NULL    | NONE  |
| p_type        | TEXT         | Yes  | false | NULL    | NONE  |
| p_size        | INT          | Yes  | false | NULL    | NONE  |
| p_container   | TEXT         | Yes  | false | NULL    | NONE  |
| p_retailprice | DECIMAL(9,0) | Yes  | false | NULL    | NONE  |
| p_comment     | TEXT         | Yes  | false | NULL    | NONE  |
+---------------+--------------+------+-------+---------+-------+
```

An S3 Table Value Function is defined as follows:

```
s3(
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "Format" = "parquet",
    "use_path_style"="true")
```

It specifies the file path, connection, and authentication.

After defining, you can view the schema of this file using the `DESC FUNCTION` statement.

As can be seen, Doris is able to automatically infer column types based on the metadata of the Parquet file.

Besides Parquet, Doris supports analysis and auto column type inference of ORC, CSV, and Json files.

CSV Schema

By default, for CSV format files, all columns are of type String. Column names and column types can be specified individually via the `csv_schema` attribute. Doris will use the specified column type for file reading. The format is as follows:

`name1:type1;name2:type2;...`

For columns with mismatched formats (such as string in the file and int defined by the user), or missing columns (such as 4 columns in the file and 5 columns defined by the user), these columns will return null.

Currently supported column types are:

name	mapping type
tinyint	tinyint
smallint	smallint
int	int
bigint	bigint
largeint	largeint
float	float
double	double
decimal(p,s)	decimalv3(p,s)
date	datev2
datetime	datetimev2
char	string
varchar	string
string	string
boolean	boolean

Example:

```
s3 (
    "uri" = "https://bucket1/inventory.dat",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "csv",
    "column_separator" = "|",
    "csv_schema" = "k1:int;k2:int;k3:int;k4:decimal(38,10)",
    "use_path_style"="true"
)
```

Query and Analysis

You can conduct queries and analysis on this Parquet file using any SQL statements:

```
SELECT * FROM s3(
    "uri" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "parquet",
    "use_path_style"="true")
LIMIT 5;
+-----------+-----------------------------------------+----------------+----------+-------------------------+--------+-------------
+---------------+---------------------+
| p_partkey | p_name                                  | p_mfgr         | p_brand  | p_type                  | p_size | p_container
| p_retailprice | p_comment           |
+-----------+-----------------------------------------+----------------+----------+-------------------------+--------+-------------
+---------------+---------------------+
|         1 | goldenrod lavender spring chocolate lace | Manufacturer#1 | Brand#13 | PROMO BURNISHED COPPER  |      7 | JUMBO PKG
|           901 | ly. slyly ironi     |
|         2 | blush thistle blue yellow saddle         | Manufacturer#1 | Brand#13 | LARGE BRUSHED BRASS     |      1 | LG CASE
|           902 | lar accounts amo    |
|         3 | spring green yellow purple cornsilk      | Manufacturer#4 | Brand#42 | STANDARD POLISHED BRASS |     21 | WRAP CASE
|           903 | egular deposits hag |
|         4 | cornflower chocolate smoke green pink    | Manufacturer#3 | Brand#34 | SMALL PLATED BRASS      |     14 | MED DRUM
|           904 | p furiously r       |
|         5 | forest brown coral puff cream            | Manufacturer#3 | Brand#32 | STANDARD POLISHED TIN   |     15 | SM PKG
|           905 |  wake carefully     |
+-----------+-----------------------------------------+----------------+----------+-------------------------+--------+-------------
+---------------+---------------------+
```

You can put the Table Value Function anywhere that you used to put Table in the SQL, such as in the WITH or FROM clause in CTE. In this way, you can treat the file as a normal table and conduct analysis conveniently.

You can also create a logic view by using `CREATE VIEW` statement for a Table Value Function. So that you can query this view, grant priv on this view or allow other user to access this Table Value Function.

```
CREATE VIEW v1 AS
SELECT * FROM s3(
    "uri" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "parquet",
    "use_path_style"="true");

DESC v1;

SELECT * FROM v1;

GRANT SELECT_PRIV ON db1.v1 TO user1;
```

Data Ingestion

Users can ingest files into Doris tables via `INSERT INTO SELECT` for faster file analysis:

```
// 1. Create Doris internal table
CREATE TABLE IF NOT EXISTS test_table
(
    id int,
```

```
    name varchar(50),
    age int
)
DISTRIBUTED BY HASH(id) BUCKETS 4
PROPERTIES("replication_num" = "1");

// 2. Insert data using S3 Table Value Function
INSERT INTO test_table (id,name,age)
SELECT cast(id as INT) as id, name, cast (age as INT) as age
FROM s3(
    "uri" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "parquet",
    "use_path_style" = "true");
```

Note

1. If the URI specified by the `S3 / HDFS` TVF is not matched with the file, or all the matched files are empty files, then the`S3 / HDFS` TVF will return to the empty result set. In this case, using the `DESC FUNCTION` to view the schema of this file, you will get a dummy column`__dummy_col`, which can be ignored.
2. If the format of the TVF is specified to `CSV`, and the read file is not a empty file but the first line of this file is empty, then it will prompt the error `The first line is empty, can not parse column numbers`. This is because the schema cannot be parsed from the first line of the file

File Caches

File Cache

File Cache accelerates queries that read the same data by caching the data files of recently accessed from remote storage system (HDFS or Object Storage). In Ad Hoc scenarios where the same data is frequently accessed, File Cache can avoid repeated remote data access costs and improve the query analysis performance and stability of hot data.

How it works

File Cache caches the accessed remote data in the local BE node. The original data file will be divided into blocks according to the read IO size, and the block will be stored in file `cache_path/hash(filepath).substr(0, 3)/hash(filepath)/offset`, and save the block meta information in the BE node. When accessing the same remote file, doris will check whether the cached data of the file exists in the local cache, and according to the offset and size of the block, confirm which data is read from the local block, which data is pulled from the remote, and cache the new data pulled from the remote. When the BE node restarts, scan `cache_path` directory, recover the meta information of the block. When the cache size reaches the upper threshold, the blocks that have not been accessed for a long time shall be cleaned according to the LRU principle.

Usage

File Cache is disabled by default. You need to set the relevant configuration in FE and BE to enable it.

Configurations for FE

Enable File Cache for a given session:

```
SET enable_file_cache = true;
```

Enable File Cache globally:

```
SET GLOBAL enable_file_cache = true;
```

> The File Cache is only applicable to external queries for files (such as Hive, Hudi). It has no effect on internal table queries, or non-file external queries (such as JDBC, Elasticsearch), etc.

Configurations for BE

Add settings to the BE node's configuration file `conf/be.conf`, and restart the BE node for the configuration to take effect.

Parameter	Description
enable_file_cache	Whether to enable File Cache, default false
file_cache_path	Parameters about cache path, json format, for example: [{"path": "/path/to/file_cache1", "total_size":53687091200,"query_limit": "10737418240"},{"path": "/path/to/file_cache2", "total_size":53687091200,"query_limit": "10737418240"},{"path": "/path/to/file_cache3", "total_size":53687091200,"query_limit": "10737418240"}]. path is the path to save cached data; total_size is the max size of cached data; query_limit is the max size of cached data for a single query.
file_cache_min_file_segment_size	Min size of a single cached block, default 1MB, should greater than 4096
file_cache_max_file_segment_size	Max size of a single cached block, default 4MB, should greater than 4096
enable_file_cache_query_limit	Whether to limit the cache size used by a single query, default false
clear_file_cache	Whether to delete the previous cache data when the BE restarts, default false

Check whether a query hits cache

Execute set enable_profile = true to enable the session variable, and you can view the query profile in the Queris tab of FE's web page. The metrics related to File Cache are as follows:

```
-  FileCache:
  -  IOHitCacheNum:  552
  -  IOTotalNum:  835
  -  ReadFromFileCacheBytes:  19.98  MB
  -  ReadFromWriteCacheBytes:  0.00
  -  ReadTotalBytes:  29.52  MB
  -  WriteInFileCacheBytes:  915.77  MB
  -  WriteInFileCacheNum:  283
```

- IOTotalNum: Number of remote access
- IOHitCacheNum: Number of cache hits
- ReadFromFileCacheBytes: Amount of data read from cache file
- ReadTotalBytes: Total amount of data read
- SkipCacheBytes: Failed to create the cache file, or the cache file was deleted. The amount of data that needs to be read from the remote again
- WriteInFileCacheBytes: Amount of data saved to cache file
- WriteInFileCacheNum: The number of blocks saved, so 'WriteInFileCacheBytes'/' WriteInFileCacheBytes' is the average size of blocks

IOHitCacheNum / IOTotalNum Equal to 1, indicating that read data only from file cache

ReadFromFileCacheBytes / ReadTotalBytes Equal to 1, indicating that read data only from file cache

ReadFromFileCacheBytes The smaller the better, the smaller the amount of data read from remote

Elastic Compute Node

Since version 1.2.1, Doris has supported the Compute Node feature.

Starting from this version, BE nodes can be divided into two categories:

- Mix

  A mixed node, which is the default type of BE node. This type of node can both participate in computation and be responsible for storing Doris data.

- Computation

  A computation node that is not responsible for data storage, only for data computation.

As a special type of BE node, the computation node does not have data storage capabilities and is only responsible for data computation. Therefore, the computation node can be seen as a stateless BE node, making it easy to add and remove nodes.

In the Lakehouse solution, the computation node can serve as an elastic node for querying external data sources such as Hive, Iceberg, JDBC, etc. Doris does not store data from external data sources, so the computation node can easily expand the computational capabilities for external data sources. Additionally, the computation node can be configured with a cache directory to cache hot data from external data sources, further accelerating data retrieval.

> **The computation node is suitable for the integrated storage and computation deployment mode in Doris, providing elastic resource control. In the storage-computation separation architecture of Doris 3.0, all BE nodes are stateless, eliminating the need for separate computation nodes.**

Using Compute Nodes

Adding Compute Nodes

Add the following configuration to the `be.conf` file of the BE:

```
be_node_role=computation
```

After that, start the BE node, and it will run as a computation node type.

You can then connect to Doris using a MySQL client and execute:

```
ALTER SYSTEM ADD BACKEND
```

This will add the BE node. After successful addition, in the `NodeRole` column of `SHOW BACKENDS`, you can see the node type as `computation`.

Using Compute Nodes

To use compute nodes, the following conditions must be met:

- The cluster contains compute nodes.
- The `fe.conf` file has the configuration item: `prefer_compute_node_for_external_table = true`

Additionally, the following FE configuration items will affect the usage strategy of compute nodes:

- `min_backend_num_for_external_table`

  Before Doris 2.0 (inclusive), the default value of this parameter was 3. After version 2.1, the default parameter is -1.

  This parameter indicates the minimum number of BE nodes expected to participate in external table data queries. `-1` indicates that this value is equivalent to the current number of compute nodes in the cluster.

  For example, suppose there are 3 compute nodes and 5 mixed nodes in the cluster.

  If `min_backend_num_for_external_table` is set to less than or equal to 3, external table queries will only use 3 compute nodes. If set to greater than 3, for example, 6, external table queries will use 3 compute nodes and additionally select 3 mixed nodes for computation.

  In summary, this parameter is mainly used for the minimum number of BE nodes that can participate in external table calculations, and it will prioritize selecting compute nodes.

> Note:
>
> 1. After version 2.1, `min_backend_num_for_external_table` can be set to `-1`. In previous versions, this parameter must be a positive number. And this parameter only takes effect when `prefer_compute_node_for_external_table = true`.
> 2. If `prefer_compute_node_for_external_table` is `false`, external table queries will select any BE node.
> 3. If there are no compute nodes in the cluster, the above parameters will not take effect.
> 4. If the value of `min_backend_num_for_external_table` is greater than the total number of BE nodes, at most all BE nodes will be selected.
> 5. The above parameters can be dynamically modified using the `ADMIN SET FRONTEND CONFIG` command without the need to restart FE nodes. All FE nodes need to be configured. Alternatively, add the configuration in `fe.conf` and restart the FE nodes.

Best Practices

Load Isolation and Elastic Scaling for Federated Queries

In federated query scenarios, users can deploy a dedicated set of compute nodes for querying external table data. This can isolate the query load of external tables (such as large-scale analysis on Hive) from the query load of internal tables (such as low-latency fast data analysis).

At the same time, compute nodes, as stateless BE nodes, can easily scale up and down. For example, you can deploy a cluster of elastic compute nodes using k8s, utilize more compute nodes for data lake analysis during business peak hours, and quickly scale down during off-peak hours to reduce costs.

Frequently Asked Questions

1. Can mixed nodes and compute nodes be converted to each other

   Compute nodes can be converted to mixed nodes. However, mixed nodes cannot be converted to compute nodes.

   - Convert compute nodes to mixed nodes

     1. Stop the BE nodes.
     2. Remove the `be_node_role` configuration in `be.conf` or configure it as `be_node_role=mix`.
     3. Configure the correct `storage_root_path` data storage directory.
     4. Start the BE nodes.

   - Convert mixed nodes to compute nodes

     In principle, this operation is not supported because mixed nodes themselves store data. If conversion is needed, first perform node safe decommissioning, then set it as a compute node in the manner of a new node.

2. Do compute nodes need to configure a data storage directory

   Yes. The data storage directory of compute nodes will not store user data, only some BE node's own information files like `cluster_id`, and some temporary files during operation.

   The storage directory of compute nodes only requires a small amount of disk space (in the MB range) and can be destroyed with the node at any time without affecting user data.

3. Can compute nodes and mixed nodes configure a file cache directory

   File cache caches data files from remote storage systems (HDFS or object storage) that have been recently accessed, speeding up subsequent queries for the same data.

   Both compute nodes and mixed nodes can set a file cache directory. The file cache directory needs to be created in advance.

   Additionally, Doris also employs strategies like consistent hashing to minimize the probability of cache invalidation during node scaling operations.

4. Do compute nodes need to be taken offline through DECOMMISION operation

   No. Compute nodes can be directly removed using the `DROP BACKEND` operation.

External Table Statistics

External Table Statistics

The collection method and content of the external table statistics are basically the same as those of the internal table. For more information, please refer to Internal table statistics. Currently supports the collection of Hive, Iceberg and Hudi external tables.

Features that are not currently supported by the external table include

1. Histogram collection is not supported yet
2. Does not support incremental collection and update of partitions
3. Automatic collection (with auto) is not supported for now, users can use periodic collection (with period) instead
4. Sampling collection is not currently supported

The following mainly introduces the examples and implementation principles of the collection of external table statistic information.

Examples of usage.

Here is an example of collecting external table statistics by executing the analyze command in Doris. Except for the 4 functions that are not supported mentioned above, the rest are the same as the internal table. The following takes the hive.tpch100 database as an example to show how to use it. The tpch100 database contains 8 tables including lineitem, orders, region, etc.

Collection of statistics

We supports two collection methods for external table: manual once collection and periodic collection.

Manual once collection

- Collect the row count of `lineitem` table and the statistics of all columns

```
mysql> ANALYZE TABLE hive.tpch100.lineitem;
+-------------+-----------------------+------------+-------------------
```

This operation is performed asynchronously, a collection job will be created in the background, and the progress of the job can be viewed using job_id

```
mysql> SHOW ANALYZE 16990;
+--------+-------------+------------------------+----------+-----------------------------
```

```
| 17027  |
| hive         | tpch100 | region     | [r_regionkey,r_comment,r_name]
| 17033  |
| hive         | tpch100 | partsupp   | [ps_suppkey,ps_availqty,ps_comment,ps_partkey,ps_supplycost]
| 17038  |
| hive         | tpch100 | orders     |
[o_orderstatus,o_clerk,o_orderdate,o_shippriority,o_custkey,o_totalprice,o_orderkey,o_comment,o_orderpriority]
| 17045  |
| hive         | tpch100 | lineitem   |
[l_returnflag,l_receiptdate,l_tax,l_shipmode,l_suppkey,l_shipdate,l_commitdate,l_partkey,l_orderkey,l_quantity,l_linestatus,l_commen
t,l_extendedprice,l_linenumber,l_discount,l_shipinstruct] | 17056  |
| hive         | tpch100 | part       | [p_partkey,p_container,p_name,p_comment,p_brand,p_type,p_retailprice,p_mfgr,p_size]
| 17074  |
| hive         | tpch100 | customer   | [c_custkey,c_phone,c_acctbal,c_mktsegment,c_address,c_nationkey,c_name,c_comment]
| 17085  |
+--------------+---------+------------+-----------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------+--------+
8 rows in set (0.29 sec)
```

This operation will submit the collection jobs of all tables under the tpch100 database in batches, which is also executed asynchronously. A job_id will be created for each table, and the job and task progress of each table can also be viewed through the job_id.

- synchronous collection

You can use `with sync` to collect table or database statistics synchronously. At this time, no background task will be created, and the client will block before the collection is completed, and return until the collection job is completed.

```
mysql> analyze table hive.tpch100.orders with sync;
Query OK, 0 rows affected (33.19 sec)
```

It should be noted that synchronous collection is affected by the query_timeout session variable. If the collection failed because of timeout, you need to increase the variable value and try again. for example `set query_timeout=3600` (This set timeout interval to 1 hour)

periodic collection

Use `with period` to create periodic execution collection job.

```
analyze table hive.tpch100.orders with period 86400;
```

This statement creates a periodic collection job, the cycle is 1 day, and the statistical information of the `orders` table is automatically collected and updated every 24 hours.

Job management

The method of job management is also the same as that of the internal table, including functions such as viewing jobs, viewing tasks, and deleting jobs. Please refer to the manage job section of Internal table statistics

- Show all job status

```
mysql> SHOW ANALYZE;
+--------+-------------+-------------------------+----------+-----------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------------------------------------------------+----------
+---------------+---------+----------------------+----------+-----------------------------------------+---------------+
| job_id | catalog_name | db_name                | tbl_name | col_name
| job_type | analysis_type | message | last_exec_time_in_ms | state    | progress                                | schedule_type
|
+--------+--------------+------------------------+----------+-----------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------+----------
+---------------+---------+----------------------+----------+-----------------------------------------+---------------+
| 16990  | hive         | default_cluster:tpch100 | lineitem |
[l_returnflag,l_receiptdate,l_tax,l_shipmode,l_suppkey,l_shipdate,l_commitdate,l_partkey,l_orderkey,l_quantity,l_linestatus,l_commen
t,l_extendedprice,l_linenumber,l_discount,l_shipinstruct] | MANUAL   | FUNDAMENTALS  |         | 2023-07-27 16:05:02  | FINISHED |
17 Finished/0 Failed/0 In Progress/17 Total | ONCE          |
+--------+--------------+------------------------+----------+-----------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------+----------
+---------------+---------+----------------------+----------+-----------------------------------------+---------------+
```

- Show all tasks status of a job

```
mysql> SHOW ANALYZE TASK STATUS 16990;
+---------+-----------------+---------+-----------------------+-----------------+----------+
| task_id | col_name        | message | last_state_change_time | time_cost_in_ms | state    |
+---------+-----------------+---------+-----------------------+-----------------+----------+
| 16991   | l_receiptdate   |         | 2023-07-27 16:05:02    | 9560            | FINISHED |
| 16992   | l_returnflag    |         | 2023-07-27 16:01:44    | 14394           | FINISHED |
| 16993   | l_tax           |         | 2023-07-27 16:01:52    | 7975            | FINISHED |
| 16994   | l_shipmode      |         | 2023-07-27 16:02:11    | 18961           | FINISHED |
| 16995   | l_suppkey       |         | 2023-07-27 16:02:17    | 6684            | FINISHED |
| 16996   | l_shipdate      |         | 2023-07-27 16:02:26    | 8518            | FINISHED |
| 16997   | l_commitdate    |         | 2023-07-27 16:02:34    | 8380            | FINISHED |
| 16998   | l_partkey       |         | 2023-07-27 16:02:40    | 6060            | FINISHED |
| 16999   | l_quantity      |         | 2023-07-27 16:02:50    | 9768            | FINISHED |
| 17000   | l_orderkey      |         | 2023-07-27 16:02:57    | 7200            | FINISHED |
| 17001   | l_comment       |         | 2023-07-27 16:03:36    | 38468           | FINISHED |
| 17002   | l_linestatus    |         | 2023-07-27 16:03:51    | 15226           | FINISHED |
| 17003   | l_extendedprice |         | 2023-07-27 16:04:00    | 8713            | FINISHED |
| 17004   | l_linenumber    |         | 2023-07-27 16:04:06    | 6659            | FINISHED |
| 17005   | l_shipinstruct  |         | 2023-07-27 16:04:36    | 29777           | FINISHED |
| 17006   | l_discount      |         | 2023-07-27 16:04:45    | 9212            | FINISHED |
| 17007   | TableRowCount   |         | 2023-07-27 16:04:52    | 6974            | FINISHED |
+---------+-----------------+---------+-----------------------+-----------------+----------+
```

- Terminate unfinished jobs

```
KILL ANALYZE [job_id]
```

- Delete periodic collection job

```
DROP ANALYZE JOB [JOB_ID]
```

Show statistics

Show statistics includes show table statistics (number of rows) and column statistics. Please refer to View statistics in Internal Table Statistics

Table statistics

```
SHOW TABLE [cached] stats TABLE_NAME;
```

View row count of the given table. If the cached parameter is specified, the row count of the specified table that has been loaded into the cache is displayed.

```
mysql> SHOW TABLE STATS hive.tpch100.orders;
+-----------+---------------------+---------------------+
| row_count | update_time         | last_analyze_time   |
+-----------+---------------------+---------------------+
| 150000000 | 2023-07-11 23:01:49 | 2023-07-11 23:01:44 |
+-----------+---------------------+---------------------+
```

Column statistics

```
SHOW COLUMN [cached] stats TABLE_NAME;
```

View the column statistics of a table. If the cached parameter is specified, the column information of the specified table that has been loaded into the cache is displayed.

```
mysql> SHOW COLUMN stats hive.tpch100.orders;
+-----------------+-------+--------------+----------+----------------------+--------------+----------------------+----------------
------------+
| column_name     | count | ndv          | num_null | data_size            | avg_size_byte | min                  | max
|
+-----------------+-------+--------------+----------+----------------------+--------------+----------------------+----------------
------------+
| o_orderstatus   | 1.5E8 | 3.0          | 0.0      | 1.50000001E8         | 1.0           | 'F'                  | 'P'
|
| o_clerk         | 1.5E8 | 100836.0     | 0.0      | 2.250000015E9        | 15.0          | 'Clerk#000000001'    |
'Clerk#000100000'          |
| o_orderdate     | 1.5E8 | 2417.0       | 0.0      | 6.00000004E8         | 4.0           | '1992-01-01'         | '1998-08-02'
|
| o_shippriority  | 1.5E8 | 1.0          | 0.0      | 6.00000004E8         | 4.0           | 0                    | 0
|
| o_custkey       | 1.5E8 | 1.0023982E7  | 0.0      | 6.00000004E8         | 4.0           | 1                    | 14999999
|
| o_totalprice    | 1.5E8 | 3.4424096E7  | 0.0      | 1.200000008E9        | 8.0           | 811.73               | 591036.15
|
| o_orderkey      | 1.5E8 | 1.51621184E8 | 0.0      | 1.200000008E9        | 8.0           | 1                    | 600000000
|
| o_comment       | 1.5E8 | 1.10204136E8 | 0.0      | 7.275038757500258E9  | 48.50025806   | ' Tiresias about the' | 'zzle? unusual
requests w' |
| o_orderpriority | 1.5E8 | 5.0          | 0.0      | 1.2600248124001656E9 | 8.40016536    | '1-URGENT'           | '5-LOW'
|
+-----------------+-------+--------------+----------+----------------------+--------------+----------------------+----------------
------------+
```

Modify the statistics

Modify statistics supports users to manually modify column statistics. You can modify the row_count, ndv, num_nulls, min_value, max_value, data_size and other information of the specified column.

Please refer to Modify the statistics in Internal Table Statistics

```
mysql> ALTER TABLE hive.tpch100.orders MODIFY COLUMN o_orderstatus SET STATS ('row_count'='6001215');
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW COLUMN stats hive.tpch100.orders;
+-----------------+-----------+--------------+----------+----------------------+--------------+----------------------+-----------
----------------+
| column_name     | count     | ndv          | num_null | data_size            | avg_size_byte | min                  | max
|
+-----------------+-----------+--------------+----------+----------------------+--------------+----------------------+-----------
----------------+
| o_orderstatus   | 6001215.0 | 0.0          | 0.0      | 0.0                  | 0.0           | 'NULL'               | 'NULL'
|
| o_clerk         | 1.5E8     | 100836.0     | 0.0      | 2.250000015E9        | 15.0          | 'Clerk#000000001'    |
'Clerk#000100000'          |
| o_orderdate     | 1.5E8     | 2417.0       | 0.0      | 6.00000004E8         | 4.0           | '1992-01-01'         | '1998-08-
02'              |
| o_shippriority  | 1.5E8     | 1.0          | 0.0      | 6.00000004E8         | 4.0           | 0                    | 0
|
| o_custkey       | 1.5E8     | 1.0023982E7  | 0.0      | 6.00000004E8         | 4.0           | 1                    | 14999999
|
| o_totalprice    | 1.5E8     | 3.4424096E7  | 0.0      | 1.200000008E9        | 8.0           | 811.73               | 591036.15
|
| o_orderkey      | 1.5E8     | 1.51621184E8 | 0.0      | 1.200000008E9        | 8.0           | 1                    | 600000000
|
| o_comment       | 1.5E8     | 1.10204136E8 | 0.0      | 7.275038757500258E9  | 48.50025806   | ' Tiresias about the' | 'zzle?
unusual requests w' |
| o_orderpriority | 1.5E8     | 5.0          | 0.0      | 1.2600248124001656E9 | 8.40016536    | '1-URGENT'           | '5-LOW'
|
+-----------------+-----------+--------------+----------+----------------------+--------------+----------------------+-----------
----------------+
```

Delete stattistics

Deleting table statistics supports users to delete table row count and column statistics of a table. If the user specifies the column names to be deleted, only the statistics of these columns will be deleted. If not specified, the statistical information of all columns of the entire table and the number of rows of the table will be deleted.

Please refer to Delete statistics in Internal Table Statistics

- Delete statistics of the entire table

```
DROP STATS hive.tpch100.orders
```

- Delete the statistics of certain columns in the table

```
DROP STATS hive.tpch100.orders (o_orderkey, o_orderdate)
```

Implementation principle

Statistics Data Sources

The optimizer (Nereids) reads statistical information through the cache, and there are two data sources for the cache to load data from.

The first data source is the internal statistics table, and the data in the statistics table is collected by the user to execute analyze statement. The structure of this part is the same as that of the internal table. Users can execute the analyze statement on the external table to collect statistical information just like analyzing the internal tables.

Different from the internal table, the statistic cache for external table has a second data source, the stats collector. The stats collector defines some interfaces for obtaining statistical information from external catalog. Hive metastore and Iceberg metadata. These interfaces can obtain existing statistical information in external catalog. Take hive as an example. If the user has performed the analyze operation in hive, then when querying in Doris, Doris can directly load the existing statistical information from the hive metastore into the cache, including the number of rows in the table, the maximum and minimum values of columns, etc. If the external data source does not have statistical information, the stats connector will roughly estimate row count based on the size of the data file and the schema of the table. In this case, the column statistics are missing, which may cause the optimizer to generate a relatively inefficient execution plan.

The Stats collector is automatically executed when there is no data in the statistics table, and it is transparent to the user, user does not need to execute commands or make any settings.

Cache loading

The loading sequence of the cache is firstly loaded through the Statistics table. If there is information in the Statistics table, it means that the user has performed analyze operation in doris. The collected statistical information through analyze is the most accurate, so we prioritize loading from the Statistics table. If you find that there is no information about the currently required table in Statistics, Doris will try to obtain it from an external data source through stats collector. If the external data source does not have column statistics either, the stats collector will estimate a row count based on file size and table schema.

Since the cache is loaded asynchronously, the statistical information may not be available for the first query, because the cache loading has just been triggered at this time. But in general, it can be guaranteed that when a table is queried for the second time, the optimizer can obtain its statistical information from the cache.

Cluster Upgrade

Overview

To upgrade, please use the steps recommended in this chapter to upgrade the cluster. The Doris cluster upgrade can be upgraded using the **rolling upgrade** method, which does not require all cluster nodes to be shut down for upgrade, which greatly reduces the impact on upper-layer applications.

Doris Release Notes

When upgrading Doris, please follow the principle of **not skipping two minor versions** and upgrade sequentially.

For example, if you are upgrading from version 0.15.x to 2.0.x, it is recommended to first upgrade to the latest version of 1.1, then upgrade to the latest version of 1.2, and finally upgrade to the latest version of 2.0.

Upgrade Steps

Upgrade Instructions

1. During the upgrade process, since Doris's RoutineLoad, Flink-Doris-Connector, and Spark-Doris-Connector have implemented a retry mechanism in the code, in a multi-BE node cluster, the rolling upgrade will not cause the task to fail .
2. The StreamLoad task requires you to implement a retry mechanism in your own code, otherwise the task will fail.
3. The cluster copy repair and balance function must be closed before and opened after the completion of a single upgrade task, regardless of whether all your cluster nodes have been upgraded.

Overview of the Upgrade Process

1. Metadata backup
2. Turn off the cluster copy repair and balance function
3. Compatibility testing
4. Upgrade BE
5. Upgrade FE
6. Turn on the cluster replica repair and balance function

Upgrade Pre-work

Please perform the upgrade in sequence according to the upgrade process

01 Metadata Backup (Important)

Make a full backup of the `doris-meta` directory of the FE-Master node!

02 Turn off the cluster replica repair and balance function

There will be node restart during the upgrade process, so unnecessary cluster balancing and replica repair logic may be triggered, first close it with the following command:

```
admin set frontend config("disable_balance" = "true");
admin set frontend config("disable_colocate_balance" = "true");
admin set frontend config("disable_tablet_scheduler" = "true");
```

03 Compatibility Testing

> **Warning**
>
> **Metadata compatibility is very important, if the upgrade fails due to incompatible metadata, it may lead to data loss! It is recommended to perform a metadata compatibility test before each upgrade!**

1. FE Compatibility Test

> **Important**
>
> 1. It is recommended to do FE compatibility test on your local development machine or BE node.
> 2. It is not recommended to test on Follower or Observer nodes to avoid link exceptions
> 3. If it must be on the Follower or Observer node, the started FE process needs to be stopped

a. Use the new version alone to deploy a test FE process

```
sh ${DORIS_NEW_HOME}/bin/start_fe.sh --daemon
```

b. Modify the FE configuration file fe.conf for testing

```
vi ${DORIS_NEW_HOME}/conf/fe.conf
```

Modify the following port information, set **all ports** to **different from online**

```
...
http_port = 18030
rpc_port = 19020
query_port = 19030
arrow_flight_sql_port = 19040
edit_log_port = 19010
...
```

Save and exit

c. Modify fe.conf

- Add ClusterID configuration in fe.conf

```
echo "cluster_id=123456" >> ${DORIS_NEW_HOME}/conf/fe.conf
```

- Add metadata failover configuration in fe.conf (**>=2.0.2 + version does not require this operation**)

```
echo "metadata_failure_recovery=true" >> ${DORIS_NEW_HOME}/conf/fe.conf
```

d. Copy the metadata directory doris-meta of the online environment Master FE to the test environment

```
cp ${DORIS_OLD_HOME}/fe/doris-meta/* ${DORIS_NEW_HOME}/fe/doris-meta
```

e. Change the cluster_id in the VERSION file copied to the test environment to 123456 (that is, the same as in step 3)

```
vi ${DORIS_NEW_HOME}/fe/doris-meta/image/VERSION
clusterId=123456
```

f. In the test environment, run the startup FE

- If the version is greater than or equal to 2.0.2, run the following command

```
sh ${DORIS_NEW_HOME}/bin/start_fe.sh --daemon --metadata_failure_recovery
```

- If the version is less than 2.0.2, run the following command `shell sh ${DORIS_NEW_HOME}/bin/start_fe.sh --daemon`

g. Observe whether the startup is successful through the FE log fe.log

```
tail -f ${DORIS_NEW_HOME}/log/fe.log
```

h. If the startup is successful, it means that there is no problem with the compatibility, stop the FE process of the test environment, and prepare for the upgrade

```
sh ${DORIS_NEW_HOME}/bin/stop_fe.sh
```

2. BE Compatibility Test

You can use the grayscale upgrade scheme to upgrade a single BE first. If there is no exception or error, the compatibility is considered normal, and subsequent upgrade actions can be performed

Upgrade process

> **Tip**
>
> Upgrade BE first, then FE
>
> Generally speaking, Doris only needs to upgrade `/bin` and `/lib` under the FE directory and `/bin` and `/lib` under the BE directory
>
> In versions 2.0.2 and later, the `custom_lib/` directory is added to the FE and BE deployment paths (if not, it can be created manually). The `custom_lib/` directory is used to store some user-defined third-party jar packages, such as `hadoop-lzo-*.jar`, `orai18n.jar`, etc.
>
> This directory does not need to be replaced during upgrade.
>
> However, when a major version is upgraded, new features may be added or old functions refactored. These modifications may require **replace/add** more directories during the upgrade to ensure the availability of all new features. Please Carefully pay attention to the Release-Note of this version when upgrading the version to avoid upgrade failures

Upgrade BE

> **Tip**
>
> In order to ensure the safety of your data, please use 3 copies to store your data to avoid data loss caused by misoperation or failure of the upgrade

1. Under the premise of multiple copies, select a BE node to stop running and perform grayscale upgrade

   ```
   sh ${DORIS_OLD_HOME}/be/bin/stop_be.sh
   ```

2. Rename the `/bin`, `/lib` directories under the BE directory

   ```
   mv ${DORIS_OLD_HOME}/be/bin ${DORIS_OLD_HOME}/be/bin_back
   mv ${DORIS_OLD_HOME}/be/lib ${DORIS_OLD_HOME}/be/lib_back
   ```

3. Copy the new version of `/bin`, `/lib` directory to the original BE directory

   ```
   cp -r ${DORIS_NEW_HOME}/be/bin ${DORIS_OLD_HOME}/be/bin
   cp -r ${DORIS_NEW_HOME}/be/lib ${DORIS_OLD_HOME}/be/lib
   ```

4. Start the BE node

   ```
   sh ${DORIS_OLD_HOME}/be/bin/start_be.sh --daemon
   ```

5. Link the cluster to view the node information

```
show backends\G
```

   If the `alive` status of the BE node is `true`, and the value of `Version` is the new version, the node upgrade is successful

6. Complete the upgrade of other BE nodes in sequence

05 Upgrade FE

> **Tip**
>
> Upgrade the non-Master nodes first, and then upgrade the Master nodes.

1. In the case of multiple FE nodes, select a non-Master node to upgrade and stop running first

```
sh ${DORIS_OLD_HOME}/fe/bin/stop_fe.sh
```

2. Rename the `/bin`, `/lib` directories under the FE directory

```
mv ${DORIS_OLD_HOME}/fe/bin ${DORIS_OLD_HOME}/fe/bin_back
mv ${DORIS_OLD_HOME}/fe/lib ${DORIS_OLD_HOME}/fe/lib_back
```

3. Copy the new version of `/bin`, `/lib` directory to the original FE directory

```
cp -r ${DORIS_NEW_HOME}/fe/bin ${DORIS_OLD_HOME}/fe/bin
cp -r ${DORIS_NEW_HOME}/fe/lib ${DORIS_OLD_HOME}/fe/lib
```

4. Start the FE node

```
sh ${DORIS_OLD_HOME}/fe/bin/start_fe.sh --daemon
```

5. Link the cluster to view the node information

```
show frontends\G
```

   If the FE node `alive` status is `true`, and the value of `Version` is the new version, the node is upgraded successfully

6. Complete the upgrade of other FE nodes in turn, **finally complete the upgrade of the Master node**

06 Turn on the cluster replica repair and balance function

After the upgrade is complete and all BE nodes become `Alive`, enable the cluster copy repair and balance function:

```
admin set frontend config("disable_balance" = "false");
admin set frontend config("disable_colocate_balance" = "false");
admin set frontend config("disable_tablet_scheduler" = "false");
```

Elastic scaling

Elastic scaling

Doris can easily expand and shrink FE, BE, Broker instances.

FE Scaling

High availability of FE can be achieved by expanding FE to three top-one nodes.

Users can login to Master FE through MySQL client. By:

```
SHOW PROC '/frontends';
```

To view the current FE node situation.

You can also view the FE node through the front-end page connection: `http://fe_hostname:fe_http_port/frontend` or `http://fe_hostname:fe_http_port/system?Path=//frontends`.

All of the above methods require Doris's root user rights.

The process of FE node expansion and contraction does not affect the current system operation.

Adding FE nodes

FE is divided into two roles: Follower and Observer. The Follower role will elect a Follower node as the Master. By default, a cluster can only have one Follower role in the Master state, and there can be multiple Followers and Observers. At the same time, it is necessary to ensure that there are an odd number of Follower roles. All Follower roles form an election group. If the Follower in the Master state goes down, the remaining Followers will automatically elect a new Master to ensure high write availability. Observer synchronizes the data of Master, but does not participate in the election. If only one FE is deployed, the FE is the Master by default.

The first FE to start automatically becomes Master. On this basis, several Followers and Observers can be added.

Configure and start Follower or Observer.

Follower and Observer are configured with Master. The following commands need to be executed at the first startup:

```
bin/start_fe.sh --helper host:edit_log_port --daemon
```

The host is the node IP of Master, and the edit_log_port in Lead's configuration file fe.conf. The –helper is only required when follower/observer is first startup.

Add Follower or Observer to the cluster

Add Follower or Observer. Connect to the started FE using mysql-client and execute:

```
ALTER SYSTEM ADD FOLLOWER "follower_host:edit_log_port";
```

or

```
ALTER SYSTEM ADD OBSERVER "observer_host:edit_log_port";
```

The follower_host and observer_host is the node IP of Follower or Observer, and the edit_log_port in its configuration file fe.conf.

View the status of Follower or Observer. Connect to any booted FE using mysql-client and execute:

```
SHOW PROC '/frontends';
```

You can view the FE currently joined the cluster and its corresponding roles.

> Notes for FE expansion:
>
> 1. The number of Follower FEs (including Masters) must be odd. It is recommended that a maximum of three constituent high availability (HA) modes be deployed.
> 2. When FE is in a highly available deployment (1 Master, 2 Follower), we recommend that the reading service capability of FE be extended by adding Observer FE. Of course, you can continue to add Follower FE, but it's almost unnecessary.
> 3. Usually a FE node can handle 10-20 BE nodes. It is suggested that the total number of FE nodes should be less than 10. Usually three can meet most of the needs.
> 4. The helper cannot point to the FE itself, it must point to one or more existing running Master/Follower FEs.

Delete FE nodes

Delete the corresponding FE node using the following command:

```
ALTER SYSTEM DROP FOLLOWER[OBSERVER] "fe_host:edit_log_port";
```

> Notes for FE contraction:
>
> 1. When deleting Follower FE, make sure that the remaining Follower (including Master) nodes are odd.

BE Scaling

Users can login to Master FE through mysql-client. By:

```
SHOW PROC '/backends';
```

To see the current BE node situation.

You can also view the BE node through the front-end page connection: `http://fe_hostname:fe_http_port/backend` or `http://fe_hostname:fe_http_port/system?Path=//backends`.

All of the above methods require Doris's root user rights.

The expansion and scaling process of BE nodes does not affect the current system operation and the tasks being performed, and does not affect the performance of the current system. Data balancing is done automatically. Depending on the amount of data available in the cluster, the cluster will be restored to load balancing in a few hours to a day. For cluster load, see the Tablet Load Balancing Document.

Add BE nodes

The BE node is added in the same way as in the **BE deployment** section. The BE node is added by the `ALTER SYSTEM ADD BACKEND` command.

> Notes for BE expansion:
>
> 1. After BE expansion, Doris will automatically balance the data according to the load, without affecting the use during the period.

Delete BE nodes

There are two ways to delete BE nodes: DROP and DECOMMISSION

The DROP statement is as follows:

```
ALTER SYSTEM DROP BACKEND "be_host:be_heartbeat_service_port";
```

Note: DROP BACKEND will delete the BE directly and the data on it will not be recovered!!! So we strongly do not recommend DROP BACKEND to delete BE nodes. When you use this statement, there will be corresponding error-proof operation hints.

DECOMMISSION clause:

```
ALTER SYSTEM DECOMMISSION BACKEND "be_host:be_heartbeat_service_port";
```

> DECOMMISSION notes:
>
> 1. This command is used to safely delete BE nodes. After the command is issued, Doris attempts to migrate the data on the BE to other BE nodes, and when all data is migrated, Doris automatically deletes the node.
> 2. The command is an asynchronous operation. After execution, you can see that the BE node's `SystemDecommissioned` status is true through `` `SHOW PROC '/backends';` `` Indicates that the node is offline.
> 3. The order **does not necessarily carry out successfully**. For example, when the remaining BE storage space is insufficient to accommodate the data on the offline BE, or when the number of

remaining machines does not meet the minimum number of replicas, the command cannot be completed, and the BE will always be in the state of `SystemDecommissioned` as true.
4. The progress of DECOMMISSION can be viewed through `SHOW PROC '/backends';` Tablet Num, and if it is in progress, Tablet Num will continue to decrease.
5. The operation can be carried out by: `CANCEL ALTER SYSTEM DECOMMISSION BACKEND "be_host:be_heartbeat_service_port";` The order was cancelled. When cancelled, the data on the BE will maintain the current amount of data remaining. Follow-up Doris re-load balancing

For expansion and scaling of BE nodes in multi-tenant deployment environments, please refer to the Multi-tenant Design Document.

Broker Scaling

There is no rigid requirement for the number of Broker instances. Usually one physical machine is deployed. Broker addition and deletion can be accomplished by following commands:

```
ALTER SYSTEM ADD BROKER broker_name "broker_host:broker_ipc_port"; ALTER SYSTEM DROP BROKER broker_name "broker_host:broker_ipc_port"; ALTER SYSTEM DROP ALL BROKER broker_name;
```

Broker is a stateless process that can be started or stopped at will. Of course, when it stops, the job running on it will fail. Just try again.

Load Balancing

When deploying multiple FE nodes, users can deploy a load balancing layer on top of multiple FEs to achieve high availability of Doris.

Code method

Retry and load balance yourself in the application layer code. For example, if a connection is found to be down, it will automatically retry on other connections. Application layer code retry requires the application to configure multiple doris front-end node addresses.

JDBC Connector

If you use mysql jdbc connector to connect to Doris, you can use jdbc's automatic retry mechanism:

```
jdbc:mysql:loadbalance://[host:port],[host:port].../[database][?propertyName1][=propertyValue1][&propertyName2][=propertyValue
```

For details, please refer to Mysql official website document

ProxySQL method

ProxySQL is a flexible and powerful MySQL proxy layer. It is a MySQL middleware that can be actually used in a production environment. It can realize read-write separation, support Query routing function, support dynamic designation of a certain SQL for cache, support dynamic loading configuration, failure Switching and some SQL filtering functions.

Doris's FE process is responsible for receiving user connections and query requests. It itself is horizontally scalable and highly available, but it requires users to set up a proxy on multiple FEs to achieve automatic connection load balancing.

Install ProxySQL (yum way)

```
 Configure yum source
# vim /etc/yum.repos.d/proxysql.repo
[proxysql_repo]
name= ProxySQL YUM repository
baseurl=http://repo.proxysql.com/ProxySQL/proxysql-1.4.x/centos/\$releasever
gpgcheck=1
gpgkey=http://repo.proxysql.com/ProxySQL/repo_pub_key

Perform installation
# yum clean all
# yum makecache
# yum -y install proxysql
View version
# proxysql --version
ProxySQL version 1.4.13-15-g69d4207, codename Truls
Set up auto start
# systemctl enable proxysql
```

```
# systemctl start proxysql
# systemctl status proxysql
After startup, it will listen to two ports, the default is 6032 and 6033. Port 6032 is the management port of ProxySQL, and 6033 is
the port for ProxySQL to provide external services (that is, the forwarding port connected to the real database of the forwarding
backend).
# netstat -tunlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:6032            0.0.0.0:*               LISTEN      23940/proxysql
tcp        0      0 0.0.0.0:6033            0.0.0.0:*               LISTEN
```

ProxySQL Config

ProxySQL has a configuration file `/etc/proxysql.cnf` and a configuration database file `/var/lib/proxysql/proxysql.db`. **Special attention is needed here**: If there is a "`proxysql.db`" file (under the `/var/lib/proxysql` directory), the ProxySQL service will only be read when it is started for the first time The `proxysql.cnf file` and parse it; after startup, the `proxysql.cnf` file will not be read! If you want the configuration in the proxysql.cnf file to take effect after restarting the proxysql service (that is, you want proxysql to read and parse the proxysql.cnf configuration file when it restarts), you need to delete `/var/lib/proxysql/proxysql first.db`database file, and then restart the proxysql service. This is equivalent to initializing the proxysql service, and a pure proxysql.db database file will be produced again (if proxysql related routing rules, etc. are configured before, it will be erased)

View and modify configuration files

Here are mainly a few parameters, which have been commented out below, and you can modify them according to your needs

```
# egrep -v "^#|^$" /etc/proxysql.cnf
datadir="/var/lib/proxysql"         #data dir
admin_variables=
{
        admin_credentials="admin:admin"  #User name and password for connecting to the management terminal
        mysql_ifaces="0.0.0.0:6032"    #Management port, used to connect to proxysql management database
}
mysql_variables=
{
        threads=4      #Specify the number of threads opened for the forwarding port
        max_connections=2048
        default_query_delay=0
        default_query_timeout=36000000
        have_compress=true
        poll_timeout=2000
        interfaces="0.0.0.0:6033"    #Specify the forwarding port, used to connect to the back-end mysql database, which is
equivalent to acting as a proxy
        default_schema="information_schema"
        stacksize=1048576
        server_version="5.5.30"        #Specify the version of the backend mysql
        connect_timeout_server=3000
        monitor_username="monitor"
        monitor_password="monitor"
        monitor_history=600000
        monitor_connect_interval=60000
        monitor_ping_interval=10000
        monitor_read_only_interval=1500
        monitor_read_only_timeout=500
        ping_interval_server_msec=120000
        ping_timeout_server=500
```

```
        commands_stats=true
        sessions_sort=true
        connect_retries_on_failure=10
}
mysql_servers =
(
)
mysql_users:
(
)
mysql_query_rules:
(
)
scheduler=
(
)
mysql_replication_hostgroups=
(
)
```

Connect to the ProxySQL management port test

```
# mysql -uadmin -padmin -P6032 -hdorisO1
View the global_variables table information of the main library (it is in this library after login by default)
MySQL [(none)]> show databases;
+-----+---------------+-------------------------------------+
| seq | name          | file                                |
+-----+---------------+-------------------------------------+
| 0   | main          |                                     |
| 2   | disk          | /var/lib/proxysql/proxysql.db       |
| 3   | stats         |                                     |
| 4   | monitor       |                                     |
| 5   | stats_history | /var/lib/proxysql/proxysql_stats.db |
+-----+---------------+-------------------------------------+
5 rows in set (0.000 sec)
MySQL [(none)]> use main;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [main]> show tables;
+--------------------------------------------+
| tables                                     |
+--------------------------------------------+
| global_variables                           |
| mysql_collations                           |
| mysql_group_replication_hostgroups         |
| mysql_query_rules                          |
| mysql_query_rules_fast_routing             |
| mysql_replication_hostgroups               |
| mysql_servers                              |
| mysql_users                                |
| proxysql_servers                           |
| runtime_checksums_values                   |
| runtime_global_variables                   |
| runtime_mysql_group_replication_hostgroups |
| runtime_mysql_query_rules                  |
| runtime_mysql_query_rules_fast_routing     |
| runtime_mysql_replication_hostgroups       |
| runtime_mysql_servers                      |
| runtime_mysql_users                        |
| runtime_proxysql_servers                   |
| runtime_scheduler                          |
```

```
| scheduler                                  |
+--------------------------------------------+
20 rows in set (0.000 sec)
```

ProxySQL configuration backend Doris FE

Use the insert statement to add the host to the mysql_servers table, where: hostgroup_id is 10 for the write group, and 20 for the read group. We don't need to read and write the license here, and it doesn't matter which one can be set randomly.

```
[root@mysql-proxy ~]# mysql -uadmin -padmin -P6032 -h127.0.0.1
............
MySQL [(none)]> insert into mysql_servers(hostgroup_id,hostname,port) values(10,'192.168.9.211',9030);
Query OK, 1 row affected (0.000 sec)

MySQL [(none)]> insert into mysql_servers(hostgroup_id,hostname,port) values(10,'192.168.9.212',9030);
Query OK, 1 row affected (0.000 sec)

MySQL [(none)]> insert into mysql_servers(hostgroup_id,hostname,port) values(10,'192.168.9.213',9030);
Query OK, 1 row affected (0.000 sec)

If an error occurs during the insertion process:
ERROR 1045 (#2800): UNIQUE constraint failed: mysql_servers.hostgroup_id, mysql_servers.hostname, mysql_servers.port

It means that other configurations may have been defined before, you can clear this table or delete the configuration of the
corresponding host
MySQL [(none)]> select * from mysql_servers;
MySQL [(none)]> delete from mysql_servers;
Query OK, 6 rows affected (0.000 sec)

Check whether these 3 nodes are inserted successfully and their status.
MySQL [(none)]> select * from mysql_servers\G;
*************************** 1. row ***************************
       hostgroup_id: 10
           hostname: 192.168.9.211
               port: 9030
             status: ONLINE
             weight: 1
        compression: 0
    max_connections: 1000
max_replication_lag: 0
            use_ssl: 0
     max_latency_ms: 0
            comment:
*************************** 2. row ***************************
       hostgroup_id: 10
           hostname: 192.168.9.212
               port: 9030
             status: ONLINE
             weight: 1
        compression: 0
    max_connections: 1000
max_replication_lag: 0
            use_ssl: 0
     max_latency_ms: 0
            comment:
*************************** 3. row ***************************
       hostgroup_id: 10
           hostname: 192.168.9.213
               port: 9030
             status: ONLINE
             weight: 1
```

```
        compression: 0
    max_connections: 1000
max_replication_lag: 0
            use_ssl: 0
     max_latency_ms: 0
            comment:
6 rows in set (0.000 sec)

ERROR: No query specified

After the above modification, load it to RUNTIME and save it to disk. The following two steps are very important, otherwise your
configuration information will be gone after you exit and must be saved
MySQL [(none)]> load mysql servers to runtime;
Query OK, 0 rows affected (0.006 sec)

MySQL [(none)]> save mysql servers to disk;
Query OK, 0 rows affected (0.348 sec)
```

Monitor Doris FE node configuration

After adding doris fe nodes, you also need to monitor these back-end nodes. For multiple FE high-availability load balancing environments on the backend, this is necessary because ProxySQL needs to be automatically adjusted by the read_only value of each node

Whether they belong to the read group or the write group.

First create a user name for monitoring on the back-end master main data node

```
Execute on the node of the doris fe master master database:
# mysql -P9030 -uroot -p
mysql> create user monitor@'192.168.9.%' identified by 'P@ssword1!';
Query OK, 0 rows affected (0.03 sec)
mysql> grant ADMIN_PRIV on *.* to monitor@'192.168.9.%';
Query OK, 0 rows affected (0.02 sec)

Then go back to the mysql-proxy proxy layer node to configure monitoring
# mysql -uadmin -padmin -P6032 -h127.0.0.1
MySQL [(none)]> set mysql-monitor_username='monitor';
Query OK, 1 row affected (0.000 sec)

MySQL [(none)]> set mysql-monitor_password='P@ssword1!';
Query OK, 1 row affected (0.000 sec)

After modification, load to RUNTIME and save to disk
MySQL [(none)]> load mysql variables to runtime;
Query OK, 0 rows affected (0.001 sec)

MySQL [(none)]> save mysql variables to disk;
Query OK, 94 rows affected (0.079 sec)

Verify the monitoring results: The indicators of the ProxySQL monitoring module are stored in the log table of the monitor library.
The following is the monitoring of whether the connection is normal (monitoring of connect indicators):
Note: There may be many connect_errors, this is because there is an error when the monitoring information is not configured. After
the configuration, if the result of connect_error is NULL, it means normal。
MySQL [(none)]> select * from mysql_server_connect_log;
+---------------+------+------------------+-------------------------+---------------+
| hostname      | port | time_start_us    | connect_success_time_us | connect_error |
+---------------+------+------------------+-------------------------+---------------+
| 192.168.9.211 | 9030 | 1548665195883957 | 762                     | NULL          |
| 192.168.9.212 | 9030 | 1548665195894099 | 399                     | NULL          |
| 192.168.9.213 | 9030 | 1548665195904266 | 483                     | NULL          |
| 192.168.9.211 | 9030 | 1548665255883715 | 824                     | NULL          |
```

```
| 192.168.9.212 | 9030 | 1548665255893942 | 656                     | NULL          |
| 192.168.9.211 | 9030 | 1548665495884125 | 615                     | NULL          |
| 192.168.9.212 | 9030  | 1548665495894254 | 441                      | NULL          |
| 192.168.9.213 | 9030 | 1548665495904479 | 638                     | NULL          |
| 192.168.9.211 | 9030 | 1548665512917846 | 487                     | NULL          |
| 192.168.9.212 | 9030 | 1548665512928071 | 994                     | NULL          |
| 192.168.9.213 | 9030 | 1548665512938268 | 613                     | NULL          |
+---------------+------+------------------+-------------------------+---------------+
20 rows in set (0.000 sec)
The following is the monitoring of heartbeat information (monitoring of ping indicators)
MySQL [(none)]> select * from mysql_server_ping_log;
+---------------+------+------------------+----------------------+------------+
| hostname      | port | time_start_us    | ping_success_time_us | ping_error |
+---------------+------+------------------+----------------------+------------+
| 192.168.9.211 | 9030 | 1548665195883407 | 98                   | NULL       |
| 192.168.9.212 | 9030 | 1548665195885128 | 119                  | NULL       |
...........
| 192.168.9.213 | 9030 | 1548665415889362 | 106                  | NULL       |
| 192.168.9.213 | 9030 | 1548665562898295 | 97                   | NULL       |
+---------------+------+------------------+----------------------+------------+
110 rows in set (0.001 sec)

The read_only log is also empty at this time (normally, when the new environment is configured, this read-only log is empty)
MySQL [(none)]> select * from mysql_server_read_only_log;
Empty set (0.000 sec)

All 3 nodes are in the group with hostgroup_id=10.
Now, load the modification of the mysql_replication_hostgroups table just now to RUNTIME to take effect。
MySQL [(none)]> load mysql servers to runtime;
Query OK, 0 rows affected (0.003 sec)

MySQL [(none)]> save mysql servers to disk;
Query OK, 0 rows affected (0.361 sec)


MySQL [(none)]> select hostgroup_id,hostname,port,status,weight from mysql_servers;
+--------------+---------------+------+--------+--------+
| hostgroup_id | hostname      | port | status | weight |
+--------------+---------------+------+--------+--------+
| 10           | 192.168.9.211 | 9030 | ONLINE | 1      |
| 20           | 192.168.9.212 | 9030 | ONLINE | 1      |
| 20           | 192.168.9.213 | 9030 | ONLINE | 1      |
+--------------+---------------+------+--------+--------+
3 rows in set (0.000 sec)
```

Configure Doris users

All the above configurations are about the back-end Doris FE node. Now you can configure the SQL statements, including: the user who sends the SQL statement, the routing rules of the SQL statement, the cache of the SQL query, the rewriting of the SQL statement, and so on.

This section is the user configuration used by the SQL request, such as the root user. This requires that we need to add relevant users to the back-end Doris FE node first. Here are examples of two user names root and doris.

```
First, execute on the Doris FE master master database node:
# mysql -P9030 -uroot -p
.........

mysql> create user doris@'%' identified by 'P@ssword1!';
Query OK, 0 rows affected, 1 warning (0.04 sec)
```

```
mysql> grant ADMIN_PRIV on *.* to doris@'%';
Query OK, 0 rows affected, 1 warning (0.03 sec)


Then go back to the mysql-proxy proxy layer node, configure the mysql_users table, and add the two users just now to the table.
admin> insert into mysql_users(username,password,default_hostgroup) values('root','',10);
Query OK, 1 row affected (0.001 sec)

admin> insert into mysql_users(username,password,default_hostgroup) values('doris','P@ssword1!',10);
Query OK, 1 row affected (0.000 sec)

admin> load mysql users to runtime;
Query OK, 0 rows affected (0.001 sec)

admin> save mysql users to disk;
Query OK, 0 rows affected (0.108 sec)

The mysql_users table has many fields. The three main fields are username, password, and default_hostgroup:
      -username: The username used by the front-end to connect to ProxySQL and ProxySQL to route SQL statements to MySQL.
      -password: the password corresponding to the user name. It can be a plain text password or a hash password. If you want to use the hash password, you can execute it on a MySQL node first  select password(PASSWORD), and then copy the encryption result to this field.
      -default_hostgroup: The default routing destination of the username. For example, when the field value of the specified root user is 10, the SQL statement sent by the root user is used by default
    In this case, it will be routed to a node in the hostgroup_id=10 group.

admin> select * from mysql_users\G
*************************** 1. row ***************************
              username: root
              password:
                active: 1
               use_ssl: 0
     default_hostgroup: 10
        default_schema: NULL
         schema_locked: 0
transaction_persistent: 1
          fast_forward: 0
               backend: 1
              frontend: 1
       max_connections: 10000
*************************** 2. row ***************************
              username: doris
              password: P@ssword1!
                active: 1
               use_ssl: 0
     default_hostgroup: 10
        default_schema: NULL
         schema_locked: 0
transaction_persistent: 1
          fast_forward: 0
               backend: 1
              frontend: 1
       max_connections: 10000
2 rows in set (0.000 sec)

Although the mysql_users table is not described in detail here, only users with active=1 are valid users, and the default active is 1.

MySQL [(none)]> load mysql users to runtime;
Query OK, 0 rows affected (0.001 sec)

MySQL [(none)]> save mysql users to disk;
Query OK, 0 rows affected (0.123 sec)
```

```
In this way, you can use the doris username and password to connect to ProxySQL through the sql client
```

Connect to Doris through ProxySQL for testing

Next, use the root user and doris user to test whether they can be routed to the default hostgroup_id=10 (it is a write group) to read data. The following is connected through the forwarding port 6033, the connection is forwarded to the real back-end database!

```
#mysql -uroot -p -P6033 -hdoris01 -e "show databases;"
Enter password:
ERROR 9001 (HY000) at line 1: Max connect timeout reached while reaching hostgroup 10 after 10000ms
At this time, an error was found, and it was not forwarded to the real doris fe on the backend.
Through the log, you can see that there is set autocommit=0 to open the transaction
Check the configuration found:

mysql-forward_autocommit=false
mysql-autocommit_false_is_transaction=false

We don’ t need to read and write separation here, just turn these two parameters into true directly through the following statement.
mysql> UPDATE global_variables SET variable_value='true' WHERE variable_name='mysql-forward_autocommit';
Query OK, 1 row affected (0.00 sec)

mysql> UPDATE global_variables SET variable_value='true' WHERE variable_name='mysql-autocommit_false_is_transaction';
Query OK, 1 row affected (0.01 sec)

mysql>  LOAD MYSQL VARIABLES TO RUNTIME;
Query OK, 0 rows affected (0.00 sec)

mysql> SAVE MYSQL VARIABLES TO DISK;
Query OK, 98 rows affected (0.12 sec)

Then we try again and it shows success
[root@doris01 ~]# mysql -udoris -pP@ssword1! -P6033 -h192.168.9.211  -e "show databases;"
Warning: Using a password on the command line interface can be insecure.
+--------------------+
| Database           |
+--------------------+
| doris_audit_db     |
| information_schema |
| retail             |
+--------------------+
```

OK, that’s the end, you can use Mysql client, JDBC, etc. to connect to ProxySQL to operate your doris.

Nginx TCP reverse proxy method

Overview

Nginx can implement load balancing of HTTP and HTTPS protocols, as well as load balancing of TCP protocol. So, the question is, can the load balancing of the Apache Doris database be achieved through Nginx? The answer is: yes. Next, let’s discuss how to use Nginx to achieve load balancing of Apache Doris.

Environmental preparation

Note: Using Nginx to achieve load balancing of Apache Doris database, the premise is to build an Apache Doris environment. The IP and port of Apache Doris FE are as follows. Here I use one FE to demonstrate, multiple FEs only You need to add multiple FE IP addresses and ports in the configuration

The Apache Doris and port to access MySQL through Nginx are shown below.

```
IP: 172.31.7.119
端口: 9030
```

Install dependencies

```
sudo apt-get install build-essential
sudo apt-get install libpcre3 libpcre3-dev
sudo apt-get install zlib1g-dev
sudo apt-get install openssl libssl-dev
```

Install Nginx

```
sudo wget http://nginx.org/download/nginx-1.18.0.tar.gz
sudo tar zxvf nginx-1.18.0.tar.gz
cd nginx-1.18.0
sudo ./configure --prefix=/usr/local/nginx --with-stream --with-http_ssl_module --with-http_gzip_static_module --with-
http_stub_status_module
sudo make && make install
```

Configure reverse proxy

Here is a new configuration file

```
vim /usr/local/nginx/conf/default.conf
```

Then add the following in it

```
events {
worker_connections 1024;
}
stream {
  upstream mysqld {
      hash $remote_addr consistent;
      server 172.31.7.119:9030 weight=1 max_fails=2 fail_timeout=60s;
      ## Note: If there are multiple FEs, just load them here.
  }
  ### Configuration for proxy port, timeout, etc.
  server {
      listen 6030;
      proxy_connect_timeout 300s;
      proxy_timeout 300s;
      proxy_pass mysqld;
```

```
  }
}
```

Start Nginx

Start the specified configuration file

```
cd /usr/local/nginx
/usr/local/nginx/sbin/nginx -c conf.d/default.conf
```

Verify

```
mysql -uroot -P6030 -h172.31.7.119
```

Parameter explanation: > - -u specifies the Doris username > - -p specifies the Doris password, my password here is empty, so there is no > - -h specifies the Nginx proxy server IP > - -P specifies the port

```
mysql -uroot -P6030 -h172.31.7.119
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.1.0 Doris version 0.15.1-rc09-Unknown

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.00 sec)

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------+
| Tables_in_test   |
+------------------+
| dwd_product_live |
+------------------+
1 row in set (0.00 sec)
mysql> desc dwd_product_live;
+-----------------+---------------+------+-------+---------+---------+
| Field           | Type          | Null | Key   | Default | Extra   |
+-----------------+---------------+------+-------+---------+---------+
| dt              | DATE          | Yes  | true  | NULL    |         |
| proId           | BIGINT        | Yes  | true  | NULL    |         |
| authorId        | BIGINT        | Yes  | true  | NULL    |         |
```

```
| roomId          | BIGINT        | Yes  | true  | NULL    |         |
| proTitle        | VARCHAR(1024) | Yes  | false | NULL    | REPLACE |
| proLogo         | VARCHAR(1024) | Yes  | false | NULL    | REPLACE |
| shopId          | BIGINT        | Yes  | false | NULL    | REPLACE |
| shopTitle       | VARCHAR(1024) | Yes  | false | NULL    | REPLACE |
| profrom         | INT           | Yes  | false | NULL    | REPLACE |
| proCategory     | BIGINT        | Yes  | false | NULL    | REPLACE |
| proPrice        | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| couponPrice     | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| livePrice       | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| volume          | BIGINT        | Yes  | false | NULL    | REPLACE |
| addedTime       | BIGINT        | Yes  | false | NULL    | REPLACE |
| offTimeUnix     | BIGINT        | Yes  | false | NULL    | REPLACE |
| offTime         | BIGINT        | Yes  | false | NULL    | REPLACE |
| createTime      | BIGINT        | Yes  | false | NULL    | REPLACE |
| createTimeUnix  | BIGINT        | Yes  | false | NULL    | REPLACE |
| amount          | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| views           | BIGINT        | Yes  | false | NULL    | REPLACE |
| commissionPrice | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| proCostPrice    | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| proCode         | VARCHAR(1024) | Yes  | false | NULL    | REPLACE |
| proStatus       | INT           | Yes  | false | NULL    | REPLACE |
| status          | INT           | Yes  | false | NULL    | REPLACE |
| maxPrice        | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| liveView        | BIGINT        | Yes  | false | NULL    | REPLACE |
| firstCategory   | BIGINT        | Yes  | false | NULL    | REPLACE |
| secondCategory  | BIGINT        | Yes  | false | NULL    | REPLACE |
| thirdCategory   | BIGINT        | Yes  | false | NULL    | REPLACE |
| fourCategory    | BIGINT        | Yes  | false | NULL    | REPLACE |
| minPrice        | DECIMAL(18,2) | Yes  | false | NULL    | REPLACE |
| liveVolume      | BIGINT        | Yes  | false | NULL    | REPLACE |
| liveClick       | BIGINT        | Yes  | false | NULL    | REPLACE |
| extensionId     | VARCHAR(128)  | Yes  | false | NULL    | REPLACE |
| beginTime       | BIGINT        | Yes  | false | NULL    | REPLACE |
| roomTitle       | TEXT          | Yes  | false | NULL    | REPLACE |
| beginTimeUnix   | BIGINT        | Yes  | false | NULL    | REPLACE |
| nickname        | TEXT          | Yes  | false | NULL    | REPLACE |
+-----------------+---------------+------+-------+---------+---------+
40 rows in set (0.06 sec)
```

Haproxy way

HAProxy is a free and open source software written in C language that provides high availability, load balancing, and application proxy based on TCP and HTTP.

Install

1. Download HAProxy

   download link: https://src.fedoraproject.org/repo/pkgs/haproxy/

2. Unzip

   ```
   tar -zxvf haproxy-2.6.15.tar.gz -C /opt/
   mv haproxy-2.6.15 haproxy
   ```

3. Compile

Enter the haproxy directory

```
yum install gcc gcc-c++ -y

make TARGET=linux-glibc PREFIX=/usr/local/haproxy

make install PREFIX=/usr/local/haproxy
```

Configuration

1. Configure the haproxy.conf file

   vim /etc/rsyslog.d/haproxy.conf

   ```
   $ModLoad imudp
   $UDPServerRun 514
   local0.* /usr/local/haproxy/logs/haproxy.log
   &~
   ```

2. Enable remote logging

   vim /etc/sysconfig/rsyslog

   `SYSLOGD_OPTIONS="-c 2 -r -m 0"`

   Parameter analysis:

   - -c 2 Use compatibility mode, default is -c 5. -r turns on remote logging
   - -m 0 mark timestamp. The unit is minutes. When it is 0, it means the function is disabled.

3. Make changes effective

   `systemctl restart rsyslog`

4. Edit load balancing file

   vim /usr/local/haproxy/haproxy.cfg

   ```
   #
   # haproxy is deployed on 172.16.0.3, this machine, and is used to proxy 172.16.0.8, 172.16.0.6, 172.16.0.4, the three
   machines where fe is deployed.
   #

   global
   maxconn         2000
   ulimit-n        40075
   log             127.0.0.1 local0 info
   uid             200
   gid             200
   chroot          /var/empty
   daemon
   group           haproxy
   user            haproxy


   defaults
   ```

```
# Global log configuration
log global
mode http
retries 3          # health examination. If the connection fails three times, the server is considered unavailable, mainly
through the subsequent check.
option redispatch  # Redirect to other healthy servers after service becomes unavailable
# Timeout configuration
timeout connect 5000
timeout client 5000
timeout server 5000
timeout check 2000

frontend agent-front
bind *:9030             # Translation port on proxy machine
mode tcp
default_backend forward-fe

backend forward-fe
mode tcp
balance roundrobin
server fe-1 172.16.0.8:9030 weight 1 check inter 3000 rise 2 fall 3
server fe-2 172.16.0.4:9030 weight 1 check inter 3000 rise 2 fall 3
server fe-3 172.16.0.6:9030 weight 1 check inter 3000 rise 2 fall 3

listen http_front              # haproxy client page
bind *:8888                    # IP address of HAProxy WEB
mode http
log 127.0.0.1 local0 err
option httplog
stats uri /haproxy             # The url of the customized page (that is, the address when accessing is:
172.16.0.3:8888/haproxy)
stats auth admin:admin         # Control panel account password Account: admin
stats refresh 10s
stats enable
```

Start up

1. Start service

```
/opt/haproxy/haproxy -f /usr/local/haproxy/haproxy.cfg
```

Time Zone

Time Zone

Doris supports custom time zone settings

Basic concepts

The following two time zone related parameters exist within Doris:

- `system_time_zone` : When the server starts up, it will be set automatically according to the time zone set by the machine, and cannot be modified after it is set.
- `time_zone` : The current time zone of the cluster. This variable will be set to the same as `system_time_zone` when the cluster starts and will not be changed again unless manually changed by the user.

Specific operations

1. `SHOW VARIABLES LIKE '% time_zone%'`

   View the current time zone related configuration

2. `SET [global] time_zone = 'Asia/Shanghai'`

   This command sets the time zone at the session level. If the `global` keyword is used, Doris FE persists the parameter and it takes effect for all new sessions afterwards.

Data source

The time zone data contains the name of the time zone, the corresponding time offset, and the change of daylight saving time. On the machine where the BE is located, the source of the data is the directory returned by command `TZDIR`. If was not supported, the directory `/usr/share/zoneinfo`.

Impact of time zone

1. functions

Includes values displayed by time functions such as `NOW()` or `CURTIME()`, and also time values in `show load`, `show backends`.

However, it does not affect the less than value of the time-type partitioned columns in `create table`, nor does it affect the display of values stored as `date/datetime` types.

Functions affected by time zone:

- `FROM_UNIXTIME`: Given a UTC timestamp, return its date and time in the time zone specified by Doris session `time_zone`. For example, when `time_zone` is `CST`, `FROM_UNIXTIME(0)` returns `1970-01-01 08: 00:00`.

- `UNIX_TIMESTAMP`: Given a date and time, return its UTC timestamp in the time zone specified by Doris session `time_zone`, such as when `time_zone` is `CST` `UNIX_TIMESTAMP('1970-01-01 08:00:00 ')` returns `0`.
- `CURTIME`: Returns the time in the time zone specified by the current Doris session `time_zone`.
- `NOW`: Returns the date and time of the current Doris session `time_zone` specified time zone.
- `CONVERT_TZ`: Convert a datetime from one specified time zone to another.

2. Values of time types

For `DATE` and `DATETIME` types, we support time zone conversion when importing data.

- If the data has a time zone, such as "2020-12-12 12:12:12+08:00" with the current Doris `time_zone = +00:00`, then the data is imported into Doris and the actual value is "2020-12-12 04:12:12".
- If the data does not contain a time zone, such as "2020-12-12 12:12:12", the time is considered to be an absolute time and no conversion occurs.

3. Daylight Saving Time

Daylight Saving Time is essentially the actual time offset of a named time zone, which changes on certain dates.

For example, the `America/Los_Angeles` time zone contains a Daylight Saving Time adjustment that begins and ends approximately in March and November of each year. That is, the `America/Los_Angeles` actual time zone offset changes from `-08:00` to `-07:00` at the start of Daylight Savings Time in March, and from `-07:00` to `-08:00` at the end of Daylight Savings Time in November. If you do not want Daylight Saving Time to be turned on, set `time_zone` to `-08:00` instead of `America/Los_Angeles`.

Usage

Time zone values can be given in a variety of formats. The following standard formats are well supported in Doris:

1. standard named time zone formats, such as "Asia/Shanghai", "America/Los_Angeles". This format is derived from time zone data on this machine. "Etc/GMT+3" etc. also belongs to this category.
2. standard offset formats, such as "+02:30", "-10:00".(special offsets such as "+12:03" are not supported)
3. abbreviated time zone formats, currently only support:
    1. "GMT", "UTC", equivalent to "+00:00" time zone
    2. "CST", which is equivalent to the "Asia/Shanghai" time zone
4. single letter Z, for Zulu time zone, equivalent to "+00:00" time zone

Besides, all parsing of alphabet is case insensitive.

Note: Some other formats are currently supported in some imports in Doris due to different implementations. **Production environments should not rely on these formats that are not listed here, and their behaviour**

may change at any time, so keep an eye on the relevant changelog for version updates.

Best Practices

Time Zone Sensitive Data

The time zone issue involves three main influences:

1. session variable `time_zone` – cluster timezone
2. header `timezone` specified during import(Stream Load, Broker Load etc.) – importing timezone
3. timezone type literal “+08:00” in “2023-12-12 08:00:00+08:00” – data timezone

We can understand it as follows:

Doris is currently compatible with importing data in various time zones into Doris. Since Doris’s own `DATETIME` and other time types do not contain time zone information, and the data will not change with time zone changes after being imported, when time data is imported into Doris, it can be divided into the following two categories:

1. Absolute time

   Absolute time means that the data scene it is associated with has nothing to do with time zones. This type of data should be imported without any time zone suffix and will be stored as-is.

2. Time in a specific time zone

   The time in a specific time zone means that the data scenario it is associated with is related to the time zone. For this type of data, it should be imported with a specific time zone suffix. When imported, they will be converted to the Doris cluster `time_zone` time zone or the header `timezone` specified in Stream Load/Broker Load.

   This type of data is converted to absolute time storage in the time zone specified during import after import, so subsequent imports and queries should maintain this time zone to avoid confusion in the meaning of the data.

- For the Insert statement, we can illustrate it through the following example:

```
Doris > select @@time_zone;
+---------------+
| @@time_zone   |
+---------------+
| Asia/Shanghai |
+---------------+

Doris > insert into dt values('2020-12-12 12:12:12+02:00'); --- The imported data specifies a time zone of +02:00

Doris > select * from dt;
+---------------------+
| dt                  |
+---------------------+
| 2020-12-12 18:12:12 | --- Is converted to the Doris cluster time zone Asia/Shanghai, subsequent imports and queries
should maintain this time zone.
+---------------------+

Doris > set time_zone = 'America/Los_Angeles';
```

```
Doris > select * from dt;
+---------------------+
| dt                  |
+---------------------+
| 2020-12-12 18:12:12 | --- If time_zone is modified, the time value will not change accordingly, and its meaning during
query will be confused.
+---------------------+
```

- For import methods such as Stream Load and Broker Load, we can achieve this by specifying header `timezone`. For example, for Stream Load, we can illustrate it through the following example:

```
cat dt.csv
2020-12-12 12:12:12+02:00

curl --location-trusted -u root: \
 -H "Expect:100-continue" \
 -H "strict_mode: true" \
 -H "timezone: Asia/Shanghai" \
 -T dt.csv -XPUT \
 http://127.0.0.1:8030/api/test/dt/_stream_load
```

```
Doris > select @@time_zone;
+---------------+
| @@time_zone   |
+---------------+
| Asia/Shanghai |
+---------------+

Doris > select * from dt;
+---------------------+
| dt                  |
+---------------------+
| 2020-12-12 18:12:12 | --- Is converted to the Doris cluster time zone Asia/Shanghai, subsequent imports and queries
should maintain this time zone.
+---------------------+
```

  - In import methods such as Stream Load and Broker Load, the header `timezone` will overwrite the Doris cluster `time_zone`, so it should be consistent during import.
  - In import methods such as Stream Load and Broker Load, the header `timezone` will affect the functions used in import conversion.
  - If the header `timezone` is not specified when importing, the East Eighth Zone will be used by default.

To sum up, the best practice for dealing with time zone issues is:

> **Best Practices**
>
> 1. Confirm the time zone represented by the cluster and set `time_zone` before use, and do not change it after that.
> 2. Set header `timezone` to be consistent with cluster `time_zone` when importing.

3. For absolute time, import without time zone suffix; for time with time zone, import with specific time zone suffix, and it will be converted to Doris `time_zone` time zone after import.

Daylight Saving Time

The start and end times for Daylight Saving Time are taken from the current time zone data source and may not necessarily correspond exactly to the actual officially recognised times for the current year's time zone location. This data is maintained by ICANN. If you need to ensure that Daylight Saving Time behaves as specified for the current year, please make sure that data source selected by Doris is the latest ICANN published time zone data. See below for download access.

Information Update

Real-world time zone and daylight saving time data may change from time to time for a variety of reasons, and IANA periodically records these changes and updates the corresponding time zone files. If you want the time zone information in Doris to be up to date with the latest IANA data, do one of the followings:

1. Use the Package Manager to update

Depending on the package manager used by the current operating system, you can update the time zone data directly using the corresponding command:

```
# yum
> sudo yum update tzdata
# apt
> sudo apt update tzdata
```

The data updated in this way is located under the system `$TZDIR` (typically `usr/share/zoneinfo`).

2. pull the IANA time zone database manually (recommended)

Most Linux distributions have a package manager where tzdata is not synchronised in a timely manner. If the accuracy of the time zone data is important, you can pull the data published by IANA on a regular basis:

```
wget https://www.iana.org/time-zones/repository/tzdb-latest.tar.lz
```

Then generate the specific zoneinfo data according the README file in the extracted folder. The generated data should be copied to override `$TZDIR` folder.

Please note that all the above operations **must** be restarted on the corresponding BE to take effect after they are done on the BE machine.

Extended Reading

- List of tz database time zones
- IANA Time Zone Database
- The tz-announce Archives

FQDN

FQDN

This article introduces how to enable the use of Apache Doris based on FQDN (Fully Qualified Domain Name). FQDN is the complete domain name of a specific computer or host on the Internet.

After Doris supports FQDN, communication between nodes is entirely based on FQDN. When adding various types of nodes, the FQDN should be directly specified. For example, the command to add a BE node is' ALTER SYSTEM ADD BACKEND "be_host:heartbeat_service_port",

'be_host' was previously the IP address of the BE node. After starting the FQDN, be_host should specify the FQDN of the BE node.

Preconditions

1. fe.conf file set `enable_fqdn_mode = true`.
2. All machines in the cluster must be configured with a host name.
3. The IP address and FQDN corresponding to other machines in the cluster must be specified in the '/etc/hosts' file for each machine in the cluster.
4. /The etc/hosts file cannot have duplicate IP addresses.

Best Practices

Enable FQDN for new cluster

1. Prepare machines, for example, if you want to deploy a cluster of 3FE 3BE, you can prepare 6 machines.
2. Each machine returns unique results when executing 'host'. Assuming that the execution results of six machines are fe1, fe2, fe3, be1, be2, and be3, respectively.
3. Configure the real IPs corresponding to 6 FQDNs in the '/etc/hosts' of 6 machines, for example:

```
172.22.0.1 fe1
172.22.0.2 fe2
172.22.0.3 fe3
172.22.0.4 be1
172.22.0.5 be2
172.22.0.6 be3
```

4. Verification: It can 'ping fe2' on FE1, and can resolve the correct IP address and ping it, indicating that the network environment is available.
5. fe.conf settings for each FE node `enable_ fqdn_ mode = true`.
6. Refer toStandard deployment

7. Select several machines to deploy broker on six machines as needed, and execute `ALTER SYSTEM ADD BROKER broker_name "fe1:8000","be1:8000",...;`.

Deployment of Doris for K8S

After an unexpected restart of the Pod, K8s cannot guarantee that the Pod's IP will not change, but it can ensure that the domain name remains unchanged. Based on this feature, when Doris enables FQDN, it can ensure that the Pod can still provide services normally after an unexpected restart.

Please refer to the method for deploying Doris in K8sKubernetes Deployment

Server change IP

After deploying the cluster according to 'Enable FQDN for new cluster', if you want to change the IP of the machine, whether it is switching network cards or replacing the machine, you only need to change the '/etc/hosts' of each machine.

Enable FQDN for old cluster

Precondition: The current program supports the syntax 'Alter SYSTEM MODIFY FRONTEND':'HOSTNAME'', If not, upgrade to a version that supports the syntax

> Note that.
>
> At least three followers are required to perform the following operations, otherwise the cluster may not start properly

Next, follow the steps below:

1. Perform the following operations on the Follower and Observer nodes one by one (and finally on the Master node):
    1. Stop the node.
    2. Check if the node has stopped. Execute 'show frontends' through the MySQL client to view the Alive status of the FE node until it becomes false
    3. set FQDN for node: `ALTER SYSTEM MODIFY FRONTEND "<fe_ip>:<edit_log_port>" HOSTNAME "<fe_hostname>"`(After stopping the master, a new master node will be selected and used to execute SQL statements)
    4. Modify node configuration. Modify the 'conf/fe.conf' file in the FE root directory and add the configuration: `enable_fqdn_mode = true`. If the newly stopped node cannot be started properly after adding the configuration to the corresponding fe.config, please add the configuration 'enable_fqdn_mode=true' to all fe.config before starting the just stopped fe node.
    5. Start the node.
2. Enabling FQDN for BE nodes only requires executing the following commands through MySQL, and there is no need to restart BE.

`ALTER SYSTEM MODIFY BACKEND "<backend_ip>:<HeartbeatPort>" HOSTNAME "<be_hostname>"`,If you don't know the number of HeartbeatPort, please use the command `show backends` to help find this port.

Common problem

- Configuration item enable_ fqdn_ Can the mode be changed freely?

  Cannot be changed arbitrarily. To change this configuration, follow the 'Enable FQDN for old cluster' procedure.

Data Backup

Data Backup

Doris supports backing up the current data in the form of files to the remote storage system like S3 and HDFS. Afterwards, you can restore data from the remote storage system to any Doris cluster through the restore command. Through this function, Doris can support periodic snapshot backup of data. You can also use this function to migrate data between different clusters.

This feature requires Doris version 0.8.2+

A brief explanation of the principle

The backup operation is to upload the data of the specified table or partition directly to the remote warehouse for storage in the form of files stored by Doris. When a user submits a Backup request, the system will perform the following operations:

1. Snapshot and snapshot upload

   The snapshot phase takes a snapshot of the specified table or partition data file. After that, backups are all operations on snapshots. After the snapshot, changes, imports, etc. to the table no longer affect the results of the backup. Snapshots only generate a hard link to the current data file, which takes very little time. After the snapshot is completed, the snapshot files will be uploaded one by one. Snapshot uploads are done concurrently by each Backend.

2. Metadata preparation and upload

   After the data file snapshot upload is complete, Frontend will first write the corresponding metadata to a local file, and then upload the local metadata file to the remote warehouse through the broker. Completing the final backup job

3. Dynamic Partition Table Description

   If the table is a dynamic partition table, the dynamic partition attribute will be automatically disabled after backup. When restoring, you need to manually enable the dynamic partition attribute of the table. The command is as follows:

```
ALTER TABLE tbl1 SET ("dynamic_partition.enable"="true")
```

4. Backup and Restore operation will NOT keep the `colocate_with` property of a table.

Start Backup

1. Create a hdfs remote warehouse example_repo (S3 skips step 1):

```
CREATE REPOSITORY `example_repo`
WITH HDFS
```

```
ON LOCATION "hdfs://hadoop-name-node:54310/path/to/repo/"
PROPERTIES
(
"fs.defaultFS"="hdfs://hdfs_host:port",
"hadoop.username" = "hadoop"
);
```

2. Create a remote repository for s3 : s3_repo (HDFS skips step 2)

```
CREATE REPOSITORY `s3_repo`
WITH S3
ON LOCATION "s3://bucket_name/test"
PROPERTIES
(
    "AWS_ENDPOINT" = "http://xxxx.xxxx.com",
    "AWS_ACCESS_KEY" = "xxxx",
    "AWS_SECRET_KEY" = "xxx",
    "AWS_REGION" = "xxx"
);
```

> Note that.
>
> ON LOCATION is followed by Bucket Name here

3. Full backup of table example_tbl under example_db to warehouse example_repo:

```
BACKUP SNAPSHOT example_db.snapshot_label1
TO example_repo
ON (example_tbl)
PROPERTIES ("type" = "full");
```

4. Under the full backup example_db, the p1, p2 partitions of the table example_tbl, and the table example_tbl2 to the warehouse example_repo:

```
BACKUP SNAPSHOT example_db.snapshot_label2
TO example_repo
ON
(
   example_tbl PARTITION (p1,p2),
   example_tbl2
);
```

5. View the execution of the most recent backup job:

```
mysql> show BACKUP\G;
*************************** 1. row ***************************
               JobId: 17891847
        SnapshotName: snapshot_label1
              DbName: example_db
               State: FINISHED
          BackupObjs: [default_cluster:example_db.example_tbl]
          CreateTime: 2022-04-08 15:52:29
SnapshotFinishedTime: 2022-04-08 15:52:32
  UploadFinishedTime: 2022-04-08 15:52:38
```

```
        FinishedTime: 2022-04-08 15:52:44
     UnfinishedTasks:
            Progress:
           TaskErrMsg:
               Status: [OK]
              Timeout: 86400
1 row in set (0.01 sec)
```

6. View existing backups in remote repositories:

```
mysql> SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "snapshot_label1";
+-----------------+---------------------+--------+
| Snapshot        | Timestamp           | Status |
+-----------------+---------------------+--------+
| snapshot_label1 | 2022-04-08-15-52-29 | OK     |
+-----------------+---------------------+--------+
1 row in set (0.15 sec)
```

For the detailed usage of BACKUP, please refer to here.

Best Practices

Backup

Currently, we support full backup with the smallest partition (Partition) granularity (incremental backup may be supported in future versions). If you need to back up data regularly, you first need to plan the partitioning and bucketing of the table reasonably when building the table, such as partitioning by time. Then, in the subsequent running process, regular data backups are performed according to the partition granularity.

Data Migration

Users can back up the data to the remote warehouse first, and then restore the data to another cluster through the remote warehouse to complete the data migration. Because data backup is done in the form of snapshots, new imported data after the snapshot phase of the backup job will not be backed up. Therefore, after the snapshot is completed and until the recovery job is completed, the data imported on the original cluster needs to be imported again on the new cluster.

It is recommended to import the new and old clusters in parallel for a period of time after the migration is complete. After verifying the correctness of data and services, migrate services to a new cluster.

Highlights

1. Operations related to backup and recovery are currently only allowed to be performed by users with ADMIN privileges.
2. Within a database, only one backup or restore job is allowed to be executed.
3. Both backup and recovery support operations at the minimum partition (Partition) level. When the amount of data in the table is large, it is recommended to perform operations by partition to reduce the cost of failed retry.
4. Because of the backup and restore operations, the operations are the actual data files. Therefore, when a table has too many shards, or a shard has too many small versions, it may take a long time to backup or restore even if the total amount of data is small. Users can use `SHOW PARTITIONS FROM table_name;` and

`SHOW TABLETS FROM table_name;` to view the number of shards in each partition and the number of file versions in each shard to estimate job execution time. The number of files has a great impact on the execution time of the job. Therefore, it is recommended to plan partitions and buckets reasonably when creating tables to avoid excessive sharding.

5. When checking job status via `SHOW BACKUP` or `SHOW RESTORE` command. It is possible to see error messages in the `TaskErrMsg` column. But as long as the `State` column is not `CANCELLED`, the job is still continuing. These tasks may retry successfully. Of course, some Task errors will also directly cause the job to fail.
6. If the recovery job is an overwrite operation (specifying the recovery data to an existing table or partition), then from the `COMMIT` phase of the recovery job, the overwritten data on the current cluster may no longer be restored. If the restore job fails or is canceled at this time, the previous data may be damaged and inaccessible. In this case, the only way to do it is to perform the recovery operation again and wait for the job to complete. Therefore, we recommend that if unnecessary, try not to restore data by overwriting unless it is confirmed that the current data is no longer used.

Related Commands

1. The commands related to the backup and restore function are as follows. For the following commands, you can use `help cmd;` to view detailed help after connecting to Doris through mysql-client.

    1. CREATE REPOSITORY

        Create a remote repository path for backup or restore. Please refer to Create Repository Reference.

    2. BACKUP

        Perform a backup operation. Please refer to Backup Reference.

    3. SHOW BACKUP

        View the execution of the most recent backup job. Please refer to Show Backup Reference。

    4. SHOW SNAPSHOT

        View existing backups in the remote repository. Please refer to Show Snapshot Reference.

    5. CANCEL BACKUP

        Cancel the currently executing backup job. Please refer to [Cancel Backup Reference] (../../sql-manual/sql-reference/Data-Definition-Statements/Backup-and-Restore/CANCEL-BACKUP.md).

    6. DROP REPOSITORY

        Delete the created remote repository. Deleting a warehouse only deletes the mapping of the warehouse in Doris, and does not delete the actual warehouse data. Please refer to [Drop Repository Reference] (../../sql-manual/sql-reference/Data-Definition-Statements/Backup-and-Restore/DROP-REPOSITORY.md).

More Help

For more detailed syntax and best practices used by BACKUP, please refer to the BACKUP command manual, You can also type `HELP BACKUP` on the MySql client command line for more help.

Data Restore

Data Recovery

Doris supports backing up the current data in the form of files to the remote storage system through the broker. Afterwards, you can restore data from the remote storage system to any Doris cluster through the restore command. Through this function, Doris can support periodic snapshot backup of data. You can also use this function to migrate data between different clusters.

This feature requires Doris version 0.8.2+

To use this function, you need to deploy the broker corresponding to the remote storage. Such as BOS, HDFS, etc. You can view the currently deployed broker through `SHOW BROKER;`.

Brief principle description

The restore operation needs to specify an existing backup in the remote warehouse, and then restore the content of the backup to the local cluster. When the user submits the Restore request, the system will perform the following operations:

1. Create the corresponding metadata locally

   This step will first create and restore the corresponding table partition and other structures in the local cluster. After creation, the table is visible, but not accessible.

2. Local snapshot

   This step is to take a snapshot of the table created in the previous step. This is actually an empty snapshot (because the table just created has no data), and its purpose is to generate the corresponding snapshot directory on the Backend for later receiving the snapshot file downloaded from the remote warehouse.

3. Download snapshot

   The snapshot files in the remote warehouse will be downloaded to the corresponding snapshot directory generated in the previous step. This step is done concurrently by each Backend.

4. Effective snapshot

   After the snapshot download is complete, we need to map each snapshot to the metadata of the current local table. These snapshots are then reloaded to take effect, completing the final recovery job.

Start Restore

1. Restore the table backup_tbl in backup snapshot_1 from example_repo to database example_db1, the time version is “2018-05-04-16-45-08”. Revert to 1 copy:

```
RESTORE SNAPSHOT example_db1.`snapshot_1`
FROM `example_repo`
ON ( `backup_tbl` )
PROPERTIES
(
    "backup_timestamp"="2022-04-08-15-52-29",
    "replication_num" = "1"
);
```

2. Restore partitions p1 and p2 of table backup_tbl in backup snapshot_2 from example_repo, and table backup_tbl2 to database example_db1, and rename it to new_tbl with time version “2018-05-04-17-11-01”. The default reverts to 3 replicas:

```
RESTORE SNAPSHOT example_db1.`snapshot_2`
FROM `example_repo`
ON
(
    `backup_tbl` PARTITION (`p1`, `p2`),
    `backup_tbl2` AS `new_tbl`
)
PROPERTIES
(
    "backup_timestamp"="2022-04-08-15-55-43"
);
```

3. View the execution of the restore job:

```
mysql> SHOW RESTORE\G;
*************************** 1. row ***************************
               JobId: 17891851
               Label: snapshot_label1
           Timestamp: 2022-04-08-15-52-29
              DbName: default_cluster:example_db1
               State: FINISHED
           AllowLoad: false
     ReplicationNum: 3
        RestoreObjs: {
  "name": "snapshot_label1",
  "database": "example_db",
  "backup_time": 1649404349050,
  "content": "ALL",
  "olap_table_list": [
    {
      "name": "backup_tbl",
      "partition_names": [
        "p1",
        "p2"
      ]
    }
  ],
  "view_list": [],
  "odbc_table_list": [],
  "odbc_resource_list": []
}
          CreateTime: 2022-04-08 15:59:01
    MetaPreparedTime: 2022-04-08 15:59:02
SnapshotFinishedTime: 2022-04-08 15:59:05
DownloadFinishedTime: 2022-04-08 15:59:12
        FinishedTime: 2022-04-08 15:59:18
     UnfinishedTasks:
```

```
            Progress:
          TaskErrMsg:
              Status: [OK]
             Timeout: 86400
1 row in set (0.01 sec)
```

For detailed usage of RESTORE, please refer to here.

Related Commands

The commands related to the backup and restore function are as follows. For the following commands, you can use `help cmd;` to view detailed help after connecting to Doris through mysql-client.

1. CREATE REPOSITORY

   Create a remote repository path for backup or restore. This command needs to use the Broker process to access the remote storage. Different brokers need to provide different parameters. For details, please refer to Broker documentation, or you can directly back up to support through the S3 protocol For the remote storage of AWS S3 protocol, directly back up to HDFS, please refer to Create Remote Warehouse Documentation

2. RESTORE

   Perform a restore operation.

3. SHOW RESTORE

   View the execution of the most recent restore job, including:

   - JobId: The id of the current recovery job.
   - Label: The name (Label) of the backup in the warehouse specified by the user.
   - Timestamp: The timestamp of the backup in the user-specified repository.
   - DbName: Database corresponding to the restore job.
   - State: The current stage of the recovery job:
     - PENDING: The initial status of the job.
     - SNAPSHOTING: The snapshot operation of the newly created table is in progress.
     - DOWNLOAD: Sending download snapshot task.
     - DOWNLOADING: Snapshot is downloading.
     - COMMIT: Prepare the downloaded snapshot to take effect.
     - COMMITTING: Validating downloaded snapshots.
     - FINISHED: Recovery is complete.
     - CANCELLED: Recovery failed or was canceled.
   - AllowLoad: Whether to allow import during restore.
   - ReplicationNum: Restores the specified number of replicas.
   - RestoreObjs: List of tables and partitions involved in this restore.
   - CreateTime: Job creation time.
   - MetaPreparedTime: Local metadata generation completion time.
   - SnapshotFinishedTime: The local snapshot completion time.
   - DownloadFinishedTime: The time when the remote snapshot download is completed.
   - FinishedTime: The completion time of this job.

- UnfinishedTasks: During `SNAPSHOTTING`, `DOWNLOADING`, `COMMITTING` and other stages, there will be multiple subtasks going on at the same time. The current stage shown here is the task id of the unfinished subtasks.
- TaskErrMsg: If there is an error in the execution of a subtask, the error message of the corresponding subtask will be displayed here.
- Status: Used to record some status information that may appear during the entire job process.
- Timeout: The timeout period of the job, in seconds.

4. CANCEL RESTORE

   Cancel the currently executing restore job.

5. DROP REPOSITORY

   Delete the created remote repository. Deleting a warehouse only deletes the mapping of the warehouse in Doris, and does not delete the actual warehouse data.

Common mistakes

1. Restore Report An Error:[20181: invalid md5 of downloaded file: /data/doris.HDD/snapshot/20220607095111.862.86400/19962/668322732/19962.hdr, expected: f05b63cca5533ea0466f62a9897289b5, get: d41d8cd98f00b204e9800998ecf8427e]

   If the number of copies of the table backed up and restored is inconsistent, you need to specify the number of copies when executing the restore command. For specific commands, please refer to RESTORE command manual

2. Restore Report An Error:[COMMON_ERROR, msg: Could not set meta version to 97 since it is lower than minimum required version 100]

   Backup and restore are not caused by the same version, use the specified meta_version to read the metadata of the previous backup. Note that this parameter is used as a temporary solution and is only used to restore the data backed up by the old version of Doris. The latest version of the backup data already contains the meta version, so there is no need to specify it. For the specific solution to the above error, specify meta_version = 100. For specific commands, please refer to RESTORE command manual

More Help

For more detailed syntax and best practices used by RESTORE, please refer to the RESTORE command manual, You can also type `HELP RESTORE` on the MySql client command line for more help.

CCR (Cross Cluster Replication)

Cross Cluster Replication (CCR)

Overview

Cross Cluster Replication (CCR) enables the synchronization of data changes from a source cluster to a target cluster at the database/table level. This feature can be used to ensure data availability for online services, isolate offline and online workloads, and build multiple data centers across various sites.

CCR is applicable to the following scenarios:

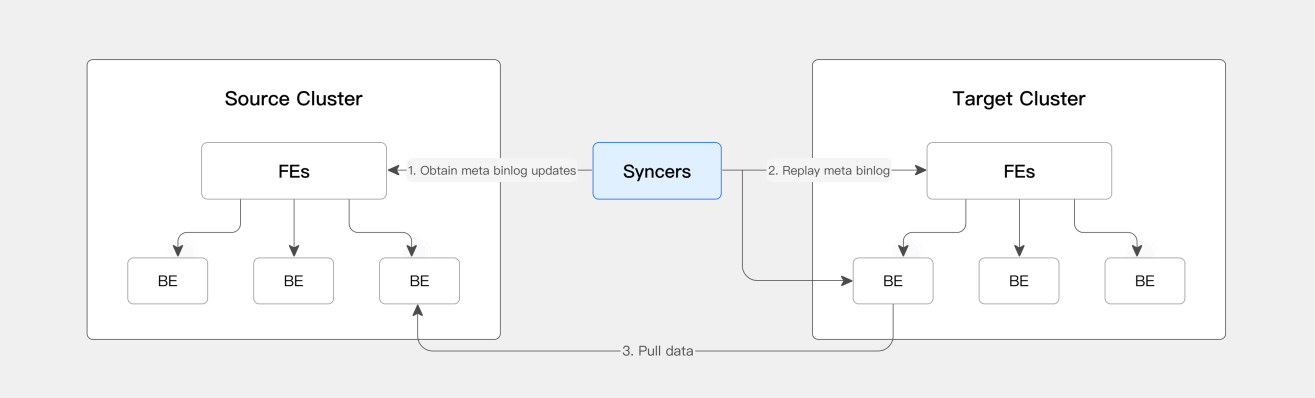
- Disaster recovery: This involves backing up enterprise data to another cluster and data center. In the event of a sudden incident causing business interruption or data loss, companies can recover data from the backup or quickly switch to the backup cluster. Disaster recovery is typically a must-have feature in use cases with high SLA requirements, such as those in finance, healthcare, and e-commerce.
- Read/write separation: This is to isolate querying and writing operations to reduce their mutual impact and improve resource utilization. For example, in cases of high writing pressure or high concurrency, read/write separation can distribute read and write operations to read-only and write-only database instances in various regions. This helps ensure high database performance and stability.
- Data transfer between headquarters and branch offices: In order to have unified data control and analysis within a corporation, the headquarters usually requires timely data synchronization from branch offices located in different regions. This avoids management confusion and wrong decision-making based on inconsistent data.
- Isolated upgrades: During system cluster upgrades, there might be a need to roll back to a previous version. Many traditional upgrade methods do not allow rolling back due to incompatible metadata. CCR in Doris can address this issue by building a standby cluster for upgrade and conducting dual-running verification. Users can ungrade the clusters one by one. CCR is not dependent on specific versions, making version rollback feasible.

Design

Concepts

- Source cluster: the cluster where business data is written and originates from, requiring Doris version 2.0
- Target cluster: the destination cluster for cross cluster replication, requiring version 2.0
- Binlog: the change log of the source cluster, including schema and data changes
- Syncer: a lightweight process

Architecture description



ccr-architecture-description

CCR relies on a lightweight process called syncer. Syncers retrieve binlogs from the source cluster, directly apply the metadata to the target cluster, and notify the target cluster to pull data from the source cluster. CCR allows both full and incremental data migration.

Usage

The usage of CCR is straightforward. Simply start the syncer service and send a command, and the syncers will take care of the rest.

1. Deploy the source Doris cluster.
2. Deploy the target Doris cluster.
3. Both the source and target clusters need to enable binlog. Configure the following information in the fe.conf and be.conf files of the source and target clusters:

```
enable_feature_binlog=true
```

4. Deploy syncers

Build CCR syncer

```
git clone https://github.com/selectdb/ccr-syncer
cd ccr-syncer
bash build.sh <-j NUM_OF_THREAD> <--output SYNCER_OUTPUT_DIR>
cd SYNCER_OUTPUT_DIR# Contact the Doris community for a free CCR binary package
```

Start and stop syncer

```
# Start
cd bin && sh start_syncer.sh --daemon

# Stop
sh stop_syncer.sh
```

5. Enable binlog in the source cluster.

```
-- If you want to synchronize the entire database, you can execute the following script:
vim shell/enable_db_binlog.sh
Modify host, port, user, password, and db in the source cluster
Or ./enable_db_binlog.sh --host $host --port $port --user $user --password $password --db $db

-- If you want to synchronize a single table, you can execute the following script and enable binlog for the target table:
ALTER TABLE enable_binlog SET ("binlog.enable" = "true");
```

6. Launch a synchronization task to the syncer

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "ccr_test",
    "src": {
      "host": "localhost",
      "port": "9030",
      "thrift_port": "9020",
      "user": "root",
      "password": "",
      "database": "your_db_name",
      "table": "your_table_name"
    },
    "dest": {
      "host": "localhost",
      "port": "9030",
      "thrift_port": "9020",
      "user": "root",
      "password": "",
      "database": "your_db_name",
      "table": "your_table_name"
    }
}' http://127.0.0.1:9190/create_ccr
```

Parameter description:

```
name: name of the CCR synchronization task, should be unique
host, port: host and mysql(jdbc) port for the master FE for the corresponding cluster
user, password: the credentials used by the syncer to initiate transactions, fetch data, etc.
If it is synchronization at the database level, specify your_db_name and leave your_table_name empty
If it is synchronization at the table level, specify both your_db_name and your_table_name
The synchronization task name can only be used once.
```

Operation manual for syncer

Start syncer

Start syncer according to the configurations and save a pid file in the default or specified path. The name of the pid file should follow `host_port.pid`.

Output file structure

The file structure can be seen under the output path after compilation:

```
output_dir
    bin
        ccr_syncer
        enable_db_binlog.sh
```

```
        start_syncer.sh
        stop_syncer.sh
    db
        [ccr.db] # Generated after running with the default configurations.
    log
        [ccr_syncer.log] # Generated after running with the default configurations.
```

The start_syncer.sh in the following text refers to the start_syncer.sh under its corresponding path.

Start options

–daemon

Run syncer in the background, set to false by default.

```
bash bin/start_syncer.sh --daemon
```

–db_type

Syncer can currently use two databases to store its metadata, `sqlite3`(for local storage) and `mysql`(for local or remote storage).

```
bash bin/start_syncer.sh --db_type mysql
```

The default value is sqlite3.

When using MySQL to store metadata, syncer will use `CREATE IF NOT EXISTS`to create a database called `ccr`, where the metadata table related to CCR will be saved.

–db_dir

This option only works when db uses `sqlite3`.

It allows you to specify the name and path of the db file generated by sqlite3.

```
bash bin/start_syncer.sh --db_dir /path/to/ccr.db
```

The default path is `SYNCER_OUTPUT_DIR/db` and the default file name is `ccr.db`.

–db_host & db_port & db_user & db_password

This option only works when db uses `mysql`.

```
bash bin/start_syncer.sh --db_host 127.0.0.1 --db_port 3306 --db_user root --db_password "qwe123456"
```

The default values of db_host and db_port are shown in the example. The default values of db_user and db_password are empty.

–log_dir

Output path of the logs:

```
bash bin/start_syncer.sh --log_dir /path/to/ccr_syncer.log
```

The default path is`SYNCER_OUTPUT_DIR/log` and the default file name is `ccr_syncer.log`.

–log_level

Used to specify the output level of syncer logs.

```
bash bin/start_syncer.sh --log_level info
```

The format of the log is as follows, where the hook will only be printed when `log_level > info`:

```
#        time         level        msg                 hooks
[2023-07-18 16:30:18] TRACE This is trace type. ccrName=xxx line=xxx
[2023-07-18 16:30:18] DEBUG This is debug type. ccrName=xxx line=xxx
[2023-07-18 16:30:18]  INFO This is info type. ccrName=xxx line=xxx
[2023-07-18 16:30:18]  WARN This is warn type. ccrName=xxx line=xxx
[2023-07-18 16:30:18] ERROR This is error type. ccrName=xxx line=xxx
[2023-07-18 16:30:18] FATAL This is fatal type. ccrName=xxx line=xxx
```

Under –daemon, the default value of log_level is `info`.

When running in the foreground, log_level defaults to `trace`, and logs are saved to log_dir using the tee command.

–host && –port

Used to specify the host and port of syncer, where host only plays the role of distinguishing itself in the cluster, which can be understood as the name of syncer, and the name of syncer in the cluster is `host: port`.

```
bash bin/start_syncer.sh --host 127.0.0.1 --port 9190
```

The default value of host is 127.0.0.1, and the default value of port is 9190.

–pid_dir

Used to specify the storage path of the pid file

The pid file is the credentials for closing the syncer. It is used in the stop_syncer.sh script. It saves the corresponding syncer process number. In order to facilitate management of syncer, you can specify the storage path of the pid file.

```
bash bin/start_syncer.sh --pid_dir /path/to/pids
```

The default value is `SYNCER_OUTPUT_DIR/bin`.

Stop syncer

Stop the syncer according to the process number in the pid file under the default or specified path. The name of the pid file should follow `host_port.pid`.

Output file structure

The file structure can be seen under the output path after compilation:

```
output_dir
    bin
        ccr_syncer
        enable_db_binlog.sh
        start_syncer.sh
        stop_syncer.sh
    db
        [ccr.db] # Generated after running with the default configurations.
    log
        [ccr_syncer.log] # Generated after running with the default configurations.
```

The start_syncer.sh in the following text refers to the start_syncer.sh under its corresponding path.

Stop options

Syncers can be stopped in three ways:

1. Stop a single syncer in the directory

Specify the host and port of the syncer to be stopped. Be sure to keep it consistent with the host specified when start_syncer

2. Batch stop the specified syncers in the directory

Specify the names of the pid files to be stopped, wrap the names in "" and separate them with spaces.

3. Stop all syncers in the directory

Follow the default configurations.

–pid_dir

Specify the directory where the pid file is located. The above three stopping methods all depend on the directory where the pid file is located for execution.

```
bash bin/stop_syncer.sh --pid_dir /path/to/pids
```

The effect of the above example is to close the syncers corresponding to all pid files under `/path/to/pids`(**method 3**). `-- pid_dir`can be used in combination with the above three syncer stopping methods.

The default value is `SYNCER_OUTPUT_DIR/bin`.

–host && –port

Stop the syncer corresponding to host: port in the pid_dir path.

```
bash bin/stop_syncer.sh --host 127.0.0.1 --port 9190
```

The default value of host is 127.0.0.1, and the default value of port is empty. That is, specifying the host alone will degrade **method 1** to **method 3**. **Method 1** will only take effect when neither the host nor the port is empty.

–files

Stop the syncer corresponding to the specified pid file name in the pid_dir path.

```
bash bin/stop_syncer.sh --files "127.0.0.1_9190.pid 127.0.0.1_9191.pid"
```

The file names should be wrapped in " " and separated with spaces.

Syncer operations

Template for requests

```
curl -X POST -H "Content-Type: application/json" -d {json_body} http://ccr_syncer_host:ccr_syncer_port/operator
```

json_body: send operation information in JSON format

operator: different operations for syncer

The interface returns JSON. If successful, the “success” field will be true. Conversely, if there is an error, it will be false, and then there will be an `ErrMsgs` field.

```
{"success":true}

or

{"success":false,"error_msg":"job ccr_test not exist"}
```

Operators

- create_ccr

Create CCR tasks

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "ccr_test",
    "src": {
    "host": "localhost",
    "port": "9030",
    "thrift_port": "9020",
    "user": "root",
    "password": "",
    "database": "demo",
    "table": "example_tbl"
    },
    "dest": {
    "host": "localhost",
    "port": "9030",
    "thrift_port": "9020",
    "user": "root",
    "password": "",
    "database": "ccrt",
    "table": "copy"
    }
}' http://127.0.0.1:9190/create_ccr
```

- name: the name of the CCR synchronization task, should be unique
- host, port: correspond to the host and mysql (jdbc) port of the cluster's master
- thrift_port: corresponds to the rpc_port of the FE
- user, password: the credentials used by the syncer to initiate transactions, fetch data, etc.
- database, table:
    - If it is a database-level synchronization, fill in the database name and leave the table name empty.
    - If it is a table-level synchronization, specify both the database name and the table name.
- get_lag

View the synchronization progress.

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "job_name"
}' http://ccr_syncer_host:ccr_syncer_port/get_lag
```

The job_name is the name specified when create_ccr.

- pause

Pause synchronization task.

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "job_name"
}' http://ccr_syncer_host:ccr_syncer_port/pause
```

- resume

Resume synchronization task.

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "job_name"
}' http://ccr_syncer_host:ccr_syncer_port/resume
```

- delete

Delete synchronization task.

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "job_name"
}' http://ccr_syncer_host:ccr_syncer_port/delete
```

- version

View version information.

```
curl http://ccr_syncer_host:ccr_syncer_port/version

# > return
{"version": "2.0.1"}
```

- job status

View job status.

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "job_name"
}' http://ccr_syncer_host:ccr_syncer_port/job_status

{
  "success": true,
  "status": {
    "name": "ccr_db_table_alias",
    "state": "running",
    "progress_state": "TableIncrementalSync"
  }
}
```

- desync job

No sync. Users can swap the source and target clusters.

```
curl -X POST -H "Content-Type: application/json" -d '{
    "name": "job_name"
}' http://ccr_syncer_host:ccr_syncer_port/desync
```

- list_jobs

List all created tasks.

```
curl http://ccr_syncer_host:ccr_syncer_port/list_jobs

{"success":true,"jobs":["ccr_db_table_alias"]}
```

Open binlog for all tables in the database

Output file structure

The file structure can be seen under the output path after compilation:

```
output_dir
    bin
        ccr_syncer
        enable_db_binlog.sh
        start_syncer.sh
        stop_syncer.sh
    db
        [ccr.db] # Generated after running with the default configurations.
    log
        [ccr_syncer.log] # Generated after running with the default configurations.
```

The start_syncer.sh in the following text refers to the start_syncer.sh under its corresponding path.

Usage

```
bash bin/enable_db_binlog.sh -h host -p port -u user -P password -d db
```

High availability of syncer

The high availability of syncers relies on MySQL. If MySQL is used as the backend storage, the syncer can discover other syncers. If one syncer crashes, the others will take over its tasks.

Privilege requirements

1. `select_priv`: read-only privileges for databases and tables
2. `load_priv`: write privileges for databases and tables, including load, insert, delete, etc.
3. `alter_priv`: privilege to modify databases and tables, including renaming databases/tables, adding/deleting/changing columns, adding/deleting partitions, etc.
4. `create_priv`: privilege to create databases, tables, and views
5. `drop_priv`: privilege to drop databases, tables, and views

Admin privileges are required (We are planning on removing this in future versions). This is used to check the `enable binlog config`.

Usage restrictions

Network constraints

- Syncer needs to have connectivity to both the upstream and downstream FEs and BEs.
- The downstream BE should have connectivity to the upstream BE.
- The external IP and Doris internal IP should be the same. In other words, the IP address visible in the output of `show frontends/backends` should be the same IP that can be directly connected to. It should not involve IP forwarding or NAT for direct connections.

ThriftPool constraints

It is recommended to increase the size of the Thrift thread pool to a number greater than the number of buckets involved in a single commit operation.

Version requirements

Minimum required version: V2.0.3

Unsupported operations

- Rename table
- Operations such as table drop-recovery
- Operations related to rename table, replace partition
- Concurrent backup/restore within the same database

Feature

Rate limit

BE-side configuration parameter

```
download_binlog_rate_limit_kbs=1024 # Limits the download speed of Binlog (including Local Snapshot) from the source cluster to 1 MB/s in a single BE node
```

1. The `download_binlog_rate_limit_kbs` parameter is configured on the BE nodes of the source cluster. By setting this parameter, the data pull rate can be effectively limited.

2. The `download_binlog_rate_limit_kbs` parameter primarily controls the speed of data transfer for each single BE node. To calculate the overall cluster rate, one would multiply the parameter value by the number of nodes in the cluster.

IS_BEING_SYNCED

> **Doris v2.0 “is_being_synced” = “true”**

During data synchronization using CCR, replica tables (referred to as target tables) are created in the target cluster for the tables within the synchronization scope of the source cluster (referred to as source tables). However, certain functionalities and attributes need to be disabled or cleared when creating replica tables to ensure the correctness of the synchronization process. For example:

- The source tables may contain information that is not synchronized to the target cluster, such as `storage_policy`, which may cause the creation of the target table to fail or result in abnormal behavior.
- The source tables may have dynamic functionalities, such as dynamic partitioning, which can lead to uncontrolled behavior in the target table and result in inconsistent partitions.

The attributes that need to be cleared during replication are:

- `storage_policy`
- `colocate_with`

The functionalities that need to be disabled during synchronization are:

- Automatic bucketing
- Dynamic partitioning

Implementation

When creating the target table, the syncer controls the addition or deletion of the `is_being_synced` property. In CCR, there are two approaches to creating a target table:

1. During table synchronization, the syncer performs a full copy of the source table using backup/restore to obtain the target table.
2. During database synchronization, for existing tables, the syncer also uses backup/restore to obtain the target table. For incremental tables, the syncer creates the target table using the CreateTableRecord binlog.

Therefore, there are two entry points for inserting the `is_being_synced` property: the restore process during full synchronization and the getDdlStmt during incremental synchronization.

During the restoration process of full synchronization, the syncer initiates a restore operation of the snapshot from the source cluster via RPC. During this process, the `is_being_synced` property is added to the RestoreStmt and takes effect in the final restoreJob, executing the relevant logic for `is_being_synced`.

During incremental synchronization, add the `boolean getDdlForSync` parameter to the getDdlStmt method to differentiate whether it is a controlled transformation to the target table DDL, and execute the relevant logic for isBeingSynced during the creation of the target table.

Regarding the disabling of the functionalities mentioned above:

- Automatic bucketing: Automatic bucketing is enabled when creating a table. It calculates the appropriate number of buckets. This may result in a mismatch in the number of buckets between the source and target tables. Therefore, during synchronization, obtain the number of buckets from the source table, as well as the information about whether the source table is an automatic bucketing table in order to restore the functionality after synchronization. The current recommended approach is to default the autobucket attribute to false when retrieving distribution information. During table restoration, check the `_auto_bucket` attribute to find out if the source table is an automatic bucketing table. If it is, set the target table's autobucket field to true to bypass the calculation of bucket numbers and directly apply the number of buckets from the source table to the target table.
- Dynamic partitioning: This is implemented by adding `olapTable.isBeingSynced()` to the condition for executing add/drop partition operations. This ensures that the target table does not perform periodic add/drop partition operations during synchronization.

Note

The `is_being_synced` property should be fully controlled by the syncer, and users should not modify this property manually unless there are exceptional circumstances.

Recover from Recycle Bin

Data Lifecycle

1. When a user executes the `drop database/table/partition` command, Doris moves the deleted database, table, or partition to the recycle bin. The recover command can be used to restore all data of the deleted database, table, or partition from the recycle bin, making them visible again.
2. When a tablet is deleted on the BE side, its data is moved to the BE recycle bin by default. In case of accidental operations or online bugs that result in some tablets being deleted on the BE, these tablets can be recovered from the BE recycle bin using maintenance tools.

In the above scenarios, the first one pertains to the situation where the database, table, or partition is no longer visible on the FE, but its metadata is still retained in the FE recycle bin. The second scenario pertains to the situation where the database, table, or partition is still visible on the FE, but some BE tablet data has been deleted.

The two recovery methods are described below.

Recover From FE Recycle Bin

In order to avoid disasters caused by misoperation, Doris supports data recovery of accidentally deleted databases/tables/partitions. After dropping table or database, Doris will not physically delete the data immediately, but will keep it in Trash for a period of time (The default is 1 day, which can be configured through the `catalog_trash_expire_second` parameter in fe.conf). The administrator can use the RECOVER command to restore accidentally deleted data.

Note that if it is deleted using `drop force`, it will be deleted directly and cannot be recovered.

Query Catalog Recycle Bin

Query FE catalog recycle bin

```
SHOW CATALOG RECYCLE BIN [ WHERE NAME [ = "name" | LIKE "name_matcher"] ]
```

For more detailed syntax and best practices, please refer to the SHOW-CATALOG-RECYCLE-BIN command manual, You can also type `help SHOW CATALOG RECYCLE BIN` on the MySql client command line for more help.

Start Data Recovery

1.restore the database named example_db

```
RECOVER DATABASE example_db;
```

2.restore the table named example_tbl

```
RECOVER TABLE example_db.example_tbl;
```

3.restore partition named p1 in table example_tbl

```
RECOVER PARTITION p1 FROM example_tbl;
```

For more detailed syntax and best practices used by RECOVER, please refer to the RECOVER command manual, You can also type `HELP RECOVER` on the MySql client command line for more help.

Recover Tablet from BE Recycle Bin

Recover data from BE Recycle Bin

During the user's use of Doris, some valid tablets (including metadata and data) may be deleted due to some misoperations or online bugs. In order to prevent data loss in these abnormal situations, Doris provides a recycle bin mechanism to protect user data. Tablet data deleted by users will not be deleted directly, but will be stored in the recycle bin for a period of time. After a period of time, there will be a regular cleaning mechanism to delete expired data. By default, when the disk space usage does not exceed 81% (BE `config.storage_flood_stage_usage_percent` * 0.9 * 100%), the data in the BE recycle bin is kept for up to 3 days (BE `config.trash_file_expire_time_sec`).

The data in the BE recycle bin includes: tablet data file (.dat), tablet index file (.idx) and tablet metadata file (.hdr). The data will be stored in a path in the following format:

```
/root_path/trash/time_label/tablet_id/schema_hash/
```

- `root_path`: a data root directory corresponding to the BE node.
- `trash`: The directory of the recycle bin.
- `time_label`: Time label, for the uniqueness of the data directory in the recycle bin, while recording the data time, use the time label as a subdirectory.

When a user finds that online data has been deleted by mistake, he needs to recover the deleted tablet from the recycle bin. This tablet data recovery function is needed.

BE provides http interface and `restore_tablet_tool.sh` script to achieve this function, and supports single tablet operation (single mode) and batch operation mode (batch mode).

- In single mode, data recovery of a single tablet is supported.
- In batch mode, support batch tablet data recovery.

In addition, users can use the command `show trash` to view the trash data capacity in BE, and use the command `admin clean trash` to clear the trash data in BE.

Operation

single mode

1. http request method

   BE provides an http interface for single tablet data recovery, the interface is as follows:

   ```
   curl -X POST "http://be_host:be_webserver_port/api/restore_tablet?tablet_id=11111\&schema_hash=12345"
   ```

   The successful results are as follows:

   ```
   {"status": "Success", "msg": "OK"}
   ```

   If it fails, the corresponding failure reason will be returned. One possible result is as follows:

   ```
   {"status": "Failed", "msg": "create link path failed"}
   ```

2. Script mode

   `restore_tablet_tool.sh` can be used to realize the function of single tablet data recovery.

   ```
   sh tools/restore_tablet_tool.sh -b "http://127.0.0.1:8040" -t 12345 -s 11111
   sh tools/restore_tablet_tool.sh --backend "http://127.0.0.1:8040" --tablet_id 12345 --schema_hash 11111
   ```

batch mode

The batch recovery mode is used to realize the function of recovering multiple tablet data.

When using, you need to put the restored tablet id and schema hash in a file in a comma-separated format in advance, one tablet per line.

The format is as follows:

```
12345,11111
12346,11111
12347,11111
```

Then perform the recovery with the following command (assuming the file name is: `tablets.txt`):

```
sh restore_tablet_tool.sh -b "http://127.0.0.1:8040" -f tablets.txt
sh restore_tablet_tool.sh --backend "http://127.0.0.1:8040" --file tablets.txt
```

Repair missing or damaged Tablet

In some very special circumstances, such as code bugs, or human misoperation, etc., all replicas of some tablets may be lost. In this case, the data has been substantially lost. However, in some scenarios, the

business still hopes to ensure that the query will not report errors even if there is data loss, and reduce the perception of the user layer. At this point, we can use the blank Tablet to fill the missing replica to ensure that the query can be executed normally.

Note: This operation is only used to avoid the problem of error reporting due to the inability to find a queryable replica, and it is impossible to recover the data that has been substantially lost.

1. View Master FE log `fe.log`

   If there is data loss, there will be a log similar to the following in the log:

   ```
   backend [10001] invalid situation. tablet[20000] has few replica[1], replica num setting is [3]
   ```

   This log indicates that all replicas of tablet 20000 have been damaged or lost.

2. Use blank replicas to fill in missing copies

   After confirming that the data cannot be recovered, you can execute the following command to generate blank replicas.

   ```
   ADMIN SET FRONTEND CONFIG ("recover_with_empty_tablet" = "true");
   ```

   - Note: You can first check whether the current version supports this parameter through the `ADMIN SHOW FRONTEND CONFIG;` command.

3. A few minutes after the setup is complete, you should see the following log in the Master FE log `fe.log`:

   ```
   tablet 20000 has only one replica 20001 on backend 10001 and it is lost. create an empty replica to recover it.
   ```

   The log indicates that the system has created a blank tablet to fill in the missing replica.

4. Judge whether it has been repaired successfully through query.

Data Recovery

Data Recovery

For the Unique Key Merge on Write table, there are bugs in some Doris versions, which may cause errors when the system calculates the delete bitmap, resulting in duplicate primary keys. At this time, the full compaction function can be used to repair the data. This function is invalid for non-Unique Key Merge on Write tables.

This feature requires Doris version 2.0+.

To use this function, it is necessary to stop the import as much as possible, otherwise problems such as import timeout may occur.

Brief principle explanation

After the full compaction is executed, the delete bitmap will be recalculated, and the wrong delete bitmap data will be deleted to complete the data restoration.

Instructions for use

`POST /api/compaction/run?tablet_id={int}&compact_type=full`

or

`POST /api/compaction/run?table_id={int}&compact_type=full`

Note that only one tablet_id and table_id can be specified, and cannot be specified at the same time. After specifying table_id, full_compaction will be automatically executed for all tablets under this table.

Example of use

```
curl -X POST "http://127.0.0.1:8040/api/compaction/run?tablet_id=10015&compact_type=full"
curl -X POST "http://127.0.0.1:8040/api/compaction/run?table_id=10104&compact_type=full"
```

Workload Group

The workload group can limit the use of compute and memory resources on a single be node for tasks within the group. Currently, query binding to workload groups is supported.

Workload group properties

- cpu_share: Required, used to set how much cpu time the workload group can acquire, which can achieve soft isolation of cpu resources. cpu_share is a relative value indicating the weight of cpu resources available to the running workload group. For example, if a user creates 3 workload groups rg-a, rg-b and rg-c with cpu_share of 10, 30 and 40 respectively, and at a certain moment rg-a and rg-b are running tasks while rg-c has no tasks, then rg-a can get 25% (10 / (10 + 30)) of the cpu resources while workload group rg-b can get 75% of the cpu resources. If the system has only one workload group running, it gets all the cpu resources regardless of the value of its cpu_share.

- memory_limit: Required, set the percentage of be memory that can be used by the workload group. The absolute value of the workload group memory limit is: `physical_memory * mem_limit * memory_limit`, where mem_limit is a be configuration item. The total memory_limit of all workload groups in the system must not exceed 100%. Workload groups are guaranteed to use the memory_limit for the tasks in the group in most cases. When the workload group memory usage exceeds this limit, tasks in the group with larger memory usage may be canceled to release the excess memory, refer to enable_memory_overcommit.

- enable_memory_overcommit: Optional, enable soft memory isolation for the workload group, default is false. if set to false, the workload group is hard memory isolated and the tasks with the largest memory usage will be canceled immediately after the workload group memory usage exceeds the limit to release the excess memory. if set to true, the workload group is hard memory isolated and the tasks with the largest memory usage will be canceled immediately after the workload group memory usage exceeds the limit to release the excess memory. if set to true, the workload group is softly isolated, if the system has free memory resources, the workload group can continue to use system memory after exceeding the memory_limit limit, and when the total system memory is tight, it will cancel several tasks in the group with the largest memory occupation, releasing part of the excess memory to relieve the system memory pressure. It is recommended that when this configuration is enabled for a workload group, the total memory_limit of all workload groups should be less than 100%, and the remaining portion should be used for workload group memory overcommit.

Workload group usage

1. Manually create a Workload Group named ‘normal’, which cannot be deleted. You can also restart FE after set FE config `experimental_enable_workload_group=true` to automatically create this group.

```
create workload group if not exists normal
properties (
    "cpu_share"="10",
    "memory_limit"="30%",
    "enable_memory_overcommit"="true"
);
```

2. Enable the experimental_enable_workload_group configuration, set in fe.conf to

```
experimental_enable_workload_group=true
```

3. To create a workload group:

```
create workload group if not exists g1
properties (
    "cpu_share"="10".
    "memory_limit"="30%".
    "enable_memory_overcommit"="true"
).
```

For details on creating a workload group, see CREATE-WORKLOAD-GROUP, and to delete a workload group, refer to DROP-WORKLOAD-GROUP; to modify a workload group, refer to ALTER-WORKLOAD-GROUP; to view the workload group, refer to: WORKLOAD_GROUPS() and SHOW-WORKLOAD-GROUPS.

4. turn on the pipeline execution engine, the workload group cpu isolation is based on the implementation of the pipeline execution engine, so you need to turn on the session variable:

```
set experimental_enable_pipeline_engine = true.
```

5. Bind the workload group.

- Bind the user to the workload group by default by setting the user property to `normal`.

```
set property 'default_workload_group' = 'g1'.
```

The current user's query will use 'g1' by default. * Specify the workload group via the session variable, which defaults to null.

```
set workload_group = 'g2'.
```

session variable `workload_group` takes precedence over user property `default_workload_group`, in case `workload_group` is empty, the query will be bound to `default_workload_group`, in case session variable `workload_group` is not empty, the query will be bound to `workload_group`.

If you are a non-admin user, you need to execute SHOW-WORKLOAD-GROUPS to check if the current user can see the workload group, if not, the workload group may not exist or the current user does not have permission to execute the query. If you cannot see the workload group, the workload group may not exist or the current user does not have privileges. To authorize the workload group, refer to: grant statement.

6. Execute the query, which will be associated with the g1 workload group.

Query queue Function

```
create workload group if not exists test_group
properties (
    "cpu_share"="10",
    "memory_limit"="30%",
```

```
    "max_concurrency" = "10",
    "max_queue_size" = "20",
    "queue_timeout" = "3000"
);
```

The current workload group supports the function of querying queues, which can be specified when creating a new group. The following three parameters are required: * max_concurrency, the maximum number of queries allowed by the current group; Queries exceeding the maximum concurrency will enter the queue logic * max_queue_size, the length of the query waiting queue; When the queue is full, new queries will be rejected * queue_timeout, the time the query waits in the queue. If the query wait time exceeds this value, the query will be rejected, and the time unit is milliseconds

It should be noted that the current queuing design is not aware of the number of FEs, and the queuing parameters only works in a single FE, for example:

A Doris cluster is configured with a work load group and set max_concurrency=1, If there is only 1 FE in the cluster, then this workload group will only run one SQL at the same time from the Doris cluster perspective, If there are 3 FEs, the maximum number of query that can be run in Doris cluster is 3.

Resource Group

The main purpose of Doris's multi-tenant and resource isolation solution is to reduce interference between multiple users when performing data operations in the same Doris cluster, and to allocate cluster resources to each user more reasonably.

The scheme is mainly divided into two parts, one is the division of resource groups at the node level in the cluster, and the other is the resource limit for a single query.

Nodes in Doris

First, let's briefly introduce the node composition of Doris. There are two types of nodes in a Doris cluster: Frontend (FE) and Backend (BE).

FE is mainly responsible for metadata management, cluster management, user request access and query plan analysis.

BE is mainly responsible for data storage and execution of query plans.

FE does not participate in the processing and calculation of user data, so it is a node with low resource consumption. The BE is responsible for all data calculations and task processing, and is a resource-consuming node. Therefore, the resource division and resource restriction schemes introduced in this article are all aimed at BE nodes. Because the FE node consumes relatively low resources and can also be scaled horizontally, there is usually no need to isolate and restrict resources, and the FE node can be shared by all users.

Node resource division

Node resource division refers to setting tags for BE nodes in a Doris cluster, and the BE nodes with the same tags form a resource group. Resource group can be regarded as a management unit of data storage and calculation. Below we use a specific example to introduce the use of resource groups.

1. Set labels for BE nodes

   Assume that the current Doris cluster has 6 BE nodes. They are host[1-6] respectively. In the initial situation, all nodes belong to a default resource group (Default).

   We can use the following command to divide these 6 nodes into 3 resource groups: group_a, group_b, group_c:

   ```
   alter system modify backend "host1:9050" set ("tag.location" = "group_a");
   alter system modify backend "host2:9050" set ("tag.location" = "group_a");
   alter system modify backend "host3:9050" set ("tag.location" = "group_b");
   alter system modify backend "host4:9050" set ("tag.location" = "group_b");
   alter system modify backend "host5:9050" set ("tag.location" = "group_c");
   alter system modify backend "host6:9050" set ("tag.location" = "group_c");
   ```

   Here we combine `host[1-2]` to form a resource group `group_a`, `host[3-4]` to form a resource group `group_b`, and `host[5-6]` to form a resource group `group_c`.

> Note: One BE only supports setting one Tag.

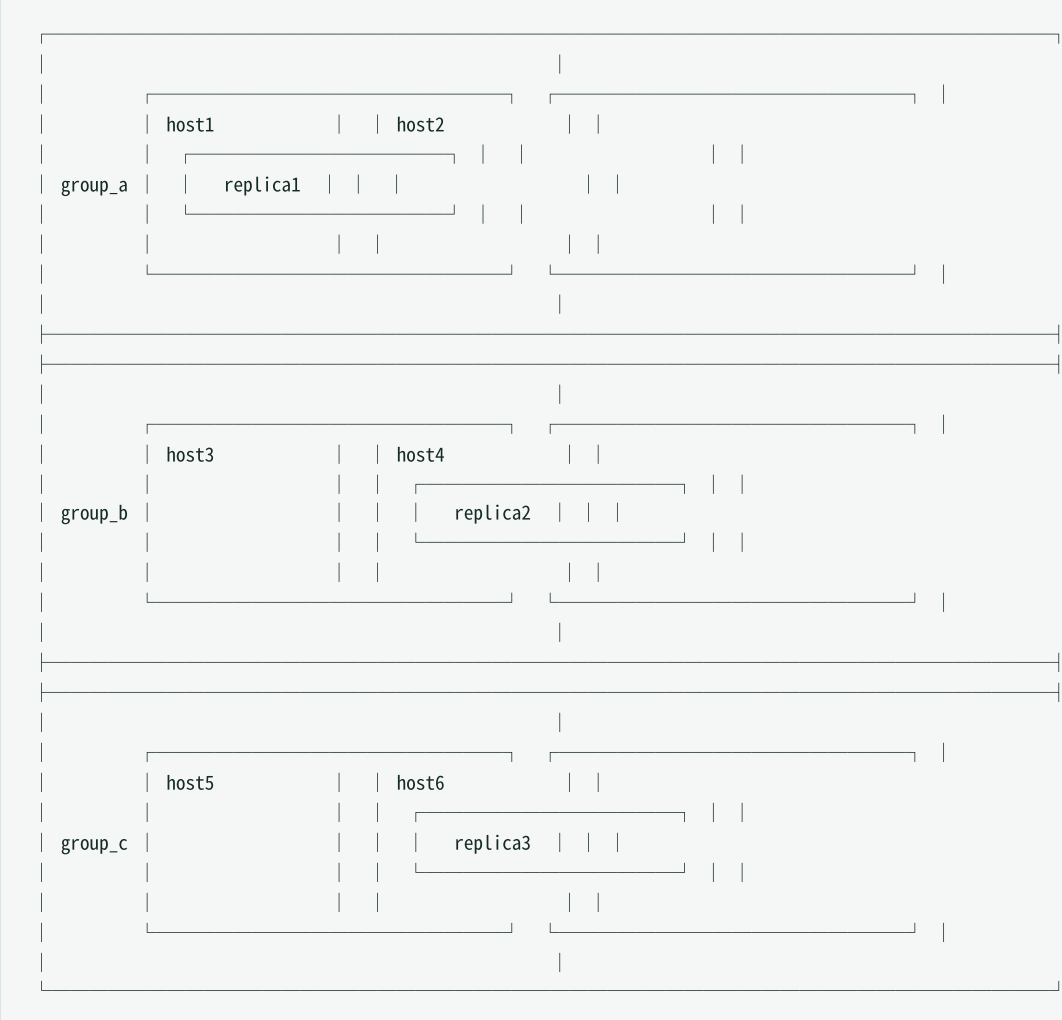
2. Distribution of data according to resource groups

   After the resource group is divided. We can distribute different copies of user data in different resource groups. Assume a user table UserTable. We want to store a copy in each of the three resource groups, which can be achieved by the following table creation statement:

```
create table UserTable
(k1 int, k2 int)
distributed by hash(k1) buckets 1
properties(
    "replication_allocation"="tag.location.group_a:1, tag.location.group_b:1, tag.location.group_c:1"
)
```

   In this way, the data in the UserTable table will be stored in the form of 3 copies in the nodes where the resource groups group_a, group_b, and group_c are located.

   The following figure shows the current node division and data distribution:



   For the convenience of setting the data distribution strategy for tables, a unified data distribution strategy can be set at the database level, but the priority of setting tables is higher than that of the database

```
CREATE DATABASE db_name PROPERTIES (
"replication_allocation" = "tag.location.group_a:1, tag.location.group_b:1"
)
```

3. Use different resource groups for data query

   After the execution of the first two steps is completed, we can limit a user's query by setting the user's resource usage permissions, and can only use the nodes in the specified resource group to execute.

   For example, we can use the following statement to restrict user1 to only use nodes in the `group_a` resource group for data query, user2 can only use the `group_b` resource group, and user3 can use 3 resource groups at the same time:

   ```
   set property for'user1''resource_tags.location' = 'group_a';
   set property for'user2''resource_tags.location' = 'group_b';
   set property for'user3''resource_tags.location' = 'group_a, group_b, group_c';
   ```

   After the setting is complete, when user1 initiates a query on the UserTable table, it will only access the data copy on the nodes in the `group_a` resource group, and the query will only use the node computing resources in the `group_a` resource group. The query of user3 can use copies and computing resources in any resource group.

   :::tip Tip

   By default, the user's `resource_tags.location` attribute is empty. In versions prior to 2.0.2 (inclusive), by default, users are not restricted by tags and can use any resource group. After version 2.0.3, normal users can only use the `default` resource group by default. Root and Admin user can use any resource group. :::

   It should be noted that after each modification of the `resource_tags.location` property, the user needs to re-establish the connection for the changes to take effect.

   In this way, we have achieved physical resource isolation for different user queries by dividing nodes and restricting user resource usage. Furthermore, we can create different users for different business departments and restrict each user from using different resource groups. In order to avoid the use of resource interference between different business parts. For example, there is a business table in the cluster that needs to be shared by all 9 business departments, but it is hoped that resource preemption between different departments can be avoided as much as possible. Then we can create 3 copies of this table and store them in 3 resource groups. Next, we create 9 users for 9 business departments, and limit the use of one resource group for every 3 users. In this way, the degree of competition for resources is reduced from 9 to 3.

   On the other hand, for the isolation of online and offline tasks. We can use resource groups to achieve this. For example, we can divide nodes into two resource groups, Online and Offline. The table data is still stored in 3 copies, of which 2 copies are stored in the Online resource group, and 1 copy is stored in the Offline resource group. The Online resource group is mainly used for online data services with high concurrency and low latency. Some large queries or offline ETL operations can be executed using nodes in the Offline resource group. So as to realize the ability to provide online and offline services simultaneously in a unified cluster.

4. Resource group assignments for load job

   The resource usage of load jobs (including insert, broker load, routine load, stream load, etc.) can be divided into two parts:

   1. Computing resources: responsible for reading data sources, data transformation and distribution.
   2. Write resource: responsible for data encoding, compression and writing to disk.

   The write resource must be the node where the replica is located, and the computing resource can theoretically select any node to complete. Therefore, the allocation of resource groups for load jobs is divided into two steps:

   1. Use user-level resource tags to limit the resource groups that computing resources can use.
   2. Use the resource tag of the replica to limit the resource group that the write resource can use.

   So if you want all the resources used by the load operation to be limited to the resource group where the data is located, you only need to set the resource tag of the user level to the same as the resource tag of the replica.

Single query resource limit

The resource group method mentioned earlier is resource isolation and restriction at the node level. In the resource group, resource preemption problems may still occur. For example, as mentioned above, the three business departments are arranged in the same resource group. Although the degree of resource competition is reduced, the queries of these three departments may still affect each other.

Therefore, in addition to the resource group solution, Doris also provides a single query resource restriction function.

At present, Doris's resource restrictions on single queries are mainly divided into two aspects: CPU and memory restrictions.

1. Memory Limitation

   Doris can limit the maximum memory overhead that a query is allowed to use. To ensure that the memory resources of the cluster will not be fully occupied by a query. We can set the memory limit in the following ways:

   ```
   # Set the session variable exec_mem_limit. Then all subsequent queries in the session (within the connection) use this
   memory limit.
   set exec_mem_limit=1G;
   # Set the global variable exec_mem_limit. Then all subsequent queries of all new sessions (new connections) use this memory
   limit.
   set global exec_mem_limit=1G;
   # Set the variable exec_mem_limit in SQL(Unit bytes). Then the variable only affects this SQL.
   select /*+ SET_VAR(exec_mem_limit=1073741824) */ id, name from tbl where xxx;
   ```

   Because Doris' query engine is based on the full-memory MPP query framework. Therefore, when the memory usage of a query exceeds the limit, the query will be terminated. Therefore, when a query cannot run under a reasonable memory limit, we need to solve it through some SQL optimization methods or cluster expansion.

2. CPU limitations

   Users can limit the CPU resources of the query in the following ways:

```
# Set the session variable cpu_resource_limit. Then all queries in the session (within the connection) will use this CPU
limit.
set cpu_resource_limit = 2
# Set the user's attribute cpu_resource_limit, then all queries of this user will use this CPU limit. The priority of this
attribute is higher than the session variable cpu_resource_limit
set property for'user1''cpu_resource_limit' = '3';
```

   The value of `cpu_resource_limit` is a relative value. The larger the value, the more CPU resources can be used. However, the upper limit of the CPU that can be used by a query also depends on the number of partitions and buckets of the table. In principle, the maximum CPU usage of a query is positively related to the number of tablets involved in the query. In extreme cases, assuming that a query involves only one tablet, even if `cpu_resource_limit` is set to a larger value, only 1 CPU resource can be used.

Through memory and CPU resource limits. We can divide user queries into more fine-grained resources within a resource group. For example, we can make some offline tasks with low timeliness requirements, but with a large amount of calculation, use less CPU resources and more memory resources. Some delay-sensitive online tasks use more CPU resources and reasonable memory resources.

Best practices and forward compatibility

Tag division and CPU limitation are new features in version 0.15. In order to ensure a smooth upgrade from the old version, Doris has made the following forward compatibility:

1. Each BE node will have a default Tag: `"tag.location": "default"`.
2. The BE node added through the `alter system add backend` statement will also set Tag: `"tag.location": "default"` by default.
3. The copy distribution of all tables is modified by default to: `"tag.location.default:xx`. xx is the number of original copies.
4. Users can still specify the number of replicas in the table creation statement by `"replication_num" = "xx"`, this attribute will be automatically converted to: `"tag.location.default:xx`. This ensures that there is no need to modify the original creation. Table statement.
5. By default, the memory limit for a single query is 2GB for a single node, and the CPU resources are unlimited, which is consistent with the original behavior. And the user's `resource_tags.location` attribute is empty, that is, by default, the user can access the BE of any Tag, which is consistent with the original behavior.

Here we give an example of the steps to start using the resource division function after upgrading from the original cluster to version 0.15:

1. Turn off data repair and balance logic

   After the upgrade, the default Tag of BE is `"tag.location": "default"`, and the default copy distribution of the table is: `"tag.location.default:xx`. So if you directly modify the Tag of BE, the system will

Automatically detect changes in the distribution of copies, and start data redistribution. This may occupy some system resources. So we can turn off the data repair and balance logic before modifying the tag to ensure that there will be no copies when we plan resources Redistribution operation.

```
ADMIN SET FRONTEND CONFIG ("disable_balance" = "true");
ADMIN SET FRONTEND CONFIG ("disable_tablet_scheduler" = "true");
```

2. Set Tag and table copy distribution

   Next, you can use the `alter system modify backend` statement to set the BE Tag. And through the `alter table` statement to modify the copy distribution strategy of the table. Examples are as follows:

   ```
   alter system modify backend "host1:9050, 1212:9050" set ("tag.location" = "group_a");
   alter table my_table modify partition p1 set ("replication_allocation" = "tag.location.group_a:2");
   ```

3. Turn on data repair and balance logic

   After the tag and copy distribution are set, we can turn on the data repair and equalization logic to trigger data redistribution.

   ```
   ADMIN SET FRONTEND CONFIG ("disable_balance" = "false");
   ADMIN SET FRONTEND CONFIG ("disable_tablet_scheduler" = "false");
   ```

   This process will continue for a period of time depending on the amount of data involved. And it will cause some colocation tables to fail colocation planning (because the copy is being migrated). You can view the progress by `show proc "/cluster_balance/"`. You can also judge the progress by the number of `UnhealthyTabletNum` in `show proc "/statistic"`. When `UnhealthyTabletNum` drops to 0, it means that the data redistribution is completed. .

4. Set the user's resource label permissions.

   After the data is redistributed. We can start to set the user's resource label permissions. Because by default, the user's `resource_tags.location` attribute is empty, that is, the BE of any tag can be accessed. Therefore, in the previous steps, the normal query of existing users will not be affected. When the `resource_tags.location` property is not empty, the user will be restricted from accessing the BE of the specified Tag.

Through the above 4 steps, we can smoothly use the resource division function after the original cluster is upgraded.

How to conveniently set replica distribution strategies when there are many tables

For example, there is a db1 with four tables under it, and the replica distribution strategy required for table1 is `group_a:1,group_b:2`, the replica distribution strategy required for tables 2, 3, and 4 is `group_c:1,group_b:2`

Then you can use the following statement to create db1:

```
CREATE DATABASE db1 PROPERTIES (
"replication_allocation" = "tag.location.group_c:1, tag.location.group_b:2"
)
```

Create table1 using the following statement:

```
CREATE TABLE table1
(k1 int, k2 int)
distributed by hash(k1) buckets 1
properties(
"replication_allocation"="tag.location.group_a:1, tag.location.group_b:2"
)
```

The table creation statements for table2, table3, and table4 do not need to specify `replication_allocation` again.

Note: Changing the replica distribution policy of the database will not affect existing tables.

SQL Interception

SQL Block Rule

This function is used to limit any sql statement (no matter DDL or DML statement). Support SQL block rule by user level:

1. by regex way to deny specify SQL
2. by setting partition_num, tablet_num, cardinality, check whether a sql reaches one of the limitations
    - partition_num, tablet_num, cardinality could be set together, and once reach one of them, the sql will be blocked.

Rule

SQL block rule CRUD - create SQL block rule,For more creation syntax seeCREATE SQL BLOCK RULE - sql: Regex pattern, Special characters need to be translated, “NULL” by default - sqlHash: Sql hash value, Used to match exactly, We print it in fe.audit.log, This parameter is the only choice between sql and sql, “NULL” by default - partition_num: Max number of partitions will be scanned by a scan node, 0L by default - tablet_num: Max number of tablets will be scanned by a scan node, 0L by default - cardinality: An inaccurate number of scan rows of a scan node, 0L by default - global: Whether global(all users)is in effect, false by default - enable: Whether to enable block rule, true by default

```
CREATE SQL_BLOCK_RULE test_rule
PROPERTIES(
  "sql"="select \\* from order_analysis",
  "global"="false",
  "enable"="true",
  "sqlHash"=""
)
```

> Notes:
>
> That the sql statement here does not end with a semicolon

When we execute the sql that we defined in the rule just now, an exception error will be returned. An example is as follows:

```
mysql> select * from order_analysis;
ERROR 1064 (HY000): errCode = 2, detailMessage = sql match regex sql block rule: order_analysis_rule
```

- create test_rule2, limits the maximum number of scanning partitions to 30 and the maximum scanning cardinality to 10 billion rows. As shown in the following example:

```
CREATE SQL_BLOCK_RULE test_rule2 PROPERTIES("partition_num" = "30", "cardinality"="10000000000","global"="false","enable"="true")
```

- show configured SQL block rules, or show all rules if you do not specify a rule name,Please see the specific grammar SHOW SQL BLOCK RULE

```
SHOW SQL_BLOCK_RULE [FOR RULE_NAME]
```

- alter SQL block rule, Allows changes sql/sqlHash/global/enable/partition_num/tablet_num/cardinality anyone,Please see the specific grammarALTER SQL BLOCK RULE
    - sql and sqlHash cannot be set both. It means if sql or sqlHash is set in a rule, another property will never be allowed to be altered
    - sql/sqlHash and partition_num/tablet_num/cardinality cannot be set together. For example, partition_num is set in a rule, then sql or sqlHash will never be allowed to be altered.

    ```
    ALTER SQL_BLOCK_RULE test_rule PROPERTIES("sql"="select \\* from test_table","enable"="true")
    ```

```
ALTER SQL_BLOCK_RULE test_rule2 PROPERTIES("partition_num" = "10","tablet_num"="300","enable"="true")
```

- drop SQL block rule, Support multiple rules, separated by ,,Please see the specific grammarDROP SQL BLOCK RULE

```
DROP SQL_BLOCK_RULE test_rule1,test_rule2
```

User bind rules

If global=false is configured, the rules binding for the specified user needs to be configured, with multiple rules separated by ‘,’

```
SET PROPERTY [FOR 'jack'] 'sql_block_rules' = 'test_rule1,test_rule2'
```

Kill Query

Kill Query

Kill connection

In Doris, each connection runs in a separate thread. You can terminate a thread using the `KILL processlist_id`statement.

The `processlist_id` for the thread can be found in the Id column from the SHOW PROCESSLIST output. Or you can use the `SELECT CONNECTION_ID()` command to query the current connection id.

Syntax:

```
KILL [CONNECTION] processlist_id
```

Kill query

You can also terminate the query command under execution based on the processlist_id or the query_id.

Syntax:

```
KILL QUERY processlist_id | query_id
```

Example

1. Check the current connection id.

```
mysql select connection_id();
+-----------------+
| connection_id() |
+-----------------+
| 48              |
+-----------------+
1 row in set (0.00 sec)
```

2. Check all connection id.

```
mysql SHOW PROCESSLIST;
+------------------+------+------+--------------------+---------------------+----------+---------+---------+------+-------+---------
-------------------------+-----------------------------------------------------------------------------------------+
| CurrentConnected | Id   | User | Host               | LoginTime           | Catalog  | Db      | Command | Time | State | QueryId
| Info                                                                                                    |
+------------------+------+------+--------------------+---------------------+----------+---------+---------+------+-------+---------
-------------------------+-----------------------------------------------------------------------------------------+
| Yes              |   48 | root | 10.16.xx.xx:44834   | 2023-12-29 16:49:47 | internal | test | Query   |    0 | OK    |
e6e4ce9567b04859-8eeab8d6b5513e38 | SHOW PROCESSLIST                                                                                          |
|                  |   50 | root | 192.168.xx.xx:52837 | 2023-12-29 16:51:34 | internal |      | Sleep   | 1837 | EOF   |
deaf13c52b3b4a3b-b25e8254b50ff8cb | SELECT @@session.transaction_isolation                                                                    |
```

```
|                  |   51 | root | 192.168.xx.xx:52843 | 2023-12-29 16:51:35 | internal |      | Sleep   |  907 | EOF   |
437f219addc0404f-9befe7f6acf9a700 | /* ApplicationName=DBeaver Ultimate 23.1.3 - Metadata */ SHOW STATUS                 |
|                  |   55 | root | 192.168.xx.xx:55533 | 2023-12-29 17:09:32 | internal | test | Sleep   |  271 | EOF   |
f02603dc163a4da3-beebbb5d1ced760c | /* ApplicationName=DBeaver Ultimate 23.1.3 - SQLEditor <Console> */ SELECT DATABASE() |
|                  |   47 | root | 10.16.xx.xx:35678   | 2023-12-29 16:21:56 | internal | test | Sleep   | 3528 | EOF   |
f4944c543dc34a99-b0d0f3986c8f1c98 | select * from test                                                                  |
+------------------+------+------+--------------------+---------------------+----------+---------+---------+------+-------+---------
-------------------------+-------------------------------------------------------------------------------------------+
5 rows in set (0.00 sec)
```

3. Kill the currently running query, which will then be displayed as canceled.

```
mysql kill query 55;
Query OK, 0 rows affected (0.01 sec)
```

Authority Management

Authority Management

Doris's new privilege management system refers to Mysql's privilege management mechanism, achieves table-level fine-grained privilege control, role-based privilege access control, and supports whitelist mechanism.

Noun Interpretation

1. user_identity

   In a permission system, a user is identified as a User Identity. User ID consists of two parts: username and userhost. Username is a user name, which is composed of English upper and lower case. Userhost represents the IP from which the user link comes. User_identity is presented as username@'userhost', representing the username from userhost.

   Another expression of user_identity is username@['domain'], where domain is the domain name, which can be resolved into a set of IPS by DNS . The final expression is a set of username@'userhost', so we use username@'userhost'to represent it.

2. Privilege

   The objects of permissions are nodes, catalogs, databases or tables. Different permissions represent different operating permissions.

3. Role

   Doris can create custom named roles. Roles can be seen as a set of permissions. When a newly created user can be assigned a role, the role's permissions are automatically granted. Subsequent changes in the role's permissions will also be reflected in all user permissions that belong to the role.

4. user_property

   User attributes are directly attached to a user, not to a user identity. That is, both cmy@'192.%'and cmy@['domain'] have the same set of user attributes, which belong to user cmy, not cmy@'192.%' or cmy@['domain'].

   User attributes include, but are not limited to, the maximum number of user connections, import cluster configuration, and so on.

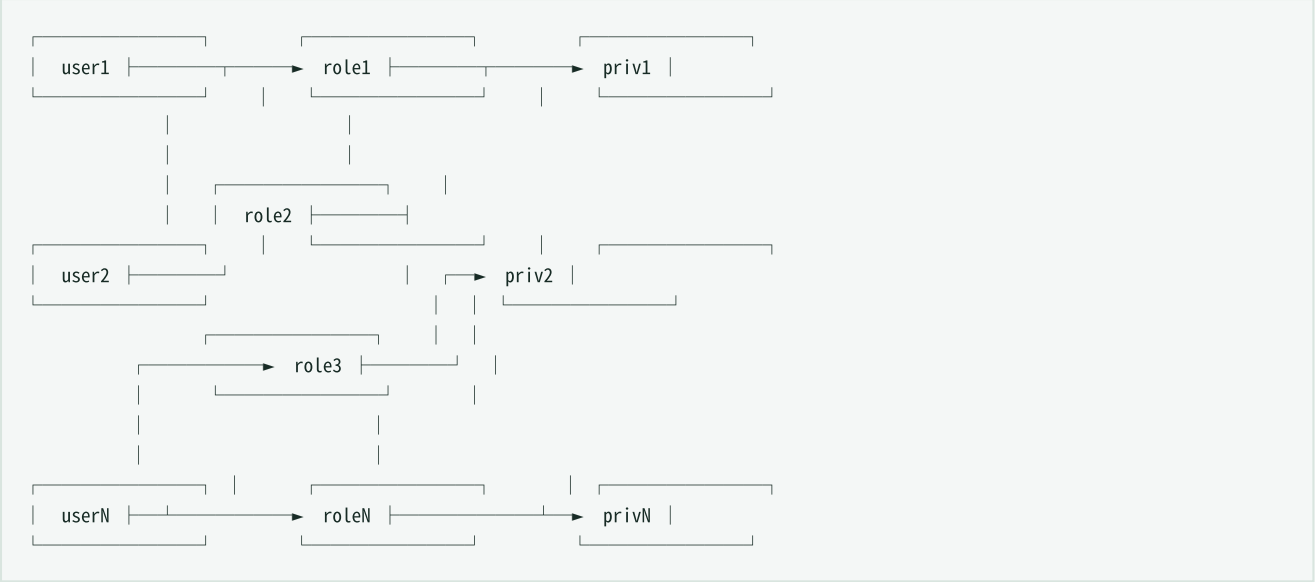
Permission framework

Doris permission design is based on RBAC (Role-Based Access Control) permission management model. Users are associated with roles, roles and permissions, and users are associated with permissions indirectly through roles.

When a role is deleted, the user automatically loses all permissions of the role.

When a user and a role are disassociated, the user automatically loses all permissions of the role.

When the role's permissions are added or deleted, the user's permissions will also change.



As shown in the figure above:

Both user1 and user2 have priv1 permissions through role1.

UserN has priv1 permissions through role3, priv2 and privN permissions through roleN, so userN has priv1, priv2 and privN permissions at the same time.

In order to facilitate user operation, users can be authorized directly. In the underlying implementation, a default role dedicated to the user is created for each user. When authorizing a user, it is actually authorizing the user's default role.

The default role cannot be deleted or assigned to others. When a user is deleted, the default role will also be deleted automatically.

Supported operations

1. Create users: CREATE USER
2. Alter users: ALTER USER
3. Delete users: DROP USER
4. Authorization/Assign roles: GRANT
5. Withdrawal/REVOKE roles: REVOKE
6. Create role: CREATE ROLE
7. Delete roles: DROP ROLE
8. View current user privileges: SHOW GRANTS
9. View all user privileges: SHOW ALL GRANTS
10. View the created roles: SHOW ROLES
11. Set user properties: SET PROPERTY
12. View user properties: SHOW PROPERTY
13. Change password :SET PASSWORD

For detailed help with the above commands, you can use help + command to get help after connecting Doris through the MySQL client. For example `HELP CREATE USER`.

Permission type

Doris currently supports the following permissions

1. Node_priv

   Nodes change permissions. Including FE, BE, BROKER node addition, deletion, offline operations. User who has Node_priv and Grant_priv permission, can grant Node_priv to other users.

   The root user has this permission by default.

   Users who have both Grant_priv and Node_priv can grant this privilege to other users.

   This permission can only be granted to the Global level.

2. Grant_priv

   Permissions change permissions. Allow the execution of operations including authorization, revocation, add/delete/change user/role, etc.

   However, a user with this permission can not grant node_priv permission to other users, unless the user itself has node_priv permission.

3. Select_priv

   Read-only access to databases and tables.

4. Load_priv

   Write permissions to databases and tables. Including Load, Insert, Delete and so on.

5. Alter_priv

   Change permissions on databases and tables. It includes renaming libraries/tables, adding/deleting/changing columns, and adding/deleting partitions.

6. Create_priv

   The right to create databases, tables, and views.

7. Drop_priv

   Delete permissions for databases, tables, and views.

8. Usage_priv

   Use of resources and workload groups.

Permission hierarchy

At the same time, according to the scope of application of permissions, we divide them into four levels:

1. GLOBAL LEVEL: Global permissions. That is, permissions on `*.*.*` granted by GRANT statements. The granted permissions apply to any table in any database.
2. CATALOG LEVEL: Catalog level permissions. That is, the permissions on `ctl.*.*` granted through the GRANT statement. The permissions granted apply to any library table in the specified Catalog.
3. DATABASE LEVEL: Database-level permissions. That is, the permissions on `ctl.db.*` granted through the GRANT statement. The privileges granted apply to any table in the specified database.
4. TABLE LEVEL: Table-level permissions. That is, the permissions on `ctl.db.tbl` granted through the GRANT statement. The privileges granted apply to the specified table in the specified database.

The privileges of the resources are divided into two levels as follows:

1. GLOBAL LEVEL: Global privileges. That is, the privileges granted on * by the GRANT statement. The privileges granted apply to the resource.
2. RESOURCE LEVEL: Resource-level privileges. This is the permission on `resource_name` granted by the GRANT statement. The privilege granted applies to the specified resource.

The workload group has only one level: 1. WORKLOAD GROUP LEVEL: privileges on `workload_group_name` that can be granted with the GRANT statement. The privileges granted apply to the specified workload group. workload_group_name supports `%` and _ match characters, `%` can match any string and _ matches any single character.

ADMIN /GRANT

ADMIN_PRIV and GRANT_PRIV have the authority of **“grant authority”** at the same time, which is more special. The operations related to these two privileges are described here one by one.

1. CREATE USER
    - Users with ADMIN privileges, or GRANT privileges at the GLOBAL and DATABASE levels can create new users.
2. DROP USER
    - Users with ADMIN authority or GRANT authority at the global level can drop users.
3. CREATE/DROP ROLE
    - Users with ADMIN authority or GRANT authority at the global level can create or drop role.
4. GRANT /REVOKE
    - Users with ADMIN or GLOBAL GRANT privileges can grant or revoke the privileges of any user.
    - Users with GRANT privileges at the DATABASE level can grant or revoke the privileges of any user on the specified database.
    - Users with GRANT privileges at TABLE level can grant or revoke the privileges of any user on the specified tables in the specified database.
5. SET PASSWORD
    - Users with ADMIN or GLOBAL GRANT privileges can set any user’s password.
    - Ordinary users can set their corresponding User Identity password. The corresponding User Identity can be viewed by `SELECT CURRENT_USER();`command.

- Users with GRANT privileges at non-GLOBAL level cannot set the password of existing users, but can only specify the password when creating users.

Some explanations

1. When Doris initializes, the following users and roles are automatically created:
    1. Operator role: This role has Node_priv and Admin_priv, i.e. all permissions for Doris. In a subsequent upgrade version, we may restrict the role's permissions to Node_priv, which is to grant only node change permissions. To meet some cloud deployment requirements.
    2. admin role: This role has Admin_priv, which is all permissions except for node changes.
    3. root@'%': root user, which allows login from any node, with the role of operator.
    4. admin@'%': admin user, allowing login from any node, role admin.
2. It is not supported to delete or change the permissions of default created roles or users.
3. The user of the operator role has one and only one user, that is, root. Users of admin roles can create multiple.
4. Operational instructions for possible conflicts
    1. Conflict between domain name and ip:

        Assume that the following users are created:

        CREATE USER cmy@['domain'];

        And authorize:

        GRANT SELECT_PRIV ON *.* TO cmy@['domain']

        The domain is resolved into two ips: IP1 and IP2

        Let's assume that we have a separate authorization for cmy@'ip1':

        GRANT ALTER_PRIV ON *.* TO cmy@'ip1';

        The permissions of CMY @'ip1'will be changed to SELECT_PRIV, ALTER_PRIV. And when we change the permissions of cmy@['domain'] again, cmy@'ip1' will not follow.

    2. duplicate IP conflicts:

        Assume that the following users are created:

        CREATE USER cmy@'%' IDENTIFIED BY "12345";

        CREATE USER cmy@'192.%' IDENTIFIED BY "abcde";

        In priority,'192.%'takes precedence over'%', so when user CMY tries to login Doris with password '12345' from 192.168.1.1, the machine will be rejected.
5. Forget passwords

If you forget your password and cannot log in to Doris, you can add `skip_localhost_auth_check` in fe config and restart FE so that logging to Doris without a password in localhost.

```
skip_localhost_auth_check = true
```

After login, the password can be reset through the SET PASSWORD command.

6. No user can reset the password of the root user except the root user himself.
7. ADMIN_PRIV permissions can only be granted or revoked at the GLOBAL level.
8. Having GRANT_PRIV at GLOBAL level is actually equivalent to having ADMIN_PRIV, because GRANT_PRIV at this level has the right to grant arbitrary permissions, please use it carefully.
9. `current_user()` and `user()`

   Users can view `current_user` and `user` respectively by `SELECT current_user();` and `SELECT user();`. Where `current_user` indicates which identity the current user is passing through the authentication system, and `user` is the user's current actual `user_identity`.

   For example, suppose the user `user1@'192.%'` is created, and then a user user1 from 192.168.10.1 is logged into the system. At this time, `current_user` is `user1@'192.%'`, and `user` is `user1@'192.168.10.1'`.

   All privileges are given to a `current_user`, and the real user has all the privileges of the corresponding `current_user`.

10. Password Validation

    In version 1.2, the verification function of user password strength has been added. This feature is controlled by the global variable `validate_password_policy`. Defaults to `NONE/0`, i.e. password strength is not checked. If set to `STRONG/2`, the password must contain 3 items of "uppercase letters", "lowercase letters", "numbers" and "special characters", and the length must be greater than or equal to 8.

Best Practices

Here are some usage scenarios of Doris privilege system.

1. Scene 1

   The users of Doris cluster are divided into Admin, RD and Client. Administrators have all the rights of the whole cluster, mainly responsible for cluster building, node management and so on. The development engineer is responsible for business modeling, including database building, data import and modification. Users access different databases and tables to get data.

   In this scenario, ADMIN or GRANT privileges can be granted to administrators. Give RD CREATE, DROP, ALTER, LOAD, SELECT permissions to any or specified database tables. Give Client SELECT permission to any or specified database table. At the same time, it can also simplify the authorization of multiple users by creating different roles.

2. Scene 2

   There are multiple services in a cluster, and each business may use one or more data. Each business needs to manage its own users. In this scenario. Administrator users can create a user with GRANT

privileges at the DATABASE level for each database. The user can only authorize the specified database for the user.

3. Blacklist

   Doris itself does not support blacklist, only whitelist, but we can simulate blacklist in some way. Suppose you first create a user named `user@'192.%'`, which allows users from `192.*` to login. At this time, if you want to prohibit users from `192.168.10.1` from logging in, you can create another user with `cmy@'192.168.10.1'` and set a new password. Since `192.168.10.1` has a higher priority than `192.%`, user can no longer login by using the old password from `192.168.10.1`.

More help

For more detailed syntax and best practices for permission management use, please refer to the GRANTS command manual. Enter `HELP GRANTS` at the command line of the MySql client for more help information.

MySQL Client Certificate

Communicate with the server over an encrypted connection

Doris supports SSL-based encrypted connections. It currently supports TLS1.2 and TLS1.3 protocols. Doris' SSL mode can be enabled through the following configuration: Modify the FE configuration file `conf/fe.conf` and add `enable_ssl = true`.

Next, connect to Doris through `mysql` client, mysql supports three SSL modes:

1. `mysql -uroot -P9030 -h127.0.0.1` is the same as `mysql --ssl-mode=PREFERRED -uroot -P9030 -h127.0.0.1`, both try to establish an SSL encrypted connection at the beginning, if it fails , a normal connection is attempted.
2. `mysql --ssl-mode=DISABLE -uroot -P9030 -h127.0.0.1`, do not use SSL encrypted connection, use normal connection directly.
3. `mysql --ssl-mode=REQUIRED -uroot -P9030 -h127.0.0.1`, force the use of SSL encrypted connections.

> Note: `--ssl-mode` parameter is introduced by mysql5.7.11 version, please refer to here for mysql client version lower than this version。 Doris needs a key certificate file to verify the SSL encrypted connection. The default key certificate file is located at `Doris/fe/mysql_ssl_default_certificate/certificate.p12`, and the default password is `doris`. You can modify the FE configuration file `conf/fe. conf`, add `mysql_ssl_default_certificate = /path/to/your/certificate` to modify the key certificate file, and you can also add the password corresponding to your custom key book file through `mysql_ssl_default_certificate_password = your_password`.

Doris also supports mTLS: Modify the FE configuration file `conf/fe.conf` and add `ssl_force_client_auth=true`.

Then you can connect to Doris via the `mysql` client:

```
mysql -ssl-mode=VERIFY_CA -uroot -P9030 -h127.0.0.1 --tls-version=TLSv1.2 --ssl-ca=/path/to/your/ca --ssl-cert=/path/to/your/cert --ssl-key=/path/to/your/key
```

The default ca, cert, and key files are located in `Doris/conf/mysql_ssl_default_certificate/client_certificate/`, named `ca.pem`, `client-cert.pem`, and `client-key.pem` respectively.

You can also generate your own certificate files using openssl or keytool.

Key Certificate Configuration

Enabling SSL functionality in Doris requires configuring both a CA key certificate and a server-side key certificate. To enable mutual authentication, a client-side key certificate must also be generated:

- The default CA key certificate file is located at `Doris/fe/mysql_ssl_default_certificate/ca_certificate.p12`, with a default password of `doris`. You can

modify the FE configuration file `conf/fe.conf` to add `mysql_ssl_default_ca_certificate = /path/to/your/certificate` to change the CA key certificate file. You can also add `mysql_ssl_default_ca_certificate_password = your_password` to specify the password for your custom key certificate file.

- The default server-side key certificate file is located at `Doris/fe/mysql_ssl_default_certificate/server_certificate.p12`, with a default password of `doris`. You can modify the FE configuration file `conf/fe.conf` to add `mysql_ssl_default_server_certificate = /path/to/your/certificate` to change the server-side key certificate file. You can also add `mysql_ssl_default_server_certificate_password = your_password` to specify the password for your custom key certificate file.
- By default, a client-side key certificate is also generated and stored in `Doris/fe/mysql_ssl_default_certificate/client-key.pem` and `Doris/fe/mysql_ssl_default_certificate/client_certificate/`.

Custom key certificate file

In addition to the Doris default certificate file, you can also generate a custom certificate file through `openssl`. Here are the steps (refer to Creating SSL Certificates and Keys Using OpenSSL):

1. Generate the CA, server-side, and client-side keys and certificates:

```
# Generate the CA certificate
openssl genrsa 2048 > ca-key.pem
openssl req -new -x509 -nodes -days 3600 \
        -key ca-key.pem -out ca.pem

# Generate the server certificate and sign it with the above CA
# server-cert.pem = public key, server-key.pem = private key
openssl req -newkey rsa:2048 -days 3600 \
        -nodes -keyout server-key.pem -out server-req.pem
openssl rsa -in server-key.pem -out server-key.pem
openssl x509 -req -in server-req.pem -days 3600 \
        -CA ca.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem

# Generate the client certificate and sign it with the above CA
# client-cert.pem = public key, client-key.pem = private key
openssl req -newkey rsa:2048 -days 3600 \
        -nodes -keyout client-key.pem -out client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -req -in client-req.pem -days 3600 \
        -CA ca.pem -CAkey ca-key.pem -set_serial 01 -out client-cert.pem
```

2. Verify the created certificates:

```
openssl verify -CAfile ca.pem server-cert.pem client-cert.pem
```

3. Combine your key and certificate in a PKCS#12 (P12) bundle. You can also specify a certificate format (PKCS12 by default). You can modify the conf/fe.conf configuration file and add parameter ssl_trust_store_type to specify the certificate format.

```
# Package the CA key and certificate
openssl pkcs12 -inkey ca-key.pem -in ca.pem -export -out ca_certificate.p12

# Package the server-side key and certificate
openssl pkcs12 -inkey server-key.pem -in server-cert.pem -export -out server_certificate.p12
```

> **Note**
>
> reference documents

FE SSL Certificate

To enable SSL function on Doris FE interface, you need to configure key certificate as follows:

1.Purchase or generate a self-signed SSL certificate. It is advised to use CA certificate in Production environment

2.Copy the SSL certificate to specified path. The default path is `${DORIS_HOME}/conf/ssl/`, and user can also specify their own path

3.Modify FE configuration file `conf/fe.conf`, and note that the following parameters are consistent with purchased or generated SSL certificate Set `enable_https = true` to enable https function, default is `false` Set certificate path `key_store_path`, default is `${DORIS_HOME}/conf/ssl/doris_ssl_certificate.keystore` Set certificate password `key_store_password`, default is null Set certificate type `key_store_type`, default is `JKS` Set certificate alias `key_store_alias`, default is `doris_ssl_certificate`

LDAP

LDAP

Access to third-party LDAP services to provide authentication login and group authorization services for Doris.

LDAP authentication login complements Doris authentication login by accessing the LDAP service for password authentication; Doris uses LDAP to authenticate the user's password first; if the user does not exist in the LDAP service, it continues to use Doris to authenticate the password; if the LDAP password is correct but there is no corresponding account in Doris, a temporary user is created to log in to Doris.

LDAP group authorization, is to map the group in LDAP to the Role in Doris, if the user belongs to multiple user groups in LDAP, after logging into Doris the user will get the permission of all groups corresponding to the Role, requiring the group name to be the same as the Role name.

Noun Interpretation

- LDAP: Lightweight directory access protocol that enables centralized management of account passwords.
- Privilege: Permissions act on nodes, databases or tables. Different permissions represent different permission to operate.
- Role: Doris can create custom named roles. A role can be thought of as a collection of permissions.

LDAP related concepts

In LDAP, data is organized in a tree structure.

Example (the following introduction will be expanded based on this example)

- dc=example,dc=com
- ou = ou1
    - cn = group1
    - cn = user1
- ou = ou2
    - cn = group2
        - cn = user2
- cn = user3

Explanation of LDAP Terms

- dc(Domain Component): It can be understood as the domain name of an organization, serving as the root node of a tree
- dn(Distinguished Name): Equivalent to a unique name, for example, the dn of user1 is cn=user1,ou=ou1,dc=example,dc=com the dn of user2 is cn=user2,cn=group2,ou=ou2,dc=example,dc=com

- rdn(Relative Distinguished Name): As part of dn, the four rdns of user1 are cn=user1 ou=ou1 dc=example and dc=com
- ou(Organization Unit): It can be understood as a sub organization, where users can be placed in ou or directly in the example.com domain
- cn(common name):name
- group: Group, which can be understood as the role of Doris
- user: User, equivalent to Doris' user
- objectClass：It can be understood as the type of data in each row, such as how to distinguish whether group1 is a group or a user. Each type of data requires different attributes below, such as CN and member (user list) for group, CN, password, uid, etc. for user

Enable LDAP Authentication

Server-side Configuration

You need to configure the LDAP basic information in the fe/conf/ldap.conf file, and the LDAP administrator password needs to be set using sql statements.

Configure the fe/conf/ldap.conf file：

- ldap_authentication_enabled = false
  Set the value to "true" to enable LDAP authentication; when the value is "false", LDAP authentication is not enabled and all other configuration items of this profile are invalid.Set the value to "true" to enable LDAP authentication; when the value is "false", LDAP authentication is not enabled and all other configuration items of this profile are invalid.

- ldap_host = 127.0.0.1
  LDAP service ip.

- ldap_port = 389
  LDAP service port, the default plaintext transfer port is 389, currently Doris' LDAP function only supports plaintext password transfer.

- ldap_admin_name = cn=admin,dc=domain,dc=com
  LDAP administrator account "Distinguished Name". When a user logs into Doris using LDAP authentication, Doris will bind the administrator account to search for user information in LDAP.

- ldap_user_basedn = ou=people,dc=domain,dc=com Doris base dn when searching for user information in LDAP,For example, only user2 in the above example is allowed to log in to Doris, which is configured as ou=ou2, dc=example, dc=com. If user1, user2, and user3 in the above example are allowed to log in to Doris, which is configured as dc=example, dc=com

- ldap_user_filter = (&(uid={login}))

  For Doris' filtering criteria when searching for user information in LDAP, the placeholder "{login}" will be replaced with the login username. You must ensure that the user searched by this filter is unique, otherwise Doris will not be able to verify the password through LDAP and the error message "ERROR 5081 (42000): user is not unique in LDAP server." will appear when logging in.

For example, if you use the LDAP user node uid attribute as the username to log into Doris, you can configure it as:
ldap_user_filter = (&(uid={login}));
This item can be configured using the LDAP user mailbox prefix as the user name:
ldap_user_filter = (&(mail={login}@baidu.com))。

- ldap_group_basedn = ou=group,dc=domain,dc=com base dn when Doris searches for group information in LDAP. if this item is not configured, LDAP group authorization will not be enabled. Same as ldap_ User_ Similar to basedn, it limits the scope of Doris searching for groups.

Set the LDAP administrator password:

After configuring the ldap.conf file, start fe, log in to Doris with the root or admin account, and execute sql:

```
set ldap_admin_password = password('ldap_admin_password');
```

Client-side configuration

MySQL Client

Client-side LDAP authentication requires the mysql client-side explicit authentication plugin to be enabled. Logging into Doris using the command line enables the mysql explicit authentication plugin in one of two ways.

- Set the environment variable LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN to value 1. For example, in a linux or max environment you can use the command:

  ```
  echo "export LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1" >> ~/.bash_profile && source ~/.bash_profile
  ```

- Add the parameter “–enable-cleartext-plugin” each time you log in to Doris.

  ```
  mysql -hDORIS_HOST -PDORIS_PORT -u user -p --enable-cleartext-plugin

  Enter ldap password
  ```

Jdbc Client

When using Jdbc Client to connect in to Doris, you need to customize the plugin.

First, create a class called `MysqlClearPasswordPluginWithoutSSL` that inherits from `MysqlClearPasswordPlugin`. In this class, override the `requiresConfidentiality()` method and return false.

```
public class MysqlClearPasswordPluginWithoutSSL extends MysqlClearPasswordPlugin {
@Override
public boolean requiresConfidentiality() {
    return false;
  }
}
```

When obtaining a database connection, you need to configure the custom plugin into the properties

That is (xxx is the package name of the custom class) - authenticationPlugins=xxx.xxx.xxx.MysqlClearPasswordPluginWithoutSSL - defaultAuthenticationPlugin=xxx.xxx.xxx.MysqlClearPasswordPluginWithoutSSL - disabledAuthenticationPlugins=com.mysql.jdbc.authentication.MysqlClearPasswordPlugin

eg:

```
 jdbcUrl = "jdbc:mysql://localhost:9030/mydatabase?
authenticationPlugins=xxx.xxx.xxx.MysqlClearPasswordPluginWithoutSSL&defaultAuthenticationPlugin=xxx.xxx.xxx.MysqlClearPasswordPlugi
nWithoutSSL&disabledAuthenticationPlugins=com.mysql.jdbc.authentication.MysqlClearPasswordPlugin";
```

LDAP authentication detailed explanation

LDAP password authentication and group authorization are complementary to Doris password authentication and authorization. Enabling LDAP functionality does not completely replace Doris password authentication and authorization, but coexists with Doris password authentication and authorization.

LDAP authentication login details

When LDAP is enabled, users have the following in Doris and LDAP:

LDAP User	Doris User	Password	Login Status	Login to Doris users
Existent	Existent	LDAP Password	Login successful	Doris User
Existent	Existent	Doris Password	Login failure	None
Non-Existent	Existent	Doris Password	Login successful	Doris User
Existent	Non-Existent	LDAP Password	Login successful	Ldap Temporary user

After LDAP is enabled, when a user logs in using mysql client, Doris will first verify the user's password through the LDAP service, and if the LDAP user exists and the password is correct, Doris will use the user to log in; at this time, if the corresponding account exists, Doris will directly log in to the account, and if the corresponding account does not exist, it will create a temporary account for the user and log in to the account. The temporary account has the appropriate pair of permissions (see LDAP Group Authorization) and is only valid for the current connection. doris does not create the user and does not generate metadata for creating the user pair. If no login user exists in the LDAP service, Doris is used for password authentication.

The following assumes that LDAP authentication is enabled, ldap_user_filter = (&(uid={login})) is configured, and all other configuration items are correct, and the client sets the environment variable LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN=1

For example:

1:Accounts exist in both Doris and LDAP.

Doris account exists: jack@'172.10.1.10', password: 123456
LDAP user node presence attribute: uid: jack user password: abcdef

The jack@‘172.10.1.10’ account can be logged into by logging into Doris using the following command:

```
mysql -hDoris_HOST -PDoris_PORT -ujack -p abcdef
```

Login will fail with the following command:

```
mysql -hDoris_HOST -PDoris_PORT -ujack -p 123456
```

2:The user exists in LDAP and the corresponding account does not exist in Doris.

LDAP user node presence attribute: uid: jack User password: abcdef
Use the following command to create a temporary user and log in to jack@‘%’, the temporary user has basic privileges DatabasePrivs: Select_priv, Doris will delete the temporary user after the user logs out and logs in:

```
mysql -hDoris_HOST -PDoris_PORT -ujack -p abcdef
```

3:LDAP does not exist for the user.

Doris account exists: jack@‘172.10.1.10’, password: 123456
Login to the account using the Doris password, successfully:

```
mysql -hDoris_HOST -PDoris_PORT -ujack -p 123456
```

LDAP group authorization details

If a DLAP user dn is the “member” attribute of an LDAP group node, Doris assumes that the user belongs to the group. Doris will revoke the corresponding role privileges after the user logs out. Before using LDAP group authorization, you should create the corresponding role pairs in Doris and authorize the roles.

Login user Privileges are related to Doris user and group Privileges, as shown in the following table:
|LDAP Users|Doris Users|Login User Privileges| |–|–|–| |exist|exist|LDAP group Privileges + Doris user Privileges| |Does not exist|Exists|Doris user Privileges| |exist|non-exist|LDAP group Privileges|

If the logged-in user is a temporary user and no group permission exists, the user has the select_priv permission of the information_schema by default

Example:
LDAP user dn is the “member” attribute of the LDAP group node then the user is considered to belong to the group, Doris will intercept the first Rdn of group dn as the group name.
For example, if user dn is “uid=jack,ou=aidp,dc=domain,dc=com”, the group information is as follows:

```
dn: cn=doris_rd,ou=group,dc=domain,dc=com
objectClass: groupOfNames
member: uid=jack,ou=aidp,dc=domain,dc=com
```

Then the group name is doris_rd.

If jack also belongs to the LDAP groups doris_qa, doris_pm; Doris exists roles: doris_rd, doris_qa, doris_pm, after logging in using LDAP authentication, the user will not only have the original permissions of the account, but will also get the roles doris_rd, doris_qa and doris _pm privileges.

> Attention:
>
> The group to which user belongs is not related to the organizational structure of the LDAP tree, and user2 in the example section may not necessarily belong to group2 If you want user2 to belong to group2, you need to add user2 to the member attribute of group2

LDAP information cache

To avoid frequent access to LDAP service, Doris will cache LDAP information into memory, you can specify the cache time for LDAP users through the `ldap_user_cache_timeout_s` configuration item in ldap.conf, the default is 12 hours; after modifying the information in LDAP service or modifying the After modifying the information in the LDAP service or modifying the Role permissions of the LDAP user group, the cache may not take effect in time because of the cache, so you can refresh the cache with the refresh ldap statement, see REFRESH-LDAP.

Limitations of LDAP authentication

- The current LDAP feature of Doris only supports plaintext password authentication, that is, when a user logs in, the password is transmitted in plaintext between client and fe and between fe and LDAP service.

FAQ

- How to determine which roles an LDAP user has in Doris?

  Log in to Doris using an LDAP user, `show grants` Can view which roles the current user has. Among them, ldapDefaultRole is the default role that every ldap user has in Doris.

- How to troubleshoot when the roles of LDAP users in Doris are less than expected?

  1. Through 'show roles`Check if the expected role exists in Doris. If it does not exist, you need to use the 'CREATE ROLE role'_ Name` Create a character.
  2. Check if the expected group is in 'ldap'_ Group_ Based on the corresponding organizational structure.
  3. Check if the expected group contains the member attribute.
  4. Check if the member attribute of the expected group contains the current user.

Authority Management based on Apache Ranger

> **Apache Ranger is supported in Doris 2.1, please refer to the Doris 2.1 documentation.**

Memory Tracker

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

The Memory Tracker records the memory usage of the Doris BE process, including the memory used in the life cycle of tasks such as query, import, Compaction, and Schema Change, as well as various caches for memory control and analysis.

principle

Each query, import and other tasks in the system will create its own Memory Tracker when it is initialized, and put the Memory Tracker into TLS (Thread Local Storage) during execution, and each memory application and release of the BE process will be in the Mem Hook Consume the Memory Tracker in the middle, and display it after the final summary.

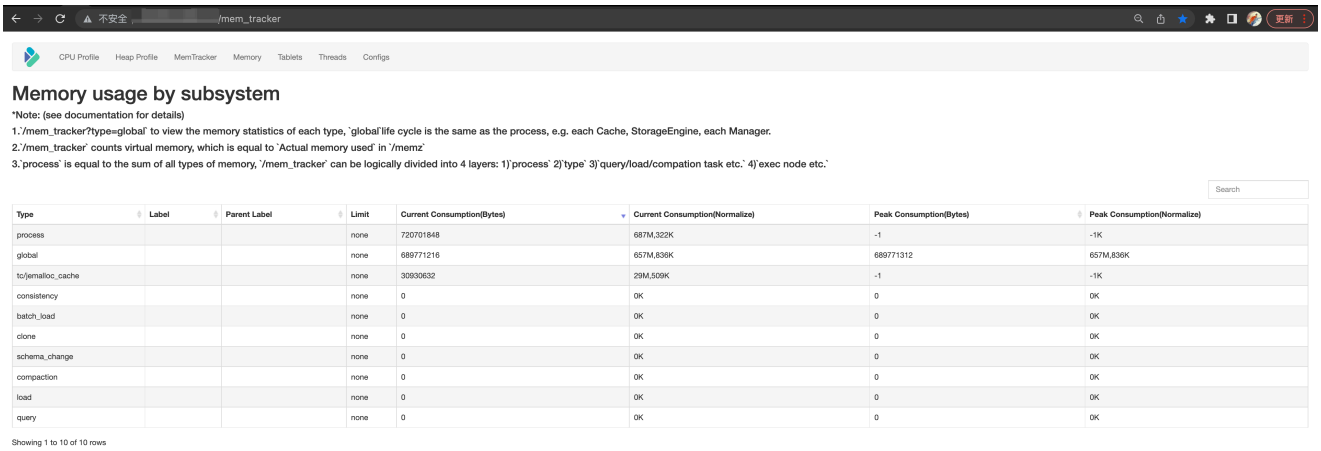
For detailed design and implementation, please refer to:
https://cwiki.apache.org/confluence/display/DORIS/DSIP-002%3A+Refactor+memory+tracker+on+BE
https://shimo.im/docs/DT6JXDRkdTvdyV3G

For more information about Memory Tracker, refer to the following documents: Say-Goodbye-to-OOM-Crashes-en, Say-Goodbye-to-OOM-Crashes-zh-CN

View statistics

The real-time memory statistics results can be viewed through Doris BE's Web page http://{be_host}:{be_web_server_port}/mem_tracker.(webserver_port default is 8040) For the memory statistics results of historical queries, you can view the `peakMemoryBytes` of each query in `fe/log/fe.audit.log`, or search `Deregister query/load memory tracker, queryId` in `be/log/be.INFO` `View memory peaks per query on a single BE.

Home `/mem_tracker`



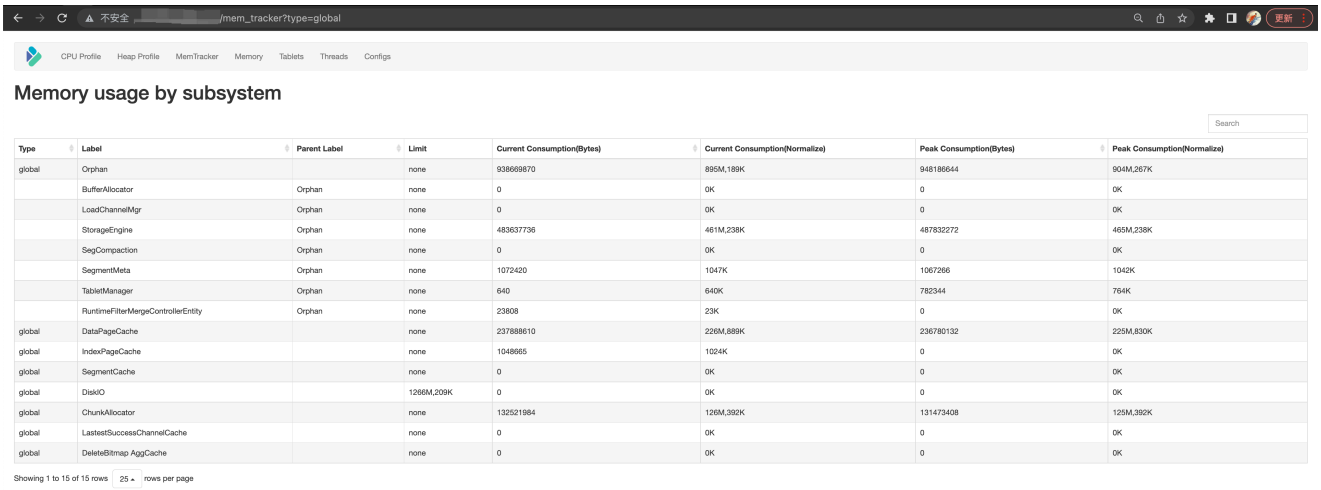
image

1. Type: Divide the memory used by Doris BE into the following categories

- process: The total memory of the process, the sum of all other types.
- global: Global Memory Tracker with the same life cycle and process, such as each Cache, Tablet Manager, Storage Engine, etc.
- query: the in-memory sum of all queries.
- load: Sum of all imported memory.
- tc/jemalloc_cache: The general memory allocator TCMalloc or Jemalloc cache, you can view the original profile of the memory allocator in real time at http://{be_host}:{be_web_server_port}/memz.
- compaction, schema_change, consistency, batch_load, clone: corresponding to the memory sum of all Compaction, Schema Change, Consistency, Batch Load, and Clone tasks respectively.

2. Current Consumption(Bytes): current memory value, unit B.
3. Current Consumption(Normalize): .G.M.K formatted output of the current memory value.
4. Peak Consumption (Bytes): The peak value of the memory after the BE process is started, the unit is B, and it will be reset after the BE restarts.
5. Peak Consumption(Normalize): The .G.M.K formatted output of the memory peak value after the BE process starts, and resets after the BE restarts.

Global Type `/mem_tracker?type=global`



Type	Label	Parent Label	Limit	Current Consumption(Bytes)	Current Consumption(Normalize)	Peak Consumption(Bytes)	Peak Consumption(Normalize)
global	Orphan		none	938669870	895M,189K	948186644	904M,267K
	BufferAllocator	Orphan	none	0	0K	0	0K
	LoadChannelMgr	Orphan	none	0	0K	0	0K
	StorageEngine	Orphan	none	483637736	461M,238K	487832272	465M,238K
	SegCompaction	Orphan	none	0	0K	0	0K
	SegmentMeta	Orphan	none	1072420	1047K	1067266	1042K
	TabletManager	Orphan	none	640	640K	782344	764K
	RuntimeFilterMergeControllerEntity	Orphan	none	23808	23K	0	0K
global	DataPageCache		none	237888610	226M,889K	236780132	225M,830K
global	IndexPageCache		none	1048665	1024K	0	0K
global	SegmentCache		none	0	0K	0	0K
global	DiskIO		1266M,209K	0	0K	0	0K
global	ChunkAllocator		none	132521984	126M,392K	131473408	125M,392K
global	LastestSuccessChannelCache		none	0	0K	0	0K
global	DeleteBitmap AggCache		none	0	0K	0	0K

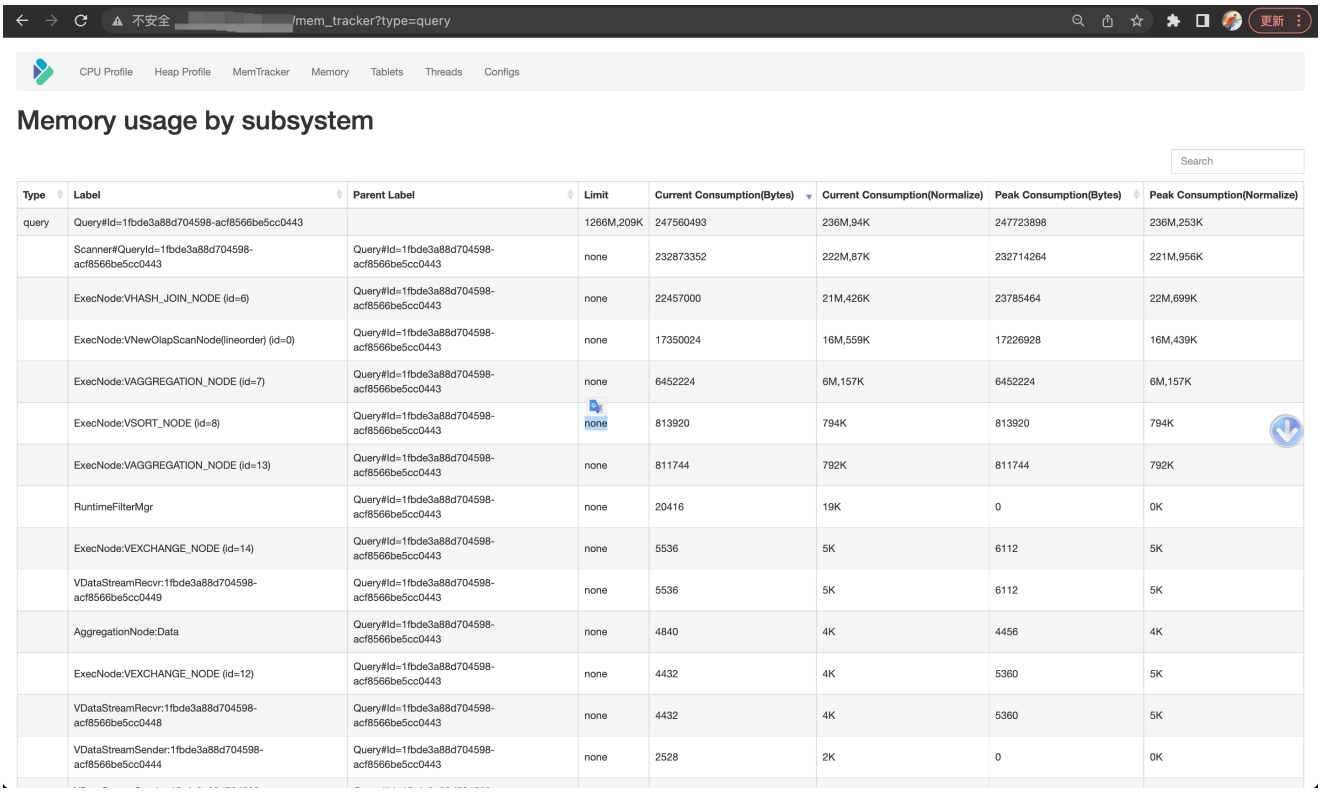
Showing 1 to 15 of 15 rows 25 rows per page

image

1. Label: Memory Tracker name
2. Parent Label: It is used to indicate the parent-child relationship between two Memory Trackers. The memory recorded by the Child Tracker is a subset of the Parent Tracker. There may be intersections between the memories recorded by different Trackers with the same Parent.

- Orphan: Tracker consumed by default. Memory that does not specify a tracker will be recorded in Orphan by default. In addition to the Child Tracker subdivided below, Orphan also includes some memory that is inconvenient to accurately subdivide and count, including BRPC.
    - LoadChannelMgr: The sum of the memory of all imported Load Channel stages, used to write the scanned data to the Segment file on disk, a subset of Orphan.
    - StorageEngine:, the memory consumed by the storage engine during loading the data directory, a subset of Orphan.
    - SegCompaction: The memory sum of all SegCompaction tasks, a subset of Orphan.
    - SegmentMeta: memory use by segment meta data such as footer or index page, a subset of Orphan.
    - TabletManager: The memory consumed by the storage engine get, add, and delete Tablet, a subset of Orphan.
- DataPageCache: Used to cache data Pages to speed up Scan.
- IndexPageCache: The index used to cache the data Page, used to speed up Scan.
- SegmentCache: Used to cache opened Segments, such as index information.
- ChunkAllocator: Used to cache power-of-2 memory blocks, and reuse memory at the application layer.
- LastestSuccessChannelCache: Used to cache the LoadChannel of the import receiver.
- DeleteBitmap AggCache: Gets aggregated delete_bitmap on rowset_id and version.

Query Type `/mem_tracker?type=query`

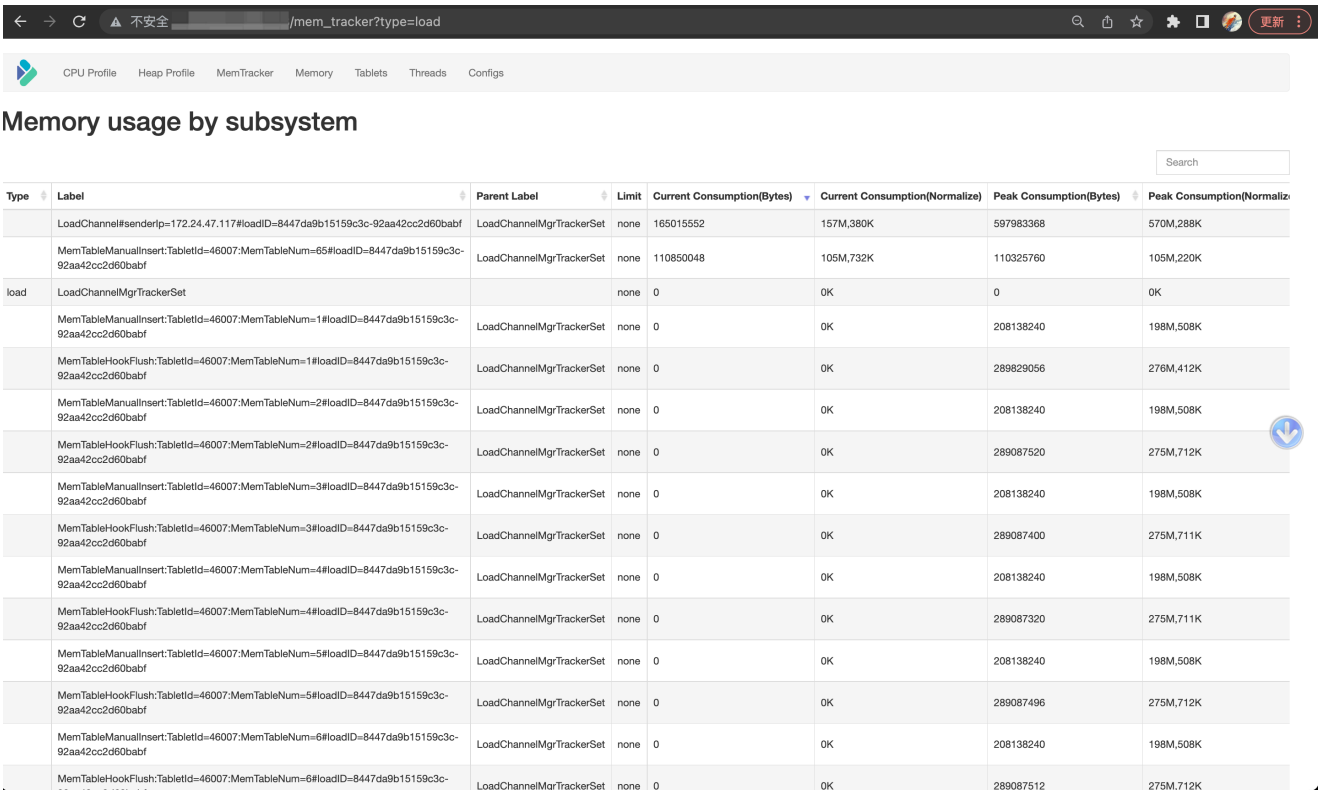


Type	Label	Parent Label	Limit	Current Consumption(Bytes)	Current Consumption(Normalize)	Peak Consumption(Bytes)	Peak Consumption(Normalize)
query	Query#Id=1fbde3a88d704598-acf8566be5cc0443		1266M,209K	247560493	236M,94K	247723898	236M,253K
	Scanner#QueryId=1fbde3a88d704598-acf8566be5cc0443	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	232873352	222M,87K	232714264	221M,956K
	ExecNode:VHASH_JOIN_NODE (id=6)	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	22457000	21M,426K	23785464	22M,699K
	ExecNode:VNewOlapScanNode(lineorder) (id=0)	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	17350024	16M,559K	17226928	16M,439K
	ExecNode:VAGGREGATION_NODE (id=7)	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	6452224	6M,157K	6452224	6M,157K
	ExecNode:VSORT_NODE (id=8)	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	813920	794K	813920	794K
	ExecNode:VAGGREGATION_NODE (id=13)	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	811744	792K	811744	792K
	RuntimeFilterMgr	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	20416	19K	0	0K
	ExecNode:VEXCHANGE_NODE (id=14)	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	5536	5K	6112	5K
	VDataStreamRecvr:1fbde3a88d704598-acf8566be5cc0449	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	5536	5K	6112	5K
	AggregationNode:Data	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	4840	4K	4456	4K
	ExecNode:VEXCHANGE_NODE (id=12)	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	4432	4K	5360	5K
	VDataStreamRecvr:1fbde3a88d704598-acf8566be5cc0448	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	4432	4K	5360	5K
	VDataStreamSender:1fbde3a88d704598-acf8566be5cc0444	Query#Id=1fbde3a88d704598-acf8566be5cc0443	none	2528	2K	0	0K

image

1. Limit: The upper limit of memory used by a single query, `show session variables` to view and modify `exec_mem_limit`.
2. Label: The label naming rule of the Tracker for a single query is `Query#Id=xxx`.
3. Parent Label: Parent is the Tracker record of `Query#Id=xxx` to query the memory used by different operators during execution.

Load Type `/mem_tracker?type=load`



Type	Label	Parent Label	Limit	Current Consumption(Bytes)	Current Consumption(Normalize)	Peak Consumption(Bytes)	Peak Consumption(Normaliz
	LoadChannel#senderIp=172.24.47.117#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	165015552	157M,380K	597983368	570M,288K
	MemTableManualInsert:TabletId=46007:MemTableNum=65#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	110850048	105M,732K	110325760	105M,220K
load	LoadChannelMgrTrackerSet		none	0	0K	0	0K
	MemTableManualInsert:TabletId=46007:MemTableNum=1#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	208138240	198M,508K
	MemTableHookFlush:TabletId=46007:MemTableNum=1#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	289829056	276M,412K
	MemTableManualInsert:TabletId=46007:MemTableNum=2#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	208138240	198M,508K
	MemTableHookFlush:TabletId=46007:MemTableNum=2#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	289087520	275M,712K
	MemTableManualInsert:TabletId=46007:MemTableNum=3#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	208138240	198M,508K
	MemTableHookFlush:TabletId=46007:MemTableNum=3#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	289087400	275M,711K
	MemTableManualInsert:TabletId=46007:MemTableNum=4#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	208138240	198M,508K
	MemTableHookFlush:TabletId=46007:MemTableNum=4#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	289087320	275M,711K
	MemTableManualInsert:TabletId=46007:MemTableNum=5#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	208138240	198M,508K
	MemTableHookFlush:TabletId=46007:MemTableNum=5#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	289087496	275M,712K
	MemTableManualInsert:TabletId=46007:MemTableNum=6#loadID=8447da9b15159c3c-92aa42cc2d60babf	LoadChannelMgrTrackerSet	none	0	0K	208138240	198M,508K
	MemTableHookFlush:TabletId=46007:MemTableNum=6#loadID=8447da9b15159c3c-	LoadChannelMgrTrackerSet	none	0	0K	289087512	275M,712K

image

1. Limit: The import is divided into two stages: Fragment Scan and Load Channel to write Segment to disk. The upper memory limit of the Scan phase can be viewed and modified through `show session variables`; the segment write disk phase does not have a separate memory upper limit for each import, but the total upper limit of all imports, corresponding to `load_process_max_memory_limit_percent`.
2. Label: The label naming rule of a single import Scan stage Tracker is `Load#Id=xxx`; the Label naming rule of a single import Segment write disk stage Tracker is `LoadChannel#senderIp=xxx#loadID=xxx`.
3. Parent Label: Parent is the Tracker of `Load#Id=xxx`, which records the memory used by different operators during the import Scan stage; Parent is the Tracker of `LoadChannelMgrTrackerSet`, which records the Insert and The memory used by the Flush disk process is associated with the last `loadID` of the Label to write to the disk stage Tracker of the Segment.

Memory Limit Exceeded Analysis

> **Tips**
>
> This featureis supported since the Apache Doris 1.2 version

When the query or import error `Memory limit exceeded` is reported, the possible reasons are: the process memory exceeds the limit, the remaining available memory of the system is insufficient, and the memory limit for a single query execution is exceeded.

```
ERROR 1105 (HY000): errCode = 2, detailMessage = Memory limit exceeded:<consuming tracker:<xxx>, xxx. backend 172.1.1.1 process memory used xxx GB, limit xxx GB. If query tracker exceeded, `set exec_mem_limit=8G ` to change limit, details mem usage see be.INFO.
```

For more information about memory management, refer to the following documents: Say-Goodbye-to-OOM-Crashes-en, Say-Goodbye-to-OOM-Crashes-zh-CN

The process memory exceeds the limit OR the remaining available memory of the system is insufficient

When the following error is returned, it means that the process memory exceeds the limit, or the remaining available memory of the system is insufficient. The specific reason depends on the memory statistics.

```
ERROR 1105 (HY000): errCode = 2, detailMessage = Memory limit exceeded:<consuming tracker:<Query#Id=3c88608cf35c461d-95fe88969aa6fc30>, process memory used 2.68 GB exceed limit 2.47 GB or sys mem available 50.95 GB less than low water mark 3.20 GB, failed alloc size 2.00 MB>, executing msg:<execute:<ExecNode:VAGGREGATION_NODE (id=7)>>. backend 172.1.1.1 process memory used 2.68 GB, limit 2.47 GB. If query tracker exceeded, `set exec_mem_limit =8G` to change limit, details mem usage see be.INFO
```

Error message analysis

The error message is divided into three parts: 1. `Memory limit exceeded:<consuming tracker:<Query#Id=3c88608cf35c461d-95fe88969aa6fc30>`: It is found that the memory limit is exceeded during the memory application process of query `3c88608cf35c461d-95fe88969aa6fc30`. 2. `process memory used 2.68 GB exceed limit 2.47 GB or sys mem available 50.95 GB less than low water mark 3.20 GB, failed alloc size 2.00 MB`: The reason for exceeding the limit is that the 2.68GB of memory used by the BE process exceeds the limit of 2.47GB limit, the value of limit comes from mem_limit * system MemTotal in be.conf, which is equal to 80% of the total memory of the operating system by default. The remaining available memory of the current operating system is 50.95 GB, which is still higher than the minimum water level of 3.2GB. This time, we are trying to apply for 2MB of memory. 3. `executing msg:<execute:<ExecNode:VAGGREGATION_NODE (id=7)>>, backend 172.24.47.117 process memory used 2.68 GB, limit 2.47 GB`: The location of this memory application is `ExecNode:VAGGREGATION_NODE (id=7)>`, the current IP of the BE node is 172.1.1.1, and print the memory statistics of the BE node again.

Log Analysis

At the same time, you can find the following log in log/be.INFO to confirm whether the memory usage of the current process meets expectations. The log is also divided into three parts: 1. `Process Memory Summary`: process memory statistics. 2. `Alloc Stacktrace`: The stack that triggers the memory overrun detection, which is not necessarily the location of the large memory application. 3. `Memory Tracker Summary`: Process memory tracker statistics, refer to Memory Tracker to analyze the location of memory usage. Notice: 1. The printing interval of the process memory overrun log is 1s. After the process memory exceeds the limit, the memory applications in most locations of BE will sense it, and try to make a predetermined callback method, and print the process memory overrun log, so if the log is If the value of Try Alloc is small, you don't need to pay attention to `Alloc Stacktrace`, just analyze `Memory Tracker Summary` directly. 2. When the process memory exceeds the limit, BE will trigger memory GC.

```
W1127 17:23:16.372572 19896 mem_tracker_limiter.cpp:214] System Mem Exceed Limit Check Failed, Try Alloc: 1062688
Process Memory Summary:
    process memory used 2.68 GB limit 2.47 GB, sys mem available 50.95 GB min reserve 3.20 GB, tc/jemalloc allocator cache 51.97 MB
Alloc Stacktrace:
    @          0x50028e8  doris::MemTrackerLimiter::try_consume()
    @          0x50027c1  doris::ThreadMemTrackerMgr::flush_untracked_mem<>()
    @          0x595f234  malloc
    @          0xb888c18  operator new()
    @          0x8f316a2  google::LogMessage::Init()
    @          0x5813fef  doris::FragmentExecState::coordinator_callback()
    @          0x58383dc  doris::PlanFragmentExecutor::send_report()
    @          0x5837ea8  doris::PlanFragmentExecutor::update_status()
    @          0x58355b0  doris::PlanFragmentExecutor::open()
    @          0x5815244  doris::FragmentExecState::execute()
    @          0x5817965  doris::FragmentMgr::_exec_actual()
    @          0x581fffb  std::_Function_handler<>::_M_invoke()
    @          0x5a6f2c1  doris::ThreadPool::dispatch_thread()
    @          0x5a6843f  doris::Thread::supervise_thread()
    @    0x7feb54f931ca  start_thread
    @    0x7feb5576add3  __GI___clone
    @             (nil)  (unknown)

Memory Tracker Summary:
    Type=consistency, Used=0(0 B), Peak=0(0 B)
    Type=batch_load, Used=0(0 B), Peak=0(0 B)
    Type=clone, Used=0(0 B), Peak=0(0 B)
    Type=schema_change, Used=0(0 B), Peak=0(0 B)
    Type=compaction, Used=0(0 B), Peak=0(0 B)
    Type=load, Used=0(0 B), Peak=0(0 B)
    Type=query, Used=206.67 MB(216708729 B), Peak=565.26 MB(592723181 B)
    Type=global, Used=930.42 MB(975614571 B), Peak=1017.42 MB(1066840223 B)
    Type=tc/jemalloc_cache, Used=51.97 MB(54494616 B), Peak=-1.00 B(-1 B)
    Type=process, Used=1.16 GB(1246817916 B), Peak=-1.00 B(-1 B)
    MemTrackerLimiter Label=Orphan, Type=global, Limit=-1.00 B(-1 B), Used=474.20 MB(497233597 B), Peak=649.18 MB(680718208 B)
    MemTracker Label=BufferAllocator, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=LoadChannelMgr, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=StorageEngine, Parent Label=Orphan, Used=320.56 MB(336132488 B), Peak=322.56 MB(338229824 B)
    MemTracker Label=SegCompaction, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=SegmentMeta, Parent Label=Orphan, Used=948.64 KB(971404 B), Peak=943.64 KB(966285 B)
    MemTracker Label=TabletManager, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTrackerLimiter Label=DataPageCache, Type=global, Limit=-1.00 B(-1 B), Used=455.22 MB(477329882 B), Peak=454.18 MB(476244180
B)
    MemTrackerLimiter Label=IndexPageCache, Type=global, Limit=-1.00 B(-1 B), Used=1.00 MB(1051092 B), Peak=0(0 B)
    MemTrackerLimiter Label=SegmentCache, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
    MemTrackerLimiter Label=DiskIO, Type=global, Limit=2.47 GB(2655423201 B), Used=0(0 B), Peak=0(0 B)
    MemTrackerLimiter Label=ChunkAllocator, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
```

```
    MemTrackerLimiter Label=LastestSuccessChannelCache, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
    MemTrackerLimiter Label=DeleteBitmap AggCache, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
```

System remaining available memory calculation

When the available memory of the system in the error message is less than the low water mark, it is also treated as a process memory limit. The value of the available memory of the system comes from `MemAvailable` in `/proc/meminfo`. When `MemAvailable` is insufficient, continue to use the memory The application may return std::bad_alloc or cause OOM of the BE process. Because both refreshing process memory statistics and BE memory GC have a certain lag, a small part of the memory buffer is reserved as a low water mark to avoid OOM as much as possible.

Among them, `MemAvailable` is the total amount of memory that the operating system can provide to the user process without triggering swap as much as possible given by the operating system considering the current free memory, buffer, cache, memory fragmentation and other factors. A simple calculation Formula: MemAvailable = MemFree - LowWaterMark + (PageCache - min(PageCache / 2, LowWaterMark)), which is the same as the `available` value seen by cmd `free`, for details, please refer to: https://serverfault.com/questions/940196/why-is-memaavailable-a-lot-less-than-memfreebufferscached https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=34e431b0ae398fc54ea69ff85ec700722c9da773

The low water mark defaults to a maximum of 1.6G, calculated based on `MemTotal`, `vm/min_free_kbytes`, `confg::mem_limit`, `config::max_sys_mem_available_low_water_mark_bytes`, and avoid wasting too much memory. Among them, `MemTotal` is the total memory of the system, and the value also comes from `/proc/meminfo`; `vm/min_free_kbytes` is the buffer reserved by the operating system for the memory GC process, and the value is usually between 0.4% and 5%. `vm/min_free_kbytes` may be 5% on some cloud servers, which will lead to visually that the available memory of the system is less than the real value; increasing `config::max_sys_mem_available_low_water_mark_bytes` will reserve more for Full GC on machines with more than 16G memory If there are more memory buffers, otherwise, the memory will be fully used as much as possible.

Query or import a single execution memory limit

When the following error is returned, it means that the memory limit of a single execution has been exceeded.

```
ERROR 1105 (HY000): errCode = 2, detailMessage = Memory limit exceeded:<consuming tracker:<Query#Id=f78208b15e064527-a84c5c0b04c04fcf>, failed alloc size 1.03 MB, exceeded tracker:<Query#Id=f78208b15e064527-a84c5c0b04c04fcf>, limit 100.00 MB, peak used 99.29 MB, current used 99.25 MB>, executing msg:<execute:<ExecNode:VHASH_JOIN_NODE (id=4)>>. backend 172.24.47.117 process memory used 1.13 GB, limit 98.92 GB. If query tracker exceed, `set exec_mem_limit=8G` to change limit, details mem usage see log/be.INFO
```

Error message analysis

The error message is divided into three parts: 1. `Memory limit exceeded:<consuming tracker:<Query#Id=f78208b15e064527-a84c5c0b04c04fcf>`: It is found that the memory limit is exceeded during the memory application process of query `f78208b15e064527-a84c5c0b04c04fcf`. 2. `failed alloc size 1.03 MB, exceeded tracker:<Query#Id=f78208b15e064527-a84c5c0b04c04fcf>, limit 100.00 MB, peak used 99.29 MB, current used 99.25 MB`: The memory requested this time is 1.03 MB The current consumption of `f78208b15e064527-a84c5c0b04c04fcf` memory tracker is 99.28MB plus 1.03MB, which exceeds the limit of 100MB. The value of limit comes from

exec_mem_limit in session variables, and the default is 4G. 3. executing msg:<execute:<ExecNode:VHASH_JOIN_NODE (id=4)>>. backend 172.24.47.117 process memory used 1.13 GB, limit 98.92 GB. If query tracker exceeds,set exec_mem_limit=8Gto change limit, details mem usage see be.INFO.: The location of this memory application is VHASH_JOIN_NODE (id=4), and it prompts that set exec_mem_limit can be used to increase the memory limit of a single query.

Log Analysis

After set global enable_profile=true, you can print a log in log/be.INFO when a single query memory exceeds the limit, to confirm whether the current query memory usage meets expectations. At the same time, you can find the following logs in log/be.INFO to confirm whether the current query memory usage meets expectations. The logs are also divided into three parts: 1. Process Memory Summary: process memory statistics. 2. Alloc Stacktrace: The stack that triggers the memory overrun detection, which is not necessarily the location of the large memory application. 3. Memory Tracker Summary: The memory tracker statistics of the current query, you can see the memory and peak value currently used by each operator. For details, please refer to Memory Tracker. Note: A query will only print the log once after the memory exceeds the limit. At this time, multiple threads of the query will sense it and try to wait for the memory to be released, or cancel the current query. If the value of Try Alloc in the log is small, there is no need to pay attentionAlloc Stacktrace, just analyze Memory Tracker Summary directly.

```
W1128 01:34:11.016165 357796 mem_tracker_limiter.cpp:191] Memory limit exceeded:<consuming tracker:<Query#Id=78208b15e064527-
a84c5c0b04c04fcf>, failed alloc size 4.00 MB, exceeded tracker:<Query#Id=78208b15e064527-a84c5c0b04c04fcf>, limit 100.00 MB, peak
used 98.59 MB,
current used 96.88 MB>, executing msg:<execute:<ExecNode:VHASH_JOIN_NODE (id=2)>>. backend 172.24.47.117 process memory used 1.13
GB, limit 98.92 GB. If query tracker exceed, `set exec_mem_limit=8G` to change limit, details mem usage see be.INFO.
Process Memory Summary:
    process memory used 1.13 GB limit 98.92 GB, sys mem available 45.15 GB min reserve 3.20 GB, tc/jemalloc allocator cache 27.62 MB
Alloc Stacktrace:
    @          0x66cf73a  doris::vectorized::HashJoinNode::_materialize_build_side()
    @          0x69cb1ee  doris::vectorized::VJoinNodeBase::open()
    @          0x66ce27a  doris::vectorized::HashJoinNode::open()
    @          0x5835dad  doris::PlanFragmentExecutor::open_vectorized_internal()
    @          0x58351d2  doris::PlanFragmentExecutor::open()
    @          0x5815244  doris::FragmentExecState::execute()
    @          0x5817965  doris::FragmentMgr::_exec_actual()
    @          0x581fffb  std::_Function_handler<>::_M_invoke()
    @          0x5a6f2c1  doris::ThreadPool::dispatch_thread()
    @          0x5a6843f  doris::Thread::supervise_thread()
    @     0x7f6faa94a1ca  start_thread
    @     0x7f6fab121dd3  __GI___clone
    @              (nil)  (unknown)

Memory Tracker Summary:
    MemTrackerLimiter Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Type=query, Limit=100.00 MB(104857600 B), Used=64.75
MB(67891182 B), Peak=104.70 MB(109786406 B)
    MemTracker Label=Scanner#QueryId=78208b15e064527-a84c5c0b04c04fcf, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=0(0 B), Peak=0(0 B)
    MemTracker Label=RuntimeFilterMgr, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=2.09 KB(2144 B), Peak=0(0 B)
    MemTracker Label=BufferedBlockMgr2, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=ExecNode:VHASH_JOIN_NODE (id=2), Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=-61.44
MB(-64426656 B), Peak=290.33 KB(297296 B)
    MemTracker Label=ExecNode:VEXCHANGE_NODE (id=9), Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=6.12 KB(6264 B),
Peak=5.84 KB(5976 B)
    MemTracker Label=VDataStreamRecvr:78208b15e064527-a84c5c0b04c04fd2, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=6.12 KB(6264 B), Peak=5.84 KB(5976 B)
    MemTracker Label=ExecNode:VEXCHANGE_NODE (id=10), Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=-41.20
MB(-43198024 B), Peak=1.46 MB(1535656 B)
```

```
    MemTracker Label=VDataStreamRecvr:78208b15e064527-a84c5c0b04c04fd2, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=-41.20 MB(-43198024 B), Peak=1.46 MB(1535656 B)
    MemTracker Label=VDataStreamSender:78208b15e064527-a84c5c0b04c04fd2, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=2.34 KB(2400 B), Peak=0(0 B)
    MemTracker Label=Scanner#QueryId=78208b15e064527-a84c5c0b04c04fcf, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=58.12 MB(60942224 B), Peak=57.41 MB(60202848 B)
    MemTracker Label=RuntimeFilterMgr, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=BufferedBlockMgr2, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=ExecNode:VNewOlapScanNode(customer) (id=1), Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=9.55
MB(10013424 B), Peak=10.20 MB(10697136 B)
    MemTracker Label=VDataStreamSender:78208b15e064527-a84c5c0b04c04fd1, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=59.80 MB(62701880 B), Peak=59.16 MB(62033048 B)
    MemTracker Label=Scanner#QueryId=78208b15e064527-a84c5c0b04c04fcf, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=0(0 B), Peak=0(0 B)
    MemTracker Label=RuntimeFilterMgr, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=13.62 KB(13952 B), Peak=0(0 B)
    MemTracker Label=BufferedBlockMgr2, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=ExecNode:VNewOlapScanNode(lineorder) (id=0), Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf, Used=6.03
MB(6318064 B), Peak=4.02 MB(4217664 B)
    MemTracker Label=VDataStreamSender:78208b15e064527-a84c5c0b04c04fd0, Parent Label=Query#Id=78208b15e064527-a84c5c0b04c04fcf,
Used=2.34 KB(2400 B), Peak=0(0 B)
```

BE OOM Analysis

BE OOM Analysis

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Ideally, in Memory Limit Exceeded Analysis, we regularly detect the remaining available memory of the operating system and respond in time when the memory is insufficient , such as triggering the memory GC to release the cache or cancel the memory overrun query, but because refreshing process memory statistics and memory GC both have a certain lag, and it is difficult for us to completely catch all large memory applications, there are still OOM risk.

Solution

Refer to BE Configuration Items to reduce `mem_limit` and increase `max_sys_mem_available_low_water_mark_bytes` in `be.conf`.

Memory analysis

If you want to further understand the memory usage location of the BE process before OOM and reduce the memory usage of the process, you can refer to the following steps to analyze.

1. `dmesg -T` confirms the time of OOM and the process memory at the time of OOM.
2. Check whether there is a `Memory Tracker Summary` log at the end of be/log/be.INFO. If it indicates that BE has detected memory overrun, go to step 3, otherwise go to step 8.

```
Memory Tracker Summary:
    Type=consistency, Used=0(0 B), Peak=0(0 B)
    Type=batch_load, Used=0(0 B), Peak=0(0 B)
    Type=clone, Used=0(0 B), Peak=0(0 B)
    Type=schema_change, Used=0(0 B), Peak=0(0 B)
    Type=compaction, Used=0(0 B), Peak=0(0 B)
    Type=load, Used=0(0 B), Peak=0(0 B)
    Type=query, Used=206.67 MB(216708729 B), Peak=565.26 MB(592723181 B)
    Type=global, Used=930.42 MB(975614571 B), Peak=1017.42 MB(1066840223 B)
    Type=tc/jemalloc_cache, Used=51.97 MB(54494616 B), Peak=-1.00 B(-1 B)
    Type=process, Used=1.16 GB(1246817916 B), Peak=-1.00 B(-1 B)
    MemTrackerLimiter Label=Orphan, Type=global, Limit=-1.00 B(-1 B), Used=474.20 MB(497233597 B), Peak=649.18 MB(680718208 B)
    MemTracker Label=BufferAllocator, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=LoadChannelMgr, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=StorageEngine, Parent Label=Orphan, Used=320.56 MB(336132488 B), Peak=322.56 MB(338229824 B)
    MemTracker Label=SegCompaction, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTracker Label=SegmentMeta, Parent Label=Orphan, Used=948.64 KB(971404 B), Peak=943.64 KB(966285 B)
    MemTracker Label=TabletManager, Parent Label=Orphan, Used=0(0 B), Peak=0(0 B)
    MemTrackerLimiter Label=DataPageCache, Type=global, Limit=-1.00 B(-1 B), Used=455.22 MB(477329882 B), Peak=454.18 MB(476244180
B)
```

```
MemTrackerLimiter Label=IndexPageCache, Type=global, Limit=-1.00 B(-1 B), Used=1.00 MB(1051092 B), Peak=0(0 B)
MemTrackerLimiter Label=SegmentCache, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
MemTrackerLimiter Label=DiskIO, Type=global, Limit=2.47 GB(2655423201 B), Used=0(0 B), Peak=0(0 B)
MemTrackerLimiter Label=ChunkAllocator, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
MemTrackerLimiter Label=LastestSuccessChannelCache, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
MemTrackerLimiter Label=DeleteBitmap AggCache, Type=global, Limit=-1.00 B(-1 B), Used=0(0 B), Peak=0(0 B)
```

3. When the end of be/log/be.INFO before OOM contains the system memory exceeded log, refer to Memory Limit Exceeded Analysis. The log analysis method in md looks at the memory usage of each category of the process. If the current `type=query` memory usage is high, if the query before OOM is known, continue to step 4, otherwise continue to step 5; if the current `type=load` memory usage is more, continue to step 6, if the current `type= Global`memory is used too much and continue to step 7.

4. `type=query` query memory usage is high, and the query before OOM is known, such as test cluster or scheduled task, restart the BE node, refer to Memory Tracker View real-time memory tracker statistics, retry the query after `set global enable_profile=true`, observe the memory usage location of specific operators, confirm whether the query memory usage is reasonable, and further consider optimizing SQL memory usage, such as adjusting the join order .

5. `type=query` query memory usage is high, and the query before OOM is unknown, such as in an online cluster, then search `Deregister query/load memory tracker from the back to the front` inbe/log/be.INFO, `queryId` and `Register query/load memory tracker, query/load id`, if the same query id prints the above two lines of logs at the same time, it means that the query or import is successful. If there is only Register but no Deregister, the query or import is still before OOM In this way, all running queries and imports before OOM can be obtained, and the memory usage of suspicious large-memory queries can be analyzed according to the method in step 4.

6. `type=load` imports a lot of memory.

7. When the `type=global` memory is used for a long time, continue to check the `type=global` detailed statistics in the second half of the `Memory Tracker Summary` log. When DataPageCache, IndexPageCache, SegmentCache, ChunkAllocator, LastestSuccessChannelCache, etc. use a lot of memory, refer to BE Configuration Item to consider modifying the size of the cache; when Orphan memory usage is too large, Continue the analysis as follows.

- If the sum of the tracker statistics of `Parent Label=Orphan` only accounts for a small part of the Orphan memory, it means that there is currently a large amount of memory that has no accurate statistics, such as the memory of the brpc process. At this time, you can consider using the heap profile Memory Tracker to further analyze memory locations.
- If the tracker statistics of `Parent Label=Orphan` account for most of Orphan's memory, when `Label=TabletManager` uses a lot of memory, further check the number of tablets in the cluster. If there are too many tablets, delete them and they will not be used table or data; when `Label=StorageEngine` uses too much memory, further check the number of segment files in the cluster, and consider manually triggering compaction if the number of segment files is too large;

8. If `be/log/be.INFO` does not print the `Memory Tracker Summary` log before OOM, it means that BE did not detect the memory limit in time, observe Grafana memory monitoring to confirm the memory growth trend of BE before OOM, if OOM is reproducible, consider adding `memory_debug=true` in `be.conf`, after restarting the cluster, the cluster memory statistics will be printed every second, observe the last `Memory Tracker Summary` log before OOM, and continue to step 3 for analysis;

Monitor Metrics

Monitor Metrics

Doris FE process and BE processes provide complete monitoring metrics. Monitoring metrics can be divided into two categories：

1. **Process monitoring**: mainly displays some monitoring values of the Doris process itself .
2. **Node monitoring**: mainly displays the monitoring of the node machine itself where the Doris process is located, such as CPU , memory, IO , network , etc.

You can obtain the current monitoring by accessing the http port of the FE or BE node . like :

```
curl http://fe_host:http_port/metrics
curl http://be_host:webserver_port/metrics
```

Monitoring metrics are generated in a Prometheus compatible format, for eg:

```
doris_fe_cache_added{type="partition"} 0
doris_fe_cache_added{type="sql"} 0
doris_fe_cache_hit{type="partition"} 0
doris_fe_cache_hit{type="sql"} 0
doris_fe_connection_total 2
```

Monitoring metrics in Json format can be fetched using `type` parameter in rest interface, for eg：

```
curl http://fe_host:http_port/metrics?type=json
curl http://be_host:webserver_port/metrics?type=json
```

Monitoring levels and best practices

The last column in the table marks the importance level of the monitoring items. P0 means the most important, the larger the value , the lower the importance.

The vast majority of monitoring metric types are Counter . That is the cumulative value. You can obtain effective information by collecting monitoring values at intervals (such as every 15 seconds) and calculating the slope per unit time .

The query error rate can be obtained by calculating the slope of `doris_fe_query_err`, presented as (error per second).

> 欢迎完善此表格以提供更全面有效的监控指标。

FE Monitoring Metrics

Process monitoring

name	Label	unit	Description	Implication	Grade
doris_fe_cache_added	{type="partition"}	Num	Cumulative value of the number of new Partition Cache		
	{type=" sql "}	Num	Cumulative value of the number of new SQL Cache		
doris_fe_cache_hit	{type="partition"}	Num	Count of partition cache hits		
	{type=" sql "}	Num	Count of SQL Cache hits		
doris_fe_connection_total		Num	Current number of FE MySQL port connections	Used to monitor the number of query connections. If the number of connections exceeds the limit, new connections will not be accessible .	P0
doris_fe_counter_hit_sql_block_rule		Num	Number of queries blocked by SQL BLOCK RULE		
doris_fe_edit_log_clean	{type="failed"}	Num	to clear historical metadata logs	It should not fail. If it fails, manual intervention is required.	P0
	{type="success"}	Num	The number of times the historical metadata log was successfully cleared		
doris_fe_edit_log	{type=" accumulated_bytes "}	byte	value of metadata log write amount	The write rate can be obtained by calculating the slope to observe whether there is a delay in metadata writing.	P0
	{type=" current_bytes "}	byte	Metadata log current value	Used to monitor editlog size. If the size exceeds the limit, manual intervention is required	P0
	{type="read"}	Num	Count of metadata log reads	Observe whether the metadata reading frequency is normal through the slope	P0
	{type="write"}	Num	Count of metadata log writes	Observe whether the metadata writing frequency is normal through the slope	P0
	{type="current"}	Num	Current number of metadata logs	Used to monitor editlog quantity. If the quantity exceeds the limit, manual intervention is required	P0
doris_fe_editlog_write_latency_ms		millisecond	metadata log write latency . For example, {quantile="0.75"} indicates the 75th percentile write latency .		

name	Label	unit	Description	Implication	Grade
doris_fe_image_clean	{type="failed"}	Num	to clean historical metadata image files	It should not fail. If it fails, manual intervention is required.	P0
	{type="success"}	Num	The number of times successful cleaning of historical metadata image files		
doris_fe_image_push	{type="failed"}	Num	The number of failures in pushing metadata image files to other FE nodes		
	{type="success"}	Num	metadata image files to other FE nodes		
doris_fe_image_write	{type="failed"}	Num	Number of failures to generate metadata image files	It should not fail. If it fails, manual intervention is required.	P0
	{type="success"}	Num	the metadata image file was successfully generated		
doris_fe_job		Num	Current count of different job types and different job statuses. For example, {job="load", type="INSERT", state="LOADING"} represents an import job of type INSERT and the number of jobs in the LOADING state .	observe the number of different types of jobs in the cluster as needed	P0
doris_fe_max_journal_id		Num	The maximum metadata log ID of the current FE node . If it is Master FE , it is the maximum ID currently written , if it is a non- Master FE , represents the maximum ID of the metadata log currently being played back	Used to observe whether the ID gap between multiple FEs is too large. If it is too large, it indicates a problem with metadata synchronization.	P0
doris_fe_max_tablet_compaction_score		Num	The largest compaction score value among all BE nodes .	This value can be used to observe the maximum compaction score of the current cluster to determine whether it is too high. If it is too high, query or write delays may occur.	P0
doris_fe_qps		Num/Sec	Current number of FE queries per second (only query requests are counted)	QPS	P0
doris_fe_query_err		Num	value of error query		
doris_fe_query_err_rate		Num/Sec	Error queries per second	Observe whether query errors occur in the cluster	P0
doris_fe_query_latency_ms		millisecond	Percentile statistics of query request latency. For example, {quantile="0.75"} indicates the query delay at the 75th percentile	Detailed observation of query latency in each quantile	P0

name	Label	unit	Description	Implication	Grade
doris_fe_query_latency_ms_db		millisecond	Percentile statistics of query request delay of each DB . For example, {quantile=“0.75”,db =“test”} indicates the query delay of the 75th percentile of DB test	Observe the query latency of each DB in detail	P0
doris_fe_query_olap_table		Num	the statistics of the number of requests for the internal table (OlapTable)		
doris_fe_query_total		Num	all query requests		
doris_fe_report_queue_size		Num	The queue length of various periodic reporting tasks of BE on the FE side	This value reflects the blocking degree of the reporting task on the Master FE node. The larger the value, the lower the processing capacity of the FE .	P0
doris_fe_request_total		Num	All operation requests received through the MySQL port (including queries and other statements)		
doris_fe_routine_load_error_rows		Num	Count the total number of error rows for all Routine Load jobs in the cluster		
doris_fe_routine_load_receive_bytes		byte	the amount of data received by all Routine Load jobs in the cluster		
doris_fe_routine_load_rows		Num	Count the number of data rows received by all Routine Load jobs in the cluster		
doris_fe_rps		Num	current number of FE requests per second (including queries and other types of statements)	Work with QPS to view the amount of requests processed by the cluster .	P0
doris_fe_scheduled_tablet_num		Num	tablets being scheduled by the Master FE node . Includes replicas being repaired and replicas being balanced	tablets being migrated . If there is a value for a long time, it means the cluster is unstable.	P0
doris_fe_tablet_max_compaction_score		Num	The compaction core reported by each BE node . For example, { backend=“172.21.0.1:9556”} represents the reported value of BE “172.21.0.1:9556”		
doris_fe_tablet_num		Num	current total number of tablets on each BE node . For example, {backend=“172.21.0.1:9556”} indicates the current number of tablets of the BE “172.21.0.1:9556”	You can check whether the tablet distribution is uniform and whether the absolute value is reasonable	P0

name	Label	unit	Description	Implication	Grade
doris_fe_tablet_status_count		Num	Statistics Master FE node The cumulative value of the number of tablets scheduled by the tablet scheduler .		
	{type=“added”}	Num	Statistics Master FE node The cumulative value of the number of tablets scheduled by the tablet scheduler . “added” indicates the number of tablets that have been scheduled		
	{type=” in_sched “}	Num	Same as above. Indicates the number of tablets that are scheduled repeatedly	If this value increases quickly, it means that a tablet has been in an unhealthy state for a long time , causing it to be scheduled repeatedly by the scheduler.	
	{type=” not_ready “}	Num	Same as above. Indicates the number of tablets that have not yet met the scheduling trigger conditions .	If this value increases quickly, it means that a large number of tablets are in an unhealthy state but cannot be scheduled.	
	{type=“total”}	Num	Same as above. Represents the cumulative number of tablets that have been checked (but not necessarily scheduled) .		
	{type=“unhealthy”}	Num	Same as above. Indicates the cumulative number of unhealthy tablets that have been checked .		
doris_fe_thread_pool		Num	Count the number of working threads and queuing status of various thread pools . “active_thread_num” Indicates the number of tasks being executed . “pool_size” Indicates the total number of threads in the thread pool . “task_in_queue” Indicates the number of tasks being queued		
	{name=“agent-task-pool”}	Num	Master FE is used to send Agent Task to BE ’s thread pool		
	{name=“connect-scheduler-check-timer”}	Num	pool for checking if MySQL idle connection has timed out		
	{name=“connect-scheduler-pool”}	Num	pool for receiving MySQL connection requests		
	{name=” mysql - nio - pool”}	Num	NIO MySQL Server thread pool for processing tasks		
	{name=“export-exporting-job-pool”}	Num	pool for export jobs in exporting state		

name	Label	unit	Description	Implication	Grade
	{name="export-pending-job-pool"}	Num	pool for export jobs in pending state		
	{name="heartbeat-mgr -pool"}	Num	Master FE is used to process the thread pool of each node's heartbeat		
	{name="loading-load-task-scheduler"}	Num	Master FE is used to schedule Broker In Load job, loading Task scheduling thread pool		
	{name="pending-load-task-scheduler"}	Num	Master FE is used to schedule Broker Load job , pending Task scheduling thread pool		
	{name="schema-change-pool"}	Num	pool used by Master FE to schedule schema change jobs		
	{name="thrift-server-pool"}	Num	worker thread pool of ThriftServer on the FE side . correspond fe.conf middle rpc_port . Used to interact with BE .		
doris_fe_txn_counter		Num	value of the number of imported transactions in each status	You can observe the execution of imported transactions .	P0
	{type="begin"}	Num	Number of transactions committed		
	{type="failed"}	Num	Number of failed transactions		
	{type="reject"}	Num	rejected transactions . (If the number of currently running transactions is greater than the threshold, new transactions will be rejected)		
	{type=" succes "}	Num	number of successful transactions		
doris_fe_txn_status		Num	Count the number of import transactions currently in various states. For example, {type="committed"} indicates the number of transactions in the committed state.	You can observe the number of imported transactions in each state to determine whether there is accumulation.	P0
doris_fe_query_instance_num		Num	Specifies the fragment that the user is currently requesting Number of instances . For example, {user=" test_u "} represents the user test_u The number of instances currently being requested	This value can be used to observe whether the specified user takes up too many query resources .	P0

name	Label	unit	Description	Implication	Grade
doris_fe_query_instance_begin		Num	Specify the fragment where the user request starts Number of instances . For example, {user=" test_u "} represents the user test_u Number of instances to start requesting	This value can be used to observe whether a given user has submitted too many queries.	P0
doris_fe_query_rpc_total		Num	of RPCs sent to the specified BE . For example, { be="192.168.10.1"} indicates the number of RPCs sent to BE with IP address 192.168.10.1	This value can be used to observe whether too many RPCs are submitted to a certain BE .	
doris_fe_query_rpc_failed		Num	RPC failures sent to the specified BE . For example, { be="192.168.10.1"} indicates the number of RPC failures sent to BE with IP address 192.168.10.1	This value can be used to observe whether a certain BE has RPC problems.	
doris_fe_query_rpc_size		Num	Specify the RPC data size of BE . For example, { be="192.168.10.1"} indicates the number of RPC data bytes sent to BE with IP address 192.168.10.1	This value can be used to observe whether an excessively large RPC is submitted to a BE .	
doris_fe_txn_exec_latency_ms		millisecond	Percentile statistics of transaction execution time. For example, {quantile="0.75"} indicates the 75th percentile transaction execution time	Observe the execution time of each digit transaction in detail	P0
doris_fe_txn_publish_latency_ms		millisecond	Percentile statistics of transaction publish time. For example, {quantile="0.75"} indicates that the 75th percentile transaction publish time is	of the publishing time of each quantile transaction	P0
doris_fe_txn_num		Num	Specifies the number of transactions being performed by the DB . For example, { db ="test"} indicates the number of transactions currently being executed by DB test .	This value can be used to observe whether a certain DB has submitted a large number of transactions .	P0
doris_fe_publish_txn_num		Num	Specify the number of transactions being published by the DB . For example, { db ="test"} indicates the number of transactions currently being published by DB test .	This value can be used to observe the number of publish transactions of a certain DB .	P0
doris_fe_txn_replica_num		Num	Specifies the number of replicas opened by the transaction being executed by the DB . For example, { db ="test"} indicates the number of copies opened by the transaction currently being executed by DB test .	This value can be used to observe whether a certain DB has too many copies opened, which may affect the execution of other transactions.	P0

name	Label	unit	Description	Implication	Grade
doris_fe_thrift_rpc_total		Num	RPC requests received by each method of the FE thrift interface . For example, {method=“report”} indicates the number of RPC requests received by the report method.	This value can observe a certain thrift rpc method payload	
doris_fe_thrift_rpc_latency_ms		millisecond	The RPC requests received by each method of the FE thrift interface take time. For example, {method=“report”} indicates that the RPC request received by the report method takes time.	This value can observe a certain thrift rpc method payload	
doris_fe_external_schema_cache	{ catalog =“hive”}	Num	SpecifyExternal Catalog _ The number of corresponding schema caches		
doris_fe_hive_meta_cache	{ catalog =“hive”}	Num			
	{type=” partition_value “}	Num	Specify External Hive Metastore Catalog The number of corresponding partition value caches		
	{type=“partition”}	Num	Specify External Hive Metastore Catalog The number of corresponding partition caches		
	{type=“file”}	Num	Specify External Hive Metastore Catalog The number of corresponding file caches		

JVM metrics

name	Label	unit	Description	Impact	Grade
jvm_heap_size_bytes		byte	JVM memory metrics. The tags include max, used, committed , corresponding to the maximum value, used and requested memory respectively.	Observe JVM memory usage	P0
jvm_non_heap_size_bytes		byte	JVM off-heap memory statistics		
jvm_old_gc			GC metrics .	GarbageCollector refers to a specific garbage collector	P0
	{type=“count”}	Num	Cumulative value of GC times		

name	Label	unit	Description	Impact	Grade
	{type="time"}	millisecond	Cumulative value of GC time consumption		
jvm_old_size_bytes		byte	JVM old generation memory statistics		P0
jvm_thread		Num	JVM thread count statistics	Observe whether the number of JVM threads is reasonable	P0
jvm_young_size_bytes		byte	JVM new generation memory statistics		P0

Machine metrics

name	Label	unit	Description	Impact	Grade
system_meminfo		byte	FE node machines. Collected from /proc/meminfo . include buffers , cached , memory_available , memory_free , memory_total		
system_snmp		FE node machines. Collected from /proc/net/ snmp .			
	{name=" tcp_in_errs "}	Num	tcp packet reception errors		
	{name=" tcp_in_segs "}	Num	tcp packets sent		
	{name=" tcp_out_segs "}	Num	tcp packets sent		
	{name=" tcp_retrans_segs "}	Num	Number of tcp packet retransmissions		

BE metrics metrics

Process metrics

name	Label	unit	Description	Impact	grade
doris_be_active_scan_context_count		Num	the number of scanners currently opened directly from the outside		
doris_be_add_batch_task_queue_size		Num	When recording import, the queue size of the thread pool that receives the batch	If it is greater than 0 , it means there is a backlog at the receiving end of the import task.	P0
agent_task_queue_size		Num	Display the length of each Agent Task processing queue, such as {type=“CREATE_TABLE”} Indicates the length of the CREATE_TABLE task queue		
doris_be_brpc_endpoint_stub_count		Num	Created _ The number of brpc stubs used for interaction between BEs		
doris_be_brpc_function_endpoint_stub_count		Num	Created _ The number of brpc stubs used to interact with Remote RPC		
doris_be_cache_capacity			Record the capacity of the specified LRU Cache		
doris_be_cache_usage			Record the usage of the specified LRU Cache	Used to observe memory usage	P0
doris_be_cache_usage_ratio			Record the usage of the specified LRU Cache		
doris_be_cache_lookup_count			Record the number of times the specified LRU Cache is searched		
doris_be_cache_hit_count			Record the number of hits in the specified LRU Cache		
doris_be_cache_hit_ratio			Record the hit rate of the specified LRU Cache	Used to observe whether the cache is effective	P0
	{name=” DataPageCache “}	Num	DataPageCache Data Page for caching data	Data Cache , directly affects query efficiency	P0
	{name=” IndexPageCache “}	Num	IndexPageCache Index Page for caching data	Index Cache , directly affects query efficiency	P0
	{name=” LastestSuccessChannelCache “}	Num	LastestSuccessChannelCache Used to cache import receivers LoadChannel		
	{name=” SegmentCache “}	Num	SegmentCache Used to cache turned on Segment , such as index information		
doris_be_chunk_pool_local_core_alloc_count		Num	ChunkAllocator , the number of times memory is allocated from the memory queue of the bound core		

name	Label	unit	Description	Impact	grade
doris_be_chunk_pool_other_core_alloc_count		Num	ChunkAllocator , the number of times memory is allocated from the memory queue of other cores		
doris_be_chunk_pool_reserved_bytes		byte	ChunkAllocator The amount of memory reserved in		
doris_be_chunk_pool_system_alloc_cost_ns		nanosecond	SystemAllocator The cumulative value of time spent applying for memory	The slope can be used to observe the time taken for memory allocation.	P0
doris_be_chunk_pool_system_alloc_count		Num	SystemAllocator Number of times to apply for memory		
doris_be_chunk_pool_system_free_cost_ns		nanosecond	SystemAllocator Cumulative value of time taken to release memory	The slope can be used to observe the time it takes to release memory.	P0
doris_be_chunk_pool_system_free_count		Num	SystemAllocator The number of times memory is released		
doris_be_compaction_bytes_total		byte	value of the amount of data processed by compaction	What is recorded is the input in the compaction task rowset The disk size . It can be observed through the slope rate of compaction	P0
	{type=“base”}	byte	Accumulated data volume of Base Compaction		
	{type=“cumulative”}	byte	The cumulative data volume of Cumulative Compaction		
doris_be_compaction_deltas_total		Num	processed by compaction rowset The cumulative value of the number	What is recorded is the input in the compaction task rowset of number	
	{type=“base”}	Num	Processed by Base Compaction rowset Cumulative number		
	{type=“cumulative”}	Num	Processed by Cumulative Compaction rowset Cumulative number		
doris_be_disks_compaction_num		Num	compaction tasks being executed on the specified data directory . like {path=“/path1/”} means /path1 The number of tasks being executed on the directory	Used to observe whether the number of compaction tasks on each disk is reasonable.	P0
doris_be_disks_compaction_score		Num	Specifies the number of compaction tokens being executed on the data directory. like {path=“/path1/”} means /path1 Number of tokens being executed on the directory		

name	Label	unit	Description	Impact	grade
doris_be_compaction_used_permits		Num	The number of tokens used by the Compaction task	Used to reflect the resource consumption of Compaction	
doris_be_compaction_waitting_permits		Num	Compaction tokens awaiting		
doris_be_data_stream_receiver_count		Num	Number of data receiving terminals Receiver	FIXME : This indicator is missing for the vectorization engine	
doris_be_disks_avail_capacity		byte	Specify the remaining space on the disk where the specified data directory is located. like {path="/path1/"} express /path1 The remaining space on the disk where the directory is located		P0
doris_be_disks_local_used_capacity		byte	the specified data directory is located		
doris_be_disks_remote_used_capacity		byte	the specified data directory is located		
doris_be_disks_state		Boolean	Specifies the disk status of the data directory . 1 means normal. 0 means abnormal		
doris_be_disks_total_capacity		byte	capacity of the disk where the specified data directory is located	Cooperate doris_be_disks_avail_capacity Calculate disk usage	P0
doris_be_engine_requests_total		Num	value of various task execution statuses on BE		
	{status=" failed ",type ="xxx"}	Num	value of the number of failures for tasks of type xxx		
	{status=" total ",type ="xxx"}	Num	The cumulative value of the total number of tasks of type xxx .	Can monitor the number of failures of various tasks on demand	P0
	{status=" skip ",type =" report_all_tablets "}	Num	value of the number of times xxx type tasks have been skipped		
doris_be_fragment_endpoint_count		Num	same	FIXME: Same as doris_be_data_stream_receiver_count number. And the vectorization engine is missing	
doris_be_fragment_request_duration_us		microseconds	All fragment intance The cumulative execution time of	the execution time of instance through slope	P0
doris_be_fragment_requests_total		Num	The cumulative number of executed fragment instances		

name	Label	unit	Description	Impact	grade
doris_be_load_channel_count		Num	The number of load channels currently open	The larger the value , the more import tasks are currently being executed.	P0
doris_be_local_bytes_read_total		byte	Depend on LocalFileReader Number of bytes read		P0
doris_be_local_bytes_written_total		byte	Depend on LocalFileWriter Number of bytes written		P0
doris_be_local_file_reader_total		Num	opened LocalFileReader Cumulative count of		
doris_be_local_file_open_reading		Num	currently open LocalFileReader number		
doris_be_local_file_writer_total		Num	opened LocalFileWriter cumulative count .		
doris_be_mem_consumption		byte	Specifies the current memory overhead of the module . For example, {type="compaction"} represents the current total memory overhead of the compaction module .	Values taken from the same type MemTracker . FIXME	
doris_be_memory_allocated_bytes		byte	BE process physical memory size, taken from /proc/self/status/ VmRSS		P0
doris_be_memory_jemalloc		byte	Jemalloc stats, taken from je_mallctl .	Meaningreference : https://jemalloc.net/jemalloc.3.html	P0
doris_be_memory_pool_bytes_total		byte	all MemPool The size of memory currently occupied. Statistical value, does not represent actual memory usage .		
doris_be_memtable_flush_duration_us		microseconds	value of the time taken to write memtable to disk	Write latency can be observed via slope	P0
doris_be_memtable_flush_total		Num	number of memtable writes to disk	The slope can be used to calculate the frequency of writing to a file	P0
doris_be_meta_request_duration		microseconds	access RocksDB The cumulative time consumption of meta in	BE metadata read and write latency through slope	P0

name	Label	unit	Description	Impact	grade
	{type="read"}	microseconds	Reading time		
	{type="write"}	microseconds	Writing time		
doris_be_meta_request_total		Num	access RocksDB The cumulative number of meta in	BE metadata access frequency by slope	P0
	{type="read"}	Num	Read times		
	{type="write"}	Num	Number of writes		
doris_be_fragment_instance_count		Num	The number of fragment instances currently received	Observe whether instance accumulation occurs	P0
doris_be_process_fd_num_limit_hard		Num	BE process. pass /proc/ pid /limits collection		
doris_be_process_fd_num_limit_soft		Num	BE process. pass /proc/ pid /limits collection		
doris_be_process_fd_num_used		Num	The number of file handles used by the BE process. pass /proc/ pid /limits collection		
doris_be_process_thread_num		Num	BE process threads. pass /proc/ pid /task collection		P0
doris_be_query_cache_memory_total_byte		byte	Number of bytes occupied by Query Cache		
doris_be_query_cache_partition_total_count		Num	current number of Partition Cache caches		
doris_be_query_cache_sql_total_count		Num	Current number of SQL Cache caches		
doris_be_query_scan_bytes		byte	Read the cumulative value of the data amount. Here we only count reads Olap The amount of data in the table		
doris_be_query_scan_bytes_per_second		Bytes/second	according to doris_be_query_scan_bytes Calculated read rate	Observe query rate	P0

name	Label	unit	Description	Impact	grade
doris_be_query_scan_rows		Num	Read the cumulative value of the number of rows. Here we only count reads Olap The amount of data in the table. and is RawRowsRead (Some data rows may be skipped by the index and not actually read, but will still be recorded in this value)	Query rate by slope observation	P0
doris_be_result_block_queue_count		Num	The number of fragment instances in the current query result cache	This queue is only used when being read directly by an external system. For example, Spark on Doris queries data through external scan.	
doris_be_result_buffer_block_count		Num	The number of queries in the current query result cache	This value reflects how many query results are currently waiting for FE consumption in BE .	P0
doris_be_routine_load_task_count		Num	The number of routine load tasks currently being executed		
doris_be_rowset_count_generated_and_in_use		Num	New and in use since the last startup The number of rowset ids .		
doris_be_s3_bytes_read_total		Num	S3FileReader The cumulative number of opens		
doris_be_s3_file_open_reading		Num	currently open S3FileReader number		
doris_be_s3_bytes_read_total		byte	S3FileReader Read the cumulative value of bytes		
doris_be_scanner_thread_pool_queue_size		Num	used for OlapScanner The current queued number of thread pools	it is greater than zero , it means that Scanner starts to accumulate.	P0
doris_be_segment_read	{type=” segment_read_total “}	Num	Cumulative value of the number of segments read		
doris_be_segment_read	{type=” segment_row_total “}	Num	value of the number of rows in the segment read	This value also includes the number of rows filtered by the index. Equivalent to the number of segments read * the total number of rows in each segment	
doris_be_send_batch_thread_pool_queue_size		Num	The number of queues in the thread pool used to send data packets when importing	it is greater than 0 , it means there is accumulation	P0
doris_be_send_batch_thread_pool_thread_num		Num	The number of threads in the thread pool used to send packets when importing		

name	Label	unit	Description	Impact	grade
doris_be_small_file_cache_count		Num	currently cached by BE		
doris_be_streaming_load_current_processing		Num	Number of stream load tasks currently running	Contains only tasks sent by the curl command	
doris_be_streaming_load_duration_ms		millisecond	The cumulative value of the execution time of all stream load tasks		
doris_be_streaming_load_requests_total		Num	value of the number of stream load tasks	Observable task submission frequency via slope	P0
doris_be_stream_load_pipe_count		Num	current stream load data pipelines	Including stream load and routine load tasks	
doris_be_stream_load	{type=" load_rows "}	Num	number of rows finally imported by stream load	Including stream load and routine load tasks	P0
doris_be_stream_load	{type=" receive_bytes "}	byte	value of the number of bytes received by stream load	Including stream load data received from http , and routine load from kafka read data	P0
doris_be_tablet_base_max_compaction_score		Num	The current largest Base Compaction Score	This value changes in real time, and peak data may be lost. The higher the value, the more serious the compaction accumulation is .	P0
doris_be_tablet_cumulative_max_compaction_score		Num	Same as above. Current largest Cumulative Compaction Score		
doris_be_tablet_version_num_distribution		Num	The histogram of the number of tablet versions .	Distribution used to reflect the number of tablet versions	P0
doris_be_thrift_connections_total		Num	thrift connections created . like {name="heartbeat"} Indicates the cumulative number of connections to the heartbeat service	This value is the connection of thrift server with BE as the server.	
doris_be_thrift_current_connections		Num	current number of thrift connections. like {name="heartbeat"} Indicates the current number of connections to the heartbeat service .	Same as above	
doris_be_thrift_opened_clients		Num	thrift clients currently open . like {name="frontend"} Indicates the number of clients accessing the FE service		

name	Label	unit	Description	Impact	grade
doris_be_thrift_used_clients		Num	thrift clients currently in use . like {name="frontend"} Indicates the number of clients being used to access the FE service		
doris_be_timeout_canceled_fragment_count		Num	Cumulative value of the number of fragment instances canceled due to timeout	This value may be recorded repeatedly. For example, some fragment instances are canceled multiple times	P0
doris_be_stream_load_txn_request	{type="begin"}	Num	value of the number of transactions started by stream load	Including stream load and routine load tasks	
doris_be_stream_load_txn_request	{type="commit"}	Num	value of the number of transactions successfully executed by stream load	Same as above	
doris_be_stream_load_txn_request	{type="rollback"}		value of the number of transactions that failed to execute stream load	Same as above	
doris_be_unused_rowsets_count		Num	The number of currently abandoned rowsets	These rowsets will be deleted regularly under normal circumstances.	
doris_be_upload_fail_count		Num	cumulative value of rowset failed to be uploaded to remote storage		
doris_be_upload_rowset_count		Num	cumulative number of rowsets successfully uploaded to remote storage		
doris_be_upload_total_byte			byte	value of rowset data successfully uploaded to remote storage	
doris_be_load_bytes		byte	Cumulative quantity sent through tablet sink	Observable amount of imported data	P0
doris_be_load_rows		Num	Cumulative number of rows sent through tablet sink	Observable amount of imported data	P0
fragment_thread_pool_queue_size		Num	current query execution thread pool waiting queue	If it is greater than zero, the query thread has been exhausted and queries will pile up.	P0
doris_be_all_rowsets_num		Num	All currently rowset number of		P0
doris_be_all_segments_num		Num	The number of all current segments		P0
doris_be_heavy_work_max_threads		Num	brpc Number of heavy thread pool threads		p0
doris_be_light_work_max_threads		Num	brpc Number of light thread pool threads		p0

name	Label	unit	Description	Impact	grade
doris_be_heavy_work_pool_queue_size		Num	brpc The maximum length of the heavy thread pool queue will block the submission of work if it exceeds it .		p0
doris_be_light_work_pool_queue_size		Num	brpc The maximum length of the light thread pool queue . If it exceeds the maximum length, the submission of work will be blocked.		p0
doris_be_heavy_work_active_threads		Num	brpc Number of active threads in heavy thread pool		p0
doris_be_light_work_active_threads		Num	brpc Number of active threads in light thread pool		p0

Machine metrics

name	Label	unit	Description	Impact	Grade
doris_be_cpu		Num	CPU related metrics metrics, from /proc/stat collection. Each value of each logical core will be collected separately . like {device="cpu0",mode="nice"} Indicates the nice value of cpu0	CPU usage can be calculated	P0
doris_be_disk_bytes_read		byte	The cumulative value of disk reads. from /proc/ diskstats collection. The values of each disk will be collected separately . like {device=" vdd "} express vvd disk value		
doris_be_disk_bytes_written		byte	The cumulative value of disk writes. The collection method is the same as above		
doris_be_disk_io_time_ms		byte	The collection method is the same as above	IO Util can be calculated	P0
doris_be_disk_io_time_weighted		byte	The collection method is the same as above		
doris_be_disk_reads_completed		byte	The collection method is the same as above		
doris_be_disk_read_time_ms		byte	The collection method is the same as above		
doris_be_disk_writes_completed		byte	The collection method is the same as above		

name	Label	unit	Description	Impact	Grade
doris_be_disk_write_time_ms		byte	The collection method is the same as above		
doris_be_fd_num_limit		Num	System file handle limit upper limit. from /proc/sys/fs/file-nr collection		
doris_be_fd_num_used		Num	The number of file handles used by the system . from /proc/sys/fs/file-nr collection		
doris_be_file_created_total		Num	Cumulative number of local file creation times	all calls local_file_writer And finally close the file count	
doris_be_load_average		Num	Machine Load Avg Metric metrics. For example, {mode=“15_minutes”} is 15 minutes Load Avg	Observe the overall machine load	P0
doris_be_max_disk_io_util_percent		percentage	value of the disk with the largest IO UTIL among all disks		P0
doris_be_max_network_receive_bytes_rate		Bytes/second	The maximum receive rate calculated among all network cards		P0
doris_be_max_network_send_bytes_rate		Bytes/second	The calculated maximum sending rate among all network cards		P0
doris_be_memory_pgpgin		byte	The amount of data written by the system from disk to memory page		
doris_be_memory_pgpgout		byte	The amount of data written to disk by system memory pages		
doris_be_memory_pswpin		byte	The number of times the system swapped from disk to memory	Normally, swap should be turned off, so this value should be 0	
doris_be_memory_pswpout		byte	The number of times the system swapped from memory to disk	Normally, swap should be turned off, so this value should be 0	
doris_be_network_receive_bytes		byte	each network card are accumulated. Collected from /proc/net/dev		
doris_be_network_receive_packets		Num	each network card is accumulated. Collected from /proc/net/dev		
doris_be_network_send_bytes		byte	each network card . Collected from /proc/net/dev		
doris_be_network_send_packets		Num	The total number of packets sent by each network card is accumulated. Collected from /proc/net/dev		
doris_be_proc	{mode=” ctxt_switch “}	Num	CPU context switches . Collected from /proc/stat	Observe whether there are abnormal context switches	P0

name	Label	unit	Description	Impact	Grade
doris_be_proc	{mode=“interrupt”}	Num	CPU interrupts . Collected from /proc/stat		
doris_be_proc	{mode=” procs_blocked “}	Num	The number of processes currently blocked in the system (such as waiting for IO). Collected from /proc/stat		
doris_be_proc	{mode=” procs_running “}	Num	The number of processes currently executing on the system . Collected from /proc/stat		
doris_be_snmp_tcp_in_errs		Num	tcp packet reception errors. Collected from /proc/net/ snmp	Observable network errors such as retransmission, packet loss, etc. Need and other snmp metrics used together with	P0
doris_be_snmp_tcp_in_segs		Num	tcp packets sent . Collected from /proc/net/ snmp		
doris_be_snmp_tcp_out_segs		Num	tcp packets sent. Collected from /proc/net/ snmp		
doris_be_snmp_tcp_retrans_segs		Num	TCP packet retransmissions . Collected from /proc/net/ snmp		

Monitoring and alarming

Monitoring and alarming

This document mainly introduces Doris's monitoring items and how to collect and display them. And how to configure alarm (TODO)

Dashboard template click download

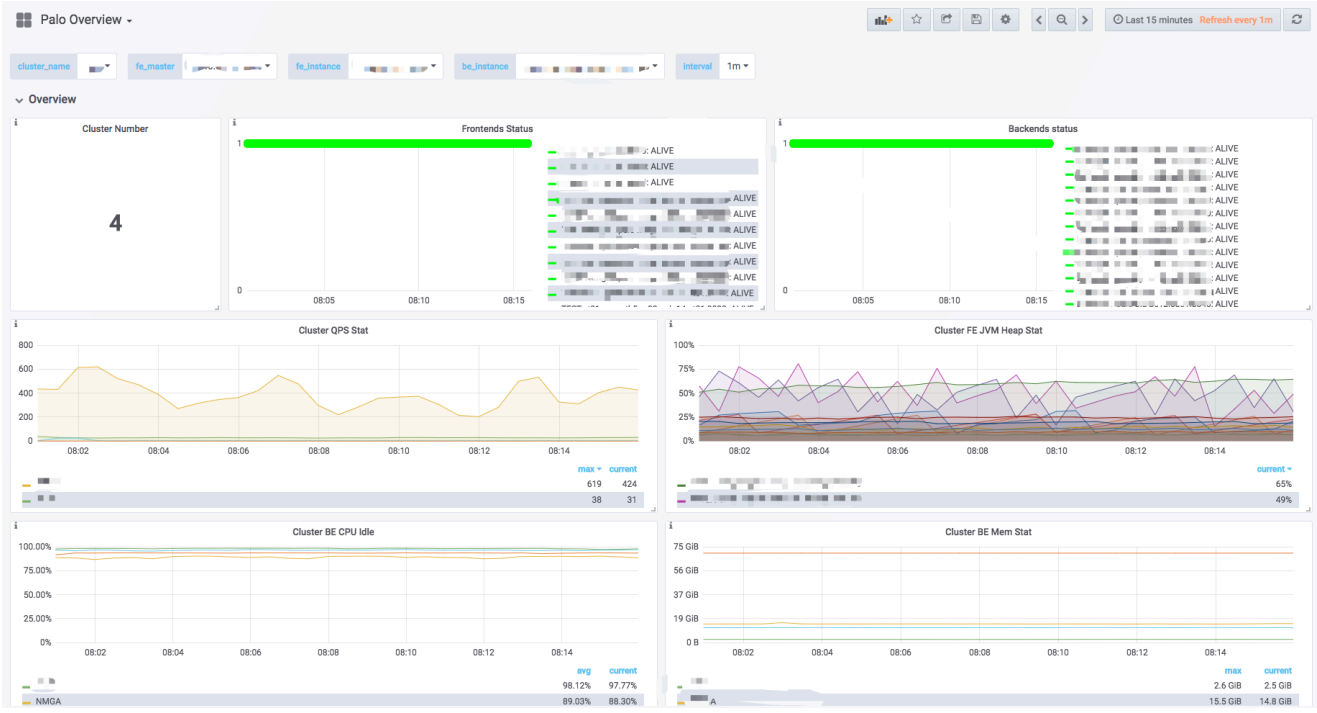
Doris Version	Dashboard Version
1.2.x	revision 5

Dashboard templates are updated from time to time. The way to update the template is shown in the last section.

Welcome to provide better dashboard.

Components

Doris uses Prometheus and Grafana to collect and display input monitoring items.



组件

1. Prometheus

   Prometheus is an open source system monitoring and alarm suite. It can collect monitored items by Pull or Push and store them in its own time series database. And through the rich multi-dimensional data query language, to meet the different data display needs of users.

2. Grafana

   Grafana is an open source data analysis and display platform. Support multiple mainstream temporal database sources including Prometheus. Through the corresponding database query statements, the display data is obtained from the data source. With flexible and configurable dashboard, these data can be quickly presented to users in the form of graphs.

> Note: This document only provides a way to collect and display Doris monitoring data using Prometheus and Grafana. In principle, these components are not developed or maintained. For more details on these components, please step through the corresponding official documents.

Monitoring data

Doris's monitoring data is exposed through the HTTP interface of Frontend and Backend. Monitoring data is presented in the form of key-value text. Each Key may also be distinguished by different Labels. When the user has built Doris, the monitoring data of the node can be accessed in the browser through the following interfaces:

- Frontend: `fe_host:fe_http_port/metrics`
- Backend: `be_host:be_web_server_port/metrics`
- Broker: Not available for now

Users will see the following monitoring item results (for example, FE partial monitoring items):

```
# HELP  jvm_heap_size_bytes jvm heap stat
# TYPE  jvm_heap_size_bytes gauge
jvm_heap_size_bytes{type="max"} 8476557312
jvm_heap_size_bytes{type="committed"} 1007550464
jvm_heap_size_bytes{type="used"} 156375280
# HELP  jvm_non_heap_size_bytes jvm non heap stat
# TYPE  jvm_non_heap_size_bytes gauge
jvm_non_heap_size_bytes{type="committed"} 194379776
jvm_non_heap_size_bytes{type="used"} 188201864
# HELP  jvm_young_size_bytes jvm young mem pool stat
# TYPE  jvm_young_size_bytes gauge
jvm_young_size_bytes{type="used"} 40652376
jvm_young_size_bytes{type="peak_used"} 277938176
jvm_young_size_bytes{type="max"} 907345920
# HELP  jvm_old_size_bytes jvm old mem pool stat
# TYPE  jvm_old_size_bytes gauge
jvm_old_size_bytes{type="used"} 114633448
jvm_old_size_bytes{type="peak_used"} 114633448
jvm_old_size_bytes{type="max"} 7455834112
# HELP  jvm_young_gc jvm young gc stat
# TYPE  jvm_young_gc gauge
jvm_young_gc{type="count"} 247
jvm_young_gc{type="time"} 860
# HELP  jvm_old_gc jvm old gc stat
# TYPE  jvm_old_gc gauge
jvm_old_gc{type="count"} 3
jvm_old_gc{type="time"} 211
# HELP  jvm_thread jvm thread stat
# TYPE  jvm_thread gauge
jvm_thread{type="count"} 162
jvm_thread{type="peak_count"} 205
```

```
jvm_thread{type="new_count"} 0
jvm_thread{type="runnable_count"} 48
jvm_thread{type="blocked_count"} 1
jvm_thread{type="waiting_count"} 41
jvm_thread{type="timed_waiting_count"} 72
jvm_thread{type="terminated_count"} 0
...
```

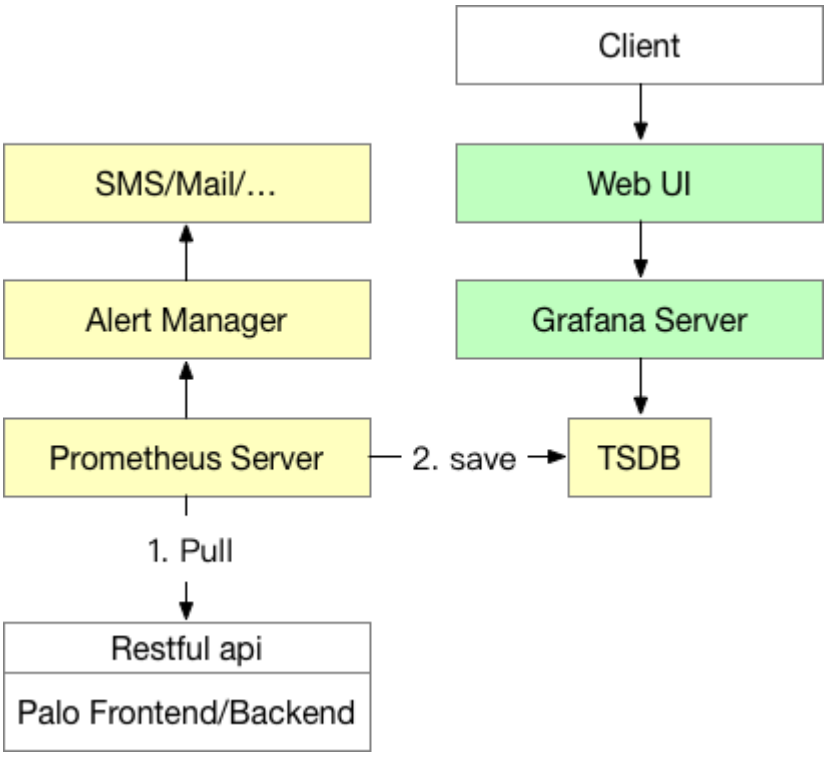
This is a monitoring data presented in Prometheus Format. We take one of these monitoring items as an example to illustrate:

```
# HELP  jvm_heap_size_bytes jvm heap stat
# TYPE  jvm_heap_size_bytes gauge
jvm_heap_size_bytes{type="max"} 8476557312
jvm_heap_size_bytes{type="committed"} 1007550464
jvm_heap_size_bytes{type="used"} 156375280
```

1. Behavior commentary line at the beginning of "#". HELP is the description of the monitored item; TYPE represents the data type of the monitored item, and Gauge is the scalar data in the example. There are also Counter, Histogram and other data types. Specifically, you can see Prometheus Official Document.
2. `jvm_heap_size_bytes` is the name of the monitored item (Key); `type= "max"` is a label named `type`, with a value of `max`. A monitoring item can have multiple Labels.
3. The final number, such as `8476557312`, is the monitored value.

Monitoring Architecture

The entire monitoring architecture is shown in the following figure:



Monitoring Architecture

1. The yellow part is Prometheus related components. Prometheus Server is the main process of Prometheus. At present, Prometheus accesses the monitoring interface of Doris node by Pull, and then

stores the time series data in the time series database TSDB (TSDB is included in the Prometheus process, and need not be deployed separately). Prometheus also supports building Push Gateway to allow monitored data to be pushed to Push Gateway by Push by monitoring system, and then data from Push Gateway by Prometheus Server through Pull.
2. Alert Manager is a Prometheus alarm component, which needs to be deployed separately (no solution is provided yet, but can be built by referring to official documents). Through Alert Manager, users can configure alarm strategy, receive mail, short messages and other alarms.
3. The green part is Grafana related components. Grafana Server is the main process of Grafana. After startup, users can configure Grafana through Web pages, including data source settings, user settings, Dashboard drawing, etc. This is also where end users view monitoring data.

Start building

Please start building the monitoring system after you have completed the deployment of Doris.

Prometheus

1. Download the latest version of Prometheus on the Prometheus Website. Here we take version 2.43.0-linux-amd64 as an example.

2. Unzip the downloaded tar file on the machine that is ready to run the monitoring service.

3. Open the configuration file prometheus.yml. Here we provide an example configuration and explain it (the configuration file is in YML format, pay attention to uniform indentation and spaces):

   Here we use the simplest way of static files to monitor configuration. Prometheus supports a variety of service discovery, which can dynamically sense the addition and deletion of nodes.

```
# my global config
global:
  scrape_interval:     15s # Global acquisition interval, default 1 m, set to 15s
  evaluation_interval: 15s # Global rule trigger interval, default 1 m, set 15s here

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'DORIS_CLUSTER' # Each Doris cluster, we call it a job. Job can be given a name here as the name of Doris
cluster in the monitoring system.
    metrics_path: '/metrics' # Here you specify the restful API to get the monitors. With host: port in the following
targets, Prometheus will eventually collect monitoring items through host: port/metrics_path.
    static_configs: # Here we begin to configure the target addresses of FE and BE, respectively. All FE and BE are written
into their respective groups.
      - targets: ['fe_host1:8030', 'fe_host2:8030', 'fe_host3:8030']
        labels:
          group: fe # Here configure the group of fe, which contains three Frontends

      - targets: ['be_host1:8040', 'be_host2:8040', 'be_host3:8040']
        labels:
          group: be # Here configure the group of be, which contains three Backends
```

```
  - job_name: 'DORIS_CLUSTER_2' # We can monitor multiple Doris clusters in a Prometheus, where we begin the configuration
of another Doris cluster. Configuration is the same as above, the following is outlined.
    metrics_path: '/metrics'
    static_configs:
      - targets: ['fe_host1:8030', 'fe_host2:8030', 'fe_host3:8030']
        labels:
          group: fe

      - targets: ['be_host1:8040', 'be_host2:8040', 'be_host3:8040']
        labels:
          group: be
```

4. start Prometheus

   Start Prometheus with the following command:

   `nohup ./prometheus --web.listen-address="0.0.0.0:8181" &`

   This command will run Prometheus in the background and specify its Web port as 8181. After startup, data is collected and stored in the data directory.

5. stop Promethues

   At present, there is no formal way to stop the process, kill - 9 directly. Of course, Prometheus can also be set as a service to start and stop in a service way.

6. access Prometheus

   Prometheus can be easily accessed through web pages. The page of Prometheus can be accessed by opening port 8181 through browser. Click on the navigation bar, `Status` -> `Targets`, and you can see all the monitoring host nodes of the grouped Jobs. Normally, all nodes should be `UP`, indicating that data acquisition is normal. Click on an `Endpoint` to see the current monitoring value. If the node state is not UP, you can first access Doris's metrics interface (see previous article) to check whether it is accessible, or query Prometheus related documents to try to resolve.

7. So far, a simple Prometheus has been built and configured. For more advanced usage, see Official Documents

Grafana

1. Download the latest version of Grafana on Grafana's official website. Here we take version 8.5.22.linux-amd64 as an example.
2. Unzip the downloaded tar file on the machine that is ready to run the monitoring service.
3. Open the configuration file conf/defaults.ini. Here we only list the configuration items that need to be changed, and the other configurations can be used by default.

```
# Path to where grafana can store temp files, sessions, and the sqlite3 db (if that is used)
data = data

# Directory where grafana can store logs
logs = data/log
```

```
# Protocol (http, https, socket)
protocal = http

# The ip address to bind to, empty will bind to all interfaces
http_addr =

# The http port to use
http_port = 8182
```

4. start Grafana

   Start Grafana with the following command

   `nohup ./bin/grafana-server &`

   This command runs Grafana in the background, and the access port is 8182 configured above.

5. stop Grafana

   At present, there is no formal way to stop the process, kill - 9 directly. Of course, you can also set Grafana as a service to start and stop as a service.

6. access Grafana

   Through the browser, open port 8182, you can start accessing the Grafana page. The default username password is admin.

7. Configure Grafana

   For the first landing, you need to set up the data source according to the prompt. Our data source here is Prometheus, which was configured in the previous step.

   The Setting page of the data source configuration is described as follows:

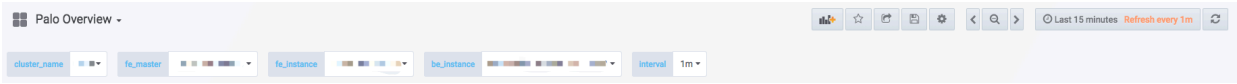
   1. Name: Name of the data source, customized, such as doris_monitor_data_source
   2. Type: Select Prometheus
   3. URL: Fill in the web address of Prometheus, such as http://host:8181
   4. Access: Here we choose the Server mode, which is to access Prometheus through the server where the Grafana process is located.
   5. The other options are available by default.
   6. Click `Save & Test` at the bottom. If `Data source is working`, it means that the data source is available.
   7. After confirming that the data source is available, click on the + number in the left navigation bar and start adding Dashboard. Here we have prepared Doris's dashboard template (at the beginning of this document). When the download is complete, click `New dashboard` -> `Import dashboard` -> `Upload.json File` above to import the downloaded JSON file.
   8. After importing, you can name Dashboard by default `Doris Overview`. At the same time, you need to select the data source, where you select the `doris_monitor_data_source` you created earlier.
   9. Click `Import` to complete the import. Later, you can see Doris's dashboard display.

8. So far, a simple Grafana has been built and configured. For more advanced usage, see Official Documents

Dashboard

Here we briefly introduce Doris Dashboard. The content of Dashboard may change with the upgrade of version. This document is not guaranteed to be the latest Dashboard description.

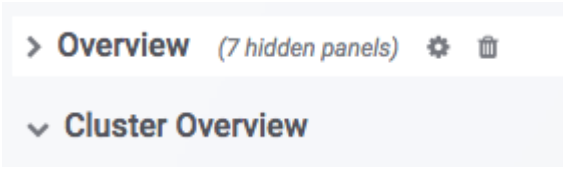
1. Top Bar



Top Bar

- The upper left corner is the name of Dashboard.
- The upper right corner shows the current monitoring time range. You can choose different time ranges by dropping down. You can also specify a regular refresh page interval.
- Cluster name: Each job name in the Prometheus configuration file represents a Doris cluster. Select a different cluster, and the chart below shows the monitoring information for the corresponding cluster.
- fe_master: The Master Frontend node corresponding to the cluster.
- fe_instance: All Frontend nodes corresponding to the cluster. Select a different Frontend, and the chart below shows the monitoring information for the Frontend.
- be_instance: All Backend nodes corresponding to the cluster. Select a different Backend, and the chart below shows the monitoring information for the Backend.
- Interval: Some charts show rate-related monitoring items, where you can choose how much interval to sample and calculate the rate (Note: 15s interval may cause some charts to be unable to display).

2. Row.

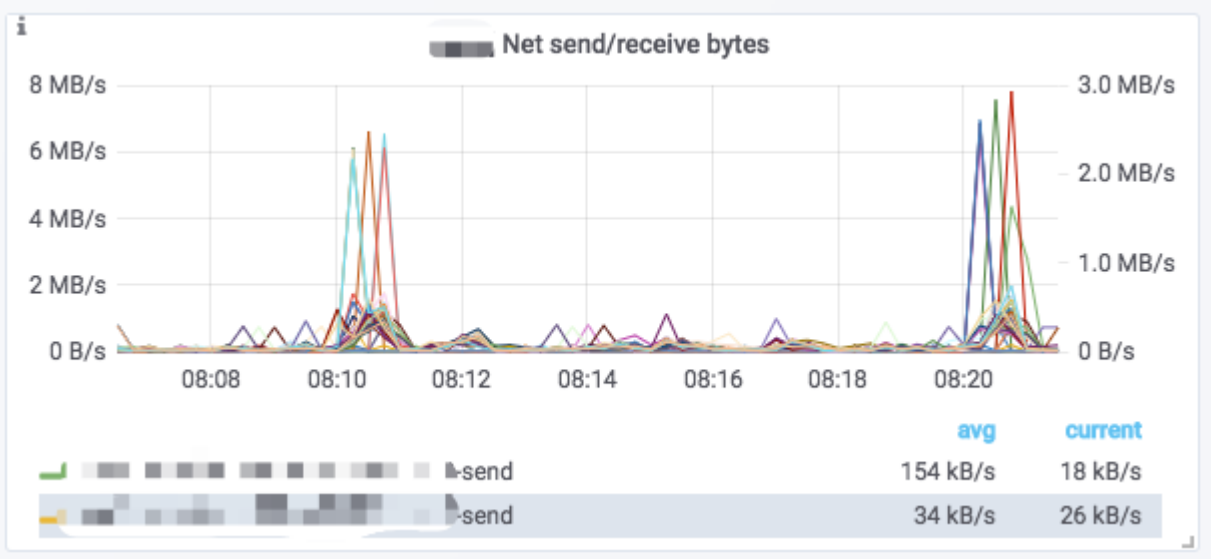


Row

In Grafana, the concept of Row is a set of graphs. As shown in the figure above, Overview and Cluster Overview are two different Rows. Row can be folded by clicking Row. Currently Dashboard has the following Rows (in continuous updates):

1. Overview: A summary display of all Doris clusters.
2. Cluster Overview: A summary display of selected clusters.
3. Query Statistic: Query-related monitoring of selected clusters.
4. FE JVM: Select Frontend's JVM monitoring.
5. BE: A summary display of the backends of the selected cluster.
6. BE Task: Display of Backends Task Information for Selected Clusters.

3. Charts



Charts

A typical icon is divided into the following parts:

1. Hover the I icon in the upper left corner of the mouse to see the description of the chart.
2. Click on the illustration below to view a monitoring item separately. Click again to display all.
3. Dragging in the chart can select the time range.
4. The selected cluster name is displayed in [] of the title.
5. Some values correspond to the Y-axis on the left and some to the right, which can be distinguished by the `-right` at the end of the legend.
6. Click on the name of the chart -> `Edit` to edit the chart.

Dashboard Update

1. Click on + in the left column of Grafana and `Dashboard`.
2. Click `New dashboard` in the upper left corner, and `Import dashboard` appears on the right.
3. Click `Upload .json File` to select the latest template file.
4. Selecting Data Sources
5. Click on `Import (Overwrite)` to complete the template update.

Disk Capacity Management

Disk Capacity Management

This document mainly introduces system parameters and processing strategies related to disk storage capacity.

If Doris' data disk capacity is not controlled, the process will hang because the disk is full. Therefore, we monitor the disk usage and remaining capacity, and control various operations in the Doris system by setting different warning levels, and try to avoid the situation where the disk is full.

Glossary

- Data Dir: Data directory, each data directory specified in the `storage_root_path` of the BE configuration file `be.conf`. Usually a data directory corresponds to a disk, so the following **disk** also refers to a data directory.

Basic Principles

BE will report disk usage to FE on a regular basis (every minute). FE records these statistical values and restricts various operation requests based on these statistical values.

Two thresholds, **High Watermark** and **Flood Stage**, are set in FE. Flood Stage is higher than High Watermark. When the disk usage is higher than High Watermark, Doris will restrict the execution of certain operations (such as replica balancing, etc.). If it is higher than Flood Stage, certain operations (such as load data) will be prohibited.

At the same time, a **Flood Stage** is also set on the BE. Taking into account that FE cannot fully detect the disk usage on BE in a timely manner, and cannot control certain BE operations (such as Compaction). Therefore, Flood Stage on the BE is used for the BE to actively refuse and stop certain operations to achieve the purpose of self-protection.

FE Parameter

High Watermark:

```
storage_high_watermark_usage_percent: default value is 85 (85%).
storage_min_left_capacity_bytes: default value is 2GB.
```

When disk capacity **more than** `storage_high_watermark_usage_percent`, **or** disk free capacity **less than** `storage_min_left_capacity_bytes`, the disk will no longer be used as the destination path for the following operations:

- Tablet Balance
- Colocation Relocation
- Decommission

Flood Stage:

```
storage_flood_stage_usage_percent: default value is 95 (95%).
storage_flood_stage_left_capacity_bytes: default value is 1GB.
```

When disk capacity **more than** `storage_flood_stage_usage_percent`, **or** disk free capacity **less than** `storage_flood_stage_left_capacity_bytes`, the disk will no longer be used as the destination path for the following operations:

- Tablet Balance
- Colocation Relocation
- Replica make up
- Restore
- Load/Insert

BE Parameter

Flood Stage:

```
storage_flood_stage_usage_percent: default value is 90 (90%).
storage_flood_stage_left_capacity_bytes: default value is 1GB.
```

When disk capacity **more than** `storage_flood_stage_usage_percent`, **and** disk free capacity **less than** `storage_flood_stage_left_capacity_bytes`, the following operations on this disk will be prohibited:

- Base/Cumulative Compaction
- Data load
- Clone Task (Usually occurs when the replica is repaired or balanced.)
- Push Task (Occurs during the Loading phase of Hadoop import, and the file is downloaded.)
- Alter Task (Schema Change or Rollup Task.)
- Download Task (The Downloading phase of the recovery operation.)

Disk Capacity Release

When the disk capacity is higher than High Watermark or even Flood Stage, many operations will be prohibited. At this time, you can try to reduce the disk usage and restore the system in the following ways.

- Delete table or partition

  By deleting tables or partitions, you can quickly reduce the disk space usage and restore the cluster.
  Note: Only the `DROP` operation can achieve the purpose of quickly reducing the disk space usage, the `DELETE` operation cannot.

  ```
  DROP TABLE tbl;
  ALTER TABLE tbl DROP PARTITION p1;
  ```

- BE expansion

After backend expansion, data tablets will be automatically balanced to BE nodes with lower disk usage. The expansion operation will make the cluster reach a balanced state in a few hours or days depending on the amount of data and the number of nodes.

- Modify replica of a table or partition

  You can reduce the number of replica of a table or partition. For example, the default 3 replica can be reduced to 2 replica. Although this method reduces the reliability of the data, it can quickly reduce the disk usage rate and restore the cluster to normal. This method is usually used in emergency recovery systems. Please restore the number of copies to 3 after reducing the disk usage rate by expanding or deleting data after recovery.
  Modifying the replica operation takes effect instantly, and the backends will automatically and asynchronously delete the redundant replica.

  ```
  ALTER TABLE tbl MODIFY PARTITION p1 SET("replication_num" = "2");
  ```

- Delete unnecessary files

  When the BE has crashed because the disk is full and cannot be started (this phenomenon may occur due to untimely detection of FE or BE), you need to delete some temporary files in the data directory to ensure that the BE process can start. Files in the following directories can be deleted directly:

  - log/: Log files in the log directory.
  - snapshot/: Snapshot files in the snapshot directory.
  - trash/ Trash files in the trash directory.

  This operation will affect Restore data from BE Recycle Bin.

  If the BE can still be started, you can use `ADMIN CLEAN TRASH ON(BackendHost:BackendHeartBeatPort);` to actively clean up temporary files. **all trash files** and expired snapshot files will be cleaned up, **This will affect the operation of restoring data from the trash bin**.

  If you do not manually execute `ADMIN CLEAN TRASH`, the system will still automatically execute the cleanup within a few minutes to tens of minutes.There are two situations as follows:

  - If the disk usage does not reach 90% of the **Flood Stage**, expired trash files and expired snapshot files will be cleaned up. At this time, some recent files will be retained without affecting the recovery of data.
  - If the disk usage has reached 90% of the **Flood Stage**, **all trash files** and expired snapshot files will be cleaned up, **This will affect the operation of restoring data from the trash bin**.

  The time interval for automatic execution can be changed by `max_garbage_sweep_interval` and `min_garbage_sweep_interval` in the configuration items.

  When the recovery fails due to lack of trash files, the following results may be returned:

  ```
  {"status": "Fail","msg": "can find tablet path in trash"}
  ```

- Delete data file (dangerous!!!)

When none of the above operations can free up capacity, you need to delete data files to free up space. The data file is in the `data/` directory of the specified data directory. To delete a tablet, you must first ensure that at least one replica of the tablet is normal, otherwise **deleting the only replica will result in data loss**.

Suppose we want to delete the tablet with id 12345:

- Find the directory corresponding to Tablet, usually under `data/shard_id/tablet_id/`. like:

  ```
  data/0/12345/
  ```

- Record the tablet id and schema hash. The schema hash is the name of the next-level directory of the previous step. The following is 352781111:

  ```
  data/0/12345/352781111
  ```

- Delete the data directory:

  ```
  rm -rf data/0/12345/
  ```

- Delete tablet metadata refer to Tablet metadata management tool

  ```
  ./lib/meta_tool --operation=delete_header --root_path=/path/to/root_path --tablet_id=12345 --schema_hash= 352781111
  ```

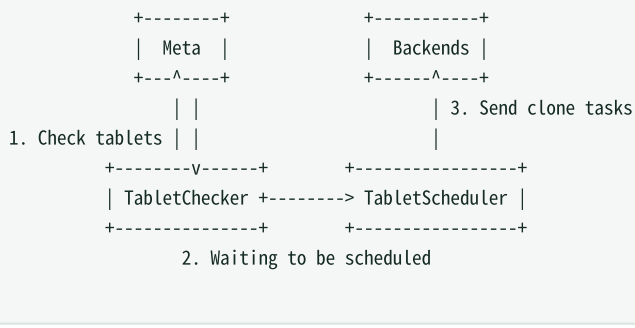
Data Replica Management

Beginning with version 0.9.0, Doris introduced an optimized replica management strategy and supported a richer replica status viewing tool. This document focuses on Doris data replica balancing, repair scheduling strategies, and replica management operations and maintenance methods. Help users to more easily master and manage the replica status in the cluster.

> Repairing and balancing copies of tables with Colocation attributes can be referred to HERE

Noun Interpretation

1. Tablet: The logical fragmentation of a Doris table, where a table has multiple fragments.
2. Replica: A sliced copy, defaulting to three copies of a slice.
3. Healthy Replica: A healthy copy that survives at Backend and has a complete version.
4. Tablet Checker (TC): A resident background thread that scans all Tablets regularly, checks the status of these Tablets, and decides whether to send them to Tablet Scheduler based on the results.
5. Tablet Scheduler (TS): A resident background thread that handles Tablets sent by Tablet Checker that need to be repaired. At the same time, cluster replica balancing will be carried out.
6. Tablet SchedCtx (TSC): is a tablet encapsulation. When TC chooses a tablet, it encapsulates it as a TSC and sends it to TS.
7. Storage Medium: Storage medium. Doris supports specifying different storage media for partition granularity, including SSD and HDD. The replica scheduling strategy is also scheduled for different storage media.



The figure above is a simplified workflow.

Duplicate status

Multiple copies of a Tablet may cause state inconsistencies due to certain circumstances. Doris will attempt to automatically fix the inconsistent copies of these states so that the cluster can recover from the wrong state as soon as possible.

The health status of a Replica is as follows:

1. BAD

   That is, the copy is damaged. Includes, but is not limited to, the irrecoverable damaged status of copies caused by disk failures, BUGs, etc.

2. VERSION_MISSING

   Version missing. Each batch of imports in Doris corresponds to a data version. A copy of the data consists of several consecutive versions. However, due to import errors, delays and other reasons, the data version of some copies may be incomplete.

3. HEALTHY

   Health copy. That is, a copy of the normal data, and the BE node where the copy is located is in a normal state (heartbeat is normal and not in the offline process).

The health status of a Tablet is determined by the status of all its copies. There are the following categories:

1. REPLICA_MISSING

   The copy is missing. That is, the number of surviving copies is less than the expected number of copies.

2. VERSION_INCOMPLETE

   The number of surviving copies is greater than or equal to the number of expected copies, but the number of healthy copies is less than the number of expected copies.

3. REPLICA_RELOCATING

   Have a full number of live copies of the replication num version, but the BE nodes where some copies are located are in unavailable state (such as decommission)

4. REPLICA_MISSING_IN_CLUSTER

   When using multi-cluster, the number of healthy replicas is greater than or equal to the expected number of replicas, but the number of replicas in the corresponding cluster is less than the expected number of replicas.

5. REDUNDANT

   Duplicate redundancy. Healthy replicas are in the corresponding cluster, but the number of replicas is larger than the expected number. Or there’s a spare copy of unavailable.

6. FORCE_REDUNDANT

   This is a special state. It only occurs when the number of existed replicas is greater than or equal to the number of available nodes, and the number of available nodes is greater than or equal to the number of expected replicas, and when the number of alive replicas is less than the number of expected replicas. In this case, you need to delete a copy first to ensure that there are available nodes for creating a new copy.

7. COLOCATE_MISMATCH

Fragmentation status of tables for Collocation attributes. Represents that the distribution of fragmented copies is inconsistent with the specified distribution of Colocation Group.

8. COLOCATE_REDUNDANT

   Fragmentation status of tables for Collocation attributes. Represents the fragmented copy redundancy of the Colocation table.

9. HEALTHY

   Healthy fragmentation, that is, conditions [1-5] are not satisfied.

Replica Repair

As a resident background process, Tablet Checker regularly checks the status of all fragments. For unhealthy fragmentation, it will be sent to Tablet Scheduler for scheduling and repair. The actual operation of repair is accomplished by clone task on BE. FE is only responsible for generating these clone tasks.

> Note 1: The main idea of replica repair is to make the number of fragmented replicas reach the desired value by creating or completing them first. Then delete the redundant copy.
>
> Note 2: A clone task is to complete the process of copying specified data from a specified remote end to a specified destination.

For different states, we adopt different repair methods:

1. REPLICA_MISSING/REPLICA_RELOCATING

   Select a low-load, available BE node as the destination. Choose a healthy copy as the source. Clone tasks copy a complete copy from the source to the destination. For replica completion, we will directly select an available BE node, regardless of the storage medium.

2. VERSION_INCOMPLETE

   Select a relatively complete copy as the destination. Choose a healthy copy as the source. The clone task attempts to copy the missing version from the source to the destination.

3. REPLICA_MISSING_IN_CLUSTER

   This state processing method is the same as REPLICA_MISSING.

4. REDUNDANT

   Usually, after repair, there will be redundant copies in fragmentation. We select a redundant copy to delete it. The selection of redundant copies follows the following priorities:

   1. The BE where the copy is located has been offline.
   2. The copy is damaged
   3. The copy is lost in BE or offline
   4. The replica is in the CLONE state (which is an intermediate state during clone task execution)
   5. The copy has version missing

6. The cluster where the copy is located is incorrect
7. The BE node where the replica is located has a high load

5. FORCE_REDUNDANT

   Unlike REDUNDANT, because at this point Tablet has a copy missing, because there are no additional available nodes for creating new copies. So at this point, a copy must be deleted to free up a available node for creating a new copy. The order of deleting copies is the same as REDUNDANT.

6. COLOCATE_MISMATCH

   Select one of the replica distribution BE nodes specified in Colocation Group as the destination node for replica completion.

7. COLOCATE_REDUNDANT

   Delete a copy on a BE node that is distributed by a copy specified in a non-Colocation Group.

   Doris does not deploy a copy of the same Tablet on a different BE of the same host when selecting a replica node. It ensures that even if all BEs on the same host are deactivated, all copies will not be lost.

Scheduling priority

Waiting for the scheduled fragments in Tablet Scheduler gives different priorities depending on the status. High priority fragments will be scheduled first. There are currently several priorities.

1. VERY_HIGH

   - REDUNDANT. For slices with duplicate redundancy, we give priority to them. Logically, duplicate redundancy is the least urgent, but because it is the fastest to handle and can quickly release resources (such as disk space, etc.), we give priority to it.
   - FORCE_REDUNDANT. Ditto.

2. HIGH

   - REPLICA_MISSING and most copies are missing (for example, 2 copies are missing in 3 copies)
   - VERSION_INCOMPLETE and most copies are missing
   - COLOCATE_MISMATCH We hope that the fragmentation related to the Collocation table can be repaired as soon as possible.
   - COLOCATE_REDUNDANT

3. NORMAL

   - REPLICA_MISSING, but most survive (for example, three copies lost one)
   - VERSION_INCOMPLETE, but most copies are complete
   - REPLICA_RELOCATING and relocate is required for most replicas (e.g. 3 replicas with 2 replicas)

4. LOW

   - REPLICA_MISSING_IN_CLUSTER
   - REPLICA_RELOCATING most copies stable

Manual priority

The system will automatically determine the scheduling priority. Sometimes, however, users want the fragmentation of some tables or partitions to be repaired faster. So we provide a command that the user can specify that a slice of a table or partition is repaired first:

```
ADMIN REPAIR TABLE tbl [PARTITION (p1, p2, ...)];
```

This command tells TC to give VERY HIGH priority to the problematic tables or partitions that need to be repaired first when scanning Tablets.

> Note: This command is only a hint, which does not guarantee that the repair will be successful, and the priority will change with the scheduling of TS. And when Master FE switches or restarts, this information will be lost.

Priority can be cancelled by the following commands:

```
ADMIN CANCEL REPAIR TABLE tbl [PARTITION (p1, p2, ...)];
```

Priority scheduling

Priority ensures that severely damaged fragments can be repaired first, and improves system availability. But if the high priority repair task fails all the time, the low priority task will never be scheduled. Therefore, we will dynamically adjust the priority of tasks according to the running status of tasks, so as to ensure that all tasks have the opportunity to be scheduled.

- If the scheduling fails for five consecutive times (e.g., no resources can be obtained, no suitable source or destination can be found, etc.), the priority will be lowered.
- If not scheduled for 30 minutes, priority will be raised.
- The priority of the same tablet task is adjusted at least five minutes apart.

At the same time, in order to ensure the weight of the initial priority, we stipulate that the initial priority is VERY HIGH, and the lowest is lowered to NORMAL. When the initial priority is LOW, it is raised to HIGH at most. The priority adjustment here also adjusts the priority set manually by the user.

Replicas Balance

Doris automatically balances replicas within the cluster. Currently supports two rebalance strategies, BeLoad and Partition. BeLoad rebalance will consider about the disk usage and replica count for each BE. Partition rebalance just aim at replica count for each partition, this helps to avoid hot spots. If you want high read/write performance, you may need this. Note that Partition rebalance do not consider about the disk usage, pay more attention to it when you are using Partition rebalance. The strategy selection config is not mutable at runtime.

BeLoad

The main idea of balancing is to create a replica of some fragments on low-load nodes, and then delete the replicas of these fragments on high-load nodes. At the same time, because of the existence of different storage media, there may or may not exist one or two storage media on different BE nodes in the same cluster. We

require that fragments of storage medium A be stored in storage medium A as far as possible after equalization. So we divide the BE nodes of the cluster according to the storage medium. Then load balancing scheduling is carried out for different BE node sets of storage media.

Similarly, replica balancing ensures that a copy of the same table will not be deployed on the BE of the same host.

BE Node Load

We use Cluster LoadStatistics (CLS) to represent the load balancing of each backend in a cluster. Tablet Scheduler triggers cluster equilibrium based on this statistic. We currently calculate a load Score for each BE as the BE load score by using **disk usage** and **number of copies**. The higher the score, the heavier the load on the BE.

Disk usage and number of copies have a weight factor, which is **capacityCoefficient** and **replicaNumCoefficient**, respectively. The sum of them is **constant to 1**. Among them, capacityCoefficient will dynamically adjust according to actual disk utilization. When the overall disk utilization of a BE is below 50%, the capacityCoefficient value is 0.5, and if the disk utilization is above 75% (configurable through the FE configuration item `capacity_used_percent_high_water`), the value is 1. If the utilization rate is between 50% and 75%, the weight coefficient increases smoothly. The formula is as follows:

```
capacityCoefficient = 2 * Disk Utilization - 0.5
```

The weight coefficient ensures that when disk utilization is too high, the backend load score will be higher to ensure that the BE load is reduced as soon as possible.

Tablet Scheduler updates CLS every 20 seconds.

Partition

The main idea of `partition rebalancing` is to decrease the skew of partitions. The skew of the partition is defined as the difference between the maximum replica count of the partition over all bes and the minimum replica count over all bes.

So we only consider about the replica count, do not consider replica size(disk usage). To fewer moves, we use TwoDimensionalGreedyAlgo which two dims are cluster & partition. It prefers a move that reduce the skew of the cluster when we want to rebalance a max skew partition.

Skew Info

The skew info is represented by `ClusterBalanceInfo`. `partitionInfoBySkew` is a multimap which key is the partition's skew, so we can get max skew partitions simply. `beByTotalReplicaCount` is a multimap which key is the total replica count of the backend.

`ClusterBalanceInfo` is in CLS, updated every 20 seconds.

When get more than one max skew partitions, we random select one partition to calculate the move.

Equilibrium strategy

Tablet Scheduler uses Load Balancer to select a certain number of healthy fragments as candidate fragments for balance in each round of scheduling. In the next scheduling, balanced scheduling will be attempted based on these candidate fragments.

Resource control

Both replica repair and balancing are accomplished by replica copies between BEs. If the same BE performs too many tasks at the same time, it will bring a lot of IO pressure. Therefore, Doris controls the number of tasks that can be performed on each node during scheduling. The smallest resource control unit is the disk (that is, a data path specified in be.conf). By default, we configure two slots per disk for replica repair. A clone task occupies one slot at the source and one slot at the destination. If the number of slots is zero, no more tasks will be assigned to this disk. The number of slots can be configured by FE's `schedule_slot_num_per_path` parameter.

In addition, by default, we provide two separate slots per disk for balancing tasks. The purpose is to prevent high-load nodes from losing space by balancing because slots are occupied by repair tasks.

Tablet State View

Tablet state view mainly looks at the state of the tablet, as well as the state of the tablet repair and balancing tasks. Most of these states **exist only in** Master FE nodes. Therefore, the following commands need to be executed directly to Master FE.

Tablet state

1. Global state checking

   Through `SHOW PROC'/cluster_health/tablet_health';`commands can view the replica status of the entire cluster.

```
+-------+--------------------------------+-----------+------------+-------------------+----------------------+-------------
---------+--------------+----------------------------+-------------------------+-------------------+----------------------+-
--------------------+----------------------+------------------+-----------------------------+-----------------+-----------
--+------------+
| DbId  | DbName                         | TabletNum | HealthyNum | ReplicaMissingNum | VersionIncompleteNum |
ReplicaRelocatingNum | RedundantNum | ReplicaMissingInClusterNum | ReplicaMissingForTagNum | ForceRedundantNum |
ColocateMismatchNum | ColocateRedundantNum | NeedFurtherRepairNum | UnrecoverableNum | ReplicaCompactionTooSlowNum |
InconsistentNum | OversizeNum | CloningNum |
+-------+--------------------------------+-----------+------------+-------------------+----------------------+-------------
---------+--------------+----------------------------+-------------------------+-------------------+----------------------+-
--------------------+----------------------+------------------+-----------------------------+-----------------+-----------
--+------------+
| 10005 | default_cluster:doris_audit_db | 84        | 84         | 0                 | 0                    | 0
| 0            | 0                          | 0                       | 0               | 0                 | 0
| 0                    | 0               | 0                         | 0              | 0           | 0          |
| 13402 | default_cluster:ssb1           | 709       | 708        | 1                 | 0                    | 0
| 0            | 0                          | 0                       | 0               | 0                 | 0
| 0                    | 0               | 0                         | 0              | 0           | 0          |
| 10108 | default_cluster:tpch1          | 278       | 278        | 0                 | 0                    | 0
| 0            | 0                          | 0                       | 0               | 0                 | 0
| 0                    | 0               | 0                         | 0              | 0           | 0          |
| Total | 3                              | 1071      | 1070       | 1                 | 0                    | 0
| 0            | 0                          | 0                       | 0               | 0                 | 0
```

```
| 0                  | 0              | 0                          | 0              | 0           | 0           |
+-------+--------------------------------+-----------+------------+------------------+---------------------+-------------
---------+--------------+----------------------------+------------------------+------------------+---------------------+-
--------------------+----------------------+------------------+----------------------------+-----------------+-----------
--+------------+
```

The `HealthyNum` column shows how many Tablets are in a healthy state in the corresponding database. `ReplicaCompactionTooSlowNum` column shows how many Tablets are in a too many versions state in the corresponding database, `InconsistentNum` column shows how many Tablets are in an inconsistent replica state in the corresponding database. The last `Total` line counts the entire cluster. Normally `TabletNum` and `HealthyNum` should be equal. If it's not equal, you can further see which Tablets are there. As shown in the figure above, one table in the ssb1 database is not healthy, you can use the following command to see which one is.

```
SHOW PROC '/cluster_health/tablet_health/13402';
```

Among them `13402` is the corresponding DbId.

```
+----------------------+--------------------------+--------------------------+------------------+-------------------------
-------+-----------------------------+----------------------+------------------------+--------------------------+--------
-----------------+----------------------+--------------------------------+--------------------+-----------------+
| ReplicaMissingTablets | VersionIncompleteTablets | ReplicaRelocatingTablets | RedundantTablets |
ReplicaMissingInClusterTablets | ReplicaMissingForTagTablets | ForceRedundantTablets | ColocateMismatchTablets |
ColocateRedundantTablets | NeedFurtherRepairTablets | UnrecoverableTablets | ReplicaCompactionTooSlowTablets |
InconsistentTablets | OversizeTablets |
+----------------------+--------------------------+--------------------------+------------------+-------------------------
-------+-----------------------------+----------------------+------------------------+--------------------------+--------
-----------------+----------------------+--------------------------------+--------------------+-----------------+
| 14679                |                          |                          |                  |
|                            |                     |                       |                         |
|                      |                               |                   |                |
+----------------------+--------------------------+--------------------------+------------------+-------------------------
-------+-----------------------------+----------------------+------------------------+--------------------------+--------
-----------------+----------------------+--------------------------------+--------------------+-----------------+
```

The figure above shows the specific unhealthy Tablet ID (14679). Later we'll show you how to view the status of each copy of a specific Tablet.

2. Table (partition) level status checking

   Users can view the status of a copy of a specified table or partition through the following commands and filter the status through a WHERE statement. If you look at table tbl1, the state on partitions P1 and P2 is a copy of OK:

   ```
   ADMIN SHOW REPLICA STATUS FROM tbl1 PARTITION (p1, p2) WHERE STATUS = "OK";
   ```

   ```
   +----------+-----------+-----------+---------+-------------------+--------------------+------------------+------------+----
   --------+-------+--------+--------+
   | TabletId | ReplicaId | BackendId | Version | LastFailedVersion | LastSuccessVersion | CommittedVersion | SchemaHash |
   VersionNum | IsBad | State  | Status |
   +----------+-----------+-----------+---------+-------------------+--------------------+------------------+------------+----
   --------+-------+--------+--------+
   | 29502429 | 29502432  | 10006     | 2       | -1                | 2                  | 1                | -1         | 2
   | false | NORMAL | OK     |
   | 29502429 | 36885996  | 10002     | 2       | -1                | -1                 | 1                | -1         | 2
   | false | NORMAL | OK     |
   | 29502429 | 48100551  | 10007     | 2       | -1                | -1                 | 1                | -1         | 2
   ```

```
| false | NORMAL | OK     |
| 29502433 | 29502434  | 10001     | 2       | -1               | 2                 | 1               | -1         | 2
| false | NORMAL | OK     |
| 29502433 | 44900737  | 10004     | 2       | -1               | -1                | 1               | -1         | 2
| false | NORMAL | OK     |
| 29502433 | 48369135  | 10006     | 2       | -1               | -1                | 1               | -1         | 2
| false | NORMAL | OK     |
+----------+-----------+-----------+---------+------------------+-------------------+-----------------+------------+----
--------+-------+--------+--------+
```

The status of all copies is shown here. Where `IsBad` is listed as `true`, the copy is damaged. The `Status` column displays other states. Specific status description, you can see help through `HELP ADMIN SHOW REPLICA STATUS`.

The `ADMIN SHOW REPLICA STATUS`command is mainly used to view the health status of copies. Users can also view additional information about copies of a specified table by using the following commands:

```
SHOW TABLETS FROM tbl1;
```

```
+----------+-----------+-----------+------------+---------+-------------+-------------------+-----------------------+------
------------+----------------------+---------------+----------+----------+--------+-------------------------+--------------
+----------------------+--------------+----------------------+----------------------+----------------------+
| TabletId | ReplicaId | BackendId | SchemaHash | Version | VersionHash | LstSuccessVersion | LstSuccessVersionHash |
LstFailedVersion | LstFailedVersionHash | LstFailedTime | DataSize | RowCount | State  | LstConsistencyCheckTime |
CheckVersion |     CheckVersionHash | VersionCount | PathHash             | MetaUrl              | CompactionStatus     |
+----------+-----------+-----------+------------+---------+-------------+-------------------+-----------------------+------
------------+----------------------+---------------+----------+----------+--------+-------------------------+--------------
+----------------------+--------------+----------------------+----------------------+----------------------+
| 29502429 | 29502432  | 10006     | 1421156361 | 2       | 0           | 2                 | 0                     | -1
| 0                    | N/A           | 784      | 0        | NORMAL | N/A                    | -1           |     -1
| 2            | -5822326203532286804 | url                  | url                  |
| 29502429 | 36885996  | 10002     | 1421156361 | 2       | 0           | -1                | 0                     | -1
| 0                    | N/A           | 784      | 0        | NORMAL | N/A                    | -1           |     -1
| 2            | -1441285706148429853 | url                  | url                  |
| 29502429 | 48100551  | 10007     | 1421156361 | 2       | 0           | -1                | 0                     | -1
| 0                    | N/A           | 784      | 0        | NORMAL | N/A                    | -1           |     -1
| 2            | -4784691547051455525 | url                  | url                  |
+----------+-----------+-----------+------------+---------+-------------+-------------------+-----------------------+------
------------+----------------------+---------------+----------+----------+--------+-------------------------+--------------
+----------------------+--------------+----------------------+----------------------+----------------------+
```

The figure above shows some additional information, including copy size, number of rows, number of versions, where the data path is located.

> Note: The contents of the `State` column shown here do not represent the health status of the replica, but the status of the replica under certain tasks, such as CLONE, SCHEMA CHANGE, ROLLUP, etc.

In addition, users can check the distribution of replicas in a specified table or partition by following commands.

```
ADMIN SHOW REPLICA DISTRIBUTION FROM tbl1;
```

```
+-----------+------------+-------+---------+
| BackendId | ReplicaNum | Graph | Percent |
```

```
+-----------+------------+-------+---------+
| 10000     | 7          |       | 7.29 %  |
| 10001     | 9          |       | 9.38 %  |
| 10002     | 7          |       | 7.29 %  |
| 10003     | 7          |       | 7.29 %  |
| 10004     | 9          |       | 9.38 %  |
| 10005     | 11         | >     | 11.46 % |
| 10006     | 18         | >     | 18.75 % |
| 10007     | 15         | >     | 15.62 % |
| 10008     | 13         | >     | 13.54 % |
+-----------+------------+-------+---------+
```

Here we show the number and percentage of replicas of table tbl1 on each BE node, as well as a simple graphical display.

3. Tablet level status checking

   When we want to locate a specific Tablet, we can use the following command to view the status of a specific Tablet. For example, check the tablet with ID 2950253:

   ```
   SHOW TABLET 29502553;
   ```

```
+------------------------+-----------+---------------+-----------+----------+----------+-------------+----------+--------+-
-------------------------------------------------------------------------------+
| DbName                 | TableName | PartitionName | IndexName | DbId     | TableId  | PartitionId | IndexId  | IsSync |
DetailCmd                                                      |
+------------------------+-----------+---------------+-----------+----------+----------+-------------+----------+--------+-
-------------------------------------------------------------------------------+
| default_cluster:test   | test      | test          | test      | 29502391 | 29502428 | 29502427    | 29502428 | true   |
SHOW PROC '/dbs/29502391/29502428/partitions/29502427/29502428/29502553'; |
+------------------------+-----------+---------------+-----------+----------+----------+-------------+----------+--------+-
-------------------------------------------------------------------------------+
```

The figure above shows the database, tables, partitions, roll-up tables and other information corresponding to this tablet. The user can copy the command in the `DetailCmd` command to continue executing:

```
Show Proc'/DBS/29502391/29502428/Partitions/29502427/29502428/29502553;
```

```
+-----------+-----------+---------+-------------+-------------------+-----------------------+------------------+-----------
-----------+---------------+------------+----------+----------+--------+--------------+----------------------+
| ReplicaId | BackendId | Version | VersionHash | LstSuccessVersion | LstSuccessVersionHash | LstFailedVersion |
LstFailedVersionHash | LstFailedTime | SchemaHash | DataSize | RowCount | State  | IsBad | VersionCount | PathHash
|
+-----------+-----------+---------+-------------+-------------------+-----------------------+------------------+-----------
-----------+---------------+------------+----------+----------+--------+--------------+----------------------+
| 43734060  | 10004     | 2       | 0           | -1                | 0                     | -1               | 0
| N/A           | -1         | 784      | 0        | NORMAL | false | 2            | -8566523878520798656 |
| 29502555  | 10002     | 2       | 0           | 2                 | 0                     | -1               | 0
| N/A           | -1         | 784      | 0        | NORMAL | false | 2            | 1885826196444191611  |
| 39279319  | 10007     | 2       | 0           | -1                | 0                     | -1               | 0
| N/A           | -1         | 784      | 0        | NORMAL | false | 2            | 1656508631294397870  |
+-----------+-----------+---------+-------------+-------------------+-----------------------+------------------+-----------
-----------+---------------+------------+----------+----------+--------+--------------+----------------------+
```

The figure above shows all replicas of the corresponding Tablet. The content shown here is the same as `SHOW TABLETS FROM tbl1;`. But here you can clearly see the status of all copies of a specific Tablet.

Duplicate Scheduling Task

1. View tasks waiting to be scheduled

```
SHOW PROC '/cluster_balance/pending_tablets';
```

```
+----------+--------+-----------------+---------+----------+----------+-------+---------+--------+----------+---------+----
-----------------+---------------------+---------------------+----------+------+-------------+---------------+-------------
--------+------------+---------------------+--------+--------------------+------------------------------+
| TabletId | Type   | Status          | State   | OrigPrio | DynmPrio | SrcBe | SrcPath | DestBe | DestPath | Timeout |
Create              | LstSched            | LstVisit            | Finished | Rate | FailedSched | FailedRunning |
LstAdjPrio          | VisibleVer | VisibleVerHash      | CmtVer | CmtVerHash          | ErrMsg                       |
+----------+--------+-----------------+---------+----------+----------+-------+---------+--------+----------+---------+----
-----------------+---------------------+---------------------+----------+------+-------------+---------------+-------------
--------+------------+---------------------+--------+--------------------+------------------------------+
| 4203036  | REPAIR | REPLICA_MISSING | PENDING | HIGH     | LOW      | -1    | -1      | -1     | -1       | 0       |
2019-02-21 15:00:20 | 2019-02-24 11:18:41 | 2019-02-24 11:18:41 | N/A      | N/A  | 2           | 0             | 2019-02-
21 15:00:43 | 1          | 0                   | 2      | 0                   | unable to find source replica |
+----------+--------+-----------------+---------+----------+----------+-------+---------+--------+----------+---------+----
-----------------+---------------------+---------------------+----------+------+-------------+---------------+-------------
--------+------------+---------------------+--------+--------------------+------------------------------+
```

   The specific meanings of each column are as follows:

   - TabletId: The ID of the Tablet waiting to be scheduled. A scheduling task is for only one Tablet
   - Type: Task type, which can be REPAIR (repair) or BALANCE (balance)
   - Status: The current status of the Tablet, such as REPLICA_MISSING (copy missing)
   - State: The status of the scheduling task may be PENDING/RUNNING/FINISHED/CANCELLED/TIMEOUT/UNEXPECTED
   - OrigPrio: Initial Priority
   - DynmPrio: Current dynamically adjusted priority
   - SrcBe: ID of the BE node at the source end
   - SrcPath: hash value of the path of the BE node at the source end
   - DestBe: ID of destination BE node
   - DestPath: hash value of the path of the destination BE node
   - Timeout: When the task is scheduled successfully, the timeout time of the task is displayed here in units of seconds.
   - Create: The time when the task was created
   - LstSched: The last time a task was scheduled
   - LstVisit: The last time a task was accessed. Here “accessed” refers to the processing time points associated with the task, including scheduling, task execution reporting, and so on.
   - Finished: Task End Time
   - Rate: Clone Task Data Copy Rate
   - Failed Sched: Number of Task Scheduling Failures
   - Failed Running: Number of task execution failures
   - LstAdjPrio: Time of last priority adjustment
   - CmtVer/CmtVerHash/VisibleVer/VisibleVerHash: version information for clone tasks
   - ErrMsg: Error messages that occur when tasks are scheduled and run

2. View running tasks

```
SHOW PROC '/cluster_balance/running_tablets';
```

The columns in the result have the same meaning as `pending_tablets`.

3. View completed tasks

   ```
   SHOW PROC '/cluster_balance/history_tablets';
   ```

   By default, we reserve only the last 1,000 completed tasks. The columns in the result have the same meaning as `pending_tablets`. If `State` is listed as `FINISHED`, the task is normally completed. For others, you can see the specific reason based on the error information in the `ErrMsg` column.

Viewing Cluster Load and Scheduling Resources

1. Cluster load

   You can view the current load of the cluster by following commands:

   ```
   SHOW PROC '/cluster_balance/cluster_load_stat/location_default';
   ```

   First of all, we can see the division of different storage media:

   ```
   +---------------+
   | StorageMedium |
   +---------------+
   | HDD           |
   | SSD           |
   +---------------+
   ```

   Click on a storage medium to see the equilibrium state of the BE node that contains the storage medium:

   ```
   SHOW PROC '/cluster_balance/cluster_load_stat/location_default/HDD';
   ```

   ```
   +----------+-----------------+-----------+---------------+----------------+-------------+------------+----------+----------
   -+--------------------+-------+
   | BeId     | Cluster         | Available | UsedCapacity  | Capacity       | UsedPercent | ReplicaNum | CapCoeff | ReplCoeff
   | Score              | Class |
   +----------+-----------------+-----------+---------------+----------------+-------------+------------+----------+----------
   -+--------------------+-------+
   | 10003    | default_cluster | true      | 3477875259079 | 19377459077121 | 17.948      | 493477     | 0.5      | 0.5
   | 0.9284678149967587 | MID   |
   | 10002    | default_cluster | true      | 3607326225443 | 19377459077121 | 18.616      | 496928     | 0.5      | 0.5
   | 0.948660871419998  | MID   |
   | 10005    | default_cluster | true      | 3523518578241 | 19377459077121 | 18.184      | 545331     | 0.5      | 0.5
   | 0.9843539990641831 | MID   |
   | 10001    | default_cluster | true      | 3535547090016 | 19377459077121 | 18.246      | 558067     | 0.5      | 0.5
   | 0.9981869446537612 | MID   |
   | 10006    | default_cluster | true      | 3636050364835 | 19377459077121 | 18.764      | 547543     | 0.5      | 0.5
   | 1.0011489897614072 | MID   |
   | 10004    | default_cluster | true      | 3506558163744 | 15501967261697 | 22.620      | 468957     | 0.5      | 0.5
   | 1.0228319835582569 | MID   |
   | 10007    | default_cluster | true      | 4036460478905 | 19377459077121 | 20.831      | 551645     | 0.5      | 0.5
   | 1.057279369420761  | MID   |
   | 10000    | default_cluster | true      | 4369719923760 | 19377459077121 | 22.551      | 547175     | 0.5      | 0.5
   | 1.0964036415787461 | MID   |
   +----------+-----------------+-----------+---------------+----------------+-------------+------------+----------+----------
   -+--------------------+-------+
   ```

Some of these columns have the following meanings:

- Available: True means that BE heartbeat is normal and not offline.
- UsedCapacity: Bytes, the size of disk space used on BE
- Capacity: Bytes, the total disk space size on BE
- UsedPercent: Percentage, disk space utilization on BE
- ReplicaNum: Number of copies on BE
- CapCoeff/ReplCoeff: Weight Coefficient of Disk Space and Copy Number
- Score: Load score. The higher the score, the heavier the load.
- Class: Classified by load, LOW/MID/HIGH. Balanced scheduling moves copies from high-load nodes to low-load nodes

Users can further view the utilization of each path on a BE, such as the BE with ID 10001:

```
SHOW PROC '/cluster_balance/cluster_load_stat/location_default/HDD/10001';
```

```
+------------------+------------------+---------------+---------------+---------+--------+----------------------+
| RootPath         | DataUsedCapacity | AvailCapacity | TotalCapacity | UsedPct | State  | PathHash             |
+------------------+------------------+---------------+---------------+---------+--------+----------------------+
| /home/disk4/palo | 498.757 GB       | 3.033 TB      | 3.525 TB      | 13.94 % | ONLINE | 4883406271918338267  |
| /home/disk3/palo | 704.200 GB       | 2.832 TB      | 3.525 TB      | 19.65 % | ONLINE | -5467083960906519443 |
| /home/disk1/palo | 512.833 GB       | 3.007 TB      | 3.525 TB      | 14.69 % | ONLINE | -7733211489989964053 |
| /home/disk2/palo | 881.955 GB       | 2.656 TB      | 3.525 TB      | 24.65 % | ONLINE | 4870995507205544622  |
| /home/disk5/palo | 694.992 GB       | 2.842 TB      | 3.525 TB      | 19.36 % | ONLINE | 1916696897889786739  |
+------------------+------------------+---------------+---------------+---------+--------+----------------------+
```

The disk usage of each data path on the specified BE is shown here.

2. Scheduling resources

Users can view the current slot usage of each node through the following commands:

```
SHOW PROC '/cluster_balance/working_slots';
```

```
+----------+----------------------+------------+------------+-------------+----------------------+
| BeId     | PathHash             | AvailSlots | TotalSlots | BalanceSlot | AvgRate              |
+----------+----------------------+------------+------------+-------------+----------------------+
| 10000    | 8110346074333016794  | 2          | 2          | 2           | 2.459007474009069E7  |
| 10000    | -5617618290584731137 | 2          | 2          | 2           | 2.4730105014001578E7 |
| 10001    | 4883406271918338267  | 2          | 2          | 2           | 1.6711402709780257E7 |
| 10001    | -5467083960906519443 | 2          | 2          | 2           | 2.7540126380326536E7 |
| 10002    | 9137404661108133814  | 2          | 2          | 2           | 2.417217089806745E7  |
| 10002    | 1885826196444191611  | 2          | 2          | 2           | 1.6327378456676323E7 |
+----------+----------------------+------------+------------+-------------+----------------------+
```

In this paper, data path is used as granularity to show the current use of slot. Among them, `AvgRate'is the copy rate of clone task in bytes/seconds on the path of historical statistics.

3. Priority repair view

The following command allows you to view the priority repaired tables or partitions set by the `ADMIN REPAIR TABLE'command.

```
SHOW PROC '/cluster_balance/priority_repair';
```

Among them, `Remaining TimeMs'indicates that these priority fixes will be automatically removed from the priority fix queue after this time. In order to prevent resources from being occupied due to the failure of priority repair.

Scheduler Statistical Status View

We have collected some statistics of Tablet Checker and Tablet Scheduler during their operation, which can be viewed through the following commands:

```
SHOW PROC '/cluster_balance/sched_stat';
```

```
+-------------------------------------------------+-------------+
| Item                                            | Value       |
+-------------------------------------------------+-------------+
| num of tablet check round                       | 12041       |
| cost of tablet check(ms)                        | 7162342     |
| num of tablet checked in tablet checker         | 18793506362 |
| num of unhealthy tablet checked in tablet checker | 7043900     |
| num of tablet being added to tablet scheduler   | 1153        |
| num of tablet schedule round                    | 49538       |
| cost of tablet schedule(ms)                     | 49822       |
| num of tablet being scheduled                   | 4356200     |
| num of tablet being scheduled succeeded         | 320         |
| num of tablet being scheduled failed            | 4355594     |
| num of tablet being scheduled discard           | 286         |
| num of tablet priority upgraded                 | 0           |
| num of tablet priority downgraded               | 1096        |
| num of clone task                               | 230         |
| num of clone task succeeded                     | 228         |
| num of clone task failed                        | 2           |
| num of clone task timeout                       | 2           |
| num of replica missing error                    | 4354857     |
| num of replica version missing error            | 967         |
| num of replica relocating                       | 0           |
| num of replica redundant error                  | 90          |
| num of replica missing in cluster error         | 0           |
| num of balance scheduled                        | 0           |
+-------------------------------------------------+-------------+
```

The meanings of each line are as follows:

- num of tablet check round: Number of Tablet Checker inspections
- cost of tablet check(ms): Total time consumed by Tablet Checker inspections (milliseconds)
- num of tablet checked in tablet checker: Number of tablets checked by the Tablet Checker
- num of unhealthy tablet checked in tablet checker: Number of unhealthy tablets checked by the Tablet Checker
- num of tablet being added to tablet scheduler: Number of tablets submitted to the Tablet Scheduler
- num of tablet schedule round: Number of Tablet Scheduler runs
- cost of tablet schedule(ms): Total time consumed by Tablet Scheduler runs (milliseconds)
- num of tablet being scheduled: Total number of tablets scheduled

- num of tablet being scheduled succeeded: Total number of tablets successfully scheduled
- num of tablet being scheduled failed: Total number of tablets that failed scheduling
- num of tablet being scheduled discard: Total number of tablets discarded due to scheduling failures
- num of tablet priority upgraded: Number of tablet priority upgrades
- num of tablet priority downgraded: Number of tablet priority downgrades
- num of clone task: Number of clone tasks generated
- num of clone task succeeded: Number of successful clone tasks
- num of clone task failed: Number of failed clone tasks
- num of clone task timeout: Number of clone tasks that timed out
- num of replica missing error: Number of tablets whose status is checked as missing replicas
- num of replica version missing error: Number of tablets checked with missing version status (this statistic includes num of replica relocating and num of replica missing in cluster error)
- num of replica relocation: Number of replica relocations
- num of replica redundant error: Number of tablets whose checked status is replica redundant
- num of replica missing in cluster error: Number of tablets checked with a status indicating they are missing from the corresponding cluster
- num of balance scheduled: Number of balanced scheduling attempts

> **Note**
>
> The above states are only historical accumulative values. We also print these statistics regularly in the FE logs, where the values in parentheses represent the number of changes in each statistical value since the last printing dependence of the statistical information.

Relevant configuration instructions

Adjustable parameters

The following adjustable parameters are all configurable parameters in fe.conf.

- use_new_tablet_scheduler
    - Description: Whether to enable the new replica scheduling mode. The new replica scheduling method is the replica scheduling method introduced in this document. If turned on, `disable_colocate_join` must be `true`. Because the new scheduling strategy does not support data fragmentation scheduling of co-location tables for the time being.
    - Default value:true

    - Importance: High

- tablet_repair_delay_factor_second

    - Note: For different scheduling priorities, we will delay different time to start repairing. In order to prevent a large number of unnecessary replica repair tasks from occurring in the process of routine restart and upgrade. This parameter is a reference coefficient. For HIGH priority, the delay is the reference coefficient * 1; for NORMAL priority, the delay is the reference coefficient * 2; for LOW priority, the delay is the reference coefficient * 3. That is, the lower the priority, the longer the delay waiting time. If the user wants to repair the copy as soon as possible, this parameter can be reduced appropriately.
    - Default value: 60 seconds
    - Importance: High

- schedule_slot_num_per_path

    - Note: The default number of slots allocated to each disk for replica repair. This number represents the number of replica repair tasks that a disk can run simultaneously. If you want to repair the copy faster, you can adjust this parameter appropriately. The higher the single value, the greater the impact on IO.
    - Default value: 2
    - Importance: High

- balance_load_score_threshold

    - Description: Threshold of Cluster Equilibrium. The default is 0.1, or 10%. When the load core of a BE node is not higher than or less than 10% of the average load core, we think that the node is balanced. If you want to make the cluster load more even, you can adjust this parameter appropriately.
    - Default value: 0.1
    - Importance:

- storage_high_watermark_usage_percent 和 storage_min_left_capacity_bytes

    - Description: These two parameters represent the upper limit of the maximum space utilization of a disk and the lower limit of the minimum space remaining, respectively. When the space utilization of a disk is greater than the upper limit or the remaining space is less than the lower limit, the disk will no longer be used as the destination address for balanced scheduling.
    - Default values: 0.85 and 2097152000 (2GB)
    - Importance:

- disable_balance

    - Description: Control whether to turn off the balancing function. When replicas are in equilibrium, some functions, such as ALTER TABLE, will be banned. Equilibrium can last for a long time. Therefore, if the user wants to do the prohibited operation as soon as possible. This parameter can be set to true to turn off balanced scheduling.
    - Default value: false
    - Importance:

The following adjustable parameters are all configurable parameters in be.conf.

- clone_worker_count
    - Description: Affects the speed of copy equalization. In the case of low disk pressure, you can speed up replica balancing by adjusting this parameter.
    - Default: 3
    - Importance: Medium #### Unadjustable parameters

The following parameters do not support modification for the time being, just for illustration.

- Tablet Checker scheduling interval

  Tablet Checker schedules checks every 20 seconds.

- Tablet Scheduler scheduling interval

  Tablet Scheduler schedules every five seconds

- Number of Tablet Scheduler Schedules per Batch

  Tablet Scheduler schedules up to 50 tablets at a time.

- Tablet Scheduler Maximum Waiting Schedule and Number of Tasks in Operation

  The maximum number of waiting tasks and running tasks is 2000. When over 2000, Tablet Checker will no longer generate new scheduling tasks to Tablet Scheduler.

- Tablet Scheduler Maximum Balanced Task Number

  The maximum number of balanced tasks is 500. When more than 500, there will be no new balancing tasks.

- Number of slots per disk for balancing tasks

  The number of slots per disk for balancing tasks is 2. This slot is independent of the slot used for replica repair.

- Update interval of cluster equilibrium

  Tablet Scheduler recalculates the load score of the cluster every 20 seconds.

- Minimum and Maximum Timeout for Clone Tasks

  A clone task timeout time range is 3 minutes to 2 hours. The specific timeout is calculated by the size of the tablet. The formula is (tablet size)/ (5MB/s). When a clone task fails three times, the task terminates.

- Dynamic Priority Adjustment Strategy

  The minimum priority adjustment interval is 5 minutes. When a tablet schedule fails five times, priority is lowered. When a tablet is not scheduled for 30 minutes, priority is raised.

Relevant issues

- In some cases, the default replica repair and balancing strategy may cause the network to be full (mostly in the case of gigabit network cards and a large number of disks per BE). At this point, some parameters

need to be adjusted to reduce the number of simultaneous balancing and repair tasks.

- Current balancing strategies for copies of Colocate Table do not guarantee that copies of the same Tablet will not be distributed on the BE of the same host. However, the repair strategy of the copy of Colocate Table detects this distribution error and corrects it. However, it may occur that after correction, the balancing strategy regards the replicas as unbalanced and rebalances them. As a result, the Colocate Group cannot achieve stability because of the continuous alternation between the two states. In view of this situation, we suggest that when using Colocate attribute, we try to ensure that the cluster is isomorphic, so as to reduce the probability that replicas are distributed on the same host.

Best Practices

Control and manage the progress of replica repair and balancing of clusters

In most cases, Doris can automatically perform replica repair and cluster balancing by default parameter configuration. However, in some cases, we need to manually intervene to adjust the parameters to achieve some special purposes. Such as prioritizing the repair of a table or partition, disabling cluster balancing to reduce cluster load, prioritizing the repair of non-colocation table data, and so on.

This section describes how to control and manage the progress of replica repair and balancing of the cluster by modifying the parameters.

1. Deleting Corrupt Replicas

   In some cases, Doris may not be able to automatically detect some corrupt replicas, resulting in frequent query or import errors on the corrupt replicas. In this case, we need to delete the corrupted copies manually. This method can be used to: delete a copy with a high version number resulting in a -235 error, delete a corrupted copy of a file, etc.

   First, find the tablet id of the corresponding copy, let's say 10001, and use `show tablet 10001;` and execute the `show proc` statement to see the details of each copy of the corresponding tablet.

   Assuming that the backend id of the copy to be deleted is 20001, the following statement is executed to mark the copy as `bad`.

   ```
   ADMIN SET REPLICA STATUS PROPERTIES("tablet_id" = "10001", "backend_id" = "20001", "status" = "bad");
   ```

   At this point, the `show proc` statement again shows that the `IsBad` column of the corresponding copy has a value of `true`.

   The replica marked as `bad` will no longer participate in imports and queries. The replica repair logic will automatically replenish a new replica at the same time. 2.

2. prioritize repairing a table or partition

   `help admin repair table;` View help. This command attempts to repair the tablet of the specified table or partition as a priority.

3. Stop the balancing task

The balancing task will take up some network bandwidth and IO resources. If you wish to stop the generation of new balancing tasks, you can do so with the following command.

```
ADMIN SET FRONTEND CONFIG ("disable_balance" = "true");
```

4. Stop all replica scheduling tasks

   Copy scheduling tasks include balancing and repair tasks. These tasks take up some network bandwidth and IO resources. All replica scheduling tasks (excluding those already running, including colocation tables and common tables) can be stopped with the following command.

```
ADMIN SET FRONTEND CONFIG ("disable_tablet_scheduler" = "true");
```

5. Stop the copy scheduling task for all colocation tables.

   The colocation table copy scheduling is run separately and independently from the regular table. In some cases, users may wish to stop the balancing and repair of colocation tables first and use the cluster resources for normal table repair with the following command.

```
ADMIN SET FRONTEND CONFIG ("disable_colocate_balance" = "true");
```

6. Repair replicas using a more conservative strategy

   Doris automatically repairs replicas when it detects missing replicas, BE downtime, etc. However, in order to reduce some errors caused by jitter (e.g., BE being down briefly), Doris delays triggering these tasks.

   - The `tablet_repair_delay_factor_second` parameter. Default 60 seconds. Depending on the priority of the repair task, it will delay triggering the repair task for 60 seconds, 120 seconds, or 180 seconds. This time can be extended so that longer exceptions can be tolerated to avoid triggering unnecessary repair tasks by using the following command.

```
ADMIN SET FRONTEND CONFIG ("tablet_repair_delay_factor_second" = "120");
```

7. use a more conservative strategy to trigger redistribution of colocation groups

   Redistribution of colocation groups may be accompanied by a large number of tablet migrations. `colocate_group_relocate_delay_second` is used to control the redistribution trigger delay. The default is 1800 seconds. If a BE node is likely to be offline for a long time, you can try to increase this parameter to avoid unnecessary redistribution by.

```
ADMIN SET FRONTEND CONFIG ("colocate_group_relocate_delay_second" = "3600");
```

8. Faster Replica Balancing

   Doris' replica balancing logic adds a normal replica first and then deletes the old one for the purpose of replica migration. When deleting the old replica, Doris waits for the completion of the import task that has already started on this replica to avoid the balancing task from affecting the import task. However, this

will slow down the execution speed of the balancing logic. In this case, you can make Doris ignore this wait and delete the old replica directly by modifying the following parameters.

```
ADMIN SET FRONTEND CONFIG ("enable_force_drop_redundant_replica" = "true");
```

This operation may cause some import tasks to fail during balancing (requiring a retry), but it will speed up balancing significantly.

Overall, when we need to bring the cluster back to a normal state quickly, consider handling it along the following lines.

1. find the tablet that is causing the highly optimal task to report an error and set the problematic copy to bad.
2. repair some tables with the `admin repair` statement.
3. Stop the replica balancing logic to avoid taking up cluster resources, and then turn it on again after the cluster is restored.
4. Use a more conservative strategy to trigger repair tasks to deal with the avalanche effect caused by frequent BE downtime.
5. Turn off scheduling tasks for colocation tables on-demand and focus cluster resources on repairing other high-optimality data.

Tablet metadata management tool

Tablet metadata management tool

Background

In the latest version of the code, we introduced RocksDB in BE to store meta-information of tablet, in order to solve various functional and performance problems caused by storing meta-information through header file. Currently, each data directory (root path) has a corresponding RocksDB instance, in which all tablets on the corresponding root path are stored in the key-value manner.

To facilitate the maintenance of these metadata, we provide an online HTTP interface and an offline meta tool to complete related management operations.

The HTTP interface is only used to view tablet metadata online, and can be used when the BE process is running.

However, meta tool is only used for off-line metadata management operations. BE must be stopped before it can be used.

The meta tool tool is stored in the Lib / directory of BE.

Operation

View Tablet Meta

Viewing Tablet Meta information can be divided into online and offline methods

Online

Access BE's HTTP interface to obtain the corresponding Tablet Meta information:

api:

```
http://{host}:{port}/api/meta/header/{tablet_id}/{schema_hash}
```

> Host: be Hostname
>
> port: BE's HTTP port
>
> tablet id: tablet id
>
> schema hash: tablet schema hash

Give an example:

`http://be_host:8040/api/meta/header/14156/2458238340`

If the final query is successful, the Tablet Meta will be returned as json.

Offline

Get Tablet Meta on a disk based on the meta tool tool.

Command:

```
./lib/meta_tool --root_path=/path/to/root_path --operation=get_meta --tablet_id=xxx --schema_hash=xxx
```

> root_path: The corresponding root_path path path configured in be.conf.

The result is also a presentation of Tablet Meta in JSON format.

Load header

The function of loading header is provided to realize manual migration of tablet. This function is based on Tablet Meta in JSON format, so if changes in the shard field and version information are involved, they can be changed directly in the JSON content of Tablet Meta. Then use the following commands to load.

Command:

```
./lib/meta_tool --operation=load_meta --root_path=/path/to/root_path --json_meta_path=path
```

Delete header

In order to realize the function of deleting a tablet meta from a disk of a BE. Support single delete and batch delete.

Single delete:

```
./lib/meta_tool --operation=delete_meta --root_path=/path/to/root_path --tablet_id=xxx --schema_hash=xxx`
```

Batch delete:

```
./lib/meta_tool --operation=batch_delete_meta --tablet_file=/path/to/tablet_file.txt
```

Each line in `tablet_file.txt` represents the information of a tablet. The format is:

`root_path,tablet_id,schema_hash`

Each column are separated by comma.

`tablet_file` example:

```
/output/be/data/,14217,352781111
/output/be/data/,14219,352781111
/output/be/data/,14223,352781111
/output/be/data/,14227,352781111
/output/be/data/,14233,352781111
/output/be/data/,14239,352781111
```

Batch delete will skip the line with incorrect tablet information format in `tablet_file`. And after the execution is completed, the number of successful deletions and the number of errors are displayed.

TabletMeta in Pb format

This command is to view the old file-based management PB format Tablet Meta, and to display Tablet Meta in JSON format.

Command:

```
./lib/meta_tool --operation=show_meta --root_path=/path/to/root_path --pb_header_path=path
```

Segment meta in Pb format

This command is to view the PB format segment meta, and to display segment meta in JSON format.

Command:

```
./meta_tool --operation=show_segment_footer --file=/path/to/segment/file
```

Tablet Restore Tool

Tablet Restore Tool

Restore data from BE Recycle Bin

During the user's use of Doris, some valid tablets (including metadata and data) may be deleted due to some misoperations or online bugs. In order to prevent data loss in these abnormal situations, Doris provides a recycle bin mechanism to protect user data. Tablet data deleted by users will not be deleted directly, but will be stored in the recycle bin for a period of time. After a period of time, there will be a regular cleaning mechanism to delete expired data. The data in the recycle bin includes: tablet data file (.dat), tablet index file (.idx) and tablet metadata file (.hdr). The data will be stored in a path in the following format:

```
/root_path/trash/time_label/tablet_id/schema_hash/
```

- `root_path`: a data root directory corresponding to the BE node.
- `trash`: The directory of the recycle bin.
- `time_label`: Time label, for the uniqueness of the data directory in the recycle bin, while recording the data time, use the time label as a subdirectory.

When a user finds that online data has been deleted by mistake, he needs to recover the deleted tablet from the recycle bin. This tablet data recovery function is needed.

BE provides http interface and `restore_tablet_tool.sh` script to achieve this function, and supports single tablet operation (single mode) and batch operation mode (batch mode).

- In single mode, data recovery of a single tablet is supported.
- In batch mode, support batch tablet data recovery.

Operation

single mode

1. http request method

   BE provides an http interface for single tablet data recovery, the interface is as follows:

   ```
   curl -X POST "http://be_host:be_webserver_port/api/restore_tablet?tablet_id=11111\&schema_hash=12345"
   ```

   The successful results are as follows:

   ```
   {"status": "Success", "msg": "OK"}
   ```

   If it fails, the corresponding failure reason will be returned. One possible result is as follows:

```
{"status": "Failed", "msg": "create link path failed"}
```

2. Script mode

   `restore_tablet_tool.sh` can be used to realize the function of single tablet data recovery.

```
sh tools/restore_tablet_tool.sh -b "http://127.0.0.1:8040" -t 12345 -s 11111
sh tools/restore_tablet_tool.sh --backend "http://127.0.0.1:8040" --tablet_id 12345 --schema_hash 11111
```

batch mode

The batch recovery mode is used to realize the function of recovering multiple tablet data.

When using, you need to put the restored tablet id and schema hash in a file in a comma-separated format in advance, one tablet per line.

The format is as follows:

```
12345,11111
12346,11111
12347,11111
```

Then perform the recovery with the following command (assuming the file name is: `tablets.txt`):

```
sh restore_tablet_tool.sh -b "http://127.0.0.1:8040" -f tablets.txt
sh restore_tablet_tool.sh --backend "http://127.0.0.1:8040" --file tablets.txt
```

Repair missing or damaged Tablet

In some very special circumstances, such as code bugs, or human misoperation, etc., all replicas of some tablets may be lost. In this case, the data has been substantially lost. However, in some scenarios, the business still hopes to ensure that the query will not report errors even if there is data loss, and reduce the perception of the user layer. At this point, we can use the blank Tablet to fill the missing replica to ensure that the query can be executed normally.

Note: This operation is only used to avoid the problem of error reporting due to the inability to find a queryable replica, and it is impossible to recover the data that has been substantially lost.

1. View Master FE log `fe.log`

   If there is data loss, there will be a log similar to the following in the log:

```
backend [10001] invalid situation. tablet[20000] has few replica[1], replica num setting is [3]
```

   This log indicates that all replicas of tablet 20000 have been damaged or lost.

2. Use blank replicas to fill in missing copies

After confirming that the data cannot be recovered, you can execute the following command to generate blank replicas.

```
ADMIN SET FRONTEND CONFIG ("recover_with_empty_tablet" = "true");
```

- Note: You can first check whether the current version supports this parameter through the `ADMIN SHOW FRONTEND CONFIG;` command.

3. A few minutes after the setup is complete, you should see the following log in the Master FE log `fe.log`:

```
tablet 20000 has only one replica 20001 on backend 10001 and it is lost. create an empty replica to recover it.
```

The log indicates that the system has created a blank tablet to fill in the missing replica.

4. Judge whether it has been repaired successfully through query.

Tablet Local Debug

Tablet Local Debug

During the online operation of Doris, various bugs may occur due to various reasons. For example: the replica is inconsistent, the data exists in the version diff, etc.

At this time, it is necessary to copy the copy data of the tablet online to the local environment for reproduction, and then locate the problem.

1. Get information about the tablet

The tablet id can be confirmed by the BE log, and then the information can be obtained by the following command (assuming the tablet id is 10020).

Get information such as DbId/TableId/PartitionId where the tablet is located.

```
mysql> show tablet 10020\G
*************************** 1. row ***************************
       DbName: default_cluster:db1
    TableName: tbl1
PartitionName: tbl1
    IndexName: tbl1
         DbId: 10004
      TableId: 10016
  PartitionId: 10015
      IndexId: 10017
       IsSync: true
        Order: 1
    DetailCmd: SHOW PROC '/dbs/10004/10016/partitions/10015/10017/10020';
```

Execute `DetailCmd` in the previous step to obtain information such as BackendId/SchemaHash.

```
mysql>  SHOW PROC '/dbs/10004/10016/partitions/10015/10017/10020'\G
*************************** 1. row ***************************
        ReplicaId: 10021
        BackendId: 10003
          Version: 3
LstSuccessVersion: 3
 LstFailedVersion: -1
    LstFailedTime: NULL
       SchemaHash: 785778507
    LocalDataSize: 780
   RemoteDataSize: 0
         RowCount: 2
            State: NORMAL
            IsBad: false
     VersionCount: 3
         PathHash: 7390150550643804973
          MetaUrl: http://192.168.10.1:8040/api/meta/header/10020
 CompactionStatus: http://192.168.10.1:8040/api/compaction/show?tablet_id=10020
```

Create tablet snapshot and get table creation statement

```
mysql> admin copy tablet 10020 properties("backend_id" = "10003", "version" = "2")\G
*************************** 1. row ***************************
         TabletId: 10020
        BackendId: 10003
               Ip: 192.168.10.1
             Path: /path/to/be/storage/snapshot/20220830101353.2.3600
ExpirationMinutes: 60
  CreateTableStmt: CREATE TABLE `tbl1` (
  `k1` int(11) NULL,
  `k2` int(11) NULL
) ENGINE=OLAP
DUPLICATE KEY(`k1`, `k2`)
DISTRIBUTED BY HASH(k1) BUCKETS 1
PROPERTIES (
"replication_num" = "1",
"version_info" = "2"
);
```

The `admin copy tablet` command can generate a snapshot file of the corresponding replica and version for the specified tablet. Snapshot files are stored in the `Path` directory of the BE node indicated by the `Ip` field.

There will be a directory named tablet id under this directory, which will be packaged as a whole for later use. (Note that the directory is kept for a maximum of 60 minutes, after which it is automatically deleted).

```
cd /path/to/be/storage/snapshot/20220830101353.2.3600
tar czf 10020.tar.gz 10020/
```

The command will also generate the table creation statement corresponding to the tablet at the same time. Note that this table creation statement is not the original table creation statement, its bucket number and replica number are both 1, and the `versionInfo` field is specified. This table building statement is used later when loading the tablet locally.

So far, we have obtained all the necessary information, the list is as follows:

1. Packaged tablet data, such as 10020.tar.gz.
2. Create a table statement.

2. Load Tablet locally

1. Build a local debugging environment

   Deploy a single-node Doris cluster (1FE, 1BE) locally, and the deployment version is the same as the online cluster. If the online deployment version is DORIS-1.1.1, the local environment also deploys the DORIS-1.1.1 version.

2. Create a table

   Create a table in the local environment using the create table statement from the previous step.

3. Get the tablet information of the newly created table

   Because the number of buckets and replicas of the newly created table is 1, there will only be one tablet with one replica:

```
mysql> show tablets from tbl1\G
*************************** 1. row ***************************
              TabletId: 10017
             ReplicaId: 10018
             BackendId: 10003
            SchemaHash: 44622287
               Version: 1
      LstSuccessVersion: 1
       LstFailedVersion: -1
          LstFailedTime: NULL
          LocalDataSize: 0
         RemoteDataSize: 0
               RowCount: 0
                  State: NORMAL
LstConsistencyCheckTime: NULL
           CheckVersion: -1
           VersionCount: -1
               PathHash: 7390150550643804973
                MetaUrl: http://192.168.10.1:8040/api/meta/header/10017
       CompactionStatus: http://192.168.10.1:8040/api/compaction/show?tablet_id=10017
```

```
mysql> show tablet 10017\G
*************************** 1. row ***************************
       DbName: default_cluster:db1
    TableName: tbl1
PartitionName: tbl1
    IndexName: tbl1
         DbId: 10004
      TableId: 10015
  PartitionId: 10014
      IndexId: 10016
       IsSync: true
        Order: 0
    DetailCmd: SHOW PROC '/dbs/10004/10015/partitions/10014/10016/10017';
```

Here we will record the following information:

- TableId
- PartitionId
- TabletId
- SchemaHash

At the same time, we also need to go to the data directory of the BE node in the debugging environment to confirm the shard id where the new tablet is located:

```
cd /path/to/storage/data/*/10017 && pwd
```

This command will enter the directory where the tablet 10017 is located and display the path. Here we will see a path similar to the following:

```
/path/to/storage/data/0/10017
```

where `0` is the shard id.

4. Modify Tablet Data

Unzip the tablet data package obtained in the first step. The editor opens the 10017.hdr.json file, and modifies the following fields to the information obtained in the previous step:

```
"table_id":10015
"partition_id":10014
"tablet_id":10017
"schema_hash":44622287
"shard_id":0
```

5. Load the tablet

   First, stop the debug environment's BE process (./bin/stop_be.sh). Then copy all the .dat files in the same level directory of the 10017.hdr.json file to the `/path/to/storage/data/0/10017/44622287` directory. This directory is the directory where the debugging environment tablet we obtained in step 3 is located. `10017/44622287` are the tablet id and schema hash respectively.

   Delete the original tablet meta with the `meta_tool` tool. The tool is located in the `be/lib` directory.

   ```
   ./lib/meta_tool --root_path=/path/to/storage --operation=delete_meta --tablet_id=10017 --schema_hash=44622287
   ```

   Where `/path/to/storage` is the data root directory of BE. If the deletion is successful, the delete successfully log will appear.

   Load the new tablet meta via the `meta_tool` tool.

   ```
   ./lib/meta_tool --root_path=/path/to/storage --operation=load_meta --json_meta_path=/path/to/10017.hdr.json
   ```

   If the load is successful, the load successfully log will appear.

6. Verification

   Restart the debug environment's BE process (./bin/start_be.sh). Query the table, if correct, you can query the data of the loaded tablet, or reproduce the online problem.

Metadata Operations and Maintenance

> **Avoid using metadata_failure_recovery unless absolutely necessary. Using it may cause metadata truncation, loss, and split-brains. Use cautiously to prevent irreversible data damage from improper operations.**

This document focuses on how to manage Doris metadata in a real production environment. It includes the proposed deployment of FE nodes, some commonly used operational methods, and common error resolution methods.

For the time being, read the Doris metadata design document to understand how Doris metadata works.

Important tips

- Current metadata design is not backward compatible. That is, if the new version has a new metadata structure change (you can see whether there is a new VERSION in the `FeMetaVersion.java` file in the FE code), it is usually impossible to roll back to the old version after upgrading to the new version. Therefore, before upgrading FE, be sure to test metadata compatibility according to the operations in the Upgrade Document.

Metadata catalog structure

Let's assume that the path of `meta_dir` specified in fe.conf is `path/to/doris-meta`. In a normal Doris cluster, the directory structure of metadata should be as follows:

```
/path/to/doris-meta/
            |-- bdb/
            |   |-- 00000000.jdb
            |   |-- je.config.csv
            |   |-- je.info.0
            |   |-- je.info.0.lck
            |   |-- je.lck
            |   `-- je.stat.csv
            `-- image/
                |-- ROLE
                |-- VERSION
                `-- image.xxxx
```

1. bdb

   We use bdbje as a distributed kV system to store metadata journal. This BDB directory is equivalent to the "data directory" of bdbje.

   The `.jdb` suffix is the data file of bdbje. These data files will increase with the increasing number of metadata journals. When Doris regularly completes the image, the old log is deleted. So normally, the total size of these data files varies from several MB to several GB (depending on how Doris is used, such as import frequency). When the total size of the data file is larger than 10GB, you may need to

wonder whether the image failed or the historical journals that failed to distribute the image could not be deleted.

`je.info.0` is the running log of bdbje. The time in this log is UTC + 0 time zone. We may fix this in a later version. From this log, you can also see how some bdbje works.

2. image directory

   The image directory is used to store metadata mirrors generated regularly by Doris. Usually, you will see a `image.xxxxx` mirror file. Where `xxxxx` is a number. This number indicates that the image contains all metadata journal before `xxxx`. And the generation time of this file (viewed through `ls -al`) is usually the generation time of the mirror.

   You may also see a `image.ckpt` file. This is a metadata mirror being generated. The `du -sh` command should show that the file size is increasing, indicating that the mirror content is being written to the file. When the mirror is written, it automatically renames itself to a new `image.xxxxx` and replaces the old image file.

   Only FE with a Master role will actively generate image files on a regular basis. After each generation, FE is pushed to other non-Master roles. When it is confirmed that all other FEs have received this image, Master FE deletes the metadata journal in bdbje. Therefore, if image generation fails or image push fails to other FEs, data in bdbje will accumulate.

   `ROLE` file records the type of FE (FOLLOWER or OBSERVER), which is a text file.

   `VERSION` file records the cluster ID of the Doris cluster and the token used to access authentication between nodes, which is also a text file.

   `ROLE` file and `VERSION` file may only exist at the same time, or they may not exist at the same time (e.g. at the first startup).

Basic operations

Start single node FE

Single node FE is the most basic deployment mode. A complete Doris cluster requires at least one FE node. When there is only one FE node, the type of the node is Follower and the role is Master.

1. First start-up
   1. Suppose the path of `meta_dir` specified in fe.conf is `path/to/doris-meta`.
   2. Ensure that `path/to/doris-meta` already exists, that the permissions are correct and that the directory is empty.
   3. Start directly through `sh bin/start_fe.sh`.
   4. After booting, you should be able to see the following log in fe.log:
      - Palo FE starting…
      - image does not exist: /path/to/doris-meta/image/image.0
      - transfer from INIT to UNKNOWN

- transfer from UNKNOWN to MASTER
- the very first time to open bdb, dbname is 1
- start fencing, epoch number is 1
- finish replay in xxx msec
- QE service start
- thrift server started

The above logs are not necessarily strictly in this order, but they are basically similar.

5. The first start-up of a single-node FE usually does not encounter problems. If you haven't seen the above logs, generally speaking, you haven't followed the document steps carefully, please read the relevant wiki carefully.

2. Restart

    1. Stopped FE nodes can be restarted by using `sh bin/start_fe.sh`.

    2. After restarting, you should be able to see the following log in fe.log:

        - Palo FE starting…
        - finished to get cluster id: xxxx, role: FOLLOWER and node name: xxxx
        - If no image has been generated before reboot, you will see:
        - image does not exist: /path/to/doris-meta/image/image.0
        - If an image is generated before the restart, you will see:
        - start load image from /path/to/doris-meta/image/image.xxx. is ckpt: false
        - finished load image in xxx ms
        - transfer from INIT to UNKNOWN
        - replayed journal id is xxxx, replay to journal id is yyyy
        - transfer from UNKNOWN to MASTER
        - finish replay in xxx msec
        - master finish replay journal, can write now.
        - begin to generate new image: image.xxxx
        - start save image to /path/to/doris-meta/image/image.ckpt. is ckpt: true
        - finished save image /path/to/doris-meta/image/image.ckpt in xxx ms. checksum is xxxx
        - push image.xxx to other nodes. totally xx nodes, push successed xx nodes
        - QE service start
        - thrift server started

The above logs are not necessarily strictly in this order, but they are basically similar.

3. Common problems

   For the deployment of single-node FE, start-stop usually does not encounter any problems. If you have any questions, please refer to the relevant Wiki and check your operation steps carefully.

Add FE

Adding FE processes is described in detail in the Elastic Expansion Documents and will not be repeated. Here are some points for attention, as well as common problems.

1. Notes

   - Before adding a new FE, make sure that the current Master FE runs properly (connection is normal, JVM is normal, image generation is normal, bdbje data directory is too large, etc.)
   - The first time you start a new FE, you must make sure that the `--helper` parameter is added to point to Master FE. There is no need to add `--helper` when restarting. (If `--helper` is specified, FE will directly ask the helper node for its role. If not, FE will try to obtain information from `ROLE` and `VERSION` files in the `doris-meta/image/` directory.
   - The first time you start a new FE, you must make sure that the `meta_dir` of the FE is created, has correct permissions and is empty.
   - Starting a new FE and executing the `ALTER SYSTEM ADD FOLLOWER/OBSERVER` statement adds FE to metadata in a sequence that is not required. If a new FE is started first and no statement is executed, the `current node is not added to the group. Please add it first.` in the new FE log. When the statement is executed, it enters the normal process.
   - Make sure that after the previous FE is added successfully, the next FE is added.
   - Connect to MASTER FE and execute `ALTER SYSTEM ADD FOLLOWER/OBSERVER` stmt.

2. Common problems

   1. this need is DETACHED

      When you first start a FE to be added, if the data in doris-meta/bdb on Master FE is large, you may see the words `this node is DETACHED.` in the FE log to be added. At this point, bdbje is copying data, and you can see that the `bdb/` directory of FE to be added is growing. This process usually takes several minutes (depending on the amount of data in bdbje). Later, there may be some bdbje-related error stack information in fe. log. If `QE service start` and `thrift server start` are displayed in the final log, the start is usually successful. You can try to connect this FE via mysql-client. If these words do not appear, it may be the problem of bdbje replication log timeout. At this point, restarting the FE directly will usually solve the problem.

   2. Failure to add due to various reasons

      - If OBSERVER is added, because OBSERVER-type FE does not participate in the majority of metadata writing, it can theoretically start and stop at will. Therefore, for the case of adding OBSERVER failure. The process of OBSERVER FE can be killed directly. After clearing the metadata directory of OBSERVER, add the process again.

      - If FOLLOWER is added, because FOLLOWER is mostly written by participating metadata. So it is possible that FOLLOWER has joined the bdbje electoral team. If there are only two

FOLLOWER nodes (including MASTER), then stopping one FE may cause another FE to quit because it cannot write most of the time. At this point, we should first delete the newly added FOLLOWER node from the metadata through the `ALTER SYSTEM DROP FOLLOWER` command, then kill the FOLLOWER process, empty the metadata and re-add the process.

Delete FE

The corresponding type of FE can be deleted by the `ALTER SYSTEM DROP FOLLOWER/OBSERVER` command. The following points should be noted:

- For OBSERVER type FE, direct DROP is enough, without risk.
- For FOLLOWER type FE. First, you should make sure that you start deleting an odd number of FOLLOWERs (three or more).
    1. If the FE of non-MASTER role is deleted, it is recommended to connect to MASTER FE, execute DROP command, and then kill the process.
    2. If you want to delete MASTER FE, first confirm that there are odd FOLLOWER FE and it works properly. Then kill the MASTER FE process first. At this point, a FE will be elected MASTER. After confirming that the remaining FE is working properly, connect to the new MASTER FE and execute the DROP command to delete the old MASTER FE.

Advanced Operations

Failure recovery

FE may fail to start bdbje and synchronize between FEs for some reasons. Phenomena include the inability to write metadata, the absence of MASTER, and so on. At this point, we need to manually restore the FE. The general principle of manual recovery of FE is to start a new MASTER through metadata in the current `meta_dir`, and then add other FEs one by one. Please follow the following steps strictly:

1. First, **stop all FE processes and all business access**. Make sure that during metadata recovery, external access will not lead to other unexpected problems.(if not, this may cause split-brain problem)
2. Identify which FE node's metadata is up-to-date:
    - First of all, **be sure to back up all FE's** `meta_dir` **directories first.**
    - Usually, Master FE's metadata is up to date. You can see the suffix of image.xxxx file in the `meta_dir/image` directory. The larger the number, the newer the metadata.
    - Usually, by comparing all FOLLOWER FE image files, you can find the latest metadata.
    - After that, we use the FE node with the latest metadata to recover.
    - If using metadata of OBSERVER node to recover will be more troublesome, it is recommended to choose FOLLOWER node as far as possible.
3. The following operations are performed on the FE nodes selected in step 2.
    1. Modify fe.conf
        - If the node is an OBSERVER, first change the `role=OBSERVER` in the `meta_dir/image/ROLE` file to `role=FOLLOWER`. (Recovery from the OBSERVER node will be more cumbersome, first follow the steps here, followed by a separate description)

        - If fe.version < 2.0.2, add configuration in fe.conf: `metadata_failure_recovery=true`.
    2. Run `sh bin/start_fe.sh --metadata_failure_recovery --daemon` to start the FE. (If you are recovering from an OBSERVER node, jump to the subsequent OBSERVER document after this step.)
    3. If normal, the FE will start in the role of MASTER, similar to the description in the previous section `Start a single node FE`. You should see the words `transfer from XXXX to MASTER` in fe.log.
    4. After the start-up is completed, connect to the FE first, and execute some query imports to check whether normal access is possible. If the operation is not normal, it may be wrong. It is recommended to read the above steps carefully and try again with the metadata previously backed up. If not, the problem may be more serious.
    5. If successful, through the `show frontends;` command, you should see all the FEs you added before, and the current FE is master.
    6. **If FE version < 2.0.2**, delete the `metadata_failure_recovery=true` configuration item in fe.conf, or set it to `false`, and restart the FE (**Important**).

    :::tip If you are recovering metadata from an OBSERVER node, after completing the above steps, you will find that the current FE role is OBSERVER, but `IsMaster` appears as `true`. This is because the "OBSERVER" seen here is recorded in Doris's metadata, but whether it is master or not, is recorded in bdbje's metadata. Because we recovered from an OBSERVER node, there was inconsistency. Please take the following steps to fix this problem (we will fix it in a later version):

    1. First, all FE nodes except this "OBSERVER" are DROPed out.

    2. A new FOLLOWER FE is added through the `ADD FOLLOWER` command, assuming that it is on hostA.

    3. Start a new FE on hostA and join the cluster by `helper`.

    4. After successful startup, you should see two FEs through the `show frontends;` statement, one is the previous OBSERVER, the other is the newly added FOLLOWER, and the OBSERVER is the master.

    5. After confirming that the new FOLLOWER is working properly, the new FOLLOWER metadata is used to perform a failure recovery operation again.

    6. The purpose of the above steps is to manufacture a metadata of FOLLOWER node artificially, and then use this metadata to restart fault recovery. This avoids inconsistencies in recovering metadata from OBSERVER.

    The meaning of `metadata_failure_recovery` is to empty the metadata of `bdbje`. In this way, bdbje will not contact other FEs before, but start as a separate FE. This parameter needs to be set to true only when restoring startup. After recovery, it must be set to false. Otherwise, once restarted, the metadata of bdbje will be emptied again, which will make other FEs unable to work properly. :::

4. After the successful execution of step 3, we delete the previous FEs from the metadata by using the `ALTER SYSTEM DROP FOLLOWER/OBSERVER` command and add them again by adding new FEs.

5. If the above operation is normal, it will be restored.

FE type change

If you need to change the existing FOLLOWER/OBSERVER type FE to OBSERVER/FOLLOWER type, please delete FE in the way described above, and then add the corresponding type FE.

FE Migration

If you need to migrate one FE from the current node to another, there are several scenarios.

1. FOLLOWER, or OBSERVER migration for non-MASTER nodes

   After adding a new FOLLOWER / OBSERVER directly, delete the old FOLLOWER / OBSERVER.

2. Single-node MASTER migration

   When there is only one FE, refer to the `Failure Recovery` section. Copy the doris-meta directory of FE to the new node and start the new MASTER in Step 3 of the `Failure Recovery` section

3. A set of FOLLOWER migrates from one set of nodes to another set of new nodes

   Deploy FE on the new node and add the new node first by adding FOLLOWER. The old nodes can be dropped by DROP one by one. In the process of DROP-by-DROP, MASTER automatically selects the new FOLLOWER node.

Replacement of FE port

FE currently has the following ports

- Ed_log_port: bdbje's communication port
- http_port: http port, also used to push image
- rpc_port: thrift server port of Frontend
- query_port: Mysql connection port
- arrow_flight_sql_port: Arrow Flight SQL connection port

1. edit_log_port

   If this port needs to be replaced, it needs to be restored with reference to the operations in the `Failure Recovery` section. Because the port has been persisted into bdbje's own metadata (also recorded in Doris's own metadata), it is necessary to clear bdbje's metadata by setting `metadata_failure_recovery` when Fe start.

2. http_port

   All FE http_ports must be consistent. So if you want to modify this port, all FEs need to be modified and restarted. Modifying this port will be more complex in the case of multiple FOLLOWER deployments (involving laying eggs and laying hens…), so this operation is not recommended. If necessary, follow the operation in the `Failure Recovery` section directly.

3. rpc_port

   After modifying the configuration, restart FE directly. Master FE informs BE of the new port through heartbeat. Only this port of Master FE will be used. However, it is still recommended that all FE ports be

consistent.

4. query_port

   After modifying the configuration, restart FE directly. This only affects mysql's connection target.

5. arrow_flight_sql_port

   After modifying the configuration, restart FE directly. This only affects arrow flight sql server connection target.

Recover metadata from FE memory

In some extreme cases, the image file on the disk may be damaged, but the metadata in the memory is intact. At this point, we can dump the metadata from the memory and replace the image file on the disk to recover the metadata. the entire non-stop query service operation steps are as follows:

1. Stop all Load, Create, Alter operations.
2. Execute the following command to dump metadata from the Master FE memory: (hereafter called image_mem)

```
curl -u $root_user:$password http://$master_hostname:8030/dump
```

3. Replace the image file in the `meta_dir/image` directory on the OBSERVER FE node with the image_mem file, restart the OBSERVER FE node, and verify the integrity and correctness of the image_mem file. You can check whether the DB and Table metadata are normal on the FE Web page, whether there is an exception in `fe.log`, whether it is in a normal replayed jour.

   Since 1.2.0, it is recommanded to use following method to verify the `image_mem` file:

   ```
   sh start_fe.sh --image path_to_image_mem
   ```

   > Notice: `path_to_image_mem` is the path of `image_mem`.
   >
   > If verify succeed, it will print: `Load image success. Image file /absolute/path/to/image.xxxxxx is valid`.
   >
   > If verify failed, it will print: `Load image failed. Image file /absolute/path/to/image.xxxxxx is invalid`.

4. Replace the image file in the `meta_dir/image` directory on the FOLLOWER FE node with the image_mem file in turn, restart the FOLLOWER FE node, and confirm that the metadata and query services are normal.
5. Replace the image file in the `meta_dir/image` directory on the Master FE node with the image_mem file, restart the Master FE node, and then confirm that the FE Master switch is normal and The Master FE node can generate a new image file through checkpoint.
6. Recover all Load, Create, Alter operations.

Note: If the Image file is large, the entire process can take a long time, so during this time, make sure Master FE does not generate a new image file via checkpoint. When the image.ckpt file in the meta_dir/image directory on the Master FE node is observed to be as large as the image.xxx file, the image.ckpt file can be deleted directly.

View data in BDBJE

The metadata log of FE is stored in BDBJE in the form of Key-Value. In some abnormal situations, FE may not be started due to metadata errors. In this case, Doris provides a way to help users query the data stored in BDBJE to facilitate troubleshooting.

First, you need to add configuration in fe.conf: `enable_bdbje_debug_mode=true`, and then start FE through `sh start_fe.sh --daemon`.

At this time, FE will enter the debug mode, only start the http server and MySQL server, and open the BDBJE instance, but will not load any metadata and other subsequent startup processes.

This is, we can view the data stored in BDBJE by visiting the web page of FE, or after connecting to Doris through the MySQL client, through `show proc "/bdbje";`.

```
mysql> show proc "/bdbje";
+----------+---------------+---------+
| DbNames  | JournalNumber | Comment |
+----------+---------------+---------+
| 110589   | 4273          |         |
| epochDB  | 4             |         |
| metricDB | 430694        |         |
+----------+---------------+---------+
```

The first level directory will display all the database names in BDBJE and the number of entries in each database.

```
mysql> show proc "/bdbje/110589";
+-----------+
| JournalId |
+-----------+
| 1         |
| 2         |

...
| 114858    |
| 114859    |
| 114860    |
| 114861    |
+-----------+
4273 rows in set (0.06 sec)
```

Entering the second level, all the entry keys under the specified database will be listed.

```
mysql> show proc "/bdbje/110589/114861";
+-----------+--------------+--------------------------------------------+
| JournalId | OpType       | Data                                       |
+-----------+--------------+--------------------------------------------+
| 114861    | OP_HEARTBEAT | org.apache.doris.persist.HbPackage@6583d5fb |
```

```
+-----------+--------------+-------------------------------------------+
1 row in set (0.05 sec)
```

The third level can display the value information of the specified key.

Best Practices

The deployment recommendation of FE is described in the Installation and Deployment Document. Here are some supplements.

- **If you don't know the operation logic of FE metadata very well, or you don't have enough experience in the operation and maintenance of FE metadata, we strongly recommend that only one FOLLOWER-type FE be deployed as MASTER in practice, and the other FEs are OBSERVER, which can reduce many complex operation and maintenance problems.** Don't worry too much about the failure of MASTER single point to write metadata. First, if you configure it properly, FE as a java process is very difficult to hang up. Secondly, if the MASTER disk is damaged (the probability is very low), we can also use the metadata on OBSERVER to recover manually through `fault recovery`.
- The JVM of the FE process must ensure sufficient memory. We **strongly recommend** that FE's JVM memory should be at least 10GB and 32GB to 64GB. And deploy monitoring to monitor JVM memory usage. Because if OOM occurs in FE, metadata writing may fail, resulting in some failures that **cannot recover**!
- FE nodes should have enough disk space to prevent the excessive metadata from causing insufficient disk space. At the same time, FE logs also take up more than a dozen gigabytes of disk space.

Other common problems

1. Output `meta out of date. current time: xxx, synchronized time: xxx, has log: xxx, fe type: xxx` in fe.log

   This is usually because the FE cannot elect Master. For example, if three FOLLOWERs are configured, but only one FOLLOWER is started, this FOLLOWER will cause this problem. Usually, just start the remaining FOLLOWER. If the problem has not been solved after the start-up, manual recovery may be required in accordance with the way in the `Failure Recovery` section.

2. `Clock delta: xxxx ms. between Feeder: xxxx and this Replica exceeds max permissible delta: xxxx ms.`

   Bdbje requires that clock errors between nodes should not exceed a certain threshold. If exceeded, the node will exit abnormally. The default threshold is 5000ms, which is controlled by FE parameter `max_bdbje_clock_delta_ms', and can be modified as appropriate. But we suggest using NTP and other clock synchronization methods to ensure the clock synchronization of Doris cluster hosts.

3. Mirror files in the `image/` directory have not been updated for a long time

   Master FE generates a mirror file by default for every 50,000 metadata journal. In a frequently used cluster, a new image file is usually generated every half to several days. If you find that the image file has not been updated for a long time (for example, more than a week), you can see the reasons in sequence as follows:

   1. Search for `memory is not enough to do checkpoint. Committed memory XXXX Bytes, used memory XXXX Bytes.` in the fe.log of Master FE. If found, it indicates that the current FE's JVM memory is

insufficient for image generation (usually we need to reserve half of the FE memory for image generation). Then you need to add JVM memory and restart FE before you can observe. Each time Master FE restarts, a new image is generated directly. This restart method can also be used to actively generate new images. Note that if there are multiple FOLLOWER deployments, then when you restart the current Master FE, another FOLLOWER FE will become MASTER, and subsequent image generation will be the responsibility of the new Master. Therefore, you may need to modify the JVM memory configuration of all FOLLOWER FE.

2. Search for `begin to generate new image: image.xxxx` in the fe.log of Master FE. If it is found, then the image is generated. Check the subsequent log of this thread, and if `checkpoint finished save image.xxxx` appears, the image is written successfully. If `Exception when generating new image file` occurs, the generation fails and specific error messages need to be viewed.

4. The size of the `bdb/` directory is very large, reaching several Gs or more.

   The BDB directory will remain large for some time after eliminating the error that the new image cannot be generated. Maybe it's because Master FE failed to push image. You can search `push image.XXXX to other nodes. totally XX nodes, push successed YY nodes` in the fe. log of Master FE. If YY is smaller than xx, then some FEs are not pushed successfully. You can see the specific error `Exception when pushing image file.url = xxx` in the fe. log.

   At the same time, you can add the configuration in the FE configuration file: `edit_log_roll_num = xxxx`. This parameter sets the number of metadata journals and makes an image once. The default is 50000. This number can be reduced appropriately to make images more frequent, thus speeding up the deletion of old journals.

5. FOLLOWER FE hangs up one after another

   Because Doris's metadata adopts the majority writing strategy, that is, a metadata journal must be written to at least a number of FOLLOWER FEs (for example, three FOLLOWERs, two must be written successfully) before it can be considered successful. If the write fails, the FE process exits on its own initiative. So suppose there are three FOLLOWERs: A, B and C. C hangs up first, and then B hangs up, then A will hang up. So as described in the `Best Practices`section, if you don't have extensive experience in metadata operations and maintenance, it's not recommended to deploy multiple FOLLOWERs.

6. fe.log 中出现 `get exception when try to close previously opened bdb database. ignore it`

   If there is the word `ignore it` behind it, there is usually no need to deal with it. If you are interested, you can search for this error in `BDBEnvironment.java`, and see the annotations.

7. From `show frontends;` Look, the `Join` of a FE is listed as `true`, but actually the FE is abnormal.

   Through `show frontends;` see the `Join` information. If the column is `true`, it only means that the FE **has joined the** cluster. It does not mean that it still exists normally in the cluster. If `false`, it means that the FE **has never joined the** cluster.

8. Configuration of FE `master_sync_policy`, `replica_sync_policy`, and `txn_rollback_limit`.

   `master_sync_policy` is used to specify whether fsync (), `replica_sync_policy` is called when Leader FE writes metadata log, and `replica_sync_policy` is used to specify whether other Follower FE calls fsync () when FE HA deploys synchronous metadata. In earlier versions of Oris, these two parameters defaulted

to `WRITE_NO_SYNC`, i.e., fsync () was not called. In the latest version of Oris, the default has been changed to `SYNC`, that is, fsync () is called. Calling fsync () significantly reduces the efficiency of metadata disk writing. In some environments, IOPS may drop to several hundred and the latency increases to 2-3ms (but it's still enough for Doris metadata manipulation). Therefore, we recommend the following configuration:

1. For a single Follower FE deployment, `master_sync_policy` is set to `SYNC`, which prevents the loss of metadata due to the downtime of the FE system.
2. For multi-Follower FE deployment, we can set `master_sync_policy` and `replica_sync_policy` to `WRITE_NO_SYNC`, because we think that the probability of simultaneous outage of multiple systems is very low.

If `master_sync_policy` is set to `WRITE_NO_SYNC` in a single Follower FE deployment, then a FE system outage may occur, resulting in loss of metadata. At this point, if other Observer FE attempts to restart, it may report an error:

```
Node xxx must rollback xx total commits(numPassedDurableCommits of which were durable) to the earliest point indicated by transaction xxxx in order to rejoin the replication group, but the transaction rollback limit of xxx prohibits this.
```

This means that some transactions that have been persisted need to be rolled back, but the number of entries exceeds the upper limit. Here our default upper limit is 100, which can be changed by setting `txn_rollback_limit`. This operation is only used to attempt to start FE normally, but lost metadata cannot be recovered.

Automated Service Startup

Automatic service start

This document describes how to configure the automatic pull-up of the Doris cluster to ensure that services are not pulled up in time after service breaks down due to special circumstances in the production environment.

The automatic pull-up service of FE and BE must be configured after the Doris cluster is completely set up.

Systemd Configures the Doris service

For details about systemd usage and parameter parsing, see here

sudo permission control

sudo permissions are required to control the doris service using systemd. To ensure the minimum granularity of sudo permission assignment, you can assign the systemd control permission of doris-fe and doris-be services to specified non-root users. Configure the systemctl management permission for doris-fe and doris-be in visudo.

```
Cmnd_Alias DORISCTL=/usr/bin/systemctl start doris-fe,/usr/bin/systemctl stop doris-fe,/usr/bin/systemctl start doris-
be,/usr/bin/systemctl stop doris-be

## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
doris   ALL=(ALL)       NOPASSWD:DORISCTL
```

Configuration procedure

1. You should config the “JAVA_HOME” variable in the config file, both fe.conf and be.conf, or you can’t use the command “systemctl start” to start doris

   ```
   echo "JAVA_HOME=your_java_home" >> /home/doris/fe/conf/fe.conf
   echo "JAVA_HOME=your_java_home" >> /home/doris/be/conf/be.conf
   ```

2. Download the doris-fe.service file: doris-fe.service

3. The details of doris-fe.service are as follows:

   ```
   # Licensed to the Apache Software Foundation (ASF) under one
   # or more contributor license agreements.  See the NOTICE file
   # distributed with this work for additional information
   # regarding copyright ownership.  The ASF licenses this file
   # to you under the Apache License, Version 2.0 (the
   # "License"); you may not use this file except in compliance
   # with the License.  You may obtain a copy of the License at
   #
   #   http://www.apache.org/licenses/LICENSE-2.0
   #
   ```

```
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.

[Unit]
Description=Doris FE
After=network-online.target
Wants=network-online.target

[Service]
Type=forking
User=root
Group=root
LimitCORE=infinity
LimitNOFILE=200000
Restart=on-failure
RestartSec=30
StartLimitInterval=120
StartLimitBurst=3
KillMode=none
ExecStart=/home/doris/fe/bin/start_fe.sh --daemon
ExecStop=/home/doris/fe/bin/stop_fe.sh

[Install]
WantedBy=multi-user.target
```

Matters needing attention

- ExecStart and ExecStop are configured based on actual fe paths

4. Download the doris-be.service file : doris-be.service

5. The details of doris-be.service are as follows: ``` # Licensed to the Apache Software Foundation (ASF) under one # or more contributor license agreements. See the NOTICE file # distributed with this work for additional information # regarding copyright ownership. The ASF licenses this file # to you under the Apache License, Version 2.0 (the # “License”); you may not use this file except in compliance # with the License. You may obtain a copy of the License at # # http://www.apache.org/licenses/LICENSE-2.0 # # Unless required by applicable law or agreed to in writing, # software distributed under the License is distributed on an # “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY # KIND, either express or implied. See the License for the # specific language governing permissions and limitations # under the License.

   [Unit] Description=Doris BE After=network-online.target Wants=network-online.target

   [Service] Type=forking User=root Group=root LimitCORE=infinity LimitNOFILE=200000 Restart=on-failure RestartSec=30 StartLimitInterval=120 StartLimitBurst=3 KillMode=none ExecStart=/home/doris/be/bin/start_be.sh –daemon ExecStop=/home/doris/be/bin/stop_be.sh

   [Install] WantedBy=multi-user.target ```

Matters needing attention

- ExecStart and ExecStop are configured based on actual be paths

6. Service configuration

   Place doris-fe.service and doris-be.service in the /usr/lib/systemd/system directory

7. Set self-start

   After you add or modify the configuration file, you need to reload it

   ```
   systemctl daemon-reload
   ```

   Set the start, the essence is in the/etc/systemd/system/multi - user. Target. Wants/add service file link

   ```
   systemctl enable doris-fe
   systemctl enable doris-be
   ```

8. Service initiation

   ```
   systemctl start doris-fe
   systemctl start doris-be
   ```

Supervisor configures the Doris service

Supervisor Specific use and parameter analysis can be referred to here

Supervisor configuration automatically pulls up the supervisor configuration. You can install the supervisor directly using the yum command or manually using pip. The pip manual installation process is complicated, and only the yum deployment mode is displayed.Manual deployment refer to [here] (http://supervisord.org/installing.html) for installation deployment.

Configuration procedure

1. yum Install supervisor

   ```
   yum install epel-release
   yum install -y supervisor
   ```

2. Start the service and view the status

   ```
   systemctl enable supervisord # bootstrap
   systemctl start supervisord # Start the supervisord service
   systemctl status supervisord # Check the supervisord service status
   ps -ef|grep supervisord # Check whether the supervisord process exists
   ```

3. Configure BE process management

   ```
   Modify the start_be.sh script remove the last symbol &

   vim /path/doris/be/bin/start_be.sh
   Take this code : nohup $LIMIT ${DORIS_HOME}/lib/palo_be "$@" >> $LOG_DIR/be.out 2>&1 </dev/null &
   Be changed to : nohup $LIMIT ${DORIS_HOME}/lib/palo_be "$@" >> $LOG_DIR/be.out 2>&1 </dev/null
   ```

Create a supervisor process management configuration file for the BE

```
vim /etc/supervisord.d/doris-be.ini

[program:doris_be]
process_name=%(program_name)s
directory=/path/doris/be/be
command=sh /path/doris/be/bin/start_be.sh
autostart=true
autorestart=true
user=root
numprocs=1
startretries=3
stopasgroup=true
killasgroup=true
startsecs=5
#redirect_stderr = true
#stdout_logfile_maxbytes = 20MB
#stdout_logfile_backups = 10
#stdout_logfile=/var/log/supervisor-palo_be.log
```

4. Configure FE process management

```
Modify the start_fe.sh script remove the last symbol &

vim /path/doris/fe/bin/start_fe.sh
Take this code : nohup $LIMIT $JAVA $final_java_opt org.apache.doris.PaloFe ${HELPER} "$@" >> $LOG_DIR/fe.out 2>&1
</dev/null &
Be changed to : nohup $LIMIT $JAVA $final_java_opt org.apache.doris.PaloFe ${HELPER} "$@" >> $LOG_DIR/fe.out 2>&1
</dev/null
```

Create a supervisor process management configuration file for FE

```
vim /etc/supervisord.d/doris-fe.ini

[program:PaloFe]
environment = JAVA_HOME="/usr/local/java"
process_name=PaloFe
directory=/path/doris/fe
command=sh /path/doris/fe/bin/start_fe.sh
autostart=true
autorestart=true
user=root
numprocs=1
startretries=3
stopasgroup=true
killasgroup=true
startsecs=10
#redirect_stderr=true
#stdout_logfile_maxbytes=20MB
#stdout_logfile_backups=10
#stdout_logfile=/var/log/supervisor-PaloFe.log
```

5. Configure Broker process management

```
Modify the start_broker.sh script remove the last symbol &

vim /path/apache_hdfs_broker/bin/start_broker.sh
Take this code : nohup $LIMIT $JAVA $JAVA_OPTS org.apache.doris.broker.hdfs.BrokerBootstrap "$@" >>
```

```
$BROKER_LOG_DIR/apache_hdfs_broker.out 2>&1 </dev/null &
Be changed to : nohup $LIMIT $JAVA $JAVA_OPTS org.apache.doris.broker.hdfs.BrokerBootstrap "$@" >>
$BROKER_LOG_DIR/apache_hdfs_broker.out 2>&1 </dev/null
```

Create the supervisor process management profile for the Broker

```
vim /etc/supervisord.d/doris-broker.ini

[program:BrokerBootstrap]
environment = JAVA_HOME="/usr/local/java"
process_name=%(program_name)s
directory=/path/apache_hdfs_broker
command=sh /path/apache_hdfs_broker/bin/start_broker.sh
autostart=true
autorestart=true
user=root
numprocs=1
startretries=3
stopasgroup=true
killasgroup=true
startsecs=5
#redirect_stderr=true
#stdout_logfile_maxbytes=20MB
#stdout_logfile_backups=10
#stdout_logfile=/var/log/supervisor-BrokerBootstrap.log
```

6. First determine whether the Doris service is stopped, then use supervisor to automatically pull up Doris, and then determine whether the process starts normally

```
supervisorctl reload # Reload all the Supervisor configuration files
supervisorctl status # Check the supervisor status and verify that the Doris service process starts normally

其他命令 :
supervisorctl start all # supervisorctl start It is capable of opening processes
supervisorctl stop doris-be # The process is supervisorctl stop
```

Matters needing attention:

- If the supervisor installed using yum starts, an error occurs: pkg_resources.DistributionNotFound: The ‘supervisor==3.4.0’ distribution was not found

```
supervisor installed directly using the yum command only supports python2,Therefore, the file contents in /usr/bin/supervisorctl and /usr/bin/supervisorctl should be changed at the beginning Change #! /usr/bin/python to #! /usr/bin/python2, python2 must be installed
```

- If the supervisor is configured to automatically pull up the Doris process, if the BE node breaks down due to abnormal factors on Doris, the error stack information that should be output to be.out will be intercepted by the supervisor. We need to look it up in supervisor’s log for further analysis.

Debug Log

Debug Log

The system operation logs of Doris's FE and BE nodes are at INFO level by default. It can usually satisfy the analysis of system behavior and the localization of basic problems. However, in some cases, it may be necessary to enable DEBUG level logs to further troubleshoot the problem. This document mainly introduces how to enable the DEBUG log level of FE and BE nodes.

> It is not recommended to adjust the log level to WARN or higher, which is not conducive to the analysis of system behavior and the location of problems.

> Enable DEBUG log may cause a large number of logs to be generated, **Please be careful to open it in production environment**.

Enable FE Debug Log

The Debug level log of FE can be turned on by modifying the configuration file, or it can be turned on at runtime through the interface or API.

1. Open via configuration file

   Add the configuration item `sys_log_verbose_modules` to fe.conf. An example is as follows:

   ```
   # Only enable Debug log for class org.apache.doris.catalog.Catalog
   sys_log_verbose_modules=org.apache.doris.catalog.Catalog

   # Open the Debug log of all classes under the package org.apache.doris.catalog
   sys_log_verbose_modules=org.apache.doris.catalog

   # Enable Debug logs for all classes under package org
   sys_log_verbose_modules=org
   ```

   Add configuration items and restart the FE node to take effect.

2. Via FE UI interface

   The log level can be modified at runtime through the UI interface. There is no need to restart the FE node. Open the http port of the FE node (8030 by default) in the browser, and log in to the UI interface. Then click on the `Log` tab in the upper navigation bar.

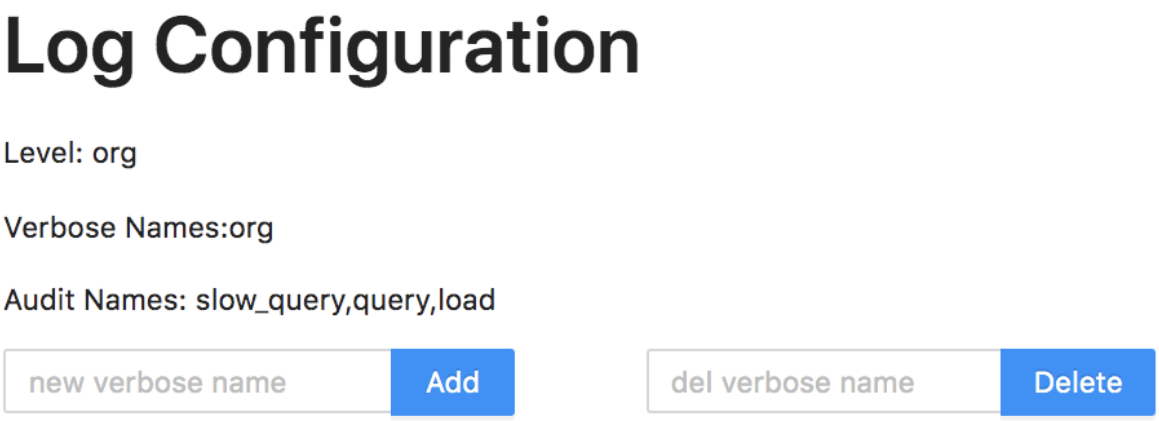


image.png

We can enter the package name or specific class name in the Add input box to open the corresponding Debug log. For example, enter `org.apache.doris.catalog.Catalog` to open the Debug log of the Catalog class:

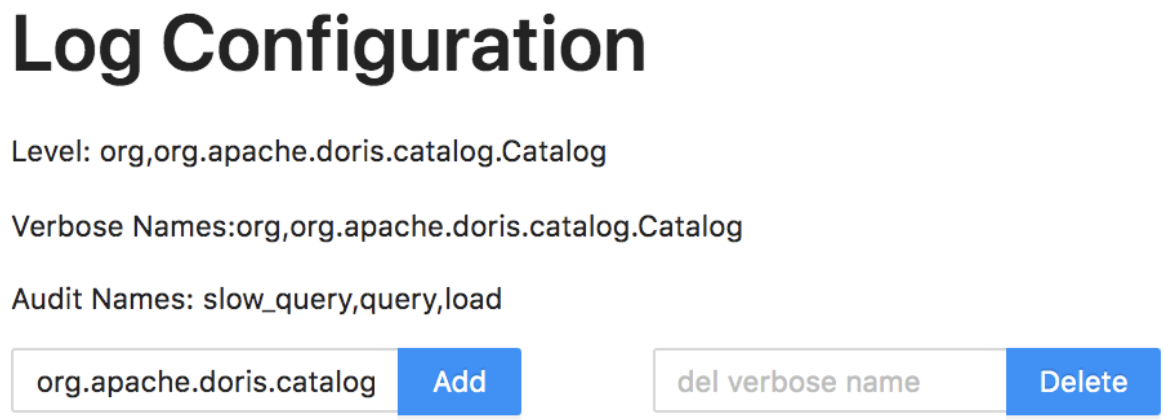


image.png

You can also enter the package name or specific class name in the Delete input box to close the corresponding Debug log.

> The modification here will only affect the log level of the corresponding FE node. Does not affect the log level of other FE nodes.

3. Modification via API

   The log level can also be modified at runtime via the following API. There is no need to restart the FE node.

   ```
   curl -X POST -uuser:passwd fe_host:http_port/rest/v1/log?add_verbose=org.apache.doris.catalog.Catalog
   ```

   The username and password are the root or admin users who log in to Doris. The `add_verbose` parameter specifies the package or class name to enable Debug logging. Returns if successful:

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "LogConfiguration": {
            "VerboseNames": "org,org.apache.doris.catalog.Catalog",
            "AuditNames": "slow_query,query,load",
            "Level": "INFO"
        }
    },
    "count": 0
}
```

Debug logging can also be turned off via the following API:

```
curl -X POST -uuser:passwd fe_host:http_port/rest/v1/log?del_verbose=org.apache.doris.catalog.Catalog
```

The `del_verbose` parameter specifies the package or class name for which to turn off Debug logging.

Enable BE Debug Log

BE's Debug log currently only supports modifying and restarting the BE node through the configuration file to take effect.

```
sys_log_verbose_modules=plan_fragment_executor,olap_scan_node
sys_log_verbose_level=3
```

`sys_log_verbose_modules` specifies the file name to be opened, which can be specified by the wildcard *. for example:

```
sys_log_verbose_modules=*
```

Indicates that all DEBUG logs are enabled.

`sys_log_verbose_level` indicates the level of DEBUG. The higher the number, the more detailed the DEBUG log. The value range is 1-10.

Config Dir

Config Dir

The configuration file directory for FE and BE is `conf/`. In addition to storing the default fe.conf, be.conf and other files, this directory is also used for the common configuration file storage directory.

Users can store some configuration files in it, and the system will automatically read them.

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

hdfs-site.xml and hive-site.xml

In some functions of Doris, you need to access data on HDFS, or access Hive metastore.

We can manually fill in various HDFS/Hive parameters in the corresponding statement of the function.

But these parameters are very many, if all are filled in manually, it is very troublesome.

Therefore, users can place the HDFS or Hive configuration file hdfs-site.xml/hive-site.xml directly in the `conf/` directory. Doris will automatically read these configuration files.

The configuration that the user fills in the command will overwrite the configuration items in the configuration file.

In this way, users only need to fill in a small amount of configuration to complete the access to HDFS/Hive.

FE Configuration

FE Configuration

This document mainly introduces the relevant configuration items of FE.

The FE configuration file `fe.conf` is usually stored in the `conf/` directory of the FE deployment path. In version 0.14, another configuration file `fe_custom.conf` will be introduced. The configuration file is used to record the configuration items that are dynamically configured and persisted by the user during operation.

After the FE process is started, it will read the configuration items in `fe.conf` first, and then read the configuration items in `fe_custom.conf`. The configuration items in `fe_custom.conf` will overwrite the same configuration items in `fe.conf`.

The location of the `fe_custom.conf` file can be configured in `fe.conf` through the `custom_config_dir` configuration item.

View configuration items

There are two ways to view the configuration items of FE:

1. FE web page

   Open the FE web page `http://fe_host:fe_http_port/variable` in the browser. You can see the currently effective FE configuration items in `Configure Info`.

2. View by command

   After the FE is started, you can view the configuration items of the FE in the MySQL client with the following command,Concrete language law reference ADMIN-SHOW-CONFIG:

   ```
   ADMIN SHOW FRONTEND CONFIG;
   ```

   The meanings of the columns in the results are as follows:

   - Key: the name of the configuration item.
   - Value: The value of the current configuration item.
   - Type: The configuration item value type, such as integer or string.
   - IsMutable: whether it can be dynamically configured. If true, the configuration item can be dynamically configured at runtime. If false, it means that the configuration item can only be configured in `fe.conf` and takes effect after restarting FE.
   - MasterOnly: Whether it is a unique configuration item of Master FE node. If it is true, it means that the configuration item is meaningful only at the Master FE node, and is meaningless to other types of FE nodes. If false, it means that the configuration item is meaningful in all types of FE nodes.
   - Comment: The description of the configuration item.

Set configuration items

There are two ways to configure FE configuration items:

1. Static configuration

   Add and set configuration items in the `conf/fe.conf` file. The configuration items in `fe.conf` will be read when the FE process starts. Configuration items not in `fe.conf` will use default values.

2. Dynamic configuration via MySQL protocol

   After the FE starts, you can set the configuration items dynamically through the following commands. This command requires administrator privilege.

   ```
   ADMIN SET FRONTEND CONFIG (" fe_config_name "=" fe_config_value ");
   ```

   Not all configuration items support dynamic configuration. You can check whether the dynamic configuration is supported by the `IsMutable` column in the`ADMIN SHOW FRONTEND CONFIG;`command result.

   If the configuration item of `MasterOnly` is modified, the command will be directly forwarded to the Master FE and only the corresponding configuration item in the Master FE will be modified.

   Configuration items modified in this way will become invalid after the FE process restarts.

   For more help on this command, you can view it through the `HELP ADMIN SET CONFIG;` command.

3. Dynamic configuration via HTTP protocol

   For details, please refer to Set Config Action

   This method can also persist the modified configuration items. The configuration items will be persisted in the `fe_custom.conf` file and will still take effect after FE is restarted.

Examples

1. Modify `async_pending_load_task_pool_size`

   Through `ADMIN SHOW FRONTEND CONFIG;` you can see that this configuration item cannot be dynamically configured (`IsMutable` is false). You need to add in `fe.conf`:

   ```
   async_pending_load_task_pool_size = 20
   ```

   Then restart the FE process to take effect the configuration.

2. Modify `dynamic_partition_enable`

   Through `ADMIN SHOW FRONTEND CONFIG;` you can see that the configuration item can be dynamically configured (`IsMutable` is true). And it is the unique configuration of Master FE. Then first we can connect to any FE and execute the following command to modify the configuration:

   ```
   ADMIN SET FRONTEND CONFIG ("dynamic_partition_enable" = "true"); `
   ```

   Afterwards, you can view the modified value with the following command:

```
set forward_to_master = true;
ADMIN SHOW FRONTEND CONFIG;
```

After modification in the above manner, if the Master FE restarts or a Master election is performed, the configuration will be invalid. You can add the configuration item directly in `fe.conf` and restart the FE to make the configuration item permanent.

3. Modify `max_distribution_pruner_recursion_depth`

   Through `ADMIN SHOW FRONTEND CONFIG;` you can see that the configuration item can be dynamically configured (`IsMutable` is true). It is not unique to Master FE.

   Similarly, we can modify the configuration by dynamically modifying the configuration command. Because this configuration is not unique to the Master FE, user need to connect to different FEs separately to modify the configuration dynamically, so that all FEs use the modified configuration values.

Configurations

Metadata And Cluster

`meta_dir`

Default：DORIS_HOME_DIR + “/doris-meta”

Type: string Description: Doris meta data will be saved here.The storage of this dir is highly recommended as to be:

- High write performance (SSD)
- Safe (RAID）

`catalog_try_lock_timeout_ms`

Default：5000 （ms）

IsMutable：true

The tryLock timeout configuration of catalog lock. Normally it does not need to change, unless you need to test something.

`enable_bdbje_debug_mode`

Default：false

If set to true, FE will be started in BDBJE debug mode

`max_bdbje_clock_delta_ms`

Default：5000 （5s）

Set the maximum acceptable clock skew between non-master FE to Master FE host. This value is checked whenever a non-master FE establishes a connection to master FE via BDBJE. The connection is abandoned if the clock skew is larger than this value.

`metadata_failure_recovery`

Default：false

If true, FE will reset bdbje replication group(that is, to remove all electable nodes info) and is supposed to start as Master. If all the electable nodes can not start, we can copy the meta data to another node and set this config to true to try to restart the FE..

`txn_rollback_limit`

Default：100

the max txn number which bdbje can rollback when trying to rejoin the group

`grpc_threadmgr_threads_nums`

Default: 4096

Num of thread to handle grpc events in grpc_threadmgr.

`bdbje_replica_ack_timeout_second`

Default：10 (s)

The replica ack timeout when writing to bdbje , When writing some relatively large logs, the ack time may time out, resulting in log writing failure. At this time, you can increase this value appropriately.

`bdbje_lock_timeout_second`

Default：5

The lock timeout of bdbje operation, If there are many LockTimeoutException in FE WARN log, you can try to increase this value

`bdbje_heartbeat_timeout_second`

Default：30

The heartbeat timeout of bdbje between master and follower. the default is 30 seconds, which is same as default value in bdbje. If the network is experiencing transient problems, of some unexpected long java GC annoying you, you can try to increase this value to decrease the chances of false timeouts

`replica_ack_policy`

Default：SIMPLE_MAJORITY

OPTION: ALL, NONE, SIMPLE_MAJORITY

Replica ack policy of bdbje. more info, see:
http://docs.oracle.com/cd/E17277_02/html/java/com/sleepycat/je/Durability.ReplicaAckPolicy.html

`replica_sync_policy`

Default：SYNC

选项：SYNC, NO_SYNC, WRITE_NO_SYNC

Follower FE sync policy of bdbje.

`master_sync_policy`

Default：SYNC

选项：SYNC, NO_SYNC, WRITE_NO_SYNC

Master FE sync policy of bdbje. If you only deploy one Follower FE, set this to 'SYNC'. If you deploy more than 3 Follower FE, you can set this and the following 'replica_sync_policy' to WRITE_NO_SYNC. more info, see:
http://docs.oracle.com/cd/E17277_02/html/java/com/sleepycat/je/Durability.SyncPolicy.html

`bdbje_reserved_disk_bytes`

The desired upper limit on the number of bytes of reserved space to retain in a replicated JE Environment.

Default: 1073741824

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: false

`ignore_meta_check`

Default：false

IsMutable：true

If true, non-master FE will ignore the meta data delay gap between Master FE and its self, even if the metadata delay gap exceeds *meta_delay_toleration_second*. Non-master FE will still offer read service. This is helpful when you try to stop the Master FE for a relatively long time for some reason, but still wish the non-master FE can offer read service.

`meta_delay_toleration_second`

Default: 300 (5 min)

Non-master FE will stop offering service if meta data delay gap exceeds *meta_delay_toleration_second*

`edit_log_port`

Default：9010

bdbje port

`edit_log_type`

Default：BDB

Edit log type. BDB: write log to bdbje LOCAL: deprecated..

`edit_log_roll_num`

Default：50000

IsMutable：true

MasterOnly：true

Master FE will save image every *edit_log_roll_num* meta journals.

`force_do_metadata_checkpoint`

Default：false

IsMutable：true

MasterOnly：true

If set to true, the checkpoint thread will make the checkpoint regardless of the jvm memory used percent

`metadata_checkpoint_memory_threshold`

Default：60 （60%）

IsMutable：true

MasterOnly：true

If the jvm memory used percent(heap or old mem pool) exceed this threshold, checkpoint thread will not work to avoid OOM.

`max_same_name_catalog_trash_num`

It is used to set the maximum number of meta information with the same name in the catalog recycle bin. When the maximum value is exceeded, the earliest deleted meta trash will be completely deleted and cannot be recovered. 0 means not to keep objects of the same name. < 0 means no limit.

Note: The judgment of metadata with the same name will be limited to a certain range. For example, the judgment of the database with the same name will be limited to the same cluster, the judgment of the table with the same name will be limited to the same database (with the same database id), the judgment of the partition with the same name will be limited to the same database (with the same database id) and the same table (with the same table) same table id).

Default: 3

Is it possible to dynamically configure: true

Is it a configuration item unique to the Master FE node: true

`cluster_id`

Default：-1

node(FE or BE) will be considered belonging to the same Palo cluster if they have same cluster id. Cluster id is usually a random integer generated when master FE start at first time. You can also specify one.

`heartbeat_mgr_blocking_queue_size`

Default：1024

MasterOnly：true

blocking queue size to store heartbeat task in heartbeat_mgr.

`heartbeat_mgr_threads_num`

Default：8

MasterOnly：true

num of thread to handle heartbeat events in heartbeat_mgr.

`disable_cluster_feature`

Default：true

IsMutable：true

The multi cluster feature will be deprecated in version 0.12 , set this config to true will disable all operations related to cluster feature, include:

1. create/drop cluster
2. add free backend/add backend to cluster/decommission cluster balance
3. change the backends num of cluster
4. link/migration db

`enable_fqdn_mode`

This configuration is mainly used in the k8s cluster environment. When enable_fqdn_mode is true, the name of the pod where the be is located will remain unchanged after reconstruction, while the ip can be changed.

Default: false

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: true

`enable_token_check`

Default: true

For forward compatibility, will be removed later. check token when download image file.

`enable_multi_tags`

Default: false

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: true

Whether to enable the multi-tags function of a single BE

`initial_root_password`

Set root user initial 2-staged SHA-1 encrypted password, default as '', means no root password. Subsequent `set password` operations for root user will overwrite the initial root password.

Example: If you want to configure a plaintext password `root@123`. You can execute Doris SQL `select password('root@123')` to generate encrypted password `*A00C34073A26B40AB4307650BFB9309D6BFA6999`.

Default: empty string

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: true

Service

`query_port`

Default：9030

FE MySQL server port

`arrow_flight_sql_port`

Default：-1

Arrow Flight SQL server port

`frontend_address`

Status: Deprecated, not recommended use. This parameter may be deleted later

Type: string

Description: Explicitly set the IP address of FE instead of using *InetAddress.getByName* to get the IP address. Usually in *InetAddress.getByName* When the expected results cannot be obtained. Only IP address is supported, not hostname.

Default value: 0.0.0.0

`priority_networks`

Default：none

Declare a selection strategy for those servers have many ips. Note that there should at most one ip match this list. this is a list in semicolon-delimited format, in CIDR notation, e.g. 10.10.10.0/24 , If no ip match this rule, will choose one randomly.

`http_port`

Default：8030

HTTP bind port. All FE http ports must be same currently.

`https_port`

Default：8050

HTTPS bind port. All FE https ports must be same currently.

`enable_https`

Default：false

Https enable flag. If the value is false, http is supported. Otherwise, both http and https are supported, and http requests are automatically redirected to https. If enable_https is true, you need to configure ssl certificate information in fe.conf.

`enable_ssl`

Default：true

If set to ture, doris will establish an encrypted channel based on the SSL protocol with mysql.

`qe_max_connection`

Default：1024

Maximal number of connections per FE.

`check_java_version`

Default：true

Doris will check whether the compiled and run Java versions are compatible, if not, it will throw a Java version mismatch exception message and terminate the startup

`rpc_port`

Default：9020

FE Thrift Server port

`thrift_server_type`

This configuration represents the service model used by The Thrift Service of FE, is of type String and is case-insensitive.

If this parameter is ‘SIMPLE’, then the ‘TSimpleServer’ model is used, which is generally not suitable for production and is limited to test use.

If the parameter is ‘THREADED’, then the ‘TThreadedSelectorServer’ model is used, which is a non-blocking I/O model, namely the master-slave Reactor model, which can timely respond to a large number of concurrent connection requests and performs well in most scenarios.

If this parameter is `THREAD_POOL`, then the `TThreadPoolServer` model is used, the model for blocking I/O model, use the thread pool to handle user connections, the number of simultaneous connections are limited by the number of thread pool, if we can estimate the number of concurrent requests in advance, and tolerant enough thread resources cost, this model will have a better performance, the service model is used by default

`thrift_server_max_worker_threads`

Default：4096

The thrift server max worker threads

`thrift_backlog_num`

Default：1024

The backlog_num for thrift server , When you enlarge this backlog_num, you should ensure it’s value larger than the linux /proc/sys/net/core/somaxconn config

`thrift_client_timeout_ms`

Default：0

The connection timeout and socket timeout config for thrift server.

The value for thrift_client_timeout_ms is set to be zero to prevent read timeout.

`thrift_max_message_size`

> **Note**
>
> This field is supported since version 2.0.12.

Default: 100MB

The maximum size of a (received) message of the thrift server, in bytes. If the size of the message sent by the client exceeds this limit, the Thrift server will reject the request and close the connection. As a result, the client will encounter the error: “connection has been closed by peer.” In this case, you can try increasing this parameter.

`use_compact_thrift_rpc`

Default: true

Whether to use compressed format to send query plan structure. After it is turned on, the size of the query plan structure can be reduced by about 50%, thereby avoiding some “send fragment timeout” errors. However, in some high-concurrency small query scenarios, the concurrency may be reduced by about 10%.

`grpc_max_message_size_bytes`

Default：1G

Used to set the initial flow window size of the GRPC client channel, and also used to max message size. When the result set is large, you may need to increase this value.

`max_mysql_service_task_threads_num`

Default：4096

The number of threads responsible for Task events.

`mysql_service_io_threads_num`

Default：4

When FE starts the MySQL server based on NIO model, the number of threads responsible for IO events.

`mysql_nio_backlog_num`

Default：1024

The backlog_num for mysql nio server, When you enlarge this backlog_num, you should enlarge the value in the linux /proc/sys/net/core/somaxconn file at the same time

`broker_timeout_ms`

Default：10000 （10s）

Default broker RPC timeout

`backend_rpc_timeout_ms`

Timeout millisecond for Fe sending rpc request to BE

Default: 60000

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: true

`drop_backend_after_decommission`

Default：false

IsMutable：true

MasterOnly：true

1. This configuration is used to control whether the system drops the BE after successfully decommissioning the BE. If true, the BE node will be deleted after the BE is successfully offline. If false,

after the BE successfully goes offline, the BE will remain in the DECOMMISSION state, but will not be dropped.

This configuration can play a role in certain scenarios. Assume that the initial state of a Doris cluster is one disk per BE node. After running for a period of time, the system has been vertically expanded, that is, each BE node adds 2 new disks. Because Doris currently does not support data balancing among the disks within the BE, the data volume of the initial disk may always be much higher than the data volume of the newly added disk. At this time, we can perform manual inter-disk balancing by the following operations:

1. Set the configuration item to false.
2. Perform a decommission operation on a certain BE node. This operation will migrate all data on the BE to other nodes.
3. After the decommission operation is completed, the BE will not be dropped. At this time, cancel the decommission status of the BE. Then the data will start to balance from other BE nodes back to this node. At this time, the data will be evenly distributed to all disks of the BE.
4. Perform steps 2 and 3 for all BE nodes in sequence, and finally achieve the purpose of disk balancing for all nodes

`max_backend_down_time_second`

Default: 3600 (1 hour)

IsMutable：true

MasterOnly：true

If a backend is down for *max_backend_down_time_second*, a BACKEND_DOWN event will be triggered.

`disable_backend_black_list`

Used to disable the BE blacklist function. After this function is disabled, if the query request to the BE fails, the BE will not be added to the blacklist. This parameter is suitable for regression testing environments to reduce occasional bugs that cause a large number of regression tests to fail.

Default: false

Is it possible to configure dynamically: true

Is it a configuration item unique to the Master FE node: false

`max_backend_heartbeat_failure_tolerance_count`

The maximum tolerable number of BE node heartbeat failures. If the number of consecutive heartbeat failures exceeds this value, the BE state will be set to dead. This parameter is suitable for regression test environments to reduce occasional heartbeat failures that cause a large number of regression test failures.

Default: 1

Is it possible to configure dynamically: true

Whether it is a configuration item unique to the Master FE node: true

`enable_access_file_without_broker`

Default：false

IsMutable：true

MasterOnly：true

This config is used to try skip broker when access bos or other cloud storage via broker

`agent_task_resend_wait_time_ms`

Default：5000

IsMutable：true

MasterOnly：true

This configuration will decide whether to resend agent task when create_time for agent_task is set, only when current_time - create_time > agent_task_resend_wait_time_ms can ReportHandler do resend agent task.

This configuration is currently mainly used to solve the problem of repeated sending of `PUBLISH_VERSION` agent tasks. The current default value of this configuration is 5000, which is an experimental value.

Because there is a certain time delay between submitting agent tasks to AgentTaskQueue and submitting to be, Increasing the value of this configuration can effectively solve the problem of repeated sending of agent tasks,

But at the same time, it will cause the submission of failed or failed execution of the agent task to be executed again for an extended period of time

`max_agent_task_threads_num`

Default：4096

MasterOnly：true

max num of thread to handle agent task in agent task thread-pool.

`remote_fragment_exec_timeout_ms`

Default：30000 （ms）

IsMutable：true

The timeout of executing async remote fragment. In normal case, the async remote fragment will be executed in a short time. If system are under high load condition, try to set this timeout longer.

`auth_token`

Default：empty

Cluster token used for internal authentication.

`enable_http_server_v2`

Default: The default is true after the official 0.14.0 version is released, and the default is false before

HTTP Server V2 is implemented by SpringBoot. It uses an architecture that separates the front and back ends. Only when httpv2 is enabled can users use the new front-end UI interface.

`http_api_extra_base_path`

In some deployment environments, user need to specify an additional base path as the unified prefix of the HTTP API. This parameter is used by the user to specify additional prefixes. After setting, user can get the parameter value through the `GET /api/basepath` interface. And the new UI will also try to get this base path first to assemble the URL. Only valid when `enable_http_server_v2` is true.

The default is empty, that is, not set

`jetty_server_acceptors`

Default：2

`jetty_server_selectors`

Default：4

`jetty_server_workers`

Default：0

With the above three parameters, Jetty's thread architecture model is very simple, divided into acceptors, selectors and workers three thread pools. Acceptors are responsible for accepting new connections, and then hand them over to selectors to process the unpacking of the HTTP message protocol, and finally workers process the request. The first two thread pools adopt a non-blocking model, and one thread can handle the read and write of many sockets, so the number of thread pools is small.

For most projects, only 1-2 acceptors threads are required, and 2 to 4 selectors threads are sufficient. Workers are obstructive business logic, often have more database operations, and require a large number of threads. The specific number depends on the proportion of QPS and IO events of the application. The higher the QPS, the more threads are required, the higher the proportion of IO, the more threads waiting, and the more total threads required.

Worker thread pool is not set by default, set according to your needs

`jetty_server_max_http_post_size`

Default: `100 * 1024 * 1024` (100MB)

This is the maximum number of bytes of the file uploaded by the put or post method, the default value: 100MB

`jetty_server_max_http_header_size`

Default: 1048576 (1M)

http header size configuration parameter, the default value is 1M.

`http_sql_submitter_max_worker_threads`

Default: 2

The max number work threads of http sql submitter

`http_load_submitter_max_worker_threads`

Default: 2

The max number work threads of http upload submitter

Query Engine

`default_max_query_instances`

The default value when user property max_query_instances is equal or less than 0. This config is used to limit the max number of instances for a user. This parameter is less than or equal to 0 means unlimited.

The default value is -1

`max_query_retry_time`

Default: 3

IsMutable: true

The number of query retries. A query may retry if we encounter RPC exception and no result has been sent to user. You may reduce this number to avoid Avalanche disaster

`max_dynamic_partition_num`

Default: 500

IsMutable: true

MasterOnly: true

Used to limit the maximum number of partitions that can be created when creating a dynamic partition table, to avoid creating too many partitions at one time. The number is determined by “start” and “end” in the dynamic partition parameters.

`dynamic_partition_enable`

Default：true

IsMutable：true

MasterOnly：true

Whether to enable dynamic partition scheduler, enabled by default

`dynamic_partition_check_interval_seconds`

Default：600 （s）

IsMutable：true

MasterOnly：true

Decide how often to check dynamic partition

> **Tips**
>
> This configuration is supported since the Apache Doris 1.2 version

`max_multi_partition_num`

Default：4096

IsMutable：true

MasterOnly：true

Use this parameter to set the partition name prefix for multi partition,Only multi partition takes effect, not dynamic partitions. The default prefix is “p_”.

`multi_partition_name_prefix`

Default：p_

IsMutable：true

MasterOnly：true

Use this parameter to set the partition name prefix for multi partition, Only multi partition takes effect, not dynamic partitions.The default prefix is “p_”.

`partition_in_memory_update_interval_secs`

Default：300 (s)

IsMutable：true

MasterOnly：true

Time to update global partition information in memory

`enable_concurrent_update`

Default：false

IsMutable：false

MasterOnly：true

Whether to enable concurrent update

`lower_case_table_names`

Default：0

IsMutable：false

MasterOnly：true

This configuration can only be configured during cluster initialization and cannot be modified during cluster restart and upgrade after initialization is complete.

0: table names are stored as specified and comparisons are case sensitive. 1: table names are stored in lowercase and comparisons are not case sensitive. 2: table names are stored as given but compared in lowercase.

`table_name_length_limit`

Default：64

IsMutable：true

MasterOnly：true

Used to control the maximum table name length

`cache_enable_sql_mode`

Default：true

IsMutable：true

MasterOnly：false

If this switch is turned on, the SQL query result set will be cached. If the interval between the last visit version time in all partitions of all tables in the query is greater than cache_last_version_interval_second, and the result set is less than cache_result_max_row_count, and the data size is less than cache_result_max_data_size, the result set will be cached, and the next same SQL will hit the cache

If set to true, fe will enable sql result caching. This option is suitable for offline data update scenarios

	case1	case2	case3	case4
enable_sql_cache	false	true	true	false
enable_partition_cache	false	false	true	true

`cache_enable_partition_mode`

Default：true

IsMutable：true

MasterOnly：false

If set to true, fe will get data from be cache, This option is suitable for real-time updating of partial partitions.

`cache_result_max_row_count`

Default：3000

IsMutable：true

MasterOnly：false

In order to avoid occupying too much memory, the maximum number of rows that can be cached is 3000 by default. If this threshold is exceeded, the cache cannot be set

`cache_result_max_data_size`

Default: 31457280

IsMutable: true

MasterOnly: false

In order to avoid occupying too much memory, the maximum data size of rows that can be cached is 10MB by default. If this threshold is exceeded, the cache cannot be set

`cache_last_version_interval_second`

Default：900

IsMutable：true

MasterOnly：false

The time interval of the latest partitioned version of the table refers to the time interval between the data update and the current version. It is generally set to 900 seconds, which distinguishes offline and real-time import

`enable_batch_delete_by_default`

Default：false

IsMutable：true

MasterOnly：true

Whether to add a delete sign column when create unique table

`max_allowed_in_element_num_of_delete`

Default：1024

IsMutable：true

MasterOnly：true

This configuration is used to limit element num of InPredicate in delete statement.

`max_running_rollup_job_num_per_table`

Default：1

IsMutable：true

MasterOnly：true

Control the concurrency limit of Rollup jobs

`max_distribution_pruner_recursion_depth`

Default：100

IsMutable：true

MasterOnly：false

This will limit the max recursion depth of hash distribution pruner. eg: where a in (5 elements) and b in (4 elements) and c in (3 elements) and d in (2 elements). a/b/c/d are distribution columns, so the recursion depth will be 5 *4* 3 * 2 = 120, larger than 100, So that distribution pruner will no work and just return all buckets. Increase the depth can support distribution pruning for more elements, but may cost more CPU.

`enable_local_replica_selection`

Default：false

IsMutable：true

If set to true, Planner will try to select replica of tablet on same host as this Frontend. This may reduce network transmission in following case:

- N hosts with N Backends and N Frontends deployed.
- The data has N replicas.
- High concurrency queries are syyuyuient to all Frontends evenly

In this case, all Frontends can only use local replicas to do the query. If you want to allow fallback to nonlocal replicas when no local replicas available, set enable_local_replica_selection_fallback to true.

`enable_local_replica_selection_fallback`

Default：false

IsMutable：true

Used with enable_local_replica_selection. If the local replicas is not available, fallback to the nonlocal replicas.

`expr_depth_limit`

Default：3000

IsMutable：true

Limit on the depth of an expr tree. Exceed this limit may cause long analysis time while holding db read lock. Do not set this if you know what you are doing

`expr_children_limit`

Default：10000

IsMutable：true

Limit on the number of expr children of an expr tree. Exceed this limit may cause long analysis time while holding database read lock.

`be_exec_version`

Used to define the serialization format for passing blocks between fragments.

Sometimes some of our code changes will change the data format of the block. In order to make the BE compatible with each other during the rolling upgrade process, we need to issue a data version from the FE to decide what format to send the data in.

Specifically, for example, there are 2 BEs in the cluster, one of which can support the latest v_1 after being upgraded, while the other only supports v_0. At this time, since the FE has not been upgraded yet, $v_0 is

issued uniformly. $, BE interact in the old data format. After all BEs are upgraded, we will upgrade FE. At this time, the new FE will issue ν_1, and the cluster will be uniformly switched to the new data format.

The default value is `max_be_exec_version`. If there are special needs, we can manually set the format version to lower, but it should not be lower than `min_be_exec_version`.

Note that we should always keep the value of this variable between `BeExecVersionManager::min_be_exec_version` and `BeExecVersionManager::max_be_exec_version` for all BEs. (That is to say, if a cluster that has completed the update needs to be downgraded, it should ensure the order of downgrading FE and then downgrading BE, or manually lower the variable in the settings and downgrade BE)

`max_be_exec_version`

The latest data version currently supported, cannot be modified, and should be consistent with the `BeExecVersionManager::max_be_exec_version` in the BE of the matching version.

`min_be_exec_version`

The oldest data version currently supported, which cannot be modified, should be consistent with the `BeExecVersionManager::min_be_exec_version` in the BE of the matching version.

`max_query_profile_num`

The max number of query profile.

Default: 100

Is it possible to dynamically configure: true

Is it a configuration item unique to the Master FE node: false

`publish_version_interval_ms`

Default：10 （ms）

minimal intervals between two publish version action

`publish_version_timeout_second`

Default：30 （s）

IsMutable：true

MasterOnly：true

Maximal waiting time for all publish version tasks of one transaction to be finished

`query_colocate_join_memory_limit_penalty_factor`

Default：1

IsMutable：true

colocote join PlanFragment instance 的 memory_limit = exec_mem_limit / min (query_colocate_join_memory_limit_penalty_factor, instance_num)

`rewrite_count_distinct_to_bitmap_hll`

Default: true

This variable is a session variable, and the session level takes effect.

- Type: boolean
- Description: **Only for the table of the AGG model**, when the variable is true, when the user query contains aggregate functions such as count(distinct c1), if the type of the c1 column itself is bitmap, count distnct will be rewritten It is bitmap_union_count(c1). When the type of the c1 column itself is hll, count distinct will be rewritten as hll_union_agg(c1) If the variable is false, no overwriting occurs..

Load And Export

`enable_pipeline_load`

Default value: false

IsMutable: true

Whether or not to enable the Pipeline engine to perform import tasks such as Streamload.

`enable_vectorized_load`

Default: true

Whether to enable vectorized load

`enable_new_load_scan_node`

Default: true

Whether to enable file scan node

`default_max_filter_ratio`

Default：0

IsMutable：true

MasterOnly：true

Maximum percentage of data that can be filtered (due to reasons such as data is irregularly) , The default value is 0.

`max_running_txn_num_per_db`

Default：1000

IsMutable：true

MasterOnly：true

This configuration is mainly used to control the number of concurrent load jobs of the same database.

When there are too many load jobs running in the cluster, the newly submitted load jobs may report errors:

```
current running txns on db xxx is xx, larger than limit xx
```

When this error is encountered, it means that the load jobs currently running in the cluster exceeds the configuration value. At this time, it is recommended to wait on the business side and retry the load jobs.

If you use the Connector, the value of this parameter can be adjusted appropriately, and there is no problem with thousands

`using_old_load_usage_pattern`

Default：false

IsMutable：true

MasterOnly：true

If set to true, the insert stmt with processing error will still return a label to user. And user can use this label to check the load job's status. The default value is false, which means if insert operation encounter errors, exception will be thrown to user client directly without load label.

`disable_load_job`

Default：false

IsMutable：true

MasterOnly：true

if this is set to true

- all pending load job will failed when call begin txn api
- all prepare load job will failed when call commit txn api
- all committed load job will waiting to be published

`commit_timeout_second`

Default：30

IsMutable：true

MasterOnly：true

Maximal waiting time for all data inserted before one transaction to be committed This is the timeout second for the command “commit”

`max_unfinished_load_job`

Default：1000

IsMutable：true

MasterOnly：true

Max number of load jobs, include PENDING, ETL, LOADING, QUORUM_FINISHED. If exceed this number, load job is not allowed to be submitted

`db_used_data_quota_update_interval_secs`

Default：300 (s)

IsMutable：true

MasterOnly：true

One master daemon thread will update database used data quota for db txn manager every `db_used_data_quota_update_interval_secs`

For better data load performance, in the check of whether the amount of data used by the database before data load exceeds the quota, we do not calculate the amount of data already used by the database in real time, but obtain the periodically updated value of the daemon thread.

This configuration is used to set the time interval for updating the value of the amount of data used by the database

`disable_show_stream_load`

Default：false

IsMutable：true

MasterOnly：true

Whether to disable show stream load and clear stream load records in memory.

`max_stream_load_record_size`

Default：5000

IsMutable：true

MasterOnly：true

Default max number of recent stream load record that can be stored in memory.

`fetch_stream_load_record_interval_second`

Default：120

IsMutable：true

MasterOnly：true

fetch stream load record interval.

`max_bytes_per_broker_scanner`

Default：`500 * 1024 * 1024 * 1024L` （500G）

IsMutable：true

MasterOnly：true

Max bytes a broker scanner can process in one broker load job. Commonly, each Backends has one broker scanner.

`default_load_parallelism`

Default: 1

IsMutable：true

MasterOnly：true

Default parallelism of the broker load execution plan on a single node. If the user to set the parallelism when the broker load is submitted, this parameter will be ignored. This parameter will determine the concurrency of import tasks together with multiple configurations such as `max broker concurrency`, `min bytes per broker scanner`.

`max_broker_concurrency`

Default：10

IsMutable：true

MasterOnly：true

Maximal concurrency of broker scanners.

`min_bytes_per_broker_scanner`

Default：67108864L (64M)

IsMutable：true

MasterOnly：true

Minimum bytes that a single broker scanner will read.

`period_of_auto_resume_min`

Default：5 （s）

IsMutable：true

MasterOnly：true

Automatically restore the cycle of Routine load

`max_tolerable_backend_down_num`

Default：0

IsMutable：true

MasterOnly：true

As long as one BE is down, Routine Load cannot be automatically restored

`max_routine_load_task_num_per_be`

Default：1024

IsMutable：true

MasterOnly：true

the max concurrent routine load task num per BE. This is to limit the num of routine load tasks sending to a BE, and it should also less than BE config ‘max_routine_load_thread_pool_size’(default 1024), which is the routine load task thread pool size on BE. In major versions of 2.0, the default value is 1024 starting with version 2.0.13. And before version 2.0.13 the default value was 5.

`max_routine_load_task_concurrent_num`

Default：256

IsMutable：true

MasterOnly：true

the max concurrent routine load task num of a single routine load job. In major versions of 2.0, the default value is 256 starting with version 2.0.13. And before version 2.0.13 the default value was 5.

`max_routine_load_job_num`

Default：100

the max routine load job num, including NEED_SCHEDULED, RUNNING, PAUSE

`desired_max_waiting_jobs`

Default：100

IsMutable：true

MasterOnly：true

Default number of waiting jobs for routine load and version 2 of load , This is a desired number. In some situation, such as switch the master, the current number is maybe more than desired_max_waiting_jobs.

`disable_hadoop_load`

Default：false

IsMutable：true

MasterOnly：true

Load using hadoop cluster will be deprecated in future. Set to true to disable this kind of load.

`enable_spark_load`

Default：false

IsMutable：true

MasterOnly：true

Whether to enable spark load temporarily, it is not enabled by default

Note: This parameter has been deleted in version 1.2, spark_load is enabled by default

`spark_load_checker_interval_second`

Default：60

Spark load scheduler run interval, default 60 seconds

`async_loading_load_task_pool_size`

Default：10

IsMutable：false

MasterOnly：true

The loading_load task executor pool size. This pool size limits the max running loading_load tasks.

Currently, it only limits the loading_load task of broker load

`async_pending_load_task_pool_size`

Default：10

IsMutable：false

MasterOnly：true

The pending_load task executor pool size. This pool size limits the max running pending_load tasks.

Currently, it only limits the pending_load task of broker load and spark load.

It should be less than ‘max_running_txn_num_per_db’

`async_load_task_pool_size`

Default：10

IsMutable：false

MasterOnly：true

This configuration is just for compatible with old version, this config has been replaced by async_loading_load_task_pool_size, it will be removed in the future.

`enable_single_replica_load`

Default：false

IsMutable：true

MasterOnly：true

Whether to enable to write single replica for stream load and broker load.

`min_load_timeout_second`

Default：1 （1s）

IsMutable：true

MasterOnly：true

Min stream load timeout applicable to all type of load

`max_stream_load_timeout_second`

Default: 259200 (3 day)

IsMutable：true

MasterOnly：true

This configuration is specifically used to limit timeout setting for stream load. It is to prevent that failed stream load transactions cannot be canceled within a short time because of the user’s large timeout setting

`max_load_timeout_second`

Default: 259200 (3 day)

IsMutable：true

MasterOnly：true

Max load timeout applicable to all type of load except for stream load

`stream_load_default_timeout_second`

Default: 86400 * 3 (3 day)

IsMutable：true

MasterOnly：true

Default stream load and streaming mini load timeout

`stream_load_default_precommit_timeout_second`

Default：3600（s）

IsMutable：true

MasterOnly：true

Default stream load pre-submission timeout

`insert_load_default_timeout_second`

Default: 3600 (1 hour)

IsMutable：true

MasterOnly：true

Default insert load timeout

`mini_load_default_timeout_second`

Default: 3600 (1 hour)

IsMutable：true

MasterOnly：true

Default non-streaming mini load timeout

`broker_load_default_timeout_second`

Default: 14400 (4 hour)

IsMutable：true

MasterOnly：true

Default broker load timeout

`spark_load_default_timeout_second`

Default: 86400 (1 day)

IsMutable：true

MasterOnly：true

Default spark load timeout

`hadoop_load_default_timeout_second`

Default: 86400 * 3 (3 day)

IsMutable：true

MasterOnly：true

Default hadoop load timeout

`load_running_job_num_limit`

Default：0

IsMutable：true

MasterOnly：true

The number of loading tasks is limited, the default is 0, no limit

`load_input_size_limit_gb`

Default: 0

IsMutable: true

MasterOnly: true

The size of the data entered by the Load job, the default is 0, unlimited

`load_etl_thread_num_normal_priority`

Default: 10

Concurrency of NORMAL priority etl load jobs. Do not change this if you know what you are doing.

`load_etl_thread_num_high_priority`

Default: 3

Concurrency of HIGH priority etl load jobs. Do not change this if you know what you are doing

`load_pending_thread_num_normal_priority`

Default: 10

Concurrency of NORMAL priority pending load jobs. Do not change this if you know what you are doing.

`load_pending_thread_num_high_priority`

Default: 3

Concurrency of HIGH priority pending load jobs. Load job priority is defined as HIGH or NORMAL. All mini batch load jobs are HIGH priority, other types of load jobs are NORMAL priority. Priority is set to avoid that a slow load job occupies a thread for a long time. This is just a internal optimized scheduling policy. Currently, you can not specified the job priority manually, and do not change this if you know what you are doing.

`load_checker_interval_second`

Default: 5 (s)

The load scheduler running interval. A load job will transfer its state from PENDING to LOADING to FINISHED. The load scheduler will transfer load job from PENDING to LOADING while the txn callback will transfer load job from LOADING to FINISHED. So a load job will cost at most one interval to finish when the concurrency has not reached the upper limit.

`label_keep_max_second`

Default：`3 * 24 * 3600` (3 day)

IsMutable：true

MasterOnly：true

labels of finished or cancelled load jobs will be removed after `label_keep_max_second`，

1. The removed labels can be reused.
2. Set a short time will lower the FE memory usage. (Because all load jobs' info is kept in memory before being removed)

In the case of high concurrent writes, if there is a large backlog of jobs and call frontend service failed, check the log. If the metadata write takes too long to lock, you can adjust this value to 12 hours, or 6 hours less

`streaming_label_keep_max_second`

Default: 43200 (12 hour)

IsMutable：true

MasterOnly：true

For some high-frequency load work, such as: INSERT, STREAMING LOAD, ROUTINE_LOAD_TASK. If it expires, delete the completed job or task.

`label_clean_interval_second`

Default：1 * 3600 (1 hour)

Load label cleaner will run every *label_clean_interval_second* to clean the outdated jobs.

`label_regex_length`

Default Value: 128 (characters)

The maximum character length of the load label, the default is 128 characters.

`transaction_clean_interval_second`

Default：30

the transaction will be cleaned after transaction_clean_interval_second seconds if the transaction is visible or aborted we should make this interval as short as possible and each clean cycle as soon as possible

`sync_commit_interval_second`

The maximum time interval for committing transactions. If there is still data in the channel that has not been submitted after this time, the consumer will notify the channel to submit the transaction.

Default: 10 (seconds)

Is it possible to configure dynamically: true

Whether it is a configuration item unique to the Master FE node: true

`sync_checker_interval_second`

Data synchronization job running status check.

Default: 10 (s)

`max_sync_task_threads_num`

The maximum number of threads in the data synchronization job thread pool.

默认值：10

`min_sync_commit_size`

The minimum number of events that must be satisfied to commit a transaction. If the number of events received by Fe is less than it, it will continue to wait for the next batch of data until the time exceeds `sync_commit_interval_second`. The default value is 10000 events. If you want to modify this configuration, please make sure that this value is smaller than the `canal.instance.memory.buffer.size` configuration on the canal side (default 16384), otherwise Fe will try to get the queue length longer than the store before ack More events cause the store queue to block until it times out.

Default: 10000

Is it possible to configure dynamically: true

Whether it is a configuration item unique to the Master FE node: true

`min_bytes_sync_commit`

The minimum data size required to commit a transaction. If the data size received by Fe is smaller than it, it will continue to wait for the next batch of data until the time exceeds `sync_commit_interval_second`. The default value is 15MB, if you want to modify this configuration, please make sure this value is less than the product of `canal.instance.memory.buffer.size` and `canal.instance.memory.buffer.memunit` on the canal side (default 16MB), otherwise Before the ack, Fe will try to obtain data that is larger than the store space, causing the store queue to block until it times out.

Default: `15*1024*1024` (15M)

Is it possible to configure dynamically: true

Whether it is a configuration item unique to the Master FE node: true

`max_bytes_sync_commit`

The maximum number of threads in the data synchronization job thread pool. There is only one thread pool in the entire FE, which is used to process all data synchronization tasks in the FE that send data to the BE. The implementation of the thread pool is in the `SyncTaskPool` class.

Default: 10

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: false

`enable_outfile_to_local`

Default：false

Whether to allow the outfile function to export the results to the local disk.

`export_tablet_num_per_task`

Default：5

IsMutable：true

MasterOnly：true

Number of tablets per export query plan

`export_task_default_timeout_second`

Default: 2 * 3600 (2 hour)

IsMutable：true

MasterOnly：true

Default timeout of export jobs.

`export_running_job_num_limit`

Default：5

IsMutable：true

MasterOnly：true

Limitation of the concurrency of running export jobs. Default is 5. 0 is unlimited

`export_checker_interval_second`

Default：5

Export checker’s running interval.

Log

`log_roll_size_mb`

Default：1024 （1G）

The max size of one sys log and audit log

`sys_log_dir`

Default: DorisFE.DORIS_HOME_DIR + “/log”

This specifies FE log dir. FE will produces 2 log files:

fe.log: all logs of FE process. fe.warn.log all WARNING and ERROR log of FE process.

`sys_log_level`

Default：INFO

log level: INFO, WARNING, ERROR, FATAL

`sys_log_roll_num`

Default：10

Maximal FE log files to be kept within an sys_log_roll_interval. default is 10, which means there will be at most 10 log files in a day

`sys_log_verbose_modules`

Default：{}

Verbose modules. VERBOSE level is implemented by log4j DEBUG level.

eg： sys_log_verbose_modules = org.apache.doris.catalog This will only print debug log of files in package org.apache.doris.catalog and all its sub packages.

`sys_log_roll_interval`

Default：DAY

sys_log_roll_interval:

- DAY: log suffix is yyyyMMdd
- HOUR: log suffix is yyyyMMddHH

`sys_log_delete_age`

Default：7d

default is 7 days, if log's last modify time is 7 days ago, it will be deleted.

support format:

- 7d 7 day
- 10h 10 hours
- 60m 60 min
- 120s 120 seconds

`sys_log_roll_mode`

Default：SIZE-MB-1024

The size of the log split, split a log file every 1 G

`sys_log_enable_compress`

Default: false

If true, will compress fe.log & fe.warn.log by gzip

`audit_log_dir`

Default：DORIS_HOME_DIR + "/log"

audit_log_dir： This specifies FE audit log dir.. Audit log fe.audit.log contains all requests with related infos such as user, host, cost, status, etc

`audit_log_roll_num`

Default：90

Maximal FE audit log files to be kept within an audit_log_roll_interval.

`audit_log_modules`

Default：{"slow_query", "query", "load", "stream_load"}

Slow query contains all queries which cost exceed *qe_slow_log_ms*

`qe_slow_log_ms`

Default: 5000 (5 seconds)

If the response time of a query exceed this threshold, it will be recorded in audit log as slow_query.

`audit_log_roll_interval`

Default：DAY

DAY: logsuffix is : yyyyMMdd HOUR: logsuffix is : yyyyMMddHH

`audit_log_delete_age`

Default：30d

default is 30 days, if log’s last modify time is 30 days ago, it will be deleted.

support format: * 7d 7 day * 10h 10 hours * 60m 60 min * 120s 120 seconds

`audit_log_enable_compress`

Default: false

If true, will compress fe.audit.log by gzip

Storage

`min_replication_num_per_tablet`

Default: 1

Used to set minimal number of replication per tablet.

`max_replication_num_per_tablet`

Default: 32767

Used to set maximal number of replication per tablet.

`default_db_data_quota_bytes`

Default：1PB

IsMutable：true

MasterOnly：true

Used to set the default database data quota size. To set the quota size of a single database, you can use:

```
Set the database data quota, the unit is:B/K/KB/M/MB/G/GB/T/TB/P/PB
ALTER DATABASE db_name SET DATA QUOTA quota;
View configuration
show data  (Detail: HELP SHOW DATA)
```

`default_db_replica_quota_size`

Default: 1073741824

IsMutable：true

MasterOnly：true

Used to set the default database replica quota. To set the quota size of a single database, you can use:

```
Set the database replica quota
ALTER DATABASE db_name SET REPLICA QUOTA quota;
View configuration
show data  (Detail: HELP SHOW DATA)
```

`recover_with_empty_tablet`

Default：false

IsMutable：true

MasterOnly：true

In some very special circumstances, such as code bugs, or human misoperation, etc., all replicas of some tablets may be lost. In this case, the data has been substantially lost. However, in some scenarios, the business still hopes to ensure that the query will not report errors even if there is data loss, and reduce the perception of the user layer. At this point, we can use the blank Tablet to fill the missing replica to ensure that the query can be executed normally.

Set to true so that Doris will automatically use blank replicas to fill tablets which all replicas have been damaged or missing

`min_clone_task_timeout_sec And max_clone_task_timeout_sec`

Default: Minimum 3 minutes, maximum two hours

IsMutable：true

MasterOnly：true

Can cooperate with `mix_clone_task_timeout_sec` to control the maximum and minimum timeout of a clone task. Under normal circumstances, the timeout of a clone task is estimated by the amount of data and the minimum transfer rate (5MB/s). In some special cases, these two configurations can be used to set the upper and lower bounds of the clone task timeout to ensure that the clone task can be completed successfully.

`disable_storage_medium_check`

Default：false

IsMutable：true

MasterOnly：true

If disable_storage_medium_check is true, ReportHandler would not check tablet’s storage medium and disable storage cool down function, the default value is false. You can set the value true when you don’t care what the storage medium of the tablet is.

`decommission_tablet_check_threshold`

Default：5000

IsMutable：true

MasterOnly：true

This configuration is used to control whether the Master FE need to check the status of tablets on decommissioned BE. If the size of tablets on decommissioned BE is lower than this threshold, FE will start a periodic check, if all tablets on decommissioned BE have been recycled, FE will drop this BE immediately.

For performance consideration, please don’t set a very high value for this configuration.

`partition_rebalance_max_moves_num_per_selection`

Default：10

IsMutable：true

MasterOnly：true

Valid only if use PartitionRebalancer,

`partition_rebalance_move_expire_after_access`

Default：600 (s)

IsMutable：true

MasterOnly：true

Valid only if use PartitionRebalancer. If this changed, cached moves will be cleared

`tablet_rebalancer_type`

Default：BeLoad

MasterOnly：true

Rebalancer type(ignore case): BeLoad, Partition. If type parse failed, use BeLoad as default

`max_balancing_tablets`

Default：100

IsMutable：true

MasterOnly：true

if the number of balancing tablets in TabletScheduler exceed max_balancing_tablets, no more balance check

`max_scheduling_tablets`

Default：2000

IsMutable：true

MasterOnly：true

if the number of scheduled tablets in TabletScheduler exceed max_scheduling_tablets skip checking.

`disable_balance`

Default：false

IsMutable：true

MasterOnly：true

if set to true, TabletScheduler will not do balance.

`disable_disk_balance`

Default：true

IsMutable：true

MasterOnly：true

if set to true, TabletScheduler will not do disk balance.

`balance_load_score_threshold`

Default: 0.1 (10%)

IsMutable：true

MasterOnly：true

the threshold of cluster balance score, if a backend's load score is 10% lower than average score, this backend will be marked as LOW load, if load score is 10% higher than average score, HIGH load will be marked

`capacity_used_percent_high_water`

Default: 0.75 (75%)

IsMutable：true

MasterOnly：true

The high water of disk capacity used percent. This is used for calculating load score of a backend

`clone_distribution_balance_threshold`

Default: 0.2

IsMutable：true

MasterOnly：true

Balance threshold of num of replicas in Backends.

`clone_capacity_balance_threshold`

Default: 0.2

IsMutable：true

MasterOnly：true

- Balance threshold of data size in BE.

  The balance algorithm is:

  1. Calculate the average used capacity(AUC) of the entire cluster. (total data size / total backends num)
  2. The high water level is (AUC * (1 + clone_capacity_balance_threshold))
  3. The low water level is (AUC * (1 - clone_capacity_balance_threshold))
  4. The Clone checker will try to move replica from high water level BE to low water level BE.

`disable_colocate_balance`

Default：false

IsMutable：true

MasterOnly：true

This configs can set to true to disable the automatic colocate tables's relocate and balance. If 'disable_colocate_balance' is set to true, ColocateTableBalancer will not relocate and balance colocate tables.

Attention:

1. Under normal circumstances, there is no need to turn off balance at all.
2. Because once the balance is turned off, the unstable colocate table may not be restored
3. Eventually the colocate plan cannot be used when querying.

`balance_slot_num_per_path`

Default: 1

IsMutable：true

MasterOnly：true

Default number of slots per path during balance.

`disable_tablet_scheduler`

Default:false

IsMutable：true

MasterOnly：true

If set to true, the tablet scheduler will not work, so that all tablet repair/balance task will not work.

`enable_force_drop_redundant_replica`

Default: false

Dynamically configured: true

Only for Master FE: true

If set to true, the system will immediately drop redundant replicas in the tablet scheduling logic. This may cause some load jobs that are writing to the corresponding replica to fail, but it will speed up the balance and repair speed of the tablet. When there are a large number of replicas waiting to be balanced or repaired in the cluster, you can try to set this config to speed up the balance and repair of replicas at the expense of partial load success rate.

`colocate_group_relocate_delay_second`

Default: 1800

Dynamically configured: true

Only for Master FE: true

The relocation of a colocation group may involve a large number of tablets moving within the cluster. Therefore, we should use a more conservative strategy to avoid relocation of colocation groups as much as possible. Reloaction usually occurs after a BE node goes offline or goes down. This parameter is used to delay the determination of BE node unavailability. The default is 30 minutes, i.e., if a BE node recovers within 30 minutes, relocation of the colocation group will not be triggered.

`allow_replica_on_same_host`

Default: false

Dynamically configured: false

Only for Master FE: false

Whether to allow multiple replicas of the same tablet to be distributed on the same host. This parameter is mainly used for local testing, to facilitate building multiple BEs to test certain multi-replica situations. Do not use it for non-test environments.

`repair_slow_replica`

Default: false

IsMutable：true

MasterOnly: true

If set to true, the replica with slower compaction will be automatically detected and migrated to other machines. The detection condition is that the version count of the fastest replica exceeds the value of `min_version_count_indicate_replica_compaction_too_slow`, and the ratio of the version count difference from the fastest replica exceeds the value of `valid_version_count_delta_ratio_between_replicas`

`min_version_count_indicate_replica_compaction_too_slow`

Default: 200

Dynamically configured: true

Only for Master FE: false

The version count threshold used to judge whether replica compaction is too slow

`skip_compaction_slower_replica`

Default: true

Dynamically configured: true

Only for Master FE: false

If set to true, the compaction slower replica will be skipped when select get queryable replicas

`valid_version_count_delta_ratio_between_replicas`

Default: 0.5

Dynamically configured: true

Only for Master FE: true

The valid ratio threshold of the difference between the version count of the slowest replica and the fastest replica. If `repair_slow_replica` is set to true, it is used to determine whether to repair the slowest replica

`min_bytes_indicate_replica_too_large`

Default: `2 * 1024 * 1024 * 1024` (2G)

Dynamically configured: true

Only for Master FE: true

The data size threshold used to judge whether replica is too large

`schedule_slot_num_per_path`

Default：2

the default slot number per path in tablet scheduler , remove this config and dynamically adjust it by clone task statistic

`tablet_repair_delay_factor_second`

Default：60 （s）

IsMutable：true

MasterOnly：true

the factor of delay time before deciding to repair tablet.

- if priority is VERY_HIGH, repair it immediately.
- HIGH, delay tablet_repair_delay_factor_second * 1;
- NORMAL: delay tablet_repair_delay_factor_second * 2;
- LOW: delay tablet_repair_delay_factor_second * 3;

`tablet_stat_update_interval_second`

Default：300（5min）

update interval of tablet stat, All frontends will get tablet stat from all backends at each interval

`storage_flood_stage_usage_percent`

Default：95 （95%）

IsMutable：true

MasterOnly：true

`storage_flood_stage_left_capacity_bytes`

Default：1 *1024* 1024 * 1024 (1GB)

IsMutable：true

MasterOnly：true

If capacity of disk reach the ‘storage_flood_stage_usage_percent’ and ‘storage_flood_stage_left_capacity_bytes’, the following operation will be rejected:

1. load job
2. restore job

`storage_high_watermark_usage_percent`

Default：85 (85%)

IsMutable：true

MasterOnly：true

`storage_min_left_capacity_bytes`

Default： `2 * 1024 * 1024 * 1024` (2GB)

IsMutable：true

MasterOnly：true

‘storage_high_watermark_usage_percent’ limit the max capacity usage percent of a Backend storage path. ‘storage_min_left_capacity_bytes’ limit the minimum left capacity of a Backend storage path. If both limitations are reached, this storage path can not be chose as tablet balance destination. But for tablet recovery, we may exceed these limit for keeping data integrity as much as possible.

`catalog_trash_expire_second`

Default: 86400L (1 day)

IsMutable：true

MasterOnly：true

After dropping database(table/partition), you can recover it by using RECOVER stmt. And this specifies the maximal data retention time. After time, the data will be deleted permanently.

`storage_cooldown_second`

> **Tips**
>
> This feature is deprecated since the Apache Doris 2.0 version

Default：`30 * 24 * 3600L` （30 day）

When create a table(or partition), you can specify its storage medium(HDD or SSD). If set to SSD, this specifies the default duration that tablets will stay on SSD. After that, tablets will be moved to HDD automatically. You can set storage cooldown time in CREATE TABLE stmt.

`default_storage_medium`

Default：HDD

When create a table(or partition), you can specify its storage medium(HDD or SSD). If not set, this specifies the default medium when creat.

`enable_storage_policy`

- Whether to enable the Storage Policy feature. This config allows users to separate hot and cold data. Default: false

Is it possible to dynamically configure: true

Is it a configuration item unique to the Master FE node: true

`check_consistency_default_timeout_second`

Default: 600 (10 minutes)

IsMutable：true

MasterOnly：true

Default timeout of a single consistency check task. Set long enough to fit your tablet size

`consistency_check_start_time`

Default：23

IsMutable：true

MasterOnly：true

Consistency check start time

Consistency checker will run from *consistency_check_start_time* to *consistency_check_end_time*.

If the two times are the same, no consistency check will be triggered.

`consistency_check_end_time`

Default：23

IsMutable：true

MasterOnly：true

Consistency check end time

Consistency checker will run from *consistency_check_start_time* to *consistency_check_end_time*.

If the two times are the same, no consistency check will be triggered.

`replica_delay_recovery_second`

Default：0

IsMutable：true

MasterOnly：true

the minimal delay seconds between a replica is failed and fe try to recovery it using clone.

`tablet_create_timeout_second`

Default：1（s）

IsMutable：true

MasterOnly：true

Maximal waiting time for creating a single replica.

eg. if you create a table with #m tablets and #n replicas for each tablet, the create table request will run at most (m *n* tablet_create_timeout_second) before timeout.

`tablet_delete_timeout_second`

Default：2

IsMutable：true

MasterOnly：true

Same meaning as *tablet_create_timeout_second*, but used when delete a tablet.

`alter_table_timeout_second`

Default: 86400 * 30 (1 month)

IsMutable：true

MasterOnly：true

Maximal timeout of ALTER TABLE request. Set long enough to fit your table data size.

`max_replica_count_when_schema_change`

The maximum number of replicas allowed when OlapTable is doing schema changes. Too many replicas will lead to FE OOM.

Default: 100000

Is it possible to configure dynamically: true

Whether it is a configuration item unique to the Master FE node: true

`history_job_keep_max_second`

Default：`7 * 24 * 3600` （7 day）

IsMutable：true

MasterOnly：true

The max keep time of some kind of jobs. like schema change job and rollup job.

`max_create_table_timeout_second`

Default：60 （s）

IsMutable：true

MasterOnly：true

In order not to wait too long for create table(index), set a max timeout.

External Table

`file_scan_node_split_num`

Default：128

IsMutable：true

MasterOnly：false

multi catalog concurrent file scanning threads

`file_scan_node_split_size`

Default：256 *1024* 1024

IsMutable：true

MasterOnly：false

multi catalog concurrent file scan size

`enable_odbc_table`

Default：false

IsMutable：true

MasterOnly：true

Whether to enable the ODBC table, it is not enabled by default. You need to manually configure it when you use it.

This parameter can be set by: ADMIN SET FRONTEND CONFIG(“key”=“value”)

Note: This parameter has been deleted in version 1.2. The ODBC External Table is enabled by default, and the ODBC External Table will be deleted in a later version. It is recommended to use the JDBC External Table

`disable_iceberg_hudi_table`

Default：true

IsMutable：true

MasterOnly：false

Starting from version 1.2, we no longer support create hudi and iceberg External Table. Please use the multi catalog.

`iceberg_table_creation_interval_second`

Default：10 (s)

IsMutable：true

MasterOnly：false

fe will create iceberg table every iceberg_table_creation_interval_second

`iceberg_table_creation_strict_mode`

Default：true

IsMutable：true

MasterOnly：true

If set to TRUE, the column definitions of iceberg table and the doris table must be consistent If set to FALSE, Doris only creates columns of supported data types.

`max_iceberg_table_creation_record_size`

Default max number of recent iceberg database table creation record that can be stored in memory.

Default：2000

IsMutable：true

MasterOnly：true

`max_hive_partition_cache_num`

The maximum number of caches for the hive partition.

Default: 100000

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: false

`hive_metastore_client_timeout_second`

The default connection timeout for hive metastore.

Default: 10

Is it possible to dynamically configure: true

Is it a configuration item unique to the Master FE node: true

`max_external_cache_loader_thread_pool_size`

Maximum thread pool size for loading external meta cache.

Default: 10

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: false

`max_external_file_cache_num`

Maximum number of file cache to use for external tables.

Default: 100000

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: false

`max_external_schema_cache_num`

Maximum number of schema cache to use for external external tables.

Default: 10000

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: false

`external_cache_expire_time_minutes_after_access`

Set how long the data in the cache expires after the last access. The unit is minutes. Applies to External Schema Cache as well as Hive Partition Cache.

Default: 1440

Is it possible to dynamically configure: false

Is it a configuration item unique to the Master FE node: false

`es_state_sync_interval_second`

Default：10

fe will call es api to get es index shard info every es_state_sync_interval_secs

External Resources

`dpp_hadoop_client_path`

Default：/lib/hadoop-client/hadoop/bin/hadoop

`dpp_bytes_per_reduce`

Default：`100 * 1024 * 1024L` (100M)

dpp_default_cluster

Default：palo-dpp

dpp_default_config_str

Default：{ hadoop_configs : ‘mapred.job.priority=NORMAL;mapred.job.map.capacity=50;mapred.job.reduce.capacity=50;mapred.hce.replace.streaming=false;abaci.long.stored.job=true;dce.shuffle.enable=false;dfs.client.authserver.force_stop=true;dfs.client.auth.method=0’ }

dpp_config_str

Default：{ palo-dpp : { hadoop_palo_path : ‘/dir’, hadoop_configs : ‘fs.default.name=hdfs://host:port;mapred.job.tracker=host:port;hadoop.job.ugi=user,password’ } }

yarn_config_dir

Default: DorisFE.DORIS_HOME_DIR + “/lib/yarn-config”

Default yarn config file directory , Each time before running the yarn command, we need to check that the config file exists under this path, and if not, create them.

yarn_client_path

Default：DORIS_HOME_DIR + “/lib/yarn-client/hadoop/bin/yarn”

Default yarn client path

spark_launcher_log_dir

Default： sys_log_dir + “/spark_launcher_log”

The specified spark launcher log dir

spark_resource_path

Default：none

Default spark dependencies path

spark_home_default_dir

Default：DORIS_HOME_DIR + “/lib/spark2x”

Default spark home dir

`spark_dpp_version`

Default: 1.0.0

Default spark dpp version

Else

`tmp_dir`

Default: DorisFE.DORIS_HOME_DIR + “/temp_dir”

temp dir is used to save intermediate results of some process, such as backup and restore process. file in this dir will be cleaned after these process is finished.

`custom_config_dir`

Default: DorisFE.DORIS_HOME_DIR + “/conf”

Custom configuration file directory

Configure the location of the `fe_custom.conf` file. The default is in the `conf/` directory.

In some deployment environments, the `conf/` directory may be overwritten due to system upgrades. This will cause the user modified configuration items to be overwritten. At this time, we can store `fe_custom.conf` in another specified directory to prevent the configuration file from being overwritten.

`plugin_dir`

Default：DORIS_HOME + “/plugins

plugin install directory

`plugin_enable`

Default:true

IsMutable：true

MasterOnly：true

Whether the plug-in is enabled, enabled by default

`small_file_dir`

Default：DORIS_HOME_DIR/small_files

Save small files

`max_small_file_size_bytes`

Default：1M

IsMutable：true

MasterOnly：true

The max size of a single file store in SmallFileMgr

`max_small_file_number`

Default：100

IsMutable：true

MasterOnly：true

The max number of files store in SmallFileMgr

`enable_metric_calculator`

Default：true

If set to true, metric collector will be run as a daemon timer to collect metrics at fix interval

`report_queue_size`

Default： 100

IsMutable：true

MasterOnly：true

This threshold is to avoid piling up too many report task in FE, which may cause OOM exception. In some large Doris cluster, eg: 100 Backends with ten million replicas, a tablet report may cost several seconds after some modification of metadata(drop partition, etc..). And one Backend will report tablets info every 1 min, so unlimited receiving reports is unacceptable. we will optimize the processing speed of tablet report in future, but now, just discard the report if queue size exceeding limit. Some online time cost: 1. disk report: 0-1 msta 2. sk report: 0-1 ms 3. tablet report 4. 10000 replicas: 200ms

`backup_job_default_timeout_ms`

Default: 86400 * 1000 (1 day)

IsMutable：true

MasterOnly：true

default timeout of backup job

`max_backup_restore_job_num_per_db`

Default: 10

This configuration is mainly used to control the number of backup/restore tasks recorded in each database.

`max_backup_tablets_per_job`

Default: 300000

IsMutable：true

MasterOnly：true

Control the max num of tablets per backup job involved, to avoid FE OOM caused by saving too much metadata.

> **TIPS**
>
> This configuration is supported since the Apache Doris 2.0.15 version

`enable_quantile_state_type`

Default：false

IsMutable：true

MasterOnly：true

Whether to enable the quantile_state data type

`enable_date_conversion`

Default：true

IsMutable：true

MasterOnly：true

FE will convert date/datetime to datev2/datetimev2(0) automatically.

`enable_decimal_conversion`

Default：true

IsMutable：true

MasterOnly：true

FE will convert DecimalV2 to DecimalV3 automatically.

`proxy_auth_magic_prefix`

Default: x@8

`proxy_auth_enable`

Default: false

`enable_func_pushdown`

Default: true

IsMutable: true

MasterOnly: false

Whether to push the filter conditions with functions down to MYSQL, when exectue query of ODBC, JDBC external tables

`jdbc_drivers_dir`

Default: `${DORIS_HOME}/jdbc_drivers`;

IsMutable: false

MasterOnly: false

The default dir to put jdbc drivers.

`max_error_tablet_of_broker_load`

Default: 3;

IsMutable: true

MasterOnly: true

Maximum number of error tablet showed in broker load.

`default_db_max_running_txn_num`

Default: -1

IsMutable: true

MasterOnly: true

Used to set the default database transaction quota size.

The default value setting to -1 means using `max_running_txn_num_per_db` instead of `default_db_max_running_txn_num`.

To set the quota size of a single database, you can use:

```
Set the database transaction quota
ALTER DATABASE db_name SET TRANSACTION QUOTA quota;
View configuration
show data  (Detail: HELP SHOW DATA)
```

`prefer_compute_node_for_external_table`

Default：false

IsMutable：true

MasterOnly：false

If set to true, query on external table will prefer to assign to compute node. And the max number of compute node is controlled by `min_backend_num_for_external_table`. If set to false, query on external table will assign to any node.

`min_backend_num_for_external_table`

Default：3

IsMutable：true

MasterOnly：false

Only take effect when `prefer_compute_node_for_external_table` is true. If the compute node number is less than this value, query on external table will try to get some mix node to assign, to let the total number of node reach this value. If the compute node number is larger than this value, query on external table will assign to compute node only.

`infodb_support_ext_catalog`

> **Tips**
>
> This configuration is supported since the Apache Doris 2.1.4 version

Default: false

IsMutable: true

MasterOnly: false

If false, when select from tables in information_schema database, the result will not contain the information of the table in external catalog. This is to avoid query time when external catalog is not reachable.

`enable_query_hit_stats`

Default: false

IsMutable: true

MasterOnly: false

Controls whether to enable query hit statistics. The default is false.

`div_precision_increment`

Default: 4

This variable indicates the number of digits by which to increase the scale of the result of division operations performed with the / operator.

`enable_convert_light_weight_schema_change`

Default：true

Temporary configuration option. After it is enabled, a background thread will be started to automatically modify all olap tables to light schema change. The modification results can be viewed through the command `show convert_light_schema_change [from db]`, and the conversion results of all non-light schema change tables will be displayed.

`disable_local_deploy_manager_drop_node`

Default：true

Forbid LocalDeployManager drop nodes to prevent errors in the cluster.info file from causing nodes to be dropped.

`mysqldb_replace_name`

Default: mysql

To ensure compatibility with the MySQL ecosystem, Doris includes a built-in database called mysql. If this database conflicts with a user's own database, please modify this field to replace the name of the Doris built-in MySQL database with a different name.

`max_auto_partition_num`

Default value: 2000

For auto-partitioned tables to prevent users from accidentally creating a large number of partitions, the number of partitions allowed per OLAP table is `max_auto_partition_num`. Default 2000.

BE Configuration

BE Configuration

This document mainly introduces the relevant configuration items of BE.

The BE configuration file `be.conf` is usually stored in the `conf/` directory of the BE deployment path. In version 0.14, another configuration file `be_custom.conf` will be introduced. The configuration file is used to record the configuration items that are dynamically configured and persisted by the user during operation.

After the BE process is started, it will read the configuration items in `be.conf` first, and then read the configuration items in `be_custom.conf`. The configuration items in `be_custom.conf` will overwrite the same configuration items in `be.conf`.

The location of the `be_custom.conf` file can be configured in `be.conf` through the `custom_config_dir` configuration item.

View configuration items

Users can view the current configuration items by visiting BE's web page:

`http://be_host:be_webserver_port/varz`

Set configuration items

There are two ways to configure BE configuration items:

1. Static configuration

   Add and set configuration items in the `conf/be.conf` file. The configuration items in `be.conf` will be read when BE starts. Configuration items not in `be.conf` will use default values.

2. Dynamic configuration

   After BE starts, the configuration items can be dynamically set with the following commands.

   ```
   curl -X POST http://{be_ip}:{be_http_port}/api/update_config?{key}={value}
   ```

   In version 0.13 and before, the configuration items modified in this way will become invalid after the BE process restarts. In 0.14 and later versions, the modified configuration can be persisted through the following command. The modified configuration items are stored in the `be_custom.conf` file.

   ```
   curl -X POST http://{be_ip}:{be_http_port}/api/update_config?{key}={value}\&persist=true
   ```

Examples

1. Modify `max_base_compaction_threads` statically

   By adding in the `be.conf` file:

   ```
   max_base_compaction_threads=5
   ```

   Then restart the BE process to take effect the configuration.

2. Modify `streaming_load_max_mb` dynamically

   After BE starts, the configuration item `streaming_load_max_mb` is dynamically set by the following command:

   ```
   curl -X POST http://{be_ip}:{be_http_port}/api/update_config?streaming_load_max_mb=1024
   ```

   The return value is as follows, indicating that the setting is successful.

   ```
   {
       "status": "OK",
       "msg": ""
   }
   ```

   The configuration will become invalid after the BE restarts. If you want to persist the modified results, use the following command:

   ```
   curl -X POST http://{be_ip}:{be_http_port}/api/update_config?streaming_load_max_mb=1024\&persist=true
   ```

Configurations

Services

`be_port`

- Type: int32
- Description: The port of the thrift server on BE which used to receive requests from FE
- Default value: 9060

`heartbeat_service_port`

- Type: int32
- Description: Heartbeat service port (thrift) on BE, used to receive heartbeat from FE
- Default value: 9050

`webserver_port`

- Type: int32

- Description: Service port of http server on BE
- Default value: 8040

`brpc_port`

- Type: int32
- Description: The port of BRPC on BE, used for communication between BEs
- Default value: 8060

`arrow_flight_sql_port`

- Type: int32
- Description: The port of Arrow Flight SQL server on BE, used for communication between Arrow Flight Client and BE
- Default value: -1

`enable_https`

- Type: bool
- Description: Whether https is supported. If so, configure `ssl_certificate_path` and `ssl_private_key_path` in be.conf.
- Default value: false

`priority_networks`

- Description: Declare a selection strategy for those servers with many IPs. Note that at most one ip should match this list. This is a semicolon-separated list in CIDR notation, such as 10.10.10.0/24. If there is no IP matching this rule, one will be randomly selected
- Default value: blank

`storage_root_path`

- Type: string
- Description: data root path, separate by ';'.you can specify the storage medium of each root path, HDD or SSD. you can add capacity limit at the end of each root path, separate by ','.If the user does not use a mix of SSD and HDD disks, they do not need to configure the configuration methods in Example 1 and Example 2 below, but only need to specify the storage directory; they also do not need to modify the default storage media configuration of FE.

  eg.1: `storage_root_path=/home/disk1/doris.HDD;/home/disk2/doris.SSD;/home/disk2/doris`

  - 1./home/disk1/doris.HDD, indicates that the storage medium is HDD;
  - 2./home/disk2/doris.SSD, indicates that the storage medium is SSD;
  - 3./home/disk2/doris, indicates that the storage medium is HDD by default

  eg.2: `storage_root_path=/home/disk1/doris,medium:hdd;/home/disk2/doris,medium:ssd`

  - 1./home/disk1/doris,medium:hdd, indicates that the storage medium is HDD;

    - 2./home/disk2/doris,medium:ssd, indicates that the storage medium is SSD;

- Default value: ${DORIS_HOME}/storage

`heartbeat_service_thread_count`

- Type: int32
- Description: The number of threads that execute the heartbeat service on BE. the default is 1, it is not recommended to modify
- Default value: 1

`ignore_broken_disk`

- Type: bool
- Description: When BE starts, check `storage_root_path` All paths under configuration.

    - `ignore_broken_disk=true`

    If the path does not exist or the file (bad disk) cannot be read or written under the path, the path will be ignored. If there are other available paths, the startup will not be interrupted.

    - `ignore_broken_disk=false`

    If the path does not exist or the file (bad disk) cannot be read or written under the path, the system will abort the startup failure and exit.

- Default value: false

`mem_limit`

- Type: string
- Description: Limit the percentage of the server's maximum memory used by the BE process. It is used to prevent BE memory from occupying too the machine's memory. This parameter must be greater than 0. When the percentage is greater than 100%, the value will default to 100%.
- Default value: 80%

`cluster_id`

- Type: int32
- Description: Configure the cluster id to which the BE belongs.
    - This value is usually delivered by the FE to the BE by the heartbeat, no need to configure. When it is confirmed that a BE belongs to a certain Drois cluster, it can be configured. The cluster_id file under the data directory needs to be modified to make sure same as this parament.
- Default value: -1

`custom_config_dir`

- Description: Configure the location of the `be_custom.conf` file. The default is in the `conf/` directory.

    - In some deployment environments, the `conf/` directory may be overwritten due to system upgrades. This will cause the user modified configuration items to be overwritten. At this time, we can store `be_custom.conf` in another specified directory to prevent the configuration file from being overwritten.
- Default value: blank

`trash_file_expire_time_sec`

- Description: The interval for cleaning the recycle bin is 72 hours. When the disk space is insufficient, the file retention period under trash may not comply with this parameter
- Default value: 259200

`es_http_timeout_ms`

- Description: The timeout period for connecting to ES via http.
- Default value: 5000 (ms)

`es_scroll_keepalive`

- Description: es scroll Keeplive hold time
- Default value: 5 (m)

`external_table_connect_timeout_sec`

- Type: int32
- Description: The timeout when establishing connection with external table such as ODBC table.
- Default value: 5 seconds

`status_report_interval`

- Description: Interval between profile reports
- Default value: 5 seconds

`brpc_max_body_size`

- Description: This configuration is mainly used to modify the parameter `max_body_size` of brpc.
    - Sometimes the query fails and an error message of `body_size is too large` will appear in the BE log. This may happen when the SQL mode is “multi distinct + no group by + more than 1T of data”.This error indicates that the packet size of brpc exceeds the configured value. At this time, you can avoid this error by increasing the configuration.

`brpc_socket_max_unwritten_bytes`

- Description: This configuration is mainly used to modify the parameter `socket_max_unwritten_bytes` of brpc.
    - Sometimes the query fails and an error message of `The server is overcrowded` will appear in the BE log. This means there are too many messages to buffer at the sender side, which may happen

when the SQL needs to send large bitmap value. You can avoid this error by increasing the configuration.

`transfer_large_data_by_brpc`

- Type: bool
- Description: This configuration is used to control whether to serialize the protoBuf request and embed the Tuple/Block data into the controller attachment and send it through http brpc when the length of the Tuple/Block data is greater than 1.8G. To avoid errors when the length of the protoBuf request exceeds 2G: Bad request, error_text=[E1003]Fail to compress request. In the past version, after putting Tuple/Block data in the attachment, it was sent through the default baidu_std brpc, but when the attachment exceeds 2G, it will be truncated. There is no 2G limit for sending through http brpc.
- Default value: true

`brpc_num_threads`

- Description: This configuration is mainly used to modify the number of bthreads for brpc. The default value is set to 256.
    - User can set this configuration to a larger value to get better QPS performance. For more information, please refer to `https://github.com/apache/brpc/blob/master/docs/cn/benchmark.md`
- Default value: 256

`thrift_rpc_timeout_ms`

- Description: thrift default timeout time
- Default value: 60000

`thrift_client_retry_interval_ms`

- Type: int64
- Description: Used to set retry interval for thrift client in be to avoid avalanche disaster in fe thrift server, the unit is ms.
- Default value: 1000

`thrift_connect_timeout_seconds`

- Description: The default thrift client connection timeout time
- Default value: 3 (m)

`thrift_server_type_of_fe`

- Type: string
- Description:This configuration indicates the service model used by FE's Thrift service. The type is string and is case-insensitive. This parameter needs to be consistent with the setting of fe's thrift_server_type parameter. Currently there are two values for this parameter, `THREADED` and `THREAD_POOL`.
    - If the parameter is `THREADED`, the model is a non-blocking I/O model.

    - If the parameter is `THREAD_POOL`, the model is a blocking I/O model.

`thrift_max_message_size`

:::info Note

This field is supported since version 2.0.12.

:::

Default: 100MB

The maximum size of a (received) message of the thrift server, in bytes. If the size of the message sent by the client exceeds this limit, the Thrift server will reject the request and close the connection. As a result, the client will encounter the error: “connection has been closed by peer.” In this case, you can try increasing this parameter.

`txn_commit_rpc_timeout_ms`

- Description:txn submit rpc timeout
- Default value: 60,000 (ms)

`txn_map_shard_size`

- Description: txn_map_lock fragment size, the value is 2^n, n=0,1,2,3,4. This is an enhancement to improve the performance of managing txn
- Default value: 128

`txn_shard_size`

- Description: txn_lock shard size, the value is 2^n, n=0,1,2,3,4, this is an enhancement function that can improve the performance of submitting and publishing txn
- Default value: 1024

`unused_rowset_monitor_interval`

- Description: Time interval for clearing expired Rowset
- Default value: 30 (s)

`max_client_cache_size_per_host`

- Description: The maximum number of client caches per host. There are multiple client caches in BE, but currently we use the same cache size configuration. If necessary, use different configurations to set up different client-side caches
- Default value: 10

`string_type_length_soft_limit_bytes`

- Type: int32

- Description: The soft limit of the maximum length of String type.
- Default value: 1,048,576

`big_column_size_buffer`

- Type: int64
- Description: When using the odbc external table, if a column type of the odbc source table is HLL, CHAR or VARCHAR, and the length of the column value exceeds this value, the query will report an error 'column value length longer than buffer length'. You can increase this value
- Default value: 65535

`small_column_size_buffer`

- Type: int64
- Description: When using the odbc external table, if a column type of the odbc source table is not HLL, CHAR or VARCHAR, and the length of the column value exceeds this value, the query will report an error 'column value length longer than buffer length'. You can increase this value
- Default value: 100

`jsonb_type_length_soft_limit_bytes`

- Type: int32
- Description: The soft limit of the maximum length of JSONB type.
- Default value: 1,048,576

Query

`fragment_pool_queue_size`

- Description: The upper limit of query requests that can be processed on a single node
- Default value: 4096

`fragment_pool_thread_num_min`

- Description: Query the number of threads. By default, the minimum number of threads is 64.
- Default value: 64

`fragment_pool_thread_num_max`

- Description: Follow up query requests create threads dynamically, with a maximum of 512 threads created.
- Default value: 2048

`doris_max_pushdown_conjuncts_return_rate`

- Type: int32

- Description: When BE performs HashJoin, it will adopt a dynamic partitioning method to push the join condition to OlapScanner. When the data scanned by OlapScanner is larger than 32768 rows, BE will check the filter condition. If the filter rate of the filter condition is lower than this configuration, Doris will stop using the dynamic partition clipping condition for data filtering.
- Default value: 90

`doris_max_scan_key_num`

- Type: int
- Description: Used to limit the maximum number of scan keys that a scan node can split in a query request. When a conditional query request reaches the scan node, the scan node will try to split the conditions related to the key column in the query condition into multiple scan key ranges. After that, these scan key ranges will be assigned to multiple scanner threads for data scanning. A larger value usually means that more scanner threads can be used to increase the parallelism of the scanning operation. However, in high concurrency scenarios, too many threads may bring greater scheduling overhead and system load, and will slow down the query response speed. An empirical value is 50. This configuration can be configured separately at the session level. For details, please refer to the description of `max_scan_key_num` in Variables.
    - When the concurrency cannot be improved in high concurrency scenarios, try to reduce this value and observe the impact.
- Default value: 48

`doris_scan_range_row_count`

- Type: int32
- Description: When BE performs data scanning, it will split the same scanning range into multiple ScanRanges. This parameter represents the scan data range of each ScanRange. This parameter can limit the time that a single OlapScanner occupies the io thread.
- Default value: 524288

`doris_scanner_row_num`

- Description: The maximum number of data rows returned by each scanning thread in a single execution
- Default value: 16384

`doris_scanner_row_bytes`

- Description: single read execute fragment row bytes
    - Note: If there are too many columns in the table, you can adjust this config if you encounter a `select *` stuck
- Default value: 10485760

`doris_scanner_thread_pool_queue_size`

- Type: int32
- Description: The queue length of the Scanner thread pool. In Doris' scanning tasks, each Scanner will be submitted as a thread task to the thread pool waiting to be scheduled, and after the number of submitted

tasks exceeds the length of the thread pool queue, subsequent submitted tasks will be blocked until there is a empty slot in the queue.
- Default value: 102400

`doris_scanner_thread_pool_thread_num`

- Type: int32
- Description: The number of threads in the Scanner thread pool. In Doris' scanning tasks, each Scanner will be submitted as a thread task to the thread pool to be scheduled. This parameter determines the size of the Scanner thread pool.
- Default value: 48

`doris_max_remote_scanner_thread_pool_thread_num`

- Type: int32
- Description: Max thread number of Remote scanner thread pool. Remote scanner thread pool is used for scan task of all external data sources.
- Default: 512

`enable_prefetch`

- Type: bool
- Description: When using PartitionedHashTable for aggregation and join calculations, whether to perform HashBuket prefetch. Recommended to be set to true
- Default value: true

`enable_quadratic_probing`

- Type: bool
- Description: When a Hash conflict occurs when using PartitionedHashTable, enable to use the square detection method to resolve the Hash conflict. If the value is false, linear detection is used to resolve the Hash conflict. For the square detection method, please refer to: quadratic_probing
- Default value: true

`exchg_node_buffer_size_bytes`

- Type: int32
- Description: The size of the Buffer queue of the ExchangeNode node, in bytes. After the amount of data sent from the Sender side is larger than the Buffer size of ExchangeNode, subsequent data sent will block until the Buffer frees up space for writing.
- Default value: 10485760

`max_pushdown_conditions_per_column`

- Type: int
- Description: Used to limit the maximum number of conditions that can be pushed down to the storage engine for a single column in a query request. During the execution of the query plan, the filter

conditions on some columns can be pushed down to the storage engine, so that the index information in the storage engine can be used for data filtering, reducing the amount of data that needs to be scanned by the query. Such as equivalent conditions, conditions in IN predicates, etc. In most cases, this parameter only affects queries containing IN predicates. Such as `WHERE colA IN (1,2,3,4, ...)`. A larger number means that more conditions in the IN predicate can be pushed to the storage engine, but too many conditions may cause an increase in random reads, and in some cases may reduce query efficiency. This configuration can be individually configured for session level. For details, please refer to the description of `max_pushdown_conditions_per_column` in Variables.

- Default value: 1024
- Example
    - The table structure is' id INT, col2 INT, col3 varchar (32),… '.
    - The query request is'WHERE id IN (v1, v2, v3, …) #### `max_send_batch_parallelism_per_job`
- Type: int
- Description: Max send batch parallelism for OlapTableSink. The value set by the user for `send_batch_parallelism` is not allowed to exceed `max_send_batch_parallelism_per_job`, if exceed, the value of `send_batch_parallelism` would be `max_send_batch_parallelism_per_job`.
- Default value: 5

`doris_scan_range_max_mb`

- Type: int32
- Description: The maximum amount of data read by each OlapScanner.
- Default value: 1024

compaction

`disable_auto_compaction`

- Type: bool
- Description: Whether disable automatic compaction task
    - Generally it needs to be turned off. When you want to manually operate the compaction task in the debugging or test environment, you can turn on the configuration.
- Default value: false

`enable_vertical_compaction`

- Type: bool
- Description: Whether to enable vertical compaction
- Default value: true

`vertical_compaction_num_columns_per_group`

- Type: int32

- Description: In vertical compaction, column number for every group
- Default value: 5

`vertical_compaction_max_row_source_memory_mb`

- Type: int32
- Description: In vertical compaction, max memory usage for row_source_buffer,The unit is MB.
- Default value: 200

`vertical_compaction_max_segment_size`

- Type: int32
- Description: In vertical compaction, max dest segment file size, The unit is m bytes.
- Default value: 268435456

`enable_ordered_data_compaction`

- Type: bool
- Description: Whether to enable ordered data compaction
- Default value: true

`ordered_data_compaction_min_segment_size`

- Type: int32
- Description: In ordered data compaction, min segment size for input rowset, The unit is m bytes.
- Default value: 10485760

`max_base_compaction_threads`

- Type: int32
- Description: The maximum of thread number in base compaction thread pool.
- Default value: 4

`generate_compaction_tasks_interval_ms`

- Description: Minimal interval (ms) to generate compaction tasks
- Default value: 10 (ms)

`base_compaction_min_rowset_num`

- Description: One of the triggering conditions of BaseCompaction: The limit of the number of Cumulative files to be reached. After reaching this limit, BaseCompaction will be triggered
- Default value: 5

`base_compaction_min_data_ratio`

- Description: One of the trigger conditions of BaseCompaction: Cumulative file size reaches the proportion of Base file
- Default value: 0.3 (30%)

`total_permits_for_compaction_score`

- Type: int64
- Description: The upper limit of “permits” held by all compaction tasks. This config can be set to limit memory consumption for compaction.
- Default value: 10000
- Dynamically modifiable: Yes

`compaction_promotion_size_mbytes`

- Type: int64
- Description: The total disk size of the output rowset of cumulative compaction exceeds this configuration size, and the rowset will be used for base compaction. The unit is m bytes.
    - Generally, if the configuration is less than 2G, in order to prevent the cumulative compression time from being too long, resulting in the version backlog.
- Default value: 1024

`compaction_promotion_ratio`

- Type: double
- Description: When the total disk size of the cumulative compaction output rowset exceeds the configuration ratio of the base version rowset, the rowset will be used for base compaction.
    - Generally, it is recommended that the configuration should not be higher than 0.1 and lower than 0.02.
- Default value: 0.05

`compaction_promotion_min_size_mbytes`

- Type: int64
- Description: If the total disk size of the output rowset of the cumulative compaction is lower than this configuration size, the rowset will not undergo base compaction and is still in the cumulative compaction process. The unit is m bytes.
    - Generally, the configuration is within 512m. If the configuration is too large, the size of the early base version is too small, and base compaction has not been performed.
- Default value: 64

`compaction_min_size_mbytes`

- Type: int64
- Description: When the cumulative compaction is merged, the selected rowsets to be merged have a larger disk size than this configuration, then they are divided and merged according to the level policy.

When it is smaller than this configuration, merge directly. The unit is m bytes.
    - Generally, the configuration is within 128m. Over configuration will cause more cumulative compaction write amplification.
- Default value: 64

`default_rowset_type`

- Type: string
- Description: Identifies the storage format selected by BE by default. The configurable parameters are: “**ALPHA**”, “**BETA**”. Mainly play the following two roles
    - When the storage_format of the table is set to Default, select the storage format of BE through this configuration.
    - Select the storage format of when BE performing Compaction
- Default value: BETA

`cumulative_compaction_min_deltas`

- Description: Cumulative compaction strategy: the minimum number of incremental files
- Default value: 5

`cumulative_compaction_max_deltas`

- Description: Cumulative compaction strategy: the maximum number of incremental files
- Default value: 1000

`base_compaction_trace_threshold`

- Type: int32
- Description: Threshold to logging base compaction’s trace information, in seconds
- Default value: 10

Base compaction is a long time cost background task, this configuration is the threshold to logging trace information. Trace information in log file looks like:

```
W0610 11:26:33.804431 56452 storage_engine.cpp:552] execute base compaction cost 0.00319222
BaseCompaction:546859:
  - filtered_rows: 0
   - input_row_num: 10
   - input_rowsets_count: 10
   - input_rowsets_data_size: 2.17 KB
   - input_segments_num: 10
   - merge_rowsets_latency: 100000.510ms
   - merged_rows: 0
   - output_row_num: 10
   - output_rowset_data_size: 224.00 B
   - output_segments_num: 1
0610 11:23:03.727535 (+     0us) storage_engine.cpp:554] start to perform base compaction
0610 11:23:03.728961 (+  1426us) storage_engine.cpp:560] found best tablet 546859
0610 11:23:03.728963 (+     2us) base_compaction.cpp:40] got base compaction lock
0610 11:23:03.729029 (+    66us) base_compaction.cpp:44] rowsets picked
0610 11:24:51.784439 (+108055410us) compaction.cpp:46] got concurrency lock and start to do compaction
0610 11:24:51.784818 (+   379us) compaction.cpp:74] prepare finished
0610 11:26:33.359265 (+101574447us) compaction.cpp:87] merge rowsets finished
```

```
0610 11:26:33.484481 (+125216us) compaction.cpp:102] output rowset built
0610 11:26:33.484482 (+     1us) compaction.cpp:106] check correctness finished
0610 11:26:33.513197 (+ 28715us) compaction.cpp:110] modify rowsets finished
0610 11:26:33.513300 (+   103us) base_compaction.cpp:49] compaction finished
0610 11:26:33.513441 (+   141us) base_compaction.cpp:56] unused rowsets have been moved to GC queue
```

`cumulative_compaction_trace_threshold`

- Type: int32
- Description: Threshold to logging cumulative compaction's trace information, in seconds
    - Similar to `base_compaction_trace_threshold`.
- Default value: 2

`compaction_task_num_per_disk`

- Type: int32
- Description: The number of compaction tasks which execute in parallel for a disk(HDD).
- Default value: 4

`compaction_task_num_per_fast_disk`

- Type: int32
- Description: The number of compaction tasks which execute in parallel for a fast disk(SSD).
- Default value: 8

`cumulative_compaction_rounds_for_each_base_compaction_round`

- Type: int32
- Description: How many rounds of cumulative compaction for each round of base compaction when compaction tasks generation.
- Default value: 9

`cumulative_compaction_policy`

- Type: string
- Description: Configure the merge strategy in the cumulative compression phase. Currently, two merge strategies are implemented, num_based and size_based
    - For details, "ordinary" is the initial version of the cumulative compression consolidation policy. After a cumulative compression, the base compression process is directly performed. size_The general policy is the optimized version of the ordinary policy. Version merging can only be performed when the disk volume of the rowset is the same order of magnitude. After merging, qualified rowsets are promoted to the base compaction stage. In the case of a large number of small batch imports, it can reduce the write magnification of base compact, balance the read magnification and space magnification, and reduce the data of file versions.
- Default value: size_based

`max_cumu_compaction_threads`

- Type: int32
- Description: The maximum of thread number in cumulative compaction thread pool.
- Default value: 10

`enable_segcompaction`

- Type: bool
- Description: Enable to use segment compaction during loading to avoid -238 error
- Default value: true

`segcompaction_batch_size`

- Type: int32
- Description: Segment compaction is triggered when the number of segments exceeds this threshold. This configuration also limits the maximum number of raw segments in a single segment compaction task.
- Default value: 10

`segcompaction_candidate_max_rows`

- Type: int32
- Description: Max row count allowed in a single source segment, bigger segments will be skipped.
- Default value: 1048576

`segcompaction_candidate_max_bytes`

- Type: int64
- Description: Max file size allowed in a single source segment, bigger segments will be skipped.
- Default value: 104857600

`segcompaction_task_max_rows`

- Type: int32
- Description: Max total row count allowed in a single segcompaction task.
- Default value: 1572864

`segcompaction_task_max_bytes`

- Type: int64
- Description: Max total file size allowed in a single segcompaction task.
- Default value: 157286400

`segcompaction_num_threads`

- Type: int32

- Description: Global segcompaction thread pool size.
- Default value: 5

`disable_compaction_trace_log`

- Type: bool
- Description: disable the trace log of compaction
    - If set to true, the `cumulative_compaction_trace_threshold` and `base_compaction_trace_threshold` won't work and log is disabled.
- Default value: true

`pick_rowset_to_compact_interval_sec`

- Type: int64
- Description: select the time interval in seconds for rowset to be compacted.
- Default value: 86400

`max_single_replica_compaction_threads`

- Type: int32
- Description: The maximum of thread number in single replica compaction thread pool.
- Default value: 10

`update_replica_infos_interval_seconds`

- Description: Minimal interval (s) to update peer replica infos
- Default value: 60 (s)

Load

`enable_stream_load_record`

- Type: bool
- Description:Whether to enable stream load record function, the default is false.
- Default value: false

`load_data_reserve_hours`

- Description: Used for mini load. The mini load data file will be deleted after this time
- Default value: 4 (h)

`push_worker_count_high_priority`

- Description: Import the number of threads for processing HIGH priority tasks
- Default value: 3

`push_worker_count_normal_priority`

- Description: Import the number of threads for processing NORMAL priority tasks
- Default value: 3

`enable_single_replica_load`

- Description: Whether to enable the single-copy data import function
- Default value: true

`load_error_log_reserve_hours`

- Description: The load error log will be deleted after this time
- Default value: 48 (h)

`load_process_max_memory_limit_percent`

- Description: The percentage of the upper memory limit occupied by all imported threads on a single node, the default is 50%
    - Set these default values very large, because we don't want to affect load performance when users upgrade Doris. If necessary, the user should set these configurations correctly
- Default value: 50 (%)

`load_process_soft_mem_limit_percent`

- Description: The soft limit refers to the proportion of the load memory limit of a single node. For example, the load memory limit for all load tasks is 20GB, and the soft limit defaults to 50% of this value, that is, 10GB. When the load memory usage exceeds the soft limit, the job with the largest memory consuption will be selected to be flushed to release the memory space, the default is 50%
- Default value: 50 (%)

`routine_load_thread_pool_size`

- Description: The thread pool size of the routine load task. This should be greater than the FE configuration'max_concurrent_task_num_per_be'
- Default value: 10

`slave_replica_writer_rpc_timeout_sec`

- Type: int32
- Description: This configuration is mainly used to modify timeout of brpc between master replica and slave replica, used for single replica load.
- Default value: 60

`max_segment_num_per_rowset`

- Type: int32

- Description: Used to limit the number of segments in the newly generated rowset when importing. If the threshold is exceeded, the import will fail with error -238. Too many segments will cause compaction to take up a lot of memory and cause OOM errors.
- Default value: 200

`high_priority_flush_thread_num_per_store`

- Type: int32
- Description: The number of flush threads per store path allocated for the high priority import task.
- Default value: 1

`routine_load_consumer_pool_size`

- Type: int32
- Description: The number of caches for the data consumer used by the routine load.
- Default value: 10

`multi_table_batch_plan_threshold`

- Type: int32
- Description: For single-stream-multi-table load. When receive a batch of messages from kafka, if the size of batch is more than this threshold, we will request plans for all related tables.
- Default value: 200

`single_replica_load_download_num_workers`

- Type: int32
- Description:This configuration is mainly used to modify the number of http worker threads for segment download, used for single replica load. When the load concurrency increases, you can adjust this parameter to ensure that the Slave replica synchronizes data files from the Master replica timely. If needed, `webserver_num_workers` should also be increased for better IO performance.
- Default value: 64

`load_task_high_priority_threshold_second`

- Type: int32
- Description: When the timeout of an import task is less than this threshold, Doris will consider it to be a high priority task. High priority tasks use a separate pool of flush threads.
- Default: 120

`min_load_rpc_timeout_ms`

- Type: int32
- Description: The minimum timeout for each rpc in the load job.
- Default: 20

`kafka_api_version_request`

- Type: bool
- Description: If the dependent Kafka version is lower than 0.10.0.0, this value should be set to false.
- Default: true

`kafka_broker_version_fallback`

- Description: If the dependent Kafka version is lower than 0.10.0.0, the value set by the fallback version kafka_broker_version_fallback will be used if the value of kafka_api_version_request is set to false, and the valid values are: 0.9.0.x, 0.8.x.y.
- Default: 0.10.0

`max_consumer_num_per_group`

- Description: The maximum number of consumers in a data consumer group, used for routine load
- Default: 3

`streaming_load_max_mb`

- Type: int64
- Description: Used to limit the maximum amount of csv data allowed in one Stream load.
    - Stream Load is generally suitable for loading data less than a few GB, not suitable for loading too large data.
- Default value: 10240 (MB)
- Dynamically modifiable: Yes

`streaming_load_json_max_mb`

- Type: int64
- Description: it is used to limit the maximum amount of json data allowed in one Stream load. The unit is MB.
    - Some data formats, such as JSON, cannot be split. Doris must read all the data into the memory before parsing can begin. Therefore, this value is used to limit the maximum amount of data that can be loaded in a single Stream load.
- Default value: 100
- Dynamically modifiable: Yes

Thread

`delete_worker_count`

- Description: Number of threads performing data deletion tasks
- Default value: 3

`clear_transaction_task_worker_count`

- Description: Number of threads used to clean up transactions
- Default value: 1

`clone_worker_count`

- Description: Number of threads used to perform cloning tasks
- Default value: 3

`be_service_threads`

- Type: int32
- Description: The number of execution threads of the thrift server service on BE which represents the number of threads that can be used to execute FE requests.
- Default value: 64

`download_worker_count`

- Description: The number of download threads.
- Default value: 1

`drop_tablet_worker_count`

- Description: Number of threads to delete tablet
- Default value: 3

`flush_thread_num_per_store`

- Description: The number of threads used to refresh the memory table per store
- Default value: 2

`num_threads_per_core`

- Description: Control the number of threads that each core runs. Usually choose 2 times or 3 times the number of cores. This keeps the core busy without causing excessive jitter
- Default value: 3

`num_threads_per_disk`

- Description: The maximum number of threads per disk is also the maximum queue depth of each disk
- Default value: 0

`number_slave_replica_download_threads`

- Description: Number of threads for slave replica synchronize data, used for single replica load.
- Default value: 64

`publish_version_worker_count`

- Description: the count of thread to publish version
- Default value: 8

`upload_worker_count`

- Description: Maximum number of threads for uploading files
- Default value: 1

`webserver_num_workers`

- Description: Webserver default number of worker threads
- Default value: 48

`send_batch_thread_pool_thread_num`

- Type: int32
- Description: The number of threads in the SendBatch thread pool. In NodeChannels' sending data tasks, the SendBatch operation of each NodeChannel will be submitted as a thread task to the thread pool to be scheduled. This parameter determines the size of the SendBatch thread pool.
- Default value: 64

`send_batch_thread_pool_queue_size`

- Type: int32
- Description: The queue length of the SendBatch thread pool. In NodeChannels' sending data tasks, the SendBatch operation of each NodeChannel will be submitted as a thread task to the thread pool waiting to be scheduled, and after the number of submitted tasks exceeds the length of the thread pool queue, subsequent submitted tasks will be blocked until there is a empty slot in the queue.
- Default value: 102400

`make_snapshot_worker_count`

- Description: Number of threads making snapshots
- Default value: 5

`release_snapshot_worker_count`

- Description: Number of threads releasing snapshots
- Default value: 5

Memory

`disable_mem_pools`

- Type: bool

- Description: Whether to disable the memory cache pool.
- Default value: false

`buffer_pool_clean_pages_limit`

- Description: Clean up pages that may be saved by the buffer pool
- Default value: 50%

`buffer_pool_limit`

- Type: string
- Description: The largest allocatable memory of the buffer pool
    - The maximum amount of memory available in the BE buffer pool. The buffer pool is a new memory management structure of BE, which manages the memory by the buffer page and enables spill data to disk. The memory for all concurrent queries will be allocated from the buffer pool. The current buffer pool only works on **AggregationNode** and **ExchangeNode**.
- Default value: 20%

`chunk_reserved_bytes_limit`

- Description: The reserved bytes limit of Chunk Allocator, usually set as a percentage of mem_limit. defaults to bytes if no unit is given, the number of bytes must be a multiple of 2. must larger than 0. and if larger than physical memory size, it will be set to physical memory size. increase this variable can improve performance, but will acquire more free memory which can not be used by other modules.
- Default value: 20%

`madvise_huge_pages`

- Type: bool
- Description: Whether to use linux memory huge pages.
- Default value: false

`max_memory_sink_batch_count`

- Description: The maximum external scan cache batch count, which means that the cache max_memory_cache_batch_count * batch_size row, the default is 20, and the default value of batch_size is 1024, which means that 20 * 1024 rows will be cached
- Default value: 20

`memory_max_alignment`

- Description: Maximum alignment memory
- Default value: 16

`mmap_buffers`

- Description: Whether to use mmap to allocate memory

- Default value: false

`memtable_mem_tracker_refresh_interval_ms`

- Description: Interval in milliseconds between memtable flush mgr refresh iterations
- Default value: 100

`download_cache_buffer_size`

- Type: int64
- Description: The size of the buffer used to receive data when downloading the cache.
- Default value: 10485760

`zone_map_row_num_threshold`

- Type: int32
- Description: If the number of rows in a page is less than this value, no zonemap will be created to reduce data expansion
- Default value: 20

`enable_tcmalloc_hook`

- Type: bool
- Description: Whether Hook TCmalloc new/delete, currently consume/release tls mem tracker in Hook.
- Default value: true

`memory_mode`

- Type: string
- Description: Control gc of tcmalloc, in performance mode doirs releases memory of tcmalloc cache when usgae >= 90% * mem_limit, otherwise, doris releases memory of tcmalloc cache when usage >= 50% * mem_limit;
- Default value: performance

`max_sys_mem_available_low_water_mark_bytes`

- Type: int64
- Description: The maximum low water mark of the system `/proc/meminfo/MemAvailable`, Unit byte, default 1.6G, actual low water mark=min(1.6G, MemTotal * 10%), avoid wasting too much memory on machines with large memory larger than 16G. Turn up max. On machines with more than 16G memory, more memory buffers will be reserved for Full GC. Turn down max. will use as much memory as possible.
- Default value: 1717986918

`memory_limitation_per_thread_for_schema_change_bytes`

- Description: Maximum memory allowed for a single schema change task
- Default value: 2147483648 (2GB)

`mem_tracker_consume_min_size_bytes`

- Type: int32
- Description: The minimum length of TCMalloc Hook when consume/release MemTracker. Consume size smaller than this value will continue to accumulate to avoid frequent calls to consume/release of MemTracker. Decreasing this value will increase the frequency of consume/release. Increasing this value will cause MemTracker statistics to be inaccurate. Theoretically, the statistical value of a MemTracker differs from the true value = (mem_tracker_consume_min_size_bytes * the number of BE threads where the MemTracker is located).
- Default value: 1,048,576

`cache_clean_interval`

- Description: File handle cache cleaning interval, used to clean up file handles that have not been used for a long time.Also the clean interval of Segment Cache.
- Default value: 1800 (s)

`min_buffer_size`

- Description: Minimum read buffer size
- Default value: 1024 (byte)

`write_buffer_size`

- Description: The size of the buffer before flashing
    - Imported data is first written to a memory block on the BE, and only written back to disk when this memory block reaches the threshold. The default size is 100MB. too small a threshold may result in a large number of small files on the BE. This threshold can be increased to reduce the number of files. However, too large a threshold may cause RPC timeouts
- Default value: 104,857,600

`remote_storage_read_buffer_mb`

- Type: int32
- Description: The cache size used when reading files on hdfs or object storage.
    - Increasing this value can reduce the number of calls to read remote data, but it will increase memory overhead.
- Default value: 16 (MB)

`file_cache_type`

- Type: string
- Description: Type of cache file.`whole_file_cache`: download the entire segment file, `sub_file_cache`: the segment file is divided into multiple files by size. if set “”, no cache, please set this parameter when caching is required.
- Default value: “”

`file_cache_alive_time_sec`

- Type: int64
- Description: Save time of cache file
- Default value: 604800 (1 week)

`file_cache_max_size_per_disk`

- Type: int64
- Description: The cache occupies the disk size. Once this setting is exceeded, the cache that has not been accessed for the longest time will be deleted. If it is 0, the size is not limited. unit is bytes.
- Default value: 0

`max_sub_cache_file_size`

- Type: int64
- Description: Cache files using sub_ file_ The maximum size of the split file during cache
- Default value: 104857600 (100MB)

`download_cache_thread_pool_thread_num`

- Type: int32
- Description: The number of threads in the DownloadCache thread pool. In the download cache task of FileCache, the download cache operation will be submitted to the thread pool as a thread task and wait to be scheduled. This parameter determines the size of the DownloadCache thread pool.
- Default value: 48

`download_cache_thread_pool_queue_size`

- Type: int32
- Description: The number of threads in the DownloadCache thread pool. In the download cache task of FileCache, the download cache operation will be submitted to the thread pool as a thread task and wait to be scheduled. After the number of submitted tasks exceeds the length of the thread pool queue, subsequent submitted tasks will be blocked until there is a empty slot in the queue.
- Default value: 102400

`generate_cache_cleaner_task_interval_sec`

- Type：int64
- Description: Cleaning interval of cache files, in seconds
- Default: 43200 (12 hours)

`path_gc_check`

- Type：bool
- Description: Whether to enable the recycle scan data thread check
- Default：true

`path_gc_check_interval_second`

- Description: Recycle scan data thread check interval
- Default：86400 (s)

`path_gc_check_step`

- Default：1000

`path_gc_check_step_interval_ms`

- Default：10 (ms)

`path_scan_interval_second`

- Default：86400

`scan_context_gc_interval_min`

- Description: This configuration is used for the context gc thread scheduling cycle. Note: The unit is minutes, and the default is 5 minutes
- Default：5

Storage

`default_num_rows_per_column_file_block`

- Type: int32
- Description: Configure how many rows of data are contained in a single RowBlock.
- Default value: 1024

`disable_storage_page_cache`

- Type: bool
- Description: Disable to use page cache for index caching, this configuration only takes effect in BETA storage format, usually it is recommended to false
- Default value: false

`disk_stat_monitor_interval`

- Description: Disk status check interval
- Default value: 5 (s)

`max_free_io_buffers`

- Description: For each io buffer size, the maximum number of buffers that IoMgr will reserve ranges from 1024B to 8MB buffers, up to about 2GB buffers.

- Default value: 128

`max_garbage_sweep_interval`

- Description: The maximum interval for disk garbage cleaning
- Default value: 3600 (s)

`max_percentage_of_error_disk`

- Type: int32
- Description: The storage engine allows the percentage of damaged hard disks to exist. After the damaged hard disk exceeds the changed ratio, BE will automatically exit.
- Default value: 0

`read_size`

- Description: The read size is the read size sent to the os. There is a trade-off between latency and the whole process, getting to keep the disk busy but not introducing seeks. For 8 MB reads, random io and sequential io have similar performance.
- Default value: 8388608

`min_garbage_sweep_interval`

- Description: The minimum interval between disk garbage cleaning
- Default value: 180 (s)

`pprof_profile_dir`

- Description: pprof profile save directory
- Default value: ${DORIS_HOME}/log

`small_file_dir`

- Description: Save files downloaded by SmallFileMgr
- Default value: ${DORIS_HOME}/lib/small_file/

`user_function_dir`

- Description: udf function directory
- Default value: ${DORIS_HOME}/lib/udf

`storage_flood_stage_left_capacity_bytes`

- Description: The min bytes that should be left of a data dir.
- Default value: 1073741824

`storage_flood_stage_usage_percent`

- Description: The storage_flood_stage_usage_percent and storage_flood_stage_left_capacity_bytes configurations limit the maximum usage of the capacity of the data directory.
- Default value: 90 (90%)

`storage_medium_migrate_count`

- Description: the count of thread to clone
- Default value: 1

`storage_page_cache_limit`

- Description: Cache for storage page size
- Default value: 20%

`storage_page_cache_shard_size`

- Description: Shard size of StoragePageCache, the value must be power of two. It's recommended to set it to a value close to the number of BE cores in order to reduce lock contentions.
- Default value: 16

`index_page_cache_percentage`

- Type: int32
- Description: Index page cache as a percentage of total storage page cache, value range is [0, 100]
- Default value: 10

`segment_cache_capacity`

- Type: int32
- Description: Max number of segment cache (the key is rowset id) entries. -1 is for backward compatibility as fd_number * 2/5.
- Default value: -1

`storage_strict_check_incompatible_old_format`

- Type: bool
- Description: Used to check incompatible old format strictly
- Default value: true
- Dynamically modify: false

`sync_tablet_meta`

- Description: Whether the storage engine opens sync and keeps it to the disk
- Default value: false

`pending_data_expire_time_sec`

- Description: The maximum duration of unvalidated data retained by the storage engine
- Default value: 1800 (s)

`ignore_rowset_stale_unconsistent_delete`

- Type: boolean
- Description:It is used to decide whether to delete the outdated merged rowset if it cannot form a consistent version path.
    - The merged expired rowset version path will be deleted after half an hour. In abnormal situations, deleting these versions will result in the problem that the consistent path of the query cannot be constructed. When the configuration is false, the program check is strict and the program will directly report an error and exit.When configured as true, the program will run normally and ignore this error. In general, ignoring this error will not affect the query, only when the merged version is dispatched by fe, -230 error will appear.
- Default value: false

`create_tablet_worker_count`

- Description: Number of worker threads for BE to create a tablet
- Default value: 3

`check_consistency_worker_count`

- Description: The number of worker threads to calculate the checksum of the tablet
- Default value: 1

`max_tablet_version_num`

- Type: int
- Description: Limit the number of versions of a single tablet. It is used to prevent a large number of version accumulation problems caused by too frequent import or untimely compaction. When the limit is exceeded, the import task will be rejected.
- Default value: 500

`number_tablet_writer_threads`

- Description: Number of tablet write threads
- Default value: 16

`tablet_map_shard_size`

- Description: tablet_map_lock fragment size, the value is 2^n, n=0,1,2,3,4, this is for better tablet management
- Default value: 4

`tablet_meta_checkpoint_min_interval_secs`

- Description: TabletMeta Checkpoint 线程轮询的时间间隔
- Default value: 600 (s)

`tablet_meta_checkpoint_min_new_rowsets_num`

- Description: The minimum number of Rowsets for storing TabletMeta Checkpoints
- Default value: 10

`tablet_stat_cache_update_interval_second`

- Description: Update interval of tablet state cache
- Default value:300 (s)

`tablet_rowset_stale_sweep_time_sec`

- Type: int64
- Description: It is used to control the expiration time of cleaning up the merged rowset version. When the current time now() minus the max created rowset‘s create time in a version path is greater than tablet_rowset_stale_sweep_time_sec, the current path is cleaned up and these merged rowsets are deleted, the unit is second.
    - When writing is too frequent, Fe may not be able to query the merged version, resulting in a query -230 error. This problem can be avoided by increasing this parameter.
- Default value: 300

`tablet_writer_open_rpc_timeout_sec`

- Description: Update interval of tablet state cache
    - The RPC timeout for sending a Batch (1024 lines) during import. The default is 60 seconds. Since this RPC may involve writing multiple batches of memory, the RPC timeout may be caused by writing batches, so this timeout can be adjusted to reduce timeout errors (such as send batch fail errors). Also, if you increase the write_buffer_size configuration, you need to increase this parameter as well.
- Default value: 60

`tablet_writer_ignore_eovercrowded`

- Type: bool
- Description: Used to ignore brpc error ‘[E1011]The server is overcrowded’ when writing data.
- Default value: false

`streaming_load_rpc_max_alive_time_sec`

- Description: The lifetime of TabletsChannel. If the channel does not receive any data at this time, the channel will be deleted.
- Default value: 1200

alter_tablet_worker_count

- Description: The number of threads making schema changes
- Default value: 3

alter_index_worker_count

- Description: The number of threads making index change
- Default value: 3

ignore_load_tablet_failure

- Type: bool
- Description: It is used to decide whether to ignore errors and continue to start be in case of tablet loading failure
- Default value: false

When BE starts, a separate thread will be started for each data directory to load the meta information of the tablet header. In the default configuration, if a data directory fails to load a tablet, the startup process will terminate. At the same time, it will be displayed in the `be The following error message is seen in the INFO` log:

```
load tablets from header failed, failed tablets size: xxx, path=xxx
```

Indicates how many tablets failed to load in the data directory. At the same time, the log will also contain specific information about the tablets that failed to load. At this time, manual intervention is required to troubleshoot the cause of the error. After troubleshooting, there are usually two ways to restore: - The tablet information cannot be repaired. If the other copies are normal, you can delete the wrong tablet with the `meta_tool` tool. - Set `ignore_load_tablet_failure` to true, BE will ignore these faulty tablets and start normally

report_disk_state_interval_seconds

- Description: The interval time for the agent to report the disk status to FE
- Default value: 60 (s)

result_buffer_cancelled_interval_time

- Description: Result buffer cancellation time
- Default value: 300 (s)

snapshot_expire_time_sec

- Description: Snapshot file cleaning interval.
- Default value:172800 (48 hours)

compress_rowbatches

- Type: bool

- Description: enable to use Snappy compression algorithm for data compression when serializing RowBatch
- Default value: true

:::tip Tips This configuration is supported since the Apache Doris 1.2 version :::

`jvm_max_heap_size`

- Type: string
- Description: The maximum size of JVM heap memory used by BE, which is the `-Xmx` parameter of JVM
- Default value: 1024M

Log

`sys_log_dir`

- Type: string
- Description: Storage directory of BE log data
- Default value: ${DORIS_HOME}/log

`sys_log_level`

- Description: Log Level: INFO < WARNING < ERROR < FATAL
- Default value: INFO

`sys_log_roll_mode`

- Description: The size of the log split, one log file is split every 1G
- Default value: SIZE-MB-1024

`sys_log_roll_num`

- Description: Number of log files kept
- Default value: 10

`sys_log_verbose_level`

- Description: Log display level, used to control the log output at the beginning of VLOG in the code
- Default value: 10

`sys_log_verbose_modules`

- Description: Log printing module, writing olap will only print the log under the olap module
- Default value: empty

`aws_log_level`

- Type: int32

- Description: log level of AWS SDK,

```
Off = 0,
Fatal = 1,
Error = 2,
Warn = 3,
Info = 4,
Debug = 5,
Trace = 6
```

- Default value: 3

`log_buffer_level`

- Description: The log flushing strategy is kept in memory by default
- Default value: empty

Else

`report_tablet_interval_seconds`

- Description: The interval time for the agent to report the olap table to the FE
- Default value: 60 (s)

`report_task_interval_seconds`

- Description: The interval time for the agent to report the task signature to FE
- Default value: 10 (s)

`periodic_counter_update_period_ms`

- Description: Update rate counter and sampling counter cycle
- Default value: 500 (ms)

`enable_metric_calculator`

- Description: If set to true, the metric calculator will run to collect BE-related indicator information, if set to false, it will not run
- Default value: true

`enable_system_metrics`

- Description: User control to turn on and off system indicators.
- Default value: true

`enable_token_check`

- Description: Used for forward compatibility, will be removed later.
- Default value: true

`max_runnings_transactions_per_txn_map`

- Description: Max number of txns for every txn_partition_map in txn manager, this is a self protection to avoid too many txns saving in manager
- Default value: 2000

`max_download_speed_kbps`

- Description: Maximum download speed limit
- Default value: 50000 (kb/s)

`download_low_speed_time`

- Description: Download time limit
- Default value: 300 (s)

`download_low_speed_limit_kbps`

- Description: Minimum download speed
- Default value: 50 (KB/s)

`doris_cgroups`

- Description: Cgroups assigned to doris
- Default value: empty

`priority_queue_remaining_tasks_increased_frequency`

- Description: the increased frequency of priority for remaining tasks in BlockingPriorityQueue
- Default value: 512

> **Tips**
>
> This configuration is supported since the Apache Doris 1.2 version

`jdbc_drivers_dir`

- Description: Default dirs to put jdbc drivers.
- Default value: `${DORIS_HOME}/jdbc_drivers`

`enable_parse_multi_dimession_array`

- Description: Whether parse multidimensional array, if false encountering will return ERROR
- Default value: true

`enable_simdjson_reader`

- Description: Whether enable simdjson to parse json while stream load
- Default value: true

`enable_query_memory_overcommit`

- Description: If true, when the process does not exceed the soft mem limit, the query memory will not be limited; when the process memory exceeds the soft mem limit, the query with the largest ratio between the currently used memory and the exec_mem_limit will be canceled. If false, cancel query when the memory used exceeds exec_mem_limit.
- Default value: true

`user_files_secure_path`

- Description: The storage directory for files queried by `local` table valued functions.
- Default value: `${DORIS_HOME}`

`brpc_streaming_client_batch_bytes`

- Description: The batch size for sending data by brpc streaming client
- Default value: 262144

`grace_shutdown_wait_seconds`

- Description: In cloud native deployment scenario, BE will be add to cluster and remove from cluster very frequently. User's query will fail if there is a fragment is running on the shuting down BE. Users could use stop_be.sh –grace, then BE will wait all running queries to stop to avoiding running query failure, but if the waiting time exceed the limit, then be will exit directly. During this period, FE will not send any queries to BE and waiting for all running queries to stop.
- Default value: 120

`enable_java_support`

- Description: BE Whether to enable the use of java-jni. When enabled, mutual calls between c++ and java are allowed. Currently supports hudi, java-udf, jdbc, max-compute, paimon, preload, avro
- Default value: true

`default_tzfiles_path`

- Description: Doris comes with its own time zone database. If the time zone file is not found in the system directory, the data in that directory is enabled.
- Default: “${DORIS_HOME}/zoneinfo”

User Property

User configuration item

This document mainly introduces related configuration items at the User level. The configuration of the User level is mainly effective for a single user. Each user can set their own User property. Does not affect each other.

View configuration items

After the FE is started, on the MySQL client, use the following command to view the User configuration items:

```
SHOW PROPERTY [FOR user] [LIKE key pattern]
```

The specific syntax can be queried through the command: `help show property;`.

Set configuration items

After FE is started, on the MySQL client, modify the User configuration items with the following command:

```
SET PROPERTY [FOR'user'] 'key' = 'value' [,'key' ='value']
```

The specific syntax can be queried through the command: `help set property;`.

User-level configuration items will only take effect for the specified users, and will not affect the configuration of other users.

Application examples

1. Modify the max_user_connections of user Billie

   Use `SHOW PROPERTY FOR 'Billie' LIKE '%max_user_connections%';` to check that the current maximum number of links for Billie users is 100.

   Use `SET PROPERTY FOR 'Billie' 'max_user_connections' = '200';` to modify the current maximum number of connections for Billie users to 200.

Configuration item list

max_user_connections

```
The maximum number of user connections, the default value is 100 In general, this parameter does not need to be changed unless the number of concurrent queries exceeds the default value.
```

max_query_instances

The maximum number of instances that the user can use at a certain point in time, The default value is -1, negative number means use default_max_query_instances config.

resource

quota

default_load_cluster

load_cluster

Audit Log Plugin

Doris's audit log plugin was developed based on FE's plugin framework. Is an optional plugin. Users can install or uninstall this plugin at runtime.

This plugin can periodically import the FE audit log into the specified Doris cluster, so that users can easily view and analyze the audit log through SQL.

Compile, Configure and Deploy

FE Configuration

The audit log plug-in framework is enabled by default in Doris and is controlled by the FE configuration `plugin_enable`

AuditLoader Configuration

1. Download the Audit Loader plugin

   The Audit Loader plug-in is provided by default in the Doris distribution. After downloading the Doris installation package through DOWNLOAD, decompress it and enter the directory, you can find the auditloader.zip file in the extensionsaudit_loader subdirectory.

2. Unzip the installation package

   ```
   unzip auditloader.zip
   ```

   Unzip and generate the following files:

   - auditloader.jar: plug-in code package.
   - plugin.properties: plugin properties file.
   - plugin.conf: plugin configuration file.

You can place this file on an http download server or copy(or unzip) it to the specified directory of all FEs. Here we use the latter.
The installation of this plugin can be found in INSTALL
After executing install, the AuditLoader directory will be automatically generated.

3. Modify plugin.conf

   The following configurations are available for modification:

   - frontend_host_port: FE node IP address and HTTP port in the format :. The default value is 127.0.0.1:8030.
   - database: Audit log database name.
   - audit_log_table: Audit log table name.
   - slow_log_table: Slow query log table name.

- enable_slow_log: Whether to enable the slow query log import function. The default value is false.
- user: Cluster username. The user must have INSERT permission on the corresponding table.
- password: Cluster user password.

创建库表

In Doris, you need to create the library and table of the audit log. The table structure is as follows:

If you need to enable the slow query log import function, you need to create an additional slow table `doris_slow_log_tbl__`, whose table structure is consistent with `doris_audit_log_tbl__`.

Among them, the `dynamic_partition` attribute selects the number of days for audit log retention according to your own needs.

```
create database doris_audit_db__;

create table doris_audit_db__.doris_audit_log_tbl__
(
    query_id varchar(48) comment "Unique query id",
    `time` datetime not null comment "Query start time",
    client_ip varchar(32) comment "Client IP",
    user varchar(64) comment "User name",
    catalog varchar(128) comment "Catalog of this query",
    db varchar(96) comment "Database of this query",
    state varchar(8) comment "Query result state. EOF, ERR, OK",
    error_code int comment "Error code of failing query.",
    error_message string comment "Error message of failing query.",
    query_time bigint comment "Query execution time in millisecond",
    scan_bytes bigint comment "Total scan bytes of this query",
    scan_rows bigint comment "Total scan rows of this query",
    return_rows bigint comment "Returned rows of this query",
    stmt_id int comment "An incremental id of statement",
    is_query tinyint comment "Is this statemt a query. 1 or 0",
    frontend_ip varchar(32) comment "Frontend ip of executing this statement",
    cpu_time_ms bigint comment "Total scan cpu time in millisecond of this query",
    sql_hash varchar(48) comment "Hash value for this query",
    sql_digest varchar(48) comment "Sql digest for this query",
    peak_memory_bytes bigint comment "Peak memory bytes used on all backends of this query",
    stmt string comment "The original statement, trimed if longer than 2G"
) engine=OLAP
duplicate key(query_id, `time`, client_ip)
partition by range(`time`) ()
distributed by hash(query_id) buckets 1
properties(
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.start" = "-30",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "1",
    "dynamic_partition.enable" = "true",
    "replication_num" = "3"
);

create table doris_audit_db__.doris_slow_log_tbl__
(
    query_id varchar(48) comment "Unique query id",
    `time` datetime not null comment "Query start time",
    client_ip varchar(32) comment "Client IP",
    user varchar(64) comment "User name",
    catalog varchar(128) comment "Catalog of this query",
    db varchar(96) comment "Database of this query",
```

```
    state varchar(8) comment "Query result state. EOF, ERR, OK",
    error_code int comment "Error code of failing query.",
    error_message string comment "Error message of failing query.",
    query_time bigint comment "Query execution time in millisecond",
    scan_bytes bigint comment "Total scan bytes of this query",
    scan_rows bigint comment "Total scan rows of this query",
    return_rows bigint comment "Returned rows of this query",
    stmt_id int comment "An incremental id of statement",
    is_query tinyint comment "Is this statemt a query. 1 or 0",
    frontend_ip varchar(32) comment "Frontend ip of executing this statement",
    cpu_time_ms bigint comment "Total scan cpu time in millisecond of this query",
    sql_hash varchar(48) comment "Hash value for this query",
    sql_digest varchar(48) comment "Sql digest for this query",
    peak_memory_bytes bigint comment "Peak memory bytes used on all backends of this query",
    stmt string comment "The original statement, trimed if longer than 2G "
) engine=OLAP
duplicate key(query_id, `time`, client_ip)
partition by range(`time`) ()
distributed by hash(query_id) buckets 1
properties(
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.start" = "-30",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "1",
    "dynamic_partition.enable" = "true",
    "replication_num" = "3"
);
```

> **In the above table structure: stmt string, this can only be used in 0.15 and later versions, in previous versions, the field type used varchar**

Deployment

You can place the packaged auditloader.zip on an http server, or copy `auditloader.zip` to the same specified directory in all FEs.

Installation

通过以下语句安装 Audit Loader 插件：

```
INSTALL PLUGIN FROM [source] [PROPERTIES ("key"="value", ...)]
```

Detailed command reference: INSTALL

After successful installation, you can see the installed plug-ins through `SHOW PLUGINS`, and the status is `INSTALLED`.

After completion, the plugin will continuously insert audit logs into this table at specified intervals.

Config Action

Config Action

Request

```
GET /rest/v1/config/fe/
```

Description

Config Action is used to obtain current FE configuration information.

Path parameters

None

Query parameters

- `conf_item`

  Optional parameters. Return specified item in FE configuration.

Request body

None

Response


```
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": ["Name", "Value"],
        "rows": [{
            "Value": "DAY",
            "Name": "sys_log_roll_interval"
        }, {
            "Value": "23",
            "Name": "consistency_check_start_time"
        }, {
            "Value": "4096",
            "Name": "max_mysql_service_task_threads_num"
        }, {
            "Value": "1000",
            "Name": "max_unfinished_load_job"
        }, {
            "Value": "100",
            "Name": "max_routine_load_job_num"
        }, {
```

```
            "Value": "SYNC",
            "Name": "master_sync_policy"
        }]
    },
    "count": 0
}
```

The returned result is the same as `System Action`. Is a description of the table.

HA Action

HA Action

Request

```
GET /rest/v1/ha
```

Description

HA Action is used to obtain the high availability group information of the FE cluster.

Path parameters

None

Query parameters

None

Request body

None

Response


```
{
    "msg": "success",
    "code": 0,
    "data": {
        "Observernodes": [],
        "CurrentJournalId": [{
            "Value": 433648,
            "Name": "FrontendRole"
        }],
        "Electablenodes": [{
            "Value": "host1",
            "Name": "host1"
        }],
        "allowedFrontends": [{
            "Value": "name: 192.168.1.1_9213_1597652404352, role: FOLLOWER, 192.168.1.1:9213",
            "Name": "192.168.1.1_9213_1597652404352"
        }],
        "removedFrontends": [],
        "CanRead": [{
            "Value": true,
            "Name": "Status"
        }],
        "databaseNames": [{
```

```
            "Value": "433436 ",
            "Name": "DatabaseNames"
        }],
        "FrontendRole": [{
            "Value": "MASTER",
            "Name": "FrontendRole"
        }],
        "CheckpointInfo": [{
            "Value": 433435,
            "Name": "Version"
        }, {
            "Value": "2020-09-03T02:07:37.000+0000",
            "Name": "lastCheckPointTime"
        }]
    },
    "count": 0
}
```

Hardware Info Action

Hardware Info Action

Request

```
GET /rest/v1/hardware_info/fe/
```

Description

Hardware Info Action is used to obtain the hardware information of the current FE.

Path parameters

None

Query parameters

None

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "VersionInfo": {
            "Git": "git://host/core@5bc28f4c36c20c7b424792df662fc988436e679e",
            "Version": "trunk",
            "BuildInfo": "cmy@192.168.1",
            "BuildTime": "Tuesday, 05 September 2019 11:07:42 CST"
        },
        "HarewareInfo": {
            "NetworkParameter": "...",
            "Processor": "...",
            "OS": "...",
            "Memory": "...",
            "FileSystem": "...",
            "NetworkInterface": "...",
            "Processes": "...",
            "Disk": "..."
        }
    },
    "count": 0
}
```

- The contents of each value in the `HarewareInfo` field are all hardware information text displayed in html format.

Help Action

Help Action

Request

`GET /rest/v1/help`

Description

Used to obtain help through fuzzy query.

Path parameters

None

Query parameters

- `query`

  Keywords to be matched, such as array and select.

Request body

None

Response

```
{
    "msg":"success",
    "code":0,
    "data":{"fuzzy":"No Fuzzy Matching Topic","matching":"No Matching Category"},
    "count":0
}
```

Log Action

Log Action

Request

```
GET /rest/v1/log
```

Description

GET is used to obtain the latest part of Doris's WARNING log, and the POST method is used to dynamically set the log level of FE.

Path parameters

None

Query parameters

- `add_verbose`

  Optional parameters for the POST method. Enable the DEBUG level log of the specified package.

- `del_verbose`

  Optional parameters for the POST method. Turn off the DEBUG level log of the specified package.

Request body

None

Response

```
GET /rest/v1/log

{
    "msg": "success",
    "code": 0,
    "data": {
        "LogContents": {
            "logPath": "/home/disk1/cmy/git/doris/core-for-ui/output/fe/log/fe.warn.log",
            "log": "<pre>2020-08-26 15:54:30,081 WARN (UNKNOWN 10.81.85.89_9213_1597652404352(-1)|1)
[Catalog.notifyNewFETypeTransfer():2356] notify new FE type transfer: UNKNOWN</br>2020-08-26 15:54:32,089 WARN (RepNode
10.81.85.89_9213_1597652404352(-1)|61) [Catalog.notifyNewFETypeTransfer():2356] notify new FE type transfer: MASTER</br>2020-08-26
15:54:35,121 WARN (stateListener|73) [Catalog.replayJournal():2510] replay journal cost too much time: 2975 replayedJournalId:
232383</br>2020-08-26 15:54:48,117 WARN (leaderCheckpointer|75) [Catalog.replayJournal():2510] replay journal cost too much time:
2812 replayedJournalId: 232383</br></pre>",
            "showingLast": "603 bytes of log"
```

```
        },
        "LogConfiguration": {
            "VerboseNames": "org",
            "AuditNames": "slow_query,query",
            "Level": "INFO"
        }
    },
    "count": 0
}
```

Among them, `data.LogContents.log` means the log content in the latest part of `fe.warn.log`.

```
POST /rest/v1/log?add_verbose=org

{
    "msg": "success",
    "code": 0,
    "data": {
        "LogConfiguration": {
            "VerboseNames": "org",
            "AuditNames": "slow_query,query",
            "Level": "INFO"
        }
    },
    "count": 0
}
```

Login Action

Login Action

Request

`POST /rest/v1/login`

Description

Used to log in to the service.

Path parameters

None

Query parameters

None

Request body

None

Response

- Login success

```
{
    "msg": "Login success!",
    "code": 200
}
```

- Login failure

```
{
    "msg": "Error msg...",
    "code": xxx,
    "data": "Error data...",
    "count": 0
}
```

Logout Action

Logout Action

Request

```
POST /rest/v1/logout
```

Description

Logout Action is used to log out of the current login.

Path parameters

None

Query parameters

None

Request body

None

Response

```
{
    "msg": "OK",
    "code": 0
}
```

Query Profile Action

Query Profile Action

Request

```
GET /rest/v1/query_profile/<query_id>
```

Description

The Query Profile Action is used to obtain the Query profile.

Path parameters

- <query_id>

  Optional parameters. When not specified, the latest query list is returned. When specified, return the profile of the specified query.

Query parameters

None

Request body

None

Response

- Not specify <query_id>

```
GET /rest/v1/query_profile/
{
    "msg": "success",
    "code": 0,
    "data": {
        "href_column": ["Query ID"],
        "column_names": ["Query ID", "User", "Default Db", "Sql Statement", "Query Type", "Start Time", "End Time",
"Total", "Query State"],
        "rows": [{
            "User": "root",
            "__hrefPath": ["/query_profile/d73a8a0b004f4b2f-b4829306441913da"],
            "Query Type": "Query",
            "Total": "5ms",
            "Default Db": "default_cluster:db1",
            "Sql Statement": "select * from tbl1",
            "Query ID": "d73a8a0b004f4b2f-b4829306441913da",
            "Start Time": "2020-09-03 10:07:54",
            "Query State": "EOF",
```

```
            "End Time": "2020-09-03 10:07:54"
        }, {
            "User": "root",
            "__hrefPath": ["/query_profile/fd706dd066824c21-9d1a63af9f5cb50c"],
            "Query Type": "Query",
            "Total": "6ms",
            "Default Db": "default_cluster:db1",
            "Sql Statement": "select * from tbl1",
            "Query ID": "fd706dd066824c21-9d1a63af9f5cb50c",
            "Start Time": "2020-09-03 10:07:54",
            "Query State": "EOF",
            "End Time": "2020-09-03 10:07:54"
        }]
    },
    "count": 3
}
```

The returned result is the same as `System Action`, which is a table description.

- Specify `<query_id>`

```
GET /rest/v1/query_profile/<query_id>

{
    "msg": "success",
    "code": 0,
    "data": "Query:</br>&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;Summary:</br>...",
    "count": 0
}
```

`data` is the text content of the profile.

Session Action

Session Action

Request

`GET /rest/v1/session`

`GET /rest/v1/session/all`

Description

Session Action is used to obtain the current session information.

Path parameters

None

Query parameters

None

Request body

None

Obtain the current session information

`GET /rest/v1/session`

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": ["Id", "User", "Host", "Cluster", "Db", "Command", "Time", "State", "Info"],
        "rows": [{
            "User": "root",
            "Command": "Sleep",
            "State": "",
            "Cluster": "default_cluster",
            "Host": "10.81.85.89:31465",
            "Time": "230",
            "Id": "0",
            "Info": "",
            "Db": "db1"
        }]
    },
```

```
    "count": 2
}
```

Obtain all FE session information

```
GET /rest/v1/session/all
```

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": ["FE", "Id", "User", "Host", "Cluster", "Db", "Command", "Time", "State", "Info"],
        "rows": [{
            "FE": "10.14.170.23",
            "User": "root",
            "Command": "Sleep",
            "State": "",
            "Cluster": "default_cluster",
            "Host": "10.81.85.89:31465",
            "Time": "230",
            "Id": "0",
            "Info": "",
            "Db": "db1"
        },
        {
            "FE": "10.14.170.24",
            "User": "root",
            "Command": "Sleep",
            "State": "",
            "Cluster": "default_cluster",
            "Host": "10.81.85.88:61465",
            "Time": "460",
            "Id": "1",
            "Info": "",
            "Db": "db1"
        }]
    },
    "count": 2
}
```

The returned result is the same as `System Action`. Is a description of the table.

System Action

System Action

Request

```
GET /rest/v1/system
```

Description

System Action is used for information about the Proc system built in Doris.

Path parameters

None

Query parameters

- `path`

  Optional parameter, specify the path of proc

Request body

None

Response

Take `/dbs/10003/10054/partitions/10053/10055` as an example:


```
{
    "msg": "success",
    "code": 0,
    "data": {
        "href_columns": ["TabletId", "MetaUrl", "CompactionStatus"],
        "column_names": ["TabletId", "ReplicaId", "BackendId", "SchemaHash", "Version", "VersionHash", "LstSuccessVersion",
"LstSuccessVersionHash", "LstFailedVersion", "LstFailedVersionHash", "LstFailedTime", "DataSize", "RowCount", "State",
"LstConsistencyCheckTime", "CheckVersion", "CheckVersionHash", "VersionCount", "PathHash", "MetaUrl", "CompactionStatus"],
        "rows": [{
            "SchemaHash": "1294206575",
            "LstFailedTime": "\\N",
            "LstFailedVersion": "-1",
            "MetaUrl": "URL",
            "__hrefPaths": ["http://192.168.100.100:8030/rest/v1/system?path=/dbs/10003/10054/partitions/10053/10055/10056",
"http://192.168.100.100:8043/api/meta/header/10056", "http://192.168.100.100:8043/api/compaction/show?tablet_id=10056"],
            "CheckVersionHash": "-1",
            "ReplicaId": "10057",
            "VersionHash": "4611804212003004639",
            "LstConsistencyCheckTime": "\\N",
```

```
            "LstSuccessVersionHash": "4611804212003004639",
            "CheckVersion": "-1",
            "Version": "6",
            "VersionCount": "2",
            "State": "NORMAL",
            "BackendId": "10032",
            "DataSize": "776",
            "LstFailedVersionHash": "0",
            "LstSuccessVersion": "6",
            "CompactionStatus": "URL",
            "TabletId": "10056",
            "PathHash": "-3259732870068082628",
            "RowCount": "21"
        }]
    },
    "count": 1
}
```

The `column_names` in the data part is the header information, and `href_columns` indicates which columns in the table are hyperlink columns. Each element in the `rows` array represents a row. Among them, `__hrefPaths` is not the table data, but the link URL of the hyperlink column, which corresponds to the column in `href_columns` one by one.

Colocate Meta Action

Colocate Meta Action

Request

`GET /api/colocate` `POST/DELETE /api/colocate/group_stable` `POST /api/colocate/bucketseq`

Description

Used to obtain or modify colocate group information.

Path parameters

None

Query parameters

- `db_id`

  Specify database id

- `group_id`

  Specify group id

Request body

None

Response

TO DO

Meta Action

Meta Action

Request

```
GET /image
GET /info
GET /version
GET /put
GET /journal_id
GET /role
GET /check
GET /dump
```

Description

This is a set of APIs related to FE metadata, except for `/dump`, they are all used for internal communication between FE nodes.

Path parameters

TODO

Query parameters

TODO

Request body

TODO

Response

TODO

Examples

TODO

Cluster Action

Cluster Action

Request

`GET /rest/v2/manager/cluster/cluster_info/conn_info`

Cluster Connection Information

`GET /rest/v2/manager/cluster/cluster_info/conn_info`

Description

Used to get cluster http, mysql connection information.

Path parameters

None

Query parameters

None

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "http": [
            "fe_host:http_ip"
        ],
        "mysql": [
            "fe_host:query_ip"
        ]
    },
    "count": 0
}
```

Examples

```
GET /rest/v2/manager/cluster/cluster_info/conn_info

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "http": [
            "127.0.0.1:8030"
        ],
        "mysql": [
            "127.0.0.1:9030"
        ]
    },
    "count": 0
}
```

Node Action

Request

```
GET /rest/v2/manager/node/frontends

GET /rest/v2/manager/node/backends

GET /rest/v2/manager/node/brokers

GET /rest/v2/manager/node/configuration_name

GET /rest/v2/manager/node/node_list

POST /rest/v2/manager/node/configuration_info

POST /rest/v2/manager/node/set_config/fe

POST /rest/v2/manager/node/set_config/be

POST /rest/v2/manager/node/{action}/be

POST /rest/v2/manager/node/{action}/fe
```

Get information about fe, be, broker nodes

```
GET /rest/v2/manager/node/frontends

GET /rest/v2/manager/node/backends

GET /rest/v2/manager/node/brokers
```

Description

Used to get cluster to get fe, be, broker node information.

Response

```
frontends:
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": [
            "Name",
            "IP",
            "HostName",
            "EditLogPort",
            "HttpPort",
            "QueryPort",
            "RpcPort",
            "ArrowFlightSqlPort",
```

```
            "Role",
            "IsMaster",
            "ClusterId",
            "Join",
            "Alive",
            "ReplayedJournalId",
            "LastHeartbeat",
            "IsHelper",
            "ErrMsg",
            "Version"
        ],
        "rows": [
            [
                ...
            ]
        ]
    },
    "count": 0
}
```

```
backends:
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": [
            "BackendId",
            "Cluster",
            "IP",
            "HostName",
            "HeartbeatPort",
            "BePort",
            "HttpPort",
            "BrpcPort",
            "LastStartTime",
            "LastHeartbeat",
            "Alive",
            "SystemDecommissioned",
            "ClusterDecommissioned",
            "TabletNum",
            "DataUsedCapacity",
            "AvailCapacity",
            "TotalCapacity",
            "UsedPct",
            "MaxDiskUsedPct",
            "ErrMsg",
            "Version",
            "Status"
        ],
        "rows": [
            [
                ...
            ]
        ]
    },
    "count": 0
}
```

```
brokers:
{
    "msg": "success",
    "code": 0,
```

```
    "data": {
        "column_names": [
            "Name",
            "IP",
            "HostName",
            "Port",
            "Alive",
            "LastStartTime",
            "LastUpdateTime",
            "ErrMsg"
        ],
        "rows": [
            [
                ...
            ]
        ]
    },
    "count": 0
}
```

Get node configuration information

`GET /rest/v2/manager/node/configuration_name`

`GET /rest/v2/manager/node/node_list`

`POST /rest/v2/manager/node/configuration_info`

Description

configuration_name Used to get the name of the node configuration item.
node_list Get the list of nodes.
configuration_info to get the node configuration details.

Query parameters

`GET /rest/v2/manager/node/configuration_name`
none

`GET /rest/v2/manager/node/node_list`
none

`POST /rest/v2/manager/node/configuration_info`

- type The value is fe or be, which specifies to get the configuration information of fe or the configuration information of be.

Request body

`GET /rest/v2/manager/node/configuration_name`
none

`GET /rest/v2/manager/node/node_list`
none

POST /rest/v2/manager/node/configuration_info

```
{
    "conf_name": [
        ""
    ],
    "node": [
        ""
    ]
}

If no body is included, the parameters in the body use the default values.
conf_name specifies which configuration items to return, the default is all configuration items.
node is used to specify which node's configuration information is returned, the default is all fe nodes or be nodes configuration
information.
```

Response

GET /rest/v2/manager/node/configuration_name

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "backend":[
            ""
        ],
        "frontend":[
            ""
        ]
    },
    "count": 0
}
```

GET /rest/v2/manager/node/node_list

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "backend": [
            ""
        ],
        "frontend": [
            ""
        ]
    },
    "count": 0
}
```

POST /rest/v2/manager/node/configuration_info?type=fe

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": [
            "Configuration Item",
```

```
            "Node",
            "Node Type",
            "Configuration Value Type",
            "MasterOnly",
            "Configuration Value",
            "Modifiable"
        ],
        "rows": [
            [
                ""
            ]
        ]
    },
    "count": 0
}
```

POST /rest/v2/manager/node/configuration_info?type=be

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": [
            "Configuration Item",
            "Node",
            "Node Type",
            "Configuration Value Type",
            "Configuration Value",
            "Modifiable"
        ],
        "rows": [
            [
                ""
            ]
        ]
    },
    "count": 0
}
```

Examples

1. Get the fe agent_task_resend_wait_time_ms configuration information:

   POST /rest/v2/manager/node/configuration_info?type=fe
   body:

   ```
   {
       "conf_name":[
           "agent_task_resend_wait_time_ms"
       ]
   }
   ```

   Response:

   ```
   {
   "msg": "success",
   "code": 0,
   "data": {
   ```

```
    "column_names": [
        "Configuration Item",
        "Node",
        "Node Type",
        "Configuration Value Type",
        "MasterOnly",
        "Configuration Value",
        "Modifiable"
    ],
    "rows": [
        [
            "agent_task_resend_wait_time_ms",
            "127.0.0.1:8030",
            "FE",
            "long",
            "true",
            "50000",
            "true"
        ]
    ]
},
"count": 0
}
```

Modify configuration values

POST /rest/v2/manager/node/set_config/fe

POST /rest/v2/manager/node/set_config/be

Description

Used to modify fe or be node configuration values

Request body

```
{
    "config_name":{
        "node":[
            ""
        ],
        "value":"",
        "persist":
    }
}

config_name is the corresponding configuration item.
node is a keyword indicating the list of nodes to be modified;
value is the value of the configuration.
persist is true for permanent modification and false for temporary modification. persist means permanent modification, false means
temporary modification. permanent modification takes effect after reboot, temporary modification fails after reboot.
```

Response

GET /rest/v2/manager/node/configuration_name

```
{
    "msg": "",
    "code": 0,
    "data": {
        "failed":[
            {
                "config_name":"name",
                "value"="",
                "node":"",
                "err_info":""
            }
        ]
    },
    "count": 0
}

failed Indicates a configuration message that failed to be modified.
```

Examples

1. Modify the agent_task_resend_wait_time_ms and alter_table_timeout_second configuration values in the fe 127.0.0.1:8030 node:

   POST /rest/v2/manager/node/set_config/fe body:

   ```
   {
       "agent_task_resend_wait_time_ms":{
           "node":[
               "127.0.0.1:8030"
           ],
           "value":"10000",
           "persist":"true"
       },
       "alter_table_timeout_second":{
           "node":[
               "127.0.0.1:8030"
           ],
           "value":"true",
           "persist":"true"
       }
   }
   ```

   Response: ``` { “msg”: “success”, “code”: 0, “data”: { “failed”: [{ “config_name”: “alter_table_timeout_second”, “node”: “10.81.85.89:8837”, “err_info”: “Unsupported configuration value type.”, “value”: “true” }] }, “count”: 0 }

   gent_task_resend_wait_time_ms configuration value modified successfully, alter_table_timeout_second modification failed. ```

Operate be node

```
POST /rest/v2/manager/node/{action}/be
```

Description

Used to add/drop/offline be node

action：ADD/DROP/DECOMMISSION

Request body

```
{
    "hostPorts": ["127.0.0.1:9050"],
    "properties": {
        "tag.location": "test"
    }
}

hostPorts A set of be node addresses to be operated, ip:heartbeat_port
properties The configuration passed in when adding a node is only used to configure the tag. If not, the default tag is used
```

Response

```
{
    "msg": "Error",
    "code": 1,
    "data": "errCode = 2, detailMessage = Same backend already exists[127.0.0.1:9050]",
    "count": 0
}

msg Success/Error
code 0/1
data ""/Error message
```

Examples

1. add be node

   post /rest/v2/manager/node/ADD/be Request body

   ```
   {
       "hostPorts": ["127.0.0.1:9050"]
   }
   ```

   Response

   ```
   {
       "msg": "success",
       "code": 0,
       "data": null,
       "count": 0
   }
   ```

2. drop be node

   post /rest/v2/manager/node/DROP/be Request body

```
{
    "hostPorts": ["127.0.0.1:9050"]
}
```

Response

```
{
    "msg": "success",
    "code": 0,
    "data": null,
    "count": 0
}
```

3. offline be node

   post /rest/v2/manager/node/DECOMMISSION/be Request body `json { "hostPorts": ["127.0.0.1:9050"] }`

   Response `json { "msg": "success", "code": 0, "data": null, "count": 0 }`

Operate fe node

`POST /rest/v2/manager/node/{action}/fe`

Description

Used to add/drop fe node

action：ADD/DROP

Request body

```
{
    "role": "FOLLOWER",
    "hostPort": "127.0.0.1:9030"
}

role FOLLOWER/OBSERVER
hostPort The address of the fe node to be operated, ip:edit_log_port
```

Response

```
{
    "msg": "Error",
    "code": 1,
    "data": "errCode = 2, detailMessage = frontend already exists name: 127.0.0.1:9030_1670495889415, role: FOLLOWER,
127.0.0.1:9030",
    "count": 0
}

msg Success/Error
```

```
code 0/1
data ""/Error message
```

Examples

1. add FOLLOWER node

   post /rest/v2/manager/node/ADD/fe Request body `json { "role": "FOLLOWER", "hostPort": "127.0.0.1:9030" }`

   Response `json { "msg": "success", "code": 0, "data": null, "count": 0 }`

2. drop FOLLOWER node

   post /rest/v2/manager/node/DROP/fe Request body `json { "role": "FOLLOWER", "hostPort": "127.0.0.1:9030" }`

   Response `json { "msg": "success", "code": 0, "data": null, "count": 0 }`

Query Profile Action

Query Profile Action

Request

```
GET /rest/v2/manager/query/query_info

GET /rest/v2/manager/query/trace/{trace_id}

GET /rest/v2/manager/query/sql/{query_id}

GET /rest/v2/manager/query/profile/text/{query_id}

GET /rest/v2/manager/query/profile/graph/{query_id}

GET /rest/v2/manager/query/profile/json/{query_id}

GET /rest/v2/manager/query/profile/fragments/{query_id}

GET /rest/v2/manager/query/current_queries

GET /rest/v2/manager/query/kill/{query_id}
```

Get the query information

```
GET /rest/v2/manager/query/query_info
```

Description

Gets information about select queries for all fe nodes in the cluster.

Query parameters

- `query_id`

  Optional, specifies the query ID of the query to be returned, default returns information for all queries.

- `search`

  Optional, specifies that query information containing strings is returned, currently only string matches are performed.

- `is_all_node`

  Optional, if true, returns query information for all fe nodes, if false, returns query information for the current fe node. The default is true.

Response

```
{
   "msg": "success",
    "code": 0,
    "data": {
        "column_names": [
            "Query ID",
            "FE Node",
            "Query User",
            "Execution Database",
            "Sql",
            "Query Type",
            "Start Time",
            "End Time",
            "Execution Duration",
            "Status"
        ],
        "rows": [
            [
                ...
            ]
        ]
    },
    "count": 0
}
```

> **Note**
>
> Since Doris Version 1.2, Admin and Root users can view all queries. Regular users can only view their own submitted queries.

Examples

```
GET /rest/v2/manager/query/query_info

{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": [
            "Query ID",
            "FE Node",
            "Query User",
            "Execution Database",
            "Sql",
            "Query Type",
            "Start Time",
            "End Time",
            "Execution Duration",
            "Status"
        ],
        "rows": [
            [
                "d7c93d9275334c35-9e6ac5f295a7134b",
                "127.0.0.1:8030",
                "root",
```

```
                "default_cluster:testdb",
                "select c.id, c.name, p.age, p.phone, c.date, c.cost from cost c join people p on c.id = p.id where p.age > 20 order
by c.id",
                "Query",
                "2021-07-29 16:59:12",
                "2021-07-29 16:59:12",
                "109ms",
                "EOF"
            ]
        ]
    },
    "count": 0
}
```

Get Query Id By Trace Id

`GET /rest/v2/manager/query/trace_id/{trace_id}`

Description

Get query id by trance id.

Before executing a Query, set a unique trace id:

```
set session_context="trace_id:your_trace_id";
```

After executing the Query within the same Session, the query id can be obtained through the trace id.

Path parameters

- `{trace_id}`

  User specific trace id.

Query parameters

Response

```
{
    "msg": "success",
    "code": 0,
    "data": "fb1d9737de914af1-a498d5c5dec638d3",
    "count": 0
}
```

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Admin and Root user can view all queries. Ordinary users can only view the Query sent by themselves. If the specified trace id does not exist or has no permission, it will return Bad Request:

```
{
    "msg": "Bad Request",
    "code": 403,
    "data": "error messages",
    "count": 0
}
```

Get the sql and text profile for the specified query

`GET /rest/v2/manager/query/sql/{query_id}`

`GET /rest/v2/manager/query/profile/text/{query_id}`

Description

Get the sql and profile text for the specified query id.

Path parameters

- `query_id`

  The query id.

Query parameters

- `is_all_node`

  Optional, if true then query for the specified query id in all fe nodes, if false then query for the specified query id in the currently connected fe nodes. The default is true.

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "sql": ""
    },
    "count": 0
}
```

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "profile": ""
    },
    "count": 0
}
```

> **Tips**

> This feature is supported since the Apache Doris 1.2 version

Admin and Root user can view all queries. Ordinary users can only view the Query sent by themselves. If the specified trace id does not exist or has no permission, it will return Bad Request:

```
{
    "msg": "Bad Request",
    "code": 403,
    "data": "error messages",
    "count": 0
}
```

Examples

1. get sql.

```
GET /rest/v2/manager/query/sql/d7c93d9275334c35-9e6ac5f295a7134b

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "sql": "select c.id, c.name, p.age, p.phone, c.date, c.cost from cost c join people p on c.id = p.id where p.age >
20 order by c.id"
    },
    "count": 0
}
```

Get the specified query fragment and instance information

`GET /rest/v2/manager/query/profile/fragments/{query_id}`

Description

Get the fragment name, instance id, host ip/port and execution time for the specified query id.

Path parameters

- `query_id`

  The query id.

Query parameters

- `is_all_node`

  Optional, if true then query for the specified query id in all fe nodes, if false then query for the specified query id in the currently connected fe nodes. The default is true.

Response

```
{
    "msg": "success",
    "code": 0,
    "data": [
        {
            "fragment_id": "",
            "time": "",
            "instance_id": {
                "": {
                  "host": "",
                  "active_time": ""
                }
            }
        }
    ],
    "count": 0
}
```

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Admin and Root user can view all queries. Ordinary users can only view the Query sent by themselves. If the specified trace id does not exist or has no permission, it will return Bad Request:

```
{
    "msg": "Bad Request",
    "code": 403,
    "data": "error messages",
    "count": 0
}
```

Examples

```
GET /rest/v2/manager/query/profile/fragments/d7c93d9275334c35-9e6ac5f295a7134b

Response:
{
    "msg": "success",
    "code": 0,
    "data": [
        {
            "fragment_id": "0",
            "time": "36.169ms",
            "instance_id": {
                "d7c93d9275334c35-9e6ac5f295a7134e": {
                    "host": "172.19.0.4:9060",
                    "active_time": "36.169ms"
                }
            }
        },
        {
            "fragment_id": "1",
            "time": "20.710ms",
```

```
            "instance_id": {
                "d7c93d9275334c35-9e6ac5f295a7134c": {
                    "host": "172.19.0.5:9060",
                    "active_time": "20.710ms"
                }
            }
        },
        {
            "fragment_id": "2",
            "time": "7.83ms",
            "instance_id": {
                "d7c93d9275334c35-9e6ac5f295a7134d": {
                    "host": "172.19.0.6:9060",
                    "active_time": "7.83ms"
                },
                "d7c93d9275334c35-9e6ac5f295a7134f": {
                    "host": "172.19.0.7:9060",
                    "active_time": "10.873ms"
                }
            }
        }
    ],
    "count": 0
}
```


Get the specified query id tree profile information

`GET /rest/v2/manager/query/profile/graph/{query_id}`

Description

Get the tree profile information of the specified query id, same as `show query profile` command.

Path parameters

- `query_id`

  The query id.

Query parameters

- `fragment_id` and `instance_id`

  Optional, both parameters must be specified or not.
  If both are not specified, a simple tree of profiles is returned, equivalent to `show query profile '/query_id'`;
  If both are specified, a detailed profile tree is returned, which is equivalent to `show query profile '/query_id/fragment_id/instance_id'`.

- `is_all_node`

  Optional, if true then query information about the specified query id in all fe nodes, if false then query information about the specified query id in the currently connected fe nodes. The default is true.

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "graph":""
    },
    "count": 0
}
```

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Admin and Root user can view all queries. Ordinary users can only view the Query sent by themselves. If the specified trace id does not exist or has no permission, it will return Bad Request:

```
{
    "msg": "Bad Request",
    "code": 403,
    "data": "error messages",
    "count": 0
}
```

Current running queries

`GET /rest/v2/manager/query/current_queries`

Description

Same as `show proc "/current_query_stmts"`, return current running queries.

Path parameters

Query parameters

- `is_all_node`

  Optional. Return current running queries from all FE if set to true. Default is true.

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "columnNames": ["Frontend", "QueryId", "ConnectionId", "Database", "User", "ExecTime", "SqlHash", "Statement"],
        "rows": [
            ["172.19.0.3", "108e47ab438a4560-ab1651d16c036491", "2", "", "root", "6074", "1a35f62f4b14b9d7961b057b77c3102f", "select
```

```
sleep(60)"],
            ["172.19.0.11", "3606cad4e34b49c6-867bf6862cacc645", "3", "", "root", "9306", "1a35f62f4b14b9d7961b057b77c3102f",
"select sleep(60)"]
        ]
    },
    "count": 0
}
```


Cancel query

`POST /rest/v2/manager/query/kill/{query_id}`

Description

Cancel query of specified connection.

Path parameters

- `{query_id}`

  query id. You can get query id by `trance_id` api.

Query parameters

Response


```
{
    "msg": "success",
    "code": 0,
    "data": null,
    "count": 0
}
```

Backends Action

Backends Action

Request

```
GET /api/backends
```

Description

Backends Action returns the Backends list, including Backend's IP, PORT and other information.

Path parameters

None

Query parameters

- `is_alive`

  Optional parameters. Whether to return the surviving BE nodes. The default is false, which means that all BE nodes are returned.

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "backends": [
            {
                "ip": "192.1.1.1",
                "http_port": 8040,
                "is_alive": true
            }
        ]
    },
    "count": 0
}
```

Bootstrap Action

Bootstrap Action

Request

`GET /api/bootstrap`

Description

It is used to judge whether the FE has started. When no parameters are provided, only whether the startup is successful is returned. If `token` and `cluster_id` are provided, more detailed information is returned.

Path parameters

none

Query parameters

- `cluster_id`

  The cluster id. It can be viewed in the file `doris-meta/image/VERSION`.

- `token`

  Cluster token. It can be viewed in the file `doris-meta/image/VERSION`.

Request body

none

Response

- No parameters provided

  ```
  {
      "msg": "OK",
      "code": 0,
      "data": null,
      "count": 0
  }
  ```

  A code of 0 means that the FE node has started successfully. Error codes other than 0 indicate other errors.

- Provide `token` and `cluster_id`

```
{
    "msg": "OK",
    "code": 0,
    "data": {
        "queryPort": 9030,
        "rpcPort": 9020,
        "arrowFlightSqlPort": 9040,
        "maxReplayedJournal": 17287
    },
    "count": 0
}
```

- `queryPort` is the MySQL protocol port of the FE node.
- `rpcPort` is the thrift RPC port of the FE node.
- `maxReplayedJournal` represents the maximum metadata journal id currently played back by the FE node.
- `arrowFlightSqlPort` is the Arrow Flight SQL port of the FE node.

Examples

1. No parameters

```
GET /api/bootstrap

Response:
{
    "msg": "OK",
    "code": 0,
    "data": null,
    "count": 0
}
```

2. Provide `token` and `cluster_id`

```
GET /api/bootstrap?cluster_id=935437471&token=ad87f6dd-c93f-4880-bcdb-8ca8c9ab3031

Response:
{
    "msg": "OK",
    "code": 0,
    "data": {
        "queryPort": 9030,
        "rpcPort": 9020,
        "arrowFlightSqlPort": 9040,
        "maxReplayedJournal": 17287
    },
    "count": 0
}
```

Cancel Load Action

Cancel Load Action

Request

`POST /api/<db>/_cancel`

Description

Used to cancel the load transaction of the specified label. RETURN VALUES Return a JSON format string: Status: Success: cancel succeed Others: cancel failed Message: Error message if cancel failed

Path parameters

- `<db>`

  Specify the database name

Query parameters

- `<label>`

  Specify the load label

Request body

None

Response

- Cancel success

```
{
    "msg": "OK",
    "code": 0,
    "data": null,
    "count": 0
}
```

- Cancel failed

```
{
    "msg": "Error msg...",
    "code": 1,
    "data": null,
```

```
    "count": 0
}
```

Examples

1. Cancel the load transaction of the specified label

```
POST /api/example_db/_cancel?label=my_label1

Response:
{
    "msg": "OK",
    "code": 0,
    "data": null,
    "count": 0
}
```

Check Decommission Action

Check Decommission Action

Request

`GET /api/check_decommission`

Description

Used to determine whether the specified BE can be decommissioned. For example, after the node being decommissioned, whether the remaining nodes can meet the space requirements and the number of replicas.

Path parameters

None

Query parameters

- `host_ports`

  Specify one or more BEs, separated by commas. Such as: `ip1:port1,ip2:port2,....`

  Where port is the heartbeat port of BE.

Request body

None

Response

Return a list of nodes that can be decommissioned

```
{
    "msg": "OK",
    "code": 0,
    "data": ["192.168.10.11:9050", "192.168.10.11:9050"],
    "count": 0
}
```

Examples

1. Check whether the specified BE node can be decommissioned

   ```
   GET /api/check_decommission?host_ports=192.168.10.11:9050,192.168.10.11:9050
   ```

Response:


```
{
    "msg": "OK",
    "code": 0,
    "data": ["192.168.10.11:9050"],
    "count": 0
}
```

Check Storage Type Action

Check Storage Type Action

Request

`GET /api/_check_storagetype`

Description

It is used to check whether the storage format of the table under the specified database is the row storage format. (The row format is deprecated)

Path parameters

None

Query parameters

- `db`

  Specify the database

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "tbl2": {},
        "tbl1": {}
    },
    "count": 0
}
```

If there is content after the table name, the base or rollup table whose storage format is row storage will be displayed.

Examples

1. Check whether the storage format of the following table of the specified database is row format

```
GET /api/_check_storagetype

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "tbl2": {},
        "tbl1": {}
    },
    "count": 0
}
```

Connection Action

Connection Action

Request

`GET /api/connection`

Description

Given a connection id, return the query id that is currently being executed for this connection or the last execution completed.

The connection id can be viewed through the id column in the MySQL command `show processlist;`.

Path parameters

无

Query parameters

- `connection_id`

  Specified connection id

Request body

None

Response

```
{
    "msg": "OK",
    "code": 0,
    "data": {
        "query_id": "b52513ce3f0841ca-9cb4a96a268f2dba"
    },
    "count": 0
}
```

Examples

1. Get the query id of the specified connection id

   ```
   GET /api/connection?connection_id=101

   Response:
   ```

```
{
    "msg": "OK",
    "code": 0,
    "data": {
        "query_id": "b52513ce3f0841ca-9cb4a96a268f2dba"
    },
    "count": 0
}
```

Extra Basepath Action

Extra Basepath Action

Request

`GET /api/basepath`

Description

Used to obtain http basepath.

Path parameters

None

Query parameters

None

Request body

None

Response


```
{
    "msg":"success",
    "code":0,
    "data":{"enable":false,"path":""},
    "count":0
}
```

Fe Version Info Action

Fe Version Info Action

Request

`GET /api/fe_version_info`

Description

Get fe version info from fe host.

Path parameters

None.

Query parameters

None.

Request body

None.

Response

```
{
    "msg":"success",
    "code":0,
    "data":{
        "feVersionInfo":{
            "dorisBuildVersionPrefix":"doris",
            "dorisBuildVersionMajor":0,
            "dorisBuildVersionMinor":0,
            "dorisBuildVersionPatch":0,
            "dorisBuildVersionRcVersion":"trunk",
            "dorisBuildVersion":"doris-0.0.0-trunk",
            "dorisBuildHash":"git://4b7b503d1cb3/data/doris/doris/be/../@a04f9814fe5a09c0d9e9399fe71cc4d765f8bff1",
            "dorisBuildShortHash":"a04f981",
            "dorisBuildTime":"Fri, 09 Sep 2022 07:57:02 UTC",
            "dorisBuildInfo":"root@4b7b503d1cb3",
            "dorisJavaCompileVersion":"openjdk full version \"1.8.0_332-b09\""
        }
    },
    "count":0
}
```

Examples

```
GET /api/fe_version_info

Response:
{
    "msg":"success",
    "code":0,
    "data":{
        "feVersionInfo":{
            "dorisBuildVersionPrefix":"doris",
            "dorisBuildVersionMajor":0,
            "dorisBuildVersionMinor":0,
            "dorisBuildVersionPatch":0,
            "dorisBuildVersionRcVersion":"trunk",
            "dorisBuildVersion":"doris-0.0.0-trunk",
            "dorisBuildHash":"git://4b7b503d1cb3/data/doris/doris/be/../@a04f9814fe5a09c0d9e9399fe71cc4d765f8bff1",
            "dorisBuildShortHash":"a04f981",
            "dorisBuildTime":"Fri, 09 Sep 2022 07:57:02 UTC",
            "dorisBuildInfo":"root@4b7b503d1cb3",
            "dorisJavaCompileVersion":"openjdk full version \"1.8.0_332-b09\""
        }
    },
    "count":0
}
```

Get DDL Statement Action

Get DDL Statement Action

Request

`GET /api/_get_ddl`

Description

Used to get the table creation statement, partition creation statement and rollup statement of the specified table.

Path parameters

None

Query parameters

- `db`

  Specify database

- `table`

  Specify table

Request body

None

Response

```
{
    "msg": "OK",
    "code": 0,
    "data": {
        "create_partition": ["ALTER TABLE `tbl1` ADD PARTITION ..."],
        "create_table": ["CREATE TABLE `tbl1` ...],
        "create_rollup": ["ALTER TABLE `tbl1` ADD ROLLUP ..."]
    },
    "count": 0
}
```

Examples

1. Get the DDL statement of the specified table

```
GET GET /api/_get_ddl?db=db1&table=tbl1

Response
{
    "msg": "OK",
    "code": 0,
    "data": {
        "create_partition": [],
        "create_table": ["CREATE TABLE `tbl1` (\n  `k1` int(11) NULL COMMENT \"\",\n  `k2` int(11) NULL COMMENT \"\"\n)
ENGINE=OLAP\nDUPLICATE KEY(`k1`, `k2`)\nCOMMENT \"OLAP\"\nDISTRIBUTED BY HASH(`k1`) BUCKETS 1\nPROPERTIES
(\n\"replication_num\" = \"1\",\n\"version_info\" = \"1,0\",\n\"in_memory\" = \"false\",\n\"storage_format\" =
\"DEFAULT\"\n);"],
        "create_rollup": []
    },
    "count": 0
}
```

Get Load Info Action

Get Load Info Action

Request

`GET /api/<db>/_load_info`

Description

Used to obtain the information of the load job of the specified label.

Path parameters

- `<db>`

  Specify database

Query parameters

- `label`

  Specify load label

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "dbName": "default_cluster:db1",
        "tblNames": ["tbl1"],
        "label": "my_label",
        "clusterName": "default_cluster",
        "state": "FINISHED",
        "failMsg": "",
        "trackingUrl": ""
    },
    "count": 0
}
```

Examples

1. Get the load job information of the specified label

```
GET /api/example_db/_load_info?label=my_label

Response
{
    "msg": "success",
    "code": 0,
    "data": {
        "dbName": "default_cluster:db1",
        "tblNames": ["tbl1"],
        "label": "my_label",
        "clusterName": "default_cluster",
        "state": "FINISHED",
        "failMsg": "",
        "trackingUrl": ""
    },
    "count": 0
}
```

Get Load State

Get Load State

Request

`GET /api/<db>/get_load_state`

Description

Returns the status of the load transaction of the specified label Return of JSON format string of the status of specified transaction: Label: The specified label. Status: Success or not of this request. Message: Error messages State: UNKNOWN/PREPARE/COMMITTED/VISIBLE/ABORTED

Path parameters

- `<db>`

  Specify database

Query parameters

- `label`

  Specify label

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": "VISIBLE",
    "count": 0
}
```

If label does not exist, return:

```
{
    "msg": "success",
    "code": 0,
    "data": "UNKNOWN",
    "count": 0
}
```

Examples

1. Get the status of the load transaction of the specified label.

```
GET /api/example_db/get_load_state?label=my_label

{
    "msg": "success",
    "code": 0,
    "data": "VISIBLE",
    "count": 0
}
```

Get FE log file

Get FE log file

Request

`HEAD /api/get_log_file`

`GET /api/get_log_file`

Description

Users can obtain FE log files through the HTTP interface.

The HEAD request is used to obtain the log file list of the specified log type. GET request is used to download the specified log file.

Path parameters

None

Query parameters

- `type`

  Specify the log type. The following types are supported:

  - `fe.audit.log`: Audit log of Frontend.

- `file`

  Specify file name

Request body

None

Response

- `HEAD`

  ```
  HTTP/1.1 200 OK
  file_infos: {"fe.audit.log":24759,"fe.audit.log.20190528.1":132934}
  content-type: text/html
  connection: keep-alive
  ```

  The returned header lists all current log files of the specified type and the size of each file.

- GET

  Download the specified log file in text form

Examples

1. Get the log file list of the corresponding type

```
HEAD /api/get_log_file?type=fe.audit.log

Response:

HTTP/1.1 200 OK
file_infos: {"fe.audit.log":24759,"fe.audit.log.20190528.1":132934}
content-type: text/html
connection: keep-alive
```

   In the returned header, the `file_infos` field displays the file list and the corresponding file size (in bytes) in json format

2. Download log file

```
GET /api/get_log_file?type=fe.audit.log&file=fe.audit.log.20190528.1

Response:

< HTTP/1.1 200
< Vary: Origin
< Vary: Access-Control-Request-Method
< Vary: Access-Control-Request-Headers
< Content-Disposition: attachment;fileName=fe.audit.log
< Content-Type: application/octet-stream;charset=UTF-8
< Transfer-Encoding: chunked

... File Content ...
```

Get Small File Action

Get Small File

Request

`GET /api/get_small_file`

Description

Through the file id, download the file in the small file manager.

Path parameters

None

Query parameters

- `token`

  The token of the cluster. It can be viewed in the file `doris-meta/image/VERSION`.

- `file_id`

  The file id displayed in the file manager. The file id can be viewed with the `SHOW FILE` command.

Request body

None

Response

```
< HTTP/1.1 200
< Vary: Origin
< Vary: Access-Control-Request-Method
< Vary: Access-Control-Request-Headers
< Content-Disposition: attachment;fileName=ca.pem
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked

... File Content ...
```

If there is an error, it returns:

```
{
    "msg": "File not found or is not content",
    "code": 1,
    "data": null,
```

```
    "count": 0
}
```

Examples

1. Download the file with the specified id

```
GET /api/get_small_file?token=98e8c0a6-3a41-48b8-a72b-0432e42a7fe5&file_id=11002

Response:

< HTTP/1.1 200
< Vary: Origin
< Vary: Access-Control-Request-Method
< Vary: Access-Control-Request-Headers
< Content-Disposition: attachment;fileName=ca.pem
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked

... File Content ...
```

Health Action

Health Action

Request

`GET /api/health`

Description

Returns the number of BE nodes currently surviving in the cluster and the number of BE nodes that are down.

Path parameters

None

Query parameters

None

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "online_backend_num": 10,
        "total_backend_num": 10
    },
    "count": 0
}
```

Meta Info Action

Meta Action

Meta Info Action is used to obtain metadata information in the cluster. Such as database list, table structure, etc.

List Database

Request

```
GET /api/meta/namespaces/<ns_name>/databases
```

Description

Get a list of all database names, arranged in alphabetical order.

Path parameters

None

Query parameters

- `limit`

  Limit the number of result rows returned

- `offset`

  Pagination information, need to be used with `limit`

Request body

None

Response

```
{
    "msg": "OK",
    "code": 0,
    "data": [
       "db1", "db2", "db3", ...
    ],
    "count": 3
}
```

- The data field returns a list of database names.

List Table

Request

```
GET /api/meta/namespaces/<ns_name>/databases/<db_name>/tables
```

Description

Get a list of tables in the specified database, arranged in alphabetical order.

Path parameters

- `<db_name>`

  Specify database

Query parameters

- `limit`

  Limit the number of result rows returned

- `offset`

  Pagination information, need to be used with `limit`

Request body

None

Response

```
{
    "msg": "OK",
    "code": 0,
    "data": [
       "tbl1", "tbl2", "tbl3", ...
    ],
    "count": 0
}
```

- The data field returns a list of table names.

Schema Info

Request

```
GET /api/meta/namespaces/<ns_name>/databases/<db_name>/tables/<tbl_name>/schema
```

Description

Get the table structure information of the specified table in the specified database.

Path parameters

- `<db_name>`

  Specify the database name

- `<tbl_name>`

  Specify table name

Query parameters

- `with_mv`

  Optional. If not specified, the table structure of the base table is returned by default. If specified, all rollup index will also be returned.

Request body

None

Response

```
GET /api/meta/namespaces/default/databases/db1/tables/tbl1/schema

{
    "msg": "success",
    "code": 0,
    "data": {
        "tbl1": {
            "schema": [{
                    "Field": "k1",
                    "Type": "INT",
                    "Null": "Yes",
                    "Extra": "",
                    "Default": null,
                    "Key": "true"
                },
                {
                    "Field": "k2",
                    "Type": "INT",
                    "Null": "Yes",
                    "Extra": "",
```

```
                    "Default": null,
                    "Key": "true"
                }
            ],
            "is_base": true
        }
    },
    "count": 0
}
```

```
GET /api/meta/namespaces/default/databases/db1/tables/tbl1/schema?with_mv?=1

{
    "msg": "success",
    "code": 0,
    "data": {
        "tbl1": {
            "schema": [{
                    "Field": "k1",
                    "Type": "INT",
                    "Null": "Yes",
                    "Extra": "",
                    "Default": null,
                    "Key": "true"
                },
                {
                    "Field": "k2",
                    "Type": "INT",
                    "Null": "Yes",
                    "Extra": "",
                    "Default": null,
                    "Key": "true"
                }
            ],
            "is_base": true
        },
        "rollup1": {
            "schema": [{
                "Field": "k1",
                "Type": "INT",
                "Null": "Yes",
                "Extra": "",
                "Default": null,
                "Key": "true"
            }],
            "is_base": false
        }
    },
    "count": 0
}
```

- The data field returns the table structure information of the base table or rollup table.

Meta Replay State Action

Meta Replay State Action

(Not Implemented)

Request

```
GET /api/_meta_replay_state
```

Description

Get the status of FE node metadata replay.

Path parameters

None

Query parameters

None

Request body

None

Response

TODO

Examples

TODO

Metrics Action

Metrics Action

Request

`GET /api/metrics`

Description

Used to obtain doris metrics infomation.

Path parameters

None

Query parameters

- `type`

  Optional parameter. Display all metrics by default, and with following values:

  - `core` Output core metrics
  - `json` Output metrics in json format

Request body

None

Response

TO DO

Profile Action

Profile Action

Request

`GET /api/profile`

Description

Used to obtain the query profile of the specified query id. If query_id is not exists, return 404 NOT FOUND ERROR If query_id exists, return query profile like this

```
Query:
  Summary:
     - Query ID: a0a9259df9844029-845331577440a3bd
     - Start Time: 2020-06-15 14:10:05
     - End Time: 2020-06-15 14:10:05
     - Total: 8ms
     - Query Type: Query
     - Query State: EOF
     - Doris Version: trunk
     - User: root
     - Default Db: default_cluster:test
     - Sql Statement: select * from table1
  Execution Profile a0a9259df9844029-845331577440a3bd:(Active: 7.315ms, % non-child: 100.00%)
    Fragment 0:
      Instance a0a9259df9844029-845331577440a3be (host=TNetworkAddress(hostname:172.26.108.176, port:9560)):(Active: 1.523ms, % non-
child: 0.24%)
         - MemoryLimit: 2.00 GB
         - PeakUsedReservation: 0.00
         - PeakMemoryUsage: 72.00 KB
         - RowsProduced: 5
         - AverageThreadTokens: 0.00
         - PeakReservation: 0.00
        BlockMgr:
           - BlocksCreated: 0
           - BlockWritesOutstanding: 0
           - BytesWritten: 0.00
           - TotalEncryptionTime: 0ns
           - BufferedPins: 0
           - TotalReadBlockTime: 0ns
           - TotalBufferWaitTime: 0ns
           - BlocksRecycled: 0
           - TotalIntegrityCheckTime: 0ns
           - MaxBlockSize: 8.00 MB
        DataBufferSender (dst_fragment_instance_id=a0a9259df9844029-845331577440a3be):
           - AppendBatchTime: 9.23us
             - ResultSendTime: 956ns
             - TupleConvertTime: 5.735us
           - NumSentRows: 5
        OLAP_SCAN_NODE (id=0):(Active: 1.506ms, % non-child: 20.59%)
           - TotalRawReadTime: 0ns
           - CompressedBytesRead: 6.47 KB
           - PeakMemoryUsage: 0.00
           - RowsPushedCondFiltered: 0
```

```
            - ScanRangesComplete: 0
            - ScanTime: 25.195us
            - BitmapIndexFilterTimer: 0ns
            - BitmapIndexFilterCount: 0
            - NumScanners: 65
            - RowsStatsFiltered: 0
            - VectorPredEvalTime: 0ns
            - BlockSeekTime: 1.299ms
            - RawRowsRead: 1.91K (1910)
            - ScannerThreadsVoluntaryContextSwitches: 0
            - RowsDelFiltered: 0
            - IndexLoadTime: 911.104us
            - NumDiskAccess: 1
            - ScannerThreadsTotalWallClockTime: 0ns
              - MaterializeTupleTime: 0ns
              - ScannerThreadsUserTime: 0ns
              - ScannerThreadsSysTime: 0ns
            - TotalPagesNum: 0
            - RowsReturnedRate: 3.319K /sec
            - BlockLoadTime: 539.289us
            - CachedPagesNum: 0
            - BlocksLoad: 384
            - UncompressedBytesRead: 0.00
            - RowsBloomFilterFiltered: 0
            - TabletCount : 1
            - RowsReturned: 5
            - ScannerThreadsInvoluntaryContextSwitches: 0
            - DecompressorTimer: 0ns
            - RowsVectorPredFiltered: 0
            - ReaderInitTime: 6.498ms
            - RowsRead: 5
            - PerReadThreadRawHdfsThroughput: 0.0 /sec
            - BlockFetchTime: 4.318ms
            - ShowHintsTime: 0ns
            - TotalReadThroughput: 0.0 /sec
            - IOTimer: 1.154ms
            - BytesRead: 48.49 KB
            - BlockConvertTime: 97.539us
            - BlockSeekCount: 0
```

Path parameters

None

Query parameters

- query_id

  Specify query id

Request body

None

Response

```
{
    "msg": "success",
```

```
    "code": 0,
    "data": {
        "profile": "query profile ..."
    },
    "count": 0
}
```

Examples

1. Get the query profile of the specified query id

```
GET /api/profile?query_id=f732084bc8e74f39-8313581c9c3c0b58

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "profile": "query profile ..."
    },
    "count": 0
}
```

Query Schema Action

Query Schema Action

Request

```
POST /api/query_schema/<ns_name>/<db_name>
```

Description

The Query Schema Action can return the table creation statement for the given SQL-related table. Can be used to test some query scenarios locally.

The API was released in version 1.2.

Path parameters

- `<db_name>`

  Specify the database name. This database will be considered as the default database for the current session, and will be used if the table name in SQL does not qualify the database name.

Query parameters

None

Request body

```
text/plain

sql
```

- “sql” field is the SQL string.

Response

- Return value

```
CREATE TABLE `tbl1` (
  `k1` int(11) NULL,
  `k2` int(11) NULL
) ENGINE=OLAP
DUPLICATE KEY(`k1`, `k2`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`k1`) BUCKETS 3
PROPERTIES (
```

```
"replication_allocation" = "tag.location.default: 1",
"in_memory" = "false",
"storage_format" = "V2",
"disable_auto_compaction" = "false"
);

CREATE TABLE `tbl2` (
  `k1` int(11) NULL,
  `k2` int(11) NULL
) ENGINE=OLAP
DUPLICATE KEY(`k1`, `k2`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`k1`) BUCKETS 3
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
"in_memory" = "false",
"storage_format" = "V2",
"disable_auto_compaction" = "false"
);
```

Example

1. Write the SQL in local file 1.sql

```
select tbl1.k2 from tbl1 join tbl2 on tbl1.k1 = tbl2.k1;
```

2. Use curl to get the table creation statement.

```
curl -X POST -H 'Content-Type: text/plain'  -uroot: http://127.0.0.1:8030/api/query_schema/internal/db1 -d@1.sql
```

Query Stats Action

Request

```
View
get api/query_stats/<catalog_name>
get api/query_stats/<catalog_name>/<db_name>
get api/query_stats/<catalog_name>/<db_name>/<tbl_name>

Clear
delete api/query_stats/<catalog_name>/<db_name>
delete api/query_stats/<catalog_name>/<db_name>/<tbl_name>
```

Description

Get or delete the statistics information of the specified catalog database or table, if it is a doris catalog, you can use default_cluster

Path parameters

- `<catalog_name>` specified catalog name
-
- `<db_name>` specified database name
- `<tbl_name>` specified table name

Query parameters

- `summary` if true, only return summary information, otherwise return all the detailed statistics information of the table, only used in get

Request body

```
GET /api/query_stats/default_cluster/test_query_db/baseall?summary=false
{
    "msg": "success",
    "code": 0,
    "data": {
        "summary": {
            "query": 2
        },
        "detail": {
            "baseall": {
                "summary": {
                    "query": 2
                }
            }
        }
    },
```

```
    "count": 0
}
```

Response

- return statistics information

Example

2. use curl

```
curl --location -u root: 'http://127.0.0.1:8030/api/query_stats/default_cluster/test_query_db/baseall?summary=false'
```

Row Count Action

Row Count Action

Request

`GET /api/rowcount`

Description

Used to manually update the row count statistics of the specified table. While updating the statistics of the number of rows, the table and the number of rows corresponding to the rollup will also be returned in JSON format

Path parameters

None

Query parameters

- `db`

  Specify database

- `table`

  Specify table

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "tbl1": 10000
    },
    "count": 0
}
```

Examples

1. Update and get the number of rows in the specified Table

```
GET /api/rowcount?db=example_db&table=tbl1

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "tbl1": 10000
    },
    "count": 0
}
```

Set Config Action

Set Config Action

Request

`GET /api/_set_config`

Description

Used to dynamically set the configuration of FE. This command is passed through the `ADMIN SET FRONTEND CONFIG` command. But this command will only set the configuration of the corresponding FE node. And it will not automatically forward the `MasterOnly` configuration item to the Master FE node.

Path parameters

None

Query parameters

- `confkey1=confvalue1`

  Specify the configuration name to be set, and its value is the configuration value to be modified.

- `persist`

  Whether to persist the modified configuration. The default is false, which means it is not persisted. If it is true, the modified configuration item will be written into the `fe_custom.conf` file and will still take effect after FE is restarted.

- `reset_persist` Whether or not to clear the original persist configuration only takes effect when the persist parameter is true. For compatibility with the original version, reset_persist defaults to true.
  If persist is set to true and reset_persist is not set or reset_persist is true, the configuration in the `fe_custom.conf` file will be cleared before this modified configuration is written to `fe_custom.conf`.
  If persist is set to true and reset_persist is false, this modified configuration item will be incrementally added to `fe_custom.conf`.

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
```

```
    "data": {
        "set": {
            "key": "value"
        },
        "err": [
            {
               "config_name": "",
               "config_value": "",
               "err_info": ""
            }
        ],
        "persist":""
    },
    "count": 0
}
```

The `set` field indicates the successfully set configuration. The `err` field indicates the configuration that failed to be set. The `persist` field indicates persistent information.

Examples

1. Set the values of `storage_min_left_capacity_bytes`, `replica_ack_policy` and `agent_task_resend_wait_time_ms`.

```
GET /api/_set_config?
storage_min_left_capacity_bytes=1024&replica_ack_policy=SIMPLE_MAJORITY&agent_task_resend_wait_time_ms=true

Response:
{
"msg": "success",
"code": 0,
"data": {
    "set": {
        "storage_min_left_capacity_bytes": "1024"
    },
    "err": [
        {
            "config_name": "replica_ack_policy",
            "config_value": "SIMPLE_MAJORITY",
            "err_info": "Not support dynamic modification."
        },
        {
            "config_name": "agent_task_resend_wait_time_ms",
            "config_value": "true",
            "err_info": "Unsupported configuration value type."
        }
    ],
    "persist": ""
},
"count": 0
}

storage_min_left_capacity_bytes  Successfully;
replica_ack_policy  Failed, because the configuration item does not support dynamic modification.
agent_task_resend_wait_time_ms  Failed, failed to set the boolean type because the configuration item is of type long.
```

2. Set `max_bytes_per_broker_scanner` and persist it.

```
GET /api/_set_config?max_bytes_per_broker_scanner=21474836480&persist=true&reset_persist=false

Response:
{
"msg": "success",
"code": 0,
"data": {
    "set": {
        "max_bytes_per_broker_scanner": "21474836480"
    },
    "err": [],
    "persist": "ok"
},
"count": 0
}
```

The fe/conf directory generates the fe_custom.conf file: `#THIS IS AN AUTO GENERATED CONFIG FILE. #You can modify this file manually, and the configurations in this file #will overwrite the configurations in fe.conf #Wed Jul 28 12:43:14 CST 2021 max_bytes_per_broker_scanner=21474836480`

Show Data Action

Show Data Action

Request

`GET /api/show_data`

Description

Used to get the total data volume of the cluster or the data volume of the specified database. Unit byte.

Path parameters

None

Query parameters

- `db`

  Optional. If specified, get the data volume of the specified database.

Request body

None

Response

1. Specify the amount of data in the database.

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "default_cluster:db1": 381
    },
    "count": 0
}
```

2. Total data

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "__total_size": 381
    },
```

```
    "count": 0
}
```

Examples

1. Get the data volume of the specified database

```
GET /api/show_data?db=db1

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "default_cluster:db1": 381
    },
    "count": 0
}
```

2. Get the total data volume of the cluster

```
GET /api/show_data

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "__total_size": 381
    },
    "count": 0
}
```

Show Meta Info Action

Show Meta Info Action

Request

`GET /api/show_meta_info`

Description

Used to display some metadata information

Path parameters

无

Query parameters

- action

  Specify the type of metadata information to be obtained. Currently supports the following:

  - `SHOW_DB_SIZE`

    Get the data size of the specified database, in bytes.

  - `SHOW_HA`

    Obtain the playback status of FE metadata logs and the status of electable groups.

Request body

None

Response

- `SHOW_DB_SIZE`

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "default_cluster:information_schema": 0,
        "default_cluster:db1": 381
    },
    "count": 0
}
```

- SHOW_HA

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "can_read": "true",
        "role": "MASTER",
        "is_ready": "true",
        "last_checkpoint_version": "1492",
        "last_checkpoint_time": "1596465109000",
        "current_journal_id": "1595",
        "electable_nodes": "",
        "observer_nodes": "",
        "master": "10.81.85.89"
    },
    "count": 0
}
```

Examples

1. View the data size of each database in the cluster

```
GET /api/show_meta_info?action=show_db_size

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "default_cluster:information_schema": 0,
        "default_cluster:db1": 381
    },
    "count": 0
}
```

2. View the FE election group situation

```
GET /api/show_meta_info?action=show_ha

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "can_read": "true",
        "role": "MASTER",
        "is_ready": "true",
        "last_checkpoint_version": "1492",
        "last_checkpoint_time": "1596465109000",
        "current_journal_id": "1595",
        "electable_nodes": "",
        "observer_nodes": "",
        "master": "10.81.85.89"
    },
    "count": 0
}
```

Show Proc Action

Show Proc Action

Request

`GET /api/show_proc`

Description

Used to obtain PROC information.

Path parameters

None

Query parameters

- path

  Specify Proc Path

- forward

  Whether to forward to Master FE for execution

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": [
        proc infos ...
    ],
    "count": 0
}
```

Examples

1. View `/statistic` information

   ```
   GET /api/show_proc?path=/statistic
   ```

```
Response:
{
    "msg": "success",
    "code": 0,
    "data": [
        ["10003", "default_cluster:db1", "2", "3", "3", "3", "3", "0", "0", "0"],
        ["10013", "default_cluster:doris_audit_db__", "1", "4", "4", "4", "4", "0", "0", "0"],
        ["Total", "2", "3", "7", "7", "7", "7", "0", "0", "0"]
    ],
    "count": 0
}
```

2. Forward to Master for execution

```
GET /api/show_proc?path=/statistic&forward=true

Response:
{
    "msg": "success",
    "code": 0,
    "data": [
        ["10003", "default_cluster:db1", "2", "3", "3", "3", "3", "0", "0", "0"],
        ["10013", "default_cluster:doris_audit_db__", "1", "4", "4", "4", "4", "0", "0", "0"],
        ["Total", "2", "3", "7", "7", "7", "7", "0", "0", "0"]
    ],
    "count": 0
}
```

Show Runtime Info Action

Show Runtime Info Action

Request

```
GET /api/show_runtime_info
```

Description

Used to obtain Runtime information of FE JVM

Path parameters

None

Query parameters

None

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "free_mem": "855642056",
        "total_mem": "1037959168",
        "thread_cnt": "98",
        "max_mem": "1037959168"
    },
    "count": 0
}
```

Examples

1. Get the JVM information of the current FE node

```
GET /api/show_runtime_info

Response:
{
    "msg": "success",
    "code": 0,
```

```
    "data": {
        "free_mem": "855642056",
        "total_mem": "1037959168",
        "thread_cnt": "98",
        "max_mem": "1037959168"
    },
    "count": 0
}
```

Show Table Data Action

Show Table Data Action

Request

`GET /api/show_table_data`

Description

Used to get the data size of all tables in all databases under all internal catalog, or the data size of the specified database or table. Unit byte.

Path parameters

NULL

Query parameters

- `db`

  Optional. If specified, get the data size of the tables under the specified database.

- `table`

  Optional. If specified, get the data size of the specified table.

- `single_replica`

  Optional. If specified, get the data size of the single replica of the table.

Request body

NULL

Response

1. The data size of all tables in the specified database.

```
{
    "msg":"success",
    "code":0,
    "data":{
        "tpch":{
            "partsupp":9024548244,
            "revenue0":0,
            "customer":1906421482
        }
    },
```

```
    "count":0
}
```

2. The data size of the specified table of the specified db.

```
{
    "msg":"success",
    "code":0,
    "data":{
        "tpch":{
            "partsupp":9024548244
        }
    },
    "count":0
}
```

3. The data size of the single replica of the specified table of the specified db.

```
{
    "msg":"success",
    "code":0,
    "data":{
        "tpch":{
            "partsupp":3008182748
        }
    },
    "count":0
}
```

Examples

1. The data size of all tables in the specified database.

```
GET /api/show_table_data?db=tpch

Response:
{
    "msg":"success",
    "code":0,
    "data":{
        "tpch":{
            "partsupp":9024548244,
            "revenue0":0,
            "customer":1906421482
        }
    },
    "count":0
}
```

2. The data size of the specified table of the specified db.

```
GET /api/show_table_data?db=tpch&table=partsupp

Response:
{
    "msg":"success",
    "code":0,
```

```
    "data":{
        "tpch":{
            "partsupp":9024548244
        }
    },
    "count":0
}
```

3. The data size of the single replica of the specified table of the specified db.

```
GET /api/show_table_data?db=tpch&table=partsupp&single_replica=true

Response:
{
    "msg":"success",
    "code":0,
    "data":{
        "tpch":{
            "partsupp":3008182748
        }
    },
    "count":0
}
```

Statement Execution Action

Statement Execution Action

Request

```
POST /api/query/<ns_name>/<db_name>
```

Description

Statement Execution Action is used to execute a statement and return the result.

Path parameters

- <db_name>

  Specify the database name. This database will be regarded as the default database of the current session. If the table name in SQL does not qualify the database name, this database will be used.

Query parameters

None

Request body

```
{
    "stmt" : "select * from tbl1"
}
```

- sql 字段为具体的 SQL

Response

- 返回结果集

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "type": "result_set",
        "data": [
            [1],
            [2]
        ],
        "meta": [{
            "name": "k1",
            "type": "INT"
```

```
        }],
        "status": {},
        "time": 10
    },
    "count": 0
}
```

- The type field is `result_set`, which means the result set is returned. The results need to be obtained and displayed based on the meta and data fields. The meta field describes the column information returned. The data field returns the result row. The column type in each row needs to be judged by the content of the meta field. The status field returns some information of MySQL, such as the number of alarm rows, status code, etc. The time field return the execution time, unit is millisecond.

- Return execution result

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "type": "exec_status",
        "status": {}
    },
    "count": 0,
    "time": 10
}
```

- The type field is `exec_status`, which means the execution result is returned. At present, if the return result is received, it means that the statement was executed successfully.

Table Query Plan Action

Table Query Plan Action

Request

`POST /api/<db>/<table>/_query_plan`

Description

Given a SQL, it is used to obtain the query plan corresponding to the SQL.

This interface is currently used in Spark-Doris-Connector, Spark obtains Doris' query plan.

Path parameters

- `<db>`

  Specify database

- `<table>`

  Specify table

Query parameters

None

Request body

```
{
    "sql": "select * from db1.tbl1;"
}
```

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "partitions": {
            "10039": {
                "routings": ["10.81.85.89:9062"],
                "version": 2,
                "versionHash": 982459448378619656,
                "schemaHash": 1294206575
            }
        },
```

```
        "opaqued_query_plan":
"DAABDAACDwABDAAAAAEIAAEAAAAACAACAAAAAAgAAwAAAAAKAAT/////////w8ABQgAAAABAAAAA8ABgIAAAABAAIACAAMABIIAAEAAAAADwACCwAAAAIAAAACazEAAAA
CazIPAAMIAAAAAgAAAAUAAAAFAgAEAQAAAA8ABAwAAAACDwABDAAAAAEIAAEAAAAQDAACDwABDAAAAAEIAAEAAAAADAACCAABAAAABQAAAAgABAAAAAAMAA8IAAEAAAAACAA
CAAAAAAAIABT/////CAAX/////wAADwABDAAAAAEIAAEAAAAQDAACDwABDAAAAAEIAAEAAAAADAACCAABAAAABQAAAAgABAAAAAAMAA8IAAEAAAABCAACAAAAAAAIABT////
/CAAX/////wAADAAFCAABAAAABgwACAAADAAGCAABAAAAAA8AAgwAAAAAAAoABwAAAAAAAAAACgAIAAAAAAAAAAAADQACCgwAAAABAAAAAAAAJzcKAAEAAAAAAAAnNwoAAgA
AAAAAAAACCgADDaJlqbrVdwgIAARNJAZvAAwAAw8AAQwAAAACCAABAAAAAAgAAgAAAAAMAAMPAAEMAAAAAQgAAQAAAAAMAAIIAAEAAAAFAAAACAAE/////wgABQAAAAQIAAY
AAAAACAAHAAAAAAsACAAAAAJrMQgACQAAAAACAAoBAAgAAQAAAAEIAAIAAAAADAADDwABDAAAAAEIAAEAAAAADAACCAABAAAABQAAAAgABP////8IAAUAAAAICAAGAAAAAAg
ABwAAAAELAAgAAAACazIIAAkAAAABAgAKAQAPAAIMAAAAAQgAAQAAAAAIAAIAAAAMCAADAAAAAQoABAAAAAAAACc1CAAFAAAAAgAPAAMMAAAAAQoAAQAAAAAACc1CAACAAA
AAQgAAwAAAAIIAAQAAAAACwAHAAAABHRibDELAAgAAAAADAALCwABAAAABHRibDEAAAAMAAQKAAFfL5rpxl1I4goAArgs6f+h6eMxAAA=",
        "status": 200
    },
    "count": 0
}
```

Among them, `opaqued_query_plan` is the binary format of the query plan.

Examples

1. Get the query plan of the specified SQL

```
POST /api/db1/tbl1/_query_plan
{
    "sql": "select * from db1.tbl1;"
}

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "partitions": {
            "10039": {
                "routings": ["192.168.1.1:9060"],
                "version": 2,
                "versionHash": 982459448378619656,
                "schemaHash": 1294206575
            }
        },
        "opaqued_query_plan": "DAABDAACDwABD...",
        "status": 200
    },
    "count": 0
}
```

Table Row Count Action

Table Row Count Action

Request

`GET /api/<db>/<table>/_count`

Description

Used to obtain statistics about the number of rows in a specified table. This interface is currently used in Spark-Doris-Connector. Spark obtains Doris table statistics.

Path parameters

- `<db>`

  Specify database

- `<table>`

  Specify table

Query parameters

None

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "size": 1,
        "status": 200
    },
    "count": 0
}
```

The `data.size` field indicates the number of rows in the specified table.

Examples

1. Get the number of rows in the specified table.

```
GET /api/db1/tbl1/_count

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "size": 1,
        "status": 200
    },
    "count": 0
}
```

Table Schema Action

Table Schema Action

Request

`GET /api/<db>/<table>/_schema`

Description

Used to obtain the table structure information of the specified table. This interface is currently used in Spark/Flink Doris Connector. obtains Doris table structure information.

Path parameters

- `<db>`

  Specify database

- `<table>`

  Specify table

Query parameters

None

Request body

None

Response

- The http interface returns as follows:

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "properties": [{
            "type": "INT",
            "name": "k1",
            "comment": "",
            "aggregation_type":""
        }, {
            "type": "INT",
            "name": "k2",
            "comment": "",
            "aggregation_type":"MAX"
```

```
        }],
        "keysType":UNIQUE_KEYS,
        "status": 200
    },
    "count": 0
}
```

- The http v2 interface returns as follows:

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "properties": [{
            "type": "INT",
            "name": "k1",
            "comment": ""
        }, {
            "type": "INT",
            "name": "k2",
            "comment": ""
        }],
        "keysType":UNIQUE_KEYS,
        "status": 200
    },
    "count": 0
}
```

Note: The difference is that the `http` method returns more `aggregation_type` fields than the `http v2` method. The `http v2` is enabled by setting `enable_http_server_v2`. For detailed parameter descriptions, see fe parameter settings

Examples

1. Get the table structure information of the specified table via http interface.

```
GET /api/db1/tbl1/_schema

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "properties": [{
            "type": "INT",
            "name": "k1",
            "comment": "",
            "aggregation_type":""
        }, {
            "type": "INT",
            "name": "k2",
            "comment": "",
            "aggregation_type":"MAX"
        }],
        "keysType":UNIQUE_KEYS,
        "status": 200
    },
    "count": 0
}
```

2. Get the table structure information of the specified table via http v2 interface.

```
GET /api/db1/tbl1/_schema

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "properties": [{
            "type": "INT",
            "name": "k1",
            "comment": ""
        }, {
            "type": "INT",
            "name": "k2",
            "comment": ""
        }],
        "keysType":UNIQUE_KEYS,
        "status": 200
    },
    "count": 0
}
```

Upload Action

Upload Action

Upload Action currently mainly serves the front-end page of FE, and is used for users to load small test files.

Upload load file

Used to upload a file to the FE node, which can be used to load the file later. Currently only supports uploading files up to 100MB.

Request

```
POST /api/<namespace>/<db>/<tbl>/upload
```

Path parameters

- `<namespace>`

  Namespace, currently only supports `default_cluster`

- `<db>`

  Specify database

- `<tbl>`

  Specify table

Query parameters

- `column_separator`

  Optional, specify the column separator of the file. Default is `\t`

- `preview`

  Optional, if set to `true`, up to 10 rows of data rows split according to `column_separator` will be displayed in the returned result.

Request body

The content of the file to be uploaded, the Content-type is `multipart/form-data`

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "id": 1,
        "uuid": "b87824a4-f6fd-42c9-b9f1-c6d68c5964c2",
        "originFileName": "data.txt",
        "fileSize": 102400,
        "absPath": "/path/to/file/data.txt"
        "maxColNum" : 5
    },
    "count": 1
}
```

Load the uploaded file

Request

```
PUT /api/<namespace>/<db>/<tbl>/upload
```

Path parameters

- `<namespace>`

  Namespace, currently only supports `default_cluster`

- `<db>`

  Specify database

- `<tbl>`

  Specify table

Query parameters

- `file_id`

  Specify the load file id, which is returned by the API that uploads the file.

- `file_uuid`

  Specify the file uuid, which is returned by the API that uploads the file.

Header

The options in the header are the same as those in the header in the Stream Load request.

Request body

None

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "TxnId": 7009,
        "Label": "9dbdfb0a-120b-47a2-b078-4531498727cb",
        "Status": "Success",
        "Message": "OK",
        "NumberTotalRows": 3,
        "NumberLoadedRows": 3,
        "NumberFilteredRows": 0,
        "NumberUnselectedRows": 0,
        "LoadBytes": 12,
        "LoadTimeMs": 71,
        "BeginTxnTimeMs": 0,
        "StreamLoadPutTimeMs": 1,
        "ReadDataTimeMs": 0,
        "WriteDataTimeMs": 13,
        "CommitAndPublishTimeMs": 53
    },
    "count": 1
}
```

Example

```
PUT /api/default_cluster/db1/tbl1/upload?file_id=1&file_uuid=b87824a4-f6fd-42c9-b9f1-c6d68c5964c2
```

Import Action

Import Action

Request

`POST /api/import/file_review`

Description

View the contents of the file in CSV or PARQUET format.

Path parameters

None

Query parameters

None

Request body

TO DO

Response

TO DO

Meta Info Action

Meta Info Action

Request

`GET /api/meta/namespaces/<ns>/databases GET /api/meta/namespaces/<ns>/databases/<db>/tables GET /api/meta/namespaces/<ns>/databases/<db>/tables/<tbl>/schema`

Description

Used to obtain metadata information about the cluster, including the database list, table list, and table schema.

Path parameters

- `ns`

  Specify cluster name.

- `db`

  Specify database name.

- `tbl`

  Specify table name.

Query parameters

None

Request body

None

Response

```
{
    "msg":"success",
    "code":0,
    "data":["database list" / "table list" / "table schema"],
    "count":0
}
```

Debug Point

Debug Point

Debug point is used in code test. When enabling a debug point, it can run related code.

Both FE and BE support debug points.

Code Example

FE example

```
private Status foo() {
    // dbug_fe_foo_do_nothing is the debug point name.
    // When it's active, DebugPointUtil.isEnable("dbug_fe_foo_do_nothing") will return true.
    if (DebugPointUtil.isEnable("dbug_fe_foo_do_nothing")) {
        return Status.Nothing;
    }

    do_foo_action();

    return Status.Ok;
}
```

BE 桩子示例代码

```
void Status foo() {
     // dbug_be_foo_do_nothing is the debug point name.
     // When it's active, DEBUG_EXECUTE_IF will execute the code block.
     DEBUG_EXECUTE_IF("dbug_be_foo_do_nothing",  { return Status.Nothing; });

     do_foo_action();

     return Status.Ok;
}
```

Global config

To activate debug points, need set `enable_debug_points` to true.

`enable_debug_points` was located in FE's fe.conf and BE's be.conf。

Enable Debug Point

API

```
    POST /api/debug_point/add/{debug_point_name}[?timeout=<int>&execute=<int>]
```

Query Parameters

- `debug_point_name` Debug point name. Require.
- `timeout` Timeout in seconds. When timeout, the debug point will be disable. Default is -1, not timeout. Optional.
- `execute` Max active times。Default is -1, unlimit active times. Optional.

Request body

None

Response

```
{
    msg: "OK",
    code: 0
}
```

Examples

Enable debug point `foo`, activate no more than five times.

```
curl -X POST "http://127.0.0.1:8030/api/debug_point/add/foo?execute=5"
```

Disable Debug Point

API

```
POST /api/debug_point/remove/{debug_point_name}
```

Query Parameters

- `debug_point_name` Debug point name. Require.

Request body

None

Response

```
    {
        msg: "OK",
        code: 0
    }
```

Examples

Disable debug point `foo`。

```
```
curl -X POST "http://127.0.0.1:8030/api/debug_point/remove/foo"

```
```

Clear Debug Points

API

```
    POST /api/debug_point/clear
```

Request body

None

Response

```
```
{
    msg: "OK",
    code: 0
}
```
```

Examples

```
```
curl -X POST "http://127.0.0.1:8030/api/debug_point/clear"
```
```

Statistic Action

Statistic Action

Request

`GET /rest/v2/api/cluster_overview`

Description

获取集群统计信息、库表数量等。

Path parameters

无

Query parameters

无

Request body

无

Response


```
{
    "msg":"success",
    "code":0,
    "data":{"diskOccupancy":0,"remainDisk":5701197971457,"feCount":1,"tblCount":27,"beCount":1,"dbCount":2},
    "count":0
}
```

Check Stub Cache

CHECK Stub Cache

Request

`GET /api/check_rpc_channel/{host_to_check}/{remot_brpc_port}/{payload_size}`

Description

Check whether the connection cache is available

Path parameters

- `host_to_check`

  Host to check

- `remot_brpc_port`

  Remot brpc port

- `payload_size`

  Load size, unit: B, value range 1~1024000.

Request body

None

Response

```
```
{
    "msg":"success",
    "code":0,
    "data": "open brpc connection to {host_to_check}:{remot_brpc_port} succcess.",
    "count":0
}
```
```

Examples

```
```
curl http://127.0.0.1:8040/api/check_rpc_channel/127.0.0.1/8060/1024000
```
```

Reset Stub Cache

Reset Stub Cache

Request

`GET /api/reset_rpc_channel/{endpoints}`

Description

Reset the connection cache of brpc

Path parameters

- `endpoints`
  - `all`: clear all caches
  - `host1:port1,host2:port2`: clear cache of the specified target

Request body

None

Response

```
{
    "msg":"success",
    "code":0,
    "data": "no cached channel.",
    "count":0
}
```

Examples

```
curl http://127.0.0.1:8040/api/reset_rpc_channel/all
```

```
curl http://127.0.0.1:8040/api/reset_rpc_channel/1.1.1.1:8080,2.2.2.2:8080
```

View Compaction Status

View Compaction Status

Request

`GET /api/compaction/run_status GET /api/compaction/show?tablet_id={int}`

Description

It is used to view the overall compaction status of a BE node and the compaction status of a specified tablet.

Query parameters

- `tablet_id` ID of the tablet

Request body

None

Response

The overall compaction status of the node

```
{
  "CumulativeCompaction": {
         "/home/disk1" : [10001, 10002],
         "/home/disk2" : [10003]
  },
  "BaseCompaction": {
         "/home/disk1" : [10001, 10002],
         "/home/disk2" : [10003]
  }
}
```

This structure represents the id of the tablet that is performing the compaction task in a certain data directory, and the type of compaction.

Specify the compaction status of the tablet

```
{
    "cumulative policy type": "SIZE_BASED",
    "cumulative point": 50,
    "last cumulative failure time": "2019-12-16 18:13:43.224",
    "last base failure time": "2019-12-16 18:13:23.320",
    "last cumu success time": ,
    "last base success time": "2019-12-16 18:11:50.780",
    "rowsets": [
```

```
        "[0-48] 10 DATA OVERLAPPING 574.00 MB",
        "[49-49] 2 DATA OVERLAPPING 574.00 B",
        "[50-50] 0 DELETE NONOVERLAPPING 574.00 B",
        "[51-51] 5 DATA OVERLAPPING 574.00 B"
    ],
    "missing_rowsets": [],
    "stale version path": [
        {
            "path id": "2",
            "last create time": "2019-12-16 18:11:15.110 +0800",
            "path list": "2-> [0-24] -> [25-48]"
        },
        {
            "path id": "1",
            "last create time": "2019-12-16 18:13:15.110 +0800",
            "path list": "1-> [25-40] -> [40-48]"
        }
    ]
}
```

Explanation of results:

- cumulative policy type: The cumulative compaction policy type which is used by current tablet.
- cumulative point: The version boundary between base and cumulative compaction. Versions before (excluding) points are handled by base compaction. Versions after (inclusive) are handled by cumulative compaction.
- last cumulative failure time: The time when the last cumulative compaction failed. After 10 minutes by default, cumulative compaction is attempted on the this tablet again.
- last base failure time: The time when the last base compaction failed. After 10 minutes by default, base compaction is attempted on the this tablet again.
- rowsets: The current rowsets collection of this tablet. [0-48] means a rowset with version 0-48. The second number is the number of segments in a rowset. The DELETE indicates the delete version. OVERLAPPING and NONOVERLAPPING indicates whether data between segments is overlap.
- missing_rowset: The missing rowsets.
- stale version path: The merged version path of the rowset collection currently merged in the tablet. It is an array structure and each element represents a merged path. Each element contains three attributes: path id indicates the version path id, and last create time indicates the creation time of the most recent rowset on the path. By default, all rowsets on this path will be deleted after half an hour at the last create time.

Examples

```
curl http://192.168.10.24:8040/api/compaction/show?tablet_id=10015
```

Manually Trigger Compaction

Manually Trigger Compaction

Request

`POST /api/compaction/run?tablet_id={int}&compact_type={enum}` `POST /api/compaction/run?table_id={int}&compact_type=full` Note that table_id=xxx will take effect only when compact_type=full is specified. `GET /api/compaction/run_status?tablet_id={int}`

Description

Used to manually trigger the comparison and show status.

Query parameters

- `tablet_id`
  - ID of the tablet
- `table_id`
  - ID of table. Note that table_id=xxx will take effect only when compact_type=full is specified, and only one tablet_id and table_id can be specified, and cannot be specified at the same time. After specifying table_id, full_compaction will be automatically executed for all tablets under this table.
- `compact_type`
  - The value is `base` or `cumulative` or `full`. For usage scenarios of full_compaction, please refer to Data Recovery.

Request body

None

Response

Trigger Compaction

If the tablet does not exist, an error in JSON format is returned:

```
{
    "status": "Fail",
    "msg": "Tablet not found"
}
```

If the tablet exists and the tablet is not running, JSON format is returned:

```
{
    "status": "Fail",
```

```
    "msg": "fail to execute compaction, error = -2000"
}
```

If the tablet exists and the tablet is running, JSON format is returned:

```
{
    "status": "Success",
    "msg": "compaction task is successfully triggered."
}
```

Explanation of results:

- status: Trigger task status, when it is successfully triggered, it is Success; when for some reason (for example, the appropriate version is not obtained), it returns Fail.
- msg: Give specific success or failure information.

Show Status

If the tablet does not exist, an error in JSON format is returned:

```
{
    "status": "Fail",
    "msg": "Tablet not found"
}
```

If the tablet exists and the tablet is not running, JSON format is returned:

```
{
    "status" : "Success",
    "run_status" : false,
    "msg" : "this tablet_id is not running",
    "tablet_id" : 11308,
    "schema_hash" : 700967178,
    "compact_type" : ""
}
```

If the tablet exists and the tablet is running, JSON format is returned:

```
{
    "status" : "Success",
    "run_status" : true,
    "msg" : "this tablet_id is running",
    "tablet_id" : 11308,
    "schema_hash" : 700967178,
    "compact_type" : "cumulative"
}
```

Explanation of results:

- run_status: Get the current manual compaction task execution status.

Examples

```
curl -X POST "http://127.0.0.1:8040/api/compaction/run?tablet_id=10015&compact_type=cumulative"
```

View Meta

View Meta

Request

`GET /api/meta/header/{tablet_id}?byte_to_base64={bool}`

Description

View meta of a tablet

Path parameters

- `tablet_id` ID of the tablet

Query parameters

- `byte_to_base64` Whether to encode by base64. Optional with default `false`.

Request body

None

Response

```
{
    "table_id": 148107,
    "partition_id": 148104,
    "tablet_id": 148193,
    "schema_hash": 2090621954,
    "shard_id": 38,
    "creation_time": 1673253868,
    "cumulative_layer_point": -1,
    "tablet_state": "PB_RUNNING",
    ...
}
```

Examples

```
curl "http://127.0.0.1:8040/api/meta/header/148193&byte_to_base64=true"

```

Make Snapshot

Make Snapshot

Request

`GET /api/snapshot?tablet_id={int}&schema_hash={int}"`

Description

Make snapshot of a tablet

Query parameters

- `tablet_id` ID of the tablet
- `schema_hash` Schema hash

Request body

None

Response

```
/path/to/snapshot
```

Examples

```
curl "http://127.0.0.1:8040/api/snapshot?tablet_id=123456&schema_hash=1111111"

```

Check All Tablet Segment Lost

Check All Tablet Segment Lost

Request

`GET /api/check_tablet_segment_lost?repair={bool}`

Description

There may be some exceptions that cause segment to be lost on BE node. However, the metadata shows that the tablet is normal. This abnormal replica is not detected by FE and cannot be automatically repaired. When query comes, exception information is thrown that `failed to initialize storage reader`. The function of this interface is to check all tablets on the current BE node that have lost segment.

Query parameters

- `repair`
  - `true`: tablets with lost segment will be set to SHUTDOWN status and treated as bad replica, which can be detected and repaired by FE.
  - `false`: all tablets with missing segment are returned and nothing is done.

Request body

None

Response

````
The return is all tablets on the current BE node that have lost segment:

```
{
    status: "Success",
    msg: "Succeed to check all tablet segment",
    num: 3,
    bad_tablets: [
        11190,
        11210,
        11216
    ],
    set_bad: true,
    host: "172.3.0.101"
}
```
````

Examples

```
curl http://127.0.0.1:8040/api/check_tablet_segment_lost?repair=false
```

Config of BE

Config of BE

Request

`GET /api/show_config POST /api/update_config?{key}={val}`

Description

Query and update the config of BE

Query parameters

- `persist` Whether to persist. Optional with default `false`.
- `key` Config item name
- `val` Config item value

Request body

None

Response

Query

```
[["agent_task_trace_threshold_sec","int32_t","2","true"], ...]
```

Update

```
[
    {
        "config_name": "agent_task_trace_threshold_sec",
        "status": "OK",
        "msg": ""
    }
]
```

```
[
    {
        "config_name": "agent_task_trace_threshold_sec",
        "status": "OK",
        "msg": ""
    },
    {
```

```
        "config_name": "enable_segcompaction",
        "status": "BAD",
        "msg": "set enable_segcompaction=false failed, reason: [NOT_IMPLEMENTED_ERROR]'enable_segcompaction' is not support to
modify."
    },
    {
        "config_name": "enable_time_lut",
        "status": "BAD",
        "msg": "set enable_time_lut=false failed, reason: [NOT_IMPLEMENTED_ERROR]'enable_time_lut' is not support to modify."
    }
]
```

Examples

```
curl "http://127.0.0.1:8040/api/show_config"
```

```
curl -X POST "http://127.0.0.1:8040/api/update_config?agent_task_trace_threshold_sec=2&persist=true"
```

```
curl -X POST "http://127.0.0.1:8040/api/update_config?
agent_task_trace_threshold_sec=2&enable_merge_on_write_correctness_check=true&persist=true"
```

Metrics

Metrics

Request

`GET /metrics?type={enum}&with_tablet={bool}`

Description

Provided for prometheus

Query parameters

- `type` Output style, Optional with default `all` and the following values:
  - `core`: Only core items
  - `json`: Json format
- `with_tablet` Whether to output tablet-related items，Optional with default `false`.

Request body

None

Response

````
```
doris_be__max_network_receive_bytes_rate LONG 60757
doris_be__max_network_send_bytes_rate LONG 16232
doris_be_process_thread_num LONG 1120
doris_be_process_fd_num_used LONG 336
’’’

```
````

Examples

````
```
    curl "http://127.0.0.1:8040/metrics?type=json&with_tablet=true"
```
````

View Tablet Distribution

View Tablet Distribution

Request

`GET /api/tablets_distribution?group_by={enum}&partition_id={int}`

Description

Get the distribution of tablets under each partition between different disks on BE node

Query parameters

- `group_by` only supports `partition`
- `partition_id` ID of the specified partition，Optional with default all partition。

Request body

None

Response

```
```
{
    msg: "OK",
    code: 0,
    data: {
        host: "***",
        tablets_distribution: [
            {
                partition_id:***,
                disks:[
                    {
                        disk_path:"***",
                        tablets_num:***,
                        tablets:[
                            {
                                tablet_id:***,
                                schema_hash:***,
                                tablet_size:***
                            },

                            ...

                        ]
                    },

                    ...

                ]
```
```

```
            }
        ]
    },
    count: ***
}
```

Examples

```
curl "http://127.0.0.1:8040/api/tablets_distribution?group_by=partition&partition_id=123"

```

Migration Tablet

Migration Tablet

Request

`GET /api/tablet_migration?goal={enum}&tablet_id={int}&schema_hash={int}&disk={string}`

Description

Migrate a tablet to the specified disk.

Query parameters

- `goal`
    - `run`: submit the migration task
    - `status`: show the status of migration task
- `tablet_id` ID of the tablet
- `schema_hash` Schema hash
- `disk` The specified disk.

Request body

None

Response

Submit Task

```
{
    status: "Success",
    msg: "migration task is successfully submitted."
}
```

Or

```
{
    status: "Fail",
    msg: "Migration task submission failed"
}
```

Show Status

```
    {
        status: "Success",
        msg: "migration task is running",
        dest_disk: "xxxxxx"
    }
```

Or

```
    {
        status: "Success",
        msg: "migration task has finished successfully",
        dest_disk: "xxxxxx"
    }
```

Or

```
    {
        status: "Success",
        msg: "migration task failed.",
        dest_disk: "xxxxxx"
    }
```

Examples

```
```
curl "http://127.0.0.1:8040/api/tablet_migration?goal=run&tablet_id=123&schema_hash=333&disk=/disk1"

```
```

View Tablet Info

View Tablet Info

Request

`GET /tablets_json?limit={int}`

Description

View Tablet Info, including ID and schema hash.

Query parameters

- `limit` Number of tablets output，Optional with default 1000. Take `all` to output all tablets.

Request body

None

Response

```
{
    msg: "OK",
    code: 0,
    data: {
        host: "10.38.157.107",
        tablets: [
            {
                tablet_id: 11119,
                schema_hash: 714349777
            },

                ...

            {
                tablet_id: 11063,
                schema_hash: 714349777
            }
        ]
    },
    count: 30
}
```

Examples

```
curl http://127.0.0.1:8040/tablets_json?limit=all

```

Checksum

Checksum

Request

`GET /api/checksum?tablet_id={int}&version={int}&schema_hash={int}`

Description

Checksum

Query parameters

- `tablet_id` ID of the tablet to be checked
- `version` Version of the tablet to be verified
- `schema_hash` Schema hash

Request body

None

Response

```
1843743562
```

Examples

```
curl "http://127.0.0.1:8040/api/checksum?tablet_id=1&version=1&schema_hash=-1"

```

Download Log about Load Error

Download Log about Load Error

Request

`GET /api/_load_error_log?token={string}&file={string}`

Description

Download log file about load error

Query parameters

- `file` Path of log
- `token` token

Request body

None

Response

```
File of log
```

Examples

```
curl "http://127.0.0.1:8040/api/_load_error_log?file=a&token=1"
```

Pad Rowset

Pad Rowset

Request

`POST /api/pad_rowset?tablet_id={int}&start_version={int}&end_version={int}`

Description

Pad one empty rowset as one substitute for error replica.

Query parameters

- `tablet_id` ID of the tablet
- `start_version` Start version
- `end_version` End version

Request body

None

Response

```
{
    msg: "OK",
    code: 0
}
```

Examples

```
curl -X POST "http://127.0.0.1:8040/api/pad_rowset?tablet_id=123456&start_version=1111111&end_version=1111112"

```

Be Version Info

Be Version Info

Request

`GET /api/be_version_info`

Description

Provide BE version info

Query parameters

None

Request body

None

Response

```
{
    "msg":"success",
    "code":0,
    "data":{
        "beVersionInfo":{
            "dorisBuildVersionPrefix":"doris",
            "dorisBuildVersionMajor":0,
            "dorisBuildVersionMinor":0,
            "dorisBuildVersionPatch":0,
            "dorisBuildVersionRcVersion":"trunk",
            "dorisBuildVersion":"doris-0.0.0-trunk",
            "dorisBuildHash":"git://4b7b503d1cb3/data/doris/doris/be/../@a04f9814fe5a09c0d9e9399fe71cc4d765f8bff1",
            "dorisBuildShortHash":"a04f981",
            "dorisBuildTime":"Fri, 09 Sep 2022 07:57:02 UTC",
            "dorisBuildInfo":"root@4b7b503d1cb3"
        }
    },
    "count":0
}
```

Examples

```
curl http://127.0.0.1:8040/api/be_version_info
```

```
```

Check Alive

Check Alive

Request

```
GET /api/health
```

Description

Provided for the monitoring service to Check whether the BE is alive，Be will respond if alive.

Query parameters

None

Request body

None

Response

```
{"status": "OK","msg": "To Be Added"}
```

Examples

```
curl http://127.0.0.1:8040/api/health
```

Reload Tablet

Reload Tablet

Request

`GET /api/reload_tablet?tablet_id={int}&schema_hash={int}&path={string}"`

Description

Reload tablet

Query parameters

- `tablet_id` ID of the tablet
- `schema_hash` Schema hash
- `path` Path of file

Request body

None

Response

```
load header succeed
```

Examples

```
curl "http://127.0.0.1:8040/api/reload_tablet?tablet_id=123456&schema_hash=1111111&path=/abc"

```

Restore Tablet

Restore Tablet

Request

`POST /api/restore_tablet?tablet_id={int}&schema_hash={int}"`

Description

To restore the tablet data from trash dir on BE

Query parameters

- `tablet_id` ID of the tablet
- `schema_hash` Schema hash

Request body

None

Response

```
{
    msg: "OK",
    code: 0
}
```

Examples

```
curl -X POST "http://127.0.0.1:8040/api/restore_tablet?tablet_id=123456&schema_hash=1111111"
```

Developing Plugins

Doris Plugin Framework

Introduction

Doris plugin framework supports install/uninstall custom plugins at runtime without restart the Doris service. Users can extend Doris's functionality by developing their own plugins.

For example, the audit plugin worked after a request execution, it can obtain information related to a request (access user, request IP, SQL, etc…) and write the information into the specified table.

Differences from UDF: * UDF is a function used for data calculation when SQL is executed. Plugin is additional function that is used to extend Doris with customized function, such as support different storage engines and different import ways, and plugin doesn't participate in data calculation when executing SQL. * The execution cycle of UDF is limited to a SQL execution. The execution cycle of plugin may be the same as the Doris process. * The usage scene is different. If you need to support special data algorithms when executing SQL, then UDF is recommended, if you need to run custom functions on Doris, or start a background thread to do tasks, then the use of plugin is recommended.

Currently the plugin framework only supports audit plugins.

> Note: Doris plugin framework is an experimental feature, currently only supports FE plugin, and is closed by default, can be opened by FE configuration `plugin_enable = true`.

Plugin

A FE Plugin can be a **.zip package** or a **directory**, which contains at least two parts: the `plugin.properties` and `.jar` files. The `plugin.properties` file is used to describe the plugin information.

The file structure of a Plugin looks like this:

```
# plugin .zip
auditodemo.zip:
    -plugin.properties
    -auditdemo.jar
    -xxx.config
    -data/
    -test_data/

# plugin local directory
auditodemo/:
    -plugin.properties
    -auditdemo.jar
    -xxx.config
    -data/
    -test_data/
```

`plugin.properties` example:

```
### required:
#
# the plugin name
name = audit_plugin_demo
#
# the plugin type
type = AUDIT
#
# simple summary of the plugin
description = just for test
#
# Doris's version, like: 0.11.0
version = 0.11.0

### FE-Plugin optional:
#
# version of java the code is built against
# use the command "java -version" value, like 1.8.0, 9.0.1, 13.0.4
java.version = 1.8.31
#
# the name of the class to load, fully-qualified.
classname = AuditPluginDemo

### BE-Plugin optional:
# the name of the so to load
soName = example.so
```

Write A Plugin

The development environment of the FE plugin depends on the development environment of Doris. So please make sure Doris's compilation and development environment works normally.

`fe_plugins` is the parent module of the fe plugins. It can uniformly manage the third-party library information that the plugin depends on. Adding a plugin can add a submodule implementation under `fe_plugins`.

Create module

We can add a submodule in the `fe_plugins` directory to implement Plugin and create a project:

```
mvn archetype: generate -DarchetypeCatalog = internal -DgroupId = org.apache -DartifactId = doris-fe-test -DinteractiveMode = false
```

The command produces a new mvn project, and a new submodule is automatically added to `fe_plugins/pom.xml`:

```
    .....
    <groupId>org.apache</groupId>
    <artifactId>doris-fe-plugins</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>auditdemo</module>
        # new plugin module
        <module>doris-fe-test</module>
```

```
    </modules>
    .....
```

The new plugin project file structure is as follows:

```
-doris-fe-test/
-pom.xml
-src/
    ---- main/java/org/apache/
    ------- App.java # mvn auto generate, ignore
    ---- test/java/org/apache
```

We will add an assembly folder under main to store `plugin.properties` and `zip.xml`. After completion, the file structure is as follows:

```
-doris-fe-test/
-pom.xml
-src/
---- main/
------ assembly/
-------- plugin.properties
-------- zip.xml
------ java/org/apache/
--------App.java # mvn auto generate, ignore
---- test/java/org/apache
```

Add zip.xml

`zip.xml`, used to describe the content of the final package of the plugin (.jar file, plugin.properties):

```
<assembly>
    <id>plugin</id>
    <formats>
        <format>zip</format>
    </formats>
    <!-IMPORTANT: must be false->
    <includeBaseDirectory>false</includeBaseDirectory>
    <fileSets>
        <fileSet>
            <directory>target</directory>
            <includes>
                <include>*.jar</include>
            </ ncludes>
            <outputDirectory>/</outputDirectory>
        </fileSet>

        <fileSet>
            <directory>src/main/assembly</directory>
            <includes>
                <include>plugin.properties</include>
            </includes>
            <outputDirectory>/</outputDirectory>
        </fileSet>
    </fileSets>
</assembly>
```

Update pom.xml

Then we need to update `pom.xml`, add doris-fe dependency, and modify maven packaging way:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://maven.apache.org/POM/4.0.0"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <groupId>org.apache</groupId>
        <artifactId>doris-fe-plugins</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>auditloader</artifactId>
    <packaging>jar</packaging>

    <dependencies>
        <!-- doris-fe dependencies -->
        <dependency>
            <groupId>org.apache</groupId>
            <artifactId>doris-fe</artifactId>
        </dependency>

        <!-- other dependencies -->
        <dependency>
            ...
        </dependency>
    </dependencies>


    <build>
        <finalName>auditloader</finalName>
        <plugins>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <version>2.4.1</version>
                <configuration>
                    <appendAssemblyId>false</appendAssemblyId>
                    <descriptors>
                        <descriptor>src/main/assembly/zip.xml</descriptor>
                    </descriptors>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>

</project>
```

Implement plugin

Then we can implement Plugin according to the needs. Plugins need to implement the `Plugin` interface. For details, please refer to the `auditdemo` plugin sample code that comes with Doris.

Compile

Before compiling the plugin, you must first execute `sh build.sh --fe` of Doris to complete the compilation of Doris FE.

Finally, execute `sh build_plugin.sh` in the ${DORIS_HOME} path and you will find the `your_plugin_name.zip` file in `fe_plugins/output`

Or you can execute `sh build_plugin.sh --plugin your_plugin_name` to only build your plugin.

Other way

The easiest way, you can implement your plugin by modifying the example `auditdemo`

Deploy

Doris's plugin can be deployed in three ways:

- Http or Https .zip, like `http://xxx.xxxxxx.com/data/plugin.zip`, Doris will download this .zip file. At the same time, the value of md5sum needs to be set in properties, or an md5 file with the same name as the `.zip` file needs to be placed, such as `http://xxx.xxxxxx.com/data/my_plugin.zip.md5`. The content is the MD5 value of the .zip file.
- Local .zip, like `/home/work/data/plugin.zip`. If the plug-in is only used for FE, it needs to be deployed in the same directory of all FE nodes. Otherwise, it needs to be deployed on all FE and BE nodes.
- Local directory, like `/home/work/data/plugin`, .zip decompressed folder. If the plug-in is only used for FE, it needs to be deployed in the same directory of all FE nodes. Otherwise, it needs to be deployed on all FE and BE nodes.

Note: Need to ensure that the plugin .zip file is available in the life cycle of doris!

Install and Uninstall

Install and uninstall the plugin through the install/uninstall statements. More details, see `HELP INSTALL PLUGIN;` `HELP UNINSTALL PLUGIN;` `HELP SHOW PLUGINS;`

```
mysql> install plugin from "/home/users/doris/auditloader.zip";
Query OK, 0 rows affected (0.09 sec)

mysql> show plugins\G
*************************** 1. row ***************************
       Name: auditloader
       Type: AUDIT
Description: load audit log to olap load, and user can view the statistic of queries
    Version: 0.12.0
JavaVersion: 1.8.31
  ClassName: AuditLoaderPlugin
     SoName: NULL
```

```
    Sources: /home/users/doris/auditloader.zip
     Status: INSTALLED
 Properties: {}
*************************** 2. row ***************************
       Name: AuditLogBuilder
       Type: AUDIT
Description: builtin audit logger
    Version: 0.12.0
JavaVersion: 1.8.31
  ClassName: org.apache.doris.qe.AuditLogBuilder
     SoName: NULL
    Sources: Builtin
     Status: INSTALLED
 Properties: {}
2 rows in set (0.00 sec)

mysql> uninstall plugin auditloader;
Query OK, 0 rows affected (0.05 sec)

mysql> show plugins;
Empty set (0.00 sec)
```

Small File Manager

File Manager

Some functions in Doris require some user-defined files. For example, public keys, key files, certificate files and so on are used to access external data sources. The File Manager provides a function that allows users to upload these files in advance and save them in Doris system, which can then be referenced or accessed in other commands.

Noun Interpretation

- BDBJE: Oracle Berkeley DB Java Edition. Distributed embedded database for persistent metadata in FE.
- SmallFileMgr: File Manager. Responsible for creating and maintaining user files.

Basic concepts

Files are files created and saved by users in Doris.

A file is located by `database`, `catalog`, `file_name`. At the same time, each file also has a globally unique ID (file_id), which serves as the identification in the system.

File creation and deletion can only be performed by users with `admin` privileges. A file belongs to a database. Users who have access to a database (queries, imports, modifications, etc.) can use the files created under the database.

Specific operation

File management has three main commands: `CREATE FILE`, `SHOW FILE` and `DROP FILE`, creating, viewing and deleting files respectively. The specific syntax of these three commands can be viewed by connecting to Doris and executing `HELP cmd;`.

CREATE FILE

This statement is used to create and upload a file to the Doris cluster. For details, see CREATE FILE.

Examples:

```
1. Create file ca.pem , classified as kafka

    CREATE FILE "ca.pem"
    PROPERTIES
    (
        "url" = "https://test.bj.bcebos.com/kafka-key/ca.pem",
        "catalog" = "kafka"
    );

2. Create a file client.key, classified as my_catalog
```

```
    CREATE FILE "client.key"
    IN my_database
    PROPERTIES
    (
        "url" = "https://test.bj.bcebos.com/kafka-key/client.key",
        "catalog" = "my_catalog",
        "md5" = "b5bb901bf10f99205b39a46ac3557dd9"
    );
```

SHOW FILE

This statement can view the files that have been created successfully. For details, see SHOW FILE.

Examples:

```
1. View uploaded files in database my_database

    SHOW FILE FROM my_database;
```

DROP FILE

This statement can view and delete an already created file. For specific operations, see DROP FILE.

Examples:

```
1. delete file ca.pem

    DROP FILE "ca.pem" properties("catalog" = "kafka");
```

Implementation details

Create and delete files

When the user executes the `CREATE FILE` command, FE downloads the file from a given URL. The contents of the file are stored in FE memory directly in the form of Base64 encoding. At the same time, the file content and meta-information related to the file will be persisted in BDBJE. All created files, their meta-information and file content reside in FE memory. If the FE goes down and restarts, meta information and file content will also be loaded into memory from the BDBJE. When a file is deleted, the relevant information is deleted directly from FE memory and persistent information is deleted from BDBJE.

Use of documents

If the FE side needs to use the created file, SmallFileMgr will directly save the data in FE memory as a local file, store it in the specified directory, and return the local file path for use.

If the BE side needs to use the created file, BE will download the file content to the specified directory on BE through FE's HTTP interface `api/get_small_file` for use. At the same time, BE also records the information of the files that have been downloaded in memory. When BE requests a file, it first checks whether the local file

exists and verifies it. If the validation passes, the local file path is returned directly. If the validation fails, the local file is deleted and downloaded from FE again. When BE restarts, local files are preloaded into memory.

Use restrictions

Because the file meta-information and content are stored in FE memory. So by default, only files with size less than 1MB can be uploaded. And the total number of files is limited to 100. The configuration items described in the next section can be modified.

Relevant configuration

1. FE configuration

- `Small_file_dir`: The path used to store uploaded files, defaulting to the `small_files/` directory of the FE runtime directory.
- `max_small_file_size_bytes`: A single file size limit in bytes. The default is 1MB. File creation larger than this configuration will be rejected.
- `max_small_file_number`: The total number of files supported by a Doris cluster. The default is 100. When the number of files created exceeds this value, subsequent creation will be rejected.

> If you need to upload more files or increase the size limit of a single file, you can modify the `max_small_file_size_bytes` and `max_small_file_number` parameters by using the `ADMIN SET CONFIG` command. However, the increase in the number and size of files will lead to an increase in FE memory usage.

2. BE configuration

- `Small_file_dir`: The path used to store files downloaded from FE by default is in the `lib/small_files/` directory of the BE runtime directory.

More Help

For more detailed syntax and best practices used by the file manager, see CREATE FILE, DROP FILE and SHOW FILE command manual, you can also enter `HELP CREATE FILE`, `HELP DROP FILE` and `HELP SHOW FILE` in the MySql client command line to get more help information.

Compaction

Compaction

Doris writes data through a structure similar to LSM-Tree, and continuously merges small files into large ordered files through compaction in the background. Compaction handles operations such as deletion and updating.

Appropriately adjusting the compaction strategy can greatly improve load and query efficiency. Doris provides the following compaction strategies for tuning:

Vertical compaction

Vertical compaction is a new compaction algorithm implemented in Doris 1.2.2, which is used to optimize compaction execution efficiency and resource overhead in large-scale and wide table scenarios. It can effectively reduce the memory overhead of compaction and improve the execution speed of compaction. The test results show that the memory consumption by vertical compaction is only 1/10 of the original compaction algorithm, and the compaction rate is increased by 15%.

In vertical compaction, merging by row is changed to merging by column group. The granularity of each merge is changed to column group, which reduces the amount of data involved in single compaction and reduces the memory usage during compaction.

BE configuration： - `enable_vertical_compaction = true` will turn on vertical compaction - `vertical_compaction_num_columns_per_group = 5` The number of columns contained in each column group, by testing, the efficiency and memory usage of a group of 5 columns by default is more friendly - `vertical_compaction_max_segment_size` is used to configure the size of the disk file after vertical compaction, the default value is 268435456 (bytes)

Segment compaction

Segment compaction mainly deals with the large-scale data load. Segment compaction operates during the load process and compact segments inside the job, which is different from normal compaction and vertical compaction. This mechanism can effectively reduce the number of generated segments and avoid the -238 (OLAP_ERR_TOO_MANY_SEGMENTS) errors.

The following features are provided by segment compaction: - reduce the number of segments generated by load - the compacting process is parallel to the load process, which will not increase the load time - memory consumption and computing resources will increase during loading, but the increase is relatively low because it is evenly distributed throughout the long load process. - data after segment compaction will have resource and performance advantages in subsequent queries and normal compaction.

BE configuration: - `enable_segcompaction=true` turn it on. - `segcompaction_batch_size` is used to configure the interval for merging. The default value 10 means that every 10 segment files will trigger a segment compaction. It is recommended to set between 10 - 30. The larger value will increase the memory usage of segment compaction.

Situations where segment compaction is recommended:

- Loading large amounts of data fails at OLAP_ ERR_ TOO_ MANY_ SEGMENTS (errcode - 238) error. Then it is recommended to turn on segment compaction to reduce the quantity of segments during the load process.
- Too many small files are generated during the load process: although the amount of loading data is reasonable, the generation of a large number of small segment files may also fail the load job because of low cardinality or memory constraints that trigger memtable to be flushed in advance. Then it is recommended to turn on this function.
- Query immediately after loading. When the load is just finished and the standard compaction has not finished, large number of segment files will affect the efficiency of subsequent queries. If the user needs to query immediately after loading, it is recommended to turn on this function.
- The pressure of normal compaction is high after loading: segment compaction evenly puts part of the pressure of normal compaction on the loading process. At this time, it is recommended to enable this function.

Situations where segment compaction is not recommended: - When the load operation itself has exhausted memory resources, it is not recommended to use the segment compaction to avoid further increasing memory pressure and causing the load job to fail.

Refer to this link for more information about implementation and test results.

Single replica compaction

By default, compaction for multiple replicas is performed independently, with each replica consuming CPU and IO resources. When single replica compaction is enabled, only one replica performs the compaction. Afterward, the other replicas pull the compacted files from this replica, resulting in CPU resources being consumed only once, saving N - 1 times CPU usage (where N is the number of replicas).

Single replica compaction is specified in the table's PROPERTIES via the parameter `enable_single_replica_compaction`, which is false by default (disabled). To enable it, set the parameter to true.

This parameter can be specified when creating the table or modified later using:

```
ALTER TABLE table_name SET("enable_single_replica_compaction" = "true");
```

Compaction strategy

The compaction strategy determines when and which small files are merged into larger files. Doris currently offers two compaction strategies, specified by the `compaction_policy` parameter in the table properties.

Size-based compaction strategy

The size-based compaction strategy is the default strategy and is suitable for most scenarios.

```
"compaction_policy" = "size_based"
```

Time series compaction strategy

The time series compaction strategy is optimized for scenarios like logs and time-series data. It leverages the time locality of time-series data, merging small files written in adjacent times into larger files. Each file participates in compaction only once, reducing write amplification from repeated compaction.

```
"compaction_policy" = "time_series"
```

The time series compaction strategy is triggered when any of the following conditions are met: - The size of unmerged files exceeds `time_series_compaction_goal_size_mbytes` (default 1 GB). - The number of unmerged files exceeds `time_series_compaction_file_count_threshold` (default 2000). - The time since the last compaction exceeds `time_series_compaction_time_threshold_seconds` (default 1 hour).

These parameters are set in the table's PROPERTIES and can be specified when creating the table or modified later using:

```
ALTER TABLE table_name SET("name" = "value");
```

Compaction concurrency control

Compaction runs in the background and consumes CPU and IO resources. The resource consumption can be controlled by adjusting the number of concurrent compaction threads.

The number of concurrent compaction threads is configured in the BE configuration file, including the following parameters: - `max_base_compaction_threads`: Number of base compaction threads, default is 4. - `max_cumu_compaction_threads`: Number of cumulative compaction threads, default is 10. - `max_single_replica_compaction_threads`: Number of threads for fetching data files during single replica compaction, default is 10.

Using rowsets

rowsets

Name

rowsets

description

`rowsets` is a built-in system table of doris, which is stored under the information_schema database. You can view the current rowsets information of each `BE` through the `rowsets` system table.

The `rowsets` table schema is:

```
MySQL [(none)]> desc information_schema.rowsets;
+------------------------+------------+------+-------+---------+-------+
| Field                  | Type       | Null | Key   | Default | Extra |
+------------------------+------------+------+-------+---------+-------+
| BACKEND_ID             | BIGINT     | Yes  | false | NULL    |       |
| ROWSET_ID              | VARCHAR(*) | Yes  | false | NULL    |       |
| TABLET_ID              | BIGINT     | Yes  | false | NULL    |       |
| ROWSET_NUM_ROWS        | BIGINT     | Yes  | false | NULL    |       |
| TXN_ID                 | BIGINT     | Yes  | false | NULL    |       |
| NUM_SEGMENTS           | BIGINT     | Yes  | false | NULL    |       |
| START_VERSION          | BIGINT     | Yes  | false | NULL    |       |
| END_VERSION            | BIGINT     | Yes  | false | NULL    |       |
| INDEX_DISK_SIZE        | BIGINT     | Yes  | false | NULL    |       |
| DATA_DISK_SIZE         | BIGINT     | Yes  | false | NULL    |       |
| CREATION_TIME          | BIGINT     | Yes  | false | NULL    |       |
| NEWEST_WRITE_TIMESTAMP | BIGINT     | Yes  | false | NULL    |       |
+------------------------+------------+------+-------+---------+-------+
```

Example

```
select * from information_schema.rowsets where BACKEND_ID = 10004 limit 10;
+------------+----------------------------------------------------+-----------+-----------------+--------+-------------+-------------
--+-------------+-----------------+---------------+---------------+----------------------+-----------------------+
| BACKEND_ID | ROWSET_ID                                          | TABLET_ID | ROWSET_NUM_ROWS | TXN_ID | NUM_SEGMENTS |
START_VERSION | END_VERSION | INDEX_DISK_SIZE | DATA_DISK_SIZE | CREATION_TIME | OLDEST_WRITE_TIMESTAMP | NEWEST_WRITE_TIMESTAMP |
+------------+----------------------------------------------------+-----------+-----------------+--------+-------------+-------------
--+-------------+-----------------+---------------+---------------+----------------------+-----------------------+
|      10004 | 020000000000000994847fbd41a42297d7c7a57d3bcb46f8c |     10771 |           66850 |      6 |            1 |
3 |           3 |            2894 |         688855 |    1659964582 |           1659964581 |            1659964581 |
|      10004 | 020000000000008d4847fbd41a42297d7c7a57d3bcb46f8c |     10771 |           66850 |      2 |            1 |
2 |           2 |            2894 |         688855 |    1659964575 |           1659964574 |            1659964574 |
|      10004 | 020000000000000894847fbd41a42297d7c7a57d3bcb46f8c |     10771 |               0 |      0 |            0 |
0 |           1 |               0 |              0 |    1659964567 |           1659964567 |            1659964567 |
|      10004 | 020000000000009a4847fbd41a42297d7c7a57d3bcb46f8c |     10773 |           66639 |      6 |            1 |
3 |           3 |            2897 |         686828 |    1659964582 |           1659964581 |            1659964581 |
|      10004 | 020000000000008e4847fbd41a42297d7c7a57d3bcb46f8c |     10773 |           66639 |      2 |            1 |
2 |           2 |            2897 |         686828 |    1659964575 |           1659964574 |            1659964574 |
|      10004 | 020000000000000884847fbd41a42297d7c7a57d3bcb46f8c |     10773 |               0 |      0 |            0 |
0 |           1 |               0 |              0 |    1659964567 |           1659964567 |            1659964567 |
```

```
|      10004 | 0200000000000000984847fbd41a42297d7c7a57d3bcb46f8c |     10757 |           66413 |      6 |            1 |
3 |           3 |            2893 |         685381 |    1659964582 |            1659964581 |            1659964581 |
|      10004 | 020000000000008c4847fbd41a42297d7c7a57d3bcb46f8c |     10757 |           66413 |      2 |            1 |
2 |           2 |            2893 |         685381 |    1659964575 |            1659964574 |            1659964574 |
|      10004 | 0200000000000000874847fbd41a42297d7c7a57d3bcb46f8c |     10757 |               0 |      0 |            0 |
0 |           1 |               0 |              0 |    1659964567 |            1659964567 |            1659964567 |
|      10004 | 020000000000009c4847fbd41a42297d7c7a57d3bcb46f8c |     10739 |            1698 |      8 |            1 |
3 |           3 |             454 |          86126 |    1659964582 |            1659964582 |            1659964582 |
+------------+--------------------------------------------------+-----------+-----------------+--------+--------------+-------------
--+-------------+-----------------+---------------+---------------+-----------------------+-----------------------+
```

KeyWords

```
rowsets, information_schema
```

Best Practice

Star Schema Benchmark

Star Schema Benchmark

Star Schema Benchmark(SSB) is a lightweight performance test set in the data warehouse scenario. SSB provides a simplified star schema data based on TPC-H, which is mainly used to test the performance of multi-table JOIN query under star schema. In addition, the industry usually flattens SSB into a wide table model (Referred as: SSB flat) to test the performance of the query engine, refer to Clickhouse.

This document mainly introduces the performance of Doris on the SSB 1000G test set.

We tested 13 queries on the SSB standard test dataset based on Apache Doris version 2.0.15.1.

1. Hardware Environment

Hardware	Configuration Instructions
Number of Machines	4 Aliyun Virtual Machine (1FE，3BEs)
CPU	Intel Xeon (Ice Lake) Platinum 8369B 32C
Memory	128G
Disk	Enterprise SSD (PL0)

2. Software Environment

- Doris Deployed 3BEs and 1FE
- Kernel Version: Linux version 5.15.0-101-generic
- OS version: Ubuntu 20.04 LTS (Focal Fossa)
- Doris software version: Apache Doris 2.0.15.1
- JDK: openjdk version “1.8.0_352-352”

3. Test Data Volume

SSB Table Name	Rows	Annotation
lineorder	5,999,989,709	Commodity Order Details
customer	30,000,000	Customer Information
part	2,000,000	Parts Information
supplier	2,000,000	Supplier Information
dates	2,556	Date
lineorder_flat	5,999,989,709	Wide Table after Data Flattening

4. SSB Flat Test Results

Here we use Apache Doris 2.0.15.1 for comparative testing. In the test, we use Query Time(ms) as the main performance indicator. The test results are as follows:

Query	Doris 2.0.15.1 (ms)
q1.1	80
q1.2	10
q1.3	110
q2.1	1680
q2.2	1210
q2.3	1060
q3.1	2010
q3.2	1560
q3.3	600
q3.4	10
q4.1	2380
q4.2	190
q4.3	120
Total	**11020**

5. Standard SSB Test Results

Here we use Apache Doris 2.0.15.1 for comparative testing. In the test, we use Query Time(ms) as the main performance indicator. The test results are as follows:

Query	Doris 2.0.15.1 (ms)
q1.1	330
q1.2	80
q1.3	80
q2.1	1780
q2.2	1970
q2.3	1510
q3.1	4000
q3.2	1720
q3.3	1510
q3.4	160
q4.1	4010

Query	Doris 2.0.15.1 (ms)
q4.2	840
q4.3	400
Total	**19390**

6. Environment Preparation

Please first refer to the official documentation to install and deploy Apache Doris first to obtain a Doris cluster which is working well(including at least 1 FE 1 BE, 1 FE 3 BEs is recommended).

7. Data Preparation

7.1 Download and Install the SSB Data Generation Tool.

Execute the following script to download and compile the ssb-tools tool.

```
sh bin/build-ssb-dbgen.sh
```

After successful installation, the `dbgen` binary will be generated under the `ssb-dbgen/` directory.

7.2 Generate SSB Test Set

Execute the following script to generate the SSB dataset:

```
sh bin/gen-ssb-data.sh -s 1000
```

> Note 1: Check the script help via `sh gen-ssb-data.sh -h`.
>
> Note 2: The data will be generated under the `ssb-data/` directory with the suffix `.tbl`. The total file size is about 600GB and may need a few minutes to an hour to generate.
>
> Note 3: A standard test data set of 100G is generated by default.

7.3 Create Table

7.3.1 Prepare the `doris-cluster.conf` File.

Before import the script, you need to write the FE's ip port and other information in the `doris-cluster.conf` file.

The file is located under `${DORIS_HOME}/tools/ssb-tools/conf/`.

The content of the file includes FE's ip, HTTP port, user name, password and the DB name of the data to be imported:

```
# Any of FE host
export FE_HOST='127.0.0.1'
# http_port in fe.conf
export FE_HTTP_PORT=8030
# query_port in fe.conf
export FE_QUERY_PORT=9030
# Doris username
export USER='root'
# Doris password
export PASSWORD=''
# The database where SSB tables located
export DB='ssb'
```

7.3.2 Execute the Following Script to Generate and Create the SSB Table:

```
sh bin/create-ssb-tables.sh -s 1000
```

Or copy the table creation statements in create-ssb-tables.sql and create-ssb-flat-table.sql and then execute them in the MySQL client.

7.4 Import data

We use the following command to complete all data import of SSB test set and SSB FLAT wide table data synthesis and then import into the table.

```
 sh bin/load-ssb-data.sh
```

7.5 Checking Imported data

```
select count(*) from part;
select count(*) from customer;
select count(*) from supplier;
select count(*) from dates;
select count(*) from lineorder;
select count(*) from lineorder_flat;
```

7.6 Query Test

- SSB-Flat Query Statement: ssb-flat-queries
- Standard SSB Queries: ssb-queries

7.6.1 SSB FLAT Test for SQL

```
--Q1.1
SELECT SUM(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue
FROM lineorder_flat
WHERE
    LO_ORDERDATE >= 19930101
    AND LO_ORDERDATE <= 19931231
    AND LO_DISCOUNT BETWEEN 1 AND 3
```

```
    AND LO_QUANTITY < 25;

--Q1.2
SELECT SUM(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue
FROM lineorder_flat
WHERE
    LO_ORDERDATE >= 19940101
  AND LO_ORDERDATE <= 19940131
  AND LO_DISCOUNT BETWEEN 4 AND 6
  AND LO_QUANTITY BETWEEN 26 AND 35;

--Q1.3
SELECT SUM(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue
FROM lineorder_flat
WHERE
    weekofyear(LO_ORDERDATE) = 6
  AND LO_ORDERDATE >= 19940101
  AND LO_ORDERDATE <= 19941231
  AND LO_DISCOUNT BETWEEN 5 AND 7
  AND LO_QUANTITY BETWEEN 26 AND 35;

--Q2.1
SELECT
    SUM(LO_REVENUE), (LO_ORDERDATE DIV 10000) AS YEAR,
    P_BRAND
FROM lineorder_flat
WHERE P_CATEGORY = 'MFGR#12' AND S_REGION = 'AMERICA'
GROUP BY YEAR, P_BRAND
ORDER BY YEAR, P_BRAND;

--Q2.2
SELECT
    SUM(LO_REVENUE), (LO_ORDERDATE DIV 10000) AS YEAR,
    P_BRAND
FROM lineorder_flat
WHERE
    P_BRAND >= 'MFGR#2221'
  AND P_BRAND <= 'MFGR#2228'
  AND S_REGION = 'ASIA'
GROUP BY YEAR, P_BRAND
ORDER BY YEAR, P_BRAND;

--Q2.3
SELECT
    SUM(LO_REVENUE), (LO_ORDERDATE DIV 10000) AS YEAR,
    P_BRAND
FROM lineorder_flat
WHERE
    P_BRAND = 'MFGR#2239'
  AND S_REGION = 'EUROPE'
GROUP BY YEAR, P_BRAND
ORDER BY YEAR, P_BRAND;

--Q3.1
SELECT
    C_NATION,
    S_NATION, (LO_ORDERDATE DIV 10000) AS YEAR,
    SUM(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE
    C_REGION = 'ASIA'
  AND S_REGION = 'ASIA'
  AND LO_ORDERDATE >= 19920101
  AND LO_ORDERDATE <= 19971231
GROUP BY C_NATION, S_NATION, YEAR
```

```
ORDER BY YEAR ASC, revenue DESC;

--Q3.2
SELECT
    C_CITY,
    S_CITY, (LO_ORDERDATE DIV 10000) AS YEAR,
    SUM(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE
    C_NATION = 'UNITED STATES'
  AND S_NATION = 'UNITED STATES'
  AND LO_ORDERDATE >= 19920101
  AND LO_ORDERDATE <= 19971231
GROUP BY C_CITY, S_CITY, YEAR
ORDER BY YEAR ASC, revenue DESC;

--Q3.3
SELECT
    C_CITY,
    S_CITY, (LO_ORDERDATE DIV 10000) AS YEAR,
    SUM(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE
    C_CITY IN ('UNITED KI1', 'UNITED KI5')
  AND S_CITY IN ('UNITED KI1', 'UNITED KI5')
  AND LO_ORDERDATE >= 19920101
  AND LO_ORDERDATE <= 19971231
GROUP BY C_CITY, S_CITY, YEAR
ORDER BY YEAR ASC, revenue DESC;

--Q3.4
SELECT
    C_CITY,
    S_CITY, (LO_ORDERDATE DIV 10000) AS YEAR,
    SUM(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE
    C_CITY IN ('UNITED KI1', 'UNITED KI5')
  AND S_CITY IN ('UNITED KI1', 'UNITED KI5')
  AND LO_ORDERDATE >= 19971201
  AND LO_ORDERDATE <= 19971231
GROUP BY C_CITY, S_CITY, YEAR
ORDER BY YEAR ASC, revenue DESC;

--Q4.1
SELECT (LO_ORDERDATE DIV 10000) AS YEAR,
    C_NATION,
    SUM(LO_REVENUE - LO_SUPPLYCOST) AS profit
FROM lineorder_flat
WHERE
    C_REGION = 'AMERICA'
  AND S_REGION = 'AMERICA'
  AND P_MFGR IN ('MFGR#1', 'MFGR#2')
GROUP BY YEAR, C_NATION
ORDER BY YEAR ASC, C_NATION ASC;

--Q4.2
SELECT (LO_ORDERDATE DIV 10000) AS YEAR,
    S_NATION,
    P_CATEGORY,
    SUM(LO_REVENUE - LO_SUPPLYCOST) AS profit
FROM lineorder_flat
WHERE
    C_REGION = 'AMERICA'
  AND S_REGION = 'AMERICA'
```

```
  AND LO_ORDERDATE >= 19970101
  AND LO_ORDERDATE <= 19981231
  AND P_MFGR IN ('MFGR#1', 'MFGR#2')
GROUP BY YEAR, S_NATION, P_CATEGORY
ORDER BY
    YEAR ASC,
    S_NATION ASC,
    P_CATEGORY ASC;

--Q4.3
SELECT (LO_ORDERDATE DIV 10000) AS YEAR,
    S_CITY,
    P_BRAND,
    SUM(LO_REVENUE - LO_SUPPLYCOST) AS profit
FROM lineorder_flat
WHERE
    S_NATION = 'UNITED STATES'
  AND LO_ORDERDATE >= 19970101
  AND LO_ORDERDATE <= 19981231
  AND P_CATEGORY = 'MFGR#14'
GROUP BY YEAR, S_CITY, P_BRAND
ORDER BY YEAR ASC, S_CITY ASC, P_BRAND ASC;
```

7.6.2 SSB Standard Test for SQL

```
--Q1.1
SELECT SUM(lo_extendedprice * lo_discount) AS REVENUE
FROM lineorder, dates
WHERE
    lo_orderdate = d_datekey
  AND d_year = 1993
  AND lo_discount BETWEEN 1 AND 3
  AND lo_quantity < 25;

--Q1.2
SELECT SUM(lo_extendedprice * lo_discount) AS REVENUE
FROM lineorder, dates
WHERE
    lo_orderdate = d_datekey
  AND d_yearmonth = 'Jan1994'
  AND lo_discount BETWEEN 4 AND 6
  AND lo_quantity BETWEEN 26 AND 35;

--Q1.3
SELECT
    SUM(lo_extendedprice * lo_discount) AS REVENUE
FROM lineorder, dates
WHERE
    lo_orderdate = d_datekey
  AND d_weeknuminyear = 6
  AND d_year = 1994
  AND lo_discount BETWEEN 5 AND 7
  AND lo_quantity BETWEEN 26 AND 35;

--Q2.1
SELECT SUM(lo_revenue), d_year, p_brand
FROM lineorder, dates, part, supplier
WHERE
    lo_orderdate = d_datekey
  AND lo_partkey = p_partkey
  AND lo_suppkey = s_suppkey
  AND p_category = 'MFGR#12'
```

```
  AND s_region = 'AMERICA'
GROUP BY d_year, p_brand
ORDER BY p_brand;

--Q2.2
SELECT SUM(lo_revenue), d_year, p_brand
FROM lineorder, dates, part, supplier
WHERE
    lo_orderdate = d_datekey
  AND lo_partkey = p_partkey
  AND lo_suppkey = s_suppkey
  AND p_brand BETWEEN 'MFGR#2221' AND 'MFGR#2228'
  AND s_region = 'ASIA'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;

--Q2.3
SELECT SUM(lo_revenue), d_year, p_brand
FROM lineorder, dates, part, supplier
WHERE
    lo_orderdate = d_datekey
  AND lo_partkey = p_partkey
  AND lo_suppkey = s_suppkey
  AND p_brand = 'MFGR#2239'
  AND s_region = 'EUROPE'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;

--Q3.1
SELECT
    c_nation,
    s_nation,
    d_year,
    SUM(lo_revenue) AS REVENUE
FROM customer, lineorder, supplier, dates
WHERE
    lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_orderdate = d_datekey
  AND c_region = 'ASIA'
  AND s_region = 'ASIA'
  AND d_year >= 1992
  AND d_year <= 1997
GROUP BY c_nation, s_nation, d_year
ORDER BY d_year ASC, REVENUE DESC;

--Q3.2
SELECT
    c_city,
    s_city,
    d_year,
    SUM(lo_revenue) AS REVENUE
FROM customer, lineorder, supplier, dates
WHERE
    lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_orderdate = d_datekey
  AND c_nation = 'UNITED STATES'
  AND s_nation = 'UNITED STATES'
  AND d_year >= 1992
  AND d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year ASC, REVENUE DESC;

--Q3.3
```

```
SELECT
    c_city,
    s_city,
    d_year,
    SUM(lo_revenue) AS REVENUE
FROM customer, lineorder, supplier, dates
WHERE
    lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_orderdate = d_datekey
  AND (
           c_city = 'UNITED KI1'
        OR c_city = 'UNITED KI5'
    )
  AND (
           s_city = 'UNITED KI1'
        OR s_city = 'UNITED KI5'
    )
  AND d_year >= 1992
  AND d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year ASC, REVENUE DESC;

--Q3.4
SELECT
    c_city,
    s_city,
    d_year,
    SUM(lo_revenue) AS REVENUE
FROM customer, lineorder, supplier, dates
WHERE
    lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_orderdate = d_datekey
  AND (
           c_city = 'UNITED KI1'
        OR c_city = 'UNITED KI5'
    )
  AND (
           s_city = 'UNITED KI1'
        OR s_city = 'UNITED KI5'
    )
  AND d_yearmonth = 'Dec1997'
GROUP BY c_city, s_city, d_year
ORDER BY d_year ASC, REVENUE DESC;

--Q4.1
SELECT
    d_year,
    c_nation,
    SUM(lo_revenue - lo_supplycost) AS PROFIT
FROM dates, customer, supplier, part, lineorder
WHERE
    lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_partkey = p_partkey
  AND lo_orderdate = d_datekey
  AND c_region = 'AMERICA'
  AND s_region = 'AMERICA'
  AND (
           p_mfgr = 'MFGR#1'
        OR p_mfgr = 'MFGR#2'
    )
GROUP BY d_year, c_nation
ORDER BY d_year, c_nation;
```

```
--Q4.2
SELECT
    d_year,
    s_nation,
    p_category,
    SUM(lo_revenue - lo_supplycost) AS PROFIT
FROM dates, customer, supplier, part, lineorder
WHERE
    lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_partkey = p_partkey
  AND lo_orderdate = d_datekey
  AND c_region = 'AMERICA'
  AND s_region = 'AMERICA'
  AND (
           d_year = 1997
       OR d_year = 1998
    )
  AND (
           p_mfgr = 'MFGR#1'
       OR p_mfgr = 'MFGR#2'
    )
GROUP BY d_year, s_nation, p_category
ORDER BY d_year, s_nation, p_category;

--Q4.3
SELECT
    d_year,
    s_city,
    p_brand,
    SUM(lo_revenue - lo_supplycost) AS PROFIT
FROM dates, customer, supplier, part, lineorder
WHERE
    lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_partkey = p_partkey
  AND lo_orderdate = d_datekey
  AND s_nation = 'UNITED STATES'
  AND (
           d_year = 1997
       OR d_year = 1998
    )
  AND p_category = 'MFGR#14'
GROUP BY d_year, s_city, p_brand
ORDER BY d_year, s_city, p_brand;
```

TPC-H Benchmark

TPC-H Benchmark

TPC-H is a decision support benchmark (Decision Support Benchmark), which consists of a set of business-oriented special query and concurrent data modification. The data that is queried and populates the database has broad industry relevance. This benchmark demonstrates a decision support system that examines large amounts of data, executes highly complex queries, and answers key business questions. The performance index reported by TPC-H is called TPC-H composite query performance index per hour (QphH@Size), which reflects multiple aspects of the system's ability to process queries. These aspects include the database size chosen when executing the query, the query processing capability when the query is submitted by a single stream, and the query throughput when the query is submitted by many concurrent users.

This document mainly introduces the performance of Doris on the TPC-H 1000G test set.

On 22 queries on the TPC-H standard test data set, we conducted a comparison test based on Apache Doris 2.1.7-rc03 and Apache Doris 2.0.15.1 versions.

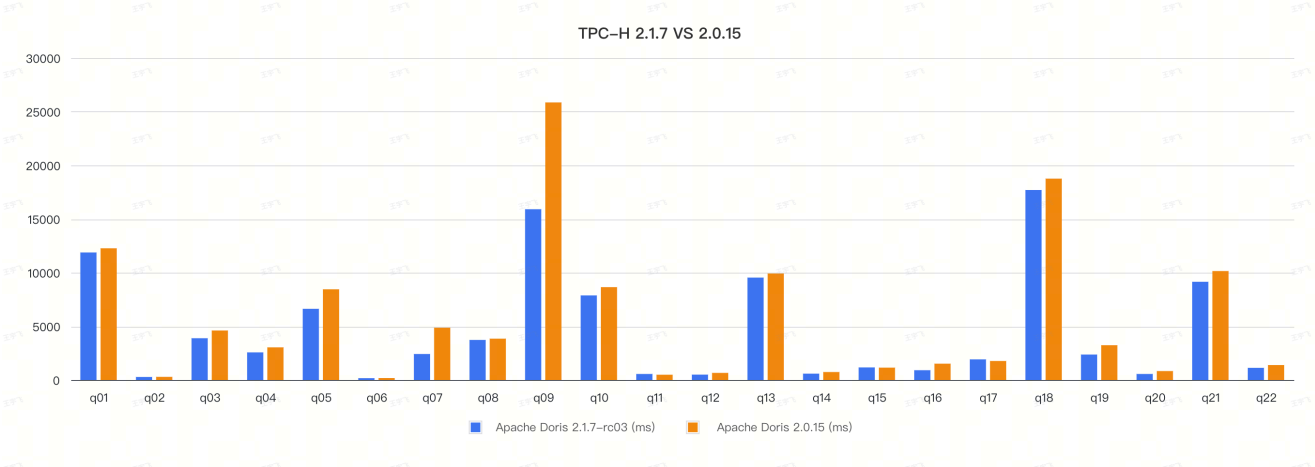


image-20220614114351241

1. Hardware Environment

Hardware	Configuration Instructions
Number of Machines	4 Aliyun Virtual Machine (1FE，3BEs)
CPU	Intel Xeon (Ice Lake) Platinum 8369B 32C
Memory	128G
Disk	Enterprise SSD (PL0)

2. Software Environment

- Doris Deployed 3BEs and 1FE
- Kernel Version: Linux version 5.15.0-101-generic

- OS version: Ubuntu 20.04 LTS (Focal Fossa)
- Doris software version: Apache Doris 2.1.7-rc03, Apache Doris 2.0.15.1
- JDK: openjdk version “1.8.0_352-352”

3. Test Data Volume

The TPC-H 1000G data generated by the simulation of the entire test are respectively imported into Apache Doris 2.1.7-rc03 and Apache Doris 2.0.15.1 for testing. The following is the relevant description and data volume of the table.

TPC-H Table Name	Rows	Annotation
REGION	5	Region
NATION	25	Nation
SUPPLIER	10,000,000	Supplier
PART	200,000,000	Parts
PARTSUPP	800,000,000	Parts Supply
CUSTOMER	150,000,000	Customer
ORDERS	1,500,000,000	Orders
LINEITEM	5,999,989,709	Order Details

4. Test SQL

TPC-H 22 test query statements : TPCH-Query-SQL

5. Test Results

Here we use Apache Doris 2.1.7-rc03 and Apache Doris 2.0.15.1 for comparative testing. In the test, we use Query Time(ms) as the main performance indicator. The test results are as follows:

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
Q1	11880	12270
Q2	280	290
Q3	3890	4610
Q4	2570	3040
Q5	6630	8450
Q6	170	180
Q7	2420	4870
Q8	3730	3850
Q9	15910	25860
Q10	7880	8650

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
Q11	560	490
Q12	500	660
Q13	9540	9920
Q14	590	740
Q15	1170	1150
Q16	910	1520
Q17	1920	1770
Q18	17700	18760
Q19	2370	3240
Q20	560	830
Q21	9150	10150
Q22	1130	1390
Total	**101460**	**122690**

6. Environmental Preparation

Please refer to the official document to install and deploy Doris to obtain a normal running Doris cluster (at least 1 FE 1 BE, 1 FE 3 BE is recommended).

7. Data Preparation

7.1 Download and Install TPC-H Data Generation Tool

Execute the following script to download and compile the tpch-tools tool.

```
sh bin/build-tpch-dbgen.sh
```

After successful installation, the `dbgen` binary will be generated under the `TPC-H_Tools_v3.0.0/` directory.

7.2 Generating the TPC-H Test Set

Execute the following script to generate the TPC-H dataset:

```
sh bin/gen-tpch-data.sh -s 1000
```

> Note 1: Check the script help via `sh gen-tpch-data.sh -h`.
>
> Note 2: The data will be generated under the `tpch-data/` directory with the suffix `.tbl`. The total file size is about 1000GB and may need a few minutes to an hour to generate.

> Note 3: A standard test data set of 100G is generated by default.

7.3 Create Table

7.3.1 Prepare the `doris-cluster.conf` File

Before import the script, you need to write the FE's ip port and other information in the `doris-cluster.conf` file.

The file is located under `${DORIS_HOME}/tools/tpch-tools/conf/` .

The content of the file includes FE's ip, HTTP port, user name, password and the DB name of the data to be imported:

```
# Any of FE host
export FE_HOST='127.0.0.1'
# http_port in fe.conf
export FE_HTTP_PORT=8030
# query_port in fe.conf
export FE_QUERY_PORT=9030
# Doris username
export USER='root'
# Doris password
export PASSWORD=''
# The database where TPC-H tables located
export DB='tpch'
```

Execute the Following Script to Generate and Create TPC-H Table

```
sh bin/create-tpch-tables.sh -s 1000
```

Or copy the table creation statement in create-tpch-tables.sql and execute it in Doris.

7.4 Import Data

Please perform data import with the following command:

```
sh bin/load-tpch-data.sh
```

7.5 Check Imported Data

Execute the following SQL statement to check that the imported data is consistent with the above data.

```
select count(*)  from  lineitem;
select count(*)  from  orders;
select count(*)  from  partsupp;
select count(*)  from  part;
select count(*)  from  customer;
select count(*)  from  supplier;
select count(*)  from  nation;
```

```
select count(*)  from  region;
select count(*)  from  revenue0;
```

7.6 Query Test

7.6.1 Executing Query Scripts

Execute the above test SQL or execute the following command

```
sh bin/run-tpch-queries.sh -s 1000
```

7.6.2 Single SQL Execution

The following is the SQL statement used in the test, you can also get the latest SQL from the code base.

```
--Q1
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date '1998-12-01' - interval '90' day
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;

--Q2
select
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 15
```

```
    and p_type like '%BRASS'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'EUROPE'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'EUROPE'
)
order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey
limit 100;

--Q3
select
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'BUILDING'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < date '1995-03-15'
    and l_shipdate > date '1995-03-15'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate
limit 10;

--Q4
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= date '1993-07-01'
    and o_orderdate < date '1993-07-01' + interval '3' month
    and exists (
        select
            *
        from
            lineitem
```

```
        where
                l_orderkey = o_orderkey
          and l_commitdate < l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;

--Q5
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'ASIA'
    and o_orderdate >= date '1994-01-01'
    and o_orderdate < date '1994-01-01' + interval '1' year
group by
    n_name
order by
    revenue desc;

--Q6
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '1994-01-01'
    and l_shipdate < date '1994-01-01' + interval '1' year
    and l_discount between .06 - 0.01 and .06 + 0.01
    and l_quantity < 24;

--Q7
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            extract(year from l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
```

```
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey = n1.n_nationkey
            and c_nationkey = n2.n_nationkey
            and (
                (n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
                or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
            )
            and l_shipdate between date '1995-01-01' and date '1996-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;

--Q8

select
    o_year,
    sum(case
        when nation = 'BRAZIL' then volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey = n1.n_nationkey
            and n1.n_regionkey = r_regionkey
            and r_name = 'AMERICA'
            and s_nationkey = n2.n_nationkey
            and o_orderdate between date '1995-01-01' and date '1996-12-31'
            and p_type = 'ECONOMY ANODIZED STEEL'
    ) as all_nations
group by
    o_year
order by
    o_year;

--Q9
select
    nation,
```

```
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
        from
            part,
            supplier,
            lineitem,
            partsupp,
            orders,
            nation
        where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%green%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;

--Q10
select
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= date '1993-10-01'
    and o_orderdate < date '1993-10-01' + interval '3' month
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,
    c_address,
    c_comment
order by
    revenue desc
limit 20;
```

```
--Q11
select
    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'GERMANY'
group by
    ps_partkey having
    sum(ps_supplycost * ps_availqty) > (
        select
        sum(ps_supplycost * ps_availqty) * 0.000002
        from
            partsupp,
            supplier,
            nation
        where
            ps_suppkey = s_suppkey
            and s_nationkey = n_nationkey
            and n_name = 'GERMANY'
    )
order by
    value desc;

--Q12
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
            then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
            then 1
        else 0
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('MAIL', 'SHIP')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= date '1994-01-01'
    and l_receiptdate < date '1994-01-01' + interval '1' year
group by
    l_shipmode
order by
    l_shipmode;

--Q13
select
    c_count,
    count(*) as custdist
from
```

```
    (
        select
            c_custkey,
            count(o_orderkey) as c_count
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%special%requests%'
        group by
            c_custkey
    ) as c_orders
group by
    c_count
order by
    custdist desc,
    c_count desc;

--Q14
select
    100.00 * sum(case
        when p_type like 'PROMO%'
            then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= date '1995-09-01'
    and l_shipdate < date '1995-09-01' + interval '1' month;

--Q15
select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue0
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from
            revenue0
    )
order by
    s_suppkey;

--Q16
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#45'
```

```
    and p_type not like 'MEDIUM POLISHED%'
    and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;

--Q17
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#23'
    and p_container = 'MED BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );

--Q18
select
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey  in  (
        select
            l_orderkey
        from
            lineitem
        group  by
            l_orderkey  having
                sum(l_quantity)  >  300
    )
    and  c_custkey  =  o_custkey
    and  o_orderkey  =  l_orderkey
group  by
    c_name,
```

```
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order  by
    o_totalprice  desc,
    o_orderdate
limit  100;


--Q19
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#12'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 1 and l_quantity <= 1 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#23'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 10 and l_quantity <= 10 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#34'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 20 and l_quantity <= 20 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );

--Q20
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
```

```
                    part
                where
                        p_name like 'forest%'
            )
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
                    l_partkey = ps_partkey
                    and l_suppkey = ps_suppkey
                    and l_shipdate >= date '1994-01-01'
                    and l_shipdate < date '1994-01-01' + interval '1' year
            )
    )
    and s_nationkey = n_nationkey
    and n_name = 'CANADA'
order by
    s_name;

--Q21
select
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
                l2.l_orderkey = l1.l_orderkey
          and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
                l3.l_orderkey = l1.l_orderkey
          and l3.l_suppkey <> l1.l_suppkey
          and l3.l_receiptdate > l3.l_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'SAUDI ARABIA'
group by
    s_name
order by
    numwait desc,
    s_name
limit 100;

--Q22
select
```

```
    cntrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substring(c_phone, 1, 2) as cntrycode,
            c_acctbal
        from
            customer
        where
            substring(c_phone, 1, 2) in
            ('13', '31', '23', '29', '30', '18', '17')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer
                where
                    c_acctbal > 0.00
                    and substring(c_phone, 1, 2) in
                      ('13', '31', '23', '29', '30', '18', '17')
            )
            and not exists (
                select
                    *
                from
                    orders
                where
                    o_custkey = c_custkey
            )
    ) as custsale
group by
    cntrycode
order by
    cntrycode;
```

TPC-DS Benchmark

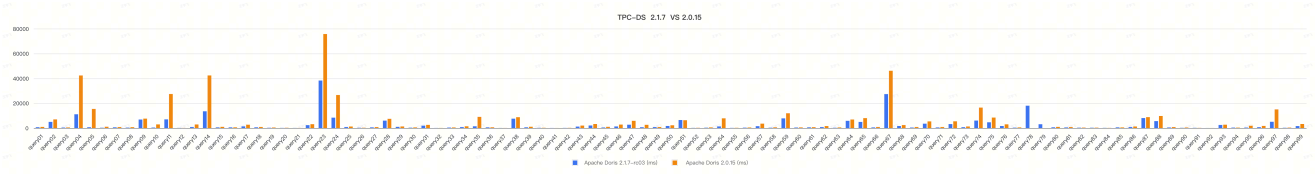
TPC-DS Benchmark

TPC-DS (Transaction Processing Performance Council Decision Support Benchmark) is a benchmark test that focuses on decision support and aims to evaluate the performance of data warehousing and analytics systems. It was developed by the Transaction Processing Performance Council (TPC) organization to compare the capabilities of different systems in handling complex queries and large-scale data analysis.

The design goal of TPC-DS is to simulate complex decision support workloads in the real world. It tests the performance of systems through a series of complex queries and data operations, including joins, aggregations, sorting, filtering, subqueries, and more. These query patterns cover various scenarios ranging from simple to complex, such as report generation, data mining, and OLAP (Online Analytical Processing).

This document mainly introduces the performance of Doris on the TPC-DS 1000G test set.

On 99 queries on the TPC-DS standard test data set, we conducted a comparison test based on Apache Doris 2.1.7-rc03 and Apache Doris 2.0.15.1 versions.



TPCDS_1000G

1. Hardware Environment

Hardware	Configuration Instructions
Number of Machines	4 Aliyun Virtual Machine (1FE，3BEs)
CPU	Intel Xeon (Ice Lake) Platinum 8369B 32C
Memory	128G
Disk	Enterprise SSD (PL0)

2. Software Environment

- Doris Deployed 3BEs and 1FE
- Kernel Version: Linux version 5.15.0-101-generic
- OS version: Ubuntu 20.04 LTS (Focal Fossa)
- Doris software version: Apache Doris 2.1.7-rc03, Apache Doris 2.0.15.1
- JDK: openjdk version “1.8.0_352-352”

3. Test Data Volume

The TPC-DS 1000G data generated by the simulation of the entire test are respectively imported into Apache Doris 2.1.7-rc03 and Apache Doris 2.0.15.1 for testing. The following is the relevant description and data volume of the table.

TPC-DS Table Name	Rows
customer_demographics	1,920,800
reason	65
warehouse	20
date_dim	73,049
catalog_sales	1,439,980,416
call_center	42
inventory	783,000,000
catalog_returns	143,996,756
household_demographics	7,200
customer_address	6,000,000
income_band	20
catalog_page	30,000
item	300,000
web_returns	71,997,522
web_site	54
promotion	1,500
web_sales	720,000,376
store	1,002
web_page	3,000
time_dim	86,400
store_returns	287,999,764
store_sales	2,879,987,999
ship_mode	20
customer	12,000,000

4. Test SQL

TPC-DS 99 test query statements : TPC-DS-Query-SQL

5. Test Results

Here we use Apache Doris 2.1.7-rc03 and Apache Doris 2.0.15.1 for comparative testing. In the test, we use Query Time(ms) as the main performance indicator. The test results are as follows: (Apache Doris 2.0.15.1 q78 q79 failed to execute due to lack of latest memory optimization and was removed when calculating the total sum)

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
query01	630	890
query02	4930	6930
query03	360	460
query04	11070	42320
query05	620	15360
query06	220	1020
query07	550	750
query08	330	670
query09	6830	7550
query10	370	2900
query11	6960	27380
query12	100	80
query13	790	2860
query14	13470	42340
query15	510	940
query16	520	550
query17	1310	2650
query18	560	820
query19	200	400
query20	100	190
query21	80	80
query22	2300	3070
query23	38240	75260
query24	8340	26580
query25	780	1190
query26	200	220
query27	530	750
query28	5940	7400

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
query29	940	1250
query30	270	490
query31	1890	2530
query32	60	70
query33	350	450
query34	750	1380
query35	1370	8970
query36	530	570
query37	60	60
query38	7520	8710
query39	560	1010
query40	150	180
query41	50	40
query42	100	140
query43	1150	1960
query44	2020	3220
query45	430	960
query46	1250	2760
query47	2660	5790
query48	630	2570
query49	730	800
query50	1640	2200
query51	6430	6270
query52	110	160
query53	250	490
query54	1280	7790
query55	110	160
query56	290	410
query57	1480	3510
query58	240	550
query59	7760	11870
query60	380	490
query61	540	670

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
query62	740	1560
query63	210	460
query64	5790	6840
query65	4900	7960
query66	480	810
query67	27320	46110
query68	1600	2380
query69	380	800
query70	3480	5330
query71	460	790
query72	3160	5390
query73	660	1250
query74	5990	16450
query75	4610	8410
query76	1590	2950
query77	300	480
query78	17970	-
query79	3040	-
query80	570	910
query81	460	760
query82	270	330
query83	220	290
query84	130	110
query85	520	470
query86	760	1220
query87	800	8760
query88	5560	9690
query89	430	750
query90	150	400
query91	150	120
query92	40	40
query93	2440	2670
query94	340	310

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
query95	350	1810
query96	660	1680
query97	5020	14990
query98	190	330
query99	1560	3230
Total	**261320**	**507380**

6. Environmental Preparation

Please refer to the official document to install and deploy Doris to obtain a normal running Doris cluster (at least 1 FE 1 BE, 1 FE 3 BE is recommended).

7. Data Preparation

7.1 Download and Install TPC-DS Data Generation Tool

Execute the following script to download and compile the tpcds-tools tool.

```
sh bin/build-tpcds-dbgen.sh
```

7.2 Generating the TPC-DS Test Set

Execute the following script to generate the TPC-H dataset:

```
sh bin/gen-tpcds-data.sh -s 1000
```

> Note 1: Check the script help via `sh gen-tpcds-data.sh -h`.
>
> Note 2: The data will be generated under the `tpcds-data/` directory with the suffix `.dat`. The total file size is about 1000GB and may need a few minutes to an hour to generate.
>
> Note 3: A standard test data set of 100G is generated by default.

7.3 Create Table

7.3.1 Prepare the `doris-cluster.conf` File

Before import the script, you need to write the FE's ip port and other information in the `doris-cluster.conf` file.

The file is located under `${DORIS_HOME}/tools/tpcds-tools/conf/` .

The content of the file includes FE's ip, HTTP port, user name, password and the DB name of the data to be imported:

```
# Any of FE host
export FE_HOST='127.0.0.1'
# http_port in fe.conf
export FE_HTTP_PORT=8030
# query_port in fe.conf
export FE_QUERY_PORT=9030
# Doris username
export USER='root'
# Doris password
export PASSWORD=''
# The database where TPC-H tables located
export DB='tpcds'
```

Execute the Following Script to Generate and Create TPC-H Table

```
sh bin/create-tpcds-tables.sh -s 1000
```

Or copy the table creation statement in create-tpcds-tables and execute it in Doris.

7.4 Import Data

Please perform data import with the following command:

```
sh bin/load-tpcds-data.sh
```

7.5 Query Test

7.5.1 Executing Query Scripts

Execute the above test SQL or execute the following command

```
sh bin/run-tpcds-queries.sh -s 1000
```

7.5.2 Single SQL Execution

You can also retrieve the latest SQL from the code repository. The address for the latest test query statements of TPC-DS.

Spark Doris Connector

Spark Doris Connector

Spark Doris Connector can support reading data stored in Doris and writing data to Doris through Spark.

Github: https://github.com/apache/doris-spark-connector

- Support reading data from `Doris`.
- Support `Spark DataFrame` batch/stream writing data to `Doris`
- You can map the `Doris` table to`DataFrame` or `RDD`, it is recommended to use`DataFrame`.
- Support the completion of data filtering on the `Doris` side to reduce the amount of data transmission.

Version Compatibility

Connector	Spark	Doris	Java	Scala
1.3.2	3.4 ~ 3.1, 2.4, 2.3	1.0 +	8	2.12, 2.11
1.2.0	3.2, 3.1, 2.3	1.0 +	8	2.12, 2.11
1.1.0	3.2, 3.1, 2.3	1.0 +	8	2.12, 2.11
1.0.1	3.1, 2.3	0.12 - 0.15	8	2.12, 2.11

How To Use

Maven

```
<dependency>
    <groupId>org.apache.doris</groupId>
    <artifactId>spark-doris-connector-3.4_2.12</artifactId>
    <version>1.3.2</version>
</dependency>
```

Note

1. Please replace the corresponding Connector version according to different Spark and Scala versions.
2. You can also download the relevant version jar package from here.

Compile

When compiling, you can directly run `sh build.sh`, for details, please refer to here.

After successful compilation, the target jar package will be generated in the `dist` directory, such as: spark-doris-connector-3.2_2.12-1.2.0-SNAPSHOT.jar. Copy this file to the `ClassPath` of `Spark` to use `Spark-Doris-Connector`. For example, for `Spark` running in `Local` mode, put this file in the `jars/` folder. For `Spark` running in `Yarn` cluster mode, put this file in the pre-deployment package. You can also

2. Execute in the source code directory:

`sh build.sh` Enter the Scala and Spark versions you need to compile according to the prompts.

After successful compilation, the target jar package will be generated in the `dist` directory, such as: `spark-doris-connector-3.2_2.12-1.2.0-SNAPSHOT.jar`. Copy this file to the `ClassPath` of `Spark` to use `Spark-Doris-Connector`.

For example, if `Spark` is running in `Local` mode, put this file in the `jars/` folder. If `Spark` is running in `Yarn` cluster mode, put this file in the pre-deployment package.

For example, upload `spark-doris-connector-3.2_2.12-1.2.0-SNAPSHOT.jar` to hdfs and add the Jar package path on hdfs to the `spark.yarn.jars` parameter

```
1. Upload `spark-doris-connector-3.2_2.12-1.2.0-SNAPSHOT.jar` to hdfs.

hdfs dfs -mkdir /spark-jars/
hdfs dfs -put /your_local_path/spark-doris-connector-3.2_2.12-1.2.0-SNAPSHOT.jar /spark-jars/

2. Add the `spark-doris-connector-3.2_2.12-1.2.0-SNAPSHOT.jar` dependency in the cluster.
spark.yarn.jars=hdfs:///spark-jars/spark-doris-connector-3.2_2.12-1.2.0-SNAPSHOT.jar
```

Example

Read

SQL

```
CREATE
TEMPORARY VIEW spark_doris
USING doris
OPTIONS(
  "table.identifier"="$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME",
  "fenodes"="$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT",
  "user"="$YOUR_DORIS_USERNAME",
  "password"="$YOUR_DORIS_PASSWORD"
);

SELECT *
FROM spark_doris;
```

DataFrame

```
val dorisSparkDF = spark.read.format("doris")
  .option("doris.table.identifier", "$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME")
  .option("doris.fenodes", "$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT")
  .option("user", "$YOUR_DORIS_USERNAME")
  .option("password", "$YOUR_DORIS_PASSWORD")
  .load()

dorisSparkDF.show(5)
```

RDD

```
import org.apache.doris.spark._

val dorisSparkRDD = sc.dorisRDD(
  tableIdentifier = Some("$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME"),
  cfg = Some(Map(
    "doris.fenodes" -> "$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT",
    "doris.request.auth.user" -> "$YOUR_DORIS_USERNAME",
    "doris.request.auth.password" -> "$YOUR_DORIS_PASSWORD"
  ))
)

dorisSparkRDD.collect()
```

pySpark

```
dorisSparkDF = spark.read.format("doris")
  .option("doris.table.identifier", "$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME")
  .option("doris.fenodes", "$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT")
  .option("user", "$YOUR_DORIS_USERNAME")
  .option("password", "$YOUR_DORIS_PASSWORD")
  .load()
// show 5 lines data
dorisSparkDF.show(5)
```

Write

SQL

```
CREATE
TEMPORARY VIEW spark_doris
USING doris
OPTIONS(
  "table.identifier"="$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME",
  "fenodes"="$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT",
  "user"="$YOUR_DORIS_USERNAME",
  "password"="$YOUR_DORIS_PASSWORD"
);

INSERT INTO spark_doris
VALUES ("VALUE1", "VALUE2", ...);
# or
INSERT INTO spark_doris
SELECT *
FROM YOUR_TABLE
# or
INSERT OVERWRITE
SELECT *
FROM YOUR_TABLE
```

DataFrame(batch/stream)

```
// batch sink
val mockDataDF = List(
  (3, "440403001005", "21.cn"),
  (1, "4404030013005", "22.cn"),
  (33, null, "23.cn")
).toDF("id", "mi_code", "mi_name")
mockDataDF.show(5)

mockDataDF.write.format("doris")
  .option("doris.table.identifier", "$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME")
  .option("doris.fenodes", "$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT")
  .option("user", "$YOUR_DORIS_USERNAME")
  .option("password", "$YOUR_DORIS_PASSWORD")
  //other options
  //specify the fields to write
  .option("doris.write.fields", "$YOUR_FIELDS_TO_WRITE")
  // Support setting Overwrite mode to overwrite data
  // .option("save_mode", SaveMode.Overwrite)
  .save()

// stream sink(StructuredStreaming)

// Result DataFrame with structured data, the configuration method is the same as the batch mode.
val sourceDf = spark.readStream.
       .format("your_own_stream_source")
       .load()

val resultDf = sourceDf.<transformations>

resultDf.writeStream
      .format("doris")
      .option("checkpointLocation", "$YOUR_CHECKPOINT_LOCATION")
      .option("doris.table.identifier", "$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME")
      .option("doris.fenodes", "$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT")
      .option("user", "$YOUR_DORIS_USERNAME")
      .option("password", "$YOUR_DORIS_PASSWORD")
      .start()
      .awaitTermination()

// There is a column value in the Result DataFrame that can be written directly, such as the value in the kafka message that
conforms to the import format

val kafkaSource = spark.readStream
  .format("kafka")
  .option("kafka.bootstrap.servers", "$YOUR_KAFKA_SERVERS")
  .option("startingOffsets", "latest")
  .option("subscribe", "$YOUR_KAFKA_TOPICS")
  .load()
kafkaSource.selectExpr("CAST(key AS STRING)", "CAST(value as STRING)")
  .writeStream
  .format("doris")
  .option("checkpointLocation", "$YOUR_CHECKPOINT_LOCATION")
  .option("doris.table.identifier", "$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME")
  .option("doris.fenodes", "$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT")
  .option("user", "$YOUR_DORIS_USERNAME")
  .option("password", "$YOUR_DORIS_PASSWORD")
  // Set this option to true, and the value column in the Kafka message will be written directly without processing.
  .option("doris.sink.streaming.passthrough", "true")
  .option("doris.sink.properties.format", "json")
  //other options
```

```
    .start()
    .awaitTermination()
```

Configuration

General

Key	DefaultValue	Comment
doris.fenodes	–	Doris FE http address, support multiple addresses, separated by commas
doris.table.identifier	–	Doris table identifier, eg, db1.tbl1
doris.request.retries	3	Number of retries to send requests to Doris
doris.request.connect.timeout.ms	30000	Connection timeout for sending requests to Doris
doris.request.read.timeout.ms	30000	Read timeout for sending request to Doris
doris.request.query.timeout.s	3600	Query the timeout time of doris, the default is 1 hour, -1 means no timeout limit

Key	DefaultValue	Comment
doris.request.tablet.size	Integer.MAX_VALUE	The number of Doris Tablets corresponding to an RDD Partition. The smaller this value is set, the more partitions will be generated. This will increase the parallelism on the Spark side, but at the same time will cause greater pressure on Doris.
doris.read.field	–	List of column names in the Doris table, separated by commas
doris.batch.size	4064	The maximum number of rows to read data from BE at one time. Increasing this value can reduce the number of connections between Spark and Doris. Thereby reducing the extra time overhead caused by network delay.
doris.exec.mem.limit	2147483648	Memory limit for a single query. The default is 2GB, in bytes.
doris.deserialize.arrow.async	false	Whether to support asynchronous conversion of Arrow format to RowBatch required for spark-doris-connector iteration

Key	Default Value	Comment
doris.deserialize.queue.size	64	Asynchronous conversion of the internal processing queue in Arrow format takes effect when doris.deserialize.arrow.async is true
doris.write.fields	–	Specifies the fields (or the order of the fields) to write to the Doris table, fileds separated by commas. By default, all fields are written in the order of Doris table fields.
doris.sink.batch.size	100000	Maximum number of lines in a single write BE
doris.sink.max-retries	0	Number of retries after writing BE, Since version 1.3.0, the default value is 0, which means no retries are performed by default. When this parameter is set greater than 0, batch-level failure retries will be performed, and data of the configured size of `doris.sink.batch.size` will be cached in the Spark Executor memory. The memory allocation may need to be appropriately increased.
doris.sink.properties.format	–	Data format of the stream load. Supported formats: csv, json, arrow More Multi-parameter details
doris.sink.properties.*	–	Import parameters for Stream Load. For example: Specify column separator: `'doris.sink.properties.column_separator' = ','.` More parameter details
doris.sink.task.partition.size	–	The number of partitions corresponding to the Writing task. After filtering and other operations, the number of partitions written in Spark RDD may be large, but the number of records corresponding to each Partition is relatively small, resulting in increased writing frequency and waste of computing resources. The smaller this value is set, the less Doris write frequency and less Doris merge pressure. It is generally used with doris.sink.task.use.repartition.
doris.sink.task.use.repartition	false	Whether to use repartition mode to control the number of partitions written by Doris. The default value is false, and coalesce is used (note: if there is no Spark action before the write, the whole computation will be less parallel). If it is set to true, then repartition is used (note: you can set the final number of partitions at the cost of shuffle).

Key	Default Value	Comment
doris.sink.batch.interval.ms	50	The interval time of each batch sink, unit ms.
doris.sink.enable-2pc	false	Whether to enable two-stage commit. When enabled, transactions will be committed at the end of the job, and all pre-commit transactions will be rolled back when some tasks fail.
doris.sink.auto-redirect	true	Whether to redirect StreamLoad requests. After being turned on, StreamLoad will write through FE and no longer obtain BE information explicitly.

SQL & Dataframe Configuration

Key	Default Value	Comment
user	–	Doris username
password	–	Doris password
doris.filter.query.in.max.count	100	In the predicate pushdown, the maximum number of elements in the in expression value list. If this number is exceeded, the in-expression conditional filtering is processed on the Spark side.
doris.ignore-type	–	In a temporary view, specify the field types to ignore when reading the schema. eg: when ‘doris.ignore-type’=‘bitmap,hll’

Structured Streaming Configuration

Key	Default Value	Comment
doris.sink.streaming.passthrough	false	Write the value of the first column directly without processing.

RDD Configuration

Key	Default Value	Comment
doris.request.auth.user	–	Doris username
doris.request.auth.password	–	Doris password
doris.filter.query	–	Filter expression of the query, which is transparently transmitted to Doris. Doris uses this expression to complete source-side data filtering.

Doris & Spark Column Type Mapping

Doris Type	Spark Type
NULL_TYPE	DataTypes.NullType
BOOLEAN	DataTypes.BooleanType
TINYINT	DataTypes.ByteType
SMALLINT	DataTypes.ShortType
INT	DataTypes.IntegerType
BIGINT	DataTypes.LongType
FLOAT	DataTypes.FloatType
DOUBLE	DataTypes.DoubleType
DATE	DataTypes.DateType
DATETIME	DataTypes.StringType[1]
DECIMAL	DecimalType
CHAR	DataTypes.StringType
LARGEINT	DecimalType
VARCHAR	DataTypes.StringType
TIME	DataTypes.DoubleType
HLL	Unsupported datatype
Bitmap	Unsupported datatype

- Note: In Connector, `DATETIME` is mapped to `String`. Due to the processing logic of the Doris underlying storage engine, when the time type is used directly, the time range covered cannot meet the demand. So use `String` type to directly return the corresponding time readable text.

FAQ

1. **How to write Bitmap type**

In Spark SQL, when writing data through insert into, if the target table of doris contains data of type `BITMAP` or `HLL`, you need to set the parameter `doris.ignore-type` to the corresponding type and map the columns through `doris.write.fields`. The usage is as follows: > BITMAP > `sql > CREATE TEMPORARY VIEW spark_doris > USING doris > OPTIONS(> "table.identifier"="$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME", > "fenodes"="$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT", > "user"="$YOUR_DORIS_USERNAME", > "password"="$YOUR_DORIS_PASSWORD" > "doris.ignore-type"="bitmap", > "doris.write.fields"="col1,col2,col3,bitmap_col2=to_bitmap(col2),bitmap_col3=bitmap_hash(col3)" >); >` > HLL > `sql > CREATE TEMPORARY VIEW spark_doris > USING doris > OPTIONS(> "table.identifier"="$YOUR_DORIS_DATABASE_NAME.$YOUR_DORIS_TABLE_NAME", > "fenodes"="$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT", > "user"="$YOUR_DORIS_USERNAME", > "password"="$YOUR_DORIS_PASSWORD" > "doris.ignore-type"="hll", > "doris.write.fields"="col1,hll_col1=hll_hash(col1)" >); >`

2. **How to use overwrite to write?**

Starting from version 1.3.0, overwrite mode writing is supported (only supports data overwriting at the full table level). The specific usage is as follows: > DataFrame > `java > resultDf.format("doris") > .option("doris.fenodes","$YOUR_DORIS_FE_HOSTNAME:$YOUR_DORIS_FE_RESFUL_PORT") > // your own options > .option("save_mode", SaveMode.Overwrite) > .save() > >` > SQL > `sql > INSERT OVERWRITE your_target_table > SELECT * FROM your_source_table >`

Flink Doris Connector

Flink Doris Connector

- Flink Doris Connector can support data stored in Doris through Flink operations (read, insert, modify, delete). This document introduces how to operate Doris through Datastream and SQL through Flink.

> **Note:**
>
> 1. Modification and deletion are only supported on the Unique Key model
> 2. The current deletion is to support Flink CDC to access data to achieve automatic deletion. If it is to delete other data access methods, you need to implement it yourself. For the data deletion usage of Flink CDC, please refer to the last section of this document

Version Compatibility

Connector Version	Flink Version	Doris Version	Java Version	Scala Version
1.0.3	1.11,1.12,1.13,1.14	0.15+	8	2.11,2.12
1.1.1	1.14	1.0+	8	2.11,2.12
1.2.1	1.15	1.0+	8	-
1.3.0	1.16	1.0+	8	-
1.4.0	1.15,1.16,1.17	1.0+	8	-
1.5.2	1.15,1.16,1.17,1.18	1.0+	8	-
1.6.2	1.15,1.16,1.17,1.18,1.19	1.0+	8	-
24.0.1	1.15,1.16,1.17,1.18,1.19,1.20	1.0+	8	-

USE

Maven

Add flink-doris-connector

```
<!-- flink-doris-connector -->
<dependency>
   <groupId>org.apache.doris</groupId>
   <artifactId>flink-doris-connector-1.16</artifactId>
   <version>24.0.1</version>
</dependency>
```

Remark

1. Please replace the corresponding Connector and Flink dependent versions according to different Flink versions.
2. You can also download the relevant version jar package from here.

compile

When compiling, you can run `sh build.sh` directly. For details, please refer to here.

After the compilation is successful, the target jar package will be generated in the `dist` directory, such as: `flink-doris-connector-1.5.0-SNAPSHOT.jar`. Copy this file to `classpath` of `Flink` to use `Flink-Doris-Connector`. For example, `Flink` running in `Local` mode, put this file in the `lib/` folder. `Flink` running in `Yarn` cluster mode, put this file into the pre-deployment package.

Instructions

Read

SQL

```
-- doris source
CREATE TABLE flink_doris_source (
     name STRING,
     age INT,
     price DECIMAL(5,2),
     sale DOUBLE
     )
     WITH (
       'connector' = 'doris',
       'fenodes' = 'FE_IP:HTTP_PORT',
       'table.identifier' = 'database.table',
       'username' = 'root',
       'password' = 'password'
);
```

> **Note**
>
> Flink Connector 24.0.0 and later versions support using Arrow Flight SQL to read data

```
CREATE TABLE doris_source (
name STRING,
age int
)
WITH (
  'connector' = 'doris',
  'fenodes' = 'FE_IP:HTTP_PORT',
  'table.identifier' = 'database.table',
  'source.use-flight-sql' = 'true',
  'source.flight-sql-port' = '{fe.conf:arrow_flight_sql_port}',
  'username' = 'root',
```

```
  'password' = ''
)
```

DataStream

```
DorisOptions.Builder builder = DorisOptions.builder()
         .setFenodes("FE_IP:HTTP_PORT")
         .setTableIdentifier("db.table")
         .setUsername("root")
         .setPassword("password");

DorisSource<List<?>> dorisSource = DorisSource.<List<?>>builder()
         .setDorisOptions(builder.build())
         .setDorisReadOptions(DorisReadOptions.builder().build())
         .setDeserializer(new SimpleListDeserializationSchema())
         .build();

env.fromSource(dorisSource, WatermarkStrategy.noWatermarks(), "doris source").print();
```

Write

SQL

```
--enable checkpoint
SET 'execution.checkpointing.interval' = '10s';

-- doris sink
CREATE TABLE flink_doris_sink (
     name STRING,
     age INT,
     price DECIMAL(5,2),
     sale DOUBLE
     )
     WITH (
       'connector' = 'doris',
       'fenodes' = 'FE_IP:HTTP_PORT',
       'table.identifier' = 'db.table',
       'username' = 'root',
       'password' = 'password',
       'sink.label-prefix' = 'doris_label'
);

-- submit insert job
INSERT INTO flink_doris_sink select name,age,price,sale from flink_doris_source
```

DataStream

DorisSink writes data to Doris through StreamLoad, and DataStream supports different serialization methods when writing

String data stream (SimpleStringSerializer)

```
// enable checkpoint
env.enableCheckpointing(10000);
// using batch mode for bounded data
env.setRuntimeMode(RuntimeExecutionMode.BATCH);

DorisSink.Builder<String> builder = DorisSink.builder();
```

```
DorisOptions.Builder dorisBuilder = DorisOptions.builder();
dorisBuilder.setFenodes("FE_IP:HTTP_PORT")
         .setTableIdentifier("db.table")
         .setUsername("root")
         .setPassword("password");

Properties properties = new Properties();
// When the upstream is writing json, the configuration needs to be enabled.
//properties.setProperty("format", "json");
//properties.setProperty("read_json_by_line", "true");
DorisExecutionOptions.Builder executionBuilder = DorisExecutionOptions.builder();
executionBuilder.setLabelPrefix("label-doris") //streamload label prefix
                 .setDeletable(false)
                 .setStreamLoadProp(properties); ;

builder.setDorisReadOptions(DorisReadOptions.builder().build())
         .setDorisExecutionOptions(executionBuilder.build())
         .setSerializer(new SimpleStringSerializer()) //serialize according to string
         .setDorisOptions(dorisBuilder.build());

//mock string source
List<Tuple2<String, Integer>> data = new ArrayList<>();
data.add(new Tuple2<>("doris",1));
DataStreamSource<Tuple2<String, Integer>> source = env. fromCollection(data);

source.map((MapFunction<Tuple2<String, Integer>, String>) t -> t.f0 + "\t" + t.f1)
       .sinkTo(builder.build());

//mock json string source
//env.fromElements("{\"name\":\"zhangsan\",\"age\":1}").sinkTo(builder.build());
```

RowData data stream (RowDataSerializer)

```
// enable checkpoint
env.enableCheckpointing(10000);
// using batch mode for bounded data
env.setRuntimeMode(RuntimeExecutionMode.BATCH);

//doris sink option
DorisSink.Builder<RowData> builder = DorisSink.builder();
DorisOptions.Builder dorisBuilder = DorisOptions.builder();
dorisBuilder.setFenodes("FE_IP:HTTP_PORT")
         .setTableIdentifier("db.table")
         .setUsername("root")
         .setPassword("password");

// json format to streamload
Properties properties = new Properties();
properties.setProperty("format", "json");
properties.setProperty("read_json_by_line", "true");
DorisExecutionOptions.Builder executionBuilder = DorisExecutionOptions.builder();
executionBuilder.setLabelPrefix("label-doris") //streamload label prefix
                 .setDeletable(false)
                 .setStreamLoadProp(properties); //streamload params

//flink rowdata's schema
String[] fields = {"city", "longitude", "latitude", "destroy_date"};
DataType[] types = {DataTypes.VARCHAR(256), DataTypes.DOUBLE(), DataTypes.DOUBLE(), DataTypes.DATE()};

builder.setDorisReadOptions(DorisReadOptions.builder().build())
         .setDorisExecutionOptions(executionBuilder.build())
         .setSerializer(RowDataSerializer.builder() //serialize according to rowdata
                             .setFieldNames(fields)
```

```
                            .setType("json") //json format
                            .setFieldType(types).build())
        .setDorisOptions(dorisBuilder.build());

//mock rowdata source
DataStream<RowData> source = env. fromElements("")
     .map(new MapFunction<String, RowData>() {
         @Override
         public RowData map(String value) throws Exception {
             GenericRowData genericRowData = new GenericRowData(4);
             genericRowData.setField(0, StringData.fromString("beijing"));
             genericRowData.setField(1, 116.405419);
             genericRowData.setField(2, 39.916927);
             genericRowData.setField(3, LocalDate.now().toEpochDay());
             return genericRowData;
         }
     });

source. sinkTo(builder. build());
```

CDC data stream (JsonDebeziumSchemaSerializer)

```
// enable checkpoint
env.enableCheckpointing(10000);

Properties props = new Properties();
props. setProperty("format", "json");
props.setProperty("read_json_by_line", "true");
DorisOptions dorisOptions = DorisOptions. builder()
         .setFenodes("127.0.0.1:8030")
         .setTableIdentifier("test.t1")
         .setUsername("root")
         .setPassword("").build();

DorisExecutionOptions.Builder executionBuilder = DorisExecutionOptions.builder();
executionBuilder.setLabelPrefix("label-prefix")
         .setStreamLoadProp(props).setDeletable(true);

DorisSink.Builder<String> builder = DorisSink.builder();
builder.setDorisReadOptions(DorisReadOptions.builder().build())
         .setDorisExecutionOptions(executionBuilder.build())
         .setDorisOptions(dorisOptions)
         .setSerializer(JsonDebeziumSchemaSerializer.builder().setDorisOptions(dorisOptions).build());

env.fromSource(mySqlSource, WatermarkStrategy.noWatermarks(), "MySQL Source")
         .sinkTo(builder.build());
```

Reference: CDCSchemaChangeExample

Lookup Join

```
CREATE TABLE fact_table (
  `id` BIGINT,
  `name` STRING,
  `city` STRING,
  `process_time` as proctime()
) WITH (
  'connector' = 'kafka',
  ...
);
```

```
create table dim_city(
  `city` STRING,
  `level` INT ,
  `province` STRING,
  `country` STRING
) WITH (
  'connector' = 'doris',
  'fenodes' = '127.0.0.1:8030',
  'jdbc-url' = 'jdbc:mysql://127.0.0.1:9030',
  'table.identifier' = 'dim.dim_city',
  'username' = 'root',
  'password' = ''
);

SELECT a.id, a.name, a.city, c.province, c.country,c.level
FROM fact_table a
LEFT JOIN dim_city FOR SYSTEM_TIME AS OF a.process_time AS c
ON a.city = c.city
```

Configuration

General configuration items

Key	Default Value	Required	Comment
fenodes	–	Y	Doris FE http address, multiple addresses are supported, separated by commas
benodes	–	N	Doris BE http address, multiple addresses are supported, separated by commas. refer to #187
jdbc-url	–	N	jdbc connection information, such as: jdbc:mysql://127.0.0.1:9030
table.identifier	–	Y	Doris table name, such as: db.tbl
username	–	Y	username to access Doris
password	–	Y	Password to access Doris
auto-redirect	true	N	Whether to redirect StreamLoad requests. After being turned on, StreamLoad will be written through FE, and BE information will no longer be displayed.
doris.request.retries	3	N	Number of retries to send requests to Doris
doris.request.connect.timeout.ms	30000	N	Connection timeout for sending requests to Doris
doris.request.read.timeout.ms	30000	N	Read timeout for sending requests to Doris

Source configuration item

Key	Default Value	Required	Comment
doris.request.query.timeout.s	3600	N	The timeout time for querying Doris, the default value is 1 hour, -1 means no timeout limit

Key	Default Value	Required	Comment
doris.request.tablet.size	Integer. MAX_VALUE	N	The number of Doris Tablets corresponding to a Partition. The smaller this value is set, the more Partitions will be generated. This improves the parallelism on the Flink side, but at the same time puts more pressure on Doris.
doris.batch.size	1024	N	The maximum number of rows to read data from BE at a time. Increasing this value reduces the number of connections established between Flink and Doris. Thereby reducing the additional time overhead caused by network delay.
doris.exec.mem.limit	2147483648	N	Memory limit for a single query. The default is 2GB, in bytes
doris.deserialize.arrow.async	FALSE	N	Whether to support asynchronous conversion of Arrow format to RowBatch needed for flink-doris-connector iterations
doris.deserialize.queue.size	64	N	Asynchronous conversion of internal processing queue in Arrow format, effective when doris.deserialize.arrow.async is true
source.use-flight-sql	FALSE	N	Whether to use Arrow Flight SQL to read
source.flight-sql-port	-	N	When using ArrowFlightSQL to read, FE's arrow_flight_sql_port

Datastream-specific configuration items

Key	Default Value	Required	Comment
doris.read.field	–	N	Read the list of column names of the Doris table, separated by commas
doris.filter.query	–	N	The expression to filter the read data, this expression is transparently passed to Doris. Doris uses this expression to complete source-side data filtering. For example age=18.

Sink configuration items

Key	Default Value	Required	Comment
sink.label-prefix	–	Y	The label prefix used by Stream load import. In the 2pc scenario, global uniqueness is required to ensure Flink's EOS semantics.

Key	Default Value	Required	Comment
sink.properties.*	–	N	Import parameters for Stream Load. For example: 'sink.properties.column_separator' = ',' defines column delimiters, 'sink.properties.escape_delimiters' = 'true' special characters as delimiters, '' will be converted to binary 0x01 JSON format import 'sink.properties.format' = 'json' 'sink.properties. read_json_by_line' = 'true' Detailed parameters refer to here. Group Commit mode 'sink.properties.group_commit' = 'sync_mode' Starting from version 1.6.2, we have introduced support for load data with group commit functionality. For a comprehensive understanding of the available parameters, please refer to the Group Commit Manual group commit.
sink.enable-delete	TRUE	N	Whether to enable delete. This option requires the Doris table to enable the batch delete function (Doris 0.15+ version is enabled by default), and only supports the Unique model.
sink.enable-2pc	TRUE	N	Whether to enable two-phase commit (2pc), the default is true, to ensure Exactly-Once semantics. For two-phase commit, please refer to here.
sink.buffer-size	1MB	N	The size of the write data cache buffer, in bytes. It is not recommended to modify, the default configuration is enough
sink.buffer-count	3	N	The number of write data buffers. It is not recommended to modify, the default configuration is enough
sink.max-retries	3	N	Maximum number of retries after Commit failure, default 3
sink.use-cache	false	N	In case of an exception, whether to use the memory cache for recovery. When enabled, the data during the Checkpoint period will be retained in the cache.
sink.enable.batch-mode	false	N	Whether to use the batch mode to write to Doris. After it is enabled, the writing timing does not depend on Checkpoint. The writing is controlled through the sink.buffer-flush.max-rows/sink.buffer-flush.max-bytes/sink.buffer-flush.interval parameter. Enter the opportunity. After being turned on at the same time, Exactly-once semantics will not be guaranteed. Uniq model can be used to achieve idempotence.
sink.flush.queue-size	2	N	In batch mode, the cached column size.
sink.buffer-flush.max-rows	500000	N	In batch mode, the maximum number of data rows written in a single batch.
sink.buffer-flush.max-bytes	100MB	N	In batch mode, the maximum number of bytes written in a single batch.
sink.buffer-flush.interval	10s	N	In batch mode, the interval for asynchronously refreshing the cache
sink.ignore.update-before	true	N	Whether to ignore the update-before event, ignored by default.

Lookup Join configuration item

Key	Default Value	Required	Comment
lookup.cache.max-rows	-1	N	The maximum number of rows in the lookup cache, the default value is -1, and the cache is not enabled
lookup.cache.ttl	10s	N	The maximum time of lookup cache, the default is 10s
lookup.max-retries	1	N	The number of retries after a lookup query fails
lookup.jdbc.async	false	N	Whether to enable asynchronous lookup, the default is false
lookup.jdbc.read.batch.size	128	N	Under asynchronous lookup, the maximum batch size for each query
lookup.jdbc.read.batch.queue-size	256	N	The size of the intermediate buffer queue during asynchronous lookup
lookup.jdbc.read.thread-size	3	N	The number of jdbc threads for lookup in each task

Doris & Flink Column Type Mapping

Doris Type	Flink Type
NULL_TYPE	NULL
BOOLEAN	BOOLEAN
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE
DATE	DATE
DATETIME	TIMESTAMP
DECIMAL	DECIMAL
CHAR	STRING
LARGEINT	STRING
VARCHAR	STRING
STRING	STRING
DECIMALV2	DECIMAL
ARRAY	ARRAY
MAP	MAP
JSON	STRING
VARIANT	STRING

Doris Type	Flink Type
IPV4	STRING
IPV6	STRING

> Starting from version connector-1.6.1, support is added for reading three data types: Variant, IPV6, and IPV4. Reading IPV6 and Variant requires Doris version 2.1.1 or higher.

Flink write Metrics

Where the metrics value of type Counter is the cumulative value of the imported task from the beginning to the current time, you can observe each metric in each table in the Flink Webui metrics.

Name	Metric Type	Description
totalFlushLoadBytes	Counter	Number of bytes imported.
flushTotalNumberRows	Counter	Number of rows imported for total processing
totalFlushLoadedRows	Counter	Number of rows successfully imported.
totalFlushTimeMs	Counter	Number of Import completion time. Unit milliseconds
totalFlushSucceededNumber	Counter	Number of times that the data-batch been successfully imported.
totalFlushFailedNumber	Counter	Number of times that the data-batch been failed.
totalFlushFilteredRows	Counter	Number of rows that do not qualify for data quality flushed
totalFlushUnselectedRows	Counter	Number of rows filtered by where condition flushed
beginTxnTimeMs	Histogram	The time cost for RPC to Fe to begin a transaction, Unit milliseconds.
putDataTimeMs	Histogram	The time cost for RPC to Fe to get a stream load plan, Unit milliseconds.
readDataTimeMs	Histogram	Read data time, Unit milliseconds.
writeDataTimeMs	Histogram	Write data time, Unit milliseconds.
commitAndPublishTimeMs	Histogram	The time cost for RPC to Fe to commit and publish a transaction, Unit milliseconds.
loadTimeMs	Histogram	Import completion time

An example of using Flink CDC to access Doris

```
SET 'execution.checkpointing.interval' = '10s';
CREATE TABLE cdc_mysql_source (
  id int
  ,name VARCHAR
  ,PRIMARY KEY (id) NOT ENFORCED
) WITH (
 'connector' = 'mysql-cdc',
 'hostname' = '127.0.0.1',
 'port' = '3306',
 'username' = 'root',
 'password' = 'password',
 'database-name' = 'database',
```

```
 'table-name' = 'table'
);

-- Support synchronous insert/update/delete events
CREATE TABLE doris_sink (
id INT,
name STRING
)
WITH (
  'connector' = 'doris',
  'fenodes' = '127.0.0.1:8030',
  'table.identifier' = 'database.table',
  'username' = 'root',
  'password' = '',
  'sink.properties.format' = 'json',
  'sink.properties.read_json_by_line' = 'true',
  'sink.enable-delete' = 'true', -- Synchronize delete events
  'sink.label-prefix' = 'doris_label'
);

insert into doris_sink select id,name from cdc_mysql_source;
```

Example of using FlinkSQL to access and implement partial column updates through CDC

```
-- enable checkpoint
SET 'execution.checkpointing.interval' = '10s';

CREATE TABLE cdc_mysql_source (
   id int
  ,name STRING
  ,bank STRING
  ,age int
  ,PRIMARY KEY (id) NOT ENFORCED
) WITH (
 'connector' = 'mysql-cdc',
 'hostname' = '127.0.0.1',
 'port' = '3306',
 'username' = 'root',
 'password' = 'password',
 'database-name' = 'database',
 'table-name' = 'table'
);

CREATE TABLE doris_sink (
    id INT,
    name STRING,
    bank STRING,
    age int
)
WITH (
  'connector' = 'doris',
  'fenodes' = '127.0.0.1:8030',
  'table.identifier' = 'database.table',
  'username' = 'root',
  'password' = '',
  'sink.properties.format' = 'json',
  'sink.properties.read_json_by_line' = 'true',
  'sink.properties.columns' = 'id,name,bank,age',
  'sink.properties.partial_columns' = 'true' --Enable partial column updates
);
```

```
insert into doris_sink select id,name,bank,age from cdc_mysql_source;
```

Use Flink CDC to access multiple tables or the entire database (Supports MySQL, Oracle, PostgreSQL, SQLServer, MongoDB)

Grammar

```
<FLINK_HOME>bin/flink run \
     -c org.apache.doris.flink.tools.cdc.CdcTools \
     lib/flink-doris-connector-1.16-1.4.0-SNAPSHOT.jar\
     <mysql-sync-database|oracle-sync-database|postgres-sync-database|sqlserver-sync-database|mongodb-sync-database> \
     --database <doris-database-name> \
     [--job-name <flink-job-name>] \
     [--table-prefix <doris-table-prefix>] \
     [--table-suffix <doris-table-suffix>] \
     [--including-tables <mysql-table-name|name-regular-expr>] \
     [--excluding-tables <mysql-table-name|name-regular-expr>] \
     --mysql-conf <mysql-cdc-source-conf> [--mysql-conf <mysql-cdc-source-conf> ...] \
     --oracle-conf <oracle-cdc-source-conf> [--oracle-conf <oracle-cdc-source-conf> ...] \
     --postgres-conf <postgres-cdc-source-conf> [--postgres-conf <postgres-cdc-source-conf> ...] \
     --sqlserver-conf <sqlserver-cdc-source-conf> [--sqlserver-conf <sqlserver-cdc-source-conf> ...] \
     --sink-conf <doris-sink-conf> [--table-conf <doris-sink-conf> ...] \
     [--table-conf <doris-table-conf> [--table-conf <doris-table-conf> ...]]
```

Key	Comment
–job-name	Flink task name, optional
–database	Database name synchronized to Doris
–table-prefix	Doris table prefix name, such as –table-prefix ods_.
–table-suffix	Same as above, the suffix name of the Doris table.
–including-tables	For MySQL tables that need to be synchronized, you can use “｜” to separate multiple tables and support regular expressions. For example –including-tables table1
–excluding-tables	For tables that do not need to be synchronized, the usage is the same as above.
–mysql-conf	MySQL CDCSource configuration, for example –mysql-conf hostname=127.0.0.1, you can find it here View all configurations MySQL-CDC, where hostname/username/password/database-name is required. When the synchronized library table contains a non-primary key table, `scan.incremental.snapshot.chunk.key-column` must be set, and only one field of non-null type can be selected. For example: `scan.incremental.snapshot.chunk.key-column=database.table:column,database.table1:column...`, different database table columns are separated by ,.
–oracle-conf	Oracle CDCSource configuration, for example –oracle-conf hostname=127.0.0.1, you can find here View all configurations Oracle-CDC, where hostname/username/password/database-name/schema-name is required.
–postgres-conf	Postgres CDCSource configuration, e.g. –postgres-conf hostname=127.0.0.1, you can find here View all configurations Postgres-CDC where hostname/username/password/database-name/schema-name/slot.name is required.

Key	Comment
–sqlserver-conf	SQLServer CDCSource configuration, for example –sqlserver-conf hostname=127.0.0.1, you can find it here View all configurations SQLServer-CDC, where hostname/username/password/database-name/schema-name is required.
–db2-conf	DB2 CDCSource configuration, for example –db2-conf hostname=127.0.0.1, you can find it here View all configurations DB2-CDC, where hostname/username/password/database-name/schema-name is required.
–mongodb-conf	MongoDB CDCSource configuration, for example –mongodb-conf hosts=127.0.0.1:27017, you can find all Mongo-CDC configurations here, where hosts/username/password/database are required. The –mongodb-conf schema.sample-percent configuration is for automatically sampling MongoDB data for creating a table in Doris, with a default value of 0.2.
–sink-conf	All configurations of Doris Sink can be found here View the complete configuration items.
–table-conf	The configuration items of the Doris table(The exception is table-buckets, non-properties attributes), that is, the content contained in properties. For example `--table-conf replication_num=1`, and the `--table-conf table-buckets="tbl1:10,tbl2:20,a.*:30,b.*:40,.*:50"` option specifies the number of buckets for different tables based on the order of regular expressions. If there is no match, the table is created with the default setting of BUCKETS AUTO.
–ignore-default-value	Turn off the default value of synchronizing MySQL table structure. It is suitable for synchronizing MySQL data to Doris when the field has a default value but the actual inserted data is null. Reference here
–use-new-schema-change	Whether to use the new schema change to support synchronization of MySQL multi-column changes and default values. since version 1.6.0, the default value has been set to true. Reference here
–schema-change-mode	The mode for parsing schema change supports two parsing modes: `debezium_structure` and `sql_parser`. The default mode is `debezium_structure`. `debezium_structure` parses the data structure used when upstream CDC synchronizes data, and determines DDL change operations by parsing this structure. `sql_parser` determines the DDL change operation by parsing the DDL statement when the upstream CDC synchronizes data, so this parsing mode is more accurate. Usage example: `--schema-change-mode debezium_structure` This feature will be available in versions after 1.6.2.1
–single-sink	Whether to use a single Sink to synchronize all tables. When turned on, newly created tables in the upstream can also be automatically recognized and tables automatically created.
–multi-to-one-origin	When writing multiple upstream tables into the same table, the configuration of the source table, for example: –multi-to-one-origin “a_.*|b_.*”， Reference here
–multi-to-one-target	Used with multi-to-one-origin, the configuration of the target table, such as: –multi-to-one-target “a|b”
–create-table-only	Whether only the table schema should be synchronized

Note

1. When synchronizing, you need to add the corresponding Flink CDC dependencies in the `$FLINK_HOME/lib` directory, such as flink-sql-connector-mysql-cdc-$version.jar, flink-sql-connector-oracle-cdc-{version}.jar , flink-sql-connector-mongodb-cdc-${version}.jar

2. The Flink CDC version that Connector 24.0.0 depends on must be above 3.1 . If Flink CDC is to be used for synchronizing data from MySQL or Oracle, relevant JDBC drivers also need to be added under `$FLINK_HOME/lib`.

MySQL synchronization example

```
<FLINK_HOME>bin/flink run \
     -Dexecution.checkpointing.interval=10s\
     -Dparallelism.default=1\
     -c org.apache.doris.flink.tools.cdc.CdcTools\
     lib/flink-doris-connector-1.18-24.0.1.jar \
     mysql-sync-database\
     --database test_db \
     --mysql-conf hostname=127.0.0.1 \
     --mysql-conf port=3306 \
     --mysql-conf username=root \
     --mysql-conf password=123456 \
     --mysql-conf database-name=mysql_db \
     --including-tables "tbl1|test.*" \
     --sink-conf fenodes=127.0.0.1:8030 \
     --sink-conf username=root \
     --sink-conf password=123456 \
     --sink-conf jdbc-url=jdbc:mysql://127.0.0.1:9030 \
     --sink-conf sink.label-prefix=label \
     --table-conf replication_num=1
```

Oracle synchronization example

```
<FLINK_HOME>bin/flink run \
      -Dexecution.checkpointing.interval=10s \
      -Dparallelism.default=1 \
      -c org.apache.doris.flink.tools.cdc.CdcTools \
      ./lib/flink-doris-connector-1.18-24.0.1.jar \
      oracle-sync-database \
      --database test_db \
      --oracle-conf hostname=127.0.0.1 \
      --oracle-conf port=1521 \
      --oracle-conf username=admin \
      --oracle-conf password="password" \
      --oracle-conf database-name=XE \
      --oracle-conf schema-name=ADMIN \
      --including-tables "tbl1|tbl2" \
      --sink-conf fenodes=127.0.0.1:8030 \
      --sink-conf username=root \
      --sink-conf password=\
      --sink-conf jdbc-url=jdbc:mysql://127.0.0.1:9030 \
      --sink-conf sink.label-prefix=label \
      --table-conf replication_num=1
```

PostgreSQL synchronization example

```
<FLINK_HOME>/bin/flink run \
     -Dexecution.checkpointing.interval=10s \
     -Dparallelism.default=1\
     -c org.apache.doris.flink.tools.cdc.CdcTools \
     ./lib/flink-doris-connector-1.18-24.0.1.jar \
```

```
     postgres-sync-database \
     --database db1\
     --postgres-conf hostname=127.0.0.1 \
     --postgres-conf port=5432 \
     --postgres-conf username=postgres \
     --postgres-conf password="123456" \
     --postgres-conf database-name=postgres \
     --postgres-conf schema-name=public \
     --postgres-conf slot.name=test \
     --postgres-conf decoding.plugin.name=pgoutput \
     --including-tables "tbl1|tbl2" \
     --sink-conf fenodes=127.0.0.1:8030 \
     --sink-conf username=root \
     --sink-conf password=\
     --sink-conf jdbc-url=jdbc:mysql://127.0.0.1:9030 \
     --sink-conf sink.label-prefix=label \
     --table-conf replication_num=1
```

SQLServer synchronization example

```
<FLINK_HOME>/bin/flink run \
     -Dexecution.checkpointing.interval=10s \
     -Dparallelism.default=1 \
     -c org.apache.doris.flink.tools.cdc.CdcTools \
     ./lib/flink-doris-connector-1.18-24.0.1.jar \
     sqlserver-sync-database \
     --database db1\
     --sqlserver-conf hostname=127.0.0.1 \
     --sqlserver-conf port=1433 \
     --sqlserver-conf username=sa \
     --sqlserver-conf password="123456" \
     --sqlserver-conf database-name=CDC_DB \
     --sqlserver-conf schema-name=dbo \
     --including-tables "tbl1|tbl2" \
     --sink-conf fenodes=127.0.0.1:8030 \
     --sink-conf username=root \
     --sink-conf password=\
     --sink-conf jdbc-url=jdbc:mysql://127.0.0.1:9030 \
     --sink-conf sink.label-prefix=label \
     --table-conf replication_num=1
```

DB2 synchronization example

```
<FLINK_HOME>bin/flink run \
    -Dexecution.checkpointing.interval=10s \
    -Dparallelism.default=1 \
    -c org.apache.doris.flink.tools.cdc.CdcTools \
    lib/flink-doris-connector-1.16-24.0.1.jar \
    db2-sync-database \
    --database db2_test \
    --db2-conf hostname=127.0.0.1 \
    --db2-conf port=50000 \
    --db2-conf username=db2inst1 \
    --db2-conf password=doris123456 \
    --db2-conf database-name=testdb \
    --db2-conf schema-name=DB2INST1 \
    --including-tables "FULL_TYPES|CUSTOMERS" \
    --single-sink true \
    --use-new-schema-change true \
    --sink-conf fenodes=127.0.0.1:8030 \
```

```
    --sink-conf username=root \
    --sink-conf password=123456 \
    --sink-conf jdbc-url=jdbc:mysql://127.0.0.1:9030 \
    --sink-conf sink.label-prefix=label \
    --table-conf replication_num=1
```

MongoDB synchronization example

```
<FLINK_HOME>/bin/flink run \
    -Dexecution.checkpointing.interval=10s \
    -Dparallelism.default=1 \
    -c org.apache.doris.flink.tools.cdc.CdcTools \
    ./lib/flink-doris-connector-1.18-24.0.1.jar \
    mongodb-sync-database \
    --database doris_db \
    --schema-change-mode debezium_structure \
    --mongodb-conf hosts=127.0.0.1:27017 \
    --mongodb-conf username=flinkuser \
    --mongodb-conf password=flinkpwd \
    --mongodb-conf database=test \
    --mongodb-conf scan.startup.mode=initial \
    --mongodb-conf schema.sample-percent=0.2 \
    --including-tables "tbl1|tbl2" \
    --sink-conf fenodes=127.0.0.1:8030 \
    --sink-conf username=root \
    --sink-conf password= \
    --sink-conf jdbc-url=jdbc:mysql://127.0.0.1:9030 \
    --sink-conf sink.label-prefix=label \
    --sink-conf sink.enable-2pc=false \
    --table-conf replication_num=1
```

Use Flink CDC to update Key column

Generally, in a business database, the number is used as the primary key of the table, such as the Student table, the number (id) is used as the primary key, but with the development of the business, the number corresponding to the data may change. In this scenario, using Flink CDC + Doris Connector to synchronize data can automatically update the data in the Doris primary key column. #### Principle The underlying collection tool of Flink CDC is Debezium. Debezium internally uses the op field to identify the corresponding operation: the values of the op field are c, u, d, and r, corresponding to create, update, delete, and read. For the update of the primary key column, Flink CDC will send DELETE and INSERT events downstream, and after the data is synchronized to Doris, it will automatically update the data of the primary key column.

Example

The Flink program can refer to the CDC synchronization example above. After the task is successfully submitted, execute the Update primary key column statement (`update student set id = '1002' where id = '1001'`) on the MySQL side to modify the data in Doris .

Use Flink to delete data based on specified columns

Generally, messages in Kafka use specific fields to mark the operation type, such as {“op_type”:“delete”,data:{…}}. For this type of data, it is hoped that the data with op_type=delete will be deleted.

By default, DorisSink will distinguish the type of event based on RowKind. Usually, in the case of cdc, the event type can be obtained directly, and the hidden column `__DORIS_DELETE_SIGN__` is assigned to achieve the purpose of deletion, while Kafka needs to be based on business logic. Judgment, display the value passed in to the hidden column.

Example

```
-- Such as upstream data: {"op_type":"delete",data:{"id":1,"name":"zhangsan"}}
CREATE TABLE KAFKA_SOURCE(
  data STRING,
  op_type STRING
) WITH (
  'connector' = 'kafka',
  ...
);

CREATE TABLE DORIS_SINK(
  id INT,
  name STRING,
  __DORIS_DELETE_SIGN__ INT
) WITH (
  'connector' = 'doris',
  'fenodes' = '127.0.0.1:8030',
  'table.identifier' = 'db.table',
  'username' = 'root',
  'password' = '',
  'sink.enable-delete' = 'false',        -- false means not to get the event type from RowKind
  'sink.properties.columns' = 'id, name, __DORIS_DELETE_SIGN__'  -- Display the import column of the specified streamload
);

INSERT INTO DORIS_SINK
SELECT json_value(data,'$.id') as id,
json_value(data,'$.name') as name,
if(op_type='delete',1,0) as __DORIS_DELETE_SIGN__
from KAFKA_SOURCE;
```

Java example

`samples/doris-demo/` An example of the Java version is provided below for reference, see here

Best Practices

Application scenarios

The most suitable scenario for using Flink Doris Connector is to synchronize source data to Doris (MySQL, Oracle, PostgreSQL) in real time/batch, etc., and use Flink to perform joint analysis on data in Doris and other data sources. You can also use Flink Doris Connector

Other

1. The Flink Doris Connector mainly relies on Checkpoint for streaming writing, so the interval between Checkpoints is the visible delay time of the data.
2. To ensure the Exactly Once semantics of Flink, the Flink Doris Connector enables two-phase commit by default, and Doris enables two-phase commit by default after version 1.1. 1.0 can be enabled by

modifying the BE parameters, please refer to two_phase_commit.

FAQ

1. **After Doris Source finishes reading data, why does the stream end?**

Currently Doris Source is a bounded stream and does not support CDC reading.

2. **Can Flink read Doris and perform conditional pushdown?**

By configuring the doris.filter.query parameter, refer to the configuration section for details.

3. **How to write Bitmap type?**

```
CREATE TABLE bitmap_sink (
dt int,
page string,
user_id int
)
WITH (
   'connector' = 'doris',
   'fenodes' = '127.0.0.1:8030',
   'table.identifier' = 'test.bitmap_test',
   'username' = 'root',
   'password' = '',
   'sink.label-prefix' = 'doris_label',
   'sink.properties.columns' = 'dt,page,user_id,user_id=to_bitmap(user_id)'
)
```

4. **errCode = 2, detailMessage = Label [label_0_1] has already been used, relate to txn [19650]**

In the Exactly-Once scenario, the Flink Job must be restarted from the latest Checkpoint/Savepoint, otherwise the above error will be reported. When Exactly-Once is not required, it can also be solved by turning off 2PC commits (sink.enable-2pc=false) or changing to a different sink.label-prefix.

5. **errCode = 2, detailMessage = transaction [19650] not found**

This occurs during the Commit stage. The transaction ID recorded in the checkpoint has expired on the FE side. When committing again at this time, the above error will occur. At this point, it's impossible to start from the checkpoint. Subsequently, you can extend the expiration time by modifying the `streaming_label_keep_max_second` configuration in `fe.conf`. The default expiration time is 12 hours. After doris version 2.0, it will also be limited by the `label_num_threshold` configuration in `fe.conf` (default 2000), which can be increased or changed to -1 (-1 means only limited by time).

6. **errCode = 2, detailMessage = current running txns on db 10006 is 100, larger than limit 100**

This is because the concurrent import of the same library exceeds 100, which can be solved by adjusting the parameter `max_running_txn_num_per_db` of fe.conf. For details, please refer to max_running_txn_num_per_db

At the same time, if a task frequently modifies the label and restarts, it may also cause this error. In the 2pc scenario (Duplicate/Aggregate model), the label of each task needs to be unique, and when restarting from the checkpoint, the Flink task will actively abort the txn that has been successfully precommitted before and has not been committed. Frequently modifying the label and restarting will cause a large number of txn that have

successfully precommitted to fail to be aborted, occupying the transaction. Under the Unique model, 2pc can also be turned off, which can realize idempotent writing.

7. **How to ensure the order of a batch of data when Flink writes to the Uniq model?**

You can add sequence column configuration to ensure that, for details, please refer to sequence

8. **The Flink task does not report an error, but the data cannot be synchronized?**

Before Connector1.1.0, it was written in batches, and the writing was driven by data. It was necessary to determine whether there was data written upstream. After 1.1.0, it depends on Checkpoint, and Checkpoint must be enabled to write.

9. **tablet writer write failed, tablet_id=190958, txn_id=3505530, err=-235**

It usually occurs before Connector1.1.0, because the writing frequency is too fast, resulting in too many versions. The frequency of Streamload can be reduced by setting the sink.batch.size and sink.batch.interval parameters. After Connector 1.1.0, the default write timing is controlled by Checkpoint, and the write frequency can be reduced by increasing the Checkpoint interval.

10. **Flink imports dirty data, how to skip it?**

When Flink imports data, if there is dirty data, such as field format, length, etc., it will cause StreamLoad to report an error, and Flink will continue to retry at this time. If you need to skip, you can disable the strict mode of StreamLoad (strict_mode=false, max_filter_ratio=1) or filter the data before the Sink operator.

11. **How should the source table and Doris table correspond?** When using Flink Connector to import data, pay attention to two aspects. The first is that the columns and types of the source table correspond to the columns and types in flink sql; the second is that the columns and types in flink sql must match those of the Doris table For the correspondence between columns and types, please refer to the above “Doris & Flink Column Type Mapping” for details

12. **TApplicationException: get_next failed: out of sequence response: expected 4 but got 3**

This is due to concurrency bugs in the Thrift. It is recommended that you use the latest connector and compatible Flink version possible.

13. **DorisRuntimeException: Fail to abort transaction 26153 with url http://192.168.0.1:8040/api/table_name/_stream_load_2pc**

You can search for the log `abort transaction response` in TaskManager and determine whether it is a client issue or a server issue based on the HTTP return code.

14. **org.apache.flink.table.api.SqlParserException when using doris.filter.query: SQL parsing failed. “xx” encountered at row x, column xx**

This problem is mainly caused by the conditional varchar/string type, which needs to be quoted. The correct way to write it is xxx = ‘’xxx’’. In this way, the Flink SQL parser will interpret two consecutive single quotes as one single quote character instead of The end of the string, and the concatenated string is used as the value of the attribute. For example: `t1 >= '2024-01-01'` can be written as `'doris.filter.query' = 't1 >=''2024-01-01'''`.

15. **Failed to connect to backend:** `http://host:webserver_port`**, and BE is still alive**

The issue may have occurred due to configuring the IP address of `be`, which is not reachable by the external Flink cluster.This is mainly because when connecting to `fe`, the address of `be` is resolved through fe. For instance, if you add a be address as '127.0.0.1', the be address obtained by the Flink cluster through fe will be '127.0.0.1:webserver_port', and Flink will connect to that address. When this issue arises, you can resolve it by adding the actual corresponding external IP address of the be to the "with" attribute:`'benodes'="be_ip:webserver_port,be_ip:webserver_port..."`.For the entire database synchronization, the following properties are available`--sink-conf benodes=be_ip:webserver,be_ip:webserver...`.

16. **When using Flink-connector to synchronize MySQL data to Doris, there is a time difference of several hours between the timestamp.**

Flink Connector synchronizes the entire database from MySQL with a default timezone of UTC+8. If your data resides in a different timezone, you can adjust it using the following configuration, for example: `--mysql-conf debezium.date.format.timestamp.zone="UTC+3"`.

17. **What is the difference between batch writing and streaming writing**

Connector 1.5.0 and later support batch writing. Batch writing does not rely on Checkpoint. Data is cached in memory and the writing timing is controlled according to the parameters sink.buffer-flush.max-rows/sink.buffer-flush.max-bytes/sink.buffer-flush.interval. Checkpoint must be enabled for streaming writing. During the entire Checkpoint period, upstream data is continuously written to Doris, and data is not cached in memory all the time.

Doris Kafka Connector

Kafka Connect is a scalable and reliable tool for data transmission between Apache Kafka and other systems. Connectors can be defined Move large amounts of data in and out of Kafka.

The Doris community provides the doris-kafka-connector plug-in, which can write data in the Kafka topic to Doris.

Usage Doris Kafka Connector

Download

doris-kafka-connector

maven dependencies

```
<dependency>
  <groupId>org.apache.doris</groupId>
  <artifactId>doris-kafka-connector</artifactId>
  <version>25.0.0</version>
</dependency>
```

Standalone mode startup

Create the plugins directory under $KAFKA_HOME and put the downloaded doris-kafka-connector jar package into it
Configure config/connect-standalone.properties

```
# Modify broker address
bootstrap.servers=127.0.0.1:9092

# Modify to the created plugins directory
# Note: Please fill in the direct path to Kafka here. For example: plugin.path=/opt/kafka/plugins
plugin.path=$KAFKA_HOME/plugins

# It is recommended to increase the max.poll.interval.ms time of Kafka to more than 30 minutes, the default is 5 minutes
# Avoid Stream Load import data consumption timeout and consumers being kicked out of the consumer group
max.poll.interval.ms=1800000
consumer.max.poll.interval.ms=1800000
```

Configure doris-connector-sink.properties

Create doris-connector-sink.properties in the config directory and configure the following content:

```
name=test-doris-sink
connector.class=org.apache.doris.kafka.connector.DorisSinkConnector
topics=topic_test
doris.topic2table.map=topic_test:test_kafka_tbl
buffer.count.records=10000
buffer.flush.time=120
```

```
buffer.size.bytes=5000000
doris.urls=10.10.10.1
doris.http.port=8030
doris.query.port=9030
doris.user=root
doris.password=
doris.database=test_db
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
```

Start Standalone

```
$KAFKA_HOME/bin/connect-standalone.sh -daemon $KAFKA_HOME/config/connect-standalone.properties $KAFKA_HOME/config/doris-connector-sink.properties
```

> **Note: It is generally not recommended to use standalone mode in a production environment.**

Distributed mode startup

Create the plugins directory under $KAFKA_HOME and put the downloaded doris-kafka-connector jar package into it

Configure config/connect-distributed.properties

```
# Modify kafka server address
bootstrap.servers=127.0.0.1:9092

# Modify group.id, the same cluster needs to be consistent
group.id=connect-cluster

# Modify to the created plugins directory
# Note: Please fill in the direct path to Kafka here. For example: plugin.path=/opt/kafka/plugins
plugin.path=$KAFKA_HOME/plugins

# It is recommended to increase the max.poll.interval.ms time of Kafka to more than 30 minutes, the default is 5 minutes
# Avoid Stream Load import data consumption timeout and consumers being kicked out of the consumer group
max.poll.interval.ms=1800000
consumer.max.poll.interval.ms=1800000
```

Start Distributed

```
$KAFKA_HOME/bin/connect-distributed.sh -daemon $KAFKA_HOME/config/connect-distributed.properties
```

Add Connector

```
curl -i http://127.0.0.1:8083/connectors -H "Content-Type: application/json" -X POST -d '{
  "name":"test-doris-sink-cluster",
  "config":{
    "connector.class":"org.apache.doris.kafka.connector.DorisSinkConnector",
    "topics":"topic_test",
    "doris.topic2table.map": "topic_test:test_kafka_tbl",
    "buffer.count.records":"10000",
    "buffer.flush.time":"120",
```

```
    "buffer.size.bytes":"5000000",
    "doris.urls":"10.10.10.1",
    "doris.user":"root",
    "doris.password":"",
    "doris.http.port":"8030",
    "doris.query.port":"9030",
    "doris.database":"test_db",
    "key.converter":"org.apache.kafka.connect.storage.StringConverter",
    "value.converter":"org.apache.kafka.connect.json.JsonConverter"
  }
}'
```

Operation Connector

```
# View connector status
curl -i http://127.0.0.1:8083/connectors/test-doris-sink-cluster/status -X GET
# Delete connector
curl -i http://127.0.0.1:8083/connectors/test-doris-sink-cluster -X DELETE
# Pause connector
curl -i http://127.0.0.1:8083/connectors/test-doris-sink-cluster/pause -X PUT
# Restart connector
curl -i http://127.0.0.1:8083/connectors/test-doris-sink-cluster/resume -X PUT
# Restart tasks within the connector
curl -i http://127.0.0.1:8083/connectors/test-doris-sink-cluster/tasks/0/restart -X POST
```

Refer to: Connect REST Interface

> **Note that when kafka-connect is started for the first time, three topics `config.storage.topic` `offset.storage.topic` and `status.storage.topic` will be created in the kafka cluster to record the shared connector configuration of kafka-connect. Offset data and status updates. How to Use Kafka Connect - Get Started**

Access an SSL-certified Kafka cluster

Accessing an SSL-certified Kafka cluster through kafka-connect requires the user to provide a certificate file (client.truststore.jks) used to authenticate the Kafka Broker public key. You can add the following configuration in the `connect-distributed.properties` file:

```
# Connect worker
security.protocol=SSL
ssl.truststore.location=/var/ssl/private/client.truststore.jks
ssl.truststore.password=test1234

# Embedded consumer for sink connectors
consumer.security.protocol=SSL
consumer.ssl.truststore.location=/var/ssl/private/client.truststore.jks
consumer.ssl.truststore.password=test1234
```

For instructions on configuring a Kafka cluster connected to SSL authentication through kafka-connect, please refer to: Configure Kafka Connect

Dead letter queue

By default, any errors encountered during or during the conversion will cause the connector to fail. Each connector configuration can also tolerate such errors by skipping them, optionally writing the details of each error and failed operation as well as the records in question (with varying levels of detail) to a dead-letter queue for logging.

```
errors.tolerance=all
errors.deadletterqueue.topic.name=test_error_topic
errors.deadletterqueue.context.headers.enable=true
errors.deadletterqueue.topic.replication.factor=1
```

Configuration items

Key	Enum	Default Value	Required	Description
name	-	-	Y	Connect application name, must be unique within the Kafka Connect environment
connector.class	-	-	Y	org.apache.doris.kafka.connector.DorisSinkConnector
topics	-	-	Y	List of subscribed topics, separated by commas. like: topic1, topic2
doris.urls	-	-	Y	Doris FE connection address. If there are multiple, separate them with commas. like: 10.20.30.1,10.20.30.2,10.20.30.3

Key	Enum	Default Value	Required	Description
doris.http.port	-	-	Y	Doris HTTP protocol port
doris.query.port	-	-	Y	Doris MySQL protocol port
doris.user	-	-	Y	Doris username
doris.password	-	-	Y	Doris password

Key	Enum	Default Value	Required	Description
doris.database	-	-	Y	The database to write to. It can be empty when there are multiple libraries. At the same time, the specific library name needs to be configured in topic2table.map.
doris.topic2table.map	-	-	Y	The corresponding relationship between topic and table table, for example: topic1:tb1,topic2:tb2 If left blank, the topic will be used as the default table name for writing. The format of multiple libraries is topic1:db1.tbl1,topic2:db2.tbl2
buffer.count.records	-	50000	N	The number of records each Kafka partition buffers in memory before flushing to doris. Default 50000 records

Key	Enum	Default Value	Required	Description
buffer.flush.time	-	120	N	Buffer refresh interval, in seconds, default 120 seconds
buffer.size.bytes	-	10485760 (100MB)	N	The cumulative size of records buffered in memory for each Kafka partition, in bytes, default 100MB
jmx	-	true	N	To obtain connector internal monitoring indicators through JMX, please refer to: Doris-Connector-JMX
enable.2pc	-	true	N	Whether to enable two-phase commit (TwoPhaseCommit) of Stream Load, the default is true.

Key	Enum	Default Value	Required	Description
enable.delete	-	false	N	Whether to delete records synchronously, default false
label.prefix	-	${name}	N	Stream load label prefix when importing data. Defaults to the Connector application name.
auto.redirect	-	true	N	Whether to redirect StreamLoad requests. After being turned on, StreamLoad will redirect to the BE where data needs to be written through FE, and the BE information will no longer be displayed.
load.model	`stream_load`, `copy_into`	stream_load	N	How to import data. Supports `stream_load` to directly import data into Doris; also supports `copy_into` to import data into object storage, and then load the data into Doris.

Key	Enum	Default Value	Required	Description
sink.properties.*	-	`'sink.properties.format':'json', 'sink.properties.read_json_by_line':'true'`	No	Import parameters for Stream Load. For example: define column separator `'sink.properties.column_separator':','` Detailed parameter reference here **Enable Group Commit**, for example, enable group commit in sync_mode mode: `"sink.properties.group_commit":"sync_mode"`. Group Commit can be configured with three modes: `off_mode`, `sync_mode`, and `async_mode`. For specific usage, please refer to: Group-Commit **Enable partial column update**, for example, enable update of partial columns of specified col2: `"sink.properties.partial_columns":"true"`, `"sink.properties.columns": " col2"`,
delivery.guarantee	`at_least_once`, `exactly_once`	at_least_once	No	How to ensure data consistency when consuming Kafka data is imported into Doris. Supports `at_least_once` `exactly_once`, default is `at_least_once`. Doris needs to be upgraded to 2.1.0 or above to ensure data `exactly_once`
converter.mode	`normal`, `debezium_ingestion`	normal	No	Type conversion mode of upstream data when using Connector to consume Kafka data. `normal` means consuming data in Kafka normally without any type conversion. `debezium_ingestion` means that when Kafka upstream data is collected through CDC (Changelog Data Capture) tools such as Debezium, the upstream data needs to undergo special type conversion to support it.

Key	Enum	Default Value	Required	Description
debezium.schema.evolution	`none`, `basic`	none	N	Use Debezium to collect upstream database systems (such as MySQL), and when structural changes occur, the added fields can be synchronized to Doris. `none` means that when the structure of the upstream database system changes, the changed structure will not be synchronized to Doris. `basic` means synchronizing the data change operation of the upstream database. Since changing the column structure is a dangerous operation (it may lead to accidentally deleting columns of the Doris table structure), currently only the operation of adding columns synchronously upstream is supported. When a column is renamed, the old column remains unchanged, and the Connector will add a new column in the target table and sink the renamed new data into the new column.
database.time_zone	-	UTC	N	When `converter.mode` is not `normal` mode, it provides a way to specify time zone conversion for date data types (such as datetime, date, timestamp, etc.). The default is UTC time zone.

Key	Enum	Default Value	Required	Description
avro.topic2schema.filepath	-	-	No	By reading the locally provided Avro Schema file, the Avro file content in the Topic is parsed to achieve decoupling from the Schema registration center provided by Confluent. This configuration needs to be used with the `key.converter` or `value.converter` prefix. For example, the local Avro Schema file for configuring avro-user and avro-product Topic is as follows: `"value.converter.avro.topic2schema.filepath":"avro-user:file:///opt/avro_user.avsc, avro-product:file:///opt/avro_product.avsc"` For specific usage, please refer to: #32
record.tablename.field	-	-	No	Configure this parameter, data from one kafka topic can flow to multiple doris tables. For configuration details, refer to: #58

Key	Enum	Default Value	Required	Description
enable.combine.flush	`true`, `false`	false	N	Whether to merge all partition data before writing. The default value is false. Enabling this option (setting to true) is only supported for tables with unique keys. If the topic contains a large number of partitions, enabling this feature can reduce resource consumption.
max.retries	-	10	N	The maximum number of times to retry on errors before failing the task.
retry.interval.ms	-	6000	N	The time in milliseconds to wait following an error before attempting a retry.

Key	Enum	Default Value	Required	Description
behavior.on.null.values	ignore, fail	ignore	N	Defined how to handle records with null values.

For other Kafka Connect Sink common configuration items, please refer to: connect_configuring

Type mapping

Doris-kafka-connector uses logical or primitive type mapping to resolve the column's data type.
Primitive types refer to simple data types represented using Kafka connect's `Schema`. Logical data types usually use the `Struct` structure to represent complex types, or date and time types.

Kafka Primitive Type	Doris Type
INT8	TINYINT
INT16	SMALLINT
INT32	INT
INT64	BIGINT
FLOAT32	FLOAT
FLOAT64	DOUBLE
BOOLEAN	BOOLEAN
STRING	STRING
BYTES	STRING

Kafka Logical Type	Doris Type
org.apache.kafka.connect.data.Decimal	DECIMAL
org.apache.kafka.connect.data.Date	DATE
org.apache.kafka.connect.data.Time	STRING
org.apache.kafka.connect.data.Timestamp	DATETIME

Debezium Logical Type	Doris Type
io.debezium.time.Date	DATE
io.debezium.time.Time	String
io.debezium.time.MicroTime	DATETIME
io.debezium.time.NanoTime	DATETIME
io.debezium.time.ZonedTime	DATETIME
io.debezium.time.Timestamp	DATETIME
io.debezium.time.MicroTimestamp	DATETIME
io.debezium.time.NanoTimestamp	DATETIME
io.debezium.time.ZonedTimestamp	DATETIME
io.debezium.data.VariableScaleDecimal	DOUBLE

Best Practices

Load plain JSON data

1. Import data sample
   In Kafka, there is the following sample data

```
kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test-data-topic --from-beginning
{"user_id":1,"name":"Emily","age":25}
{"user_id":2,"name":"Benjamin","age":35}
{"user_id":3,"name":"Olivia","age":28}
{"user_id":4,"name":"Alexander","age":60}
{"user_id":5,"name":"Ava","age":17}
{"user_id":6,"name":"William","age":69}
{"user_id":7,"name":"Sophia","age":32}
{"user_id":8,"name":"James","age":64}
{"user_id":9,"name":"Emma","age":37}
{"user_id":10,"name":"Liam","age":64}
```

2. Create the table that needs to be imported
   In Doris, create the imported table, the specific syntax is as follows `sql CREATE TABLE test_db.test_kafka_connector_tbl( user_id BIGINT NOT NULL COMMENT "user id", name VARCHAR(20) COMMENT "name", age INT COMMENT "age" ) DUPLICATE KEY(user_id) DISTRIBUTED BY HASH(user_id) BUCKETS 12;`

3. Create an import task
   On the machine where Kafka-connect is deployed, submit the following import task through the curl

command `shell curl -i http://127.0.0.1:8083/connectors -H "Content-Type: application/json" -X POST -d '{ "name":"test-doris-sink-cluster", "config":{ "connector.class":"org.apache.doris.kafka.connector.DorisSinkConnector", "tasks.max":"10", "topics":"test-data-topic", "doris.topic2table.map": "test-data-topic:test_kafka_connector_tbl", "buffer.count.records":"10000", "buffer.flush.time":"120", "buffer.size.bytes":"5000000", "doris.urls":"10.10.10.1", "doris.user":"root", "doris.password":"", "doris.http.port":"8030", "doris.query.port":"9030", "doris.database":"test_db", "key.converter":"org.apache.kafka.connect.storage.StringConverter", "value.converter":"org.apache.kafka.connect.storage.StringConverter" } }'`

Load data collected by Debezium components

1. The MySQL database has the following table

```
CREATE TABLE test.test_user (
user_id int NOT NULL ,
name varchar(20),
age int,
PRIMARY KEY (user_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

insert into test.test_user values(1,'zhangsan',20);
insert into test.test_user values(2,'lisi',21);
insert into test.test_user values(3,'wangwu',22);
```

2. Create the imported table in Doris

```
CREATE TABLE test_db.test_user(
user_id            BIGINT       NOT NULL COMMENT "user id",
name               VARCHAR(20)           COMMENT "name",
age                INT                   COMMENT "age"
)
UNIQUE KEY(user_id)
DISTRIBUTED BY HASH(user_id) BUCKETS 12;
```

3. Deploy the Debezium connector for MySQL component, refer to: Debezium connector for MySQL
4. Create doris-kafka-connector import task
   Assume that the MySQL table data collected through Debezium is in the `mysql_debezium.test.test_user` Topic

```
curl -i http://127.0.0.1:8083/connectors -H "Content-Type: application/json" -X POST -d '{
"name":"test-debezium-doris-sink",
"config":{
"connector.class":"org.apache.doris.kafka.connector.DorisSinkConnector",
"tasks.max":"10",
"topics":"mysql_debezium.test.test_user",
"doris.topic2table.map": "mysql_debezium.test.test_user:test_user",
"buffer.count.records":"10000",
"buffer.flush.time":"120",
"buffer.size.bytes":"5000000",
"doris.urls":"10.10.10.1",
"doris.user":"root",
"doris.password":"",
"doris.http.port":"8030",
"doris.query.port":"9030",
```

```
   "doris.database":"test_db",
   "converter.mode":"debezium_ingestion",
   "enable.delete":"true",
   "key.converter":"org.apache.kafka.connect.json.JsonConverter",
   "value.converter":"org.apache.kafka.connect.json.JsonConverter"
   }
   }'
```

Load Avro serialized data

```
curl -i http://127.0.0.1:8083/connectors -H "Content-Type: application/json" -X POST -d '{
  "name":"doris-avro-test",
  "config":{
    "connector.class":"org.apache.doris.kafka.connector.DorisSinkConnector",
    "topics":"avro_topic",
    "tasks.max":"10",
    "doris.topic2table.map": "avro_topic:avro_tab",
    "buffer.count.records":"100000",
    "buffer.flush.time":"120",
    "buffer.size.bytes":"10000000",
    "doris.urls":"127.0.0.1",
    "doris.user":"root",
    "doris.password":"",
    "doris.http.port":"8030",
    "doris.query.port":"9030",
    "doris.database":"test",
    "load.model":"stream_load",
    "key.converter":"io.confluent.connect.avro.AvroConverter",
    "key.converter.schema.registry.url":"http://127.0.0.1:8081",
    "value.converter":"io.confluent.connect.avro.AvroConverter",
    "value.converter.schema.registry.url":"http://127.0.0.1:8081"
  }
}'
```

Load Protobuf serialized data

```
curl -i http://127.0.0.1:8083/connectors -H "Content-Type: application/json" -X POST -d '{
  "name":"doris-protobuf-test",
  "config":{
    "connector.class":"org.apache.doris.kafka.connector.DorisSinkConnector",
    "topics":"proto_topic",
    "tasks.max":"10",
    "doris.topic2table.map": "proto_topic:proto_tab",
    "buffer.count.records":"100000",
    "buffer.flush.time":"120",
    "buffer.size.bytes":"10000000",
    "doris.urls":"127.0.0.1",
    "doris.user":"root",
    "doris.password":"",
    "doris.http.port":"8030",
    "doris.query.port":"9030",
    "doris.database":"test",
    "load.model":"stream_load",
    "key.converter":"io.confluent.connect.protobuf.ProtobufConverter",
    "key.converter.schema.registry.url":"http://127.0.0.1:8081",
    "value.converter":"io.confluent.connect.protobuf.ProtobufConverter",
    "value.converter.schema.registry.url":"http://127.0.0.1:8081"
  }
}'
```

Loading Data with Kafka Connect Single Message Transforms

For example, consider data in the following format:

```
{
  "registertime": 1513885135404,
  "userid": "User_9",
  "regionid": "Region_3",
  "gender": "MALE"
}
```

To add a hard-coded column to Kafka messages, InsertField can be used. Additionally, TimestampConverter can be used to convert Bigint type timestamps to time strings.

```
curl -i http://127.0.0.1:8083/connectors -H "Content-Type: application/json" -X POST -d '{
  "name": "insert_field_tranform",
  "config": {
    "connector.class": "org.apache.doris.kafka.connector.DorisSinkConnector",
    "tasks.max": "1",
    "topics": "users",
    "doris.topic2table.map": "users:kf_users",
    "buffer.count.records": "10",
    "buffer.flush.time": "11",
    "buffer.size.bytes": "5000000",
    "doris.urls": "127.0.0.1:8030",
    "doris.user": "root",
    "doris.password": "123456",
    "doris.http.port": "8030",
    "doris.query.port": "9030",
    "doris.database": "testdb",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter.schemas.enable": "false",
    "transforms": "InsertField,TimestampConverter",
    // Insert Static Field
    "transforms.InsertField.type": "org.apache.kafka.connect.transforms.InsertField$Value",
    "transforms.InsertField.static.field": "repo",
    "transforms.InsertField.static.value": "Apache Doris",
    // Convert Timestamp Format
    "transforms.TimestampConverter.type": "org.apache.kafka.connect.transforms.TimestampConverter$Value",
    "transforms.TimestampConverter.field": "registertime",
    "transforms.TimestampConverter.format": "yyyy-MM-dd HH:mm:ss.SSS",
    "transforms.TimestampConverter.target.type": "string"
  }
}'
```

After InsertField and TimestampConverter transformations, the data becomes:

```
{
  "userid": "User_9",
  "regionid": "Region_3",
  "gender": "MALE",
  "repo": "Apache Doris",// Static field added
   "registertime": "2017-12-21 03:38:55.404"  // Unix timestamp converted to string
}
```

For more examples of Kafka Connect Single Message Transforms (SMT), please refer to the SMT documentation.

FAQ

1. The following error occurs when reading Json type data:

```
Caused by: org.apache.kafka.connect.errors.DataException: JsonConverter with schemas.enable requires "schema" and "payload" fields
and may not contain additional fields. If you are trying to deserialize plain JSON data, set schemas.enable=false in your converter
configuration.
    at org.apache.kafka.connect.json.JsonConverter.toConnectData(JsonConverter.java:337)
    at org.apache.kafka.connect.storage.Converter.toConnectData(Converter.java:91)
    at org.apache.kafka.connect.runtime.WorkerSinkTask.lambda$convertAndTransformRecord$4(WorkerSinkTask.java:536)
    at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execAndRetry(RetryWithToleranceOperator.java:180)
    at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execAndHandleError(RetryWithToleranceOperator.java:214)
```

reason: This is because using the `org.apache.kafka.connect.json.JsonConverter` converter requires matching the “schema” and “payload” fields.

Two solutions, choose one: 1. Replace `org.apache.kafka.connect.json.JsonConverter` with `org.apache.kafka.connect.storage.StringConverter` 2. If the startup mode is **Standalone** mode, change `value.converter.schemas.enable` or `key.converter.schemas.enable` in config/connect-standalone.properties to false; If the startup mode is **Distributed** mode, change `value.converter.schemas.enable` or `key.converter.schemas.enable` in config/connect-distributed.properties to false

2. The consumption times out and the consumer is kicked out of the consumption group:

```
org.apache.kafka.clients.consumer.CommitFailedException: Offset commit cannot be completed since the consumer is not part of an
active group for auto partition assignment; it is likely that the consumer was kicked out of the group.
        at org.apache.kafka.clients.consumer.internals.ConsumerCoordinator.sendOffsetCommitRequest(ConsumerCoordinator.java:1318)
        at org.apache.kafka.clients.consumer.internals.ConsumerCoordinator.doCommitOffsetsAsync(ConsumerCoordinator.java:1127)
        at org.apache.kafka.clients.consumer.internals.ConsumerCoordinator.commitOffsetsAsync(ConsumerCoordinator.java:1093)
        at org.apache.kafka.clients.consumer.KafkaConsumer.commitAsync(KafkaConsumer.java:1590)
        at org.apache.kafka.connect.runtime.WorkerSinkTask.doCommitAsync(WorkerSinkTask.java:361)
        at org.apache.kafka.connect.runtime.WorkerSinkTask.doCommit(WorkerSinkTask.java:376)
        at org.apache.kafka.connect.runtime.WorkerSinkTask.commitOffsets(WorkerSinkTask.java:467)
        at org.apache.kafka.connect.runtime.WorkerSinkTask.commitOffsets(WorkerSinkTask.java:381)
        at org.apache.kafka.connect.runtime.WorkerSinkTask.iteration(WorkerSinkTask.java:221)
        at org.apache.kafka.connect.runtime.WorkerSinkTask.execute(WorkerSinkTask.java:206)
        at org.apache.kafka.connect.runtime.WorkerTask.doRun(WorkerTask.java:204)
        at org.apache.kafka.connect.runtime.WorkerTask.run(WorkerTask.java:259)
        at org.apache.kafka.connect.runtime.isolation.Plugins.lambda$withClassLoader$1(Plugins.java:181)
        at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:539)
        at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
        at java.base/java.lang.Thread.run(Thread.java:833)
```

Solution:

Increase `max.poll.interval.ms` in Kafka according to the scenario. The default value is `300000` - If it is started in Standalone mode, add the `max.poll.interval.ms` and `consumer.max.poll.interval.ms` parameters in the configuration file of config/connect-standalone.properties, and configure the parameter values. - If it is started in Distributed mode, add the `max.poll.interval.ms` and `consumer.max.poll.interval.ms` parameters in the configuration file of config/connect-distributed.properties, and configure the parameter values.

After adjusting the parameters, restart kafka-connect

3. Doris-kafka-connector reports an error when upgrading version from 1.0.0 or 1.1.0 to 24.0.0

```
org.apache.kafka.common.config.ConfigException: Topic 'connect-status' supplied via the 'status.storage.topic' property is required
to have 'cleanup.policy=compact' to guarantee consistency and durability of connector and task statuses, but found the topic
currently has 'cleanup.policy=delete'. Continuing would likely result in eventually losing connector and task statuses and problems
restarting this Connect cluster in the future. Change the 'status.storage.topic' property in the Connect worker configurations to
use a topic with 'cleanup.policy=compact'.
    at org.apache.kafka.connect.util.TopicAdmin.verifyTopicCleanupPolicyOnlyCompact(TopicAdmin.java:581)
    at org.apache.kafka.connect.storage.KafkaTopicBasedBackingStore.lambda$topicInitializer$0(KafkaTopicBasedBackingStore.java:47)
    at org.apache.kafka.connect.util.KafkaBasedLog.start(KafkaBasedLog.java:247)
    at org.apache.kafka.connect.util.KafkaBasedLog.start(KafkaBasedLog.java:231)
    at org.apache.kafka.connect.storage.KafkaStatusBackingStore.start(KafkaStatusBackingStore.java:228)
    at org.apache.kafka.connect.runtime.AbstractHerder.startServices(AbstractHerder.java:164)
    at org.apache.kafka.connect.runtime.distributed.DistributedHerder.run
```

Solution: Adjust the clearing strategy of `connect-configs` `connect-status` Topic to compact

```
$KAFKA_HOME/bin/kafka-configs.sh --alter --entity-type topics --entity-name connect-configs --add-config cleanup.policy=compact --
bootstrap-server 127.0.0.1:9092
$KAFKA_HOME/bin/kafka-configs.sh --alter --entity-type topics --entity-name connect-status --add-config cleanup.policy=compact --
bootstrap-server 127.0.0.1:9092
```

4. Table schema change failed in `debezium_ingestion` converter mode

```
[2025-01-07 14:26:20,474] WARN [doris-normal_test_sink-connector|task-0] Table 'test_sink' cannot be altered because schema
evolution is disabled. (org.apache.doris.kafka.connector.converter.RecordService:183)
[2025-01-07 14:26:20,475] ERROR [doris-normal_test_sink-connector|task-0] WorkerSinkTask{id=doris-normal_test_sink-connector-0} Task
threw an uncaught and unrecoverable exception. Task is being killed and will not recover until manually restarted. Error: Cannot
alter table org.apache.doris.kafka.connector.model.TableDescriptor@67cd8027 because schema evolution is disabled
(org.apache.kafka.connect.runtime.WorkerSinkTask:612)
org.apache.doris.kafka.connector.exception.SchemaChangeException: Cannot alter table
org.apache.doris.kafka.connector.model.TableDescriptor@67cd8027 because schema evolution is disabled
    at org.apache.doris.kafka.connector.converter.RecordService.alterTableIfNeeded(RecordService.java:186)
    at org.apache.doris.kafka.connector.converter.RecordService.checkAndApplyTableChangesIfNeeded(RecordService.java:150)
    at org.apache.doris.kafka.connector.converter.RecordService.processStructRecord(RecordService.java:100)
    at org.apache.doris.kafka.connector.converter.RecordService.getProcessedRecord(RecordService.java:305)
    at org.apache.doris.kafka.connector.writer.DorisWriter.putBuffer(DorisWriter.java:155)
    at org.apache.doris.kafka.connector.writer.DorisWriter.insertRecord(DorisWriter.java:124)
    at org.apache.doris.kafka.connector.writer.StreamLoadWriter.insert(StreamLoadWriter.java:151)
    at org.apache.doris.kafka.connector.service.DorisDefaultSinkService.insert(DorisDefaultSinkService.java:154)
    at org.apache.doris.kafka.connector.service.DorisDefaultSinkService.insert(DorisDefaultSinkService.java:135)
    at org.apache.doris.kafka.connector.DorisSinkTask.put(DorisSinkTask.java:97)
    at org.apache.kafka.connect.runtime.WorkerSinkTask.deliverMessages(WorkerSinkTask.java:583)
    at org.apache.kafka.connect.runtime.WorkerSinkTask.poll(WorkerSinkTask.java:336)
    at org.apache.kafka.connect.runtime.WorkerSinkTask.iteration(WorkerSinkTask.java:237)
    at org.apache.kafka.connect.runtime.WorkerSinkTask.execute(WorkerSinkTask.java:206)
    at org.apache.kafka.connect.runtime.WorkerTask.doRun(WorkerTask.java:202)
    at org.apache.kafka.connect.runtime.WorkerTask.run(WorkerTask.java:257)
    at org.apache.kafka.connect.runtime.isolation.Plugins.lambda$withClassLoader$1(Plugins.java:177)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)
    at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
    at java.base/java.lang.Thread.run(Thread.java:829)
```

Solution:

In `debezium_ingestion` converter mode, table schema changes are turned off by default. You need to configure `debezium.schema.evolution` to `basic` to enable table schema changes.
It should be noted that enabling table structure changes does not accurately keep this changed column as the

only column in the Doris table (see `debezium.schema.evolution` parameter description for details). If you need to keep only unique columns in the upstream and downstream, it is best to manually add the changed columns to the Doris table, and then restart the Connector task. The Connector will continue to consume the unconsumed `offset` to maintain data consistency.

Doris Streamloader

Overview

Doris Streamloader is a client tool designed for loading data into Apache Doris. In comparison to single-threaded load using `curl`, it reduces the load latency of large datasets by its concurrent loading capabilities. It comes with the following features:

- **Parallel loading**: multi-threaded load for the Stream Load method. You can set the parallelism level using the `workers` parameter.
- **Multi-file load:** simultaneously load of multiple files and directories with one shot. It supports recursive file fetching and allows you to specify file names with wildcard characters.
- **Path traversal support:** support path traversal when the source files are in directories
- **Resilience and continuity:** in case of partial load failures, it can resume data loading from the point of failure.
- **Automatic retry mechanism:** in case of loading failures, it can automatically retry a default number of times. If the loading remains unsuccessful, it will print the command for manual retry.

Installation

Source Code: https://github.com/apache/doris-streamloader/ Binary File: https://doris.apache.org/download

> **The obtained result is the executable binary.**

How to use

```
doris-streamloader --source_file={FILE_LIST} --url={FE_OR_BE_SERVER_URL}:{PORT} --header={STREAMLOAD_HEADER} --db={TARGET_DATABASE}
--table={TARGET_TABLE}
```

1. `FILE_LIST` **support:**

- Single file

  E.g. Load a single file

  ```
  doris-streamloader --source_file="dir" --url="http://localhost:8330" --header="column_separator:|?columns:col1,col2" --
  db="testdb" --table="testtbl"
  ```

- Single directory

  E.g. Load a single directory

```
doris-streamloader --source_file="dir" --url="http://localhost:8330" --header="column_separator:|?columns:col1,col2" --
db="testdb" --table="testtbl"
```

- File names with wildcard characters (enclosed in quotes)

  E.g. Load file0.csv, file1.csv, file2.csv

```
doris-streamloader --source_file="file*" --url="http://localhost:8330" --header="column_separator:|?columns:col1,col2" --
db="testdb" --table="testtbl"
```

- A list of files separated by commas

  E.g. Load file0.csv, file1.csv, file2.csv

```
 doris-streamloader --source_file="file0.csv,file1.csv,file2.csv" --url="http://localhost:8330" --
header="column_separator:|?columns:col1,col2" --db="testdb" --table="testtbl"
```

- A list of directories separated by commas

  E.g. Load dir1, dir2, dir3

```
 doris-streamloader --source_file="dir1,dir2,dir3" --url="http://localhost:8330" --header="column_separator:|?
columns:col1,col2" --db="testdb" --table="testtbl"
```

2. `STREAMLOAD_HEADER` **supports all streamload headers separated with '?' if there is more than one**

Example:

```
doris-streamloader --source_file="data.csv" --url="http://localhost:8330" --header="column_separator:|?columns:col1,col2" --
db="testdb" --table="testtbl"
```

The parameters above are required, and the following parameters are optional:

Parameter	Description	Default Value	Suggestions
–u	Username of the database	root	
–p	Password	empty string	
–compress	Whether to compress data upon HTTP transmission	false	Remain as default. Compression and decompression can increase pressure on Doris Streamloader side and the CPU resources on Doris BE side, so it is advised to only enable this when network bandwidth is constrained.
–timeout	Timeout of the HTTP request sent to Doris (seconds)	60*60*10	Remain as default
–batch	Granularity of batch reading and sending of files (rows)	4096	Remain as default

Parameter	Description	Default Value	Suggestions
–batch_byte	Granularity of batch reading and sending of files (byte)	943718400 (900MB)	Remain as default
–workers	Concurrency level of data loading	0	“0” means the auto mode, in which the streamload speed is based on the data size and disk throughput. You can dial this up for a high-performance cluster, but it is advised to keep it below 10. If you observe excessive memory usage (via the memtracker in log), you can dial this down.
–disk_throughput	Disk throughput (MB/s)	800	Usually remain as default. This parameter is a basis of the automatic inference of workers. You can adjust this based on your needs to get a more appropriate value of workers.
–streamload_throughput	Streamload throughput (MB/s)	100	Usually remain as default. The default value is derived from the streamload throughput and predicted performance provided by the daily performance testing environment. To get a more appropriate value of workers, you can configure this based on your measured streamload throughput: (LoadBytes*1000)/(LoadTimeMs* 1024*1024)
–max_byte_per_task	Maximum data size for each load task. For a dataset exceeding this size, the remaining part will be split into a new load task.	107374182400 (100G)	This is recommended to be large in order to reduce the number of load versions. However, if you encounter a “body exceed max size” and try to avoid adjusting the streaming_load_max_mb parameter (which requires restarting the backend), or if you encounter a “-238 TOO MANY SEGMENT” error, you can temporarily dial this down.
–check_utf8		true	Remain as default
–debug	Print debug log	false	Remain as default

Parameter	Description	Default Value	Suggestions
–auto_retry	The list of failed workers and tasks for auto retry	empty string	This is only used when there is an load failure. The serial numbers of the failed workers and tasks will be shown and all you need is to copy and execute the the entire command. For example, if –auto_retry="1,1;2,1", that means the failed tasks include the first task in the first worker and the first task in the second worker.
–auto_retry_times	Times of auto retries	3	Remain as default. If you don't need retries, you can set this to 0.
–auto_retry_interval	Interval of auto retries	60	Remain as default. If the load failure is caused by a Doris downtime, it is recommended to set this parameter based on the restart interval of Doris.
–log_filename	Path for log storage	""	Logs are printed to the console by default. To print them to a log file, you can set the path, such as –log_filename="/var/log".

Result description

A result will be returned no matter the data loading succeeds or fails.

Parameter	Description
Status	Loading succeeded or failed
TotalRows	Total number of rows
FailLoadRows	Number of rows failed to be loaded
LoadedRows	Number of rows loaded
FilteredRows	Number of rows filtered
UnselectedRows	Number of rows unselected
LoadBytes	Number of bytes loaded
LoadTimeMs	Actual loading time
LoadFiles	List of loaded files

Examples:

- If the loading succeeds, you will see a result like:

```
Load Result: {
        "Status": "Success",
```

```
        "TotalRows": 120,
        "FailLoadRows": 0,
        "LoadedRows": 120,
        "FilteredRows": 0,
        "UnselectedRows": 0,
        "LoadBytes": 40632,
        "LoadTimeMs": 971,
        "LoadFiles": [
                "basic.csv",
                "basic_data1.csv",
                "basic_data2.csv",
                "dir1/basic_data.csv",
                "dir1/basic_data.csv.1",
                "dir1/basic_data1.csv"
        ]
}
```

- If the loading fails (or partially fails), you will see a retry message:

```
load has some error and auto retry failed, you can retry by :
./doris-streamloader --source_file /mnt/disk1/laihui/doris/tools/tpch-tools/bin/tpch-data/lineitem.tbl.1  --
url="http://127.0.0.1:8239" --header="column_separator:|?columns: l_orderkey, l_partkey, l_suppkey, l_linenumber,
l_quantity, l_extendedprice, l_discount, l_tax, l_returnflag,l_linestatus,
l_shipdate,l_commitdate,l_receiptdate,l_shipinstruct,l_shipmode,l_comment,temp" --db="db" --table="lineitem1" -u root -p ""
--compress=false --timeout=36000 --workers=3 --batch=4096 --batch_byte=943718400 --max_byte_per_task=1073741824 --
check_utf8=true --report_duration=1 --auto_retry="2,1;1,1;0,1" --auto_retry_times=0 --auto_retry_interval=60
```

You can copy and execute the command. The failure message will also be provided:

```
Load Result: {
      "Status": "Failed",
      "TotalRows": 1,
      "FailLoadRows": 1,
      "LoadedRows": 0,
      "FilteredRows": 0,
      "UnselectedRows": 0,
      "LoadBytes": 0,
      "LoadTimeMs": 104,
      "LoadFiles": [
              "/mnt/disk1/laihui/doris/tools/tpch-tools/bin/tpch-data/lineitem.tbl.1"
      ]
}
```

Best practice

Parameter suggestions

1. Required parameters:
   `--source_file=FILE_LIST --url=FE_OR_BE_SERVER_URL_WITH_PORT --header=STREAMLOAD_HEADER --db=TARGET_DATABASE --table=TARGET_TABLE` If you need to load multiple files, you should configure all of them at a time in `source_file`.

2. The default value of `workers` is the number of CPU cores. When that is large, for example, 96 cores, the value of `workers` should be dialed down. **The recommended value for most cases is 8.**

3. `max_byte_per_task` is recommended to be large in order to reduce the number of load versions. However, if you encounter a “body exceed max size” and try to avoid adjusting the streaming_load_max_mb parameter (which requires restarting the backend), or if you encounter a `-238 TOO MANY SEGMENT` error, you can temporarily dial this down. **For most cases, this can remain as default.**

Two parameters that impacts the number of versions:

- `workers`: The more `workers`, the higher concurrency level, and thus the more versions. The recommended value for most cases is 8.
- `max_byte_per_task`: The larger `max_byte_per_task` , the larger data size in one single version, and thus the less versions. However, if this is excessively high, it could easily cause an `-238 TOO MANY SEGMENT` error. For most cases, this can remain as default.

Recommended commands

In most cases, you only need to set the required parameters and `workers`.

```
./doris-streamloader --source_file="demo.csv,demoFile*.csv,demoDir" --url="http://127.0.0.1:8030" --header="column_separator:," --db="demo" --table="test_load" --u="root" --workers=8
```

FAQ

- Before resumable loading was available, to fix any partial failures in loading would require deleting the current table and starting over. In this case, Doris Streamloader would retry automatically. If the retry fails, a retry command will be printed so you can copy and execute it.
- The default maximum data loading size for Doris Streamloader is limited by BE config `streaming_load_max_mb` (default: 100GB). If you don’t want to restart BE, you can also dial down `max_byte_per_task`.

  To show current `streaming_load_max_mb`:

  ```
  curl "http://127.0.0.1:8040/api/show_config"
  ```

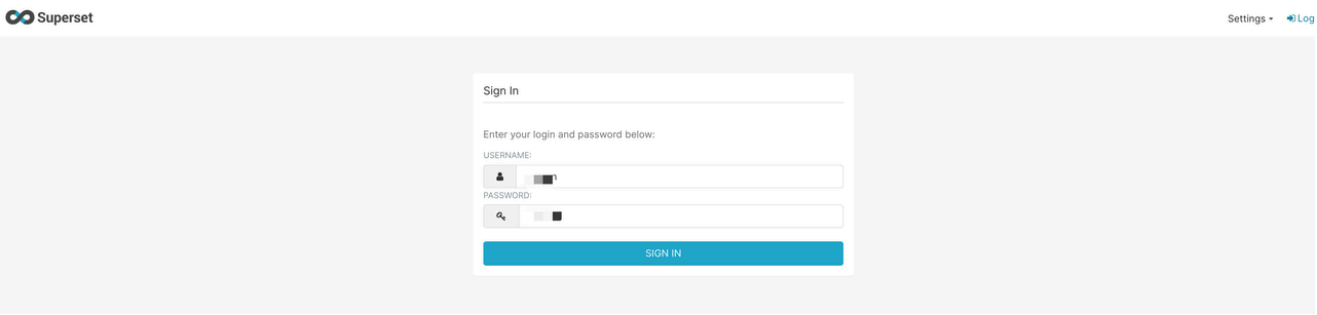
- If you encounter an `-238 TOO MANY SEGMENT` error, you can dial down `max_byte_per_task`.

Apache Superset

Introduction

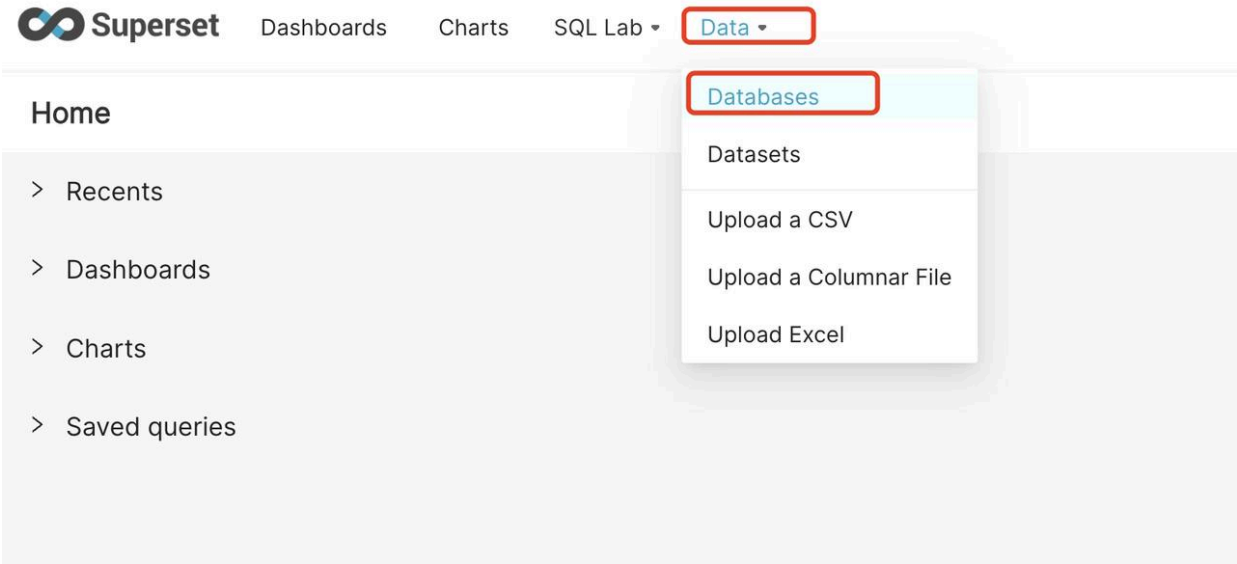
Apache Superset is an open-source data exploration platform. It supports a rich variety of data source connections and numerous visualization methods. It also enables fine-grained access control for users. The main features of this tool include self-service analysis, customizable dashboards, visualization of analytical results (with export functionality), and user/role permission control. Moreover, it integrates an SQL editor for conducting SQL editing and queries.

In Apache Superset version 3.1 official support has been introduced for querying and visualizing both internal and external data from Apache Doris. ### Preconditions Ensure you have completed the following tool installations: 1. Install the Python client for Apache Doris on the Apache Superset server. pip install pydoris 2. Install Apache Superset version 3.1 or above. For detailed instructions, refer to Installing Superset from PyPI or Installing Superset Locally Using Docker Compose. ### Add data source 1. You can access Superset by visiting the corresponding startup port.



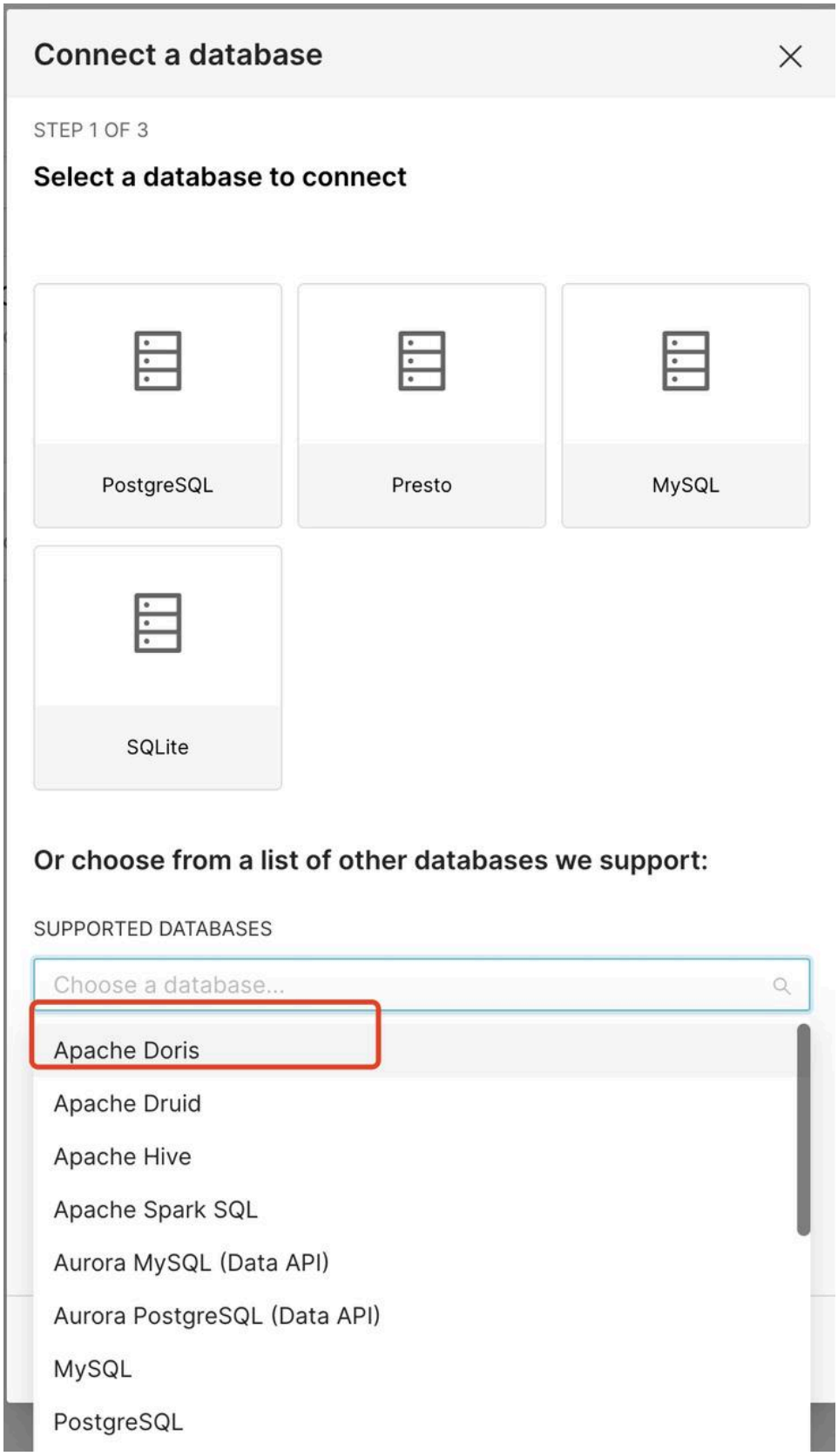
login page

2. Select the “Add Database Connection” option after logging into Superset.



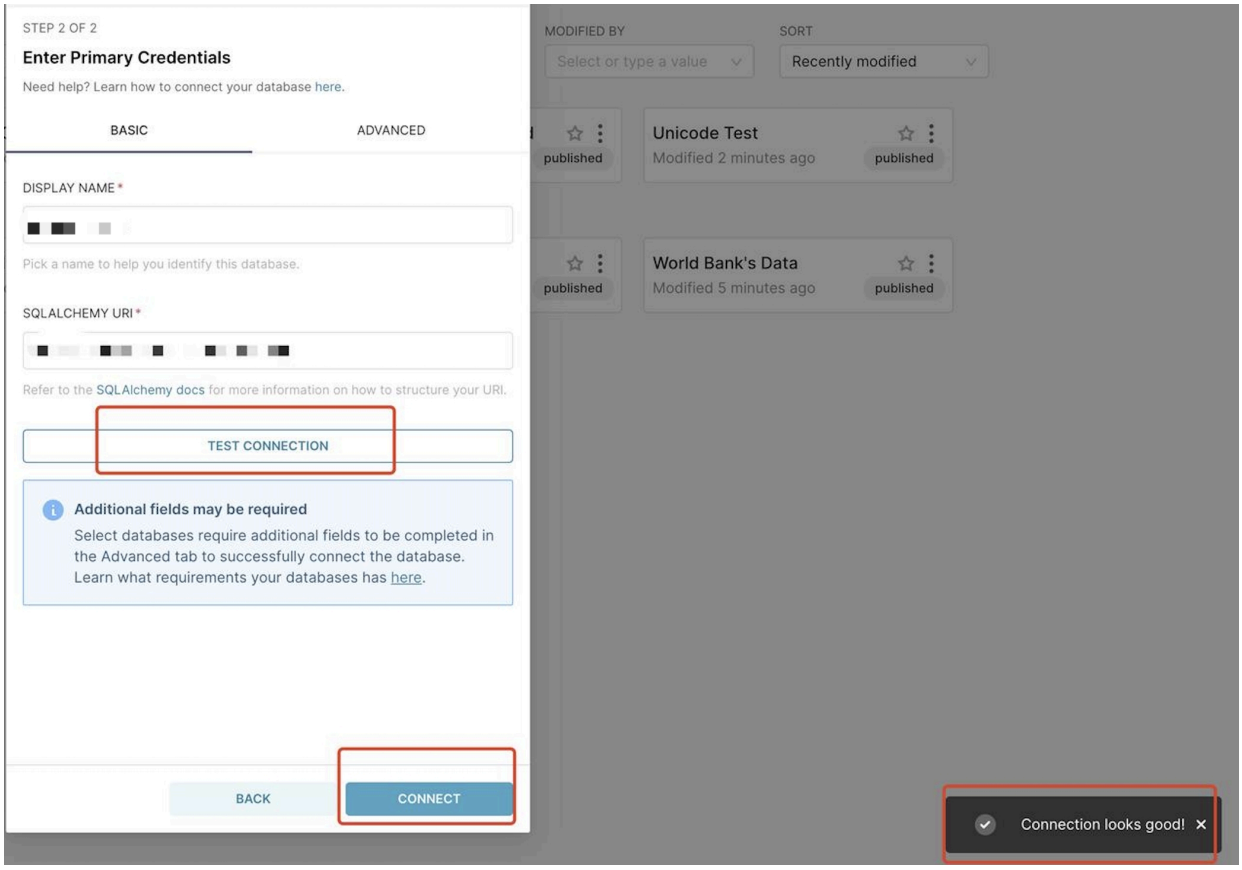
add databases

3. In the connection page, select the “Apache Doris” option.



select databases

4. Fill in the SQLAlchemy URI information in the connection details and proceed with the relevant connectivity verification.



test connection

When creating a data source in Apache Superset, please pay attention to the following two points: - Choose Apache Doris as the data source in SUPPORTED DATABASES. - In the SQLAlchemy URI, fill in the URI following the Apache Doris SQLAlchemy URI format as shown below.

`doris://<User>:<Password>@<Host>:<Port>/<Catalog>.<Database>` - URI parameters are explained as follows: - User: The username for logging into the Apache Doris cluster, e.g., admin. - Password: The password for logging into the Apache Doris cluster. - Host: The IP address of the FE (Frontend) host in the Apache Doris cluster. - Port: The query port of the FE in the Apache Doris cluster, e.g. 9030. - Catalog: The target Catalog in the Apache Doris cluster. Both Internal Catalog and External Catalog are supported. - Database: The target database in the Apache Doris cluster. Both internal and external databases are supported.

> **1. When deploying Apache Superset using the latest Docker image, if you encounter the issue of not finding the Apache Doris data source, it may be because the default Apache Superset Docker image includes only basic data source builds. You need to manually install the pydoris package. You can refer to the 'How to extend this image' section in the Apache Superset Docker tutorial for the deployment steps of Apache Superset.**
>
> 2. It is recommended to use Apache Doris 2.0.4 and above.

FineBI

FineBI Introduction

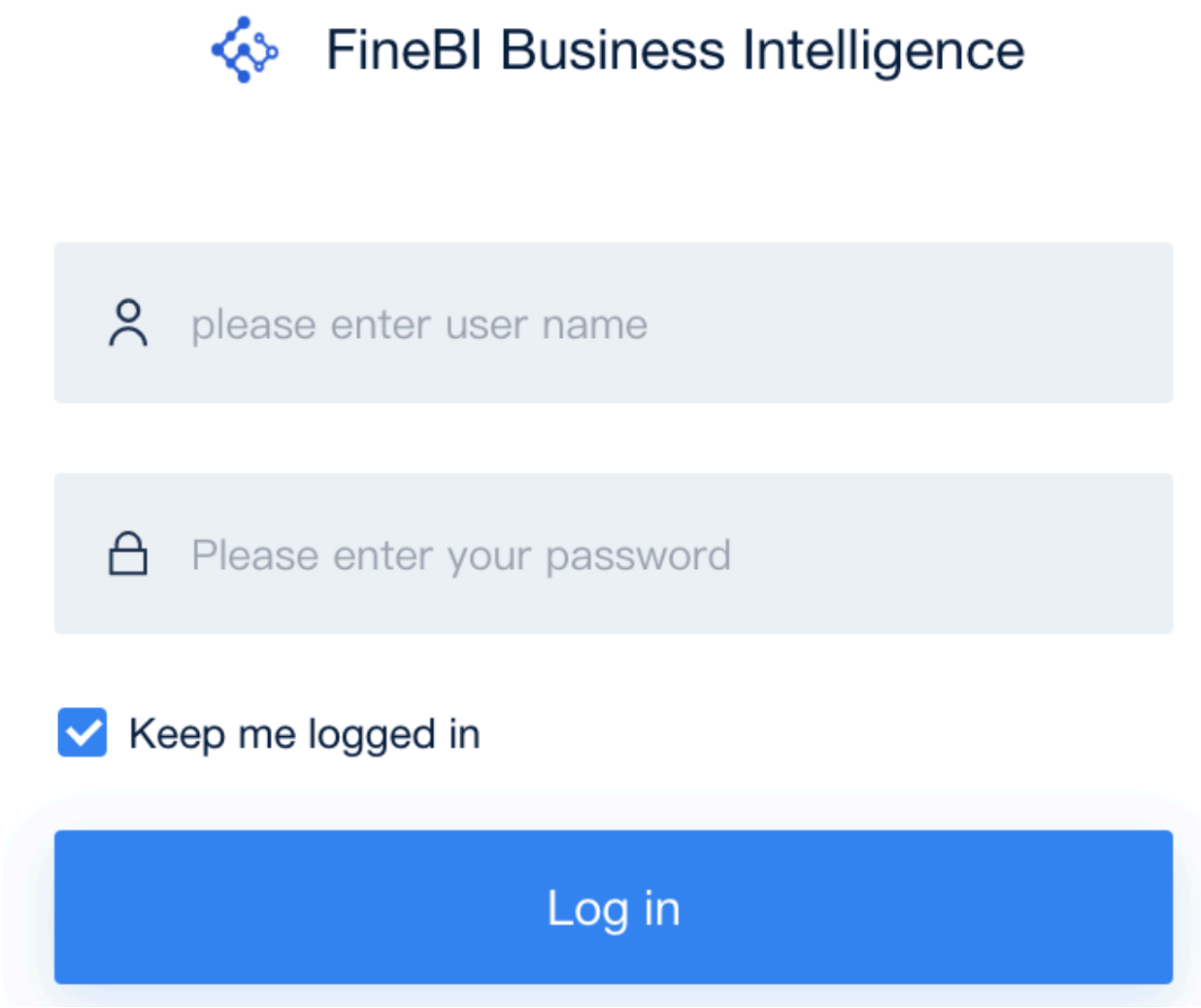
As a business intelligence product, FineBI has a system architecture of data processing, real-time analysis, multidimensional analysis Dashboard and other functions. FineBI supports rich data source connection and analysis and management of tables with multiple views. FineBI can successfully support the modeling and visualization of internal and external data of Apache Doris.

Precondition

Install FineBI 5.0 or later, Download link: https://intl.finebi.com/

Login and Connection

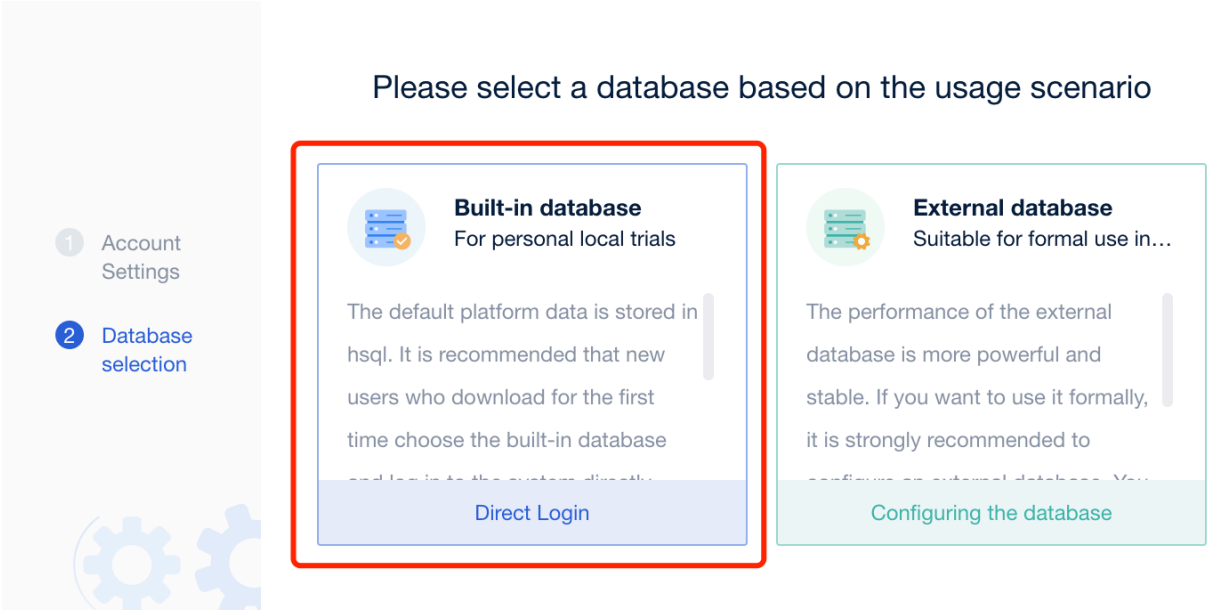
1. Create account and log in FineBI



login page

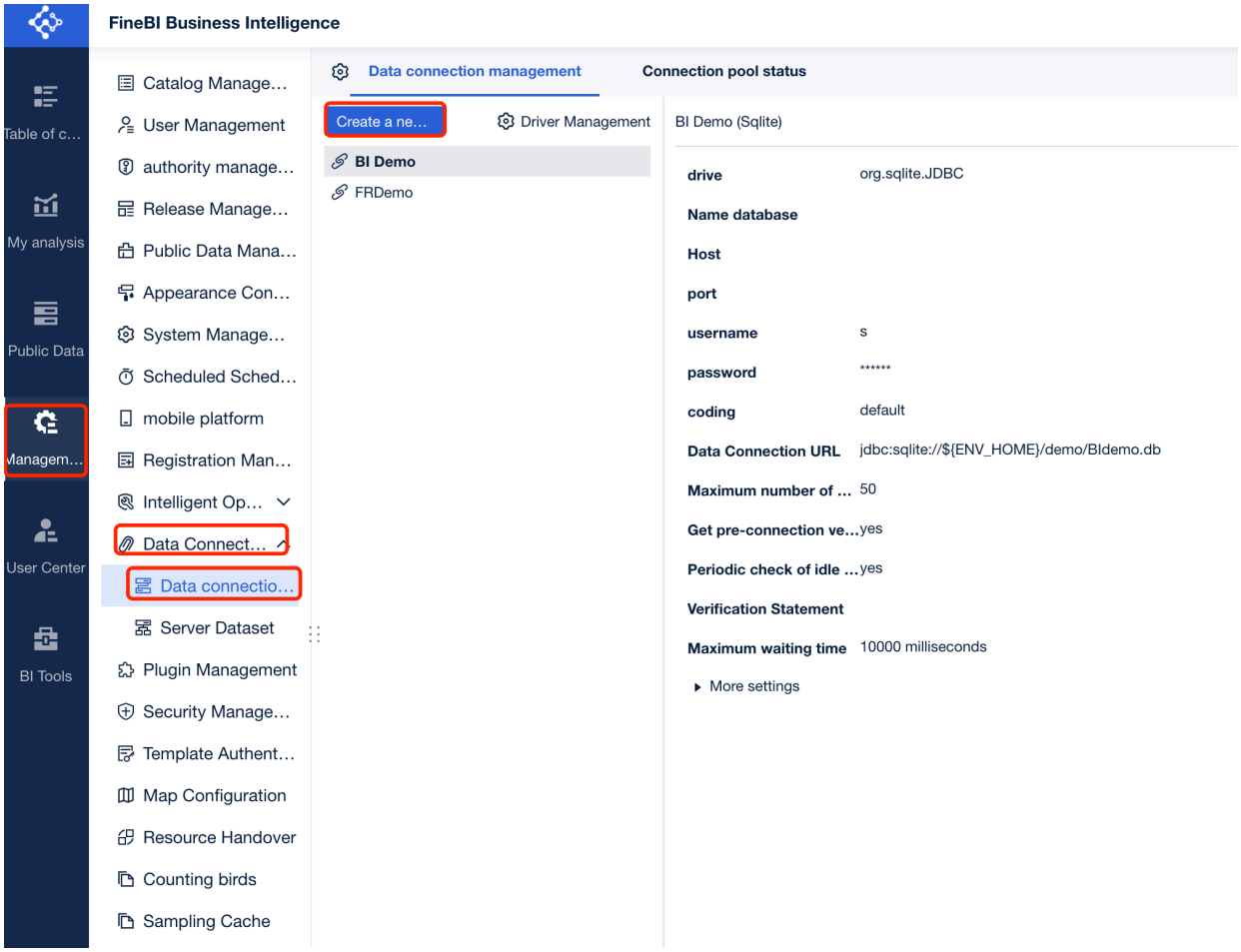
2. Select the Built-in database, If you need to select an external database configuration, the documentation is available：https://help.fanruan.com/finebi-en/doc-view-4437.html

   :::info Note It is recommended to select the built-in database as the information repository of FineBI. The database type selected here is not the target database for querying and analyzing data, but the database for storing and maintaining FineBI model, dashboard and other information. FineBI needs to add, delete, modify and check it. :::



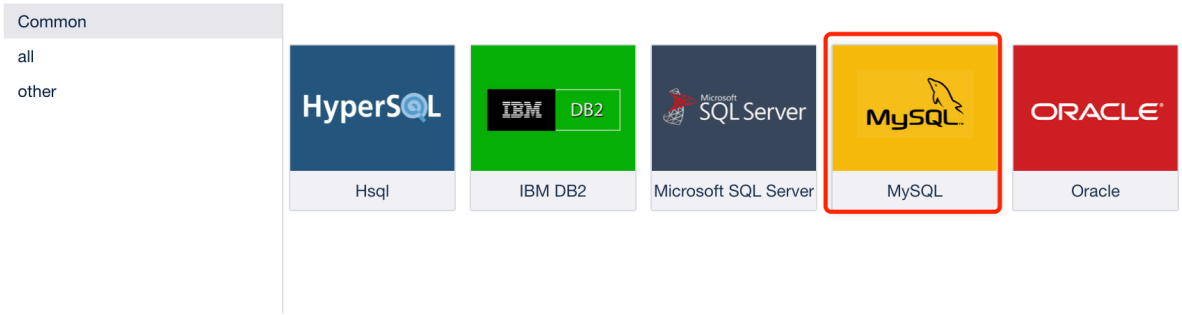
select database

3. Click the Management System button and select the database connection management in Data Connections to create a new database connection.



data connection

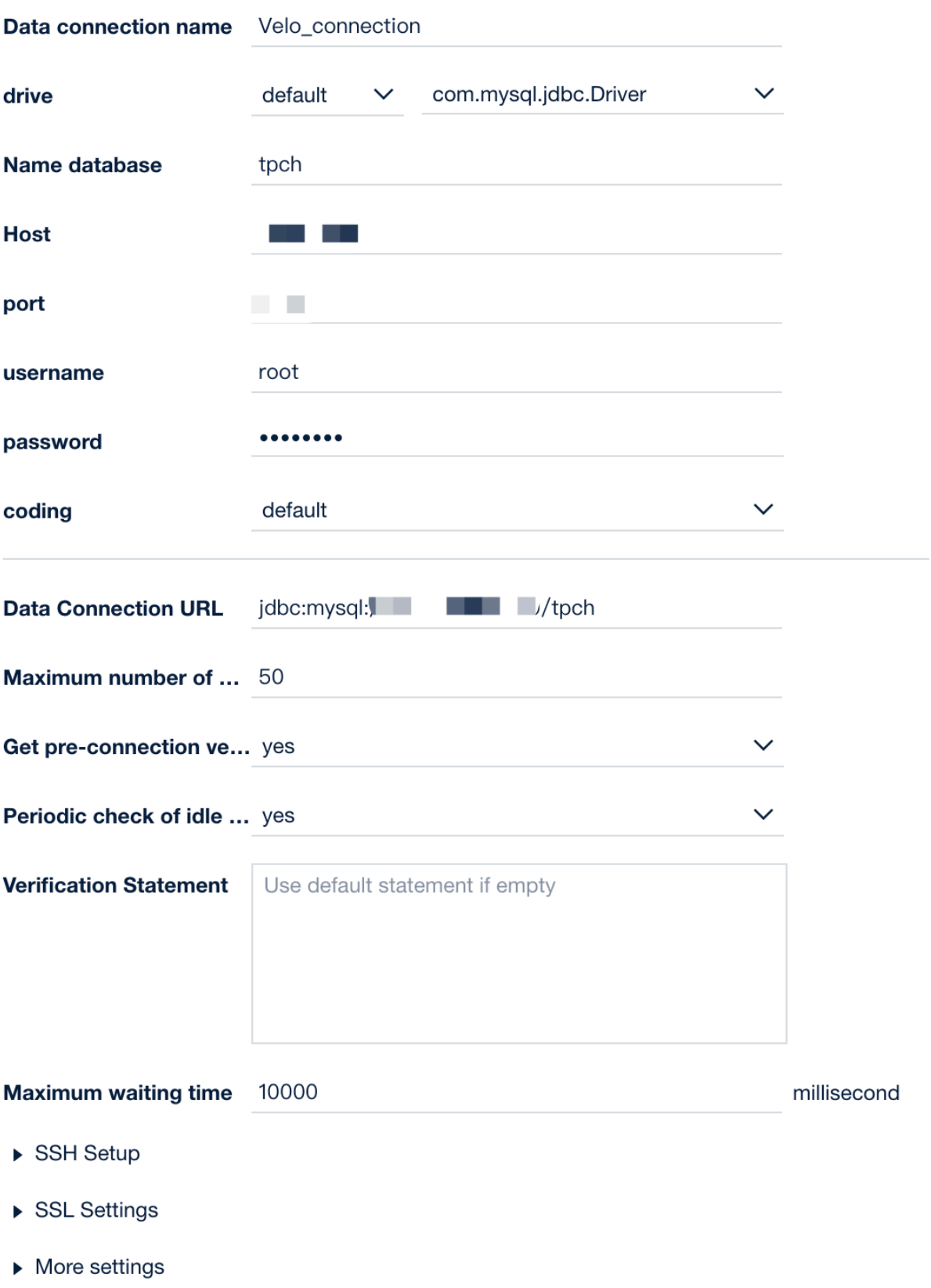
4. On the new database connection page, select MySQL database



select connection

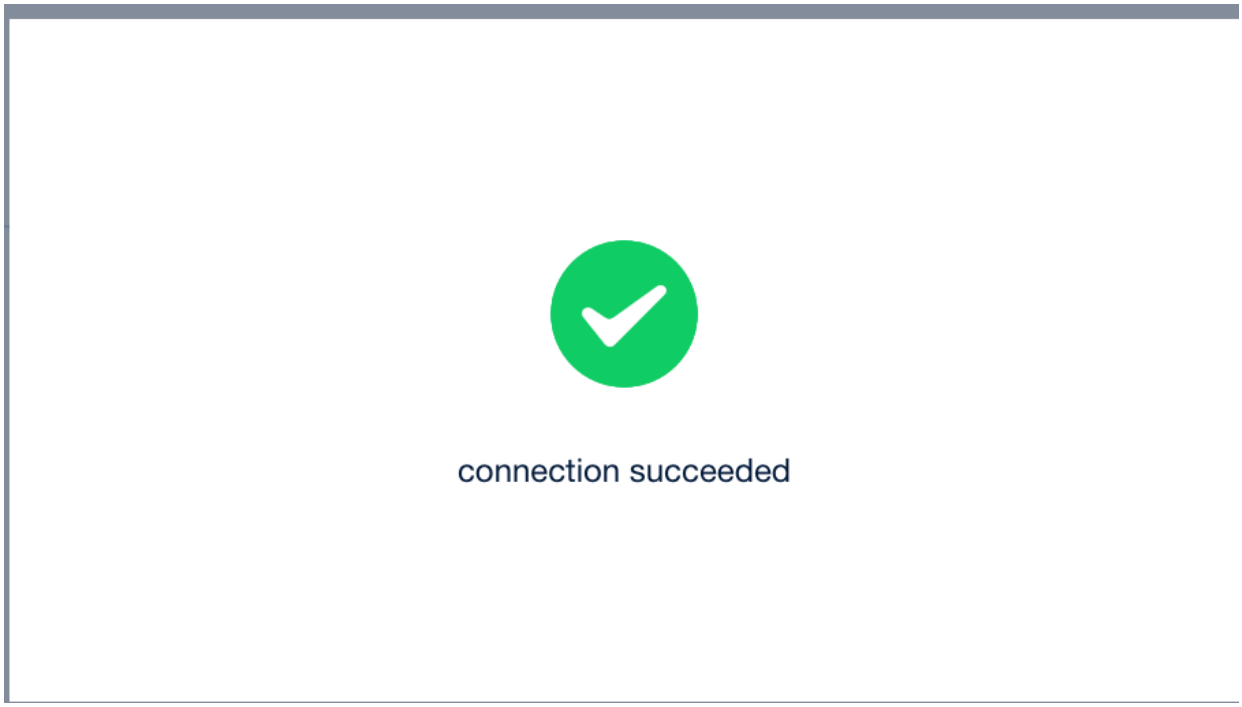
5. Fill in the link information of the Doris database
    - Parameters are described as follows：
        - Username：The username for logging into Doris。
        - Password：Password of the current user。
        - Host：IP address of the FE host in the Doris cluster。
        - Port：FE query port of the Doris cluster。

- Coding：Encoding format of the Doris cluster。
- Name Database：Target database in Doris cluster。



connection information

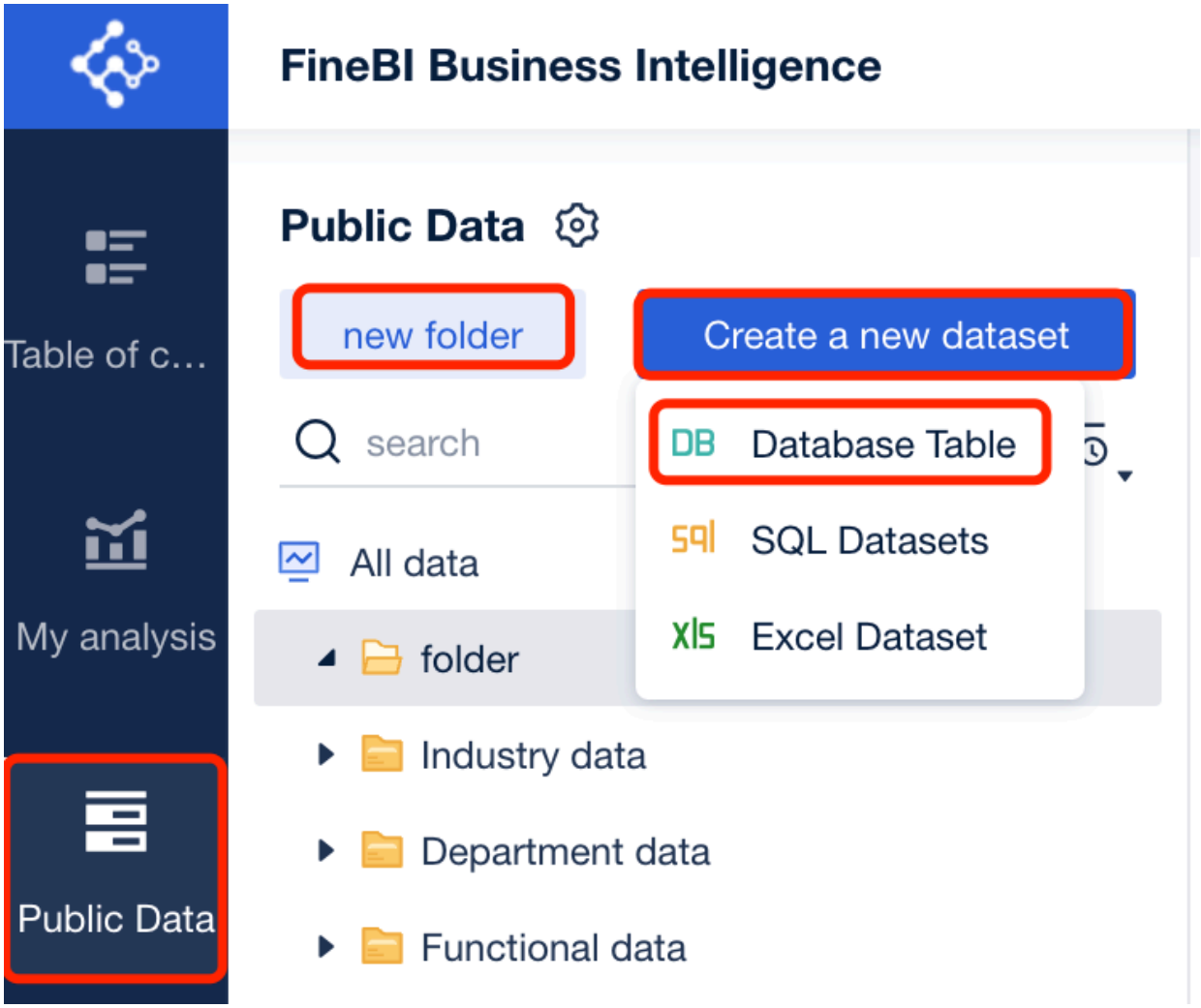
6. Click the test link. Connection succeeded is displayed when the connection information is correct



connection test

Create model

1. In the “Public Data” section, click to create a new dataset. Then click the database table



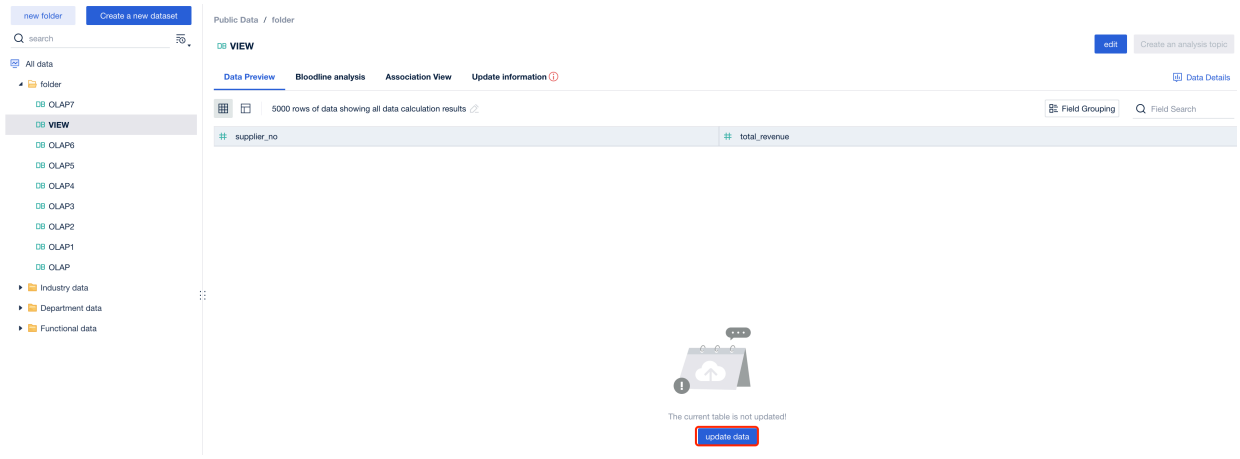
new dataset

2. You need to import tables in the existing database connection



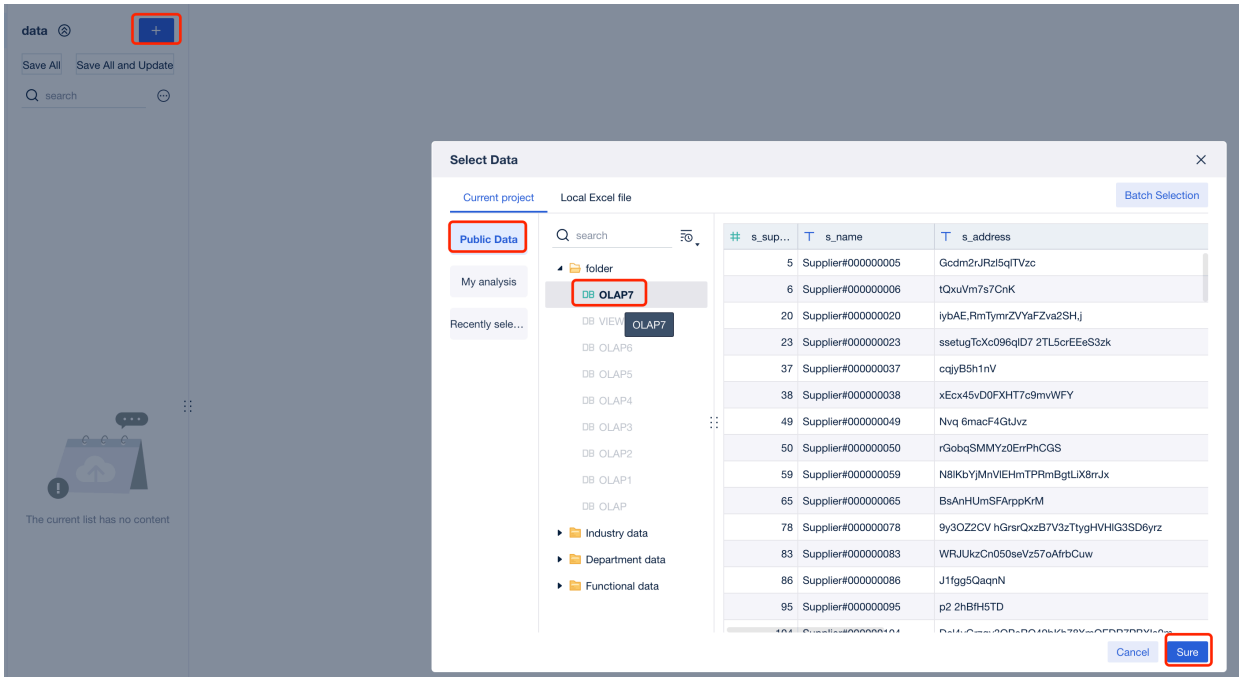
select table

3. You need to refresh each imported table after importing the table. You can analyze the table in the topic only after refreshing the table



refresh table

4. Add the imported public data to the edited topic, and then conduct compass analysis and configuration according to business logic.



data analysis

Power BI

Power BI Introduction

Power BI is a collection of software services and application connectors that can connect to multiple data sources, including Excel, SQL Server, Azure, Google Analytics, etc., so that users can easily consolidate and clean their data. With Power BI's data modeling, users can create relational models, data analysis expressions, and data relationships to support advanced data analysis and visualization. Power BI offers a wealth of visualization options, including ICONS, maps, dashboards, and custom visualization tools to help users make a more intuitive sense of data.

Apache Doris is highly compatible with MySQL protocol and can be connected to Power BI through MySQL Driver. At present, internal data modeling, data query and visualization processing of Apache Doris have been officially supported in Power BI.

Precondition

If you do not have Power BI desktop installed, you can download it from https://www.microsoft.com/en-us/power-platform/products/power-bi/desktop

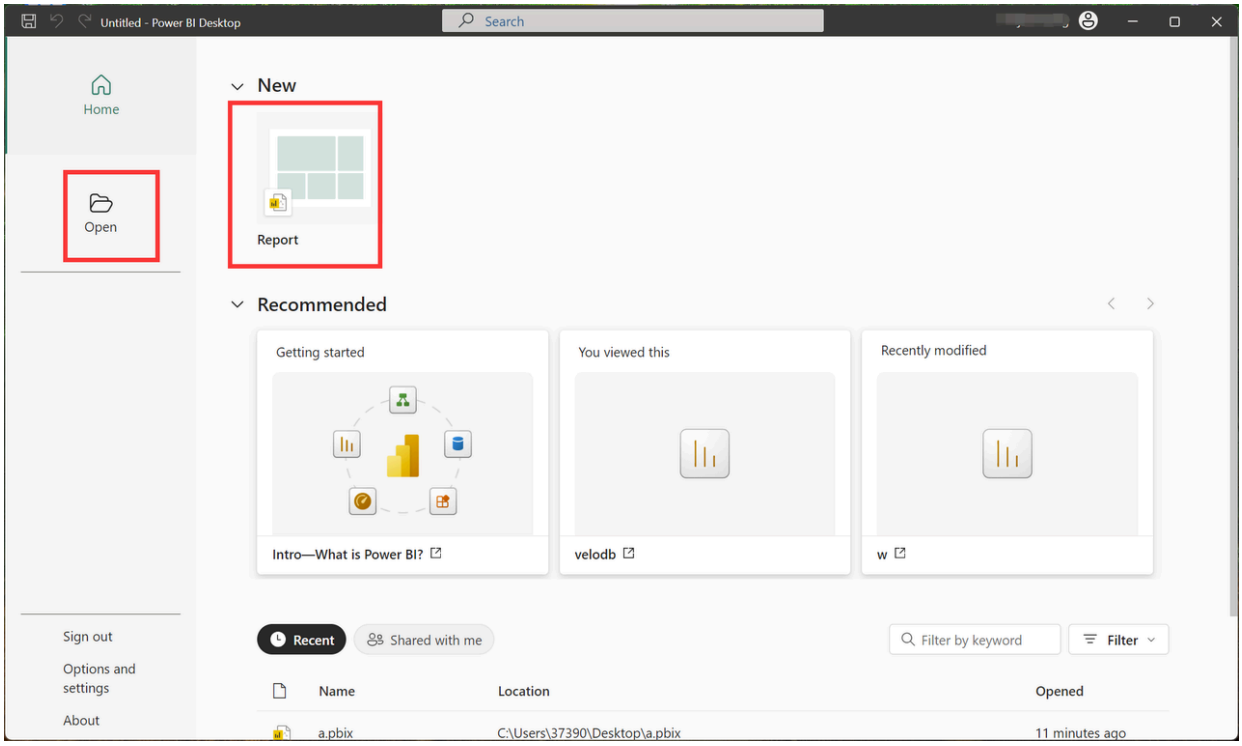
Connector configuration of Power BI and Doris

> **Note**
>
> Currently verified using MySQL JDBC Connector version 8.0.26

Download and installation MySQL Connector Download link: https://downloads.mysql.com/archives/c-net/. Select version 8.0.26. There are incompatibilities in higher versions

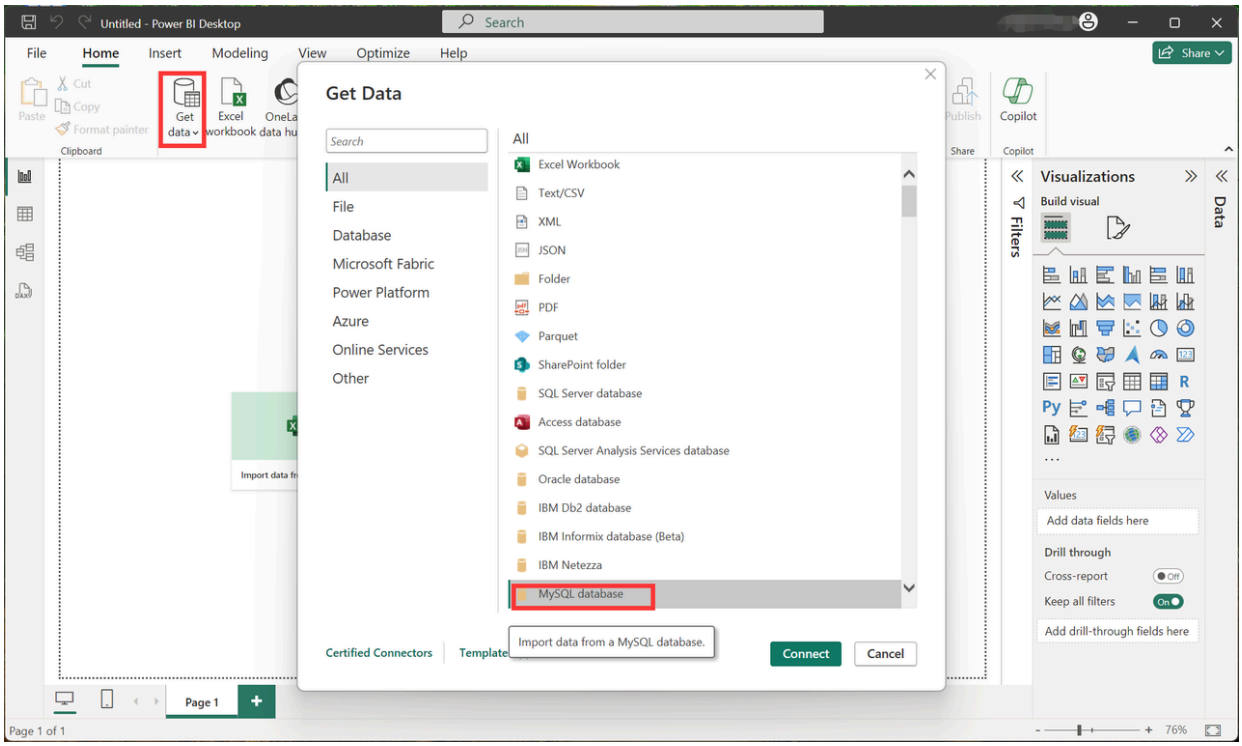
Load data locally and create models

1. Start the Power BI Desktop
2. Open the Power BI Desktop screen and click Create Report. If a local report exists, you can open it.



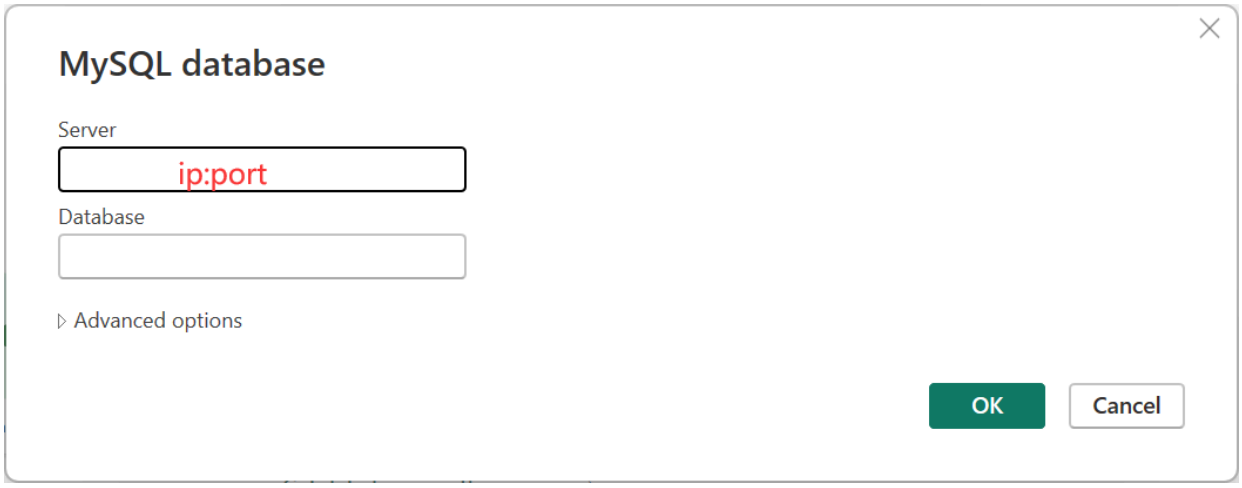
start page

3. Click get data. In the dialog box that is displayed, select MySQL database.



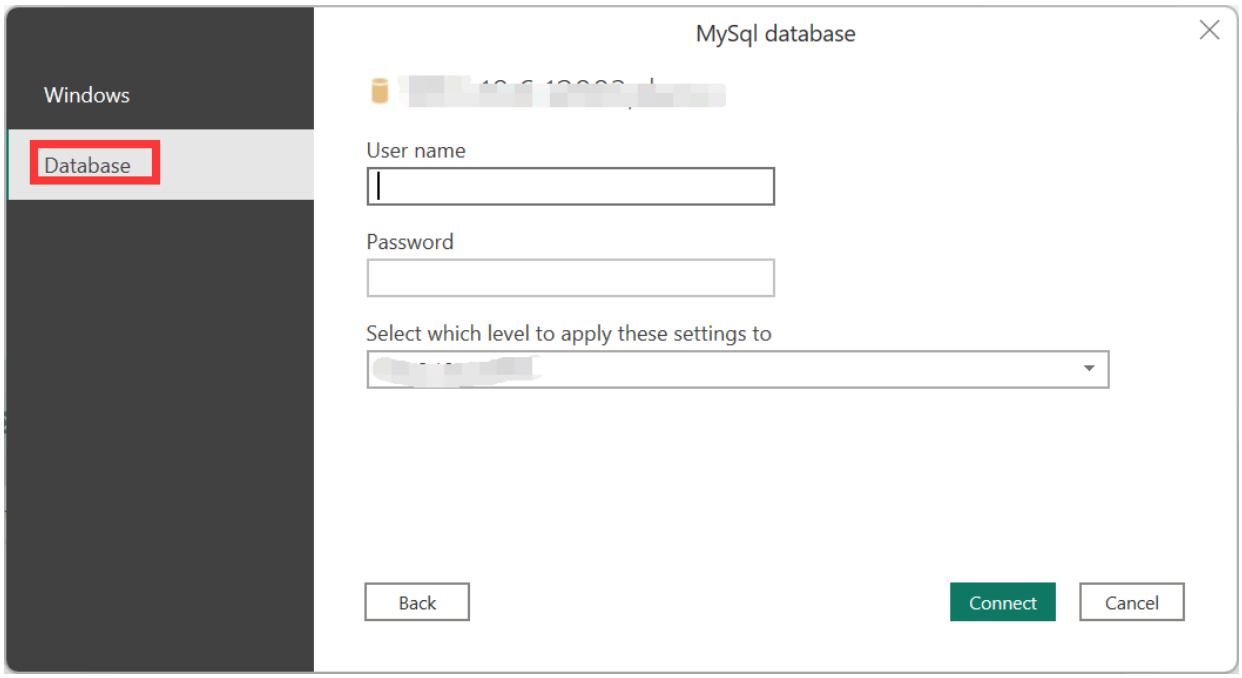
get data

4. Configure the database connection information and enter ip:port in the server text box. The default port number for Doris is 9030.



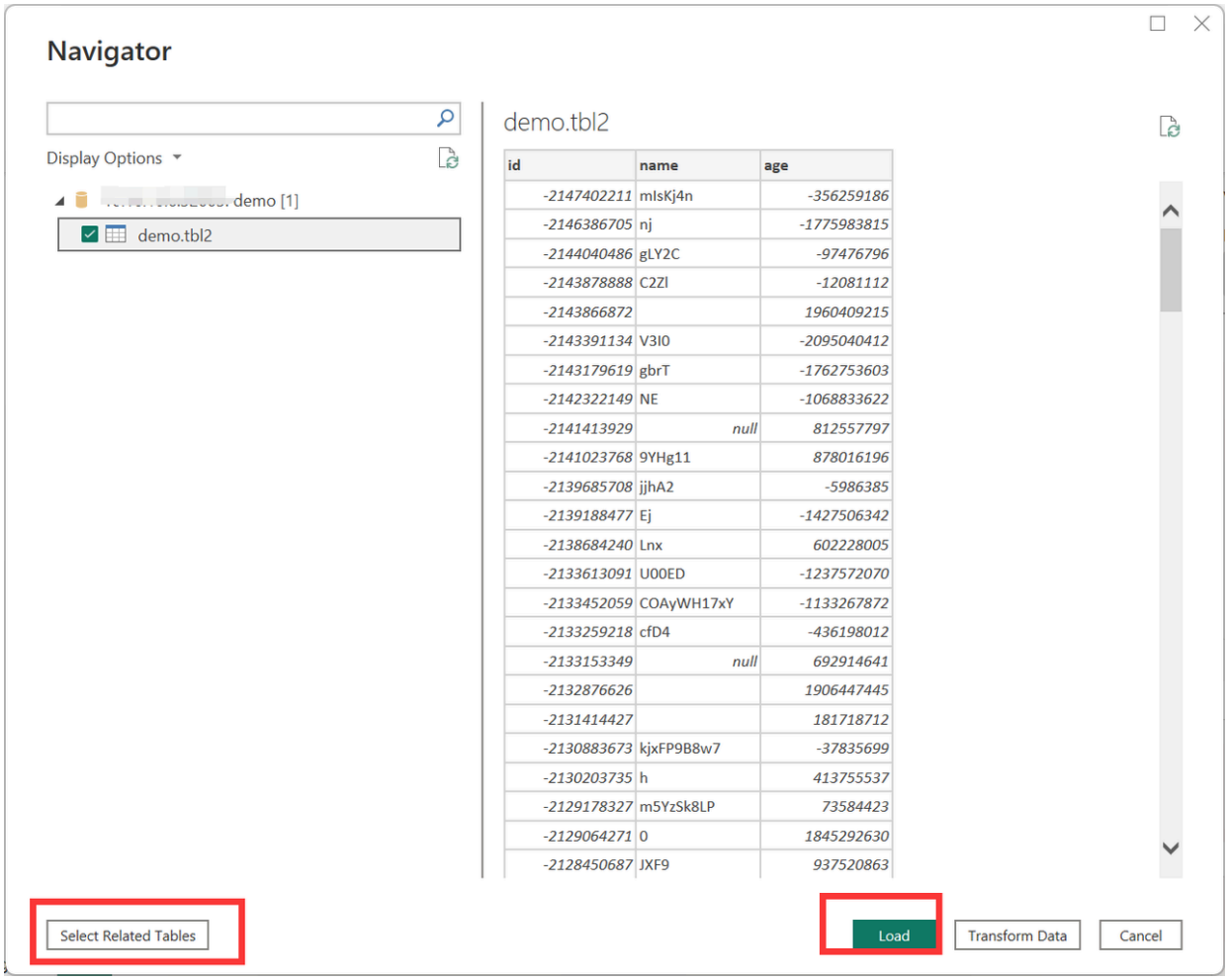
connection information

5. Click OK in the previous step and select “Database” in the new connection window to connect, and fill in the connection information of Doris in the username and password.



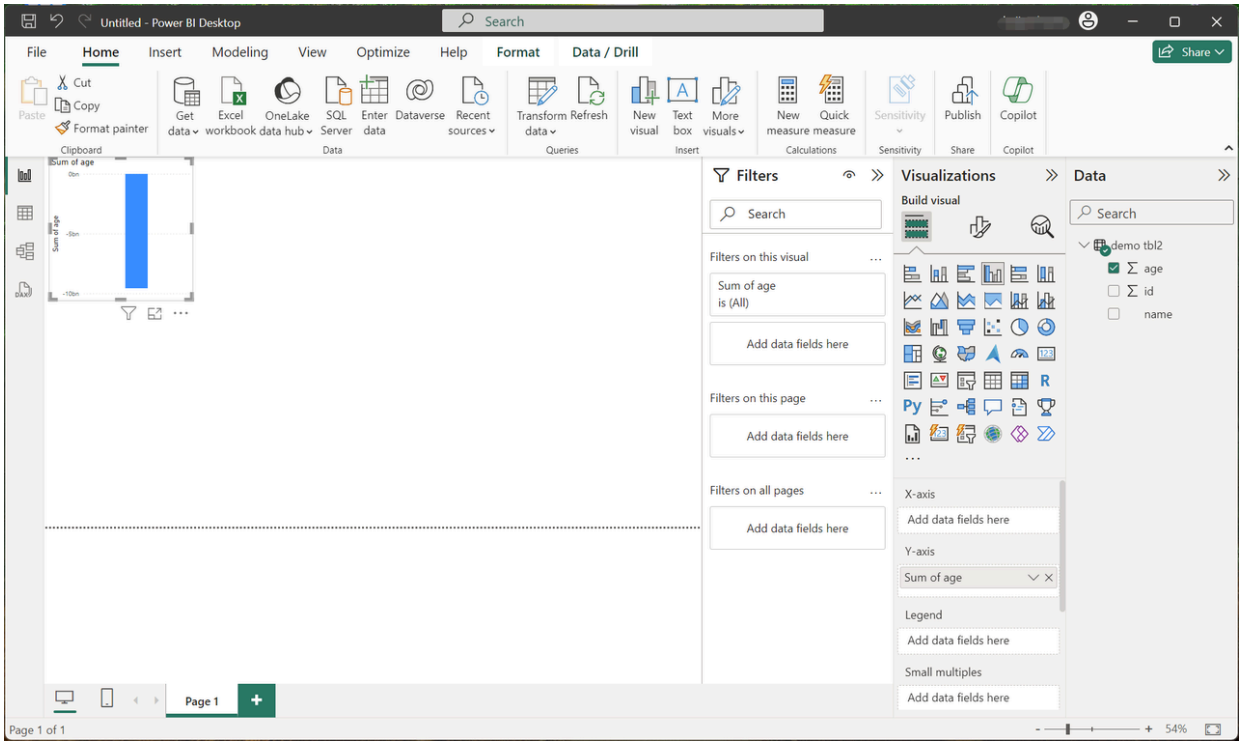
uname and pwd

6. Load the selected table to the Power BI Desktop



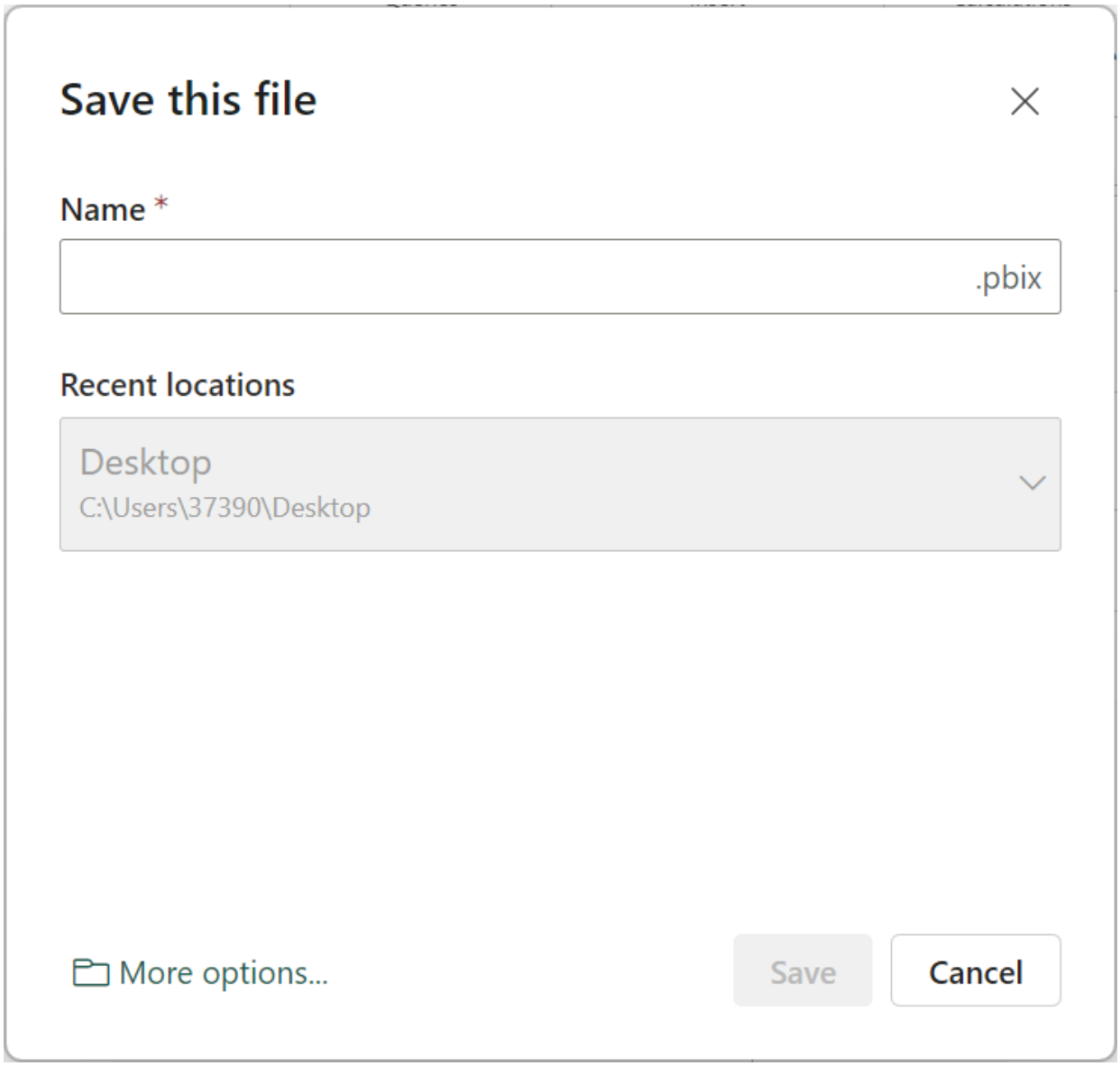
load data

7. Configure statistical compass



create compass

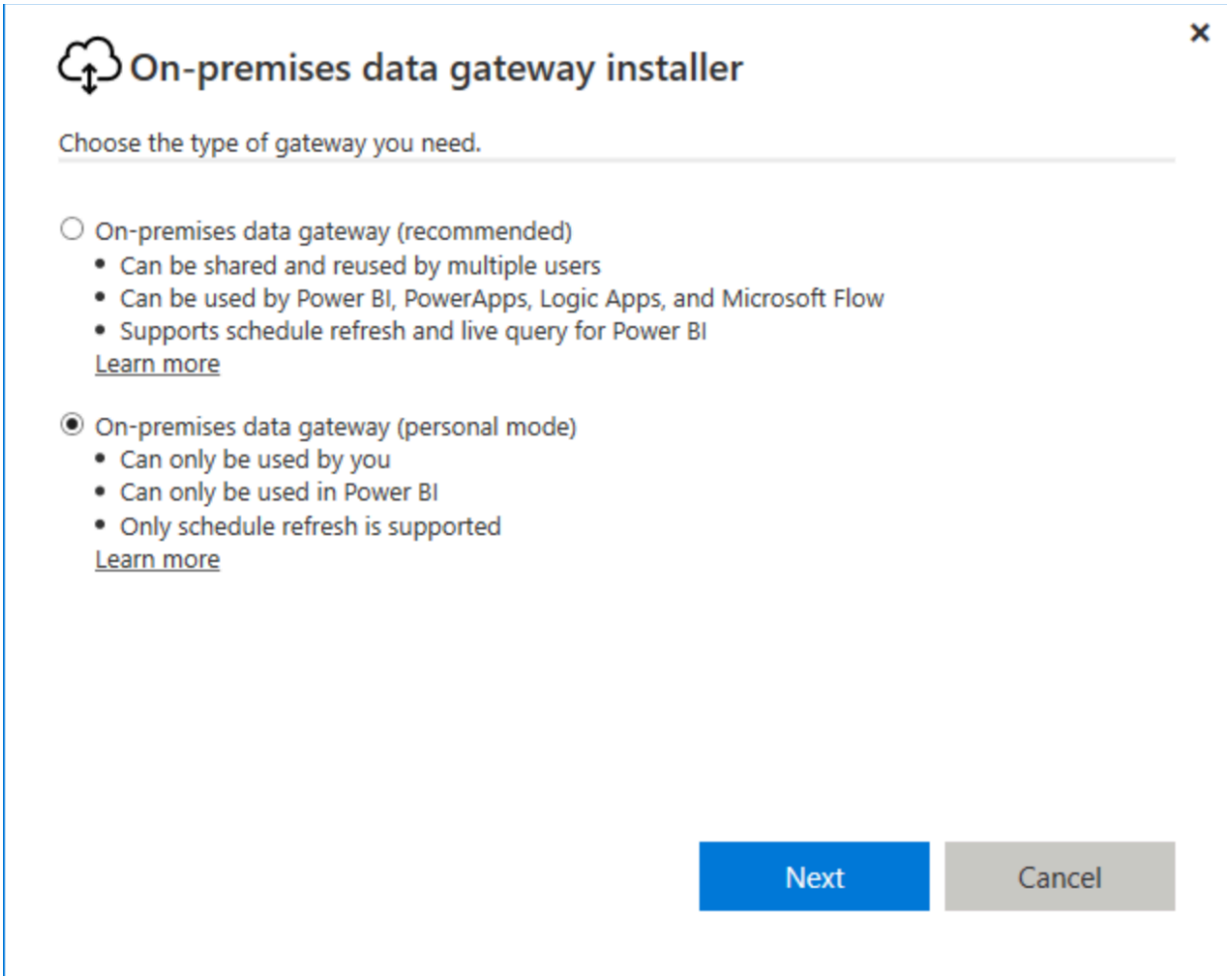
8. Save statistical compass to location



save file

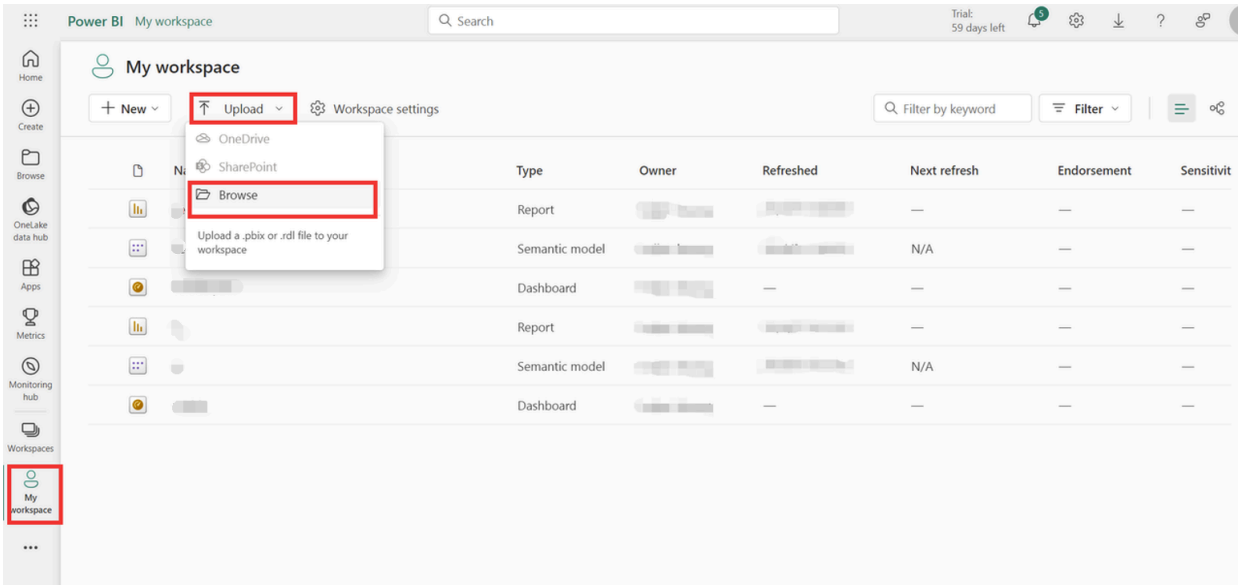
Set data refresh automatic

1. Download the On-premises data gateway. Download address: https://learn.microsoft.com/en-us/power-bi/connect-data/service-gateway-personal-mode
2. Install the On-premises data gateway



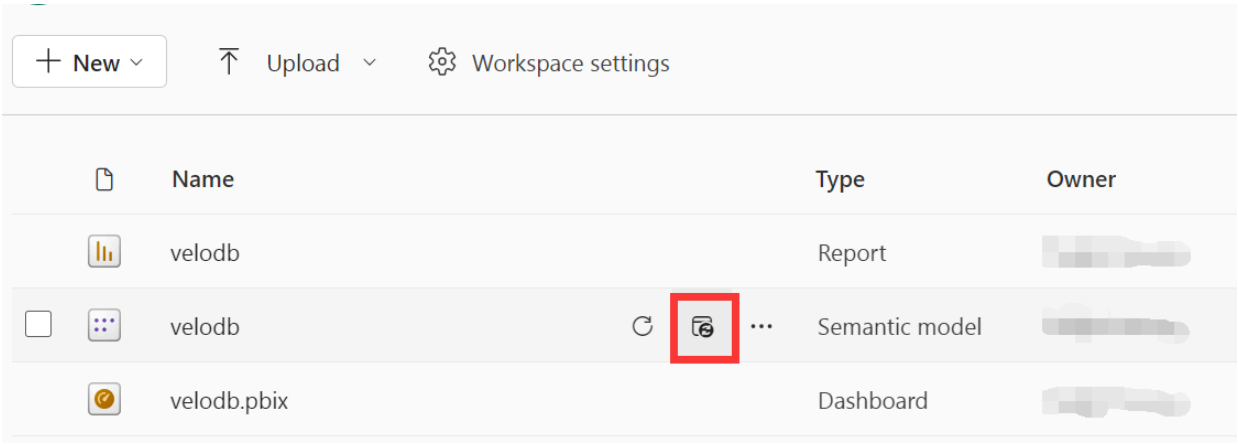
gateway install

3. Log into Power BI Online and import the local model in your workspace



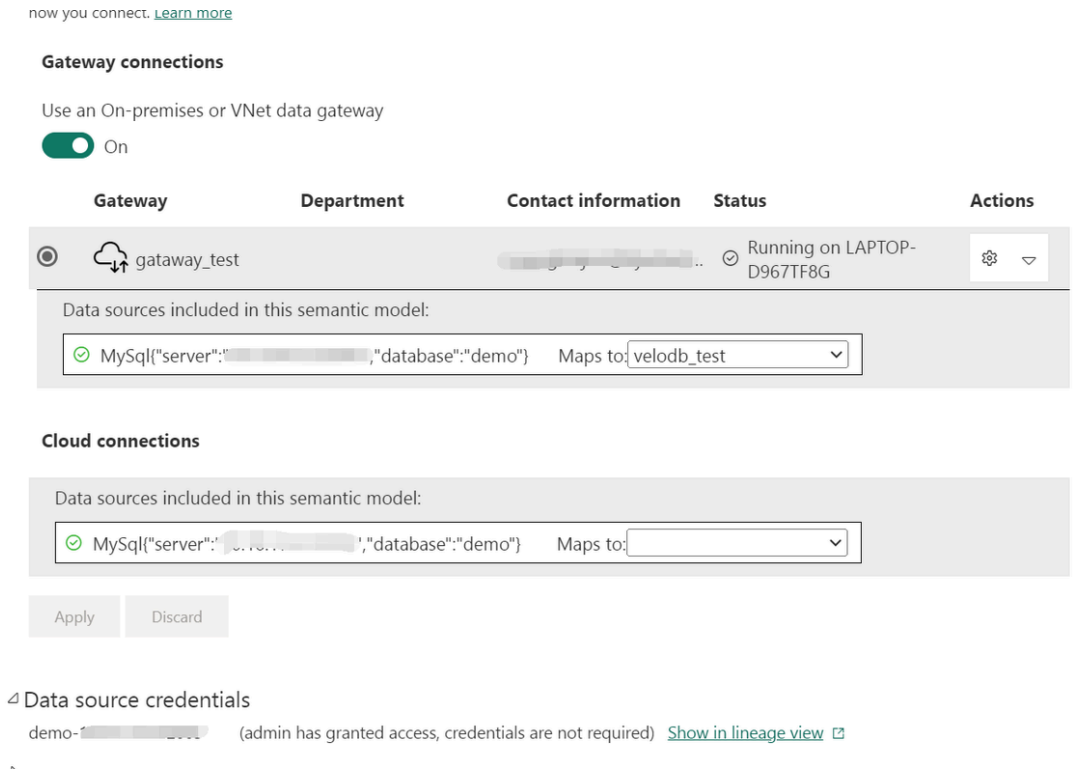
upload

4. Click the model to set the automatic refresh time



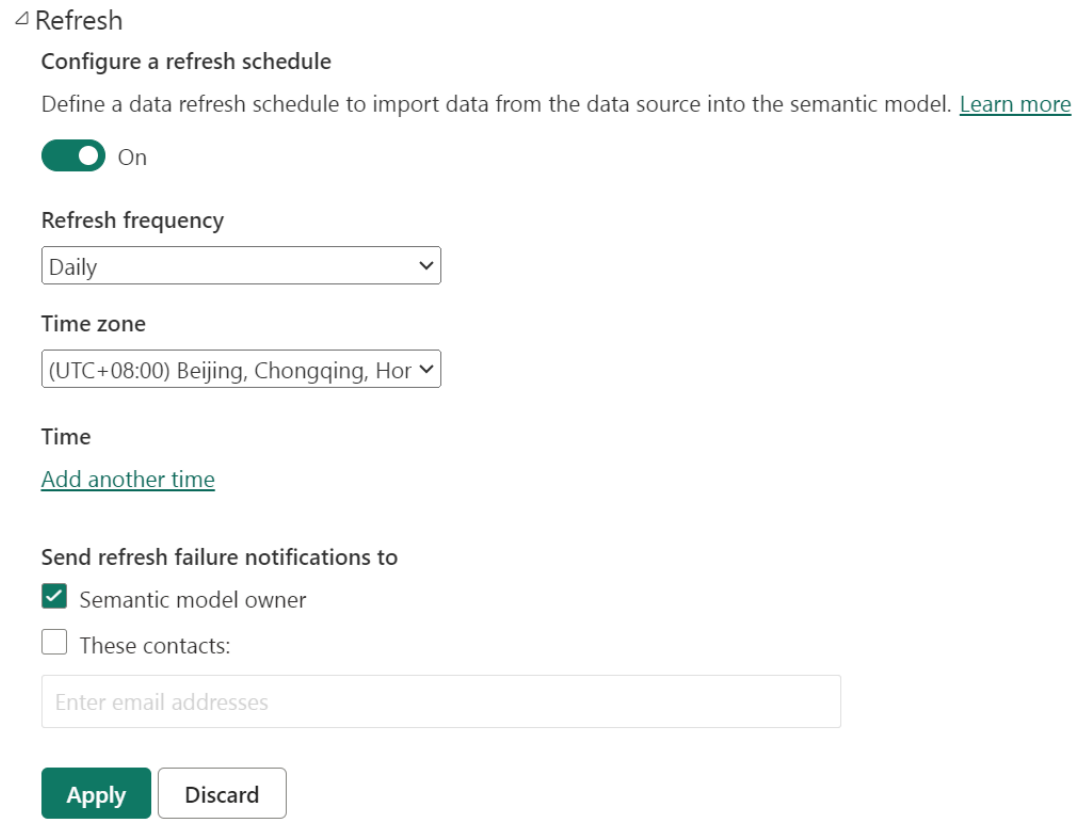
click module

5. The data refresh configuration requires a gataway connection. After the gateway is enabled locally, you can see the started gateway in the gateway connection locally. Select the local gateway.



config gateway

6. Configure the refresh schedule to complete the automatic data refresh on Power BI


Refresh
Configure a refresh schedule
Define a data refresh schedule to import data from the data source into the semantic model. Learn more
On
Refresh frequency
Daily
Time zone
(UTC+08:00) Beijing, Chongqing, Hor
Time
Add another time
Send refresh failure notifications to
Semantic model owner
These contacts:
Enter email addresses
Apply
Discard

make plan

Tableau

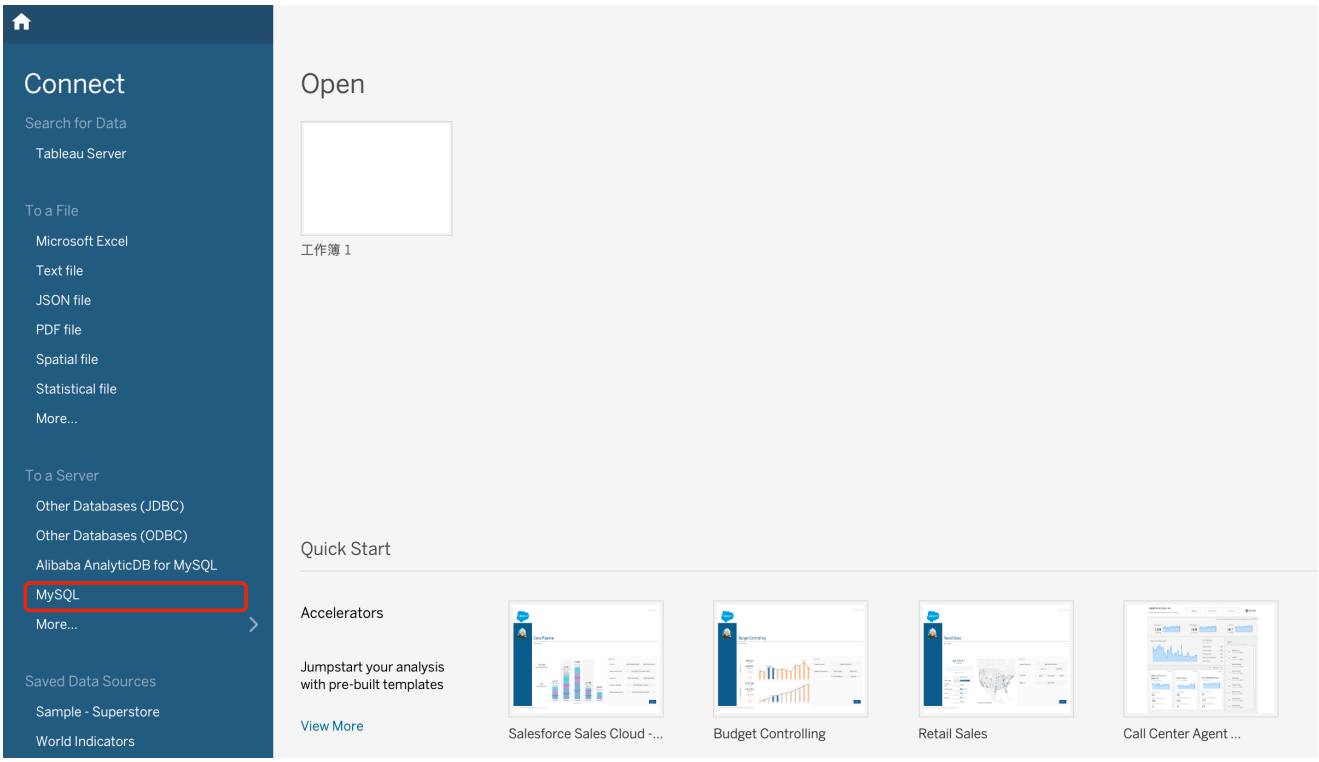
Introduction

Tableau is a lightweight data visualization analysis platform that combines data operations with beautiful charts perfectly. It seamlessly combines data computation with visually appealing charts, requiring no coding from the user. By simply dragging and dropping, users can quickly gain insights into the data. They can explore different views and even easily combine multiple data sources to complete tasks such as data visualization, exploration, and analysis. ### Prerequisites Regardless of whether your Mac is based on an Intel chips or Apple Silicon Chips, for Tableau Desktop on Mac, you need to download and install the Intel version. This is to ensure compatibility with the MySQL driver. You can select and download the Intel version from the Support Releases page.

Driver Installation

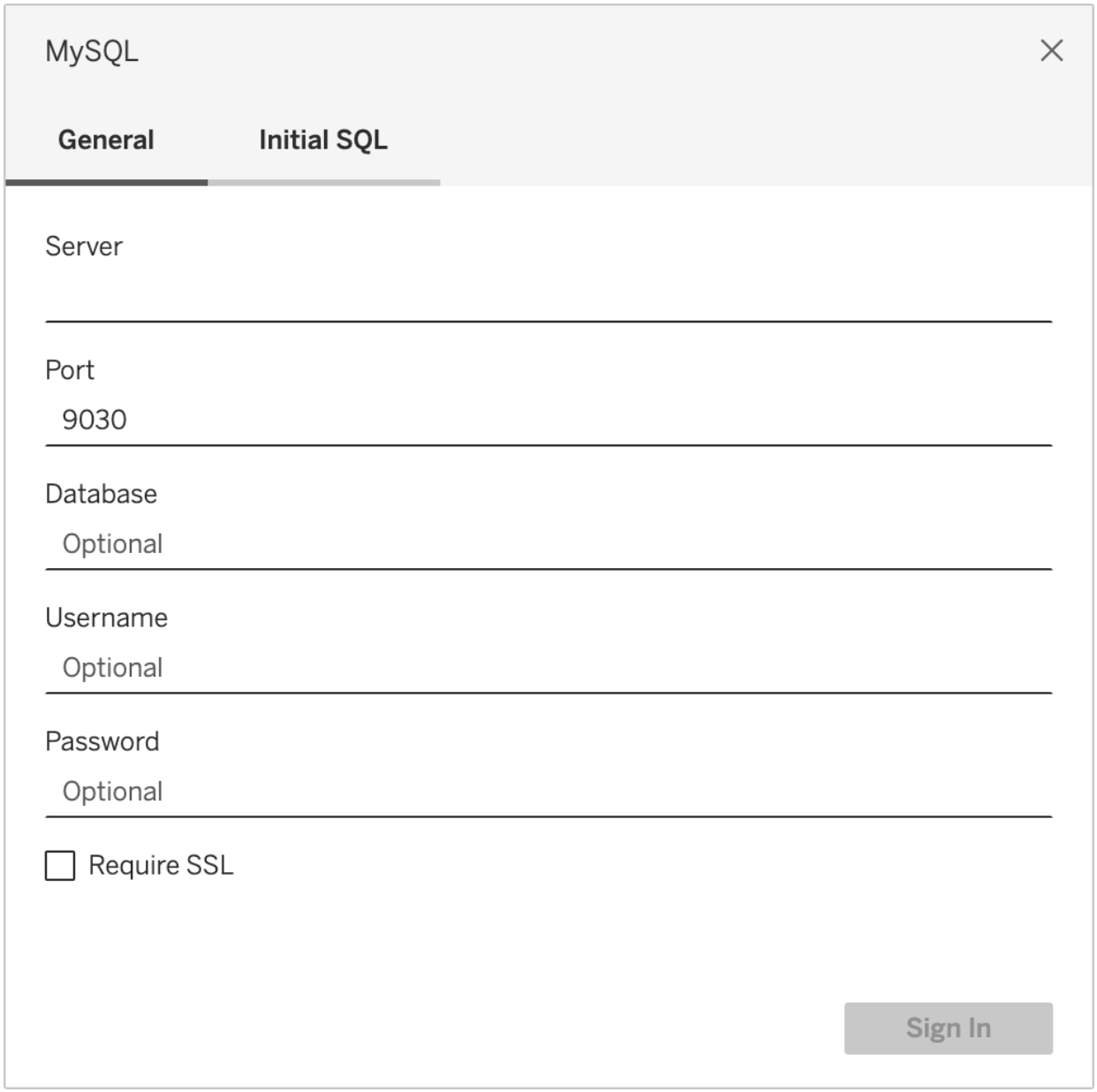
1. iODBC Installation
    1. Close Tableau Desktop
    2. Download the latest Driver Manager (mxkozzz.dmg) from the official iODBC website
    3. Install the downloaded dmg file
2. Install the MySQL driver

When choosing the ODBC driver for MySQL to connect to Doris, you should install the MySQL 5.x ODBC driver. Using the latest MySQL driver may result in an “Unsupported command” error when connecting to Doris. ### Connection Configuration and Usage 1. Click the Tableau Desktop home page and select MySQL at the connection data source



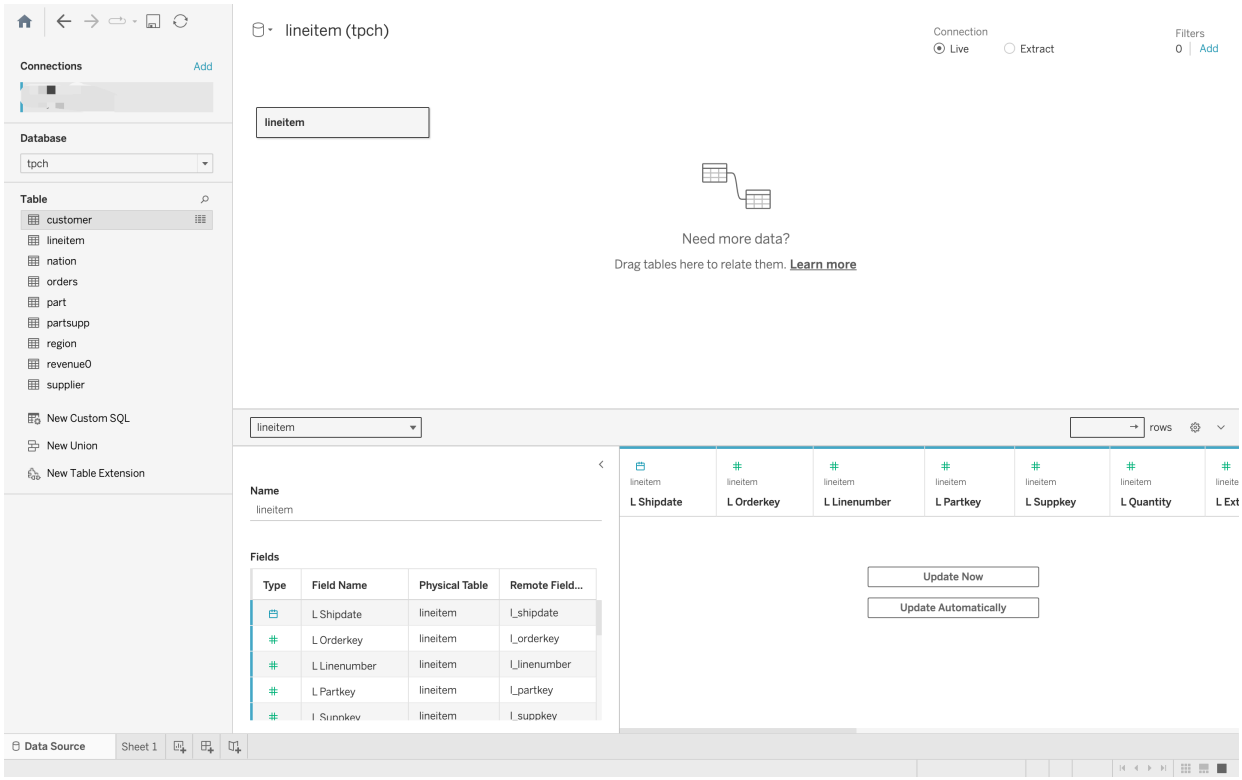
main page

2. Fill in the Doris server address, port and other relevant information, and click sigin in button after correctly filling



sign in page

3. After entering Tableau, select the corresponding library table to carry out the relevant compass processing.


lineitem (tpch)
Connection
Live
Extract
Filters
0 | Add
Connections
Add
Database
tpch
Table
customer
lineitem
nation
orders
part
partsupp
region
revenue0
supplier
New Custom SQL
New Union
New Table Extension
lineitem
Need more data?
Drag tables here to relate them. Learn more
lineitem
rows
Name
lineitem
Fields
Type
Field Name
Physical Table
Remote Field...
L Shipdate
lineitem
l_shipdate
L Orderkey
lineitem
l_orderkey
L Linenumber
lineitem
l_linenumber
L Partkey
lineitem
l_partkey
L Suppkey
lineitem
l_suppkey
L Shipdate
L Orderkey
L Linenumber
L Partkey
L Suppkey
L Quantity
Update Now
Update Automatically
Data Source
Sheet 1

usage page

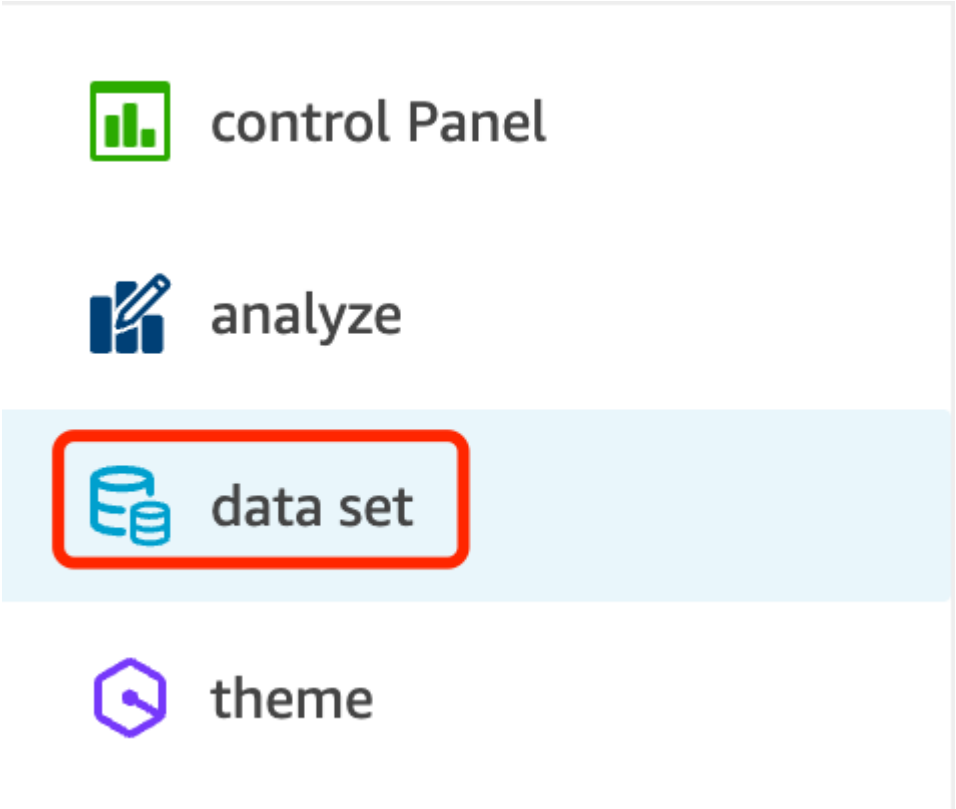
QuickSight

Introduction

QuickSight is a robust data visualization and analysis platform that seamlessly integrates data computation with visually appealing charts. It does not necessitate programming skills from users, as data insights can be quickly obtained through simple drag-and-drop operations. QuickSight offers a diverse range of viewing options for exploring data from various perspectives. Furthermore, it facilitates easy integration of multiple data sources, simplifying and streamlining the process of presenting, exploring, and analyzing data. This platform not only accelerates data processing but also enhances the intuitiveness and comprehensibility of the analysis process, significantly improving user work efficiency and analytical capabilities.

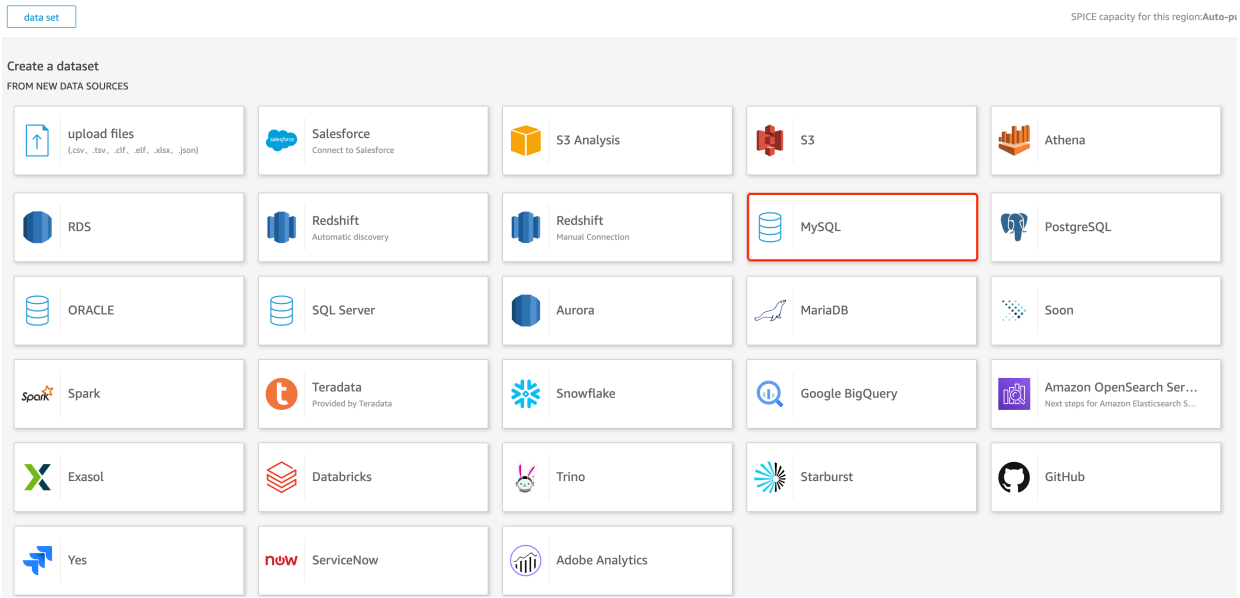
Data configuration

1. Upon completing the registration process, please proceed to sign in to QuickSight and choose the desired dataset.



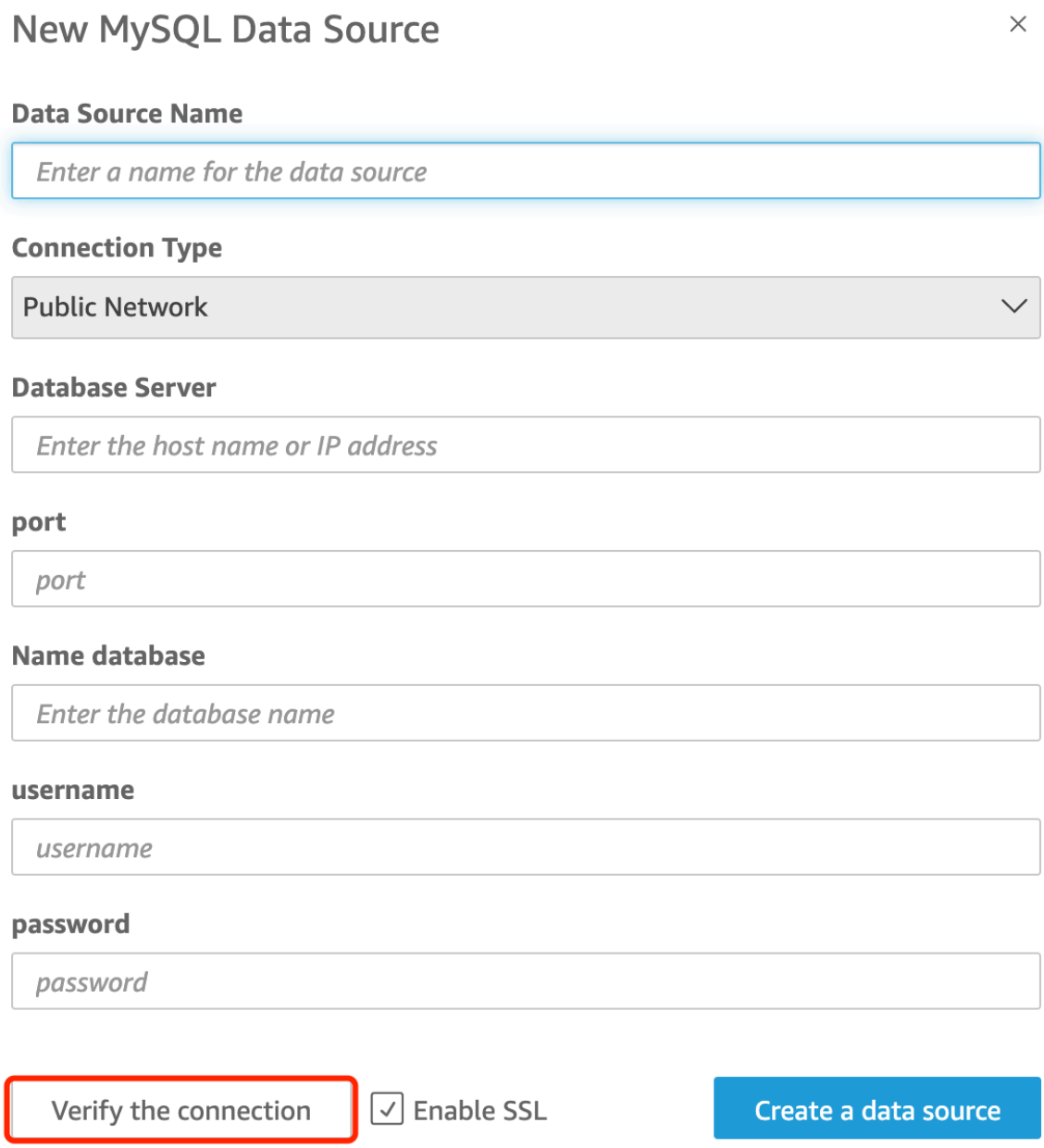
dataset select

2. Select MySQL on the dataset selection page.



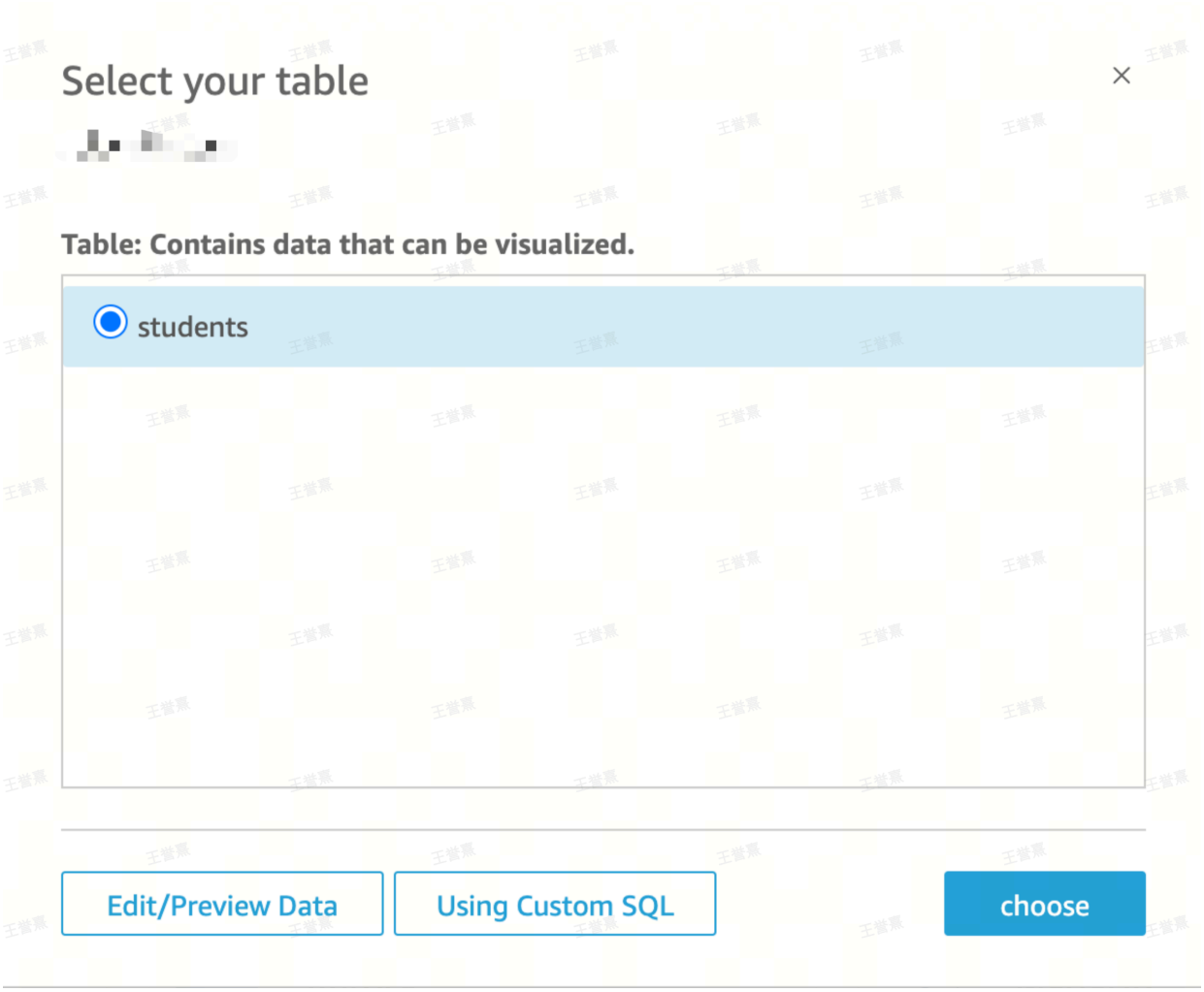
database select

3. On the data connection page, fill in the corresponding IP, port, database name, username, and password for Doris. After filling in the information, click “Verify the Connection”. If the test is successful, create a data source.



connection information

4. Select the table for which you want to create the dashboard.



jdbc connector download

5. QuickSight offers two methods for importing tables. You can choose the appropriate import method based on your needs.

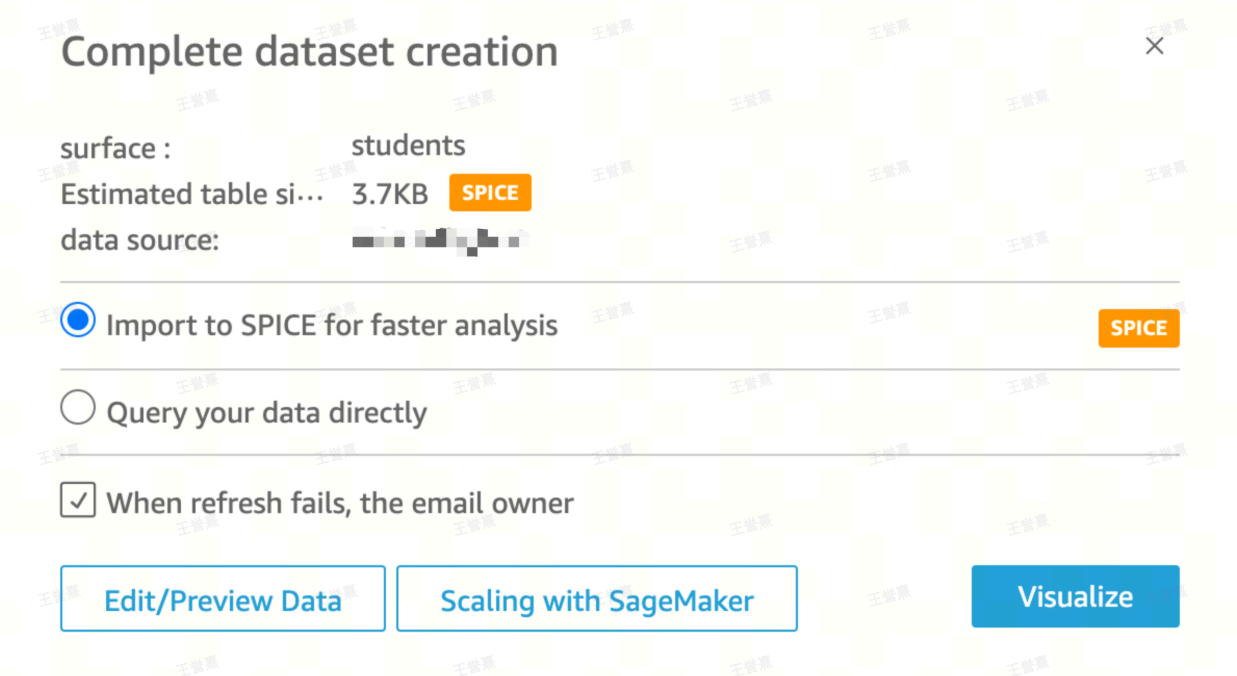


table select

6. After importing the table, you can create the desired dashboard.

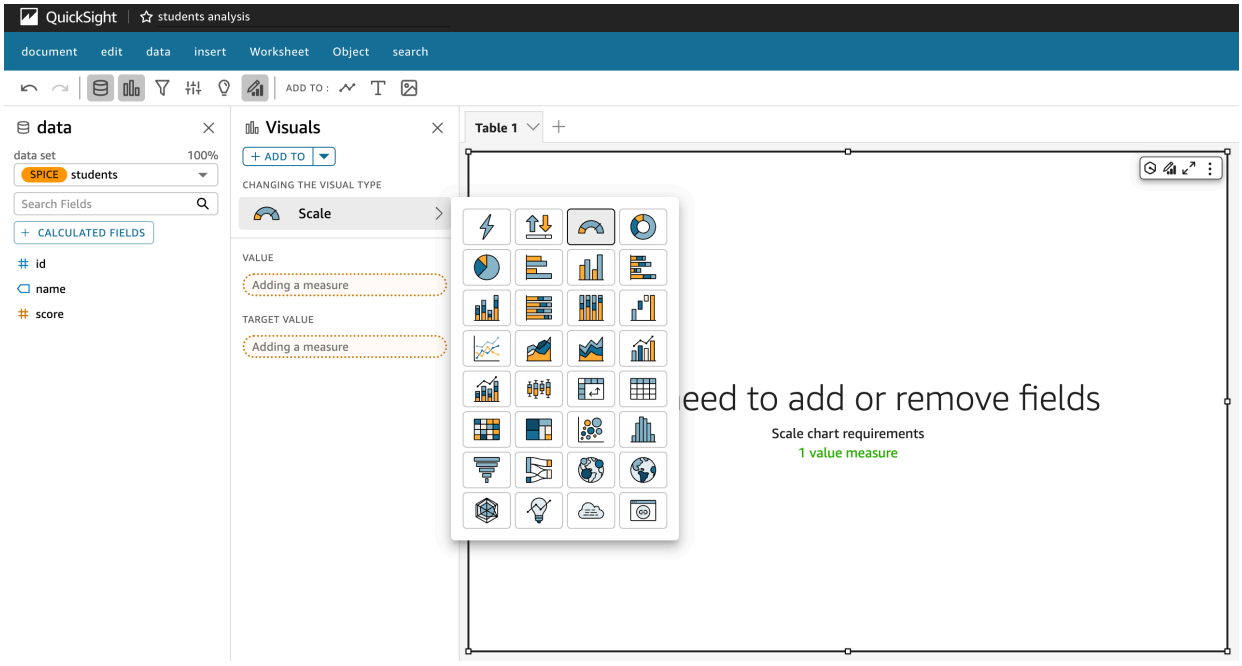


table import

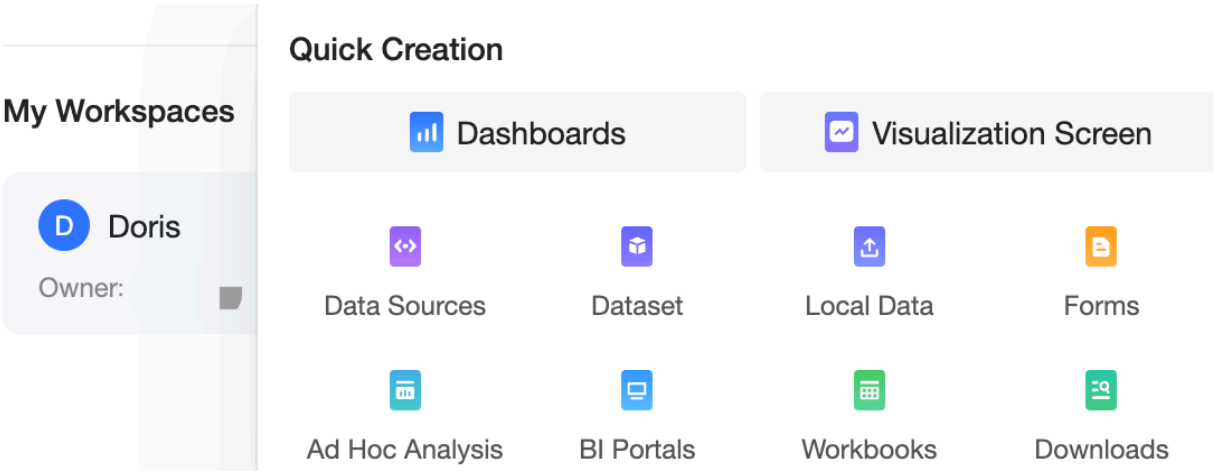
Quick BI

Introduction

Quick BI is a data warehouse-based business intelligence tool that helps enterprises set up impressive visual analyses quickly. Quick BI supports a variety of data sources, including databases like MySQL, Oracle, SQL Server, and Apache Doris, as well as file formats such as Excel, CSV, and JSON. It offers a wealth of visualization components, such as tables, charts, and maps, allowing users to easily achieve data visualization through simple drag-and-drop operations.

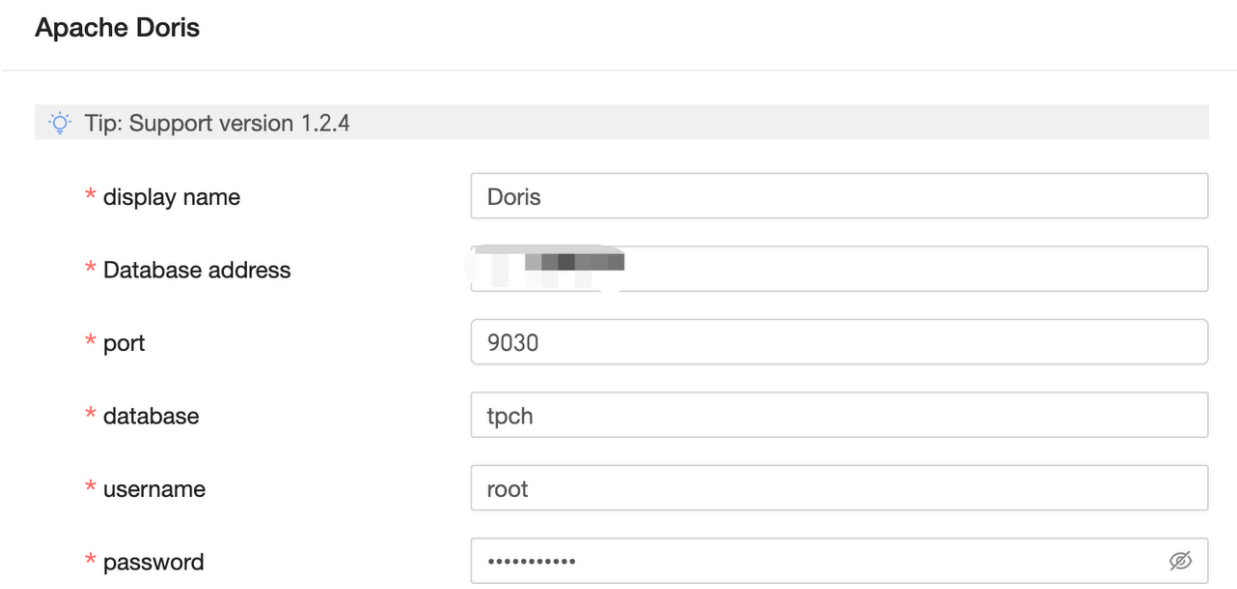
Data connection and application

1. Login Quick BI and create a workspace.
2. Click Data Source under the current workspace.



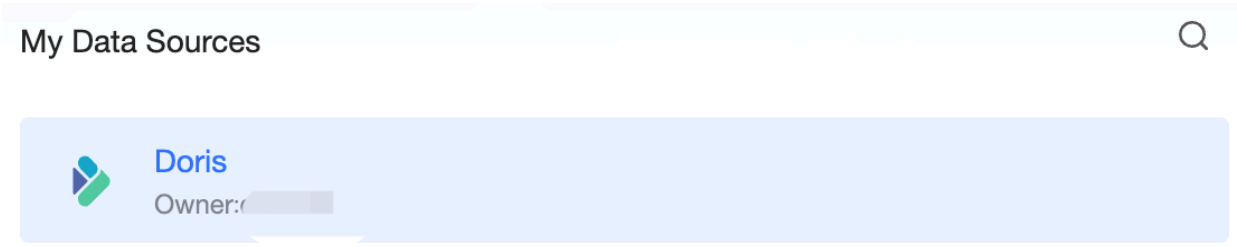
create workspace

3. Select Apache Doris in the already created data source and fill in the corresponding Apache Doris connection information.



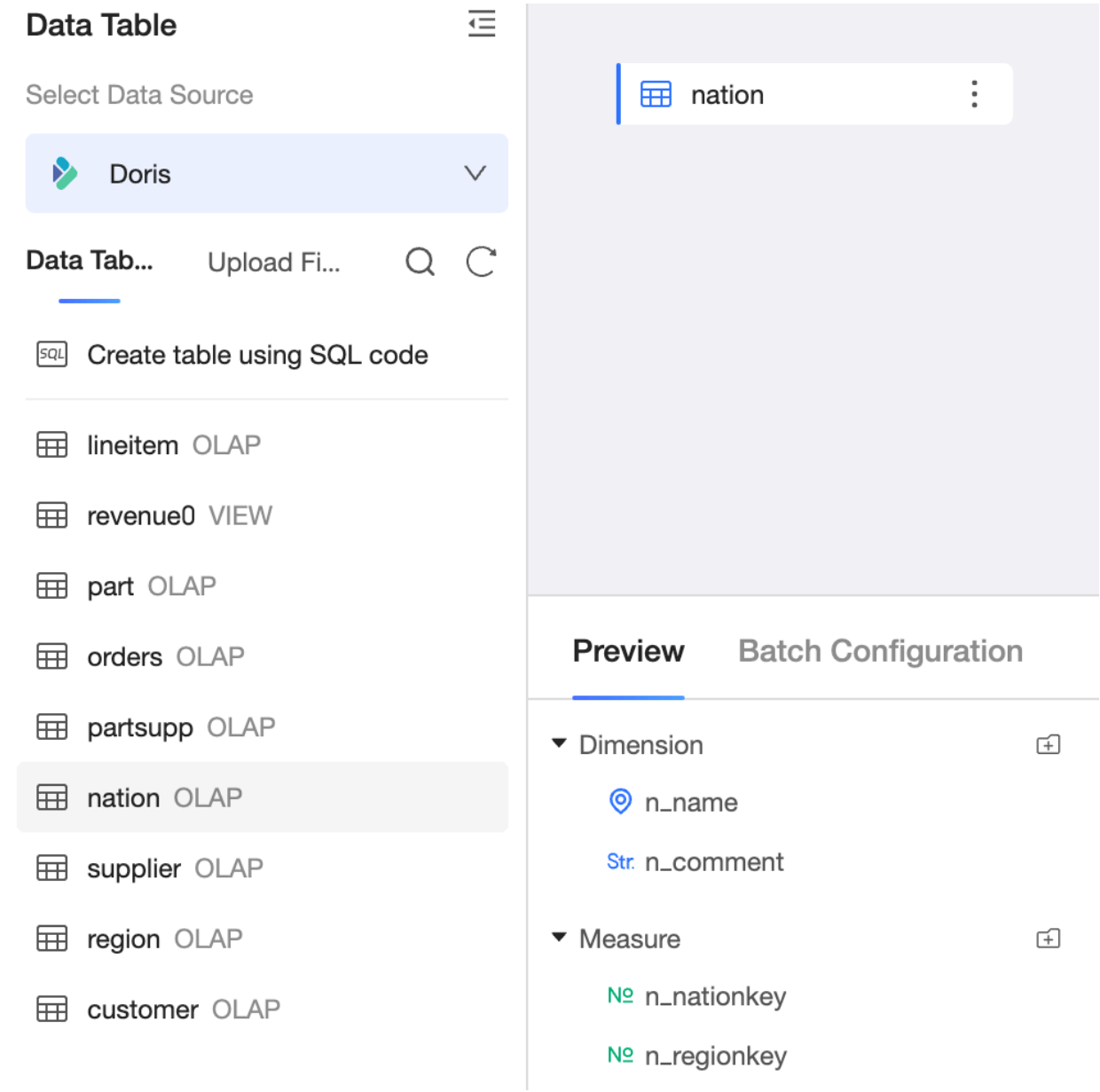
Doris information

4. Once the connection is successful, you can see the data source we created in the data source list.



data source

5. Create a dataset in the data source we created, using the TPC-H dataset as an example. After the dataset is created, you can set the corresponding report.


Data Table
Select Data Source
Doris
Data Tab...
Upload Fi...
Create table using SQL code
lineitem OLAP
revenue0 VIEW
part OLAP
orders OLAP
partsupp OLAP
nation OLAP
supplier OLAP
region OLAP
customer OLAP
nation
Preview
Batch Configuration
Dimension
n_name
n_comment
Measure
n_nationkey
n_regionkey

Doris table

Smartbi

Smartbi Introduction

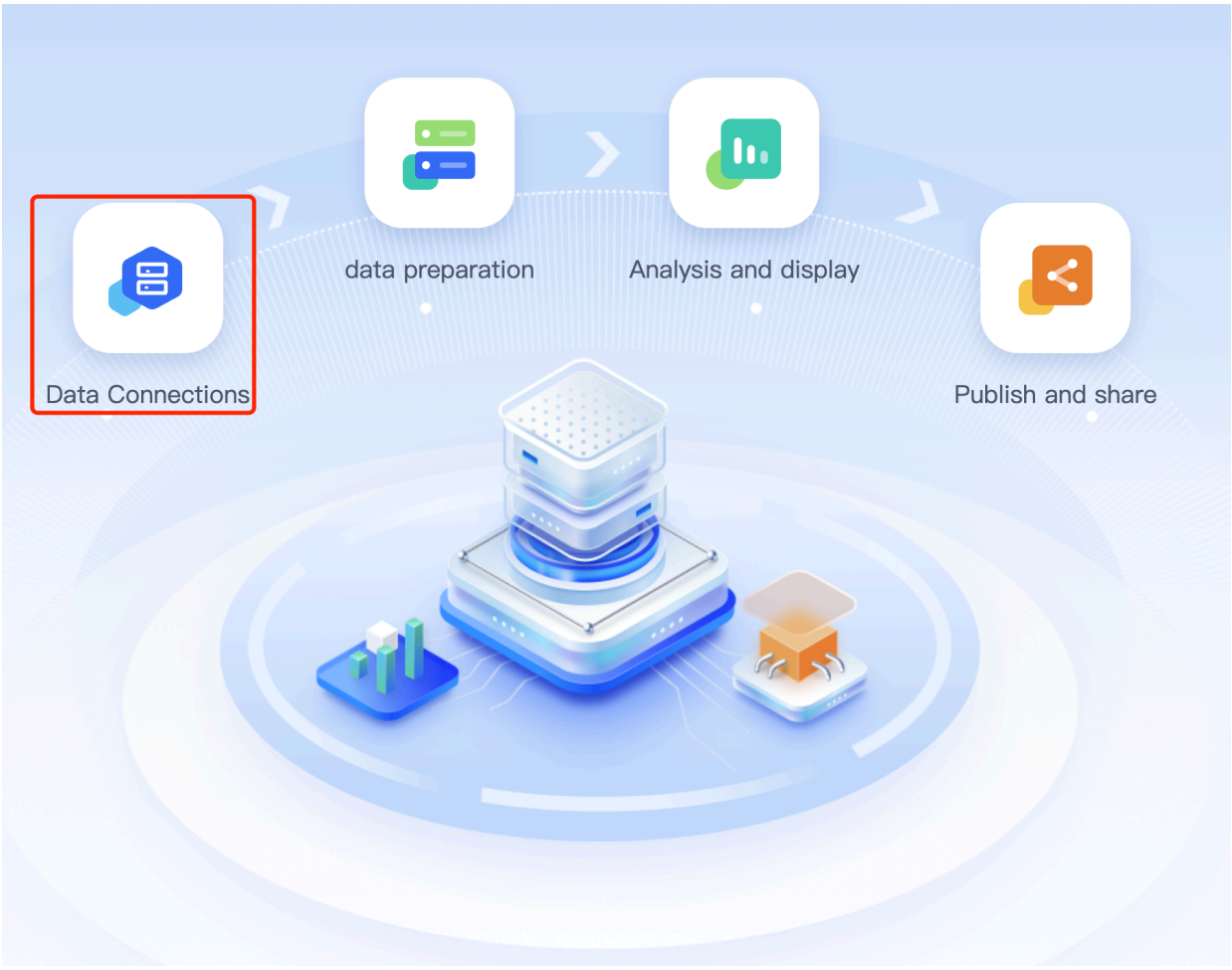
Smartbi is a collection of software services and application connectors that can connect to a variety of data sources, including Oracle, SQL Server, MySQL, and Doris, enabling users to integrate and cleanse their data easily. With Smartbi's data modeling capabilities, users can not only create complex relational models but also write data analysis expressions and establish data relationships, laying the foundation for advanced data analysis and visualization. SmartBI offers a diverse range of visualization options, including various types of charts, geographic maps, interactive dashboards, and customizable visualization tools. These powerful tools help users understand and present their data more intuitively and comprehensively, enhancing the effectiveness and efficiency of data analysis.

Precondition

you can visit https://www.smartbi.info/download to download and install Smartbi.

Data connection and application

1. Log into Smartbi and click Data Connections.



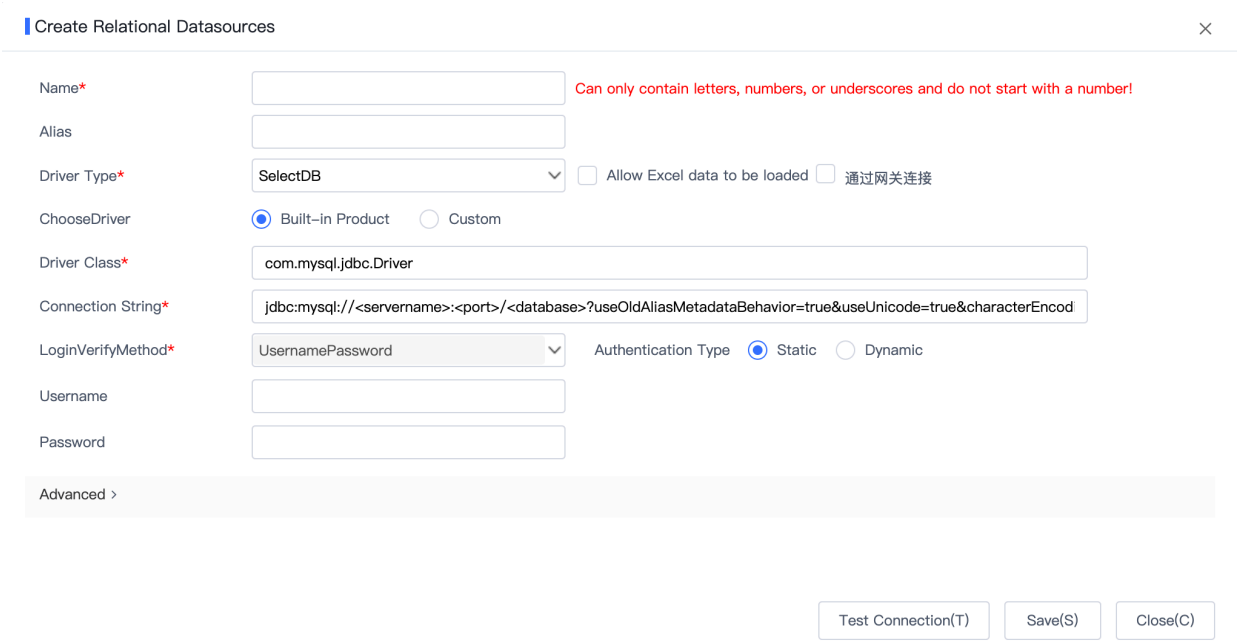
main page

2. After clicking the data connections, you can find a list of databases to be connected with and select the Doris database.



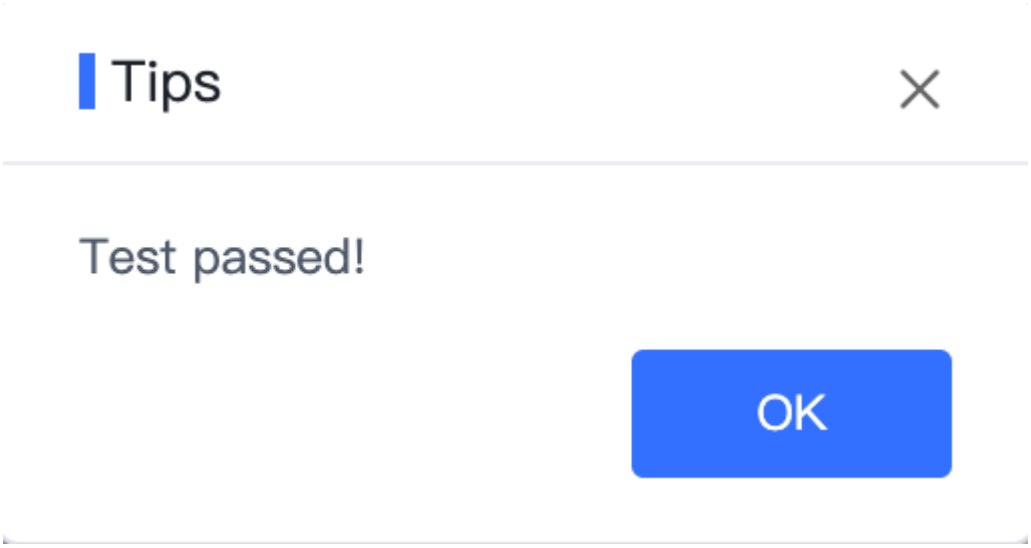
selectdb

3. After selecting the database, fill in the connection information of the Doris database



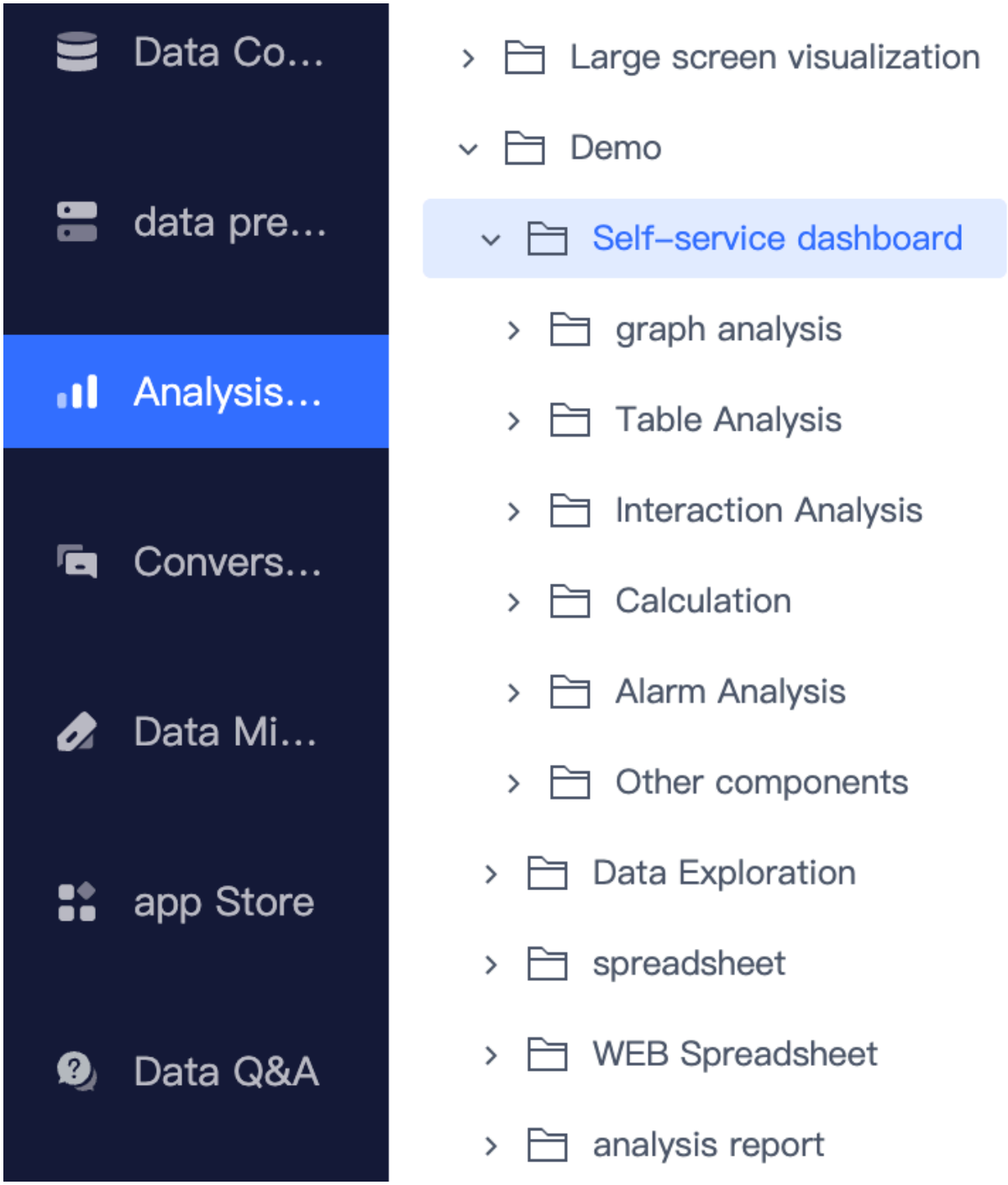
data source connection

4. If the information is correct and the network is connected, a message is displayed indicating that the Test passed.



test passed

5. After the connection is successful, you can customize and set the required report information in the data analysis module Self-service dashboard.



data analysis

CloudDM

Introduction

CloudDM, developed by Clougence, is a powerful cross-platform database management tool designed for both teams and individual users.

It provides dedicated support for key features of Apache Doris, offering capabilities such as data access, data masking, visual editing, and database CI/CD workflows.

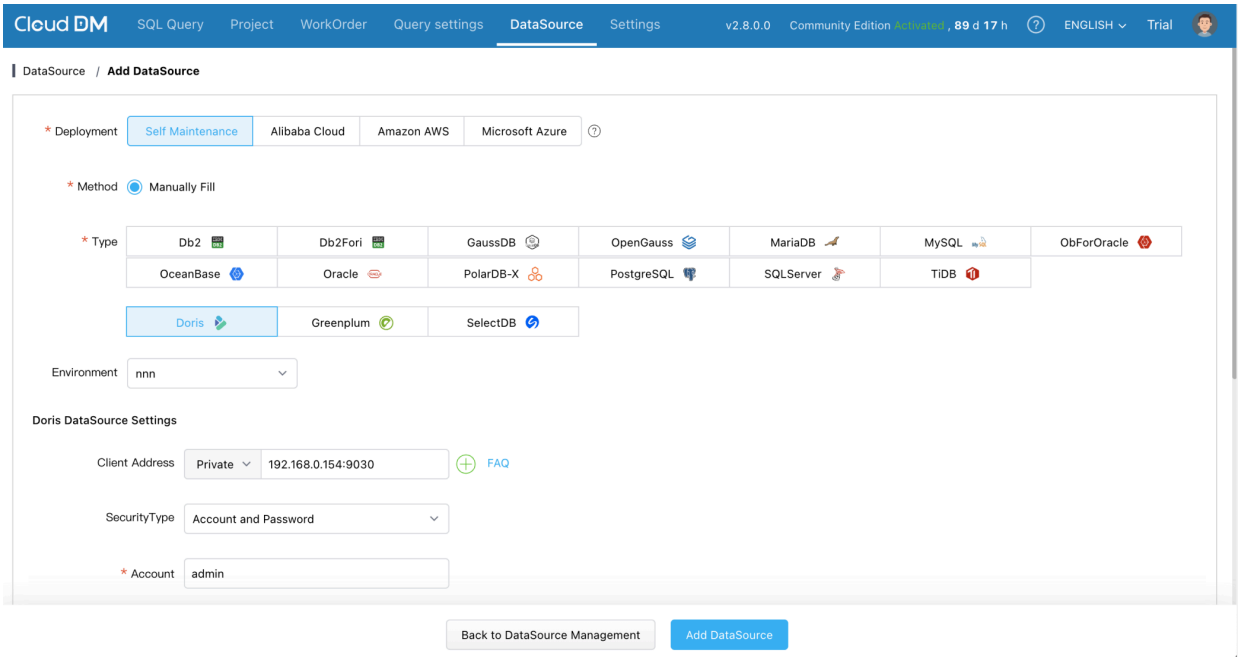
Preconditions

Install CloudDM. You can download and install CloudDM from https://www.cdmgr.com/.

Add data source

> **Note**
>
> CloudDM version should be 2.8.0.0 or above.

1. Log in to CloudDM.
2. Click **DataSources** > **Add DataSources**.
3. Select **Doris** for the Type.



add datasource

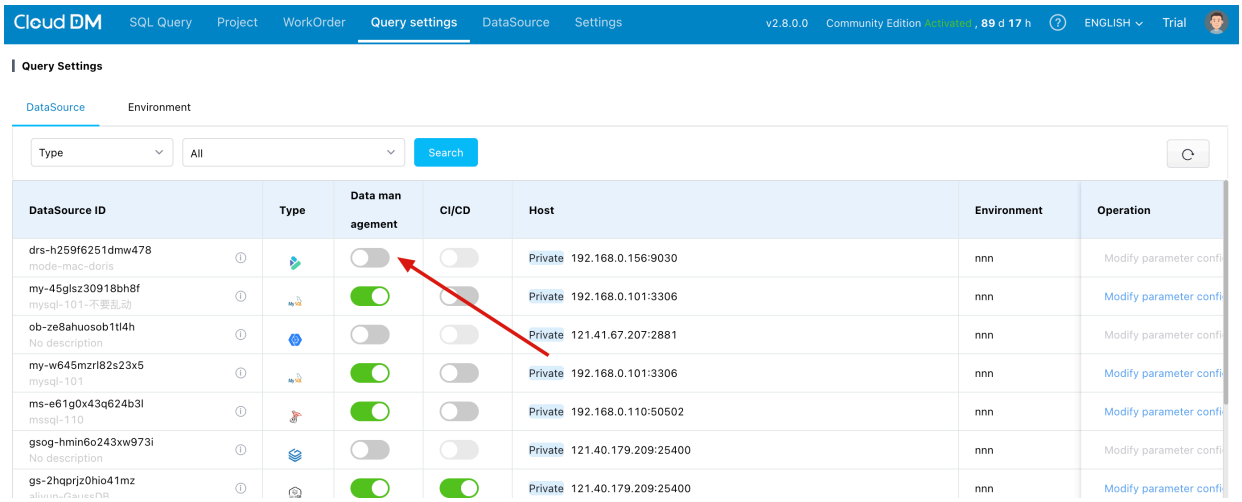
4. Enter the following required information to connect to the Doris instance.

- Client Address: The FE query port of the Doris cluster, for example: hostID:9030.
- Account: Username used to log in to the Doris cluster, for example: admin.
- Password: Password used to log in to the Doris cluster.

:::tip CloudDM can manage both internal catalogs and external catalogs in Doris. :::

:::info Note Managing the external catalog connected to Doris through the Database form of catalog.db requires Doris version 2.1.0 and above. :::

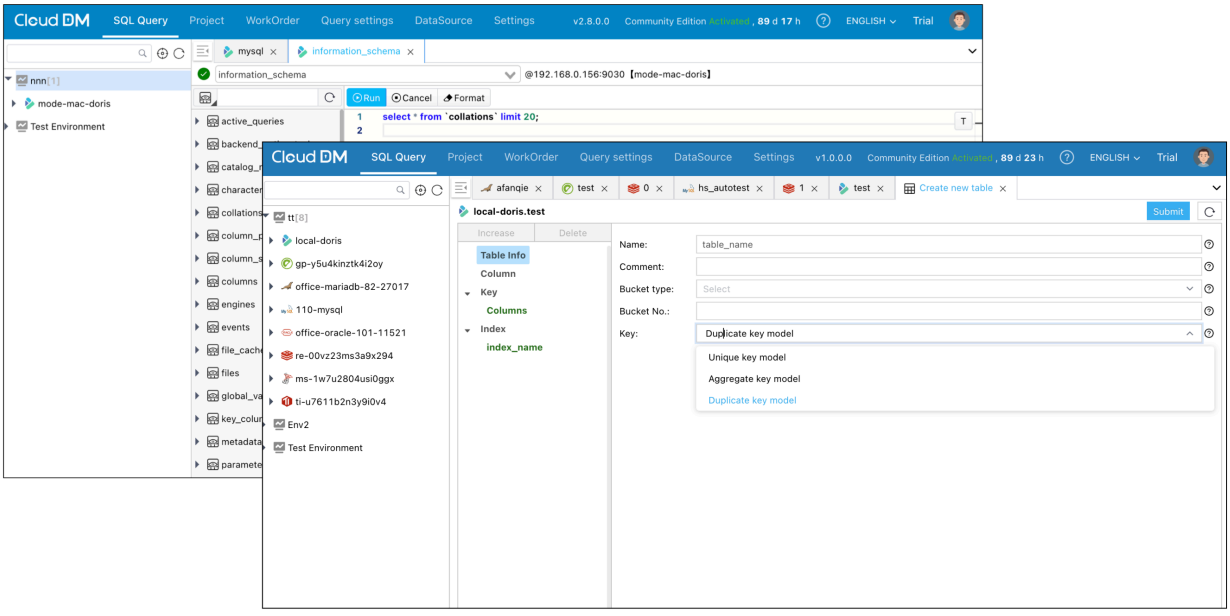
5. Click **Query Settings** in the top navigation bar. Then enable **Data Management**.



enable data manage

6. Access data.

   In the database connection navigation panel on the left, you can view the added Doris connection. Then you can start to manage the data through **CloudDM**.



connect to data source

Function support

- Query Client

    - Visually manage database objects in Doris
    - Write and execute SQL in the console
    - Export query results
- Team Collaboration
    - Statement-level authorization with table-level granularity
    - Workflow and approval for SQL requests
    - Database CI/CD
    - Data masking for sensitive information
    - SQL review rules

DBeaver

introduce

DBeaver is a cross-platform database tool for developers, database administrators, analysts and anyone who works with data.

Apache Doris is highly compatible with the MySQL protocol. You can use DBeaver's MySQL driver to connect to Apache Doris and query data in the internal catalog and external catalog.

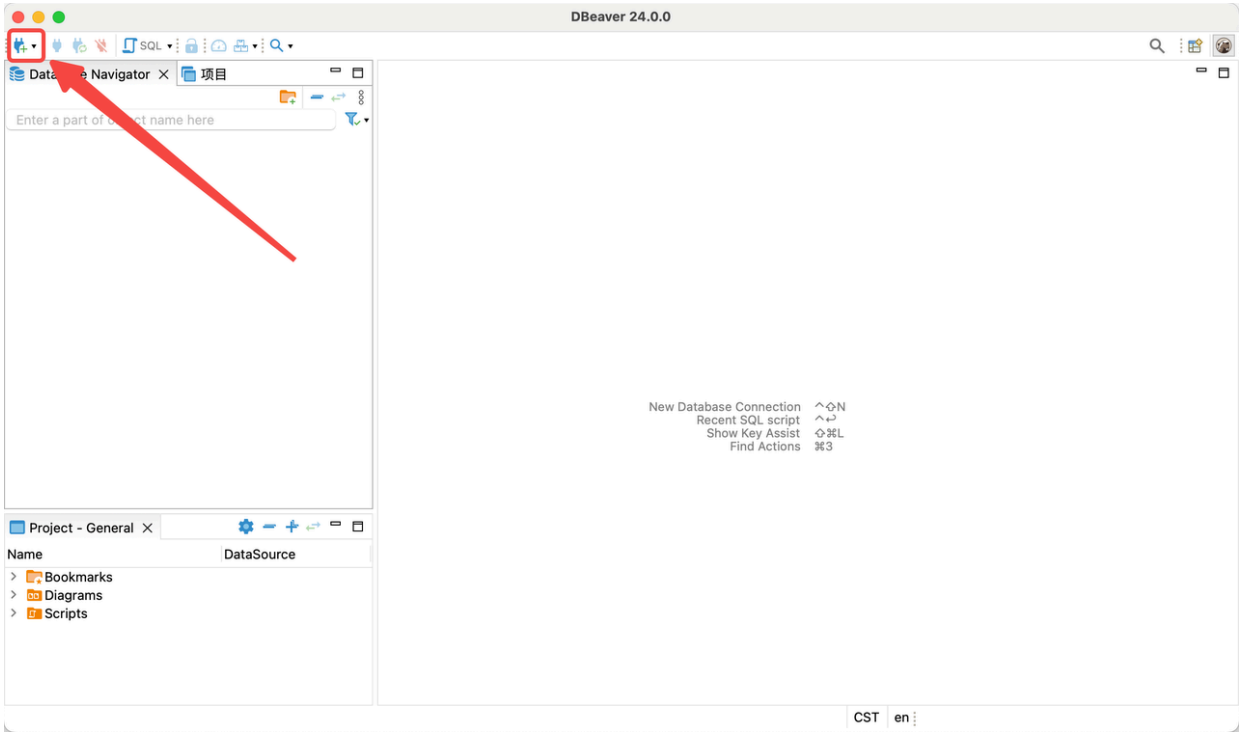
Preconditions

Dbeaver installed You can visit https://dbeaver.io to download and install DBeaver

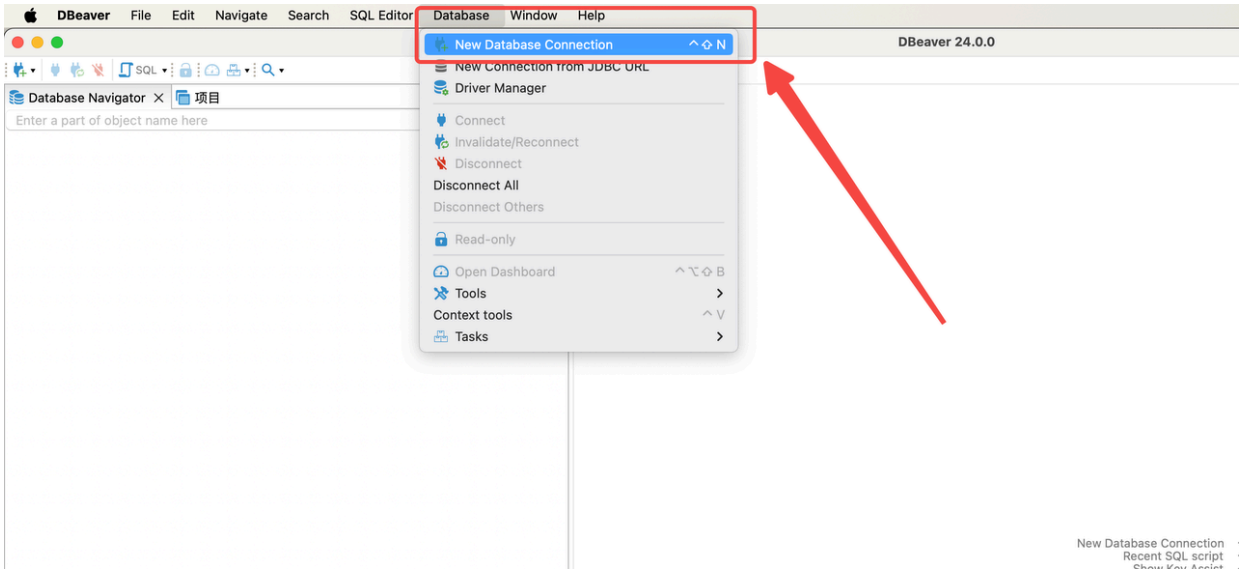
Add data source

> **Note**
>
> Currently verified using DBeaver version 24.0.0

1. Start DBeaver
2. Click the plus sign (**+**) icon in the upper left corner of the DBeaver window, or select **Database > New Database Connection** in the menu bar to open the **Connect to a database** interface.



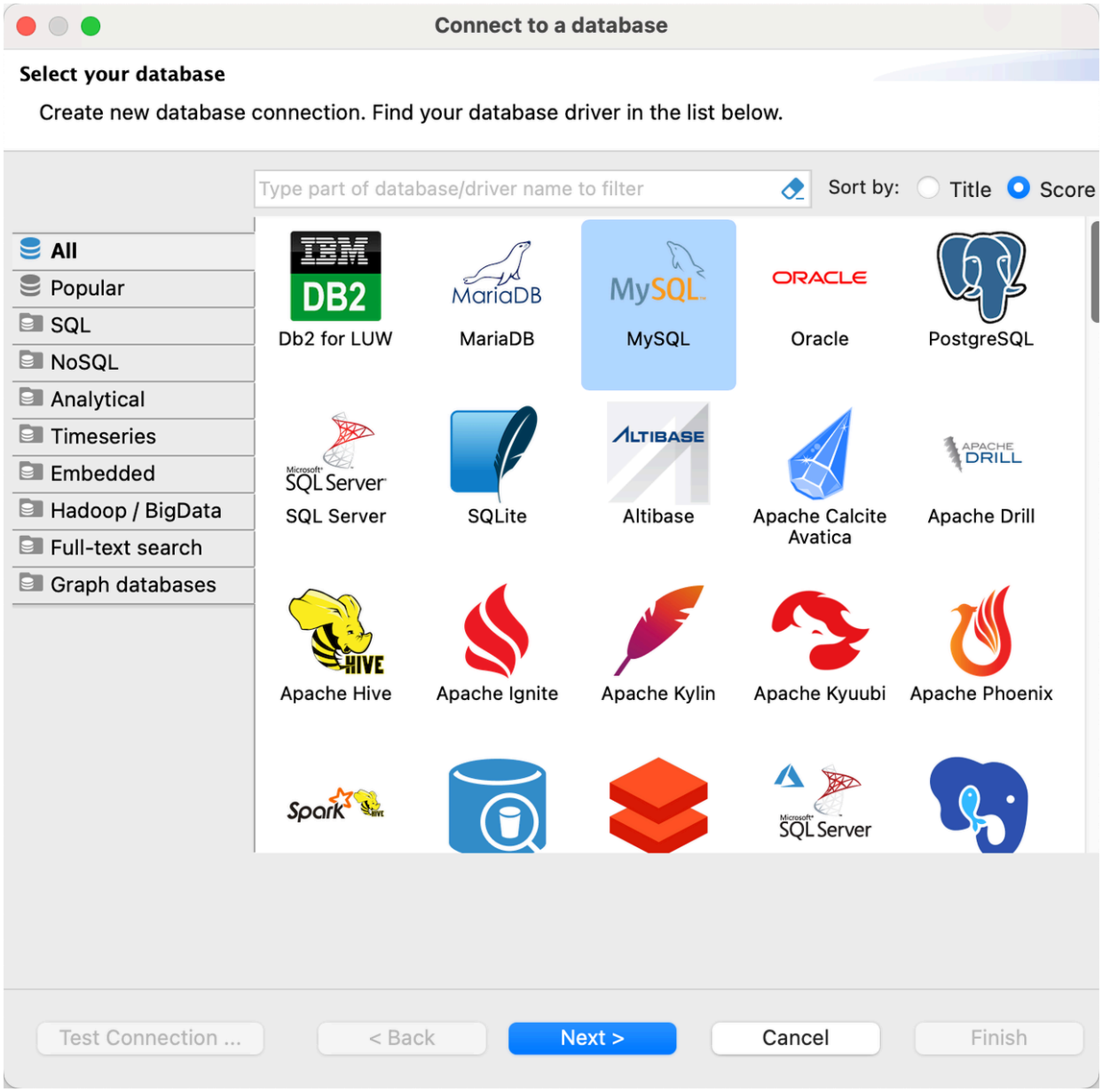
add connection 1



add connection 2

3. Select the MySQL driver

   In the **Select your database** window, select **MySQL**.



chose driver

4. Configure Doris connection

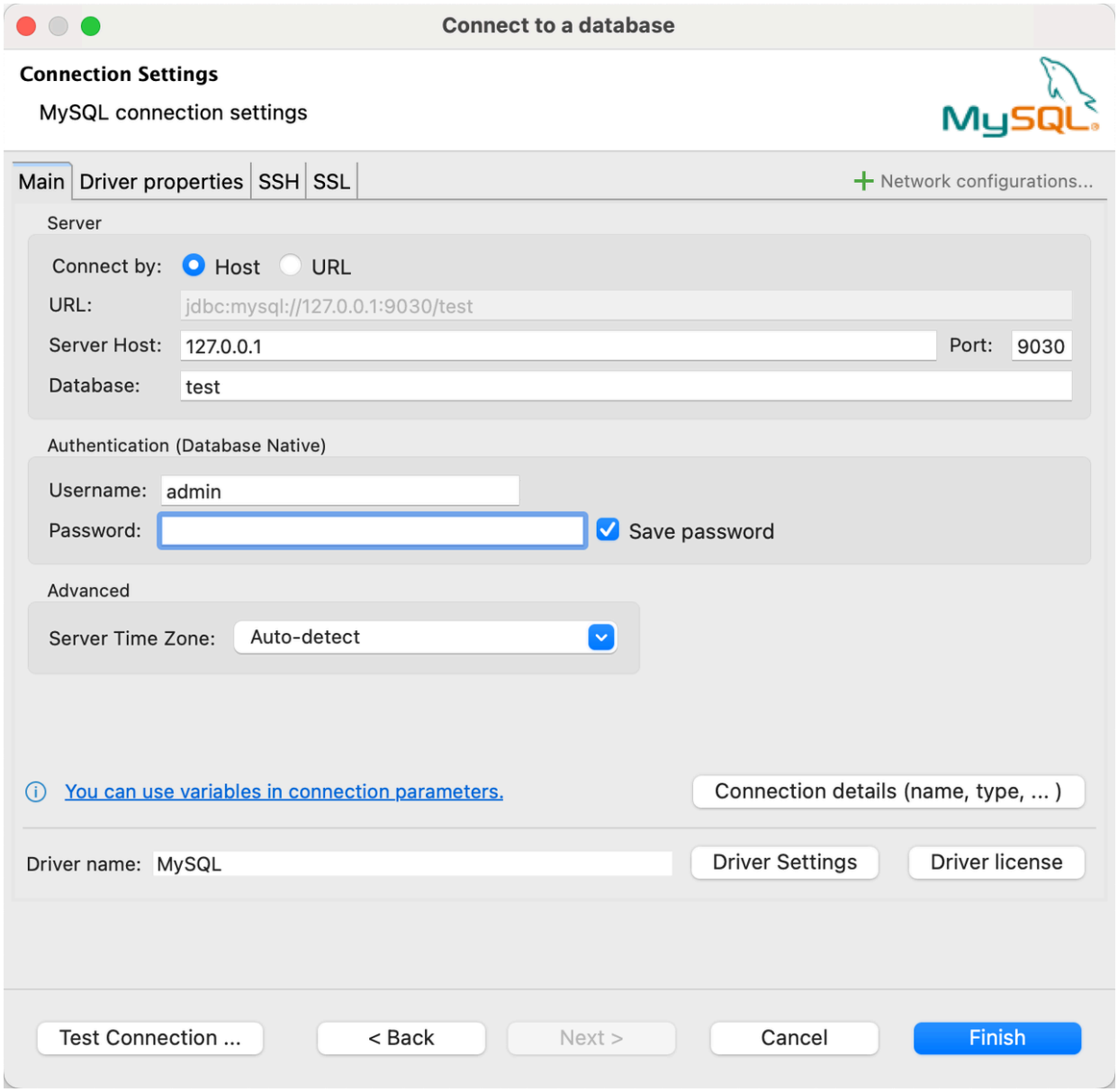
   In the **main** tab of the **Connection Settings** window, configure the following connection information:

- Server Host: FE host IP address of the Doris cluster.
- Port: FE query port of Doris cluster, such as 9030.
- Database: The target database in the Doris cluster.
- Username: The username used to log in to the Doris cluster, such as admin.
- Password: User password used to log in to the Doris cluster.

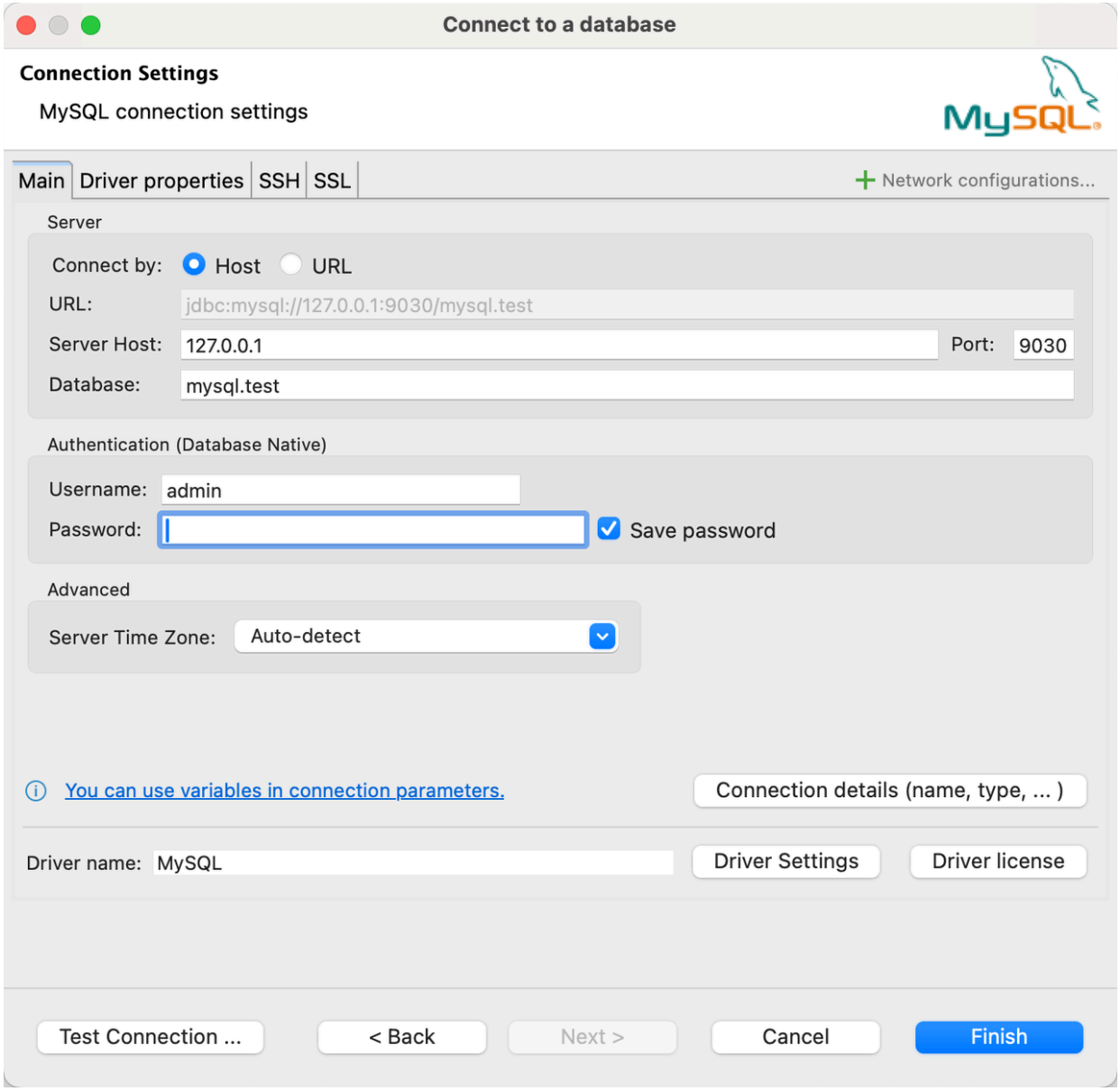
:::tip Database can be used to distinguish between internal catalog and external catalog. If only the Database name is filled in, the current data source will be connected to the internal catalog by default. If the format is catalog.db, the current data source will be connected to the catalog filled in Database by default, as shown in DBeaver The database tables are also database tables in the connected catalog, so you can use DBeaver's MySQL driver to create multiple Doris data sources to manage different Catalogs in Doris. :::

:::info Note Managing the external catalog connected to Doris through the Database form of catalog.db requires Doris version 2.1.0 and above. :::

- internal catalog



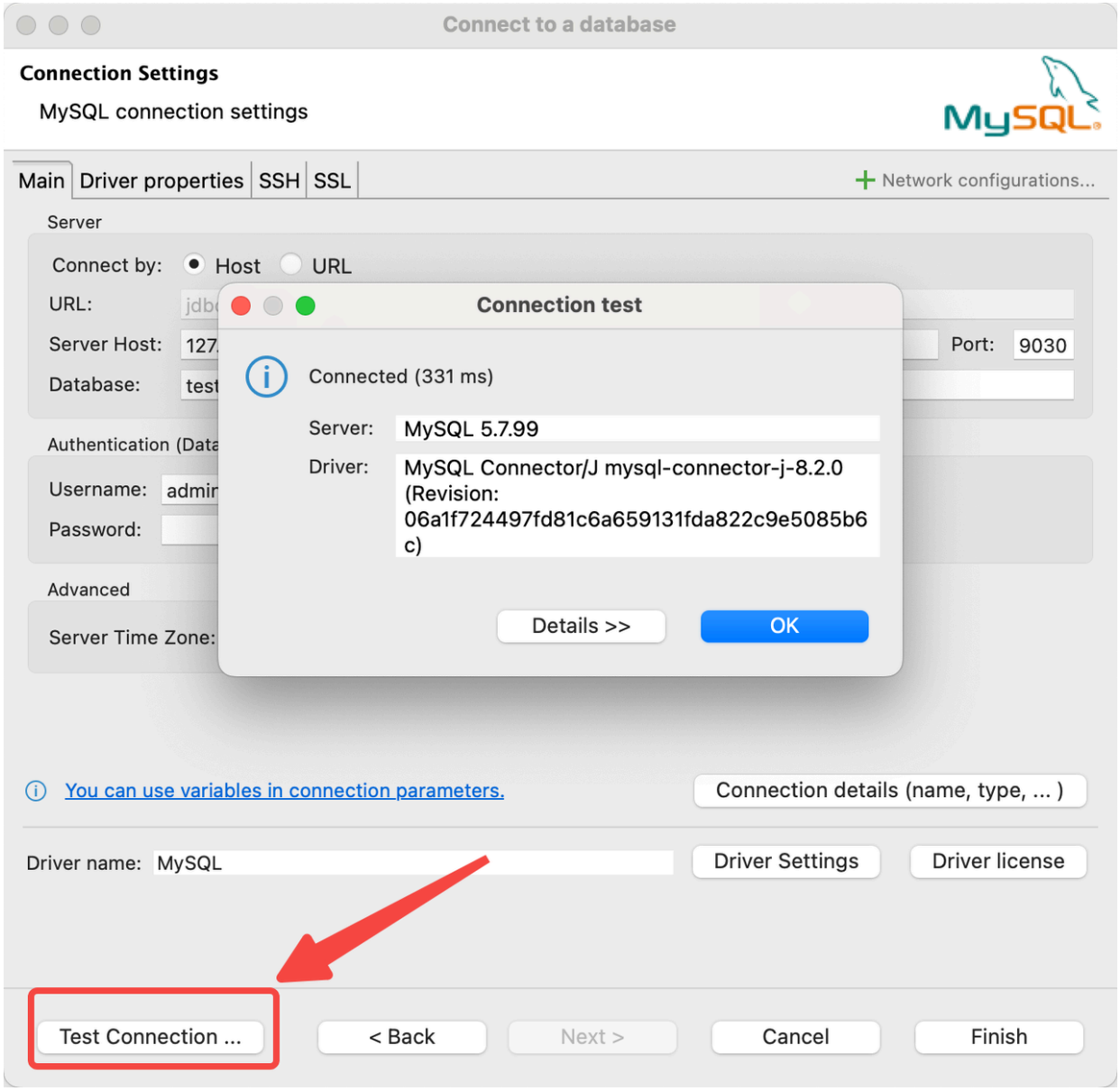
- external catalog



5. Test data source connection

   After filling in the connection information, click Test Connection in the lower left corner to verify the accuracy of the database connection information. DBeaver returns to the following dialog box to confirm the configuration of the connection information. Click OK to confirm that the configured connection information is correct. Then click Finish in the lower right corner to complete the connection

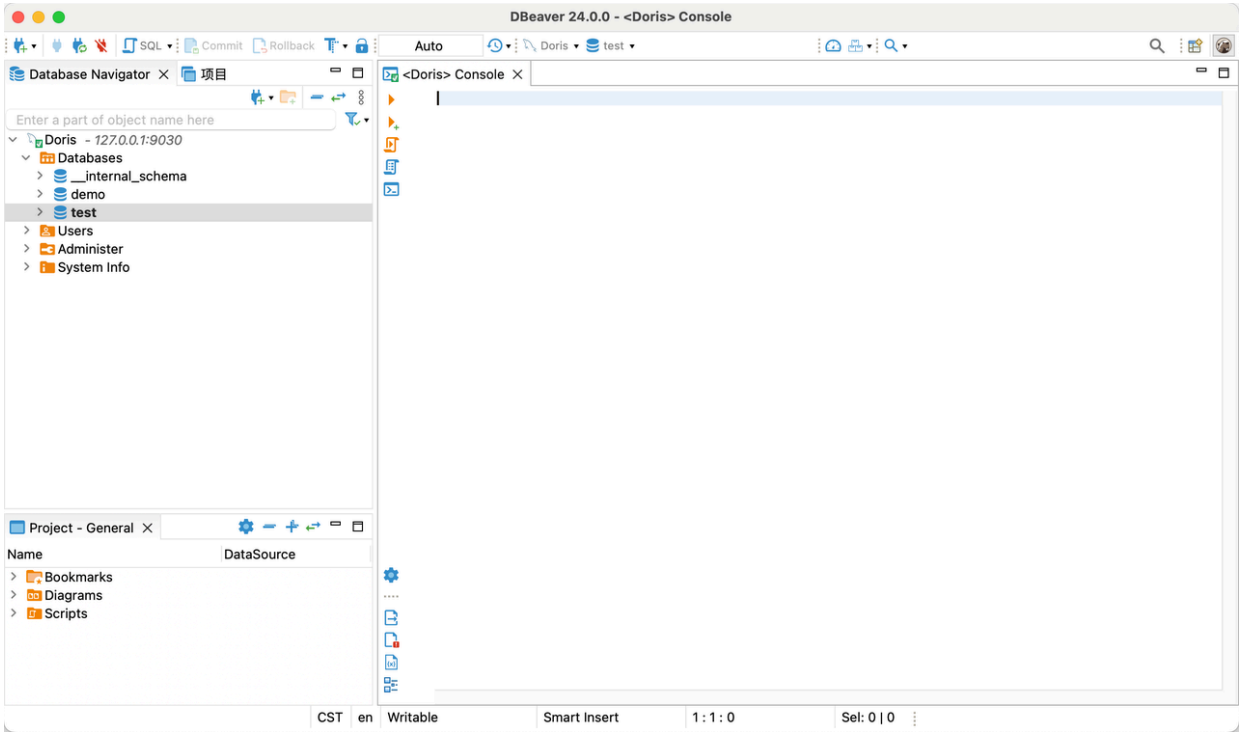
configuration.



6. Connect to database

   After the database connection is established, you can see the created data source connection in the database connection navigation on the left, and you can connect and manage the database through

DBeaver.



Function support

- fully support
    - Visual viewing class
        - Databases
            - Tables
            - Views
        - Users
            - Administer
        - Session Manager
        - System Info
            - Session Variables
            - Global Variables
            - Engines
            - Charsets
            - User Priviages
            - Plugin
        - Operation class
            - SQL editor
            - SQL console
- basic support

  The basic support part means that you can click to view without error, but due to protocol compatibility issues, there may be incomplete display.

    - Visual viewing class

        - dash board
        - Users/user/properties
        - Session Status
        - Global Status

- not support

  The unsupported part means that when using DBeaver to manage Doris, errors may be reported when performing certain visual operations, or some visual operations are not verified. Such as visual creation of database tables, schema change, addition, deletion and modification of data, etc.

DataGrip

introduce

DataGrip is a powerful cross-platform database tool for relational and NoSQL databases from JetBrains.

Apache Doris is highly compatible with the MySQL protocol. You can use DataGrip's MySQL data source to connect to Apache Doris and query data in the internal catalog and external catalog.

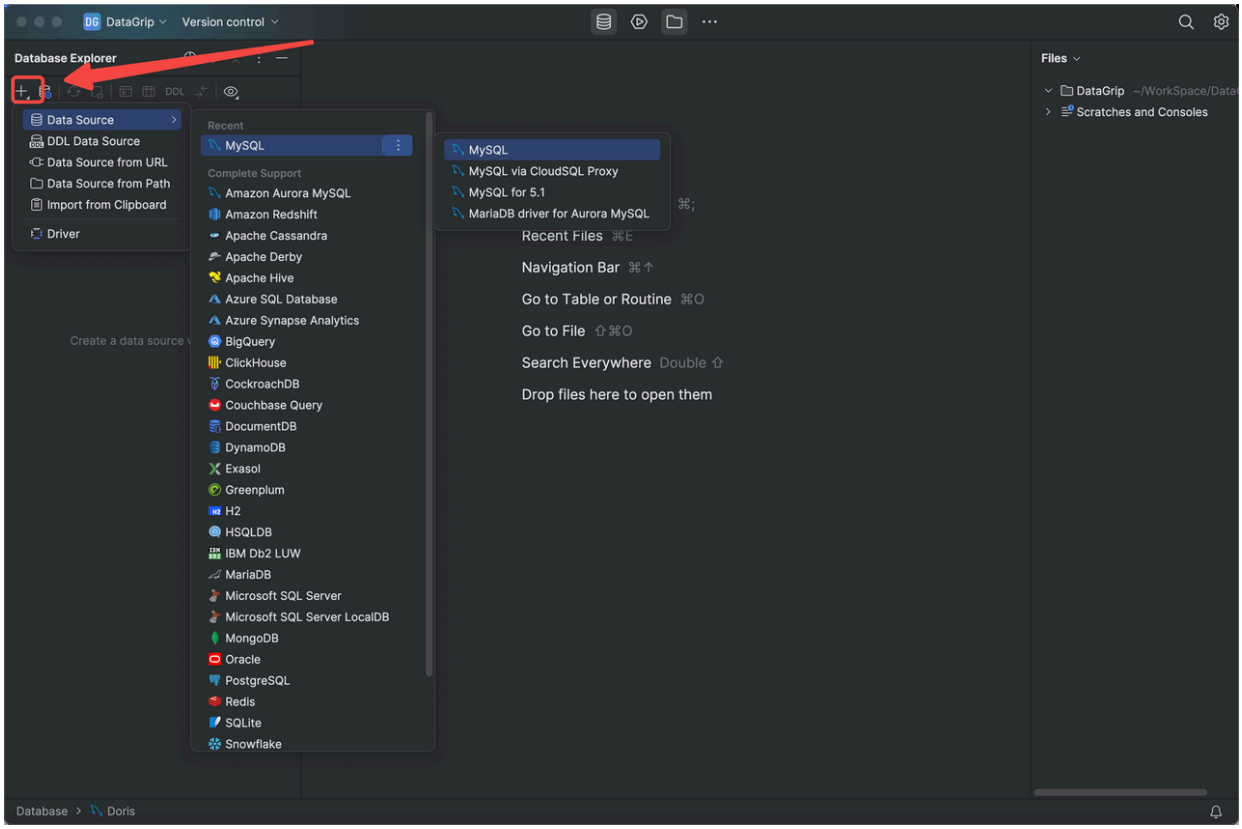
Preconditions

DataGrip installed You can visit www.jetbrains.com/datagrip/ to download and install DataGrip

Add data source

> **Note**
>
> Currently verified using DataGrip version 2023.3.4

1. Start DataGrip
2. Click the plus sign (**+**) icon in the upper left corner of the DataGrip window and select the MySQL data source



add data source

3. Configure Doris connection

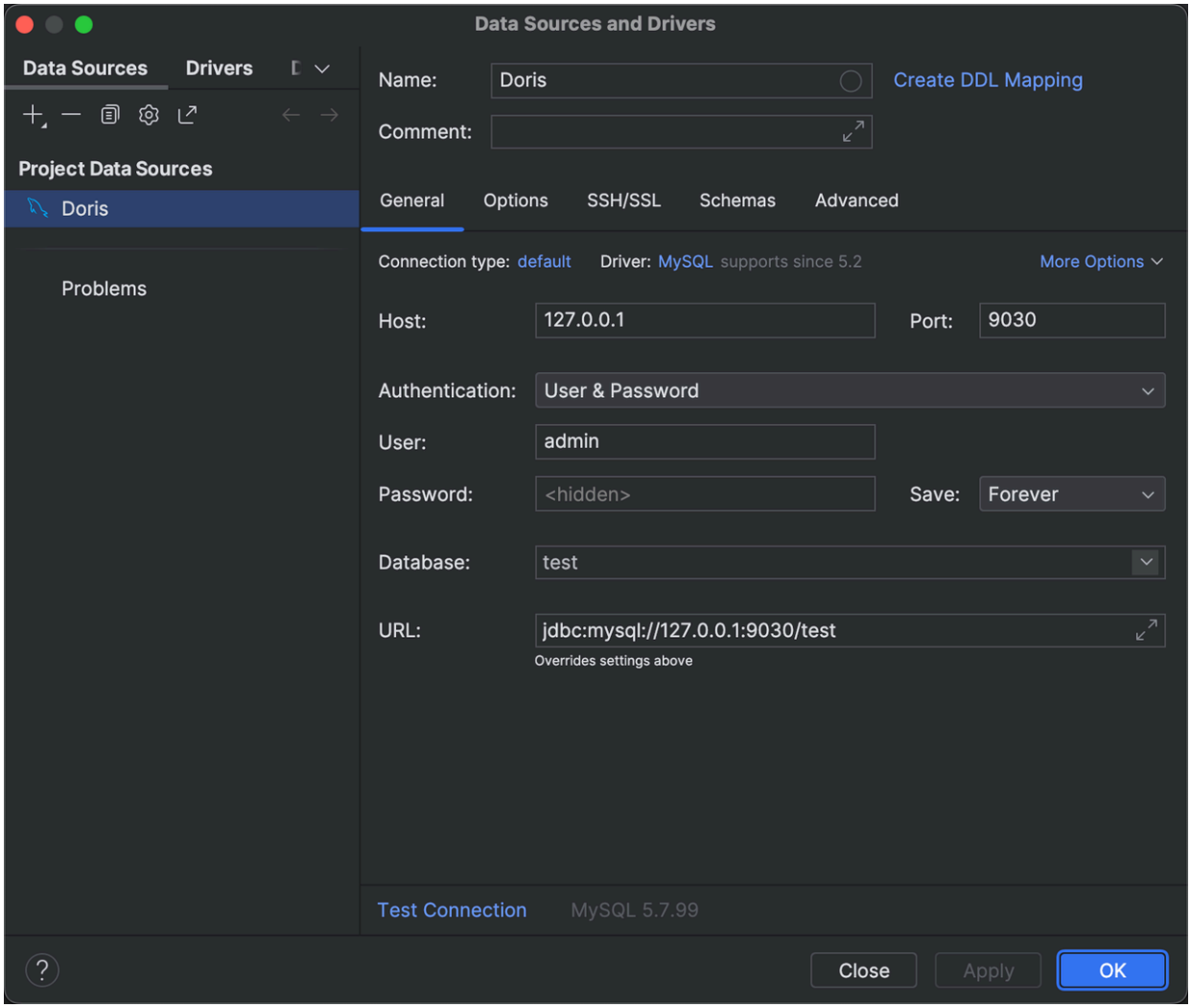
   On the General tab of the Data Sources and Drivers window, configure the following connection information:

- Host: FE host IP address of the Doris cluster.
- Port: FE query port of Doris cluster, such as 9030.
- Database: The target database in the Doris cluster.
- User: User name used to log in to the Doris cluster, such as admin.
- Password: User password used to log in to the Doris cluster.

  Database can be used to distinguish between internal catalog and external catalog. If only the Database name is filled in, the current data source will be connected to the internal catalog by default. If the format is catalog.db, the current data source will be connected to the catalog filled in Database by default, as shown in DataGrip The database table is also a database table in the connected catalog. In this way, you can use DataGrip's MySQL data source to create multiple Doris data sources to manage different Catalogs in Doris.

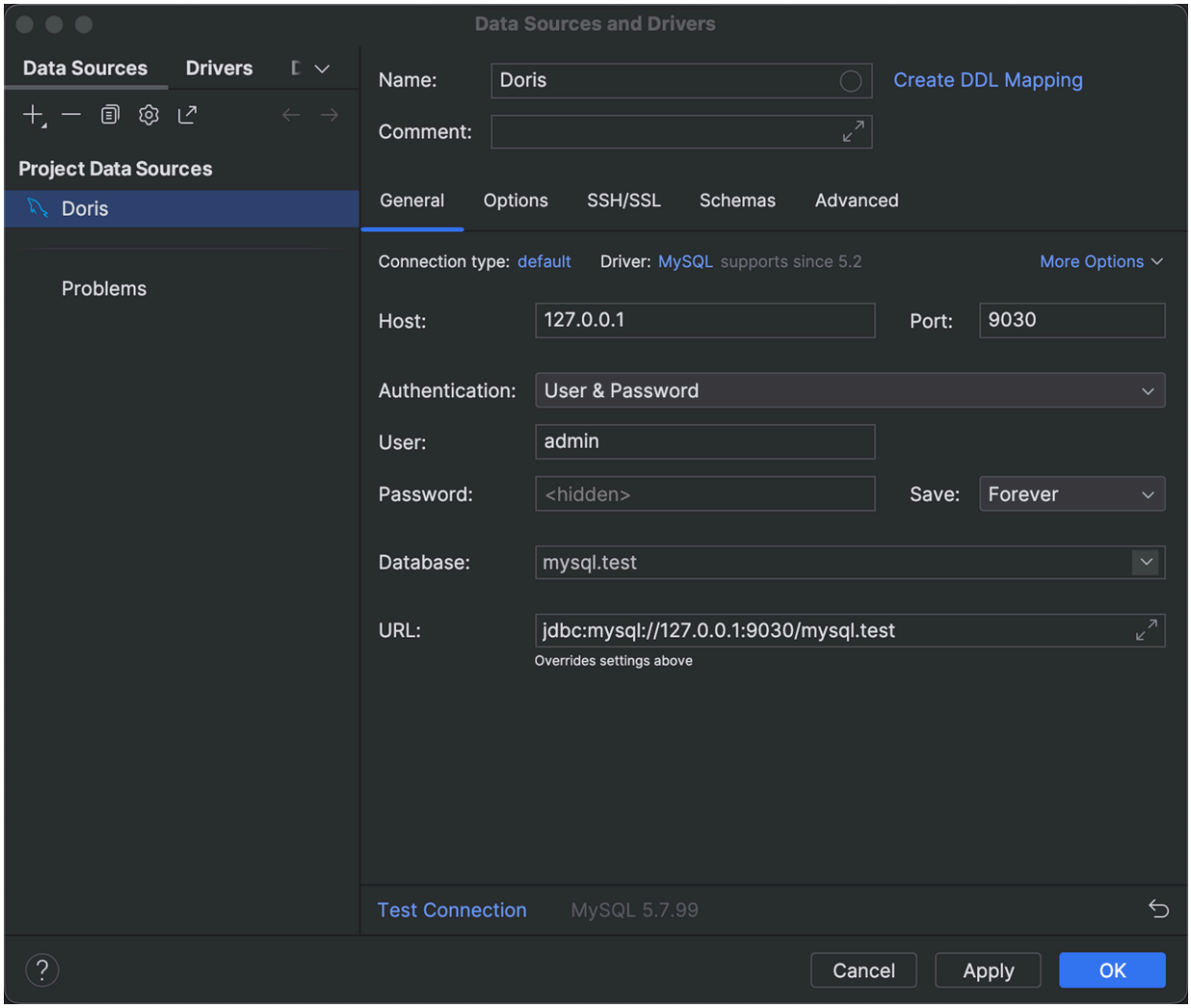
  :::info Note Managing the external catalog connected to Doris through the Database form of catalog.db requires Doris version 2.1.0 and above. :::

- internal catalog



connect internal catalog

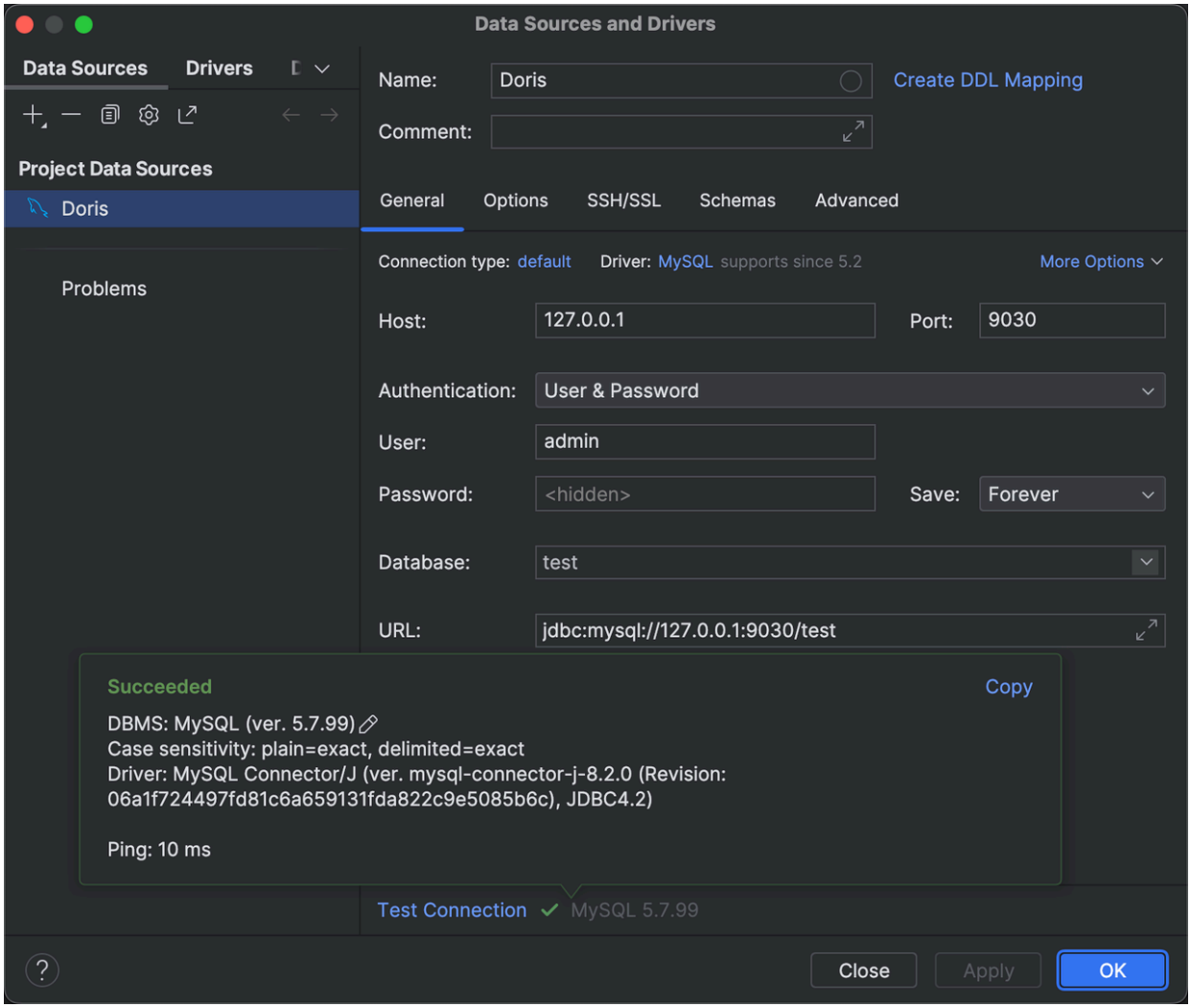
- external catalog



connect external catalog

5. Test data source connection

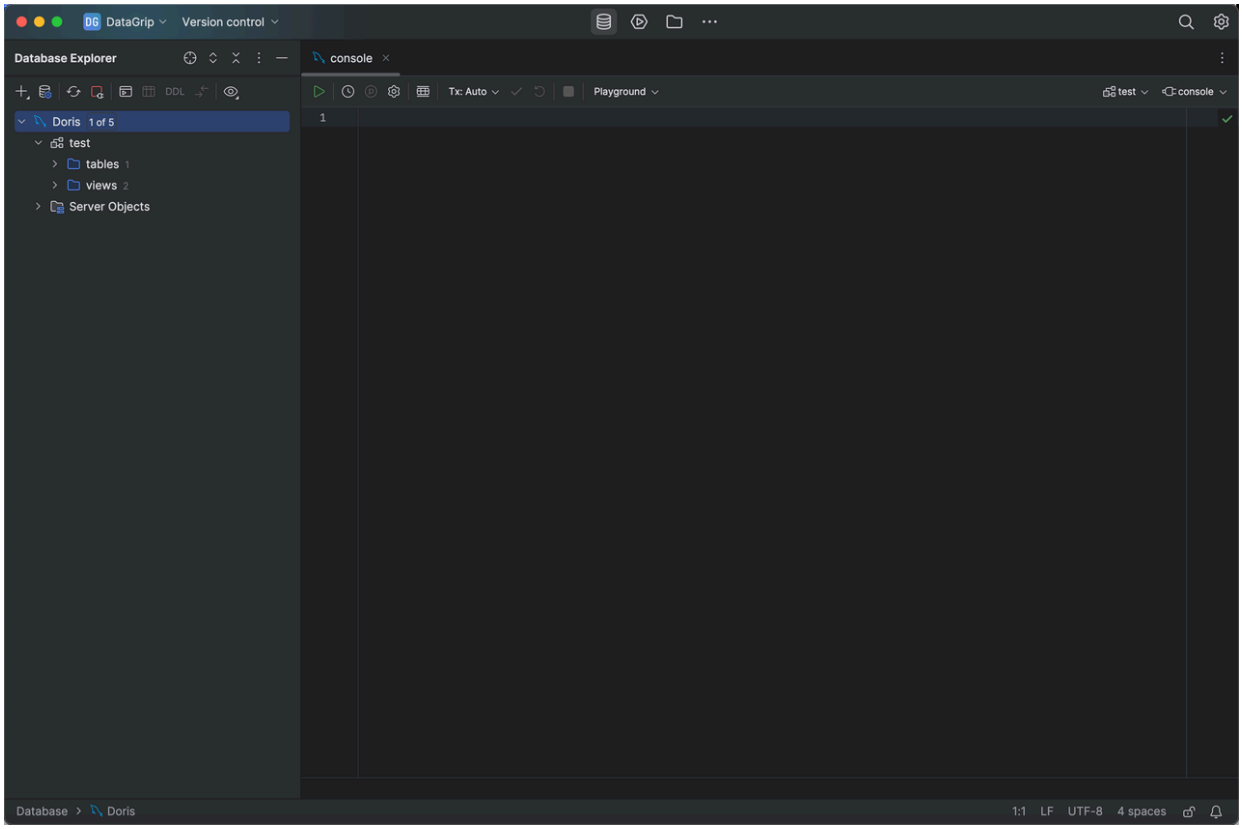
   After filling in the connection information, click Test Connection in the lower left corner to verify the accuracy of the database connection information. If DBeaver returns the following pop-up window, the test connection is successful. Then click OK in the lower right corner to complete the connection configuration.



test connection

6. Connect to database

   After the database connection is established, you can see the created data source connection in the database connection navigation on the left, and you can connect and manage the database through DataGrip.



create connection

Function support

Basically supports most visual viewing operations, as well as SQL console writing SQL operations Doris does not support or has not been verified various operations such as creating database tables, schema changes, and adding, deleting, and modifying data.

Logstash

> **Unavailable**
>
> Some package downloads in this document are currently unavailable. The related download commands may not work until the files are restored.

Logstash Doris output plugin

Introduction

Logstash is a log ETL framework (collect, preprocess, send to storage systems) that supports custom output plugins to write data into storage systems. The Logstash Doris output plugin is a plugin for outputting data to Doris.

The Logstash Doris output plugin calls the Doris Stream Load HTTP interface to write data into Doris in real-time, offering capabilities such as multi-threaded concurrency, failure retries, custom Stream Load formats and parameters, and output write speed.

Using the Logstash Doris output plugin mainly involves three steps: 1. Install the plugin into Logstash 2. Configure the Doris output address and other parameters 3. Start Logstash to write data into Doris in real-time

Installation

Obtaining the Plugin

You can download the plugin from the official website or compile it from the source code yourself.

- Download from the official website

```
# Installation package with dependencies
wget https://assets-cdn.selectdb.com/downloads/releases/extension/logstash-output-doris-1.2.0-java.gem
```

- Compile from source code

```
cd extension/logstash/

gem build logstash-output-doris.gemspec
```

Installing the Plugin

- Standard Installation

`${LOGSTASH_HOME}` is the installation directory of Logstash. Run the `bin/logstash-plugin` command under it to install the plugin.

```
${LOGSTASH_HOME}/bin/logstash-plugin install logstash-output-doris-1.2.0.gem

Validating logstash-output-doris-1.2.0.gem
Installing logstash-output-doris
Installation successful
```

The standard installation mode will automatically install the ruby modules that the plugin depends on. In cases where the network is not available, it will get stuck and cannot complete. In such cases, you can download the zip installation package with dependencies for a completely offline installation, noting that you need to use `file://` to specify the local file system.

- Offline Installation

```

export JARS_SKIP="true"

${LOGSTASH_HOME}/bin/logstash-plugin install logstash-output-doris-1.2.0.gem
```

Configuration

The configuration for the Logstash Doris output plugin is as follows:

Configuration	Description
`http_hosts`	Stream Load HTTP address, formatted as a string array, can have one or more elements, each element is host:port. For example: [“http://fe1:8030”, “http://fe2:8030”]
`user`	Doris username, this user needs to have import permissions for the corresponding Doris database and table
`password`	Password for the Doris user
`db`	The Doris database name to write into
`table`	The Doris table name to write into
`label_prefix`	Doris Stream Load Label prefix, the final generated Label is *{label_prefix}{db}{table}{yyyymmdd_hhmmss}{uuid}*, the default value is logstash
`headers`	Doris Stream Load headers parameter, the syntax format is a ruby map, for example: headers => { “format” => “json”, “read_json_by_line” => “true” }
`mapping`	Mapping from Logstash fields to Doris table fields, refer to the usage examples in the subsequent sections
`message_only`	A special form of mapping, only outputs the Logstash @message field to Doris, default is false
`max_retries`	Number of retries for Doris Stream Load requests on failure, default is -1 for infinite retries to ensure data reliability

Configuration	Description
log_request	Whether to output Doris Stream Load request and response metadata in logs for troubleshooting, default is false
log_speed_interval	Time interval for outputting speed in logs, unit is seconds, default is 10, setting to 0 can disable this type of logging

Usage Example

TEXT Log Collection Example

This example demonstrates TEXT log collection using Doris FE logs as an example.

1. Data

FE log files are typically located at the fe/log/fe.log file under the Doris installation directory. They are typical Java program logs, including fields such as timestamp, log level, thread name, code location, and log content. Not only do they contain normal logs, but also exception logs with stacktraces, which are multiline. Log collection and storage need to combine the main log and stacktrace into a single log entry.

```
2024-07-08 21:18:01,432 INFO (Statistics Job Appender|61) [StatisticsJobAppender.runAfterCatalogReady():70] Stats table not
available, skip
2024-07-08 21:18:53,710 WARN (STATS_FETCH-0|208) [StmtExecutor.executeInternalQuery():3332] Failed to run internal SQL:
OriginStatement{originStmt='SELECT * FROM __internal_schema.column_statistics WHERE part_id is NULL  ORDER BY update_time DESC LIMIT
500000', idx=0}
org.apache.doris.common.UserException: errCode = 2, detailMessage = tablet 10031 has no queryable replicas. err: replica 10032's
backend 10008 does not exist or not alive
        at org.apache.doris.planner.OlapScanNode.addScanRangeLocations(OlapScanNode.java:931) ~[doris-fe.jar:1.2-SNAPSHOT]
        at org.apache.doris.planner.OlapScanNode.computeTabletInfo(OlapScanNode.java:1197) ~[doris-fe.jar:1.2-SNAPSHOT]
```

2. Table Creation

The table structure includes fields such as the log's creation time, collection time, hostname, log file path, log type, log level, thread name, code location, and log content.

```
CREATE TABLE `doris_log` (
  `log_time` datetime NULL COMMENT 'log content time',
  `collect_time` datetime NULL COMMENT 'log agent collect time',
  `host` text NULL COMMENT 'hostname or ip',
  `path` text NULL COMMENT 'log file path',
  `type` text NULL COMMENT 'log type',
  `level` text NULL COMMENT 'log level',
  `thread` text NULL COMMENT 'log thread',
  `position` text NULL COMMENT 'log code position',
  `message` text NULL COMMENT 'log message',
  INDEX idx_host (`host`) USING INVERTED COMMENT '',
  INDEX idx_path (`path`) USING INVERTED COMMENT '',
  INDEX idx_type (`type`) USING INVERTED COMMENT '',
  INDEX idx_level (`level`) USING INVERTED COMMENT '',
  INDEX idx_thread (`thread`) USING INVERTED COMMENT '',
  INDEX idx_position (`position`) USING INVERTED COMMENT '',
  INDEX idx_message (`message`) USING INVERTED PROPERTIES("parser" = "unicode", "support_phrase" = "true") COMMENT ''
) ENGINE=OLAP
DUPLICATE KEY(`log_time`)
COMMENT 'OLAP'
PARTITION BY RANGE(`log_time`) ()
```

```
DISTRIBUTED BY RANDOM BUCKETS 10
PROPERTIES (
"replication_num" = "1",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.start" = "-7",
"dynamic_partition.end" = "1",
"dynamic_partition.prefix" = "p",
"dynamic_partition.buckets" = "10",
"dynamic_partition.create_history_partition" = "true",
"compaction_policy" = "time_series"
);
```

3. Logstash Configuration

Logstash mainly has two types of configuration files: one for the entire Logstash system and another for a specific log collection.

The configuration file for the entire Logstash system is usually located at config/logstash.yml. To improve performance when writing to Doris, it is necessary to modify the batch size and batch delay. For logs with an average size of a few hundred bytes per line, a batch size of 1,000,000 lines and a batch delay of 10 seconds are recommended.

```
pipeline.batch.size: 1000000
pipeline.batch.delay: 10000
```

The configuration file for a specific log collection, such as logstash_doris_log.conf, mainly consists of three parts corresponding to the various stages of ETL: 1. Input is responsible for reading the raw data. 2. Filter is responsible for data transformation. 3. Output is responsible for sending the data to the output destination.

```
# 1. input is responsible for reading raw data
# File input is an input plugin that can be configured to read the log file of the configured path. It uses the multiline codec to
concatenate lines that do not start with a timestamp to the end of the previous line, achieving the effect of merging stacktraces
with the main log. File input saves the log content in the @message field, and there are also some metadata fields such as host,
log.file.path. Here, we manually add a field named type through add_field, with its value set to fe.log.
input {
    file {
        path => "/mnt/disk2/xiaokang/opt/doris_master/fe/log/fe.log"
        add_field => {"type" => "fe.log"}
        codec => multiline {
            # valid line starts with timestamp
            pattern => "^%{TIMESTAMP_ISO8601} "
            # any line not starting with a timestamp should be merged with the previous line
            negate => true
            what => "previous"
        }
    }
}

# 2. filter section is responsible for data transformation
# grok is a commonly used data transformation plugin that has some built-in patterns, such as TIMESTAMP_ISO8601 for parsing
timestamps, and also supports writing regular expressions to extract fields.
filter {
    grok {
        match => {
            # parse log_time, level, thread, position fields from message
            "message" => "%{TIMESTAMP_ISO8601:log_time} (?<level>[A-Z]+) \((?<thread>[^\[]*)\) \[(?<position>[^\]]*)\]"
        }
    }
```

```
}

# 3. output section is responsible for data output
# Doris output sends data to Doris using the Stream Load HTTP interface. The data format for Stream Load is specified as JSON through the headers parameter, and the mapping parameter specifies the mapping from Logstash fields to JSON fields. Since headers specify "format" => "json", Stream Load will automatically parse the JSON fields and write them into the corresponding fields of the Doris table.
output {
    doris {
        http_hosts => ["http://localhost:8630"]
        user => "root"
        password => ""
        db => "log_db"
        table => "doris_log"
        headers => {
          "format" => "json"
          "read_json_by_line" => "true"
          "load_to_single_tablet" => "true"
        }
        mapping => {
          "log_time" => "%{log_time}"
          "collect_time" => "%{@timestamp}"
          "host" => "%{[host][name]}"
          "path" => "%{[log][file][path]}"
          "type" => "%{type}"
          "level" => "%{level}"
          "thread" => "%{thread}"
          "position" => "%{position}"
          "message" => "%{message}"
        }
        log_request => true
    }
}
```

4. Running Logstash

```
${LOGSTASH_HOME}/bin/logstash -f config/logstash_doris_log.conf

# When log_request is set to true, the log will output the request parameters and response results of each Stream Load.
[2024-07-08T22:35:34,772][INFO ][logstash.outputs.doris   ][main][e44d2a24f17d764647ce56f5fed24b9bbf08d3020c7fddcc3298800daface80a]
doris stream load response:
{
    "TxnId": 45464,
    "Label": "logstash_log_db_doris_log_20240708_223532_539_6c20a0d1-dcab-4b8e-9bc0-76b46a929bd1",
    "Comment": "",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 452,
    "NumberLoadedRows": 452,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 277230,
    "LoadTimeMs": 1797,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 18,
    "ReadDataTimeMs": 9,
    "WriteDataTimeMs": 1758,
    "CommitAndPublishTimeMs": 18
}

# By default, speed information is logged every 10 seconds, including the amount of data since startup (in MB and ROWS), the total
```

```
speed (in MB/s and R/s), and the speed in the last 10 seconds.

[2024-07-08T22:35:38,285][INFO ][logstash.outputs.doris   ][main] total 11 MB 18978 ROWS, total speed 0 MB/s 632 R/s, last 10
seconds speed 1 MB/s 1897 R/s
```

JSON Log Collection Example

This example demonstrates JSON log collection using data from the GitHub events archive.

1. Data

The GitHub events archive contains archived data of GitHub user actions, formatted as JSON. It can be downloaded from here, for example, the data for January 1, 2024, at 3 PM.

```
wget https://data.gharchive.org/2024-01-01-15.json.gz
```

Below is a sample of the data. Normally, each piece of data is on a single line, but for ease of display, it has been formatted here.

```
{
  "id": "37066529221",
  "type": "PushEvent",
  "actor": {
    "id": 46139131,
    "login": "Bard89",
    "display_login": "Bard89",
    "gravatar_id": "",
    "url": "https://api.github.com/users/Bard89",
    "avatar_url": "https://avatars.githubusercontent.com/u/46139131?"
  },
  "repo": {
    "id": 780125623,
    "name": "Bard89/talk-to-me",
    "url": "https://api.github.com/repos/Bard89/talk-to-me"
  },
  "payload": {
    "repository_id": 780125623,
    "push_id": 17799451992,
    "size": 1,
    "distinct_size": 1,
    "ref": "refs/heads/add_mvcs",
    "head": "f03baa2de66f88f5f1754ce3fa30972667f87e81",
    "before": "85e6544ede4ae3f132fe2f5f1ce0ce35a3169d21"
  },
  "public": true,
  "created_at": "2024-04-01T23:00:00Z"
}
```

2. Table Creation

```
CREATE DATABASE log_db;
USE log_db;


CREATE TABLE github_events
(
  `created_at` DATETIME,
```

```
  `id` BIGINT,
  `type` TEXT,
  `public` BOOLEAN,
  `actor.id` BIGINT,
  `actor.login` TEXT,
  `actor.display_login` TEXT,
  `actor.gravatar_id` TEXT,
  `actor.url` TEXT,
  `actor.avatar_url` TEXT,
  `repo.id` BIGINT,
  `repo.name` TEXT,
  `repo.url` TEXT,
  `payload` TEXT,
  `host` TEXT,
  `path` TEXT,
  INDEX `idx_id` (`id`) USING INVERTED,
  INDEX `idx_type` (`type`) USING INVERTED,
  INDEX `idx_actor.id` (`actor.id`) USING INVERTED,
  INDEX `idx_actor.login` (`actor.login`) USING INVERTED,
  INDEX `idx_repo.id` (`repo.id`) USING INVERTED,
  INDEX `idx_repo.name` (`repo.name`) USING INVERTED,
  INDEX `idx_host` (`host`) USING INVERTED,
  INDEX `idx_path` (`path`) USING INVERTED,
  INDEX `idx_payload` (`payload`) USING INVERTED PROPERTIES("parser" = "unicode", "support_phrase" = "true")
)
ENGINE = OLAP
DUPLICATE KEY(`created_at`)
PARTITION BY RANGE(`created_at`) ()
DISTRIBUTED BY RANDOM BUCKETS 10
PROPERTIES (
"replication_num" = "1",
"compaction_policy" = "time_series",
"enable_single_replica_compaction" = "true",
"dynamic_partition.enable" = "true",
"dynamic_partition.create_history_partition" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.start" = "-30",
"dynamic_partition.end" = "1",
"dynamic_partition.prefix" = "p",
"dynamic_partition.buckets" = "10",
"dynamic_partition.replication_num" = "1"
);
```

3. Logstash Configuration

The configuration file differs from the previous TEXT log collection in the following aspects:

1. The codec parameter for file input is json. Logstash will parse each line of text as JSON format and use the parsed fields for subsequent processing.
2. No filter plugin is used because no additional processing or transformation is needed.

```
input {
    file {
        path => "/tmp/github_events/2024-04-01-23.json"
        codec => json
    }
}

output {
    doris {
        http_hosts => ["http://fe1:8630", "http://fe2:8630", "http://fe3:8630"]
        user => "root"
```

```
        password => ""
        db => "log_db"
        table => "github_events"
        headers => {
          "format" => "json"
          "read_json_by_line" => "true"
          "load_to_single_tablet" => "true"
        }
        mapping => {
          "created_at" => "%{created_at}"
          "id" => "%{id}"
          "type" => "%{type}"
          "public" => "%{public}"
          "actor.id" => "%{[actor][id]}"
          "actor.login" => "%{[actor][login]}"
          "actor.display_login" => "%{[actor][display_login]}"
          "actor.gravatar_id" => "%{[actor][gravatar_id]}"
          "actor.url" => "%{[actor][url]}"
          "actor.avatar_url" => "%{[actor][avatar_url]}"
          "repo.id" => "%{[repo][id]}"
          "repo.name" => "%{[repo][name]}"
          "repo.url" => "%{[repo][url]}"
          "payload" => "%{[payload]}"
          "host" => "%{[host][name]}"
          "path" => "%{[log][file][path]}"
        }
        log_request => true
    }
}
```

4. Running Logstash

```
${LOGSTASH_HOME}/bin/logstash -f logstash_github_events.conf
```

Filebeat

Beats Doris output plugin

Beats is a data collection agent that supports custom output plugins to write data into storage systems, with the Beats Doris output plugin being the one for outputting to Doris.

The Beats Doris output plugin supports Filebeat, Metricbeat, Packetbeat, Winlogbeat, Auditbeat, and Heartbeat.

By invoking the Doris Stream Load HTTP interface, the Beats Doris output plugin writes data into Doris in real-time, offering capabilities such as multi-threaded concurrency, failure retries, custom Stream Load formats and parameters, and output write speed.

To use the Beats Doris output plugin, there are three main steps: 1. Download or compile the Beats binary program that includes the Doris output plugin. 2. Configure the Beats output address and other parameters. 3. Start Beats to write data into Doris in real-time.

Installation

Download from the Official Website

https://download.velodb.io/extension/filebeat-doris-2.1.1

Compile from Source Code

Execute the following commands in the `extension/beats/` directory:

```
cd doris/extension/beats

go build -o filebeat-doris filebeat/filebeat.go
go build -o metricbeat-doris metricbeat/metricbeat.go
go build -o winlogbeat-doris winlogbeat/winlogbeat.go
go build -o packetbeat-doris packetbeat/packetbeat.go
go build -o auditbeat-doris auditbeat/auditbeat.go
go build -o heartbeat-doris heartbeat/heartbeat.go
```

Configuration

The configuration for the Beats Doris output plugin is as follows:

Configuration	Description
`http_hosts`	Stream Load HTTP address, formatted as a string array, can have one or more elements, each element is host:port. For example: [“http://fe1:8030”, “http://fe2:8030”]
`user`	Doris username, this user needs to have import permissions for the corresponding Doris database and table

Configuration	Description
password	Doris user's password
database	The Doris database name to write into
table	The Doris table name to write into
label_prefix	Doris Stream Load Label prefix, the final generated Label is *{label_prefix}{db}{table}{yyyymmdd_hhmmss}{uuid}*, the default value is beats
headers	Doris Stream Load headers parameter, syntax format is YAML map
codec_format_string	The format string for outputting to Doris Stream Load, %{[a][b]} represents the a.b field in the input, refer to the usage examples in subsequent sections
bulk_max_size	Doris Stream Load batch size, default is 100000
max_retries	Number of retries for Doris Stream Load requests on failure, default is -1 for infinite retries to ensure data reliability
log_request	Whether to output Doris Stream Load request and response metadata in logs for troubleshooting, default is true
log_progress_interval	Time interval for outputting speed in logs, unit is seconds, default is 10, setting to 0 can disable this type of logging

Usage Example

TEXT Log Collection Example

This example demonstrates TEXT log collection using Doris FE logs as an example.

1. Data

FE log files are typically located at the fe/log/fe.log file under the Doris installation directory. They are typical Java program logs, including fields such as timestamp, log level, thread name, code location, and log content. Not only do they contain normal logs, but also exception logs with stacktraces, which are multiline. Log collection and storage need to combine the main log and stacktrace into a single log entry.

```
2024-07-08 21:18:01,432 INFO (Statistics Job Appender|61) [StatisticsJobAppender.runAfterCatalogReady():70] Stats table not
available, skip
2024-07-08 21:18:53,710 WARN (STATS_FETCH-0|208) [StmtExecutor.executeInternalQuery():3332] Failed to run internal SQL:
OriginStatement{originStmt='SELECT * FROM __internal_schema.column_statistics WHERE part_id is NULL  ORDER BY update_time DESC LIMIT
500000', idx=0}
org.apache.doris.common.UserException: errCode = 2, detailMessage = tablet 10031 has no queryable replicas. err: replica 10032's
backend 10008 does not exist or not alive
        at org.apache.doris.planner.OlapScanNode.addScanRangeLocations(OlapScanNode.java:931) ~[doris-fe.jar:1.2-SNAPSHOT]
        at org.apache.doris.planner.OlapScanNode.computeTabletInfo(OlapScanNode.java:1197) ~[doris-fe.jar:1.2-SNAPSHOT]
```

2. Table Creation

The table structure includes fields such as the log's creation time, collection time, hostname, log file path, log type, log level, thread name, code location, and log content.

```
CREATE TABLE `doris_log` (
  `log_time` datetime NULL COMMENT 'log content time',
  `collect_time` datetime NULL COMMENT 'log agent collect time',
  `host` text NULL COMMENT 'hostname or ip',
  `path` text NULL COMMENT 'log file path',
  `type` text NULL COMMENT 'log type',
  `level` text NULL COMMENT 'log level',
  `thread` text NULL COMMENT 'log thread',
  `position` text NULL COMMENT 'log code position',
  `message` text NULL COMMENT 'log message',
  INDEX idx_host (`host`) USING INVERTED COMMENT '',
  INDEX idx_path (`path`) USING INVERTED COMMENT '',
  INDEX idx_type (`type`) USING INVERTED COMMENT '',
  INDEX idx_level (`level`) USING INVERTED COMMENT '',
  INDEX idx_thread (`thread`) USING INVERTED COMMENT '',
  INDEX idx_position (`position`) USING INVERTED COMMENT '',
  INDEX idx_message (`message`) USING INVERTED PROPERTIES("parser" = "unicode", "support_phrase" = "true") COMMENT ''
) ENGINE=OLAP
DUPLICATE KEY(`log_time`)
COMMENT 'OLAP'
PARTITION BY RANGE(`log_time`) ()
DISTRIBUTED BY RANDOM BUCKETS 10
PROPERTIES (
"replication_num" = "1",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.start" = "-7",
"dynamic_partition.end" = "1",
"dynamic_partition.prefix" = "p",
"dynamic_partition.buckets" = "10",
"dynamic_partition.create_history_partition" = "true",
"compaction_policy" = "time_series"
);
```

3. Configuration

The filebeat log collection configuration file, such as filebeat_doris_log.yml, is in YAML format and mainly consists of four parts corresponding to the various stages of ETL: 1. Input is responsible for reading the raw data. 2. Processor is responsible for data transformation. 3. queue.mem configures the internal buffer queue of filebeat. 4. Output is responsible for sending the data to the output destination.

```
# 1. input is responsible for reading raw data
# type: log is a log input plugin that can be configured to read the path of the log file. It uses the multiline feature to
concatenate lines that do not start with a timestamp to the end of the previous line, achieving the effect of merging stacktraces
with the main log. The log input saves the log content in the message field, and there are also some metadata fields such as
agent.host, log.file.path.

filebeat.inputs:
- type: log
   enabled: true
   paths:
     - /path/to/your/log
   # multiline can concatenate multi-line logs (e.g., Java stacktraces)
   multiline:
     type: pattern
     # Effect: Lines starting with yyyy-mm-dd HH:MM:SS are considered as a new log, others are concatenated to the previous log
     pattern: '^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}'
     negate: true
     match: after
     skip_newline: true
```

```
# 2. processors section is responsible for data transformation
processors:
# Use the js script plugin to replace \t in logs with spaces to avoid JSON parsing errors
- script:
     lang: javascript
     source: >
         function process(event) {
             var msg = event.Get("message");
             msg = msg.replace(/\t/g, "   ");
             event.Put("message", msg);
         }
# Use the dissect plugin for simple log parsing
- dissect:
     # Example log: 2024-06-08 18:26:25,481 INFO (report-thread|199) [ReportHandler.cpuReport():617] begin to handle
     tokenizer: "%{day} %{time} %{log_level} (%{thread}) [%{position}] %{content}"
     target_prefix: ""
     ignore_failure: true
     overwrite_keys: true

# 3. internal buffer Queue total count, flush batch size, flush interval
queue.mem:
   events: 1000000
   flush.min_events: 100000
   flush.timeout: 10s

# 4. output section is responsible for data output
# The doris output sends data to Doris using the Stream Load HTTP interface. The data format for Stream Load is specified as JSON through the headers parameter, and the codec_format_string parameter formats the output to Doris in a printf-like manner. For example, the following example formats a JSON based on filebeat internal fields such as agent.hostname, and fields produced by processors like dissect, such as day, using %{[a][b]} to reference them. Stream Load will automatically write the JSON fields into the corresponding fields of the Doris table.

output.doris:
   fenodes: [ "http://fehost1:http_port", "http://fehost2:http_port", "http://fehost3:http_port" ]
   user: "your_username"
   password: "your_password"
   database: "your_db"
   table: "your_table"
   # Output string format
   ## %{[agent][hostname]} %{[log][file][path]} are filebeat自带的metadata
   ## Common filebeat metadata also includes采集时间戳 %{[@timestamp]}
   ## %{[day]} %{[time]} are fields obtained from the above dissect parsing
   codec_format_string: '{"ts": "%{[day]} %{[time]}", "host": "%{[agent][hostname]}", "path": "%{[log][file][path]}", "message": "%{[message]}" }'
   headers:
     format: "json"
     read_json_by_line: "true"
     load_to_single_tablet: "true"
```

4. Running filebeat

```
./filebeat-doris -f config/filebeat_doris_log.yml

# When log_request is set to true, the log will output the request parameters and response results of each Stream Load.

doris stream load response:
{
    "TxnId": 45464,
    "Label": "logstash_log_db_doris_log_20240708_223532_539_6c20a0d1-dcab-4b8e-9bc0-76b46a929bd1",
    "Comment": "",
    "TwoPhaseCommit": "false",
    "Status": "Success",
```

```
    "Message": "OK",
    "NumberTotalRows": 452,
    "NumberLoadedRows": 452,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 277230,
    "LoadTimeMs": 1797,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 18,
    "ReadDataTimeMs": 9,
    "WriteDataTimeMs": 1758,
    "CommitAndPublishTimeMs": 18
}

# By default, speed information is logged every 10 seconds, including the amount of data since startup (in MB and ROWS), the total
speed (in MB/s and R/S), and the speed in the last 10 seconds.

total 11 MB 18978 ROWS, total speed 0 MB/s 632 R/s, last 10 seconds speed 1 MB/s 1897 R/s
```

JSON Log Collection Example

This example demonstrates JSON log collection using data from the GitHub events archive.

1. Data

The GitHub events archive contains archived data of GitHub user actions, formatted as JSON. It can be downloaded from here, for example, the data for January 1, 2024, at 3 PM.

```
wget https://data.gharchive.org/2024-01-01-15.json.gz
```

Below is a sample of the data. Normally, each piece of data is on a single line, but for ease of display, it has been formatted here.

```
{
  "id": "37066529221",
  "type": "PushEvent",
  "actor": {
    "id": 46139131,
    "login": "Bard89",
    "display_login": "Bard89",
    "gravatar_id": "",
    "url": "https://api.github.com/users/Bard89",
    "avatar_url": "https://avatars.githubusercontent.com/u/46139131?"
  },
  "repo": {
    "id": 780125623,
    "name": "Bard89/talk-to-me",
    "url": "https://api.github.com/repos/Bard89/talk-to-me"
  },
  "payload": {
    "repository_id": 780125623,
    "push_id": 17799451992,
    "size": 1,
    "distinct_size": 1,
    "ref": "refs/heads/add_mvcs",
    "head": "f03baa2de66f88f5f1754ce3fa30972667f87e81",
    "before": "85e6544ede4ae3f132fe2f5f1ce0ce35a3169d21"
  },
  "public": true,
```

```
  "created_at": "2024-04-01T23:00:00Z"
}
```

2. Table Creation

```
CREATE DATABASE log_db;
USE log_db;


CREATE TABLE github_events
(
  `created_at` DATETIME,
  `id` BIGINT,
  `type` TEXT,
  `public` BOOLEAN,
  `actor` VARIANT,
  `repo` VARIANT,
  `payload` TEXT,
  INDEX `idx_id` (`id`) USING INVERTED,
  INDEX `idx_type` (`type`) USING INVERTED,
  INDEX `idx_actor` (`actor`) USING INVERTED,
  INDEX `idx_host` (`repo`) USING INVERTED,
  INDEX `idx_payload` (`payload`) USING INVERTED PROPERTIES("parser" = "unicode", "support_phrase" = "true")
)
ENGINE = OLAP
DUPLICATE KEY(`created_at`)
PARTITION BY RANGE(`created_at`) ()
DISTRIBUTED BY RANDOM BUCKETS 10
PROPERTIES (
"replication_num" = "1",
"compaction_policy" = "time_series",
"enable_single_replica_compaction" = "true",
"dynamic_partition.enable" = "true",
"dynamic_partition.create_history_partition" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.start" = "-30",
"dynamic_partition.end" = "1",
"dynamic_partition.prefix" = "p",
"dynamic_partition.buckets" = "10",
"dynamic_partition.replication_num" = "1"
);
```

3. Filebeat Configuration

This configuration file differs from the previous TEXT log collection in the following aspects:

1. Processors are not used because no additional processing or transformation is needed.
2. The codec_format_string in the output is simple, directly outputting the entire message, which is the raw content.

```
# input
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /path/to/your/log

# queue and batch
queue.mem:
  events: 1000000
```

```
  flush.min_events: 100000
  flush.timeout: 10s

# output
output.doris:
  fenodes: [ "http://fehost1:http_port", "http://fehost2:http_port", "http://fehost3:http_port" ]
  user: "your_username"
  password: "your_password"
  database: "your_db"
  table: "your_table"
  # output string format
  ## Directly outputting the raw message of each line from the original file. Since headers specify format: "json", Stream Load will
automatically parse the JSON fields and write them into the corresponding fields of the Doris table.
  codec_format_string: '%{[message]}'
  headers:
    format: "json"
    read_json_by_line: "true"
    load_to_single_tablet: "true"
```

4. Running Filebeat

```
./filebeat-doris -f config/filebeat_github_events.yml
```

FluentBit

Fluent Bit is a fast log processor and forwarder that supports custom output plugins to write data into storage systems, with the Fluent Bit Doris output plugin being the one for outputting to Doris.

By invoking the Doris Stream Load HTTP interface, the Fluent Bit Doris output plugin writes data into Doris in real-time, offering capabilities such as multi-threaded concurrency, failure retries, custom Stream Load formats and parameters, and output write speed.

To use the Fluent Bit Doris output plugin, there are three main steps: 1. Download or compile the Fluent Bit binary program that includes the Doris output plugin. 2. Configure the Fluent Bit output address and other parameters. 3. Start Fluent Bit to write data into Doris in real-time.

Installation (alpha)

Download from the Official Website

https://download.velodb.io/integrations/fluent-bit-doris-3.1.9

Compile from Source Code

Clone the dev branch of https://github.com/joker-star-l/fluent-bit and run the following commands in the build/ directory

```
cmake -DFLB_RELEASE=ON ..
make
```

The build output is build/bin/fluent-bit.

Configuration

The configuration for the Fluent Bit Doris output plugin is as follows:

Configuration	Description
`host`	Stream Load HTTP host
`port`	Stream Load HTTP port
`user`	Doris username, this user needs to have import permissions for the corresponding Doris database and table
`password`	Doris user's password
`database`	The Doris database name to write into
`table`	The Doris table name to write into

Configuration	Description
label_prefix	Doris Stream Load Label prefix，the final generated Label is *{label_prefix}_{timestamp}_{uuid}*，the default value is fluentbit. If set to false, no Label will be added
time_key	The name of the timestamp column to add to the data. The default value is date. If set to false, the column will not be added
header	Doris Stream Load headers parameter, can be set more than one
log_request	Whether to output Doris Stream Load request and response metadata in logs for troubleshooting, default is true
log_progress_interval	Time interval for outputting speed in logs, unit is seconds, default is 10, setting to 0 can disable this type of logging
retry_limit	Doris Stream Load request failure retry number, default value is 1, if set to false will not limit the number of retries
workers	Number of workers to perform Doris Stream Load, default value is 2

Usage Example

TEXT Log Collection Example

This example demonstrates TEXT log collection using Doris FE logs as an example.

1. Data

FE log files are typically located at the fe/log/fe.log file under the Doris installation directory. They are typical Java program logs, including fields such as timestamp, log level, thread name, code location, and log content. Not only do they contain normal logs, but also exception logs with stacktraces, which are multiline. Log collection and storage need to combine the main log and stacktrace into a single log entry.

```
2024-07-08 21:18:01,432 INFO (Statistics Job Appender|61) [StatisticsJobAppender.runAfterCatalogReady():70] Stats table not
available, skip
2024-07-08 21:18:53,710 WARN (STATS_FETCH-0|208) [StmtExecutor.executeInternalQuery():3332] Failed to run internal SQL:
OriginStatement{originStmt='SELECT * FROM __internal_schema.column_statistics WHERE part_id is NULL  ORDER BY update_time DESC LIMIT
500000', idx=0}
org.apache.doris.common.UserException: errCode = 2, detailMessage = tablet 10031 has no queryable replicas. err: replica 10032's
backend 10008 does not exist or not alive
        at org.apache.doris.planner.OlapScanNode.addScanRangeLocations(OlapScanNode.java:931) ~[doris-fe.jar:1.2-SNAPSHOT]
        at org.apache.doris.planner.OlapScanNode.computeTabletInfo(OlapScanNode.java:1197) ~[doris-fe.jar:1.2-SNAPSHOT]
```

2. Table Creation

The table structure includes fields such as the log's creation time, collection time, hostname, log file path, log type, log level, thread name, code location, and log content.

```
CREATE TABLE `doris_log` (
  `log_time` datetime NULL COMMENT 'log content time',
  `collect_time` datetime NULL COMMENT 'log agent collect time',
  `host` text NULL COMMENT 'hostname or ip',
  `path` text NULL COMMENT 'log file path',
  `type` text NULL COMMENT 'log type',
```

```
  `level` text NULL COMMENT 'log level',
  `thread` text NULL COMMENT 'log thread',
  `position` text NULL COMMENT 'log code position',
  `message` text NULL COMMENT 'log message',
  INDEX idx_host (`host`) USING INVERTED COMMENT '',
  INDEX idx_path (`path`) USING INVERTED COMMENT '',
  INDEX idx_type (`type`) USING INVERTED COMMENT '',
  INDEX idx_level (`level`) USING INVERTED COMMENT '',
  INDEX idx_thread (`thread`) USING INVERTED COMMENT '',
  INDEX idx_position (`position`) USING INVERTED COMMENT '',
  INDEX idx_message (`message`) USING INVERTED PROPERTIES("parser" = "unicode", "support_phrase" = "true") COMMENT ''
) ENGINE=OLAP
DUPLICATE KEY(`log_time`)
COMMENT 'OLAP'
PARTITION BY RANGE(`log_time`) ()
DISTRIBUTED BY RANDOM BUCKETS 10
PROPERTIES (
"replication_num" = "1",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.start" = "-7",
"dynamic_partition.end" = "1",
"dynamic_partition.prefix" = "p",
"dynamic_partition.buckets" = "10",
"dynamic_partition.create_history_partition" = "true",
"compaction_policy" = "time_series"
);
```

3. Configuration

The configuration file of Fluent Bit log collection is as follows, doris_log.conf is used to define various parts of ETL components, and parsers.conf is used to define different log parsers.

doris_log.conf:

```
# config for Fluent Bit service
[SERVICE]
    log_level info
    # parsers file
    parsers_file parsers.conf

# use input tail
[INPUT]
    name tail
    path /path/to/your/log
    # add log file name to the record, key is 'path'
    path_key path
    # set multiline parser
    multiline.parser multiline_java

# parse log
[FILTER]
    match *
    name parser
    key_name log
    parser fe_log
    reserve_data true

# add host info
[FILTER]
    name sysinfo
    match *
```

```
    # add hostname to the record, key is 'host'
    hostname_key host

# output to doris
[OUTPUT]
    name doris
    match *
    host fehost
    port feport
    user your_username
    password your_password
    database your_db
    table your_table
    # add 'collect_time' to the record
    time_key collect_time
    # 'collect_time' is timestamp, change it to datatime
    header columns collect_time=from_unixtime(collect_time)
    log_request true
    log_progress_interval 10
```

parsers.conf:

```
[MULTILINE_PARSER]
    name          multiline_java
    type          regex
    flush_timeout 1000
    # Regex rules for multiline parsing
    # ---------------------------------
    #
    # configuration hints:
    #
    #  - first state always has the name: start_state
    #  - every field in the rule must be inside double quotes
    #
    # rules   |   state name   | regex pattern | next state name
    # --------|----------------|---------------|-----------------
    rule         "start_state"   "/(^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2})(.*)/"  "cont"
    rule         "cont"          "/(^(?![0-9]{4}-[0-9]{2}-[0-9]{2}))(.*)/"     "cont"


[PARSER]
    name        fe_log
    format      regex
    # parse and add 'log_time', 'level', 'thread', 'position', 'message' to the record
    regex       ^(?<log_time>[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2},[0-9]{3}) (?<level>[^ ]+) \((?<thread>[^\)]+)\) \
[(?<position>[^\]]+)\] (?<message>(\n|.)*)\n$
```

4. Running Fluent Bit

```
fluent-bit -c doris_log.conf

# log stream load response

[2024/10/31 18:39:55] [ info] [output:doris:doris.1] 127.0.0.1:8040, HTTP status=200
{
    "TxnId": 32155,
    "Label": "fluentbit_1730371195_91cca1aa-c15f-45d2-b503-fe7d2e839c2a",
    "Comment": "",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 1,
```

```
    "NumberLoadedRows": 1,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 836,
    "LoadTimeMs": 298,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 3,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 268,
    "CommitAndPublishTimeMs": 25
}

# log speed info

[2024/10/31 18:40:13] [ info] [output:doris:doris.1] total 0 MB 2 ROWS, total speed 0 MB/s 0 R/s, last 10 seconds speed 0 MB/s 0 R/s
```

JSON Log Collection Example

This example demonstrates JSON log collection using data from the GitHub events archive.

1. Data

The GitHub events archive contains archived data of GitHub user actions, formatted as JSON. It can be downloaded from here, for example, the data for January 1, 2024, at 3 PM.

```
wget https://data.gharchive.org/2024-01-01-15.json.gz
```

Below is a sample of the data. Normally, each piece of data is on a single line, but for ease of display, it has been formatted here.

```
{
  "id": "37066529221",
  "type": "PushEvent",
  "actor": {
    "id": 46139131,
    "login": "Bard89",
    "display_login": "Bard89",
    "gravatar_id": "",
    "url": "https://api.github.com/users/Bard89",
    "avatar_url": "https://avatars.githubusercontent.com/u/46139131?"
  },
  "repo": {
    "id": 780125623,
    "name": "Bard89/talk-to-me",
    "url": "https://api.github.com/repos/Bard89/talk-to-me"
  },
  "payload": {
    "repository_id": 780125623,
    "push_id": 17799451992,
    "size": 1,
    "distinct_size": 1,
    "ref": "refs/heads/add_mvcs",
    "head": "f03baa2de66f88f5f1754ce3fa30972667f87e81",
    "before": "85e6544ede4ae3f132fe2f5f1ce0ce35a3169d21"
  },
  "public": true,
  "created_at": "2024-04-01T23:00:00Z"
}
```

2. Table Creation

```
CREATE DATABASE log_db;
USE log_db;


CREATE TABLE github_events
(
  `created_at` DATETIME,
  `id` BIGINT,
  `type` TEXT,
  `public` BOOLEAN,
  `actor` VARIANT,
  `repo` VARIANT,
  `payload` TEXT,
  INDEX `idx_id` (`id`) USING INVERTED,
  INDEX `idx_type` (`type`) USING INVERTED,
  INDEX `idx_actor` (`actor`) USING INVERTED,
  INDEX `idx_host` (`repo`) USING INVERTED,
  INDEX `idx_payload` (`payload`) USING INVERTED PROPERTIES("parser" = "unicode", "support_phrase" = "true")
)
ENGINE = OLAP
DUPLICATE KEY(`created_at`)
PARTITION BY RANGE(`created_at`) ()
DISTRIBUTED BY RANDOM BUCKETS 10
PROPERTIES (
"replication_num" = "1",
"compaction_policy" = "time_series",
"enable_single_replica_compaction" = "true",
"dynamic_partition.enable" = "true",
"dynamic_partition.create_history_partition" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.start" = "-30",
"dynamic_partition.end" = "1",
"dynamic_partition.prefix" = "p",
"dynamic_partition.buckets" = "10",
"dynamic_partition.replication_num" = "1"
);
```

3. Configuration

In contrast to the previous TEXT log collection, this configuration does not use FILTER because no additional processing transformations are required.

github_events.conf:

```
[SERVICE]
    log_level info
    parsers_file github_parsers.conf

[INPUT]
    name tail
    parser github
    path /path/to/your/log

[OUTPUT]
    name doris
    match *
    host fehost
    port feport
    user your_username
```

```
    password your_password
    database your_db
    table your_table
    time_key false
    log_request true
    log_progress_interval 10
```

github_parsers.conf:

```
[PARSER]
    name github
    format json
```

4. Running Fluent Bit

```
fluent-bit -c github_events.conf
```

BladePipe

BladePipe is a **real-time end-to-end data replication** tool, moving data between **30+** databases, message queues, search engines, caching, real-time data warehouses, data lakes and more, with **ultra-low latency less than 3 seconds**. It features efficiency, stability and scalability, compatibility with diverse database engines, one-stop management, enhanced security, and complex data transformation.

Functions

BladePipe presents a visual management interface, allowing you to easily create DataJobs to realize **schema migration, data migration, synchronization, verification and correction**, etc. In addition, more refined and customized configurations are supported by setting parameters. Now BladePipe supports data movement from the following source DataSources to Doris:

Source DataSource	Schema Migration	Data Migration	Data Sync	Verification & Correction
MySQL/MariaDB/AuroraMySQL	Yes	Yes	Yes	Yes
Oracle	Yes	Yes	Yes	Yes
PostgreSQL/AuroraPostgreSQL	Yes	Yes	Yes	Yes
SQL Server	Yes	Yes	Yes	Yes
Kafka	No	No	Yes	No
AutoMQ	No	No	Yes	No
TiDB	Yes	Yes	Yes	Yes
Hana	Yes	Yes	Yes	Yes
PolarDB-X	Yes	Yes	Yes	Yes

> **For more functions and parameter settings, please refer to BladePipe Connections.**

Installation

Follow the instructions in Install Worker (Docker) or Install Worker (Binary) to download and install a BladePipe Worker.

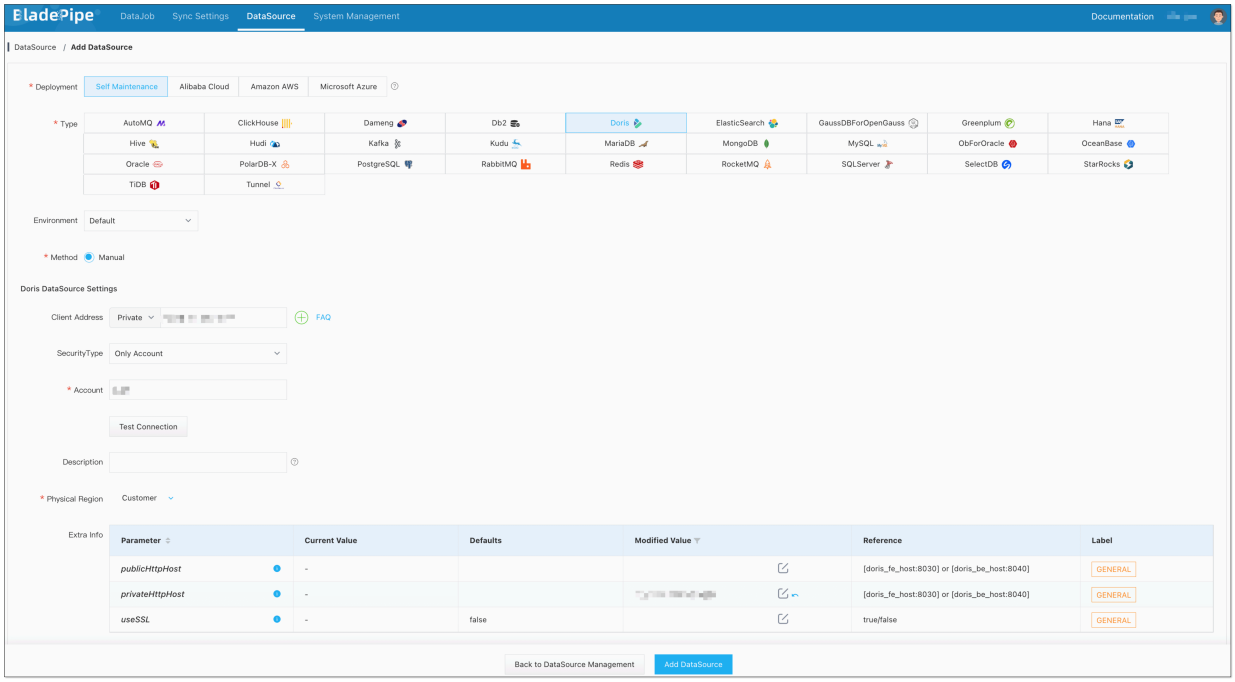
Example

Taking a MySQL instance as an example, the following part describes how to move data from MySQL to Doris.

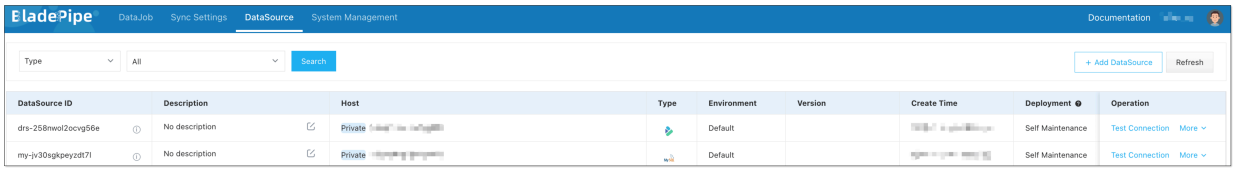
Add DataSources

1. Log in to the BladePipe Cloud. Click **DataSource** > **Add DataSource**.

2. Select MySQL and Doris as the Type respectively, and fill in the setup form accordingly.



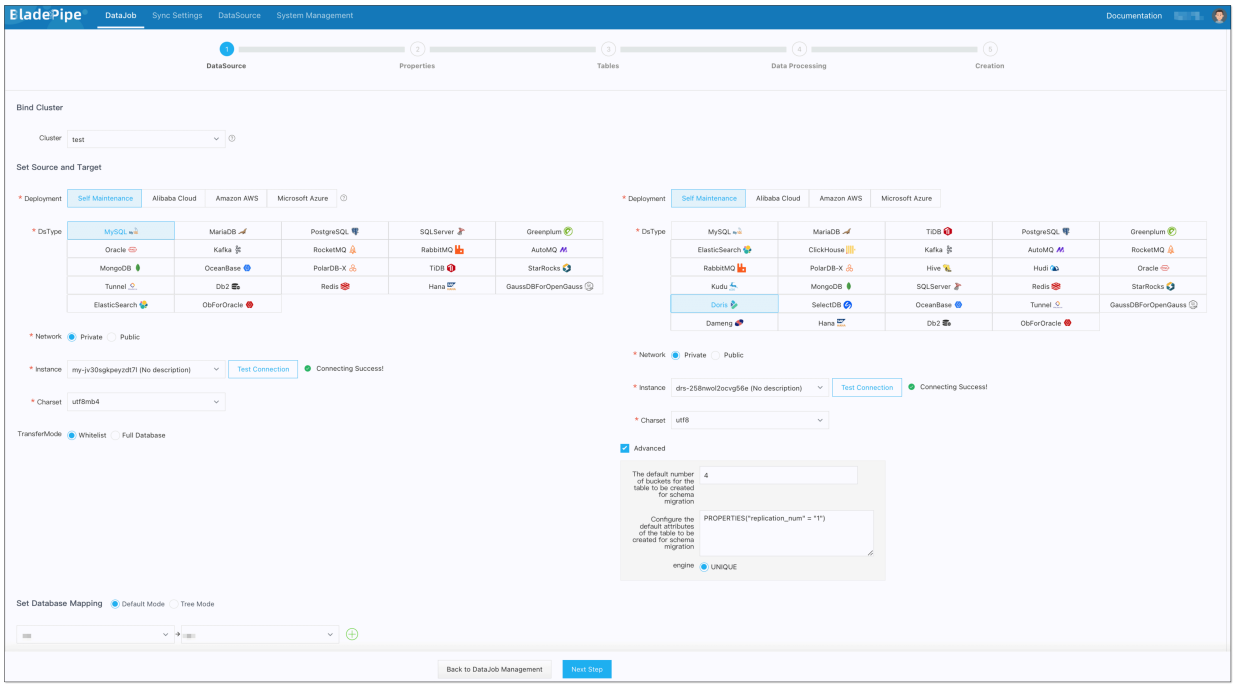
3. Click **Test Connection**. After successful connection, click **Add DataSource** to add the DataSource.



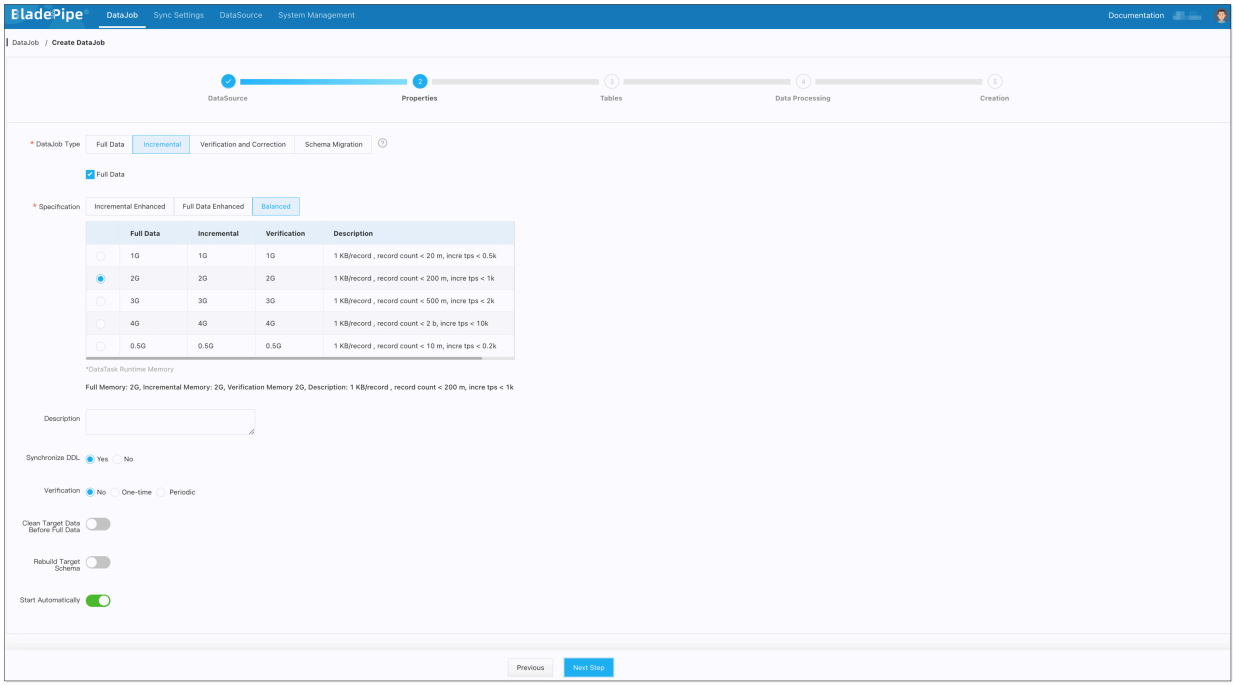
Create a DataJob

1. Click DataJob > Create DataJob.

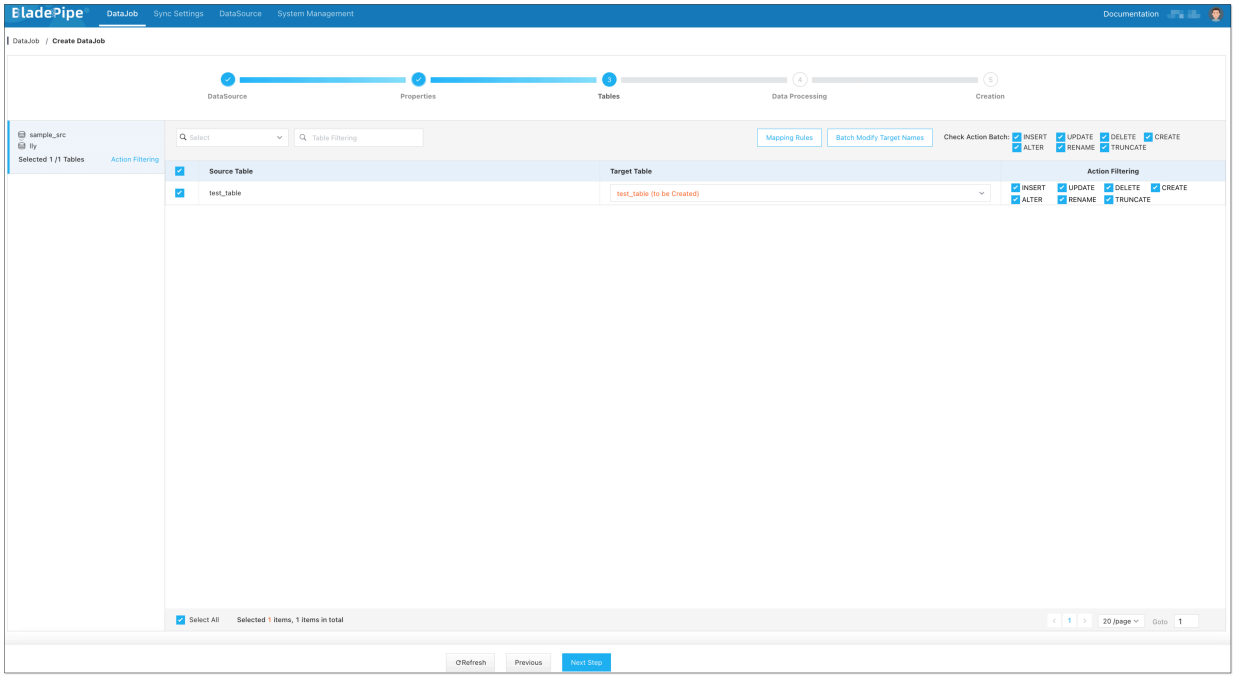
2. Select the source and target DataSources, and click **Test Connection** to ensure the connection to the source and target DataSources are both successful.



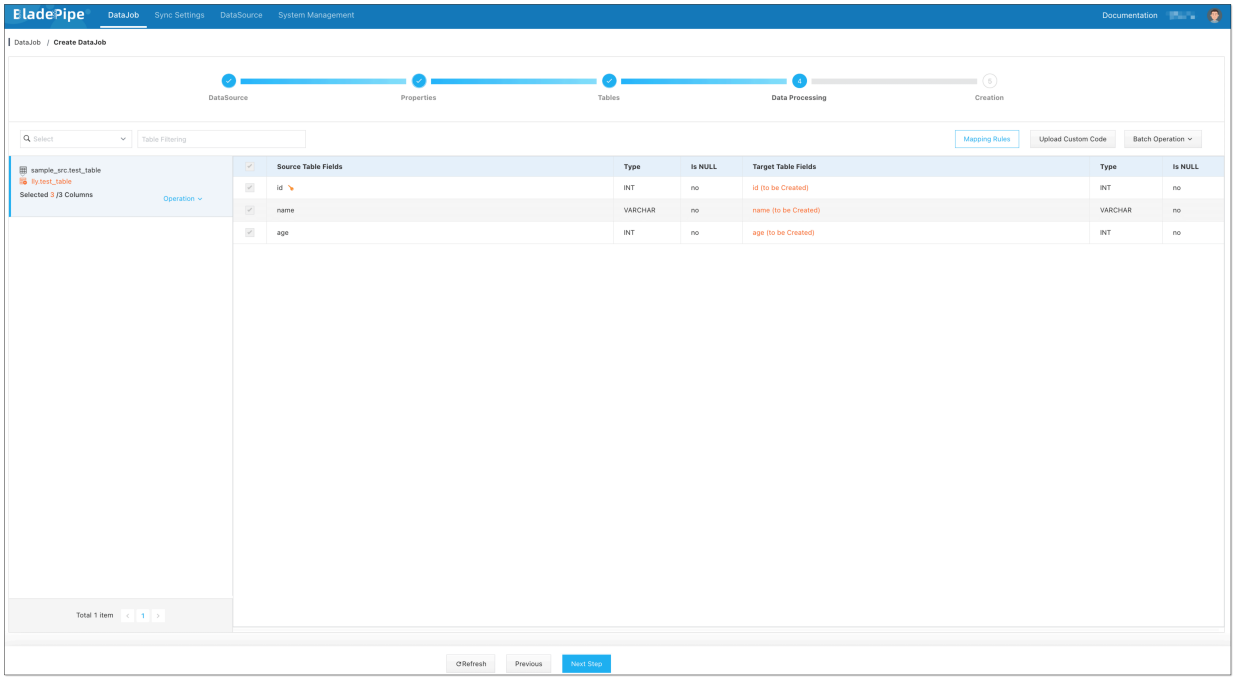
3. Select **Incremental** for DataJob Type, together with the **Full Data** option.



4. Select the tables to be replicated.



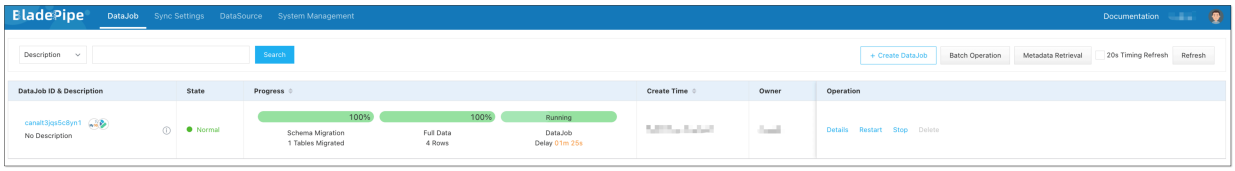
5. Select the columns to be replicated.



6. Confirm the DataJob creation.

7. The DataJob runs automatically. BladePipe will automatically run the following DataTasks:

- **Schema Migration**: The schemas of the source tables will be migrated to the target instance.
- **Full Data**: All existing data of the source tables will be fully migrated to the target instance.
- **Incremental**: Ongoing data changes will be continuously synchronized to the target instance.



DataX Doriswriter

DataX Doriswriter

The DataX Doriswriter plugin supports synchronizing data from various data sources, such as MySQL, Oracle, and SQL Server, into Doris using the Stream Load method.

> **Note**
>
> This plugin needs to be used together with the DataX service. DataX supports multiple data sources. For more details, see here.

Usage

Directly Download the DataX Installation Package

DataX provides an official installation package that already includes DataX, which can be downloaded and used directly. For more details, refer to here.

Compile the DorisWriter Plugin Manually

Download the source code for the DorisWriter plugin.

1. Run `init-env.sh`
2. Modify code of doriswriter in `DataX/doriswriter` if you need.
3. Build doriswriter

> Build doriswriter along:

```
`mvn clean install -pl plugin-rdbms-util,doriswriter -DskipTests`
```

> If you need to compile the entire DataX project, please refer to here

> Compilation error

```
If you encounter the following compilation errors:
```

```
Could not find artifact com.alibaba.datax:datax-all:pom:0.0.1-SNAPSHOT ...
```

You can try the following solutions:

1. Download [alibaba-datax-maven-m2-20210928.tar.gz](https://doris-thirdparty-repo.bj.bcebos.com/thirdparty/alibaba-datax-maven-m2-20210928.tar.gz)
2. After decompression, copy the resulting `alibaba/datax/` directory to `.m2/repository/com/alibaba/` corresponding to the maven used, and try to compile again.

Datax DorisWriter parameter introduction:

- **jdbcUrl**
  - Description: Doris's JDBC connection string, the user executes preSql or postSQL.
  - Mandatory: Yes
  - Default: None
- **loadUrl**
  - Description: As a connection target for Stream Load. The format is "ip:port". Where IP is the FE node IP, port is the http_port of the FE node. You can fill in more than one, separated by commas in English: ,, doriswriter will visit in a polling manner.
  - Mandatory: Yes
  - Default: None
- **username**
  - Description: The username to access the Doris database
  - Mandatory: Yes
  - Default: None
- **password**
  - Description: Password to access Doris database
  - Mandatory: No
  - Default: empty
- **connection.selectedDatabase**
  - Description: The name of the Doris database that needs to be written.
  - Mandatory: Yes
  - Default: None
- **connection. table**
  - Description: The name of the Doris table that needs to be written.
    - Mandatory: Yes
    - Default: None
- **flushInterval**

    - Description: The time interval at which data is written in batches. If this time interval is set too small, it will cause Doris write blocking problem, error code -235, and if you set this time interval too small, `maxBatchRows` and `batchSize` parameters are set too large, then it may not be able to reach you The data size set by this will also be imported.
    - Mandatory: No
    - Default: 30000 (ms)

- **column**

    - Description: The fields that the destination table needs to write data into, these fields will be used as the field names of the generated Json data. Fields are separated by commas. For example: “column”: [“id”,“name”,“age”].
    - Mandatory: Yes
    - Default: No

- **preSql**

    - Description: Before writing data to the destination table, the standard statement here will be executed first.
    - Mandatory: No
    - Default: None

- **postSql**

    - Description: After writing data to the destination table, the standard statement here will be executed.
    - Mandatory: No
    - Default: None

- **maxBatchRows**

    - Description: The maximum number of rows for each batch of imported data. Together with **batchSize**, it controls the number of imported record rows per batch. When each batch of data reaches one of the two thresholds, the data of this batch will start to be imported.
    - Mandatory: No
    - Default: 500000

- **batchSize**

    - Description: The maximum amount of data imported in each batch. Works with **maxBatchRows** to control the number of imports per batch. When each batch of data reaches one of the two thresholds, the data of this batch will start to be imported.
    - Mandatory: No
    - Default: 94371840

- **maxRetries**

    - Description: The number of retries after each batch of failed data imports.
    - Mandatory: No
    - Default: 3

- **labelPrefix**

    - Description: The label prefix for each batch of imported tasks. The final label will have `labelPrefix + UUID` to form a globally unique label to ensure that data will not be imported repeatedly
    - Mandatory: No
    - Default: `datax_doris_writer_`

- **loadProps**

    - Description: The request parameter of StreamLoad. For details, refer to the StreamLoad introduction page. Stream load - Apache Doris

      This includes the imported data format: format, etc. The imported data format defaults to csv, which supports JSON. For details, please refer to the type conversion section below, or refer to the official information of Stream load above.

    - Mandatory: No

    - Default: None

Example

1. Stream reads the data and imports it to Doris

For instructions on using the doriswriter plug-in, please refer to here.

2.Mysql reads the data and imports it to Doris

1.Mysql table structure

```
CREATE TABLE `t_test`(
 `id`bigint(30) NOT NULL,
 `order_code` varchar(30) DEFAULT NULL COMMENT '',
 `line_code` varchar(30) DEFAULT NULL COMMENT '',
 `remark` varchar(30) DEFAULT NULL COMMENT '',
 `unit_no` varchar(30) DEFAULT NULL COMMENT '',
 `unit_name` varchar(30) DEFAULT NULL COMMENT '',
 `price` decimal(12,2) DEFAULT NULL COMMENT '',
 PRIMARY KEY(`id`) USING BTREE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=DYNAMIC COMMENT='';
```

2.Doris table structure

```
CREATE TABLE `ods_t_test` (
 `id`bigint(30) NOT NULL,
 `order_code` varchar(30) DEFAULT NULL COMMENT '',
 `line_code` varchar(30) DEFAULT NULL COMMENT '',
 `remark` varchar(30) DEFAULT NULL COMMENT '',
 `unit_no` varchar(30) DEFAULT NULL COMMENT '',
 `unit_name` varchar(30) DEFAULT NULL COMMENT '',
 `price` decimal(12,2) DEFAULT NULL COMMENT ''
) ENGINE=OLAP
UNIQUE KEY(id`, `order_code`)
DISTRIBUTED BY HASH(`order_code`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3",
```

```
"in_memory" = "false",
"storage_format" = "V2"
);
```

3.Create datax script

my_import.json

```
{
    "job": {
        "content": [
            {
                "reader": {
                    "name": "mysqlreader",
                    "parameter": {
                        "column": ["id","order_code","line_code","remark","unit_no","unit_name","price"],
                        "connection": [
                            {
                                "jdbcUrl": ["jdbc:mysql://localhost:3306/demo"],
                                "table": ["employees_1"]
                            }
                        ],
                        "username": "root",
                        "password": "xxxxx",
                        "where": ""
                    }
                },
                "writer": {
                    "name": "doriswriter",
                    "parameter": {
                        "loadUrl": ["127.0.0.1:8030"],
                        "column": ["id","order_code","line_code","remark","unit_no","unit_name","price"],
                        "username": "root",
                        "password": "xxxxxx",
                        "postSql": ["select count(1) from all_employees_info"],
                        "preSql": [],
                        "flushInterval":30000,
                        "connection": [
                          {
                            "jdbcUrl": "jdbc:mysql://127.0.0.1:9030/demo",
                            "selectedDatabase": "demo",
                            "table": ["all_employees_info"]
                          }
                        ],
                        "loadProps": {
                            "format": "json",
                            "strip_outer_array":"true",
                            "line_delimiter": "\\x02"
                        }
                    }
                }
            }
        ],
        "setting": {
            "speed": {
                "channel": "1"
            }
        }
    }
}
```

Remark:

```
"loadProps": {
  "format": "json",
  "strip_outer_array": "true",
  "line_delimiter": "\\x02"
}
```

1. Here we use JSON format to import data
2. `line_delimiter` defaults to a newline character, which may conflict with the value in the data, we can use some special characters or invisible characters to avoid import errors
3. strip_outer_array : Represents multiple rows of data in a batch of imported data. Doris will expand the array when parsing, and then parse each Object in it as a row of data in turn.
4. For more parameters of Stream load, please refer to Stream load - Apache Doris
5. If it is in CSV format, we can use it like this

```
"loadProps": {
   "format": "csv",
   "column_separator": "\\x01",
   "line_delimiter": "\\x02"
}
```

**CSV format should pay special attention to row and column separators to avoid conflicts with special characters in the data. Hidden characters are recommended here. The default column separator is: row separator: *

4.Execute the datax task, refer to the specific datax official website

```
python bin/datax.py my_import.json
```

After execution, we can see the following information

```
2022-11-16 14:28:54.012 [job-0] INFO  JobContainer - jobContainer starts to do prepare ...
2022-11-16 14:28:54.012 [job-0] INFO  JobContainer - DataX Reader.Job [mysqlreader] do prepare work .
2022-11-16 14:28:54.013 [job-0] INFO  JobContainer - DataX Writer.Job [doriswriter] do prepare work .
2022-11-16 14:28:54.020 [job-0] INFO  JobContainer - jobContainer starts to do split ...
2022-11-16 14:28:54.020 [job-0] INFO  JobContainer - Job set Channel-Number to 1 channels.
2022-11-16 14:28:54.023 [job-0] INFO  JobContainer - DataX Reader.Job [mysqlreader] splits to [1] tasks.
2022-11-16 14:28:54.023 [job-0] INFO  JobContainer - DataX Writer.Job [doriswriter] splits to [1] tasks.
2022-11-16 14:28:54.033 [job-0] INFO  JobContainer - jobContainer starts to do schedule ...
2022-11-16 14:28:54.036 [job-0] INFO  JobContainer - Scheduler starts [1] taskGroups.
2022-11-16 14:28:54.037 [job-0] INFO  JobContainer - Running by standalone Mode.
2022-11-16 14:28:54.041 [taskGroup-0] INFO  TaskGroupContainer - taskGroupId=[0] start [1] channels for [1] tasks.
2022-11-16 14:28:54.043 [taskGroup-0] INFO  Channel - Channel set byte_speed_limit to -1, No bps activated.
2022-11-16 14:28:54.043 [taskGroup-0] INFO  Channel - Channel set record_speed_limit to -1, No tps activated.
2022-11-16 14:28:54.049 [taskGroup-0] INFO  TaskGroupContainer - taskGroup[0] taskId[0] attemptCount[1] is started
2022-11-16 14:28:54.052 [0-0-0-reader] INFO  CommonRdbmsReader$Task - Begin to read record by Sql: [select
taskid,projectid,taskflowid,templateid,template_name,status_task from dwd_universal_tb_task
] jdbcUrl:[jdbc:mysql://localhost:3306/demo?
yearIsDateType=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&rewriteBatchedStatements=true].
Wed Nov 16 14:28:54 GMT+08:00 2022 WARN: Establishing SSL connection without server's identity verification is not recommended.
According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't
set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
```

```
to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
2022-11-16 14:28:54.071 [0-0-0-reader] INFO  CommonRdbmsReader$Task - Finished read record by Sql: [select
taskid,projectid,taskflowid,templateid,template_name,status_task from dwd_universal_tb_task
] jdbcUrl:[jdbc:mysql://localhost:3306/demo?
yearIsDateType=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&rewriteBatchedStatements=true].
2022-11-16 14:28:54.104 [Thread-1] INFO  DorisStreamLoadObserver - Start to join batch data: rows[2] bytes[438]
label[datax_doris_writer_c4e08cb9-c157-4689-932f-db34acc45b6f].
2022-11-16 14:28:54.104 [Thread-1] INFO  DorisStreamLoadObserver - Executing stream load to:
'http://127.0.0.1:8030/api/demo/dwd_universal_tb_task/_stream_load', size: '441'
2022-11-16 14:28:54.224 [Thread-1] INFO  DorisStreamLoadObserver - StreamLoad response :
{"Status":"Success","BeginTxnTimeMs":0,"Message":"OK","NumberUnselectedRows":0,"CommitAndPublishTimeMs":17,"Label":"datax_doris_writ
er_c4e08cb9-c157-4689-932f-
db34acc45b6f","LoadBytes":441,"StreamLoadPutTimeMs":1,"NumberTotalRows":2,"WriteDataTimeMs":11,"TxnId":217056,"LoadTimeMs":31,"TwoPh
aseCommit":"false","ReadDataTimeMs":0,"NumberLoadedRows":2,"NumberFilteredRows":0}
2022-11-16 14:28:54.225 [Thread-1] INFO  DorisWriterManager - Async stream load finished: label[datax_doris_writer_c4e08cb9-c157-
4689-932f-db34acc45b6f].
2022-11-16 14:28:54.249 [taskGroup-0] INFO  TaskGroupContainer - taskGroup[0] taskId[0] is successed, used[201]ms
2022-11-16 14:28:54.250 [taskGroup-0] INFO  TaskGroupContainer - taskGroup[0] completed it's tasks.
2022-11-16 14:29:04.048 [job-0] INFO  StandAloneJobContainerCommunicator - Total 2 records, 214 bytes | Speed 21B/s, 0 records/s |
Error 0 records, 0 bytes |  All Task WaitWriterTime 0.000s |  All Task WaitReaderTime 0.000s | Percentage 100.00%
2022-11-16 14:29:04.049 [job-0] INFO  AbstractScheduler - Scheduler accomplished all tasks.
2022-11-16 14:29:04.049 [job-0] INFO  JobContainer - DataX Writer.Job [doriswriter] do post work.
Wed Nov 16 14:29:04 GMT+08:00 2022 WARN: Establishing SSL connection without server's identity verification is not recommended.
According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't
set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either
to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
2022-11-16 14:29:04.187 [job-0] INFO  DorisWriter$Job - Start to execute preSqls:[select count(1) from dwd_universal_tb_task].
context info:jdbc:mysql://172.16.0.13:9030/demo.
2022-11-16 14:29:04.204 [job-0] INFO  JobContainer - DataX Reader.Job [mysqlreader] do post work.
2022-11-16 14:29:04.204 [job-0] INFO  JobContainer - DataX jobId [0] completed successfully.
2022-11-16 14:29:04.204 [job-0] INFO  HookInvoker - No hook invoked, because base dir not exists or is a file: /data/datax/hook
2022-11-16 14:29:04.205 [job-0] INFO  JobContainer -
         [total cpu info] =>
                averageCpu                     | maxDeltaCpu                    | minDeltaCpu
                -1.00%                         | -1.00%                         | -1.00%


         [total gc info] =>
                 NAME                 | totalGCCount       | maxDeltaGCCount    | minDeltaGCCount    | totalGCTime        |
maxDeltaGCTime     | minDeltaGCTime
                 PS MarkSweep         | 1                  | 1                  | 1                  | 0.017s             | 0.017s
| 0.017s
                 PS Scavenge          | 1                  | 1                  | 1                  | 0.007s             | 0.007s
| 0.007s

2022-11-16 14:29:04.205 [job-0] INFO  JobContainer - PerfTrace not enable!
2022-11-16 14:29:04.206 [job-0] INFO  StandAloneJobContainerCommunicator - Total 2 records, 214 bytes | Speed 21B/s, 0 records/s |
Error 0 records, 0 bytes |  All Task WaitWriterTime 0.000s |  All Task WaitReaderTime 0.000s | Percentage 100.00%
2022-11-16 14:29:04.206 [job-0] INFO  JobContainer -
Task Start Time                        : 2022-11-16 14:28:53
Task End Time                          : 2022-11-16 14:29:04
Total Task Duration                    : 10s
Average Task Throughput                : 21B/s
Record Write Speed                     : 0rec/s
Total Records Read                     : 2
Total Read/Write Failures              : 0
```

DBT Doris Adapter

DBT Doris Adapter

DBT(Data Build Tool) is a component that focuses on doing T (Transform) in ELT (extraction, loading, transformation) - the “transformation data” link The `dbt-doris` adapter is developed based on `dbt-core` 1.5.0 and relies on the `mysql-connector-python` driver to convert data to doris.

git: https://github.com/apache/doris/tree/master/extension/dbt-doris

version

doris	python	dbt-core
>=1.2.5	>=3.8,<=3.10	>=1.5.0

dbt-doris adapter Instructions

dbt-doris adapter install

use pip install:

```
pip install dbt-doris
```

check version:

```
dbt --version
```

if command not found: dbt:

```
ln -s /usr/local/python3/bin/dbt /usr/bin/dbt
```

dbt-doris adapter project init

```
dbt init
```

Users need to prepare the following information to init dbt project

name	default	meaning
project		project name

name	default	meaning
database		Enter the corresponding number to select the adapter
host		doris host
port	9030	doris MySQL Protocol Port
schema		In dbt-doris, it is equivalent to database, Database name
username		doris username
password		doris password
threads	1	Parallelism in dbt-doris (setting a parallelism that does not match the cluster capability will increase the risk of dbt running failure)

dbt-doris adapter run

For dbt run documentation, please refer to here. Go to the project directory and execute the default dbt model:

```
dbt run
```

model：`my_first_dbt_model`和`my_second_dbt_model`

They are materialized `table` and `view` respectively. then login to doris to view the data results and table creation statements of `my_first_dbt_model` and `my_second_dbt_model`. #### dbt-doris adapter Materialization dbt-doris Materialization support three: 1. view 2. table 3. incremental

View

Using `view` as the materialization, Models will be rebuilt as views each time they are run through the create view as statement. (By default, the materialization method of dbt is view)

```
Advantages: No extra data is stored, and views on top of the source data will always contain the latest records.
Disadvantages: View queries that perform large transformations or are nested on top of other views are slow.
Recommendation: Usually start with the view of the model and only change to another materialization if there are performance issues.
Views are best suited for models that do not undergo major transformations, such as renaming, column changes.
```

config：

```
models:
  <resource-path>:
    +materialized: view
```

Or write in the model file

```
{{ config(materialized = "view") }}
```

Table

When using the `table` materialization mode, your model is rebuilt as a table at each run with a `create table as select` statement. For the tablet materialization of dbt, dbt-doris uses the following steps to ensure the atomicity of data changes: 1. first create a temporary table: `create table this_table_temp as {{ model sql}}`. 2. Determine whether `this_table` does not exist, that is, it is created for the first time, execute `rename`, and change the temporary table to the final table. 3. if already exists, then `alter table this_table REPLACE WITH TABLE this_table_temp PROPERTIES('swap' = 'False')`, This operation can exchange the table name and delete the `this_table_temp` temporary table,this guarantees the atomicity of this operation through the transaction mechanism of the Doris.

```
Advantages: table query speed will be faster than view.
Disadvantages: The table takes a long time to build or rebuild, additional data will be stored, and incremental data synchronization cannot be performed.
Recommendation: It is recommended to use the table materialization method for models queried by BI tools or models with slow operations such as downstream queries and conversions.
```

config:

```
models:
  <resource-path>:
    +materialized: table
    +duplicate_key: [ <column-name>, ... ],
    +replication_num: int,
    +partition_by: [ <column-name>, ... ],
    +partition_type: <engine-type>,
    +partition_by_init: [<pertition-init>, ... ]
    +distributed_by: [ <column-name>, ... ],
    +buckets: int | 'auto',
    +properties: {<key>:<value>,...}
```

Or write in the model file:

```
{{ config(
    materialized = "table",
    duplicate_key = [ "<column-name>", ... ],
    replication_num = "<int>"
    partition_by = [ "<column-name>", ... ],
    partition_type = "<engine-type>",
    partition_by_init = ["<pertition-init>", ... ]
    distributed_by = [ "<column-name>", ... ],
    buckets = "<int>" | "auto",
    properties = {"<key>":"<value>",...}
      ...
    ]
) }}
```

The details of the above configuration items are as follows:

item	description	Required?
materialized	The materialized form of the table (Doris Duplicate table)	Required
duplicate_key	Doris Duplicate key	Optional
replication_num	Number of table replicas	Optional
partition_by	Table partition column	Optional
partition_type	Table partition type, range or list.(default: RANGE)	Optional
partition_by_init	Initialized table partitions	Optional
distributed_by	Table distributed column	Optional
buckets	Bucket size	Optional
properties	Doris table properties	Optional

Incremental

Based on the incremental model results of the last run of dbt, records are incrementally inserted or updated into the table. There are two ways to realize the increment of doris. incremental_strategy has two incremental strategies: * insert_overwrite: Depends on the doris unique model. If there is an incremental requirement, specify the materialization as incremental when initializing the data of the model, and aggregate by specifying the aggregation column to achieve incremental data coverage. * append: Depends on the doris duplicate model, it only appends incremental data and does not involve modifying any historical data. So no need to specify unique_key.

```
Advantages: Significantly reduces build time by only converting new records.
Disadvantages: incremental mode requires additional configuration, which is an advanced usage of dbt, and requires the support of complex scenarios and the adaptation of corresponding components.
Recommendation: The incremental model is best for event-based scenarios or when dbt runs become too slow
```

config:

```
models:
  <resource-path>:
    +materialized: incremental
    +incremental_strategy: <strategy>
    +unique_key: [ <column-name>, ... ],
    +replication_num: int,
    +partition_by: [ <column-name>, ... ],
    +partition_type: <engine-type>,
    +partition_by_init: [<pertition-init>, ... ]
    +distributed_by: [ <column-name>, ... ],
    +buckets: int | 'auto',
    +properties: {<key>:<value>,...}
```

Or write in the model file:

```
{{ config(
    materialized = "incremental",
    incremental_strategy = "<strategy>"
    unique_key = [ "<column-name>", ... ],
    replication_num = "<int>"
```

```
    partition_by = [ "<column-name>", ... ],
    partition_type = "<engine-type>",
    partition_by_init = ["<pertition-init>", ... ]
    distributed_by = [ "<column-name>", ... ],
    buckets = "<int>" | "auto",
    properties = {"<key>":"<value>",...}
      ...
    ]
) }}
```

The details of the above configuration items are as follows:

item	description	Required?
`materialized`	The materialized form of the table (Doris Duplicate/Unique table)	Required
`incremental_strategy`	Incremental_strategy	Optional
`unique_key`	Doris Unique key	Optional
`replication_num`	Number of table replicas	Optional
`partition_by`	Table partition column	Optional
`partition_type`	Table partition type, `range` or `list`.(default: `RANGE`)	Optional
`partition_by_init`	Initialized table partitions	Optional
`distributed_by`	Table distributed column	Optional
`buckets`	Bucket size	Optional
`properties`	Doris table properties	Optional

dbt-doris adapter seed

`seed` is a functional module used to load data files such as csv. It is a way to load files into the library and participate in model building, but there are the following precautions: 1. Seeds should not be used to load raw data (for example, large CSV exports from a production database). 2. Since seeds are version controlled, they are best suited to files that contain business-specific logic, for example a list of country codes or user IDs of employees. 3. Loading CSVs using dbt's seed functionality is not performant for large files. Consider using `streamload` to load these CSVs into doris.

Users can see the seeds directory under the dbt project directory, upload the csv file and seed configuration file in it and run

```
 dbt seed --select seed_name
```

Common seed configuration file writing method supports the definition of column types:

```
seeds:
  seed_name:
    config:
      schema: demo_seed
      full_refresh: true
      replication_num: 1
      column_types:
        id: bigint
```

```
        phone: varchar(32)
        ip: varchar(15)
        name: varchar(20)
        cost: DecimalV3(19,10)
```

Usage Examples

View Model Sample Reference

```
{{ config(materialized='view') }}

select
    u.user_id,
    max(o.create_time) as create_time,
    sum (o.cost) as balance
from {{ ref('sell_order') }} as o
left join {{ ref('sell_user') }} as u
on u.account_id=o.account_id
group by u.user_id
order by u.user_id
```

Table Model Sample Reference

```
{{ config(materialized='table') }}

select
    u.user_id,
    max(o.create_time) as create_time,
    sum (o.cost) as balance
from {{ ref('sell_order') }} as o
left join {{ ref('sell_user') }} as u
on u.account_id=o.account_id
group by u.user_id
order by u.user_id
```

Incremental model sample reference (duplicate mode)

Create a table in duplicate mode, without data aggregation, and without specifying unique_key

```
{{ config(
    materialized='incremental',
    replication_num=1
) }}

with source_data as (
    select
        *
    from {{ ref('sell_order2') }}
)

select * from source_data
```

Incremental model sample reference (unique mode)

Create a table in unique mode, data aggregation, unique_key must be specified

```
{{ config(
materialized='incremental',
unique_key=['account_id','create_time']
) }}

with source_data as (
    select
        *
    from {{ ref('sell_order2') }}
)

select * from source_data
```

Incremental model full refresh sample reference

```
{{ config(
    materialized='incremental',
    full_refresh = true
)}}

select * from
 {{ source('dbt_source', 'sell_user') }}
```

Example of setting bucketing rules

Here buckets can be filled with auto or a positive integer, representing automatic bucketing and setting a fixed number of buckets respectively.

```
{{ config(
    materialized='incremental',
    unique_key=['account_id',"create_time"],
    distributed_by=['account_id'],
    buckets='auto'
) }}

with source_data as (
    select
        *
    from {{ ref('sell_order') }}
)

select
    *
    from source_data

{% if is_incremental() %}
    where
    create_time > (select max(create_time) from {{this}})
{% endif %}
```

Setting the number of replicas example reference

```
{{ config(
    materialized='table',
    replication_num=1
)}}

with source_data as (
    select
        *
    from {{ ref('sell_order2') }}
)

select * from source_data
```

Dynamic partition sample reference

```
{{ config(
    materialized='incremental',
    partition_by = 'create_time',
    partition_type = 'range',
        -- The properties here are the properties in the create table statement, which contains the configuration related to dynamic
partitioning
    properties = {
        "dynamic_partition.time_unit":"DAY",
        "dynamic_partition.end":"8",
        "dynamic_partition.prefix":"p",
        "dynamic_partition.buckets":"4",
        "dynamic_partition.create_history_partition":"true",
        "dynamic_partition.history_partition_num":"3"
    }
) }}

with source_data as (
    select
        *
    from {{ ref('sell_order2') }}
)

select
    *
    from source_data

{% if is_incremental() %}
    where
    create_time = DATE_SUB(CURDATE(), INTERVAL 1 DAY)
{% endif %}
```

Conventional partition sample reference

```
{{ config(
    materialized='incremental',
    partition_by = 'create_time',
    partition_type = 'range',
        -- partition_by_init here refers to the historical partitions for creating partition tables. The historical partitions of
the current doris version need to be manually specified.
    partition_by_init = [
        "PARTITION `p20240601` VALUES [(\"2024-06-01\"),  (\"2024-06-02\"))",
        "PARTITION `p20240602` VALUES [(\"2024-06-02\"),  (\"2024-06-03\"))"
```

```
    ]
 )}}

with source_data as (
    select
        *
    from {{ ref('sell_order2') }}
)

select
    *
    from source_data

{% if is_incremental() %}
    where
    -- If the my_date variable is provided, use this path (via the dbt run --vars '{"my_date": "\"2024-06-03\""}' command). If the
my_date variable is not provided (directly using dbt run), use the day before the current date. For the incremental selection here,
it is recommended to directly use doris's CURDATE() function, which is also a common path in production environments.
    create_time = {{ var('my_date' , 'DATE_SUB(CURDATE(), INTERVAL 1 DAY)') }}

{% endif %}
```

Batch date setting parameter sample reference

```
{{ config(
    materialized='incremental',
    partition_by = 'create_time',
    partition_type = 'range',
    ...
)}}

with source_data as (
    select
        *
    from {{ ref('sell_order2') }}
)

select
    *
    from source_data

{% if is_incremental() %}
    where
    -- If the my_date variable is provided, use this path (via the dbt run --vars '{"my_date": "\"2024-06-03\""}' command). If the
my_date variable is not provided (directly using dbt run), use the day before the current date. For the incremental selection here,
it is recommended to directly use doris's CURDATE() function, which is also a common path in production environments.
    create_time = {{ var('my_date' , 'DATE_SUB(CURDATE(), INTERVAL 1 DAY)') }}

{% endif %}
```

Seatunnel Doris Sink

About SeaTunnel

SeaTunnel is a very easy-to-use ultra-high-performance distributed data integration platform that supports real-time synchronization of massive data. It can synchronize tens of billions of data stably and efficiently every day.

Connector-V2

The connector-v2 for SeaTunnel supports Doris Sink since version 2.3.1 and supports exactly-once write and CDC data synchronization

Plugin Code

SeaTunnel Doris Sink Plugin Code

Options

name	type	required	default value
fenodes	string	yes	-
username	string	yes	-
password	string	yes	-
table.identifier	string	yes	-
sink.label-prefix	string	yes	-
sink.enable-2pc	bool	no	true
sink.enable-delete	bool	no	false
doris.config	map	yes	-

`fenodes [string]`

Doris cluster FE Nodes address, the format is `"fe_ip:fe_http_port, ..."`

`username [string]`

Doris user username

`password [string]`

Doris`user password

`table.identifier [string]`

The name of Doris table,The format is DBName.TableName

`sink.label-prefix [string]`

The label prefix used by stream load imports. In the 2pc scenario, global uniqueness is required to ensure the EOS semantics of SeaTunnel.

`sink.enable-2pc [bool]`

Whether to enable two-phase commit (2pc), the default is true, to ensure Exactly-Once semantics. For two-phase commit, please refer to here.

`sink.enable-delete [bool]`

Whether to enable deletion. This option requires Doris table to enable batch delete function (0.15+ version is enabled by default), and only supports Unique model. you can get more detail at this link:

batch delete

`doris.config [map]`

The parameter of the stream load `data_desc`, you can get more detail at this link:

More Stream Load parameters

Example

Use JSON format to import data

```
sink {
    Doris {
        fenodes = "doris_fe:8030"
        username = root
        password = ""
        table.identifier = "test.table_sink"
        sink.enable-2pc = "true"
        sink.label-prefix = "test_json"
        doris.config = {
            format="json"
            read_json_by_line="true"
        }
    }
}
```

Use CSV format to import data

```
sink {
    Doris {
        fenodes = "doris_fe:8030"
        username = root
        password = ""
        table.identifier = "test.table_sink"
        sink.enable-2pc = "true"
        sink.label-prefix = "test_csv"
        doris.config = {
          format = "csv"
          column_separator = ","
        }
    }
}
```

Connector-V1

Flink Sink Doris

Plugin Code

Seatunnel Flink Sink Doris plugin code

Options

name	type	required	default value	engine
fenodes	string	yes	-	Flink
database	string	yes	-	Flink
table	string	yes	-	Flink
user	string	yes	-	Flink
password	string	yes	-	Flink
batch_size	int	no	100	Flink
interval	int	no	1000	Flink
max_retries	int	no	1	Flink
doris.*	-	no	-	Flink

`fenodes [string]`

Doris Fe http url, eg: 127.0.0.1:8030

`database [string]`

Doris database

`table [string]`

Doris table

`user [string]`

Doris user

`password [string]`

Doris password

`batch_size [int]`

The maximum number of lines to write to Doris at a time, the default value is 100

`interval [int]`

The flush interval (in milliseconds), after which the asynchronous thread writes the data in the cache to Doris. Set to 0 to turn off periodic writes.

`max_retries [int]`

Number of retries after writing to Doris fails

`doris.* [string]`

Import parameters for Stream load. For example: 'doris.column_separator' = ',' etc.

More Stream Load parameter configuration

Examples

Socket To Doris

```
env {
  execution.parallelism = 1
}
source {
    SocketStream {
      host = 127.0.0.1
      port = 9999
      result_table_name = "socket"
      field_name = "info"
    }
}
transform {
}
sink {
  DorisSink {
      fenodes = "127.0.0.1:8030"
      user = root
      password = 123456
      database = test
      table = test_tbl
      batch_size = 5
      max_retries = 1
      interval = 5000
    }
}
```

Start command

```
sh bin/start-seatunnel-flink.sh --config config/flink.streaming.conf
```

Spark Sink Doris

Plugin Code

Seatunnel Spark Sink Doris plugin code

Options

name	type	required	default value	engine
fenodes	string	yes	-	Spark
database	string	yes	-	Spark
table	string	yes	-	Spark
user	string	yes	-	Spark
password	string	yes	-	Spark
batch_size	int	yes	100	Spark
doris.*	string	no	-	Spark

`fenodes [string]`

Doris FE address:8030

`database [string]`

Doris target database name

`table [string]`

Doris target table name

`user [string]`

Doris user name

`password [string]`

Doris user's password

`batch_size [string]`

Doris number of submissions per batch

`doris. [string]` Doris stream_load properties,you can use 'doris.' prefix + stream_load properties

More Doris stream_load Configurations

Examples

Hive to Doris

Config properties

```
env{
  spark.app.name = "hive2doris-template"
}

spark {
```

```
  spark.sql.catalogImplementation = "hive"
}

source {
  hive {
    preSql = "select * from tmp.test"
    result_table_name = "test"
  }
}

transform {
}


sink {

Console {

  }

Doris {
   fenodes="xxxx:8030"
   database="gl_mint_dim"
   table="dim_date"
   user="root"
   password="root"
   batch_size=1000
   doris.column_separator="\t"
   doris.columns="date_key,date_value,day_in_year,day_in_month"
   }
}
```

Start command

```
sh bin/start-waterdrop-spark.sh --master local[4] --deploy-mode client --config ./config/spark.conf
```

Kettle Doris Plugin

Kettle Doris Plugin

Kettle Doris Plugin is used to write data from other data sources to Doris through Stream Load in Kettle.

This plug-in uses the Stream Load function of Doris to import data. It needs to be used in conjunction with the Kettle service.

About Kettle

Kettle is an open source ETL (Extract, Transform, Load) tool, first developed by Pentaho, Kettle is one of the core components of the Pentaho product suite, mainly used for data integration and data processing, and can easily complete the tasks of extracting data from various sources, cleaning and transforming data, and loading it into the target system.

For more information, please refer to: `https://pentaho.com/`

User Manual

Download Kettle and install

Kettle download address: https://pentaho.com/download/#download-pentaho After downloading, unzip it and run spoon.sh to start kettle You can also compile it yourself, refer to the Compilation Chapter

Compile Kettle Doris Plugin

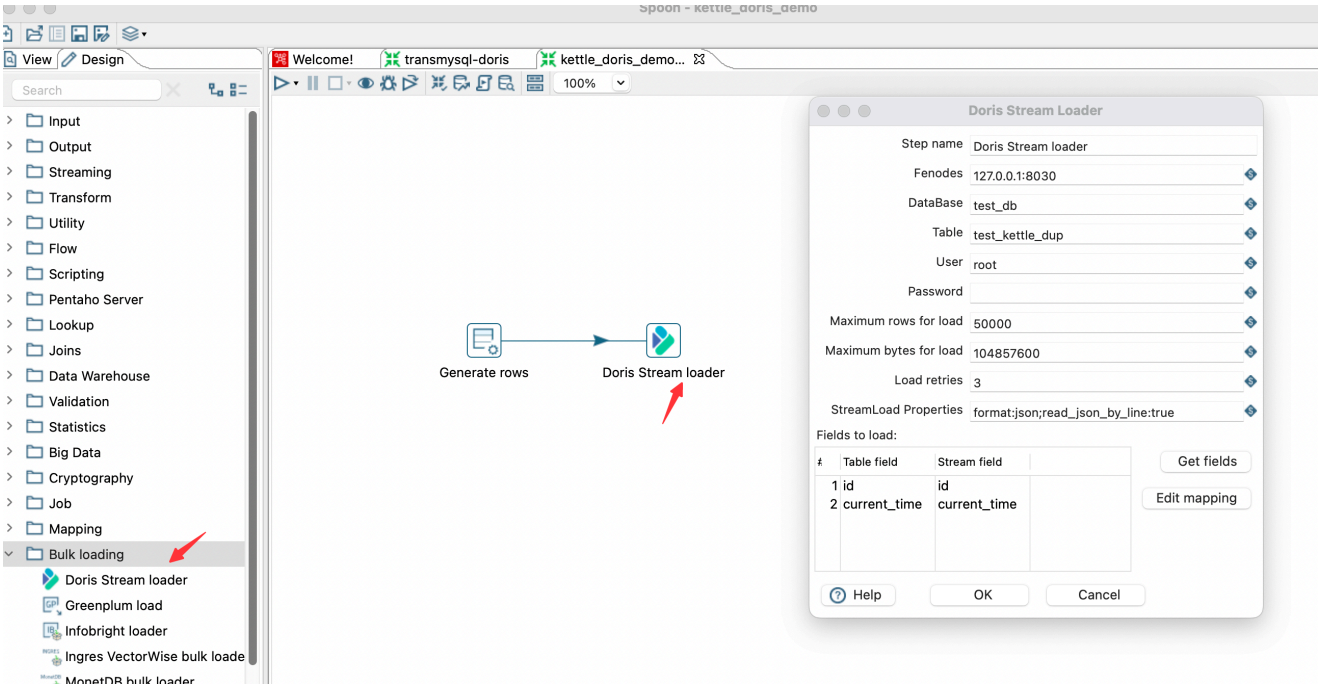
```
cd doris/extension/kettle
mvn clean package -DskipTests
```

After compiling, unzip the plug-in package and copy it to the plugins directory of kettle

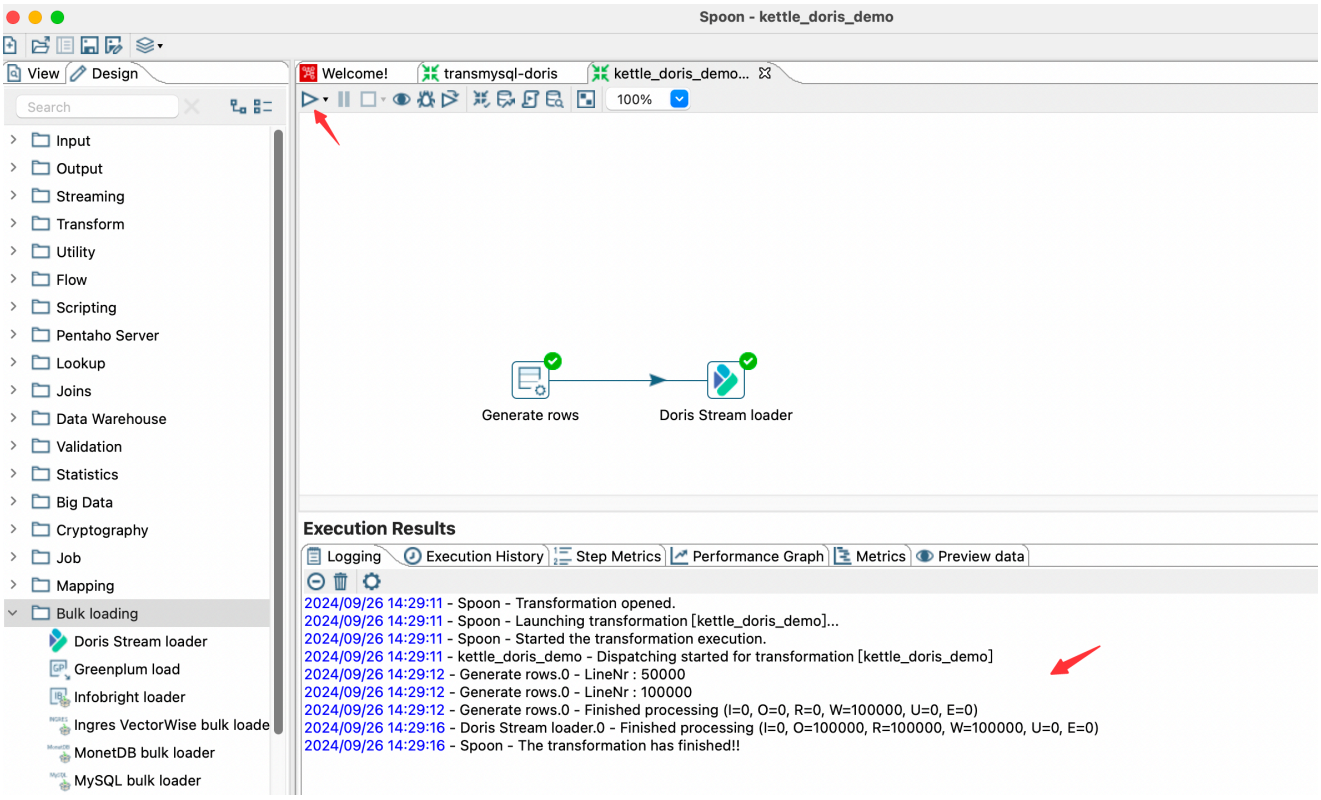
```
cd assemblies/plugin/target
unzip doris-stream-loader-plugins-9.4.0.0-343.zip
cp -r doris-stream-loader ${KETTLE_HOME}/plugins/
mvn clean package -DskipTests
```

Build a job

Find Doris Stream Loader in the batch loading in Kettle and build a job



Click Start Running the Job to complete data synchronization



Parameter Description

Key	Default Value	Required	Comment
Step name	–	Y	Step name

Key	Default Value	Required	Comment
fenodes	–	Y	Doris FE http address, supports multiple addresses, separated by commas
Database	–	Y	Doris write database
Target table	–	Y	Doris's write table
Username	–	Y	Username to access Doris
Password	–	N	Password to access Doris
Maximum number of rows for a single import	10000	N	Maximum number of rows for a single import
Maximum bytes for a single import	10485760 (10MB)	N	Maximum byte size for a single import
Number of import retries	3	N	Number of retries after import failure
StreamLoad properties	–	N	Streamload request header
Delete Mode	N	N	Whether to enable delete mode. By default, Stream Load performs insert operations. After the delete mode is enabled, all Stream Load writes are delete operations.

Kyuubi

Use Kyuubi with Doris

Introduction

Apache Kyuubi is a distributed and multi-tenant gateway to provide serverless SQL on Data Warehouses and Lakehouses. Apache Kyuubi is providing varied protocols like Thrift, Trino, MySQL etc., to the engines including Spark, Flink, Hive, JDBC, etc. Doris could be connected as JDBC data source with Doris dialect supported in Apache Kyuubi. Apache Kyuubi also provides a series of useful features including HA, service discovery, unified authentication, engine lifecycle management, etc.

Usage

Download Apache Kyuubi

Download Apache Kyuubi from https://kyuubi.apache.org/zh/releases.html

Get Apache Kyuubi 1.6.0 or above and extract it to folder.

Config Doris as Kyuubi data source

- Update Kyuubi configurations in `$KYUUBI_HOME/conf/kyuubi-defaults.conf`

```
kyuubi.engine.type=jdbc
kyuubi.engine.jdbc.type=doris
kyuubi.engine.jdbc.driver.class=com.mysql.cj.jdbc.Driver
kyuubi.engine.jdbc.connection.url=jdbc:mysql://xxx:xxx
kyuubi.engine.jdbc.connection.user=***
kyuubi.engine.jdbc.connection.password=***
```

Configuration	Description
kyuubi.engine.type	Engine Type, specify to `jdbc`
kyuubi.engine.jdbc.type	JDBC service type, specify to `doris`
kyuubi.engine.jdbc.driver.class	JDBC driver class name, specify to `com.mysql.cj.jdbc.Driver`
kyuubi.engine.jdbc.connection.url	JDBC url to Doris FE
kyuubi.engine.jdbc.connection.user	JDBC username
kyuubi.engine.jdbc.connection.password	JDBC password

- For other configuration in Apache Kyuubi, please refer to Apache Kyuubi Configuration Docs .

Add MySQL JDBC Driver

Copy the Mysql JDBC Driver `mysql-connector-j-8.X.X.jar` to `$KYUUBI_HOME/externals/engines/jdbc`.

Start Kyuubi Server

Run `$KYUUBI_HOME/bin/kyuubi start`. After started, port 10009 by default is listened by Kyuubi Server with Thrift protocol.

Example

The following example shows basic example of querying Doris with Kyuubi with beeline CLI in Thrift protocol.

Connect to Kyuubi with Beeline

```
$ $KYUUBI_HOME/bin/beeline -u "jdbc:hive2://xxxx:10009/"
```

Execute Query to Kyuubi

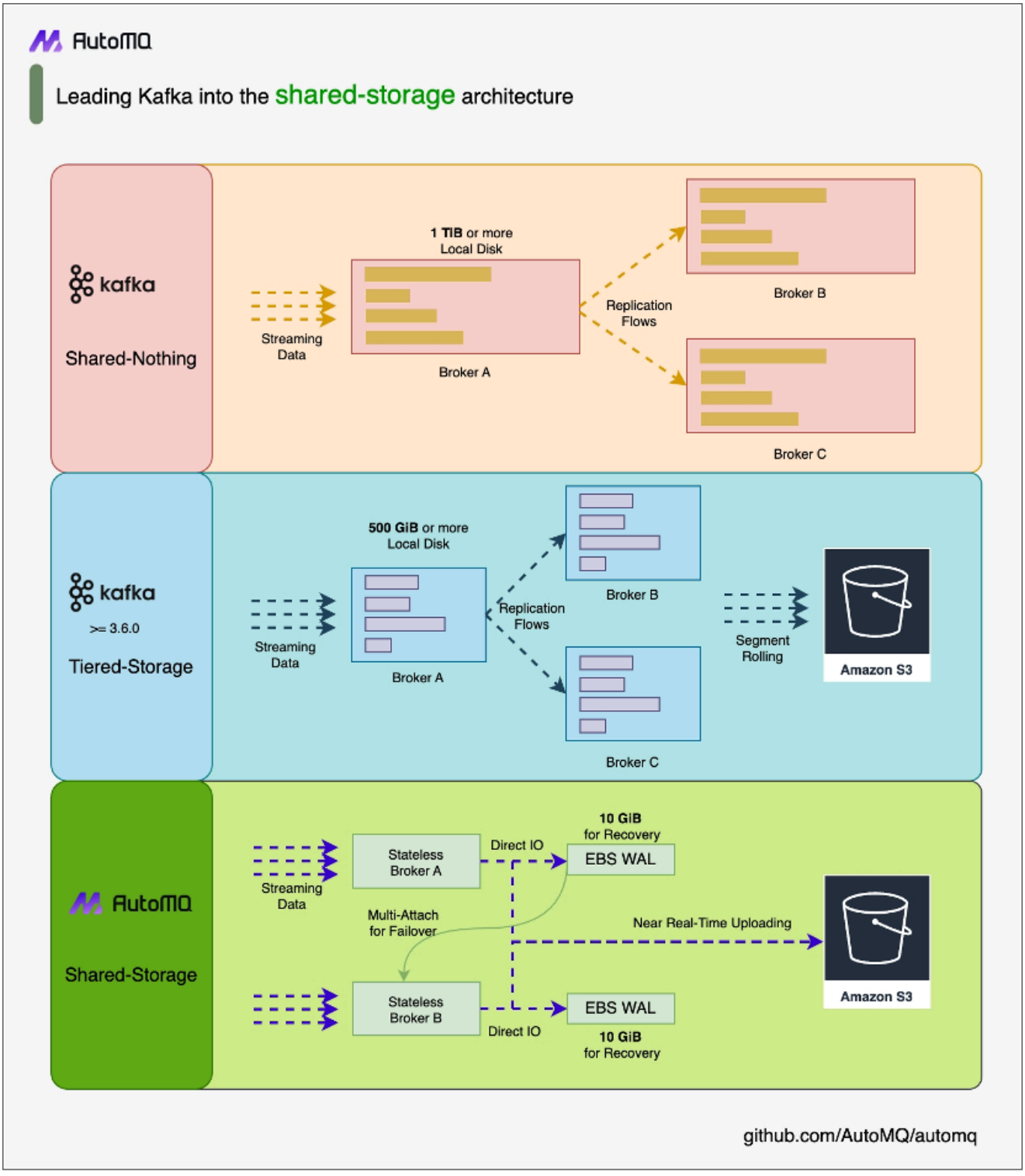
Execute query statement `select * from demo.expamle_tbl;` with query results returned.

```
0: jdbc:hive2://xxxx:10009/> select * from demo.example_tbl;

2023-03-07 09:29:14.771 INFO org.apache.kyuubi.operation.ExecuteStatement: Processing anonymous's query[bdc59dd0-ceea-4c02-8c3a-
23424323f5db]: PENDING_STATE -> RUNNING_STATE, statement:
select * from demo.example_tbl
2023-03-07 09:29:14.786 INFO org.apache.kyuubi.operation.ExecuteStatement: Query[bdc59dd0-ceea-4c02-8c3a-23424323f5db] in
FINISHED_STATE
2023-03-07 09:29:14.787 INFO org.apache.kyuubi.operation.ExecuteStatement: Processing anonymous's query[bdc59dd0-ceea-4c02-8c3a-
23424323f5db]: RUNNING_STATE -> FINISHED_STATE, time taken: 0.015 seconds
+----------+-------------+-------+------+------+----------------------+-------+-----------------+-----------------+
| user_id  |    date     | city  | age  | sex  |    last_visit_date     | cost  | max_dwell_time  | min_dwell_time  |
+----------+-------------+-------+------+------+----------------------+-------+-----------------+-----------------+
| 10000    | 2017-10-01  | Beijing | 20   | 0    | 2017-10-01 07:00:00.0  | 70    | 10              | 2               |
| 10001    | 2017-10-01  | Beijing | 30   | 1    | 2017-10-01 17:05:45.0  | 4     | 22              | 22              |
| 10002    | 2017-10-02  | Shanghai| 20   | 1    | 2017-10-02 12:59:12.0  | 400   | 5               | 5               |
| 10003    | 2017-10-02  | Guangzhou| 32   | 0    | 2017-10-02 11:20:00.0  | 60    | 11              | 11              |
| 10004    | 2017-10-01  | Shenzhen| 35   | 0    | 2017-10-01 10:00:15.0  | 200   | 3               | 3               |
| 10004    | 2017-10-03  | Shenzhen| 35   | 0    | 2017-10-03 10:20:22.0  | 22    | 6               | 6               |
+----------+-------------+-------+------+------+----------------------+-------+-----------------+-----------------+
6 rows selected (0.068 seconds)
```

AutoMQ Load

AutoMQ is a cloud-native fork of Kafka by separating storage to object storage like S3. It remains 100% compatible with Apache Kafka® while offering users up to a 10x cost-effective and 100x elasticity . Through its innovative shared storage architecture, it achieves capabilities such as reassign partitions in seconds, self-balancing and auto scaling in seconds while ensuring high throughput and low latency.



This article will explain how to use Apache Doris Routine Load to import data from AutoMQ into Doris. For more details on Routine Load, please refer to the Routine Load document.

Environment Preparation

Prepare Apache Doris and Test Data

Ensure that a working Apache Doris cluster is already set up. For demonstration purposes, we have deployed a test Apache Doris environment on Linux following the Deploying with Docker document. Create databases and test tables:

```
create database automq_db;
CREATE TABLE automq_db.users (
                                id bigint NOT NULL,
                                name string NOT NULL,
                                timestamp string NULL,
                                status string NULL

) DISTRIBUTED BY hash (id) PROPERTIES ('replication_num' = '1');
```

Prepare Kafka Command Line Tools

Download the latest TGZ package from AutoMQ Releases and extract it. Assuming the extraction directory is $AUTOMQ_HOME, this article will use the scripts under $AUTOMQ_HOME/bin to create topics and generate test data.

Prepare AutoMQ and test data

Refer to the AutoMQ official deployment documentation to deploy a functional cluster, ensuring network connectivity between AutoMQ and Apache Doris. Quickly create a topic named example_topic in AutoMQ and write a test JSON data to it by following these steps.

Create Topic

Use the Apache Kafka® command line tool in AutoMQ to create the topic, ensuring that you have access to a Kafka environment and that the Kafka service is running. Here is an example command to create a topic:

```
$AUTOMQ_HOME/bin/kafka-topics.sh --create --topic exampleto_topic --bootstrap-server 127.0.0.1:9092  --partitions 1 --replication-factor 1
```

> Tips: When executing the command, replace `topic` and `bootstarp-server` with the actual AutoMQ Bootstrap Server address.

After creating the topic, you can use the following command to verify that the topic has been successfully created.

```
$AUTOMQ_HOME/bin/kafka-topics.sh --describe example_topic --bootstrap-server 127.0.0.1:9092
```

Generate test data

Create a JSON-formatted test data entry, corresponding to the table mentioned earlier.

```
{
  "id": 1,
  "name": "testuser",
  "timestamp": "2023-11-10T12:00:00",
  "status": "active"
}
```

Write test data

Use Kafka's command-line tools or a programming approach to write the test data to a topic named `example_topic`. Below is an example using the command-line tool:

```
echo '{"id": 1, "name": "testuser", "timestamp": "2023-11-10T12:00:00", "status": "active"}' | sh kafka-console-producer.sh --
broker-list 127.0.0.1:9092 --topic example_topic
```

To view the data just written to the topic, use the following command:

```
sh $AUTOMQ_HOME/bin/kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic example_topic --from-beginning
```

> Tips: When executing the command, replace `topic` and `bootstarp-server` with the actual AutoMQ Bootstrap Server address.

Create a Routine Load import job

In the Apache Doris command line, create a Routine Load job that accepts JSON data to continuously import data from an AutoMQ Kafka topic. For detailed parameter information of Routine Load, please refer to [Doris Routine Load].

```
CREATE ROUTINE LOAD automq_example_load ON users
COLUMNS(id, name, timestamp, status)
PROPERTIES
(
    "format" = "json",
    "jsonpaths" = "[\"$.id\",\"$.name\",\"$.timestamp\",\"$.status\"]"
 )
FROM KAFKA
(
    "kafka_broker_list" = "127.0.0.1:9092",
    "kafka_topic" = "example_topic",
    "property.kafka_default_offsets" = "OFFSET_BEGINNING"
);
```

> Tips: When executing the command, you need to replace kafka_broker_list with the actual AutoMQ Bootstrap Server address.

Verify data import

First, check the status of the Routine Load import job to ensure that the task is running.

```
show routine load\G;
```

Then query the relevant tables in the Apache Doris database, and you will see that the data has been successfully imported.

```
select * from users;
+------+--------------+---------------------+--------+
| id   | name         | timestamp           | status |
+------+--------------+---------------------+--------+
|    1 | testuser     | 2023-11-10T12:00:00 | active |
|    2 | testuser     | 2023-11-10T12:00:00 | active |
+------+--------------+---------------------+--------+
2 rows in set (0.01 sec)
```

Hive Bitmap UDF

Hive UDF

Hive Bitmap UDF provides UDFs for generating bitmap and bitmap operations in hive tables. The bitmap in Hive is exactly the same as the Doris bitmap. The bitmap in Hive can be imported into doris through (spark bitmap load).

the main purpose: 1. Reduce the time of importing data into doris, and remove processes such as dictionary building and bitmap pre-aggregation; 2. Save hive storage, use bitmap to compress data, reduce storage cost; 3. Provide flexible bitmap operations in hive, such as: intersection, union, and difference operations, and the calculated bitmap can also be directly imported into doris; imported into doris;

How To Use

Create Bitmap type table in Hive

```
-- Example: Create Hive Bitmap Table
CREATE TABLE IF NOT EXISTS `hive_bitmap_table`(
  `k1`   int       COMMENT '',
  `k2`   String    COMMENT '',
  `k3`   String    COMMENT '',
  `uuid` binary    COMMENT 'bitmap'
) comment  'comment'

-- Example: Create Hive Table
CREATE TABLE IF NOT EXISTS `hive_table`(
  `k1`   int       COMMENT '',
  `k2`   String    COMMENT '',
  `k3`   String    COMMENT '',
  `uuid` int       COMMENT ''
) comment  'comment'
```

Hive Bitmap UDF Usage:

Hive Bitmap UDF used in Hive/Spark,First, you need to compile fe to get hive-udf-jar-with-dependencies.jar. Compilation preparation:If you have compiled the ldb source code, you can directly compile fe,If you have compiled the ldb source code, you can compile it directly. If you have not compiled the ldb source code, you need to manually install thrift, Reference:Setting Up dev env for FE.

```
--clone doris code
git clone https://github.com/apache/doris.git
cd doris
git submodule update --init --recursive
--install thrift
--Enter the fe directory
cd fe
--Execute the maven packaging command (All sub modules of fe will be packaged)
mvn package -Dmaven.test.skip=true
```

```
--You can also just package the hive-udf module
mvn package -pl hive-udf -am -Dmaven.test.skip=true
```

After packaging and compiling, enter the hive-udf directory and there will be a target directory,There will be hive-udf.jar package

```
-- Load the Hive Bitmap Udf jar package (Upload the compiled hive-udf jar package to HDFS)
add jar hdfs://node:9001/hive-udf-jar-with-dependencies.jar;
-- Create Hive Bitmap UDAF function
create temporary function to_bitmap as 'org.apache.doris.udf.ToBitmapUDAF' USING JAR 'hdfs://node:9001/hive-udf-jar-with-
dependencies.jar';
create temporary function bitmap_union as 'org.apache.doris.udf.BitmapUnionUDAF' USING JAR 'hdfs://node:9001/hive-udf-jar-with-
dependencies.jar';
-- Create Hive Bitmap UDF function
create temporary function bitmap_count as 'org.apache.doris.udf.BitmapCountUDF' USING JAR 'hdfs://node:9001/hive-udf-jar-with-
dependencies.jar';
create temporary function bitmap_and as 'org.apache.doris.udf.BitmapAndUDF' USING JAR 'hdfs://node:9001/hive-udf-jar-with-
dependencies.jar';
create temporary function bitmap_or as 'org.apache.doris.udf.BitmapOrUDF' USING JAR 'hdfs://node:9001/hive-udf-jar-with-
dependencies.jar';
create temporary function bitmap_xor as 'org.apache.doris.udf.BitmapXorUDF' USING JAR 'hdfs://node:9001/hive-udf-jar-with-
dependencies.jar';
-- Example: Generate bitmap by to_bitmap function and write to Hive Bitmap table
insert into hive_bitmap_table
select
    k1,
    k2,
    k3,
    to_bitmap(uuid) as uuid
from
    hive_table
group by
    k1,
    k2,
    k3
-- Example: The bitmap_count function calculate the number of elements in the bitmap
select k1,k2,k3,bitmap_count(uuid) from hive_bitmap_table
-- Example: The bitmap_union function calculate the grouped bitmap union
select k1,bitmap_union(uuid) from hive_bitmap_table group by k1
```

Hive Bitmap UDF Description

Hive Bitmap import into Doris

Method 1：Catalog (recommended)

When create a Hive table in the format specified as TEXT, for Binary type, Hive will be saved as a bash64 encoded string. Therefore, the binary data can be directly saved as Bitmap through bitmap_from_base64 function by using Doris's Hive Catalog.

Here is a full example:

1. Creating Hive Tables in Hive

```
CREATE TABLE IF NOT EXISTS `test`.`hive_bitmap_table`(
`k1`   int       COMMENT '',
`k2`   String    COMMENT '',
```

```
`k3`   String    COMMENT '',
`uuid` binary    COMMENT 'bitmap'
) stored as textfile
```

2. Creating a Catalog in Doris

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://127.0.0.1:9083'
);
```

3. Create Doris internal table

```
CREATE TABLE IF NOT EXISTS `test`.`doris_bitmap_table`(
    `k1`   int                   COMMENT '',
    `k2`   String                COMMENT '',
    `k3`   String                COMMENT '',
    `uuid` BITMAP  BITMAP_UNION  COMMENT 'bitmap'
)
AGGREGATE KEY(k1, k2, k3)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
    "replication_allocation" = "tag.location.default: 1"
);
```

4. Inserting data from Hive into Doris

```
insert into doris_bitmap_table select k1, k2, k3, bitmap_from_base64(uuid) from hive.test.hive_bitmap_table;
```

Method 2: Spark Load

see details: Spark Load -> Basic operation -> Create load(Example 3: when the upstream data source is hive binary type table)

Install Error

Operation and Maintenance Error

This document is mainly used to record the common problems of operation and maintenance during the use of Doris. It will be updated from time to time.

The name of the BE binary that appears in this doc is `doris_be`, which was `palo_be` in previous versions.

Q1. Why is there always some tablet left when I log off the BE node through DECOMMISSION?

During the offline process, use show backends to view the tabletNum of the offline node, and you will observe that the number of tabletNum is decreasing, indicating that data shards are being migrated from this node. When the number is reduced to 0, the system will automatically delete the node. But in some cases, tabletNum will not change after it drops to a certain value. This is usually possible for two reasons:

1. The tablets belong to the table, partition, or materialized view that was just dropped. Objects that have just been deleted remain in the recycle bin. The offline logic will not process these shards. The time an object resides in the recycle bin can be modified by modifying the FE configuration parameter catalog_trash_expire_second. These tablets are disposed of when the object is removed from the recycle bin.
2. There is a problem with the migration task for these tablets. At this point, you need to view the errors of specific tasks through show proc `show proc "/cluster_balance"`.

For the above situation, you can first check whether there are unhealthy shards in the cluster through `show proc "/cluster_health/tablet_health";`. If it is 0, you can delete the BE directly through the drop backend statement. Otherwise, you also need to check the replicas of unhealthy shards in detail.

Q2. How should priorty_network be set?

priorty_network is a configuration parameter for both FE and BE. This parameter is mainly used to help the system select the correct network card IP as its own IP. It is recommended to explicitly set this parameter in any case to prevent the problem of incorrect IP selection caused by adding new network cards to subsequent machines.

The value of priorty_network is expressed in CIDR format. Divided into two parts, the first part is the IP address in dotted decimal, and the second part is a prefix length. For example 10.168.1.0/8 will match all 10.xx.xx.xx IP addresses, and 10.168.1.0/16 will match all 10.168.xx.xx IP addresses.

The reason why the CIDR format is used instead of specifying a specific IP directly is to ensure that all nodes can use a uniform configuration value. For example, there are two nodes: 10.168.10.1 and 10.168.10.2, then we can use 10.168.10.0/24 as the value of priorty_network.

Q3. What are the Master, Follower and Observer of FE?

First of all, make it clear that FE has only two roles: Follower and Observer. The Master is just an FE selected from a group of Follower nodes. Master can be regarded as a special kind of Follower. So when we were asked how many FEs a cluster had and what roles they were, the correct answer should be the number of all FE nodes, the number of Follower roles and the number of Observer roles.

All FE nodes of the Follower role will form an optional group, similar to the group concept in the Paxos consensus protocol. A Follower will be elected as the Master in the group. When the Master hangs up, a new Follower will be automatically selected as the Master. The Observer will not participate in the election, so the Observer will not be called Master.

A metadata log needs to be successfully written in most Follower nodes to be considered successful. For example, if there are 3 FEs, only 2 can be successfully written. This is why the number of Follower roles needs to be an odd number.

The role of Observer is the same as the meaning of this word. It only acts as an observer to synchronize the metadata logs that have been successfully written, and provides metadata reading services. He will not be involved in the logic of the majority writing.

Typically, 1 Follower + 2 Observer or 3 Follower + N Observer can be deployed. The former is simple to operate and maintain, and there is almost no consistency agreement between followers to cause such complex error situations (Most companies use this method). The latter can ensure the high availability of metadata writing. If it is a high concurrent query scenario, Observer can be added appropriately.

Q4. A new disk is added to the node, why is the data not balanced to the new disk?

The current Doris balancing strategy is based on nodes. That is to say, the cluster load is judged according to the overall load index of the node (number of shards and total disk utilization). And migrate data shards from high-load nodes to low-load nodes. If each node adds a disk, from the overall point of view of the node, the load does not change, so the balancing logic cannot be triggered.

In addition, Doris currently does not support balancing operations between disks within a single node. Therefore, after adding a new disk, the data will not be balanced to the new disk.

However, when data is migrated between nodes, Doris takes the disk into account. For example, when a shard is migrated from node A to node B, the disk with low disk space utilization in node B will be preferentially selected.

Here we provide 3 ways to solve this problem:

1. Rebuild the new table

   Create a new table through the create table like statement, and then use the insert into select method to synchronize data from the old table to the new table. Because when a new table is created, the data shards of the new table will be distributed in the new disk, so the data will also be written to the new disk. This method is suitable for situations where the amount of data is small (within tens of GB).

2. Through the Decommission command

The decommission command is used to safely decommission a BE node. This command will first migrate the data shards on the node to other nodes, and then delete the node. As mentioned earlier, during data migration, the disk with low disk utilization will be prioritized, so this method can “force” the data to be migrated to the disks of other nodes. When the data migration is completed, we cancel the decommission operation, so that the data will be rebalanced back to this node. When we perform the above steps on all BE nodes, the data will be evenly distributed on all disks of all nodes.

Note that before executing the decommission command, execute the following command to avoid the node being deleted after being offline.

```
admin set frontend config("drop_backend_after_decommission" = "false");
```

3. Manually migrate data using the API

   Doris provides HTTP API, which can manually specify the migration of data shards on one disk to another disk.

Q5. How to read FE/BE logs correctly?

In many cases, we need to troubleshoot problems through logs. The format and viewing method of the FE/BE log are described here.

1. FE

   FE logs mainly include:

   - fe.log: main log. Includes everything except fe.out.
   - fe.warn.log: A subset of the main log, only WARN and ERROR level logs are logged.
   - fe.out: log for standard/error output (stdout and stderr).
   - fe.audit.log: Audit log, which records all SQL requests received by this FE.

   A typical FE log is as follows:

   ```
   2021-09-16 23:13:22,502 INFO (tablet scheduler|43) [BeLoadRebalancer.selectAlternativeTabletsForCluster():85] cluster is
   balance: default_cluster with medium: HDD.skip
   ```

   - `2021-09-16 23:13:22,502`: log time.
   - `INFO`: log level, default is INFO.
   - `(tablet scheduler|43)`: thread name and thread id. Through the thread id, you can view the context information of this thread and check what happened in this thread.
   - `BeLoadRebalancer.selectAlternativeTabletsForCluster():85`: class name, method name and code line number.
   - `cluster is balance xxx`: log content.

   Usually, we mainly view the fe.log log. In special cases, some logs may be output to fe.out.

2. BE

   BE logs mainly include:

   - be.INFO: main log. This is actually a soft link, connected to the latest be.INFO.xxxx.

- be.WARNING: A subset of the main log, only WARN and FATAL level logs are logged. This is actually a soft link, connected to the latest be.WARN.xxxx.
- be.out: log for standard/error output (stdout and stderr).

A typical BE log is as follows:

```
I0916 23:21:22.038795 28087 task_worker_pool.cpp:1594] finish report TASK. master host: 10.10.10.10, port: 9222
```

- `I0916 23:21:22.038795`: log level and datetime. The capital letter I means INFO, W means WARN, and F means FATAL.
- `28087`: thread id. Through the thread id, you can view the context information of this thread and check what happened in this thread.
- `task_worker_pool.cpp:1594`: code file and line number.
- `finish report TASK xxx`: log content.

Usually we mainly look at the be.INFO log. In special cases, such as BE downtime, you need to check be.out.

Q6. How to troubleshoot the FE/BE node is down?

1. BE

   The BE process is a C/C++ process, which may hang due to some program bugs (memory out of bounds, illegal address access, etc.) or Out Of Memory (OOM). At this point, we can check the cause of the error through the following steps:

   1. View be.out

      The BE process realizes that when the program exits due to an exception, it will print the current error stack to be.out (note that it is be.out, not be.INFO or be.WARNING). Through the error stack, you can usually get a rough idea of where the program went wrong.

      Note that if there is an error stack in be.out, it is usually due to a program bug, and ordinary users may not be able to solve it by themselves. Welcome to the WeChat group, github discussion or dev mail group for help, and post the corresponding error stack, so that you can quickly Troubleshoot problems.

   2. dmesg

      If there is no stack information in be.out, the probability is that OOM was forcibly killed by the system. At this time, you can use the dmesg -T command to view the Linux system log. If a log similar to Memory cgroup out of memory: Kill process 7187 (doris_be) score 1007 or sacrifice child appears at the end, it means that it is caused by OOM.

      Memory problems can have many reasons, such as large queries, imports, compactions, etc. Doris is also constantly optimizing memory usage. Welcome to the WeChat group, github discussion or dev mail group for help.

   3. Check whether there are logs beginning with F in be.INFO.

Logs starting with F are Fatal logs. For example, F0916 , indicating the Fatal log on September 16th. Fatal logs usually indicate a program assertion error, and an assertion error will directly cause the process to exit (indicating a bug in the program). Welcome to the WeChat group, github discussion or dev mail group for help.

2. FE

FE is a java process, and the robustness is better than the C/C++ program. Usually the reason for FE to hang up may be OOM (Out-of-Memory) or metadata write failure. These errors usually have an error stack in fe.log or fe.out. Further investigation is required based on the error stack information.

Q7. About the configuration of data directory SSD and HDD, create table encounter error `Failed to find enough host with storage medium and tag`

Doris supports one BE node to configure multiple storage paths. Usually, one storage path can be configured for each disk. At the same time, Doris supports storage media properties that specify paths, such as SSD or HDD. SSD stands for high-speed storage device and HDD stands for low-speed storage device.

If the cluster only has one type of medium, such as all HDD or all SSD, the best practice is not to explicitly specify the medium property in be.conf. If encountering the error `Failed to find enough host with storage medium and tag` mentioned above, it is generally because be.conf only configures the SSD medium, while the table creation stage explicitly specifies `properties {"storage_medium" = "hdd"}`; similarly, if be.conf only configures the HDD medium, and the table creation stage explicitly specifies `properties {"storage_medium" = "ssd"}`, the same error will occur. The solution is to modify the properties parameter in the table creation to match the configuration; or remove the explicit configuration of SSD/HDD in be.conf.

By specifying the storage medium properties of the path, we can take advantage of Doris's hot and cold data partition storage function to store hot data in SSD at the partition level, while cold data is automatically transferred to HDD.

It should be noted that Doris does not automatically perceive the actual storage medium type of the disk where the storage path is located. This type needs to be explicitly indicated by the user in the path configuration. For example, the path "/path/to/data1.SSD" means that this path is an SSD storage medium. And "data1.SSD" is the actual directory name. Doris determines the storage media type based on the ".SSD" suffix after the directory name, not the actual storage media type. That is to say, the user can specify any path as the SSD storage medium, and Doris only recognizes the directory suffix and does not judge whether the storage medium matches. If no suffix is written, it will default to HDD.

In other words, ".HDD" and ".SSD" are only used to identify the "relative" "low speed" and "high speed" of the storage directory, not the actual storage medium type. Therefore, if the storage path on the BE node has no medium difference, the suffix does not need to be filled in.

Q8. Multiple FEs cannot log in when using Nginx to implement web UI load balancing

Doris can deploy multiple FEs. When accessing the Web UI, if Nginx is used for load balancing, there will be a constant prompt to log in again because of the session problem. This problem is actually a problem of session sharing. Nginx provides centralized session sharing. The solution, here we use the ip_hash technology in nginx, ip_hash can direct the request of an ip to the same backend, so that a client and a backend under this ip can establish a stable session, ip_hash is defined in the upstream configuration:

```
upstream doris.com {
   server 172.22.197.238:8030 weight=3;
   server 172.22.197.239:8030 weight=4;
   server 172.22.197.240:8030 weight=4;
   ip_hash;
}
```

The complete Nginx example configuration is as follows:

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;
    #include /etc/nginx/custom/*.conf;
    upstream doris.com {
      server 172.22.197.238:8030 weight=3;
      server 172.22.197.239:8030 weight=4;
      server 172.22.197.240:8030 weight=4;
      ip_hash;
    }

    server {
        listen 80;
        server_name gaia-pro-bigdata-fe02;
        if ($request_uri ~ _load) {
           return 307 http://$host$request_uri ;
        }

        location / {
            proxy_pass http://doris.com;
            proxy_redirect default;
        }
        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root html;
```

```
        }
    }
}
```

Q9. FE fails to start, “wait catalog to be ready. FE type UNKNOWN” keeps scrolling in fe.log

There are usually two reasons for this problem:

1. The local IP obtained when FE is started this time is inconsistent with the last startup, usually because `priority_network` is not set correctly, which causes FE to match the wrong IP address when it starts. Restart FE after modifying `priority_network`.
2. Most Follower FE nodes in the cluster are not started. For example, there are 3 Followers, and only one is started. At this time, at least one other FE needs to be started, so that the FE electable group can elect the Master to provide services.

If the above situation cannot be solved, you can restore it according to the [metadata operation and maintenance document] (../admin-manual/trouble-shooting/metadata-operation.md) in the Doris official website document.

Q10. Lost connection to MySQL server at ‘reading initial communication packet’, system error: 0

If the following problems occur when using MySQL client to connect to Doris, this is usually caused by the different jdk version used when compiling FE and the jdk version used when running FE. Note that when using docker to compile the image, the default JDK version is openjdk 11, and you can switch to openjdk 8 through the command (see the compilation documentation for details).

Q11. recoveryTracker should overlap or follow on disk last VLSN of 4,422,880 recoveryFirst= 4,422,882 UNEXPECTED_STATE_FATAL

Sometimes when FE is restarted, the above error will occur (usually only in the case of multiple Followers). And the two values in the error differ by 2. Causes FE to fail to start.

This is a bug in bdbje that has not yet been resolved. In this case, you can only restore the metadata by performing the operation of failure recovery in Metadata Operation and Maintenance Documentation.

Q12. Doris compile and install JDK version incompatibility problem

When compiling Doris using Docker, start FE after compiling and installing, and the exception message `java.lang.Suchmethoderror: java.nio.ByteBuffer.limit (I)Ljava/nio/ByteBuffer;` appears, this is because the default in Docker It is JDK 11. If your installation environment is using JDK8, you need to switch the JDK environment to JDK8 in Docker. For the specific switching method, please refer to Compile Documentation

Q13. Error starting FE or unit test locally Cannot find external parser table action_table.dat

Run the following command

```
cd fe && mvn clean install -DskipTests
```

If the same error is reported, Run the following command

```
cp fe-core/target/generated-sources/cup/org/apache/doris/analysis/action_table.dat fe-core/target/classes/org/apache/doris/analysis
```

Q14. Doris upgrades to version 1.0 or later and reports error `Failed to set ciphers to use (2026) in MySQL appearance via ODBC.

This problem occurs after doris upgrades to version 1.0 and uses Connector/ODBC 8.0.x or higher. Connector/ODBC 8.0.x has multiple access methods, such as `/usr/lib64/libmyodbc8w.so` which is installed via yum and relies on `libssl.so.10` and `libcrypto.so.10`. In doris 1.0 onwards, openssl has been upgraded to 1.1 and is built into the doris binary package, so this can lead to openssl conflicts and errors like the following

```
ERROR 1105 (HY000): errCode = 2, detailMessage = driver connect Error: HY000 [MySQL][ODBC 8.0(w) Driver]SSL connection error: Failed to set ciphers to use (2026)
```

The solution is to use the `Connector/ODBC 8.0.28` version of ODBC Connector and select `Linux - Generic` in the operating system, this version of ODBC Driver uses openssl version 1.1. Or use a lower version of ODBC connector, e.g. Connector/ODBC 5.3.14. For details, see the ODBC exterior documentation.

You can verify the version of openssl used by MySQL ODBC Driver by

```
ldd /path/to/libmyodbc8w.so |grep libssl.so
```

If the output contains `libssl.so.10`, there may be problems using it, if it contains `libssl.so.1.1`, it is compatible with doris 1.0

Q15. After upgrading to version 1.2, the BE NoClassDefFoundError issue failed to start

Java UDF dependency error If the upgrade support starts be, the following Java `NoClassDefFoundError` error occurs

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/doris/udf/IniUtil
Caused by: java.lang.ClassNotFoundException: org.apache.doris.udf.JniUtil
```

You need to download the Java UDF function dependency package of `apache-doris-java-udf-jar-with-dependencies-1.2.0` from the official website, put it in the lib directory under the BE installation directory, and then restart BE

Q16. After upgrading to version 1.2, BE startup shows Failed to initialize JNI

If the following `Failed to initialize JNI` error occurs when starting BE after upgrading

```
Failed to initialize JNI: Failed to find the library libjvm.so.
```

You need to set the `JAVA_HOME` environment variable, or set `JAVA_HOME` variable in be.conf and restart the BE node.

Q17. Docker: backend fails to start

This may be due to the CPU not supporting AVX2, check the backend logs with `docker logs -f be`. If the CPU does not support AVX2, the `apache/doris:1.2.2-be-x86_64-noavx2` image must be used, instead of `apache/doris:1.2.2-be-x86_64`. Note that the image version number will change over time, check Dockerhub for the most recent version.

Data Operation Error

Data Operation Error

This document is mainly used to record common problems of data operation during the use of Doris. It will be updated from time to time.

Q1. Use Stream Load to access FE's public network address to import data, but is redirected to the intranet IP?

When the connection target of stream load is the http port of FE, FE will only randomly select a BE node to perform the http 307 redirect operation, so the user's request is actually sent to a BE assigned by FE. The redirect returns the IP of the BE, that is, the intranet IP. So if you send the request through the public IP of FE, it is very likely that you cannot connect because it is redirected to the internal network address.

The usual way is to ensure that you can access the intranet IP address, or to assume a load balancer for all BE upper layers, and then directly send the stream load request to the load balancer, and the load balancer will transparently transmit the request to the BE node .

Q2. Does Doris support changing column names?

After version 1.2.0, when the `"light_schema_change"="true"` option is enabled, column names can be modified.

Before version 1.2.0 or when the `"light_schema_change"="true"` option is not enabled, modifying column names is not supported. The reasons are as follows:

Doris supports modifying database name, table name, partition name, materialized view (Rollup) name, as well as column type, comment, default value, etc. But unfortunately, modifying column names is currently not supported.

For some historical reasons, the column names are currently written directly to the data file. When Doris queries, it also finds the corresponding column through the class name. Therefore, modifying the column name is not only a simple metadata modification, but also involves data rewriting, which is a very heavy operation.

We do not rule out some compatible means to support lightweight column name modification operations in the future.

Q3. Does the table of the Unique Key model support creating a materialized view?

not support.

The table of the Unique Key model is a business-friendly table. Because of its unique function of deduplication according to the primary key, it can easily synchronize business databases with frequently changed data. Therefore, many users will first consider using the Unique Key model when accessing data into Doris.

But unfortunately, the table of the Unique Key model cannot establish a materialized view. The reason is that the essence of the materialized view is to "pre-compute" the data through pre-computation, so that the

calculated data is directly returned during the query to speed up the query. In the materialized view, the “pre-computed” data is usually some aggregated indicators, such as sum and count. At this time, if the data changes, such as update or delete, because the pre-computed data has lost detailed information, it cannot be updated synchronously. For example, a sum value of 5 may be 1+4 or 2+3. Because of the loss of detailed information, we cannot distinguish how this summation value is calculated, so we cannot meet the needs of updating.

Q4. tablet writer write failed, tablet_id=27306172, txn_id=28573520, err=-235 or -238

This error usually occurs during data import operations. The error code is -235. The meaning of this error is that the data version of the corresponding tablet exceeds the maximum limit (default 500, controlled by the BE parameter `max_tablet_version_num`), and subsequent writes will be rejected. For example, the error in the question means that the data version of the tablet 27306172 exceeds the limit.

This error is usually caused by the import frequency being too high, which is greater than the compaction speed of the backend data, causing versions to pile up and eventually exceed the limit. At this point, we can first pass the show tablet 27306172 statement, and then execute the show proc statement in the result to check the status of each copy of the tablet. The versionCount in the result represents the number of versions. If you find that a copy has too many versions, you need to reduce the import frequency or stop importing and observe whether the number of versions drops. If the number of versions does not decrease after the import is stopped, you need to go to the corresponding BE node to view the be.INFO log, search for the tablet id and compaction keyword, and check whether the compaction is running normally. For compaction tuning, you can refer to the ApacheDoris official account article: Doris Best Practices - Compaction Tuning (3)

The -238 error usually occurs when the same batch of imported data is too large, resulting in too many Segment files for a tablet (default is 200, controlled by the BE parameter `max_segment_num_per_rowset`). At this time, it is recommended to reduce the amount of data imported in one batch, or appropriately increase the BE configuration parameter value to solve the problem. Since version 2.0, users can enable segment compaction feature to reduce segment file number by setting `enable_segcompaction=true` in BE config.

Q5. tablet 110309738 has few replicas: 1, alive backends: [10003]

This error can occur during a query or import operation. Usually means that the copy of the corresponding tablet has an exception.

At this point, you can first check whether the BE node is down by using the show backends command. For example, the isAlive field is false, or the LastStartTime is a recent time (indicating that it has been restarted recently). If the BE is down, you need to go to the node corresponding to the BE and check the be.out log. If BE is down for abnormal reasons, the exception stack is usually printed in be.out to help troubleshoot the problem. If there is no error stack in be.out. Then you can use the linux command dmesg -T to check whether the process is killed by the system because of OOM.

If no BE node is down, you need to pass the show tablet 110309738 statement, and then execute the show proc statement in the result to check the status of each tablet copy for further investigation.

Q6. disk xxxxx on backend xxx exceed limit usage

Usually occurs in operations such as Import, Alter, etc. This error means that the usage of the corresponding disk corresponding to the BE exceeds the threshold (default 95%). In this case, you can first use the show

backends command, where MaxDiskUsedPct shows the usage of the disk with the highest usage on the corresponding BE. If If it exceeds 95%, this error will be reported.

At this point, you need to go to the corresponding BE node to check the usage in the data directory. The trash directory and snapshot directory can be manually cleaned to free up space. If the data directory occupies a large space, you need to consider deleting some data to free up space. For details, please refer to Disk Space Management.

Q7. Calling stream load to import data through a Java program may result in a Broken Pipe error when a batch of data is large.

Apart from Broken Pipe, some other weird errors may occur.

This situation usually occurs after enabling httpv2. Because httpv2 is an http service implemented using spring boot, and uses tomcat as the default built-in container. However, there seems to be some problems with tomcat's handling of 307 forwarding, so the built-in container was modified to jetty later. In addition, the version of apache http client in the java program needs to use the version after 4.5.13. In the previous version, there were also some problems with the processing of forwarding.

So this problem can be solved in two ways:

1. Disable httpv2

   Restart FE after adding enable_http_server_v2=false in fe.conf. However, the new version of the UI interface can no longer be used, and some new interfaces based on httpv2 can not be used. (Normal import queries are not affected).

2. Upgrade

   Upgrading to Doris 0.15 and later has fixed this issue.

Q8. Error -214 is reported when importing and querying

When performing operations such as import, query, etc., you may encounter the following errors:

```
failed to initialize storage reader. tablet=63416.1050661139.aa4d304e7a7aff9c-f0fa7579928c85a0, res=-214, backend=192.168.100.10
```

A -214 error means that the data version for the corresponding tablet is missing. For example, the above error indicates that the data version of the copy of tablet 63416 on the BE of 192.168.100.10 is missing. (There may be other similar error codes, which can be checked and repaired in the following ways).

Typically, if your data has multiple copies, the system will automatically repair these problematic copies. You can troubleshoot with the following steps:

First, check the status of each copy of the corresponding tablet by executing the `show tablet 63416` statement and executing the `show proc xxx` statement in the result. Usually we need to care about the data in the `Version` column.

Normally, the Version of multiple copies of a tablet should be the same. And it is the same as the VisibleVersion version of the corresponding partition.

You can view the corresponding partition version with `show partitions from tblx` (the partition corresponding to the tablet can be obtained in the `show tablet` statement.)

At the same time, you can also visit the URL in the CompactionStatus column in the `show proc` statement (just open it in a browser) to view more specific version information to check which versions are missing.

If there is no automatic repair for a long time, you need to use the `show proc "/cluster_balance"` statement to view the tablet repair and scheduling tasks currently being executed by the system. It may be because there are a large number of tablets waiting to be scheduled, resulting in a longer repair time. You can follow records in `pending_tablets` and `running_tablets`.

Further, you can use the `admin repair` statement to specify a table or partition to be repaired first. For details, please refer to `help admin repair`;

If it still can't be repaired, then in the case of multiple replicas, we use the `admin set replica status` command to force the replica in question to go offline. For details, see the example of setting the replica status to bad in `help admin set replica status`. (After set to bad, the copy will no longer be accessed. And it will be automatically repaired later. But before operation, you should make sure that other copies are normal)

Q9. Not connected to 192.168.100.1:8060 yet, server_id=384

We may encounter this error when importing or querying. If you go to the corresponding BE log, you may also find similar errors.

This is an RPC error, and there are usually two possibilities: 1. The corresponding BE node is down. 2. rpc congestion or other errors.

If the BE node is down, you need to check the specific downtime reason. Only the problem of rpc congestion is discussed here.

One case is OVERCROWDED, which means that the rpc source has a large amount of unsent data that exceeds the threshold. BE has two parameters associated with it:

1. `brpc_socket_max_unwritten_bytes`: The default value is 1GB. If the unsent data exceeds this value, an error will be reported. This value can be modified appropriately to avoid OVERCROWDED errors. (But this cures the symptoms but not the root cause, and there is still congestion in essence).
2. `tablet_writer_ignore_eovercrowded`: Default is false. If set to true, Doris will ignore OVERCROWDED errors during import. This parameter is mainly to avoid import failure and improve the stability of import.

The second is that the packet size of rpc exceeds max_body_size. This problem may occur if the query has a very large String type, or a bitmap type. It can be circumvented by modifying the following BE parameters:

```
brpc_max_body_size: default 3GB.
```

Q10. [Broker load] org.apache.thrift.transport.TTransportException: java.net.SocketException: Broken pipe

`org.apache.thrift.transport.TTransportException: java.net.SocketException: Broken pipe` during import.

The reason for this problem may be that when importing data from external storage (such as HDFS), because there are too many files in the directory, it takes too long to list the file directory. Here, the Broker RPC Timeout defaults to 10 seconds, and the timeout needs to be adjusted appropriately here. time.

Modify the `fe.conf` configuration file to add the following parameters:

```
broker_timeout_ms = 10000
##The default here is 10 seconds, you need to increase this parameter appropriately
```

Adding parameters here requires restarting the FE service.

Q11. [Routine load] ReasonOfStateChanged: ErrorReason{code=errCode = 104, msg='be 10004 abort task with reason: fetch failed due to requested offset not available on the broker: Broker: Offset out of range'}

The reason for this problem is that Kafka's cleanup policy defaults to 7 days. When a routine load task is suspended for some reason and the task is not restored for a long time, when the task is resumed, the routine load records the consumption offset, and This problem occurs when kafka has cleaned up the corresponding offset

So this problem can be solved with alter routine load:

View the smallest offset of kafka, use the ALTER ROUTINE LOAD command to modify the offset, and resume the task

```
ALTER ROUTINE LOAD FOR db.tb
FROM kafka
(
 "kafka_partitions" = "0",
 "kafka_offsets" = "xxx",
 "property.group.id" = "xxx"
);
```

Q12. ERROR 1105 (HY000): errCode = 2, detailMessage = (192.168.90.91)[CANCELLED][INTERNAL_ERROR]error setting certificate verify locations: CAfile: /etc/ssl/certs/ca-certificates.crt CApath: none

```
yum install -y ca-certificates
ln -s /etc/pki/ca-trust/extracted/openssl/ca-bundle.trust.crt /etc/ssl/certs/ca-certificates.crt
```

Q13. create partition failed. partition numbers will exceed limit variable max_auto_partition_num

To prevent accidental creation of too many partitions when importing data for auto-partitioned tables, we use the FE configuration item `max_auto_partition_num` to control the maximum number of partitions to be created automatically for such tables. If you really need to create more partitions, please modify this config item of FE Master node.

SQL Error

SQL Error

Q1. Query error: Failed to get scan range, no queryable replica found in tablet: xxxx

This happens because the corresponding tablet does not find a copy that can be queried, usually because the BE is down, the copy is missing, etc. You can first pass the `show tablet tablet_id` statement and then execute the following `show proc` statement to view the replica information corresponding to this tablet and check whether the replica is complete. At the same time, you can also query the progress of replica scheduling and repair in the cluster through `show proc "/cluster_balance"` information.

For commands related to data copy management, please refer to Data Copy Management.

Q2. Show backends/frontends The information viewed is incomplete

After executing certain statements such as `show backends/frontends`, some columns may be found to be incomplete in the results. For example, the disk capacity information cannot be seen in the show backends result.

Usually this problem occurs when the cluster has multiple FEs. If users connect to non-Master FE nodes to execute these statements, they will see incomplete information. This is because some information exists only on the Master FE node. For example, BE's disk usage information, etc. Therefore, complete information can only be obtained after a direct connection to the Master FE.

Of course, users can also execute `set forward_to_master=true;` before executing these statements. After the session variable is set to true, some information viewing statements executed subsequently will be automatically forwarded to the Master FE to obtain the results. In this way, no matter which FE the user is connected to, the complete result can be obtained.

Q3. invalid cluster id: xxxx

This error may appear in the results of the show backends or show frontends commands. Usually appears in the error message column of an FE or BE node. The meaning of this error is that after the Master FE sends the heartbeat information to the node, the node finds that the cluster id carried in the heartbeat information is different from the cluster id stored locally, so it refuses to respond to the heartbeat.

The Master FE node of Doris will actively send heartbeats to each FE or BE node, and will carry a cluster_id in the heartbeat information. cluster_id is the unique cluster ID generated by the Master FE when a cluster is initialized. When the FE or BE receives the heartbeat information for the first time, the cluster_id will be saved locally in the form of a file. The file of FE is in the image/ directory of the metadata directory, and the BE has a cluster_id file in all data directories. After that, each time the node receives the heartbeat, it will compare the content of the local cluster_id with the content in the heartbeat. If it is inconsistent, it will refuse to respond to the heartbeat.

This mechanism is a node authentication mechanism to prevent receiving false heartbeat messages sent by nodes outside the cluster.

If needed to recover from this error. The first thing to do is to make sure that all the nodes are in the correct cluster. After that, for the FE node, you can try to modify the cluster_id value in the image/VERSION file in the metadata directory and restart the FE. For the BE node, you can delete all the cluster_id files in the data directory and restart the BE.

Q4. Unique Key model query results are inconsistent

In some cases, when a user uses the same SQL to query a table with a Unique Key model, the results of multiple queries may be inconsistent. And the query results always change between 2-3 kinds.

This may be because, in the same batch of imported data, there are data with the same key but different values, which will lead to inconsistent results between different replicas due to the uncertainty of the sequence of data overwriting.

For example, the table is defined as k1, v1. A batch of imported data is as follows:

```
1, "abc"
1, "def"
```

Then maybe the result of copy 1 is `1, "abc"`, and the result of copy 2 is `1, "def"`. As a result, the query results are inconsistent.

To ensure that the data sequence between different replicas is unique, you can refer to the Sequence Column function.

Q5. The problem of querying bitmap/hll type data returns NULL

In version 1.1.x, when vectorization is enabled, and the bitmap type field in the query data table returns a NULL result,

1. First you have to `set return_object_data_as_binary=true;`
2. Turn off vectorization `set enable_vectorized_engine=false;`
3. Turn off SQL cache `set [global] enable_sql_cache = false;`

This is because the bitmap / hll type is in the vectorized execution engine: the input is all NULL, and the output result is also NULL instead of 0

Q5. The problem of querying bitmap/hll type data returns NULL

In version 1.1.x, when vectorization is turned on, and the bitmp type field in the query data table returns a NULL result,

1. First you have to `set return_object_data_as_binary=true;`
2. Turn off vectorization `set enable_vectorized_engine=false;`
3. Turn off SQL cache `set [global] enable_sql_cache = false;`

This is because the bitmap/hll type is in the vectorized execution engine: the input is all NULL, and the output result is also NULL instead of 0

Q6. Error when accessing object storage: curl 77: Problem with the SSL CA cert

If the `curl 77: Problem with the SSL CA cert` error appears in the be.INFO log. You can try to solve it in the following ways:

1. Download the certificate at https://curl.se/docs/caextract.html: cacert.pem
2. Copy the certificate to the specified location: `sudo cp /tmp/cacert.pem /etc/ssl/certs/ca-certificates.crt`
3. Restart the BE node.

Q7. import error:“Message”: “[INTERNAL_ERROR]single replica load is disabled on BE.”

1. Make sure this parameters `enable_single_replica_load` in be.conf is set true
2. Restart the BE node.

Data Lakehouse FAQ

Certificate Issues

1. When querying, an error `curl 77: Problem with the SSL CA cert.` occurs. This indicates that the current system certificate is too old and needs to be updated locally.
    - You can download the latest CA certificate from `https://curl.haxx.se/docs/caextract.html`.
    - Place the downloaded `cacert-xxx.pem` into the `/etc/ssl/certs/` directory, for example: `sudo cp cacert-xxx.pem /etc/ssl/certs/ca-certificates.crt`.
2. When querying, an error occurs: `ERROR 1105 (HY000): errCode = 2, detailMessage = (x.x.x.x)[CANCELLED][INTERNAL_ERROR]error setting certificate verify locations: CAfile: /etc/ssl/certs/ca-certificates.crt CApath: none`.

```
yum install -y ca-certificates
ln -s /etc/pki/ca-trust/extracted/openssl/ca-bundle.trust.crt /etc/ssl/certs/ca-certificates.crt
```

Kerberos

1. When connecting to a Hive Metastore authenticated with Kerberos, an error `GSS initiate failed` is encountered.

    This is usually due to incorrect Kerberos authentication information. You can troubleshoot by following these steps:

    1. In versions prior to 1.2.1, the libhdfs3 library that Doris depends on did not enable gsasl. Please update to versions 1.2.2 and later.

    2. Ensure that correct keytab and principal are set for each component and verify that the keytab file exists on all FE and BE nodes.

        - `hadoop.kerberos.keytab`/`hadoop.kerberos.principal`: Used for Hadoop hdfs access, fill in the corresponding values for hdfs.
        - `hive.metastore.kerberos.principal`: Used for hive metastore.

    3. Try replacing the IP in the principal with a domain name (do not use the default `_HOST` placeholder).

    4. Ensure that the `/etc/krb5.conf` file exists on all FE and BE nodes.

2. When connecting to a Hive database through the Hive Catalog, an error occurs: `RemoteException: SIMPLE authentication is not enabled. Available:[TOKEN, KERBEROS]`.

    If the error occurs during the query when there are no issues with `show databases` and `show tables`, follow these two steps:

    - Place core-site.xml and hdfs-site.xml in the fe/conf and be/conf directories.
    - Execute Kerberos kinit on the BE node, restart BE, and then proceed with the query.

When encountering the error `GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos Ticket)` while querying a table configured with Kerberos, restarting FE and BE nodes usually resolves the issue.

- Before restarting all nodes, configure `-Djavax.security.auth.useSubjectCredsOnly=false` in the JAVA_OPTS parameter in "`${DORIS_HOME}/be/conf/be.conf`" to obtain JAAS credentials information through the underlying mechanism rather than the application.
- Refer to JAAS Troubleshooting for solutions to common JAAS errors.

To resolve the error `Unable to obtain password from user` when configuring Kerberos in the Catalog:

- Ensure the principal used is listed in klist by checking with `klist -kt your.keytab`.
- Verify the catalog configuration for any missing settings such as `yarn.resourcemanager.principal`.
- If the above checks are fine, it may be due to the JDK version installed by the system's package manager not supporting certain encryption algorithms. Consider installing JDK manually and setting the `JAVA_HOME` environment variable.
- Kerberos typically uses AES-256 for encryption. For Oracle JDK, JCE must be installed. Some distributions of OpenJDK automatically provide unlimited strength JCE, eliminating the need for separate installation.
- JCE versions correspond to JDK versions; download the appropriate JCE zip package and extract it to the `$JAVA_HOME/jre/lib/security` directory based on the JDK version:
  - JDK6: JCE6
  - JDK7: JCE7
  - JDK8: JCE8

When encountering the error `java.security.InvalidKeyException: Illegal key size` while accessing HDFS with KMS, upgrade the JDK version to >= Java 8 u162 or install the corresponding JCE Unlimited Strength Jurisdiction Policy Files.

If configuring Kerberos in the Catalog results in the error `SIMPLE authentication is not enabled. Available:[TOKEN, KERBEROS]`, place the `core-site.xml` file in the "`${DORIS_HOME}/be/conf`" directory.

If accessing HDFS results in the error `No common protection layer between client and server`, ensure that the `hadoop.rpc.protection` properties on the client and server are consistent.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

    <property>
        <name>hadoop.security.authentication</name>
        <value>kerberos</value>
    </property>

</configuration>
```

When using Broker Load with Kerberos configured and encountering the error `Cannot locate default realm.`:

Add the configuration item `-Djava.security.krb5.conf=/your-path` to the `JAVA_OPTS` in the `start_broker.sh` script for Broker Load.

3. When using Kerberos configuration in the Catalog, the `hadoop.username` property cannot be used simultaneously.

4. Accessing Kerberos with JDK 17

   When running Doris with JDK 17 and accessing Kerberos services, you may encounter issues accessing due to the use of deprecated encryption algorithms. You need to add the `allow_weak_crypto=true` property in krb5.conf or upgrade the encryption algorithm in Kerberos.

   For more details, refer to: https://seanjmullan.org/blog/2021/09/14/jdk17#kerberos

JDBC Catalog

1. Error connecting to SQLServer via JDBC Catalog: `unable to find valid certification path to requested target`

   Add the `trustServerCertificate=true` option in the `jdbc_url`.

2. Connecting to MySQL database via JDBC Catalog results in Chinese character garbling or incorrect Chinese character query conditions

   Add `useUnicode=true&characterEncoding=utf-8` in the `jdbc_url`.

   > Note: Starting from version 1.2.3, when connecting to MySQL database via JDBC Catalog, these parameters will be automatically added.

3. Error connecting to MySQL database via JDBC Catalog: `Establishing SSL connection without server's identity verification is not recommended`

   Add `useSSL=true` in the `jdbc_url`.

4. When synchronizing MySQL data to Doris using JDBC Catalog, date data synchronization error occurs. Verify if the MySQL version matches the MySQL driver package, for example, MySQL 8 and above require the driver com.mysql.cj.jdbc.Driver.

5. When a single field is too large, a Java memory OOM occurs on the BE side during a query.

   When Jdbc Scanner reads data through JDBC, the session variable `batch_size` determines the number of rows processed in the JVM per batch. If a single field is too large, it may cause `field_size * batch_size` (approximate value, considering JVM static memory and data copy overhead) to exceed the JVM memory limit, resulting in OOM.

   Solutions:

   - Reduce the `batch_size` value by executing `set batch_size = 512;`. The default value is 4064.
   - Increase the BE JVM memory by modifying the `-Xmx` parameter in `JAVA_OPTS`. For example: `-Xmx8g`.

Hive Catalog

1. Error accessing Iceberg table via Hive Metastore: `failed to get schema` or `Storage schema reading not supported`

   Place the relevant `iceberg` runtime jar files in Hive's lib/ directory.

   Configure in `hive-site.xml`:

   ```
   metastore.storage.schema.reader.impl=org.apache.hadoop.hive.metastore.SerDeStorageSchemaReader
   ```

   After configuration, restart the Hive Metastore.

2. Error connecting to Hive Catalog: `Caused by: java.lang.NullPointerException`

   If the fe.log contains the following stack trace:

   ```
   Caused by: java.lang.NullPointerException
       at
   org.apache.hadoop.hive.ql.security.authorization.plugin.AuthorizationMetaStoreFilterHook.getFilteredObjects(AuthorizationMe
   taStoreFilterHook.java:78) ~[hive-exec-3.1.3-core.jar:3.1.3]
       at
   org.apache.hadoop.hive.ql.security.authorization.plugin.AuthorizationMetaStoreFilterHook.filterDatabases(AuthorizationMetaS
   toreFilterHook.java:55) ~[hive-exec-3.1.3-core.jar:3.1.3]
       at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.getAllDatabases(HiveMetaStoreClient.java:1548) ~[doris-
   fe.jar:3.1.3]
       at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.getAllDatabases(HiveMetaStoreClient.java:1542) ~[doris-
   fe.jar:3.1.3]
       at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[?:1.8.0_181]
   ```

   Try adding `"metastore.filter.hook" = "org.apache.hadoop.hive.metastore.DefaultMetaStoreFilterHookImpl"` in the `create catalog` statement to resolve.

3. If after creating Hive Catalog, `show tables` works fine but querying results in `java.net.UnknownHostException: xxxxx`

   Add the following in the CATALOG's PROPERTIES:

   ```
   'fs.defaultFS' = 'hdfs://<your_nameservice_or_actually_HDFS_IP_and_port>'
   ```

4. Tables in orc format in Hive 1.x may encounter system column names in the underlying orc file schema as `_col0`, `_col1`, `_col2`, etc. In this case, add `hive.version` as 1.x.x in the catalog configuration to map with the column names in the hive table.

   ```
   CREATE CATALOG hive PROPERTIES (
       'hive.version' = '1.x.x'
   );
   ```

5. When querying table data using Catalog, errors related to Hive Metastore such as `Invalid method name` are encountered, set the `hive.version` parameter.

6. When querying a table in ORC format, if the FE reports `Could not obtain block` or `Caused by: java.lang.NoSuchFieldError: types`, it may be due to the FE accessing HDFS to retrieve file information and perform file splitting by default. In some cases, the FE may not be able to access HDFS. This can be resolved by adding the following parameter: `"hive.exec.orc.split.strategy" = "BI"`. Other options include HYBRID (default) and ETL.

7. In Hive, you can find the partition field values of a Hudi table, but in Doris, you cannot. Doris and Hive currently have different ways of querying Hudi. In Doris, you need to add the partition fields in the avsc file structure of the Hudi table. If not added, Doris will query with partition_val being empty (even if `hoodie.datasource.hive_sync.partition_fields=partition_val` is set).

```
{
    "type": "record",
    "name": "record",
    "fields": [{
        "name": "partition_val",
        "type": [
            "null",
            "string"
            ],
        "doc": "Preset partition field, empty string when not partitioned",
        "default": null
        },
        {
        "name": "name",
        "type": "string",
        "doc": "Name"
        },
        {
        "name": "create_time",
        "type": "string",
        "doc": "Creation time"
        }
    ]
}
```

8. When querying a Hive external table, if you encounter the error `java.lang.ClassNotFoundException: Class com.hadoop.compression.lzo.LzoCodec not found`, search for `hadoop-lzo-*.jar` in the Hadoop environment, place it in the `"${DORIS_HOME}/fe/lib/"` directory, and restart the FE. Starting from version 2.0.2, you can place this file in the `custom_lib/` directory of the FE (if it does not exist, create it manually) to prevent file loss when upgrading the cluster due to the lib directory being replaced.

9. When creating a Hive table specifying the serde as `org.apache.hadoop.hive.contrib.serde2.MultiDelimitserDe`, and encountering the error `storage schema reading not supported` when accessing the table, add the following configuration to the hive-site.xml file and restart the HMS service:

```
<property>
  <name>metastore.storage.schema.reader.impl</name>
  <value>org.apache.hadoop.hive.metastore.SerDeStorageSchemaReader</value>
</property>
```

10. Error: `java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty`. The complete error message in the FE log is as follows:

```
org.apache.doris.common.UserException: errCode = 2, detailMessage = S3 list path failed. path=s3://bucket/part-*,msg=errors
while get file status listStatus on s3://bucket: com.amazonaws.SdkClientException: Unable to execute HTTP request:
Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty: Unable to
execute HTTP request: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must
be non-empty
org.apache.doris.common.UserException: errCode = 2, detailMessage = S3 list path exception. path=s3://bucket/part-*, err:
errCode = 2, detailMessage = S3 list path failed. path=s3://bucket/part-*,msg=errors while get file status listStatus on
s3://bucket: com.amazonaws.SdkClientException: Unable to execute HTTP request: Unexpected error:
java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty: Unable to execute HTTP
request: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty
org.apache.hadoop.fs.s3a.AWSClientIOException: listStatus on s3://bucket: com.amazonaws.SdkClientException: Unable to
execute HTTP request: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must
be non-empty: Unable to execute HTTP request: Unexpected error: java.security.InvalidAlgorithmParameterException: the
trustAnchors parameter must be non-empty
Caused by: com.amazonaws.SdkClientException: Unable to execute HTTP request: Unexpected error:
java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty
Caused by: javax.net.ssl.SSLException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors
parameter must be non-empty
Caused by: java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors
parameter must be non-empty
Caused by: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty
```

Try updating the CA certificate on the FE node using `update-ca-trust (CentOS/RockyLinux)`, and then restart the FE process.

11. BE error: `java.lang.InternalError`. If you see an error similar to the following in `be.INFO`:

```
W20240506 15:19:57.553396 266457 jni-util.cpp:259] java.lang.InternalError
        at org.apache.hadoop.io.compress.zlib.ZlibDecompressor.init(Native Method)
        at org.apache.hadoop.io.compress.zlib.ZlibDecompressor.<init>(ZlibDecompressor.java:114)
        at org.apache.hadoop.io.compress.GzipCodec$GzipZlibDecompressor.<init>(GzipCodec.java:229)
        at org.apache.hadoop.io.compress.GzipCodec.createDecompressor(GzipCodec.java:188)
        at org.apache.hadoop.io.compress.CodecPool.getDecompressor(CodecPool.java:183)
        at org.apache.parquet.hadoop.CodecFactory$HeapBytesDecompressor.<init>(CodecFactory.java:99)
        at org.apache.parquet.hadoop.CodecFactory.createDecompressor(CodecFactory.java:223)
        at org.apache.parquet.hadoop.CodecFactory.getDecompressor(CodecFactory.java:212)
        at org.apache.parquet.hadoop.CodecFactory.getDecompressor(CodecFactory.java:43)
```

It is because the Doris built-in `libz.a` conflicts with the system environment's `libz.so`. To resolve this issue, first execute `export LD_LIBRARY_PATH=/path/to/be/lib:$LD_LIBRARY_PATH`, and then restart the BE process.

HDFS

1. When accessing HDFS 3.x, if you encounter the error `java.lang.VerifyError: xxx`, in versions prior to 1.2.1, Doris depends on Hadoop version 2.8. You need to update to 2.10.2 or upgrade Doris to versions after 1.2.2.

2. Using Hedged Read to optimize slow HDFS reads. In some cases, high load on HDFS may lead to longer read times for data replicas on a specific HDFS, thereby slowing down overall query efficiency. The HDFS Client provides the Hedged Read feature. This feature initiates another read thread to read the same data if a read request exceeds a certain threshold without returning, and the result returned first is used.

   Note: This feature may increase the load on the HDFS cluster, so use it judiciously.

You can enable this feature in two ways:

- Specify it in the parameters when creating the Catalog:

```
create catalog regression properties (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.16.47:7004',
    'dfs.client.hedged.read.threadpool.size' = '128',
    'dfs.client.hedged.read.threshold.millis' = "500"
);
```

`dfs.client.hedged.read.threadpool.size` represents the number of threads used for Hedged Read, which are shared by an HDFS Client. Typically, for an HDFS cluster, BE nodes will share an HDFS Client.

`dfs.client.hedged.read.threshold.millis` is the read threshold in milliseconds. When a read request exceeds this threshold without returning, a Hedged Read is triggered.

When enabled, you can see the related parameters in the Query Profile:

`TotalHedgedRead`: Number of times Hedged Read was initiated.

`HedgedReadWins`: Number of successful Hedged Reads (times when the request was initiated and returned faster than the original request)

Note that these values are cumulative for a single HDFS Client, not for a single query. The same HDFS Client can be reused by multiple queries.

3. `Couldn't create proxy provider class org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider`

   In the start scripts of FE and BE, the environment variable `HADOOP_CONF_DIR` is added to the CLASSPATH. If `HADOOP_CONF_DIR` is set incorrectly, such as pointing to a non-existent or incorrect path, it may load the wrong xxx-site.xml file, resulting in reading incorrect information.

   Check if `HADOOP_CONF_DIR` is configured correctly or remove this environment variable.

4. `BlockMissingExcetpion: Could not obtain block: BP-XXXXXXXXX No live nodes contain current block`

   Possible solutions include:

   - Use `hdfs fsck file -files -blocks -locations` to check if the file is healthy.
   - Check connectivity with datanodes using `telnet`.
   - Check datanode logs.

   If you encounter the following error:

   `org.apache.hadoop.hdfs.server.datanode.DataNode: Failed to read expected SASL data transfer protection handshake from client at /XXX.XXX.XXX.XXX:XXXXX. Perhaps the client is running an older version of Hadoop which does not support SASL data transfer protection` it means that the current hdfs has enabled encrypted transmission, but the client has not, causing the error.

   Use any of the following solutions:

   - Copy hdfs-site.xml and core-site.xml to be/conf and fe/conf directories. (Recommended)

- In hdfs-site.xml, find the corresponding configuration `dfs.data.transfer.protection` and set this parameter in the catalog.

DLF Catalog

1. When using the DLF Catalog, if `Invalid address` occurs during BE reading JindoFS data, add the domain name appearing in the logs to IP mapping in `/etc/hosts`.
2. If there is no permission to read data, use the `hadoop.username` property to specify a user with permission.
3. The metadata in the DLF Catalog should be consistent with DLF. When managing metadata using DLF, newly imported partitions in Hive may not be synchronized by DLF, leading to inconsistencies between DLF and Hive metadata. To address this, ensure that Hive metadata is fully synchronized by DLF.

Other Issues

1. Query results in garbled characters after mapping Binary type to Doris

   Doris natively does not support the Binary type, so when mapping Binary types from various data lakes or databases to Doris, it is usually done using the String type. The String type can only display printable characters. If you need to query the content of Binary data, you can use the `TO_BASE64()` function to convert it to Base64 encoding before further processing.

2. Analyzing Parquet files

   When querying Parquet files, due to potential differences in the format of Parquet files generated by different systems, such as the number of RowGroups, index values, etc., sometimes it is necessary to check the metadata of Parquet files for issue identification or performance analysis. Here is a tool provided to help users analyze Parquet files more conveniently:

   1. Download and unzip Apache Parquet Cli 1.14.0
   2. Download the Parquet file to be analyzed to your local machine, assuming the path is `/path/to/file.parquet`
   3. Use the following command to analyze the metadata of the Parquet file:

      ```
      ./parquet-tools meta /path/to/file.parquet
      ```

   4. For more functionalities, refer to Apache Parquet Cli documentation

BI FAQ

Power BI

Q1. An error occurs when you use JDBC to pull data into Desktop Power BI. “Timeout expired. The timeout period elapsed prior to completion of the operation or the server is not responding”.

Usually, this is Power BI pulling the time timeout of the data source. When filling in the data source server and database, click the advanced option, which has a timeout time, set the time higher.

Q2. When the 2.1.x version uses JDBC to connect to Power BI, an error occurs “An error happened while reading data from the provider: the given key was not present in the dictionary”.

Run “show collation” in the database first. Generally, only utf8mb4_900_bin is displayed, and the charset is utf8mb4. The main reason for this error is that ID 33 needs to be found when connecting to Power BI. That is, rows with 33ids in the table need to be upgraded to version 2.1.5 or later.

Q3. Connection Doris Times error “Reading data from the provider times error index and count must refer to the location within the string”.

The cause of the problem is that global parameters are loaded during the connection process, and the SQL column names and values are the same

```
SELECT
@@max_allowed_packet  as max_allowed_packet, @@character_set_client ,@@character_set_connection ,
@@license,@@sql_mode ,@@lower_case_table_names , @@autocommit ;
```

The new optimizer can be turned off in the current version or upgraded to version 2.0.7 or 2.1.6 or later.

Q4. JDBC connection version 2.1.x error message “Character set ‘utf8mb3’ is not supported by.net.Framework”.

This problem is easily encountered in version 2.1.x. If this problem occurs, you need to upgrade the JDBC Driver to 8.0.32.

Tableau

Q1. Version 2.0.x reports that Tableau cannot connect to the data source with error code 37CE01A3.

Turn off the new optimizer in the current version or upgrade to 2.0.7 or later

Q2. SSL connection error:protocol version mismatch Failed to connect to the MySQL server

The cause of this error is that SSL authentication is enabled on Doris, but SSL connections are not used during the connection. You need to disable the enable_ssl variable in fe.conf.

Q3. Connection error Unsupported command(COM_STMT_PREPARED)

The MySQL driver version is improperly installed. Install the MySQL 5.1.x connection driver instead.

Data Integrity FAQ

Data Integrity Issues

This document is mainly used to record common data integrity issues encountered during the use of Doris. It will be updated periodically.

The term “duplicate key data in the table” refers to the appearance of duplicate key data in the merge-on-write Unique table. Duplicate key issues in merge-on-write Unique tables can be fixed by triggering a full compaction, while other types of integrity issues may require specific solutions based on the situation. For assistance, please contact community support.

Issue Description	Occurrence Conditions	Affected Versions	Fix Versions	Impacted Scope	Fix PR
Stream Load import does not delete data that meets the `delete` condition in merge-on-write Unique table	Stream Load import with `merge_type: MERGE`, `partial_columns: true`, and `delete` parameters	<2.0.15	>=2.0.15	Partial Column Update	#40730
Duplicate key data in the table	User adds sequence column functionality to a merge-on-write Unique table that does not support sequence columns using `ALTER TABLE tbl ENABLE FEATURE "SEQUENCE_LOAD" WITH ...`, followed by new imports	<2.0.15	>=2.0.15		#39958
Partial column update import results in data corruption in merge-on-write Unique table	Concurrent partial column updates in merge-on-write Unique table, with BE restart during the import process	<2.0.15	>=2.0.15	Partial Column Update	#38331
Duplicate key data in the table	Large single import in merge-on-write Unique table with sequence column triggers segment compaction	<2.0.15	>=2.0.15		#38369
Duplicate key data in the table	Failed full clone on merge-on-write Unique table	<2.0.13	>=2.0.13		#37001

Issue Description	Occurrence Conditions	Affected Versions	Fix Versions	Impacted Scope	Fix PR
Inconsistent multi-replica data in merge-on-write Unique table	Partial column update import with `__DORIS_DELETE_SIGN__` column on merge-on-write Unique table, and inconsistent Base Compaction progress across replicas during import	<2.0.15	>=2.0.15	Partial Column Update	#36210
Duplicate key data in the table	Concurrent partial column updates and BE restart during import on merge-on-write Unique table	<2.0.11	>=2.0.11	Partial Column Update	#35739

Overview

Numeric Types

Doris supports the following numeric data types:

BOOLEAN

There are two possible values: 0 represents false, and 1 represents true.

For more info, please refer BOOLEAN。

Integer

All are signed integers. The differences among the INT types are the number of bytes occupied and the range of values they can represent:

- **TINYINT**: 1 byte, [-128, 127]
- **SMALLINT**: 2 bytes, [-32768, 32767]
- **INT**: 4 bytes, [-2147483648, 2147483647]
- **BIGINT**: 8 bytes, [-9223372036854775808, 9223372036854775807]
- **LARGEINT**: 16 bytes, [-2^127, 2^127 - 1]

Floating-point

Including imprecise floating-point types FLOAT and DOUBLE, corresponding to the `float` and `double` in common programming languages

Fixed-point

The precise fixed-point type DECIMAL, used in financial and other cases that require strict accuracy.

Date Types

Date types include DATE, TIME and DATETIME, DATE type only stores the date accurate to the day, DATETIME type stores the date and time, which can be accurate to microseconds. TIME type only stores the time, and **does not support the construction of the table storage for the time being, can only be used in the query process**.

Do calculation for datetime types or converting them to numeric types, please use functions like TIME_TO_SEC, DATE_DIFF, UNIX_TIMESTAMP . The result of directly converting them as numeric types as not guaranteed.

For more information refer to DATE, TIME and DATETIME documents.

String Types

Doris supports both fixed-length and variable-length strings, including:

- **CHAR(M)**: A fixed-length string, where M is the byte length. The range for M is [1, 255].
- **VARCHAR(M)**: A variable-length string, where M is the maximum length. The range for M is [1, 65533].
- **STRING**: A variable-length string with a default maximum length of 1,048,576 bytes (1 MB). This maximum length can be increased up to 2,147,483,643 bytes (2 GB) by configuring the `string_type_length_soft_limit_bytes`setting.

Semi-Structured Types

Doris supports different semi-structured data types for JSON data processing, each tailored to different use cases.

- **ARRAY** / **MAP** / **STRUCT**: They support nested data and fixed schema, making them well-suited for analytical workloads such as user behavior and profile analysis, as well as querying data lake formats like Parquet. Due to the fixed schema, there is no overhead for dynamic schema inference, resulting in high write and analysis performance.
- **VARIANT**: It supports nested data and flexible schema. It is well-suited for analytical workloads such as log, trace, and IoT data analysis. It can accommodate any legal JSON data, which will be automatically expanded into sub-columns in a columnar storage format. This approach enables high compression rate in storage and high performance in data aggregation, filtering, and sorting.
- **JSON**: It supports nested data and flexible schema. It is optimized for high-concurrency point query use cases. The flexible schema allows for ingesting any legal JSON data, which will be stored in a binary format. Extracting fields from this binary JSON format is more than 2X faster than using regular JSON strings.

Aggregation Types

The aggregation data types store aggregation results or intermediate results during aggregation. They are used for accelerating aggregation-heavy queries.

- **BITMAP**: It is used for exact deduplication, such as in (UV) statistics and audience segmentation. It works in conjunction with BITMAP functions like `bitmap_union`, `bitmap_union_count`, `bitmap_hash`, and `bitmap_hash64`.
- **HLL**: It is used for approximate deduplication and provides better performance than `COUNT DISTINCT`. It works in conjunction with HLL functions like `hll_union_agg`, `hll_raw_agg`, `hll_cardinality`, and `hll_hash`.
- **QUANTILE_STATE**: It is used for approximate percentile calculations and offers better performance than the `PERCENTILE` function. It works with functions like `QUANTILE_PERCENT`, `QUANTILE_UNION`, and `TO_QUANTILE_STATE`.
- **AGG_STATE**: It is used to accelerate aggregations, utilized in combination with aggregation function combinators like state/merge/union.

BOOLEAN

BOOLEAN

Description

BOOL, BOOLEAN Like TINYINT, 0 stands for false and 1 for true.

keywords

BOOLEAN

TINYINT

TINYINT

Description

TINYINT 1 byte signed integer, range [-128, 127]

keywords

TINYINT

SMALLINT

SMALLINT

Description

SMALLINT 2-byte signed integer, range [-32768, 32767]

keywords

SMALLINT

INT

INT

Description

INT 4-byte signed integer, range [-2147483648, 2147483647]

keywords

INT

BIGINT

BIGINT

Description

BIGINT 8-byte signed integer, range [-9223372036854775808, 9223372036854775807]

keywords

BIGINT

LARGEINT

LARGEINT

Description

LARGEINT 16-byte signed integer, range [-2^127 + 1 ~ 2^127 - 1]

keywords

BIGINT

FLOAT

FLOAT

Description

FLOAT 4-byte floating point number

keywords

FLOAT

DOUBLE

DOUBLE

Description

DOUBLE 8-byte floating point number

keywords

DOUBLE

DECIMAL

DECIMAL

DECIMAL

Description

DECIMAL (M [,D])

High-precision fixed-point number, M represents the total number of significant digits, and D represents the scale.

The range of M is [1, 38], and the range of D is [0, precision].

The default value is DECIMAL(9, 0).

Precision Deduction

DECIMAL has a very complex set of type inference rules. For different expressions, different rules will be applied for precision inference.

Arithmetic Expressions

- Plus / Minus: DECIMAL(a, b) + DECIMAL(x, y) -> DECIMAL(max(a - b, x - y) + max(b, y) + 1, max(b, y)).
- Multiply: DECIMAL(a, b) + DECIMAL(x, y) -> DECIMAL(a + x, b + y).
- Divide: DECIMAL(p1, s1) + DECIMAL(p2, s2) -> DECIMAL(p1 + s2 + div_precision_increment, s1 + div_precision_increment).div_precision_increment default 4. It is worth noting that the process of division calculation is as follows: DECIMAL(p1, s1) / DECIMAL(p2, s2) is first converted to DECIMAL(p1 + s2 + div_precision_increment, s1 + s2) / DECIMAL(p2, s2) and then the calculation is performed. Therefore, it is possible that DECIMAL(p1 + s2 + div_precision_increment, s1 + div_precision_increment) satisfies the range of DECIMAL, but due to the conversion to DECIMAL(p1 + s2 + div_precision_increment, s1 + s2), it exceeds the range. Currently, Doris handles this by converting it to Double for calculation. ##### Aggregation functions
- SUM / MULTI_DISTINCT_SUM: SUM(DECIMAL(a, b)) -> DECIMAL(38, b).
- AVG: AVG(DECIMAL(a, b)) -> DECIMAL(38, max(b, 4)).

Default rules

Except for the expressions mentioned above, other expressions use default rules for precision deduction. That is, for the expression `expr(DECIMAL(a, b))`, the result type is also DECIMAL(a, b).

Adjust the result precision

Different users have different accuracy requirements for DECIMAL. The above rules are the default behavior of Doris. If users **have different accuracy requirements, they can adjust the accuracy in the following ways**:

- If the expected result precision is greater than the default precision, you can adjust the result precision by adjusting the parameter's precision. For example, if the user expects to calculate `AVG(col)` and get DECIMAL(x, y) as the result, where the type of `col` is DECIMAL (a, b), the expression can be rewritten to `AVG(CAST(col as DECIMAL (x, y))`.
- If the expected result precision is less than the default precision, the desired precision can be obtained by approximating the output result. For example, if the user expects to calculate `AVG(col)` and get DECIMAL(x, y) as the result, where the type of `col` is DECIMAL(a, b), the expression can be rewritten as `ROUND(AVG(col), y)`.

Why DECIMAL is required

DECIMAL in Doris is a real high-precision fixed-point number. Decimal has the following core advantages: 1. It can represent a wider range. The value ranges of both precision and scale in DECIMAL have been significantly expanded. 2. Higher performance. The old version of DECIMAL requires 16 bytes in memory and 12 bytes in storage, while DECIMAL has made adaptive adjustments as shown below.

```
+----------------------+-----------------------------+
|     precision        | Space occupied (memory/disk) |
+----------------------+-----------------------------+
| 0 < precision <= 9   |            4 bytes           |
+----------------------+-----------------------------+
| 9 < precision <= 18  |            8 bytes           |
+----------------------+-----------------------------+
| 18 < precision <= 38 |           16 bytes           |
+----------------------+-----------------------------+
```

3. More complete precision deduction. For different expressions, different precision inference rules are applied to deduce the precision of the results.

keywords

DECIMAL

DATE

DATE

DATE

Description

Syntax

date Date type, the current range of values is [‘0000-01-01’,‘9999-12-31’], and the default print form is ‘yyyy-MM-dd’.

example

```
SELECT DATE('2003-12-31 01:02:03');
+----------------------------+
| DATE('2003-12-31 01:02:03') |
+----------------------------+
| 2003-12-31                 |
+----------------------------+
```

keywords

DATE

TIME

TIME

name

TIME

description

TIME type Time type, can appear as a query result, does not support table storage for the time being. The storage range is `[-838:59:59, 838:59:59]`. Currently in Doris, the correctness of TIME as a result of calculations is guaranteed (e.g., functions such as `timediff`), but **manual CAST generation of the TIME type is not recommended**. The calculation of TIME type in constant folding is prohibited.

example

```
mysql> select timediff('2020-01-01 12:05:03', '2020-01-01 08:02:15');
+-----------------------------------------------------------------------------------------------------+
| timediff(cast('2020-01-01 12:05:03' as DATETIMEV2(0)), cast('2020-01-01 08:02:15' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------------------+
| 04:02:48                                                                                            |
+-----------------------------------------------------------------------------------------------------+
1 row in set (0.12 sec)

mysql> select timediff('2020-01-01', '2000-01-01');
+-------------------------------------------------------------------------------+
| timediff(cast('2020-01-01' as DATETIMEV2(0)), cast('2000-01-01' as DATETIMEV2(0))) |
+-------------------------------------------------------------------------------+
| 838:59:59                                                                     |
+-------------------------------------------------------------------------------+
1 row in set (0.11 sec)
```

keywords

```
TIME
```

DATETIME

DATETIME

DATETIME

Description

DATETIME([P]) Date and time type. The optional parameter P indicates the time precision and the value range is [0, 6], that is, it supports up to 6 decimal places (microseconds). 0 when not set. Value range is [‘0000-01-01 00:00:00[.000000]’,‘9999-12-31 23:59:59[.999999]’]. The form of printing is ‘yyyy-MM-dd HH:mm:ss.SSSSSS’

note

DATETIME supports temporal precision up to microseconds. When parsing imported DATETIME type data using the BE side (e.g. using Stream load, Spark load, etc.), or using the FE side with the Nereids on, decimals exceeding the current precision will be **rounded**. Inserting a DATETIME value with a fractional seconds part into a column of the same type but having fewer fractional digits results in **rounded**. DATETIME reads support resolving the time zone in the format of the original DATETIME literal followed by the time zone:

```
<date> <time>[<timezone>]
```

For the specific supported formats for `<timezone>`, see timezone. Note that the `DATE`, `DATEV2`, `DATETIME`, and `DATETIMEV2` types **don’t** contain time zone information. For example, if an input time string “2012-12-12 08:00:00+08:00” is parsed and converted to the current time zone “+02:00”, and the actual value “2012-12-12 02:00:00” is stored in the DATETIME column, the value itself will not change, no matter how much the cluster environment variables are changed.

example

```
mysql> select @@time_zone;
+----------------+
| @@time_zone    |
+----------------+
| Asia/Hong_Kong |
+----------------+
1 row in set (0.11 sec)

mysql> insert into dtv23 values ("2020-12-12 12:12:12Z"), ("2020-12-12 12:12:12GMT"), ("2020-12-12 12:12:12+02:00"), ("2020-12-12
12:12:12America/Los_Angeles");
Query OK, 4 rows affected (0.17 sec)

mysql> select * from dtv23;
+-------------------------+
| k0                      |
+-------------------------+
| 2020-12-12 20:12:12.000 |
| 2020-12-12 20:12:12.000 |
| 2020-12-13 04:12:12.000 |
| 2020-12-12 18:12:12.000 |
```

```
+-------------------------+
4 rows in set (0.15 sec)
```

CHAR

CHAR

Description

CHAR(M) A fixed-length string, M represents the byte length of a fixed-length string. The range of M is 1-255.

keywords

CHAR

VARCHAR

VARCHAR

Description

VARCHAR(M) A variable length string, M represents the byte length of a variable length string. The range of M is 1-65533.

Note: Variable length strings are stored in UTF-8 encoding, so usually English characters occupies 1 byte, and Chinese characters occupies 3 bytes.

keywords

VARCHAR

STRING

STRING

Description

STRING (M) A variable length string. Default support is 1048576 bytes (1M), adjustable up to 2147483643 bytes (2G),and the length of the String type is also limited by the configuration string_type_length_soft_limit_bytes(a soft limit of string type length) of be. the String type can only be used in the value column, not in the key column and the partition and bucket columns

Note: Variable length strings are stored in UTF-8 encoding, so usually English characters occupies 1 byte, and Chinese characters occupies 3 bytes.

keywords

STRING

ARRAY

ARRAY

ARRAY

description

ARRAY<T>

An array of T-type items, it cannot be used as a key column. Now ARRAY can only used in Duplicate Model Tables.

After version 2.0, it supports the use of non-key columns in Unique model tables.

T-type could be any of:

```
BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, LARGEINT, FLOAT, DOUBLE, DECIMAL, DATE,
DATEV2, DATETIME, DATETIMEV2, CHAR, VARCHAR, STRING
```

example

Create table example:

```
mysql> CREATE TABLE `array_test` (
  `id` int(11) NULL COMMENT "",
  `c_array` ARRAY<int(11)> NULL COMMENT ""
) ENGINE=OLAP
DUPLICATE KEY(`id`)
COMMENT "OLAP"
DISTRIBUTED BY HASH(`id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
"in_memory" = "false",
"storage_format" = "V2"
);
```

Insert data example:

```
mysql> INSERT INTO `array_test` VALUES (1, [1,2,3,4,5]);
mysql> INSERT INTO `array_test` VALUES (2, [6,7,8]), (3, []), (4, null);
```

Select data example:

```
mysql> SELECT * FROM `array_test`;
+------+-----------------+
| id   | c_array         |
+------+-----------------+
|    1 | [1, 2, 3, 4, 5] |
|    2 | [6, 7, 8]       |
```

```
|    3 | []              |
|    4 | NULL            |
+------+-----------------+
```

keywords

```
ARRAY
```

MAP

MAP

name

MAP

description

`MAP<K, V>`

A Map of K, V items, it cannot be used as a key column. Now MAP can only used in Duplicate and Unique Model Tables.

Need to manually enable the support, it is disabled by default.

```
admin set frontend config("enable_map_type" = "true");
```

K,V could be any of:

```
BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, LARGEINT, FLOAT, DOUBLE, DECIMAL, DECIMALV3, DATE,
DATEV2, DATETIME, DATETIMEV2, CHAR, VARCHAR, STRING
```

example

Create table example:

```
CREATE TABLE IF NOT EXISTS test.simple_map (
  `id` INT(11) NULL COMMENT "",
  `m` Map<STRING, INT> NULL COMMENT ""
) ENGINE=OLAP
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
"storage_format" = "V2"
);
```

Insert data example:

```
mysql> INSERT INTO simple_map VALUES(1, {'a': 100, 'b': 200});
```

stream_load examples: See STREAM LOAD for syntax details.

```
# load the map data from json file
curl --location-trusted -uroot: -T events.json -H "format: json" -H "read_json_by_line: true"
http://fe_host:8030/api/test/simple_map/_stream_load
```

```
# 返回结果
{
    "TxnId": 106134,
    "Label": "5666e573-9a97-4dfc-ae61-2d6b61fdffd2",
    "Comment": "",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 10293125,
    "NumberLoadedRows": 10293125,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 2297411459,
    "LoadTimeMs": 66870,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 80,
    "ReadDataTimeMs": 6415,
    "WriteDataTimeMs": 10550,
    "CommitAndPublishTimeMs": 38
}
```

Select all data example:

```
mysql> SELECT * FROM simple_map;
+------+----------------------------+
| id   | m                          |
+------+----------------------------+
|    1 | {'a':100, 'b':200}         |
|    2 | {'b':100, 'c':200, 'd':300} |
|    3 | {'a':10, 'd':200}          |
+------+----------------------------+
```

Select map column example:

```
mysql> SELECT m FROM simple_map;
+----------------------------+
| m                          |
+----------------------------+
| {'a':100, 'b':200}         |
| {'b':100, 'c':200, 'd':300} |
| {'a':10, 'd':200}          |
+----------------------------+
```

Select map value according given key example:

```
mysql> SELECT m['a'] FROM simple_map;
+----------------------------+
| %element_extract%(`m`, 'a') |
+----------------------------+
|                        100 |
|                       NULL |
|                         10 |
+----------------------------+
```

map functions examples:

```
# map construct
```

```
mysql> SELECT map('k11', 1000, 'k22', 2000)['k11'];
+-----------------------------------------------------+
| %element_extract%(map('k11', 1000, 'k22', 2000), 'k11') |
+-----------------------------------------------------+
|                                                1000 |
+-----------------------------------------------------+

mysql> SELECT map('k11', 1000, 'k22', 2000)['nokey'];
+-------------------------------------------------------+
| %element_extract%(map('k11', 1000, 'k22', 2000), 'nokey') |
+-------------------------------------------------------+
|                                                  NULL |
+-------------------------------------------------------+
1 row in set (0.06 sec)

# map size

mysql> SELECT map_size(map('k11', 1000, 'k22', 2000));
+----------------------------------------+
| map_size(map('k11', 1000, 'k22', 2000)) |
+----------------------------------------+
|                                      2 |
+----------------------------------------+

mysql> SELECT id, m, map_size(m) FROM simple_map ORDER BY id;
+------+-----------------------------+--------------+
| id   | m                           | map_size(`m`) |
+------+-----------------------------+--------------+
|    1 | {"a":100, "b":200}          |             2 |
|    2 | {"b":100, "c":200, "d":300} |             3 |
|    2 | {"a":10, "d":200}           |             2 |
+------+-----------------------------+--------------+
3 rows in set (0.04 sec)

# map_contains_key

mysql> SELECT map_contains_key(map('k11', 1000, 'k22', 2000), 'k11');
+-------------------------------------------------------+
| map_contains_key(map('k11', 1000, 'k22', 2000), 'k11') |
+-------------------------------------------------------+
|                                                     1 |
+-------------------------------------------------------+
1 row in set (0.08 sec)

mysql> SELECT id, m, map_contains_key(m, 'k1') FROM simple_map ORDER BY id;
+------+-----------------------------+----------------------------+
| id   | m                           | map_contains_key(`m`, 'k1') |
+------+-----------------------------+----------------------------+
|    1 | {"a":100, "b":200}          |                          0 |
|    2 | {"b":100, "c":200, "d":300} |                          0 |
|    2 | {"a":10, "d":200}           |                          0 |
+------+-----------------------------+----------------------------+
3 rows in set (0.10 sec)

mysql> SELECT id, m, map_contains_key(m, 'a') FROM simple_map ORDER BY id;
+------+-----------------------------+---------------------------+
| id   | m                           | map_contains_key(`m`, 'a') |
+------+-----------------------------+---------------------------+
|    1 | {"a":100, "b":200}          |                         1 |
|    2 | {"b":100, "c":200, "d":300} |                         0 |
|    2 | {"a":10, "d":200}           |                         1 |
+------+-----------------------------+---------------------------+
3 rows in set (0.17 sec)

# map_contains_value
```

```
mysql> SELECT map_contains_value(map('k11', 1000, 'k22', 2000), NULL);
+-----------------------------------------------------------+
| map_contains_value(map('k11', 1000, 'k22', 2000), NULL) |
+-----------------------------------------------------------+
|                                                         0 |
+-----------------------------------------------------------+
1 row in set (0.04 sec)

mysql> SELECT id, m, map_contains_value(m, '100') FROM simple_map ORDER BY id;
+------+-----------------------------+-----------------------------+
| id   | m                           | map_contains_value(`m`, 100) |
+------+-----------------------------+-----------------------------+
|    1 | {"a":100, "b":200}          |                           1 |
|    2 | {"b":100, "c":200, "d":300} |                           1 |
|    2 | {"a":10, "d":200}           |                           0 |
+------+-----------------------------+-----------------------------+
3 rows in set (0.11 sec)

# map_keys

mysql> SELECT map_keys(map('k11', 1000, 'k22', 2000));
+----------------------------------------+
| map_keys(map('k11', 1000, 'k22', 2000)) |
+----------------------------------------+
| ["k11", "k22"]                         |
+----------------------------------------+
1 row in set (0.04 sec)

mysql> SELECT id, map_keys(m) FROM simple_map ORDER BY id;
+------+-----------------+
| id   | map_keys(`m`)   |
+------+-----------------+
|    1 | ["a", "b"]      |
|    2 | ["b", "c", "d"] |
|    2 | ["a", "d"]      |
+------+-----------------+
3 rows in set (0.19 sec)

# map_values

mysql> SELECT map_values(map('k11', 1000, 'k22', 2000));
+------------------------------------------+
| map_values(map('k11', 1000, 'k22', 2000)) |
+------------------------------------------+
| [1000, 2000]                             |
+------------------------------------------+
1 row in set (0.03 sec)

mysql> SELECT id, map_values(m) FROM simple_map ORDER BY id;
+------+-----------------+
| id   | map_values(`m`) |
+------+-----------------+
|    1 | [100, 200]      |
|    2 | [100, 200, 300] |
|    2 | [10, 200]       |
+------+-----------------+
3 rows in set (0.18 sec)
```

keywords

```
MAP
```

STRUCT

STRUCT

name

STRUCT

description

```
STRUCT<field_name:field_type [COMMENT 'comment_string'], ... >
```

Represents value with structure described by multiple fields, which can be viewed as a collection of multiple columns.

Need to manually enable the support, it is disabled by default.

```
admin set frontend config("enable_struct_type" = "true");
```

It cannot be used as a Key column. Now STRUCT can only used in Duplicate Model Tables.

The names and number of Fields in a Struct is fixed and always Nullable, and a Field typically consists of the following parts.

- field_name: Identifier naming the field, non repeatable.
- field_type: A data type.
- COMMENT: An optional string describing the field. (currently not supported)

The currently supported types are:

```
BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, LARGEINT, FLOAT, DOUBLE, DECIMAL, DECIMALV3, DATE,
DATEV2, DATETIME, DATETIMEV2, CHAR, VARCHAR, STRING
```

We have a todo list for future version:

```
TODO: Supports nested Struct or other complex types
```

example

Create table example:

```
mysql> CREATE TABLE `struct_test` (
  `id` int(11) NULL,
  `s_info` STRUCT<s_id:int(11), s_name:string, s_address:string> NULL
) ENGINE=OLAP
DUPLICATE KEY(`id`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`id`) BUCKETS 1
```

```
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
"storage_format" = "V2",
"light_schema_change" = "true",
"disable_auto_compaction" = "false"
);
```

Insert data example:

```
INSERT INTO `struct_test` VALUES (1, {1, 'sn1', 'sa1'});
INSERT INTO `struct_test` VALUES (2, struct(2, 'sn2', 'sa2'));
INSERT INTO `struct_test` VALUES (3, named_struct('s_id', 3, 's_name', 'sn3', 's_address', 'sa3'));
```

Stream load:

test.csv:

```
1|{"s_id":1, "s_name":"sn1", "s_address":"sa1"}
2|{s_id:2, s_name:sn2, s_address:sa2}
3|{"s_address":"sa3", "s_name":"sn3", "s_id":3}
```

example:

```
curl --location-trusted -u root -T test.csv  -H "label:test_label" http://host:port/api/test/struct_test/_stream_load
```

Select data example:

```
mysql> select * from struct_test;
+------+-------------------+
| id   | s_info            |
+------+-------------------+
|    1 | {1, 'sn1', 'sa1'} |
|    2 | {2, 'sn2', 'sa2'} |
|    3 | {3, 'sn3', 'sa3'} |
+------+-------------------+
3 rows in set (0.02 sec)
```

keywords

```
STRUCT
```

JSON

JSON

The JSON data type stores JSON data efficiently in a binary format and allows access to its internal fields through JSON functions.

By default, it supports up to 1048576 bytes (1MB), and can be increased up to 2147483643 bytes (2GB). This can be adjusted via the `string_type_length_soft_limit_bytes` configuration.

Compared to storing JSON strings in a regular STRING type, the JSON type has two main advantages: 1. JSON format validation during data insertion. 2. More efficient binary storage format, enabling faster access to JSON internal fields using functions like `json_extract`, compared to `get_json_xx` functions.

Note: In version 1.2.x, the JSON type was named JSONB. To maintain compatibility with MySQL, it was renamed to JSON starting from version 2.0.0. Older tables can still use the previous name.

Syntax

Definition:

```
json_column_name JSON
```

Insertion: - Using `INSERT INTO VALUES` with the format as a string surrounded by quotes. For example:

```
INSERT INTO table_name(id, json_column_name) VALUES (1, '{"k1": "100"}')
```

- For STREAM LOAD, the format for the corresponding column is a string without additional quotes. For example:

```
12  {"k1":"v31", "k2": 300}
13  []
14  [123, 456]
```

Query: - Directly select the entire JSON column:

```
SELECT json_column_name FROM table_name;
```

- Extract specific fields or other information from JSON using JSON functions. For example:

```
SELECT json_extract(json_column_name, '$.k1') FROM table_name;
```

- The JSON type can be cast to and from integers, strings, BOOLEAN, ARRAY, and MAP. For example:

```
SELECT CAST('{"k1": "100"}' AS JSON);
SELECT CAST(json_column_name AS STRING) FROM table_name;
```

```
SELECT CAST(json_extract(json_column_name, '$.k1') AS INT) FROM table_name;
```

> **The JSON type currently cannot be used for `GROUP BY`, `ORDER BY`, or comparison operations.**

example

A tutorial for JSON datatype including create table, load data and query.

create database and table

```
CREATE DATABASE testdb;

USE testdb;

CREATE TABLE test_json (
  id INT,
  j JSON
)
DUPLICATE KEY(id)
DISTRIBUTED BY HASH(id) BUCKETS 10
PROPERTIES("replication_num" = "1");
```

Load data

stream load test_json.csv test data

- there are 2 columns, the 1st column is id and the 2nd column is json string
- there are 25 rows, the first 18 rows are valid json and the last 7 rows are invalid

```
1   \N
2   null
3   true
4   false
5   100
6   10000
7   1000000000
8   1152921504606846976
9   6.18
10  "abcd"
11  {}
12  {"k1":"v31", "k2": 300}
13  []
14  [123, 456]
15  ["abc", "def"]
16  [null, true, false, 100, 6.18, "abc"]
17  [{"k1":"v41", "k2": 400}, 1, "a", 3.14]
18  {"k1":"v31", "k2": 300, "a1": [{"k1":"v41", "k2": 400}, 1, "a", 3.14]}
19  ''
20  'abc'
21  abc
22  100x
23  6.a8
```

```
24  {x
25  [123, abc]
```

- due to the 28% of rows is invalid，stream load with default configuration will fail with error message “too many filtered rows”

```
curl --location-trusted -u root: -T test_json.csv http://127.0.0.1:8840/api/testdb/test_json/_stream_load
{
    "TxnId": 12019,
    "Label": "744d9821-9c9f-43dc-bf3b-7ab048f14e32",
    "TwoPhaseCommit": "false",
    "Status": "Fail",
    "Message": "too many filtered rows",
    "NumberTotalRows": 25,
    "NumberLoadedRows": 18,
    "NumberFilteredRows": 7,
    "NumberUnselectedRows": 0,
    "LoadBytes": 380,
    "LoadTimeMs": 48,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 1,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 45,
    "CommitAndPublishTimeMs": 0,
    "ErrorURL": "http://172.21.0.5:8840/api/_load_error_log?file=__shard_2/error_log_insert_stmt_95435c4bf5f156df-
426735082a9296af_95435c4bf5f156df_426735082a9296af"
}
```

- stream load will success after set header configuration ‘max_filter_ratio: 0.3’

```
curl --location-trusted -u root: -H 'max_filter_ratio: 0.3' -T test_json.csv http://127.0.0.1:8840/api/testdb/test_json/_stream_load
{
    "TxnId": 12017,
    "Label": "f37a50c1-43e9-4f4e-a159-a3db6abe2579",
    "TwoPhaseCommit": "false",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 25,
    "NumberLoadedRows": 18,
    "NumberFilteredRows": 7,
    "NumberUnselectedRows": 0,
    "LoadBytes": 380,
    "LoadTimeMs": 68,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 2,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 45,
    "CommitAndPublishTimeMs": 19,
    "ErrorURL": "http://172.21.0.5:8840/api/_load_error_log?file=__shard_0/error_log_insert_stmt_a1463f98a7b15caf-
c79399b920f5bfa3_a1463f98a7b15caf_c79399b920f5bfa3"
}
```

- use SELECT to view the data loaded by stream load. The column with JSON type will be displayed as plain JSON string.

```
mysql> SELECT * FROM test_json ORDER BY id;
+------+-------------------------------------------------------------+
| id   | j                                                           |
+------+-------------------------------------------------------------+
```

```
|    1 |                                                        NULL |
|    2 |                                                        null |
|    3 |                                                        true |
|    4 |                                                       false |
|    5 |                                                         100 |
|    6 |                                                       10000 |
|    7 |                                                  1000000000 |
|    8 |                                         1152921504606846976 |
|    9 |                                                        6.18 |
|   10 |                                                      "abcd" |
|   11 |                                                         {} |
|   12 |                                       {"k1":"v31","k2":300} |
|   13 |                                                         [] |
|   14 |                                                  [123,456] |
|   15 |                                              ["abc","def"] |
|   16 |                              [null,true,false,100,6.18,"abc"] |
|   17 |                            [{"k1":"v41","k2":400},1,"a",3.14] |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |
+------+-------------------------------------------------------------+
18 rows in set (0.03 sec)
```

write data using insert into

- total rows increae from 18 to 19 after insert 1 row

```
mysql> INSERT INTO test_json VALUES(26, '{"k1":"v1", "k2": 200}');
Query OK, 1 row affected (0.09 sec)
{'label':'insert_4ece6769d1b42fd_ac9f25b3b8f3dc02', 'status':'VISIBLE', 'txnId':'12016'}

mysql> SELECT * FROM test_json ORDER BY id;
+------+-------------------------------------------------------------+
| id   | j                                                           |
+------+-------------------------------------------------------------+
|    1 |                                                        NULL |
|    2 |                                                        null |
|    3 |                                                        true |
|    4 |                                                       false |
|    5 |                                                         100 |
|    6 |                                                       10000 |
|    7 |                                                  1000000000 |
|    8 |                                         1152921504606846976 |
|    9 |                                                        6.18 |
|   10 |                                                      "abcd" |
|   11 |                                                         {} |
|   12 |                                       {"k1":"v31","k2":300} |
|   13 |                                                         [] |
|   14 |                                                  [123,456] |
|   15 |                                              ["abc","def"] |
|   16 |                              [null,true,false,100,6.18,"abc"] |
|   17 |                            [{"k1":"v41","k2":400},1,"a",3.14] |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |
|   26 |                                        {"k1":"v1","k2":200} |
+------+-------------------------------------------------------------+
19 rows in set (0.03 sec)
```

Query

extract some filed from json by json_extract functions

1. extract the whole json, ‘$’ stands for root in json path

```
+------+-------------------------------------------------------------+-------------------------------------------------------------
---+
| id   | j                                                           | json_extract(`j`, '$')
|
+------+-------------------------------------------------------------+-------------------------------------------------------------
---+
|    1 |                                                        NULL |
NULL |
|    2 |                                                        null |
null |
|    3 |                                                        true |
true |
|    4 |                                                       false |
false |
|    5 |                                                         100 |
100 |
|    6 |                                                       10000 |
10000 |
|    7 |                                                1000000000 |
1000000000 |
|    8 |                                         1152921504606846976 |
1152921504606846976 |
|    9 |                                                        6.18 |
6.18 |
|   10 |                                                      "abcd" |
"abcd" |
|   11 |                                                         {} |
{} |
|   12 |                                       {"k1":"v31","k2":300} |
{"k1":"v31","k2":300} |
|   13 |                                                         [] |
[] |
|   14 |                                                  [123,456] |
[123,456] |
|   15 |                                            ["abc","def"] |
["abc","def"] |
|   16 |                              [null,true,false,100,6.18,"abc"] |
[null,true,false,100,6.18,"abc"] |
|   17 |                         [{"k1":"v41","k2":400},1,"a",3.14] |
[{"k1":"v41","k2":400},1,"a",3.14] |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} | {"k1":"v31","k2":300,"a1":
[{"k1":"v41","k2":400},1,"a",3.14]} |
|   26 |                                        {"k1":"v1","k2":200} |
{"k1":"v1","k2":200} |
+------+-------------------------------------------------------------+-------------------------------------------------------------
---+
19 rows in set (0.03 sec)
```

1. extract k1 field, return NULL if it does not exist

```
mysql> SELECT id, j, json_extract(j, '$.k1') FROM test_json ORDER BY id;
+------+-------------------------------------------------------------+----------------------------+
| id   | j                                                           | json_extract(`j`, '$.k1') |
+------+-------------------------------------------------------------+----------------------------+
```

```
|    1 |                                                          NULL |                      NULL |
|    2 |                                                          null |                      NULL |
|    3 |                                                          true |                      NULL |
|    4 |                                                         false |                      NULL |
|    5 |                                                           100 |                      NULL |
|    6 |                                                         10000 |                      NULL |
|    7 |                                                    1000000000 |                      NULL |
|    8 |                                               1152921504606846976 |                      NULL |
|    9 |                                                          6.18 |                      NULL |
|   10 |                                                       "abcd" |                      NULL |
|   11 |                                                           {} |                      NULL |
|   12 |                                          {"k1":"v31","k2":300} |                     "v31" |
|   13 |                                                           [] |                      NULL |
|   14 |                                                    [123,456] |                      NULL |
|   15 |                                              ["abc","def"] |                      NULL |
|   16 |                                [null,true,false,100,6.18,"abc"] |                      NULL |
|   17 |                              [{"k1":"v41","k2":400},1,"a",3.14] |                      NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                     "v31" |
|   26 |                                           {"k1":"v1","k2":200} |                      "v1" |
+------+----------------------------------------------------------------+---------------------------+
19 rows in set (0.03 sec)
```

1. extract element 0 of the top level array

```
mysql> SELECT id, j, json_extract(j, '$[0]') FROM test_json ORDER BY id;
+------+----------------------------------------------------------------+---------------------------+
| id   | j                                                              | json_extract(`j`, '$[0]') |
+------+----------------------------------------------------------------+---------------------------+
|    1 |                                                          NULL |                      NULL |
|    2 |                                                          null |                      NULL |
|    3 |                                                          true |                      NULL |
|    4 |                                                         false |                      NULL |
|    5 |                                                           100 |                      NULL |
|    6 |                                                         10000 |                      NULL |
|    7 |                                                    1000000000 |                      NULL |
|    8 |                                               1152921504606846976 |                      NULL |
|    9 |                                                          6.18 |                      NULL |
|   10 |                                                       "abcd" |                      NULL |
|   11 |                                                           {} |                      NULL |
|   12 |                                          {"k1":"v31","k2":300} |                      NULL |
|   13 |                                                           [] |                      NULL |
|   14 |                                                    [123,456] |                       123 |
|   15 |                                              ["abc","def"] |                     "abc" |
|   16 |                                [null,true,false,100,6.18,"abc"] |                      null |
|   17 |                              [{"k1":"v41","k2":400},1,"a",3.14] |      {"k1":"v41","k2":400} |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                      NULL |
|   26 |                                           {"k1":"v1","k2":200} |                      NULL |
+------+----------------------------------------------------------------+---------------------------+
19 rows in set (0.03 sec)
```

1. extract a whole json array of name a1

```
mysql> SELECT id, j, json_extract(j, '$.a1') FROM test_json ORDER BY id;
+------+----------------------------------------------------------------+-----------------------------------+
| id   | j                                                              | json_extract(`j`, '$.a1')         |
+------+----------------------------------------------------------------+-----------------------------------+
|    1 |                                                          NULL |                              NULL |
|    2 |                                                          null |                              NULL |
|    3 |                                                          true |                              NULL |
|    4 |                                                         false |                              NULL |
|    5 |                                                           100 |                              NULL |
|    6 |                                                         10000 |                              NULL |
```

```
|    7 |                                                  1000000000 |                                NULL |
|    8 |                                             1152921504606846976 |                                NULL |
|    9 |                                                        6.18 |                                NULL |
|   10 |                                                      "abcd" |                                NULL |
|   11 |                                                         {} |                                NULL |
|   12 |                                          {"k1":"v31","k2":300} |                                NULL |
|   13 |                                                         [] |                                NULL |
|   14 |                                                  [123,456] |                                NULL |
|   15 |                                              ["abc","def"] |                                NULL |
|   16 |                                 [null,true,false,100,6.18,"abc"] |                                NULL |
|   17 |                               [{"k1":"v41","k2":400},1,"a",3.14] |                                NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} | [{"k1":"v41","k2":400},1,"a",3.14] |
|   26 |                                           {"k1":"v1","k2":200} |                                NULL |
+------+--------------------------------------------------------------+-----------------------------------+
19 rows in set (0.02 sec)
```

1. extract nested field from an object in an array

```
mysql> SELECT id, j, json_extract(j, '$.a1[0]'), json_extract(j, '$.a1[0].k1') FROM test_json ORDER BY id;
+------+--------------------------------------------------------------+-------------------------------+---------------------------
------+
| id   | j                                                            | json_extract(`j`, '$.a1[0]') | json_extract(`j`,
'$.a1[0].k1') |
+------+--------------------------------------------------------------+-------------------------------+---------------------------
------+
|    1 |                                                         NULL |                          NULL |
NULL |
|    2 |                                                         null |                          NULL |
NULL |
|    3 |                                                         true |                          NULL |
NULL |
|    4 |                                                        false |                          NULL |
NULL |
|    5 |                                                          100 |                          NULL |
NULL |
|    6 |                                                        10000 |                          NULL |
NULL |
|    7 |                                                  1000000000 |                          NULL |
NULL |
|    8 |                                             1152921504606846976 |                          NULL |
NULL |
|    9 |                                                        6.18 |                          NULL |
NULL |
|   10 |                                                      "abcd" |                          NULL |
NULL |
|   11 |                                                         {} |                          NULL |
NULL |
|   12 |                                          {"k1":"v31","k2":300} |                          NULL |
NULL |
|   13 |                                                         [] |                          NULL |
NULL |
|   14 |                                                  [123,456] |                          NULL |
NULL |
|   15 |                                              ["abc","def"] |                          NULL |
NULL |
|   16 |                                 [null,true,false,100,6.18,"abc"] |                          NULL |
NULL |
|   17 |                               [{"k1":"v41","k2":400},1,"a",3.14] |                          NULL |
NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |         {"k1":"v41","k2":400} |
"v41" |
|   26 |                                           {"k1":"v1","k2":200} |                          NULL |
NULL |
```

```
+------+-------------------------------------------------------------+-----------------------------+-------------------------
------+
19 rows in set (0.02 sec)
```

1. extract field with specific datatype

- json_extract_string will extract field with string type，convert to string if the field is not string

```
mysql> SELECT id, j, json_extract_string(j, '$') FROM test_json ORDER BY id;
+------+-------------------------------------------------------------+-----------------------------------------------------------
---+
| id   | j                                                           | json_extract_string(`j`, '$')
|
+------+-------------------------------------------------------------+-----------------------------------------------------------
---+
|    1 | NULL                                                        | NULL
|
|    2 | null                                                        | null
|
|    3 | true                                                        | true
|
|    4 | false                                                       | false
|
|    5 | 100                                                         | 100
|
|    6 | 10000                                                       | 10000
|
|    7 | 1000000000                                                  | 1000000000
|
|    8 | 1152921504606846976                                         | 1152921504606846976
|
|    9 | 6.18                                                        | 6.18
|
|   10 | "abcd"                                                      | abcd
|
|   11 | {}                                                          | {}
|
|   12 | {"k1":"v31","k2":300}                                       | {"k1":"v31","k2":300}
|
|   13 | []                                                          | []
|
|   14 | [123,456]                                                   | [123,456]
|
|   15 | ["abc","def"]                                               | ["abc","def"]
|
|   16 | [null,true,false,100,6.18,"abc"]                            | [null,true,false,100,6.18,"abc"]
|
|   17 | [{"k1":"v41","k2":400},1,"a",3.14]                          | [{"k1":"v41","k2":400},1,"a",3.14]
|
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} | {"k1":"v31","k2":300,"a1":
[{"k1":"v41","k2":400},1,"a",3.14]} |
|   26 | {"k1":"v1","k2":200}                                        | {"k1":"v1","k2":200}
|
+------+-------------------------------------------------------------+-----------------------------------------------------------
---+
19 rows in set (0.02 sec)

mysql> SELECT id, j, json_extract_string(j, '$.k1') FROM test_json ORDER BY id;
+------+-------------------------------------------------------------+----------------------------------+
| id   | j                                                           | json_extract_string(`j`, '$.k1') |
+------+-------------------------------------------------------------+----------------------------------+
|    1 |                                                        NULL | NULL                             |
|    2 |                                                        null | NULL                             |
```

```
|    3 |                                                    true | NULL                            |
|    4 |                                                   false | NULL                            |
|    5 |                                                     100 | NULL                            |
|    6 |                                                   10000 | NULL                            |
|    7 |                                              1000000000 | NULL                            |
|    8 |                                   1152921504606846976 | NULL                            |
|    9 |                                                    6.18 | NULL                            |
|   10 |                                                  "abcd" | NULL                            |
|   11 |                                                      {} | NULL                            |
|   12 |                                  {"k1":"v31","k2":300} | v31                             |
|   13 |                                                      [] | NULL                            |
|   14 |                                               [123,456] | NULL                            |
|   15 |                                         ["abc","def"] | NULL                            |
|   16 |                     [null,true,false,100,6.18,"abc"] | NULL                            |
|   17 |                   [{"k1":"v41","k2":400},1,"a",3.14] | NULL                            |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} | v31                             |
|   26 |                                   {"k1":"v1","k2":200} | v1                              |
+------+--------------------------------------------------------+---------------------------------+
19 rows in set (0.03 sec)
```

- json_extract_int will extract field with int type，return NULL if the field is not int

```
mysql> SELECT id, j, json_extract_int(j, '$') FROM test_json ORDER BY id;
+------+--------------------------------------------------------+-----------------------------+
| id   | j                                                      | json_extract_int(`j`, '$') |
+------+--------------------------------------------------------+-----------------------------+
|    1 |                                                   NULL |                        NULL |
|    2 |                                                   null |                        NULL |
|    3 |                                                   true |                        NULL |
|    4 |                                                  false |                        NULL |
|    5 |                                                    100 |                         100 |
|    6 |                                                  10000 |                       10000 |
|    7 |                                             1000000000 |                  1000000000 |
|    8 |                                  1152921504606846976 |                        NULL |
|    9 |                                                   6.18 |                        NULL |
|   10 |                                                 "abcd" |                        NULL |
|   11 |                                                     {} |                        NULL |
|   12 |                                 {"k1":"v31","k2":300} |                        NULL |
|   13 |                                                     [] |                        NULL |
|   14 |                                              [123,456] |                        NULL |
|   15 |                                        ["abc","def"] |                        NULL |
|   16 |                    [null,true,false,100,6.18,"abc"] |                        NULL |
|   17 |                  [{"k1":"v41","k2":400},1,"a",3.14] |                        NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                        NULL |
|   26 |                                  {"k1":"v1","k2":200} |                        NULL |
+------+--------------------------------------------------------+-----------------------------+
19 rows in set (0.02 sec)

mysql> SELECT id, j, json_extract_int(j, '$.k2') FROM test_json ORDER BY id;
+------+--------------------------------------------------------+-------------------------------+
| id   | j                                                      | json_extract_int(`j`, '$.k2') |
+------+--------------------------------------------------------+-------------------------------+
|    1 |                                                   NULL |                          NULL |
|    2 |                                                   null |                          NULL |
|    3 |                                                   true |                          NULL |
|    4 |                                                  false |                          NULL |
|    5 |                                                    100 |                          NULL |
|    6 |                                                  10000 |                          NULL |
|    7 |                                             1000000000 |                          NULL |
|    8 |                                  1152921504606846976 |                          NULL |
|    9 |                                                   6.18 |                          NULL |
|   10 |                                                 "abcd" |                          NULL |
|   11 |                                                     {} |                          NULL |
```

```
|   12 |                                      {"k1":"v31","k2":300} |                            300 |
|   13 |                                                        [] |                           NULL |
|   14 |                                                 [123,456] |                           NULL |
|   15 |                                           ["abc","def"] |                           NULL |
|   16 |                              [null,true,false,100,6.18,"abc"] |                           NULL |
|   17 |                            [{"k1":"v41","k2":400},1,"a",3.14] |                           NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                            300 |
|   26 |                                       {"k1":"v1","k2":200} |                            200 |
+------+--------------------------------------------------------------+--------------------------------+
19 rows in set (0.03 sec)
```

- json_extract_bigint will extract field with bigint type，return NULL if the field is not bigint

```
mysql> SELECT id, j, json_extract_bigint(j, '$') FROM test_json ORDER BY id;
+------+--------------------------------------------------------------+--------------------------------+
| id   | j                                                            | json_extract_bigint(`j`, '$') |
+------+--------------------------------------------------------------+--------------------------------+
|    1 |                                                         NULL |                           NULL |
|    2 |                                                         null |                           NULL |
|    3 |                                                         true |                           NULL |
|    4 |                                                        false |                           NULL |
|    5 |                                                          100 |                            100 |
|    6 |                                                        10000 |                          10000 |
|    7 |                                                   1000000000 |                     1000000000 |
|    8 |                                             1152921504606846976 |            1152921504606846976 |
|    9 |                                                         6.18 |                           NULL |
|   10 |                                                       "abcd" |                           NULL |
|   11 |                                                          {} |                           NULL |
|   12 |                                      {"k1":"v31","k2":300} |                           NULL |
|   13 |                                                          [] |                           NULL |
|   14 |                                                   [123,456] |                           NULL |
|   15 |                                             ["abc","def"] |                           NULL |
|   16 |                               [null,true,false,100,6.18,"abc"] |                           NULL |
|   17 |                             [{"k1":"v41","k2":400},1,"a",3.14] |                           NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                           NULL |
|   26 |                                       {"k1":"v1","k2":200} |                           NULL |
+------+--------------------------------------------------------------+--------------------------------+
19 rows in set (0.03 sec)

mysql> SELECT id, j, json_extract_bigint(j, '$.k2') FROM test_json ORDER BY id;
+------+--------------------------------------------------------------+----------------------------------+
| id   | j                                                            | json_extract_bigint(`j`, '$.k2') |
+------+--------------------------------------------------------------+----------------------------------+
|    1 |                                                         NULL |                             NULL |
|    2 |                                                         null |                             NULL |
|    3 |                                                         true |                             NULL |
|    4 |                                                        false |                             NULL |
|    5 |                                                          100 |                             NULL |
|    6 |                                                        10000 |                             NULL |
|    7 |                                                   1000000000 |                             NULL |
|    8 |                                             1152921504606846976 |                             NULL |
|    9 |                                                         6.18 |                             NULL |
|   10 |                                                       "abcd" |                             NULL |
|   11 |                                                          {} |                             NULL |
|   12 |                                      {"k1":"v31","k2":300} |                              300 |
|   13 |                                                          [] |                             NULL |
|   14 |                                                   [123,456] |                             NULL |
|   15 |                                             ["abc","def"] |                             NULL |
|   16 |                               [null,true,false,100,6.18,"abc"] |                             NULL |
|   17 |                             [{"k1":"v41","k2":400},1,"a",3.14] |                             NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                              300 |
|   26 |                                       {"k1":"v1","k2":200} |                              200 |
```

```
+------+-----------------------------------------------------------------+----------------------------------+
19 rows in set (0.02 sec)
```

- json_extract_double will extract field with double type，return NULL if the field is not double

```
mysql> SELECT id, j, json_extract_double(j, '$') FROM test_json ORDER BY id;
+------+-----------------------------------------------------------------+------------------------------+
| id   | j                                                               | json_extract_double(`j`, '$') |
+------+-----------------------------------------------------------------+------------------------------+
|    1 |                                                            NULL |                         NULL |
|    2 |                                                            null |                         NULL |
|    3 |                                                            true |                         NULL |
|    4 |                                                           false |                         NULL |
|    5 |                                                             100 |                          100 |
|    6 |                                                           10000 |                        10000 |
|    7 |                                                      1000000000 |                   1000000000 |
|    8 |                                                1152921504606846976 |     1.152921504606847e+18 |
|    9 |                                                            6.18 |                         6.18 |
|   10 |                                                          "abcd" |                         NULL |
|   11 |                                                             {} |                         NULL |
|   12 |                                          {"k1":"v31","k2":300} |                         NULL |
|   13 |                                                             [] |                         NULL |
|   14 |                                                      [123,456] |                         NULL |
|   15 |                                                ["abc","def"] |                         NULL |
|   16 |                                [null,true,false,100,6.18,"abc"] |                         NULL |
|   17 |                              [{"k1":"v41","k2":400},1,"a",3.14] |                         NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                         NULL |
|   26 |                                           {"k1":"v1","k2":200} |                         NULL |
+------+-----------------------------------------------------------------+------------------------------+
19 rows in set (0.02 sec)

mysql> SELECT id, j, json_extract_double(j, '$.k2') FROM test_json ORDER BY id;
+------+-----------------------------------------------------------------+---------------------------------+
| id   | j                                                               | json_extract_double(`j`, '$.k2') |
+------+-----------------------------------------------------------------+---------------------------------+
|    1 |                                                            NULL |                            NULL |
|    2 |                                                            null |                            NULL |
|    3 |                                                            true |                            NULL |
|    4 |                                                           false |                            NULL |
|    5 |                                                             100 |                            NULL |
|    6 |                                                           10000 |                            NULL |
|    7 |                                                      1000000000 |                            NULL |
|    8 |                                                1152921504606846976 |                            NULL |
|    9 |                                                            6.18 |                            NULL |
|   10 |                                                          "abcd" |                            NULL |
|   11 |                                                             {} |                            NULL |
|   12 |                                          {"k1":"v31","k2":300} |                             300 |
|   13 |                                                             [] |                            NULL |
|   14 |                                                      [123,456] |                            NULL |
|   15 |                                                ["abc","def"] |                            NULL |
|   16 |                                [null,true,false,100,6.18,"abc"] |                            NULL |
|   17 |                              [{"k1":"v41","k2":400},1,"a",3.14] |                            NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                             300 |
|   26 |                                           {"k1":"v1","k2":200} |                             200 |
+------+-----------------------------------------------------------------+---------------------------------+
19 rows in set (0.03 sec)
```

- json_extract_bool will extract field with boolean type，return NULL if the field is not boolean

```
mysql> SELECT id, j, json_extract_bool(j, '$') FROM test_json ORDER BY id;
+------+-----------------------------------------------------------------+-----------------------------+
| id   | j                                                               | json_extract_bool(`j`, '$') |
```

```
+------+-------------------------------------------------------------+-----------------------------+
|    1 |                                                        NULL |                        NULL |
|    2 |                                                        null |                        NULL |
|    3 |                                                        true |                           1 |
|    4 |                                                       false |                           0 |
|    5 |                                                         100 |                        NULL |
|    6 |                                                       10000 |                        NULL |
|    7 |                                                  1000000000 |                        NULL |
|    8 |                                             1152921504606846976 |                        NULL |
|    9 |                                                        6.18 |                        NULL |
|   10 |                                                      "abcd" |                        NULL |
|   11 |                                                         {} |                        NULL |
|   12 |                                          {"k1":"v31","k2":300} |                        NULL |
|   13 |                                                         [] |                        NULL |
|   14 |                                                  [123,456] |                        NULL |
|   15 |                                              ["abc","def"] |                        NULL |
|   16 |                                [null,true,false,100,6.18,"abc"] |                        NULL |
|   17 |                              [{"k1":"v41","k2":400},1,"a",3.14] |                        NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                        NULL |
|   26 |                                           {"k1":"v1","k2":200} |                        NULL |
+------+-------------------------------------------------------------+-----------------------------+
19 rows in set (0.01 sec)

mysql> SELECT id, j, json_extract_bool(j, '$[1]') FROM test_json ORDER BY id;
+------+-------------------------------------------------------------+--------------------------------+
| id   | j                                                           | json_extract_bool(`j`, '$[1]') |
+------+-------------------------------------------------------------+--------------------------------+
|    1 |                                                        NULL |                           NULL |
|    2 |                                                        null |                           NULL |
|    3 |                                                        true |                           NULL |
|    4 |                                                       false |                           NULL |
|    5 |                                                         100 |                           NULL |
|    6 |                                                       10000 |                           NULL |
|    7 |                                                  1000000000 |                           NULL |
|    8 |                                             1152921504606846976 |                           NULL |
|    9 |                                                        6.18 |                           NULL |
|   10 |                                                      "abcd" |                           NULL |
|   11 |                                                         {} |                           NULL |
|   12 |                                          {"k1":"v31","k2":300} |                           NULL |
|   13 |                                                         [] |                           NULL |
|   14 |                                                  [123,456] |                           NULL |
|   15 |                                              ["abc","def"] |                           NULL |
|   16 |                                [null,true,false,100,6.18,"abc"] |                              1 |
|   17 |                              [{"k1":"v41","k2":400},1,"a",3.14] |                           NULL |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                           NULL |
|   26 |                                           {"k1":"v1","k2":200} |                           NULL |
+------+-------------------------------------------------------------+--------------------------------+
19 rows in set (0.01 sec)
```

- json_extract_isnull will extract field with json null type，return 1 if the field is json null , else 0
- json null is different from SQL NULL. SQL NULL stands for no value for a field, but json null stands for an field with special value null.

```
mysql> SELECT id, j, json_extract_isnull(j, '$') FROM test_json ORDER BY id;
+------+-------------------------------------------------------------+--------------------------------+
| id   | j                                                           | json_extract_isnull(`j`, '$') |
+------+-------------------------------------------------------------+--------------------------------+
|    1 |                                                        NULL |                           NULL |
|    2 |                                                        null |                              1 |
|    3 |                                                        true |                              0 |
|    4 |                                                       false |                              0 |
|    5 |                                                         100 |                              0 |
|    6 |                                                       10000 |                              0 |
```

```
|    7 |                                                1000000000 |                              0 |
|    8 |                                         1152921504606846976 |                              0 |
|    9 |                                                      6.18 |                              0 |
|   10 |                                                    "abcd" |                              0 |
|   11 |                                                       {} |                              0 |
|   12 |                                          {"k1":"v31","k2":300} |                              0 |
|   13 |                                                       [] |                              0 |
|   14 |                                                 [123,456] |                              0 |
|   15 |                                               ["abc","def"] |                              0 |
|   16 |                                 [null,true,false,100,6.18,"abc"] |                              0 |
|   17 |                               [{"k1":"v41","k2":400},1,"a",3.14] |                              0 |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                              0 |
|   26 |                                           {"k1":"v1","k2":200} |                              0 |
+------+------------------------------------------------------------+-------------------------------+
19 rows in set (0.03 sec)
```

check if a field is existed in json by json_exists_path

```
mysql> SELECT id, j, json_exists_path(j, '$') FROM test_json ORDER BY id;
+------+------------------------------------------------------------+-----------------------------+
| id   | j                                                          | json_exists_path(`j`, '$') |
+------+------------------------------------------------------------+-----------------------------+
|    1 |                                                       NULL |                        NULL |
|    2 |                                                       null |                           1 |
|    3 |                                                       true |                           1 |
|    4 |                                                      false |                           1 |
|    5 |                                                        100 |                           1 |
|    6 |                                                      10000 |                           1 |
|    7 |                                                1000000000 |                           1 |
|    8 |                                         1152921504606846976 |                           1 |
|    9 |                                                      6.18 |                           1 |
|   10 |                                                    "abcd" |                           1 |
|   11 |                                                       {} |                           1 |
|   12 |                                          {"k1":"v31","k2":300} |                           1 |
|   13 |                                                       [] |                           1 |
|   14 |                                                 [123,456] |                           1 |
|   15 |                                               ["abc","def"] |                           1 |
|   16 |                                 [null,true,false,100,6.18,"abc"] |                           1 |
|   17 |                               [{"k1":"v41","k2":400},1,"a",3.14] |                           1 |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                           1 |
|   26 |                                           {"k1":"v1","k2":200} |                           1 |
+------+------------------------------------------------------------+-----------------------------+
19 rows in set (0.02 sec)

mysql> SELECT id, j, json_exists_path(j, '$.k1') FROM test_json ORDER BY id;
+------+------------------------------------------------------------+-------------------------------+
| id   | j                                                          | json_exists_path(`j`, '$.k1') |
+------+------------------------------------------------------------+-------------------------------+
|    1 |                                                       NULL |                          NULL |
|    2 |                                                       null |                             0 |
|    3 |                                                       true |                             0 |
|    4 |                                                      false |                             0 |
|    5 |                                                        100 |                             0 |
|    6 |                                                      10000 |                             0 |
|    7 |                                                1000000000 |                             0 |
|    8 |                                         1152921504606846976 |                             0 |
|    9 |                                                      6.18 |                             0 |
|   10 |                                                    "abcd" |                             0 |
|   11 |                                                       {} |                             0 |
|   12 |                                          {"k1":"v31","k2":300} |                             1 |
|   13 |                                                       [] |                             0 |
```

```
|   14 |                                                [123,456] |                              0 |
|   15 |                                            ["abc","def"] |                              0 |
|   16 |                              [null,true,false,100,6.18,"abc"] |                              0 |
|   17 |                            [{"k1":"v41","k2":400},1,"a",3.14] |                              0 |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                              1 |
|   26 |                                         {"k1":"v1","k2":200} |                              1 |
+------+------------------------------------------------------------+----------------------------------+
19 rows in set (0.03 sec)

mysql> SELECT id, j, json_exists_path(j, '$[2]') FROM test_json ORDER BY id;
+------+------------------------------------------------------------+----------------------------------+
| id   | j                                                          | json_exists_path(`j`, '$[2]') |
+------+------------------------------------------------------------+----------------------------------+
|    1 |                                                       NULL |                             NULL |
|    2 |                                                       null |                                0 |
|    3 |                                                       true |                                0 |
|    4 |                                                      false |                                0 |
|    5 |                                                        100 |                                0 |
|    6 |                                                      10000 |                                0 |
|    7 |                                                 1000000000 |                                0 |
|    8 |                                            1152921504606846976 |                                0 |
|    9 |                                                       6.18 |                                0 |
|   10 |                                                     "abcd" |                                0 |
|   11 |                                                        {} |                                0 |
|   12 |                                          {"k1":"v31","k2":300} |                                0 |
|   13 |                                                        [] |                                0 |
|   14 |                                                [123,456] |                                0 |
|   15 |                                            ["abc","def"] |                                0 |
|   16 |                              [null,true,false,100,6.18,"abc"] |                                1 |
|   17 |                            [{"k1":"v41","k2":400},1,"a",3.14] |                                1 |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} |                                0 |
|   26 |                                         {"k1":"v1","k2":200} |                                0 |
+------+------------------------------------------------------------+----------------------------------+
19 rows in set (0.02 sec)
```

get the datatype of a field in json by json_type

- return the data type of the field specified by json path, NULL if not existed.

```
mysql> SELECT id, j, json_type(j, '$') FROM test_json ORDER BY id;
+------+------------------------------------------------------------+----------------------+
| id   | j                                                          | json_type(`j`, '$') |
+------+------------------------------------------------------------+----------------------+
|    1 |                                                       NULL | NULL                 |
|    2 |                                                       null | null                 |
|    3 |                                                       true | bool                 |
|    4 |                                                      false | bool                 |
|    5 |                                                        100 | int                  |
|    6 |                                                      10000 | int                  |
|    7 |                                                 1000000000 | int                  |
|    8 |                                            1152921504606846976 | bigint               |
|    9 |                                                       6.18 | double               |
|   10 |                                                     "abcd" | string               |
|   11 |                                                        {} | object               |
|   12 |                                          {"k1":"v31","k2":300} | object               |
|   13 |                                                        [] | array                |
|   14 |                                                [123,456] | array                |
|   15 |                                            ["abc","def"] | array                |
|   16 |                              [null,true,false,100,6.18,"abc"] | array                |
|   17 |                            [{"k1":"v41","k2":400},1,"a",3.14] | array                |
```

```
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} | object              |
|   26 |                                          {"k1":"v1","k2":200} | object              |
+------+-----------------------------------------------------------------+---------------------+
19 rows in set (0.02 sec)

mysql> select id, j, json_type(j, '$.k1') from test_json order by id;
+------+-----------------------------------------------------------------+-------------------------+
| id   | j                                                               | json_type(`j`, '$.k1') |
+------+-----------------------------------------------------------------+-------------------------+
|    1 |                                                            NULL | NULL                    |
|    2 |                                                            null | NULL                    |
|    3 |                                                            true | NULL                    |
|    4 |                                                           false | NULL                    |
|    5 |                                                             100 | NULL                    |
|    6 |                                                           10000 | NULL                    |
|    7 |                                                      1000000000 | NULL                    |
|    8 |                                               1152921504606846976 | NULL                    |
|    9 |                                                            6.18 | NULL                    |
|   10 |                                                          "abcd" | NULL                    |
|   11 |                                                             {} | NULL                    |
|   12 |                                          {"k1":"v31","k2":300} | string                  |
|   13 |                                                             [] | NULL                    |
|   14 |                                                      [123,456] | NULL                    |
|   15 |                                                ["abc","def"] | NULL                    |
|   16 |                                [null,true,false,100,6.18,"abc"] | NULL                    |
|   17 |                              [{"k1":"v41","k2":400},1,"a",3.14] | NULL                    |
|   18 | {"k1":"v31","k2":300,"a1":[{"k1":"v41","k2":400},1,"a",3.14]} | string                  |
|   26 |                                          {"k1":"v1","k2":200} | string                  |
+------+-----------------------------------------------------------------+-------------------------+
19 rows in set (0.03 sec)
```

FAQ

1. Is there a difference between null in JSON and NULL in SQL (i.e., IS NULL)?

Yes, there is a difference. In JSON, null (e.g., {“key1”: null}) means that the key key1 exists and its value is explicitly null. This is a special type that gets encoded into JSON binary.

In contrast, SQL NULL (when using IS NULL) can indicate that the key doesn’t exist at all in the JSON object.

For example:

```
mysql> SELECT JSON_EXTRACT_STRING('{"key1" : null}', "$.key1") IS NULL;
+-----------------------------------------------------------+
| JSON_EXTRACT_STRING('{"key1" : null}', "$.key1") IS NULL |
+-----------------------------------------------------------+
|                                                         0 |
+-----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT JSON_EXTRACT_STRING('{"key1" : null}', "$.key_not_exist") IS NULL;
+-------------------------------------------------------------------+
| JSON_EXTRACT_STRING('{"key1" : null}', "$.key_not_exist") IS NULL |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+
1 row in set (0.01 sec)
```

2. What’s the difference between `GET_JSON_XXX` and `JSON_EXTRACT_XXX` functions, and how should I choose between them?

The `GET_JSON_XXX` functions are designed for use on string types — they extract values directly from raw JSON strings. On the other hand, `JSON_EXTRACT_XXX` functions are implemented specifically for the JSON data type and are optimized for it.

keywords

JSONB, JSON, json_parse, json_parse_error_to_null, json_parse_error_to_value, json_extract, json_extract_isnull, json_extract_bool, json_extract_int, json_extract_bigint, json_extract_double, json_extract_string, json_exists_path, json_type

HLL (HyperLogLog)

HLL (HyperLogLog)

Description

HLL

HLL cannot be used as a key column. The columns of the HLL type can be used in Aggregate tables, Duplicate tables and Unique tables. When used in an Aggregate table, the aggregation type is HLL_UNION when building table. The user does not need to specify the length and default value. The length is controlled within the system according to the degree of data aggregation. And HLL columns can only be queried or used through the matching hll_union_agg, hll_raw_agg, hll_cardinality, and hll_hash.

HLL is approximate count of distinct elements, and its performance is better than Count Distinct when the amount of data is large. The error of HLL is usually around 1%, sometimes up to 2%.

example

```
select hour, HLL_UNION_AGG(pv) over(order by hour) uv from(
   select hour, HLL_RAW_AGG(device_id) as pv
   from metric_table -- Query the accumulated UV per hour
   where datekey=20200922
group by hour order by 1
) final;
```

keywords

HLL,HYPERLOGLOG

BITMAP

BITMAP

Description

BITMAP

The columns of the BITMAP type can be used in Aggregate table, Unique table or Duplicate table. When used in a Unique table or Duplicate table, they must be used as non-key columns. When used in an Aggregate table, they must be used as non-key columns, and the aggregation type is BITMAP_UNION when building the table. The user does not need to specify the length and default value. The length is controlled within the system according to the degree of data aggregation. And the BITMAP column can only be queried or used by supporting functions such as bitmap_union_count, bitmap_union, bitmap_hash and bitmap_hash64.

The use of BITMAP in offline scenarios will affect the import speed. In the case of a large amount of data, the query speed will be slower than HLL and better than Count Distinct. Note: If BITMAP does not use a global dictionary in real-time scenarios, using bitmap_hash() may cause an error of about one-thousandth. If the error rate is not tolerable, bitmap_hash64 can be used instead.

example

Create table example:

```
create table metric_table (
  datekey int,
  hour int,
  device_id bitmap BITMAP_UNION
)
aggregate key (datekey, hour)
distributed by hash(datekey, hour) buckets 1
properties(
  "replication_num" = "1"
);
```

Insert data example:

```
insert into metric_table values
(20200622, 1, to_bitmap(243)),
(20200622, 2, bitmap_from_array([1,2,3,4,5,434543])),
(20200622, 3, to_bitmap(287667876573));
```

Query data example:

```
select hour, BITMAP_UNION_COUNT(pv) over(order by hour) uv from(
   select hour, BITMAP_UNION(device_id) as pv
   from metric_table -- Query the accumulated UV per hour
   where datekey=20200622
```

```
group by hour order by 1
) final;
```

You can set the session variable `return_object_data_as_binary` to `true`. Consequently, the bitmap will be returned in binary format.

keywords

BITMAP

QUANTILE_STATE

QUANTILE_STATE

description

QUANTILE_STATE

In 2.0, we support the agg_state function, and it is recommended to use agg_state quantile_union(quantile_state not null) instead of this type.

QUANTILE_STATE cannot be used as a key column. The columns of the HLL type can be used in Aggregate tables, Duplicate tables and Unique tables. When used in an Aggregate table, the aggregation type is HLL_UNION when building table.

The user does not need to specify the length and default value. The length is controlled within the system according to the degree of data aggregation. And the QUANTILE_STATE column can only be queried or used through the supporting QUANTILE_PERCENT, QUANTILE_UNION and TO_QUANTILE_STATE functions. QUANTILE_STATE is a type for calculating the approximate value of quantiles. Different values with the same key are pre-aggregated during loading process. When the number of aggregated values does not exceed 2048, all data are recorded in detail. When the number of aggregated values is greater than 2048, [TDigest] is used. (https://github.com/tdunning/t-digest/blob/main/docs/t-digest-paper/histo.pdf) algorithm to aggregate (cluster) the data and save the centroid points after clustering.

related functions:

```
QUANTILE_UNION(QUANTILE_STATE):

  This function is an aggregation function, which is used to aggregate the intermediate results of different quantile calculations.
The result returned by this function is still QUANTILE_STATE


TO_QUANTILE_STATE(DOUBLE raw_data [,FLOAT compression]):

   This function converts a numeric type to a QUANTILE_STATE type
   The compression parameter is optional and can be set in the range [2048, 10000].
   The larger the value, the higher the precision of quantile approximation calculations, the greater the memory consumption, and
the longer the calculation time.
   An unspecified or set value for the compression parameter is outside the range [2048, 10000], run with the default value of 2048

QUANTILE_PERCENT(QUANTILE_STATE, percent):
   This function converts the intermediate result variable (QUANTILE_STATE) of the quantile calculation into a specific quantile
value
```

notice

Now QUANTILE_STATE can only used in Aggregate Model Tables. We should turn on the switch for the QUANTILE_STATE types feature with the following command before use:

```
$ mysql-client > admin set frontend config("enable_quantile_state_type"="true");
```

In this way the config will be reset after the FE process restarts. For permanent setting, you can add config `enable_quantile_state_type=true` inside fe.conf.

example

```
select QUANTILE_PERCENT(QUANTILE_UNION(v1), 0.5) from test_table group by k1, k2, k3;
```

keywords

```
QUANTILE_STATE, QUANTILE_UNION, TO_QUANTILE_STATE, QUANTILE_PERCENT
```

AGG_STATE

AGG_STATE

description

```
  AGG_STATE cannot be used as a key column, and the signature of the aggregation function must be declared at the same time when creating the table.
  User does not need to specify length and default value. The actual stored data size is related to the function implementation.
```

AGG_STATE can only be used with state /merge/union function combiner usage.

It should be noted that the signature of the aggregation function is also part of the type, and agg_state with different signatures cannot be mixed. For example, if the signature of the table creation statement is `max_by(int,int)`, then `max_by(bigint,int)` or `group_concat(varchar)` cannot be inserted. The nullable attribute here is also part of the signature. If you can confirm that you will not enter a null value, you can declare the parameter as not null, which can obtain a smaller storage size and reduce serialization/deserialization overhead.

Note: Because agg_state stores the intermediate result of the aggregation function, the read and write process is strongly dependent on the specific implementation of the aggregation function, if the implementation of the aggregation function is modified during the Doris version upgrade, it may cause incompatible situations. If the incompatibility occurs, the materialized view using the corresponding agg_state needs to be dropped and recreated, and the underlying aggregation table involved will be directly unavailable, so you need to use agg_state with caution.

example

Create table example:

```
-- after doris-2.1.1
create table a_table(
    k1 int null,
    k2 agg_state<max_by(int not null,int)> generic,
    k3 agg_state<group_concat(string) generic
)
aggregate key (k1)
distributed BY hash(k1) buckets 3
properties("replication_num" = "1");

-- until doris-2.1.0
create table a_table(
    k1 int null,
    k2 agg_state max_by(int not null,int),
    k3 agg_state group_concat(string)
)
aggregate key (k1)
distributed BY hash(k1) buckets 3
properties("replication_num" = "1");
```

Here k2 and k3 use max_by and group_concat as aggregation types respectively.

Insert data example:

```
insert into a_table values(1,max_by_state(3,1),group_concat_state('a'));
insert into a_table values(1,max_by_state(2,2),group_concat_state('bb'));
insert into a_table values(2,max_by_state(1,3),group_concat_state('ccc'));
```

For the agg_state column, the insert statement must use the state function to generate the corresponding agg_state data, where the functions and input parameter types must completely correspond to agg_state.

Select data example:

```
mysql [test]>select k1,max_by_merge(k2),group_concat_merge(k3) from a_table group by k1 order by k1;
+------+--------------------+-------------------------+
| k1   | max_by_merge(`k2`) | group_concat_merge(`k3`) |
+------+--------------------+-------------------------+
|    1 |                  2 | bb,a                    |
|    2 |                  1 | ccc                     |
+------+--------------------+-------------------------+
```

If you need to get the actual result, you need to use the corresponding merge function.

```
mysql [test]>select max_by_merge(u2),group_concat_merge(u3) from (
select k1,max_by_union(k2) as u2,group_concat_union(k3) u3 from a_table group by k1 order by k1
) t;
+--------------------+-------------------------+
| max_by_merge(`u2`) | group_concat_merge(`u3`) |
+--------------------+-------------------------+
|                  1 | ccc,bb,a                |
+--------------------+-------------------------+
```

If you want to aggregate only the agg_state without getting the actual result during the process, you can use the union function.

For more examples, see datatype_p0/agg_state #### keywords

```
AGG_STATE
```

ARRAY

array()

array() #### description

Syntax

```
ARRAY<T> array(T, ...)
```

construct an array with variadic elements and return it, T could be column or literal

example

```
mysql> select array("1", 2, 1.1);
+----------------------+
| array('1', 2, '1.1') |
+----------------------+
| ['1', '2', '1.1']    |
+----------------------+
1 row in set (0.00 sec)


mysql> select array(null, 1);
+----------------+
| array(NULL, 1) |
+----------------+
| [NULL, 1]      |
+----------------+
1 row in set (0.00 sec)

mysql> select array(1, 2, 3);
+----------------+
| array(1, 2, 3) |
+----------------+
| [1, 2, 3]      |
+----------------+
1 row in set (0.00 sec)

mysql>  select array(qid, creationDate, null) from nested  limit 4;
+-----------------------------------+
| array(`qid`, `creationDate`, NULL) |
+-----------------------------------+
| [1000038, 20090616074056, NULL]    |
| [1000069, 20090616075005, NULL]    |
| [1000130, 20090616080918, NULL]    |
| [1000145, 20090616081545, NULL]    |
+-----------------------------------+
4 rows in set (0.01 sec)
```

keywords

ARRAY,ARRAY,CONSTRUCTOR

ARRAY_MAX

array_max

array_max

description

Syntax

```
T array_max(ARRAY<T> array1)
```

Get the maximum element in an array (`NULL` values are skipped). When the array is empty or all elements in the array are `NULL` values, the function returns `NULL`.

example

```
mysql> create table array_type_table(k1 INT, k2 Array<int>) duplicate key (k1)
    -> distributed by hash(k1) buckets 1 properties('replication_num' = '1');
mysql> insert into array_type_table values (0, []), (1, [NULL]), (2, [1, 2, 3]), (3, [1, NULL, 3]);
mysql> select k2, array_max(k2) from array_type_table;
+--------------+-----------------+
| k2           | array_max(`k2`) |
+--------------+-----------------+
| []           |            NULL |
| [NULL]       |            NULL |
| [1, 2, 3]    |               3 |
| [1, NULL, 3] |               3 |
+--------------+-----------------+
4 rows in set (0.02 sec)
```

keywords

ARRAY,MAX,ARRAY_MAX

ARRAY_MIN

array_min

array_min

description

Syntax

```
T array_min(ARRAY<T> array1)
```

Get the minimum element in an array (`NULL` values are skipped). When the array is empty or all elements in the array are `NULL` values, the function returns `NULL`.

example

```
mysql> create table array_type_table(k1 INT, k2 Array<int>) duplicate key (k1)
    -> distributed by hash(k1) buckets 1 properties('replication_num' = '1');
mysql> insert into array_type_table values (0, []), (1, [NULL]), (2, [1, 2, 3]), (3, [1, NULL, 3]);
mysql> select k2, array_min(k2) from array_type_table;
+--------------+-----------------+
| k2           | array_min(`k2`) |
+--------------+-----------------+
| []           |            NULL |
| [NULL]       |            NULL |
| [1, 2, 3]    |               1 |
| [1, NULL, 3] |               1 |
+--------------+-----------------+
4 rows in set (0.02 sec)
```

keywords

ARRAY,MIN,ARRAY_MIN

ARRAY_MAP

array_map

array_map(lambda,array,….)

description

Syntax

```
ARRAY<T> array_map(lambda, ARRAY<T> array1, ARRAY<T> array2)
```

Use a lambda expression as the input parameter to calculate the corresponding expression for the internal data of other input ARRAY parameters. The number of parameters entered in the lambda expression is 1 or more, which must be consistent with the number of input array columns. The scalar functions can be executed in lambda, and aggregate functions are not supported.

```
array_map(x->x, array1);
array_map(x->(x+2), array1);
array_map(x->(abs(x)-2), array1);

array_map((x,y)->(x = y), array1, array2);
array_map((x,y)->(power(x,2)+y), array1, array2);
array_map((x,y,z)->(abs(x)+y*z), array1, array2, array3);
```

example

```
mysql [test]>select *, array_map(x->x,[1,2,3]) from array_test2 order by id;
+------+-----------------+-------------------------+----------------------------------------+
| id   | c_array1        | c_array2                | array_map([x] -> x(0), ARRAY(1, 2, 3)) |
+------+-----------------+-------------------------+----------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [1, 2, 3]                              |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [1, 2, 3]                              |
|    3 | [1]             | [-100]                  | [1, 2, 3]                              |
|    4 | NULL            | NULL                    | [1, 2, 3]                              |
+------+-----------------+-------------------------+----------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select *, array_map(x->x+2,[1,2,3]) from array_test2 order by id;
+------+-----------------+-------------------------+----------------------------------------+
| id   | c_array1        | c_array2                | array_map([x] -> x(0) + 2, ARRAY(1, 2, 3)) |
+------+-----------------+-------------------------+----------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [3, 4, 5]                              |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [3, 4, 5]                              |
|    3 | [1]             | [-100]                  | [3, 4, 5]                              |
|    4 | NULL            | NULL                    | [3, 4, 5]                              |
+------+-----------------+-------------------------+----------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select c_array1, c_array2, array_map(x->x,[1,2,3]) from array_test2 order by id;
+-----------------+-------------------------+----------------------------------------+
| c_array1        | c_array2                | array_map([x] -> x(0), ARRAY(1, 2, 3)) |
```

```
+-----------------+-------------------------+--------------------------------------+
| [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [1, 2, 3]                            |
| [6, 7, 8]       | [10, 12, 13]            | [1, 2, 3]                            |
| [1]             | [-100]                  | [1, 2, 3]                            |
| NULL            | NULL                    | [1, 2, 3]                            |
+-----------------+-------------------------+--------------------------------------+
4 rows in set (0.01 sec)

mysql [test]>select c_array1, c_array2, array_map(x->power(x,2),[1,2,3]) from array_test2 order by id;
+-----------------+-------------------------+--------------------------------------------------+
| c_array1        | c_array2                | array_map([x] -> power(x(0), 2.0), ARRAY(1, 2, 3)) |
+-----------------+-------------------------+--------------------------------------------------+
| [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [1, 4, 9]                                        |
| [6, 7, 8]       | [10, 12, 13]            | [1, 4, 9]                                        |
| [1]             | [-100]                  | [1, 4, 9]                                        |
| NULL            | NULL                    | [1, 4, 9]                                        |
+-----------------+-------------------------+--------------------------------------------------+

mysql [test]>select c_array1, c_array2, array_map((x,y)->x+y,c_array1,c_array2) from array_test2 order by id;
+-----------------+-------------------------+-------------------------------------------------------------+
| c_array1        | c_array2                | array_map([x, y] -> x(0) + y(1), `c_array1`, `c_array2`) |
+-----------------+-------------------------+-------------------------------------------------------------+
| [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [11, 22, -37, 84, -95]                                      |
| [6, 7, 8]       | [10, 12, 13]            | [16, 19, 21]                                                |
| [1]             | [-100]                  | [-99]                                                       |
| NULL            | NULL                    | NULL                                                        |
+-----------------+-------------------------+-------------------------------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select c_array1, c_array2, array_map((x,y)->power(x,2)+y,c_array1, c_array2) from array_test2 order by id;
+-----------------+-------------------------+-----------------------------------------------------------------------+
| c_array1        | c_array2                | array_map([x, y] -> power(x(0), 2.0) + y(1), `c_array1`, `c_array2`) |
+-----------------+-------------------------+-----------------------------------------------------------------------+
| [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [11, 24, -31, 96, -75]                                                |
| [6, 7, 8]       | [10, 12, 13]            | [46, 61, 77]                                                          |
| [1]             | [-100]                  | [-99]                                                                 |
| NULL            | NULL                    | NULL                                                                  |
+-----------------+-------------------------+-----------------------------------------------------------------------+
4 rows in set (0.03 sec)

mysql [test]>select *,array_map(x->x=3,c_array1) from array_test2 order by id;
+------+-----------------+-------------------------+---------------------------------------+
| id   | c_array1        | c_array2                | array_map([x] -> x(0) = 3, `c_array1`) |
+------+-----------------+-------------------------+---------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [0, 0, 1, 0, 0]                       |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [0, 0, 0]                             |
|    3 | [1]             | [-100]                  | [0]                                   |
|    4 | NULL            | NULL                    | NULL                                  |
+------+-----------------+-------------------------+---------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select *,array_map(x->x>3,c_array1) from array_test2 order by id;
+------+-----------------+-------------------------+---------------------------------------+
| id   | c_array1        | c_array2                | array_map([x] -> x(0) > 3, `c_array1`) |
+------+-----------------+-------------------------+---------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [0, 0, 0, 1, 1]                       |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [1, 1, 1]                             |
|    3 | [1]             | [-100]                  | [0]                                   |
|    4 | NULL            | NULL                    | NULL                                  |
+------+-----------------+-------------------------+---------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select *,array_map((x,y)->x>y,c_array1,c_array2) from array_test2 order by id;
+------+-----------------+-------------------------+-------------------------------------------------------------+
| id   | c_array1        | c_array2                | array_map([x, y] -> x(0) > y(1), `c_array1`, `c_array2`) |
```

```
+------+----------------+------------------------+-------------------------------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [0, 0, 1, 0, 1]                                             |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [0, 0, 0]                                                   |
|    3 | [1]             | [-100]                  | [1]                                                         |
|    4 | NULL            | NULL                    | NULL                                                        |
+------+----------------+------------------------+-------------------------------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select array_map(x->cast(x as string), c_array1) from test_array_map_function;
+----------------+-----------------------------------------------------+
| c_array1       | array_map([x] -> CAST(x(0) AS CHARACTER), `c_array1`) |
+----------------+-----------------------------------------------------+
| [1, 2, 3, 4, 5] | ['1', '2', '3', '4', '5']                           |
| [6, 7, 8]       | ['6', '7', '8']                                     |
| []              | []                                                  |
| NULL            | NULL                                                |
+----------------+-----------------------------------------------------+
4 rows in set (0.01 sec)
```

keywords

ARRAY,MAP,ARRAY_MAP

ARRAY_FILTER

array_filter

array_filter(lambda,array)

> **Tips**
>
> This functions is supported since the Apache Doris 2.0.2 version

array array_filter(array arr, array_bool filter_column)

description

Syntax

```
ARRAY<T> array_filter(lambda, ARRAY<T> arr)
ARRAY<T> array_filter(ARRAY<T> arr, ARRAY<Bool> filter_column)
```

Use the lambda expression as the input parameter to calculate and filter the data of the ARRAY column of the other input parameter. And filter out the values of 0 and NULL in the result.

```
array_filter(x->x>0, array1);
array_filter(x->(x+2)=10, array1);
array_filter(x->(abs(x)-2)>0, array1);
array_filter(c_array,[0,1,0]);
```

example

```
mysql [test]>select c_array,array_filter(c_array,[0,1,0]) from array_test;
+-----------------+-----------------------------------------------------+
| c_array         | array_filter(`c_array`, ARRAY(FALSE, TRUE, FALSE)) |
+-----------------+-----------------------------------------------------+
| [1, 2, 3, 4, 5] | [2]                                                 |
| [6, 7, 8]       | [7]                                                 |
| []              | []                                                  |
| NULL            | NULL                                                |
+-----------------+-----------------------------------------------------+

mysql [test]>select array_filter(x->(x > 1),[1,2,3,0,null]);
+-----------------------------------------------------------------------------------------+
| array_filter(ARRAY(1, 2, 3, 0, NULL), array_map([x] -> (x(0) > 1), ARRAY(1, 2, 3, 0, NULL))) |
+-----------------------------------------------------------------------------------------+
| [2, 3]                                                                                  |
+-----------------------------------------------------------------------------------------+

mysql [test]>select *, array_filter(x->x>0,c_array2) from array_test2;
+------+----------------+-----------------------+-----------------------------------------------------------------------+
| id   | c_array1       | c_array2              | array_filter(`c_array2`, array_map([x] -> x(0) > 0, `c_array2`)) |
```

```
+------+-----------------+-------------------------+-------------------------------------------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [10, 20, 80]                                                            |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [10, 12, 13]                                                            |
|    3 | [1]             | [-100]                  | []                                                                      |
|    4 | NULL            | NULL                    | NULL                                                                    |
+------+-----------------+-------------------------+-------------------------------------------------------------------------+
4 rows in set (0.01 sec)

mysql [test]>select *, array_filter(x->x%2=0,c_array2) from array_test2;
+------+-----------------+-------------------------+-------------------------------------------------------------------------+
| id   | c_array1        | c_array2                | array_filter(`c_array2`, array_map([x] -> x(0) % 2 = 0, `c_array2`)) |
+------+-----------------+-------------------------+-------------------------------------------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [10, 20, -40, 80, -100]                                                 |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [10, 12]                                                                |
|    3 | [1]             | [-100]                  | [-100]                                                                  |
|    4 | NULL            | NULL                    | NULL                                                                    |
+------+-----------------+-------------------------+-------------------------------------------------------------------------+

mysql [test]>select *, array_filter(x->(x*(-10)>0),c_array2) from array_test2;
+------+-----------------+-------------------------+-----------------------------------------------------------------------------------+
| id   | c_array1        | c_array2                | array_filter(`c_array2`, array_map([x] -> (x(0) * (-10) > 0), `c_array2`)) |
+------+-----------------+-------------------------+-----------------------------------------------------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [-40, -100]                                                                       |
|    2 | [6, 7, 8]       | [10, 12, 13]            | []                                                                                |
|    3 | [1]             | [-100]                  | [-100]                                                                            |
|    4 | NULL            | NULL                    | NULL                                                                              |
+------+-----------------+-------------------------+-----------------------------------------------------------------------------------+

mysql [test]>select *, array_filter(x->x>0, array_map((x,y)->(x>y), c_array1,c_array2)) as res from array_test2;
+------+-----------------+-------------------------+--------+
| id   | c_array1        | c_array2                | res    |
+------+-----------------+-------------------------+--------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [1, 1] |
|    2 | [6, 7, 8]       | [10, 12, 13]            | []     |
|    3 | [1]             | [-100]                  | [1]    |
|    4 | NULL            | NULL                    | NULL   |
+------+-----------------+-------------------------+--------+
```

keywords

ARRAY,FILTER,ARRAY_FILTER

ARRAY_AVG

array_avg

array_avg

description

Syntax

```
Array<T> array_avg(arr)
```

Get the average of all elements in an array (`NULL` values are skipped). When the array is empty or all elements in the array are `NULL` values, the function returns `NULL`.

example

```
mysql> create table array_type_table(k1 INT, k2 Array<int>) duplicate key (k1)
    -> distributed by hash(k1) buckets 1 properties('replication_num' = '1');
mysql> insert into array_type_table values (0, []), (1, [NULL]), (2, [1, 2, 3]), (3, [1, NULL, 3]);
mysql> select k2, array_avg(k2) from array_type_table;
+--------------+-----------------+
| k2           | array_avg(`k2`) |
+--------------+-----------------+
| []           |            NULL |
| [NULL]       |            NULL |
| [1, 2, 3]    |               2 |
| [1, NULL, 3] |               2 |
+--------------+-----------------+
4 rows in set (0.01 sec)
```

keywords

ARRAY,AVG,ARRAY_AVG

ARRAY_SUM

array_sum

array_sum

description

Syntax

```
T array_sum(ARRAY<T> src, Array<T> key)
T array_sum(lambda, Array<T> arr1, Array<T> arr2 ....)
```

Get the sum of all elements in an array (`NULL` values are skipped). When the array is empty or all elements in the array are `NULL` values, the function returns `NULL`.

example

```
mysql> create table array_type_table(k1 INT, k2 Array<int>) duplicate key (k1)
    -> distributed by hash(k1) buckets 1 properties('replication_num' = '1');
mysql> insert into array_type_table values (0, []), (1, [NULL]), (2, [1, 2, 3]), (3, [1, NULL, 3]);
mysql> select k2, array_sum(k2) from array_type_table;
+--------------+-----------------+
| k2           | array_sum(`k2`) |
+--------------+-----------------+
| []           |            NULL |
| [NULL]       |            NULL |
| [1, 2, 3]    |               6 |
| [1, NULL, 3] |               4 |
+--------------+-----------------+
4 rows in set (0.01 sec)
```

keywords

ARRAY,SUM,ARRAY_SUM

ARRAY_SIZE

array_size (size, cardinality)

array_size (size, cardinality) #### description

Syntax

```
BIGINT size(ARRAY<T> arr)
BIGINT array_size(ARRAY<T> arr)
BIGINT cardinality(ARRAY<T> arr)
```

Returns the size of the array, returns NULL for NULL input.

example

```
mysql> select k1,k2,size(k2) from array_test;
+------+-----------+------------+
| k1   | k2        | size(`k2`) |
+------+-----------+------------+
|    1 | [1, 2, 3] |          3 |
|    2 | []        |          0 |
|    3 | NULL      |       NULL |
+------+-----------+------------+

mysql> select k1,k2,array_size(k2) from array_test;
+------+-----------+------------------+
| k1   | k2        | array_size(`k2`) |
+------+-----------+------------------+
|    1 | [1, 2, 3] |                3 |
|    2 | []        |                0 |
|    3 | NULL      |             NULL |
+------+-----------+------------------+

mysql> select k1,k2,cardinality(k2) from array_test;
+------+-----------+-------------------+
| k1   | k2        | cardinality(`k2`) |
+------+-----------+-------------------+
|    1 | [1, 2, 3] |                 3 |
|    2 | []        |                 0 |
|    3 | NULL      |              NULL |
+------+-----------+-------------------+
```

keywords

ARRAY_SIZE, SIZE, CARDINALITY

ARRAY_REMOVE

array_remove

array_remove

description

Syntax

```
ARRAY<T> array_remove(ARRAY<T> arr, T val)
```

Remove all elements that equal to element from array.

example

```
mysql> select array_remove(['test', NULL, 'value'], 'value');
+-----------------------------------------------------+
| array_remove(ARRAY('test', NULL, 'value'), 'value') |
+-----------------------------------------------------+
| [test, NULL]                                        |
+-----------------------------------------------------+

mysql> select k1, k2, array_remove(k2, 1) from array_type_table_1;
+------+--------------------+-----------------------+
| k1   | k2                 | array_remove(`k2`, 1) |
+------+--------------------+-----------------------+
|    1 | [1, 2, 3]          | [2, 3]                |
|    2 | [1, 3]             | [3]                   |
|    3 | NULL               | NULL                  |
|    4 | [1, 3]             | [3]                   |
|    5 | [NULL, 1, NULL, 2] | [NULL, NULL, 2]       |
+------+--------------------+-----------------------+

mysql> select k1, k2, array_remove(k2, k1) from array_type_table_1;
+------+--------------------+-------------------------+
| k1   | k2                 | array_remove(`k2`, `k1`) |
+------+--------------------+-------------------------+
|    1 | [1, 2, 3]          | [2, 3]                  |
|    2 | [1, 3]             | [1, 3]                  |
|    3 | NULL               | NULL                    |
|    4 | [1, 3]             | [1, 3]                  |
|    5 | [NULL, 1, NULL, 2] | [NULL, 1, NULL, 2]      |
+------+--------------------+-------------------------+

mysql> select k1, k2, array_remove(k2, date('2022-10-10')) from array_type_table_date;
+------+-------------------------+-------------------------------------------------+
| k1   | k2                      | array_remove(`k2`, date('2022-10-10 00:00:00')) |
+------+-------------------------+-------------------------------------------------+
|    1 | [2021-10-10, 2022-10-10] | [2021-10-10]                                    |
|    2 | [NULL, 2022-05-14]       | [NULL, 2022-05-14]                              |
+------+-------------------------+-------------------------------------------------+

mysql> select k1, k2, array_remove(k2, k1) from array_type_table_nullable;
+------+-----------+-------------------------+
```

```
| k1   | k2        | array_remove(`k2`, `k1`) |
+------+-----------+--------------------------+
| NULL | [1, 2, 3] | NULL                     |
|    1 | NULL      | NULL                     |
| NULL | [NULL, 1] | NULL                     |
|    1 | [NULL, 1] | [NULL]                   |
+------+-----------+--------------------------+
```

keywords

ARRAY,REMOVE,ARRAY_REMOVE

ARRAY_SLICE

array_slice

array_slice

description

Syntax

```
ARRAY<T> array_slice(ARRAY<T> arr, BIGINT off, BIGINT len)
```

Returns a slice of the array.

```
A positive off indicates an indent on the left
A negative off indicates an indent on the right.
An empty array is returned when the off is not within the actual range of the array.
A negative len will be treated as 0.
```

example

```
mysql> select k2, k2[2:2] from array_type_table_nullable;
+-----------------+-------------------------+
| k2              | array_slice(`k2`, 2, 2) |
+-----------------+-------------------------+
| [1, 2, 3]       | [2, 3]                  |
| [1, NULL, 3]    | [NULL, 3]               |
| [2, 3]          | [3]                     |
| NULL            | NULL                    |
+-----------------+-------------------------+

mysql> select k2, array_slice(k2, 2, 2) from array_type_table_nullable;
+-----------------+-------------------------+
| k2              | array_slice(`k2`, 2, 2) |
+-----------------+-------------------------+
| [1, 2, 3]       | [2, 3]                  |
| [1, NULL, 3]    | [NULL, 3]               |
| [2, 3]          | [3]                     |
| NULL            | NULL                    |
+-----------------+-------------------------+

mysql> select k2, k2[2:2] from array_type_table_nullable_varchar;
+---------------------------+-------------------------+
| k2                        | array_slice(`k2`, 2, 2) |
+---------------------------+-------------------------+
| ['hello', 'world', 'c++'] | ['world', 'c++']        |
| ['a1', 'equals', 'b1']    | ['equals', 'b1']        |
| ['hasnull', NULL, 'value'] | [NULL, 'value']         |
| ['hasnull', NULL, 'value'] | [NULL, 'value']         |
+---------------------------+-------------------------+

mysql> select k2, array_slice(k2, 2, 2) from array_type_table_nullable_varchar;
+---------------------------+-------------------------+
| k2                        | array_slice(`k2`, 2, 2) |
```

```
+---------------------------+------------------------+
| ['hello', 'world', 'c++']  | ['world', 'c++']       |
| ['a1', 'equals', 'b1']     | ['equals', 'b1']       |
| ['hasnull', NULL, 'value'] | [NULL, 'value']        |
| ['hasnull', NULL, 'value'] | [NULL, 'value']        |
+---------------------------+------------------------+
```

Negative off:

```
mysql> select k2, k2[-2:1] from array_type_table_nullable;
+-----------+--------------------------+
| k2        | array_slice(`k2`, -2, 1) |
+-----------+--------------------------+
| [1, 2, 3] | [2]                      |
| [1, 2, 3] | [2]                      |
| [2, 3]    | [2]                      |
| [2, 3]    | [2]                      |
+-----------+--------------------------+

mysql> select k2, array_slice(k2, -2, 1) from array_type_table_nullable;
+-----------+--------------------------+
| k2        | array_slice(`k2`, -2, 1) |
+-----------+--------------------------+
| [1, 2, 3] | [2]                      |
| [1, 2, 3] | [2]                      |
| [2, 3]    | [2]                      |
| [2, 3]    | [2]                      |
+-----------+--------------------------+

mysql> select k2, k2[-2:2] from array_type_table_nullable_varchar;
+---------------------------+-------------------------+
| k2                        | array_slice(`k2`, -2, 2) |
+---------------------------+-------------------------+
| ['hello', 'world', 'c++']  | ['world', 'c++']        |
| ['a1', 'equals', 'b1']     | ['equals', 'b1']        |
| ['hasnull', NULL, 'value'] | [NULL, 'value']         |
| ['hasnull', NULL, 'value'] | [NULL, 'value']         |
+---------------------------+-------------------------+

mysql> select k2, array_slice(k2, -2, 2) from array_type_table_nullable_varchar;
+---------------------------+-------------------------+
| k2                        | array_slice(`k2`, -2, 2) |
+---------------------------+-------------------------+
| ['hello', 'world', 'c++']  | ['world', 'c++']        |
| ['a1', 'equals', 'b1']     | ['equals', 'b1']        |
| ['hasnull', NULL, 'value'] | [NULL, 'value']         |
| ['hasnull', NULL, 'value'] | [NULL, 'value']         |
+---------------------------+-------------------------+
```

```
mysql> select k2, array_slice(k2, 0) from array_type_table;
+-----------+------------------------+
| k2        | array_slice(`k2`, 0) |
+-----------+------------------------+
| [1, 2, 3] | []                     |
+-----------+------------------------+

mysql> select k2, array_slice(k2, -5) from array_type_table;
+-----------+---------------------+
| k2        | array_slice(`k2`, -5) |
+-----------+---------------------+
```

```
| [1, 2, 3] | []                  |
+-----------+---------------------+
```

keywords

ARRAY,SLICE,ARRAY_SLICE

ARRAY_SORT

array_sort

array_sort

description

Syntax

```
ARRAY<T> array_sort(ARRAY<T> arr)
```

Return the array which has been sorted in ascending order. Return NULL for NULL input. If the element of array is NULL, it will be placed in the front of the sorted array.

example

``mysql> select k1, k2, array_sort(k2) from array_test; +------+----------------------------+---------------------------+ | k1 | k2 | array_sort(k2`) | +——+————————————+————————————+ | 1 | [1, 2, 3, 4, 5] | [1, 2, 3, 4, 5] | | 2 | [6, 7, 8] | [6, 7, 8] | | 3 | [] | [] | | 4 | NULL | NULL | | 5 | [1, 2, 3, 4, 5, 4, 3, 2, 1] | [1, 1, 2, 2, 3, 3, 4, 4, 5] | | 6 | [1, 2, 3, NULL] | [NULL, 1, 2, 3] | | 7 | [1, 2, 3, NULL, NULL] | [NULL, NULL, 1, 2, 3] | | 8 | [1, 1, 2, NULL, NULL] | [NULL, NULL, 1, 1, 2] | | 9 | [1, NULL, 1, 2, NULL, NULL] | [NULL, NULL, NULL, 1, 1, 2] | +——+————————————+————————————+

mysql> select k1, k2, array_sort(k2) from array_test01; +——+————————————————+ ————————————————+ | k1 | k2 | array_sort(k2) | +——+————————————————+ ————————————————+ | 1 | [‘a’, ‘b’, ‘c’, ‘d’, ‘e’] | [‘a’, ‘b’, ‘c’, ‘d’, ‘e’] | | 2 | [‘f’, ‘g’, ‘h’] | [‘f’, ‘g’, ‘h’] | | 3 | [’’] | [’’] | | 3 | [NULL] | [NULL] | | 5 | [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘a’, ‘b’, ‘c’] | [‘a’, ‘a’, ‘b’, ‘b’, ‘c’, ‘c’, ‘d’, ‘e’] | | 6 | NULL | NULL | | 7 | [‘a’, ‘b’, NULL] | [NULL, ‘a’, ‘b’] | | 8 | [‘a’, ‘b’, NULL, NULL] | [NULL, NULL, ‘a’, ‘b’] | +——+ ————————————————+————————————————+ ```

keywords

ARRAY, SORT, ARRAY_SORT

ARRAY_REVERSE_SORT

array_reverse_sort

array_reverse_sort

description

Syntax

```
ARRAY<T> array_reverse_sort(ARRAY<T> arr)
```

Return the array which has been sorted in descending order. Return NULL for NULL input. If the element of array is NULL, it will be placed in the last of the sorted array.

example

``mysql> select k1, k2, array_reverse_sort(k2) from array_test; +------+----------------------------+---------------------+ | k1 | k2 | array_reverse_sort(k2`) | +——+————————————+————————————+ | 1 | [1, 2, 3, 4, 5] | [5, 4, 3, 2, 1] | | 2 | [6, 7, 8] | [8, 7, 6] | | 3 | [] | [] | | 4 | NULL | NULL | | 5 | [1, 2, 3, 4, 5, 4, 3, 2, 1] | [5, 4, 4, 3, 3, 2, 2, 1, 1] | | 6 | [1, 2, 3, NULL] | [3, 2, 1, NULL] | | 7 | [1, 2, 3, NULL, NULL] | [3, 2, 1, NULL, NULL] | | 8 | [1, 1, 2, NULL, NULL] | [2, 1, 1, NULL, NULL] | | 9 | [1, NULL, 1, 2, NULL, NULL] | [2, 1, 1, NULL, NULL, NULL] | +——+————————————+————————————+

mysql> select k1, k2, array_reverse_sort(k2) from array_test01; +——+————————————————+————————————————+ | k1 | k2 | array_reverse_sort(k2) | +——+————————————————+————————————————+ | 1 | [‘a’, ‘b’, ‘c’, ‘d’, ‘e’] | [‘e’, ‘d’, ‘c’, ‘b’, ‘a’] | | 2 | [‘f’, ‘g’, ‘h’] | [‘h’, ‘g’, ‘f’] | | 3 | [’’] | [’’] | | 3 | [NULL] | [NULL] | | 5 | [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘a’, ‘b’, ‘c’] | [‘e’, ‘d’, ‘c’, ‘c’, ‘b’, ‘b’, ‘a’, ‘a’] | | 6 | NULL | NULL | | 7 | [‘a’, ‘b’, NULL] | [‘b’, ‘a’, NULL] | | 8 | [‘a’, ‘b’, NULL, NULL] | [‘b’, ‘a’, NULL, NULL] | +——+ ————————————————+————————————————+ ```

keywords

ARRAY, SORT, REVERSE, ARRAY_SORT, ARRAY_REVERSE_SORT

ARRAY_SORTBY

array_sortby

array_sortby

description

Syntax

```
ARRAY<T> array_sortby(ARRAY<T> src,Array<T> key)
ARRAY<T> array_sortby(lambda,array....)
```

First, arrange the key column in ascending order, and then return the corresponding column of the src column sorted in this order as the result; Returns NULL if the input array src is NULL. If the input array key is NULL, the order in which src is returned remains unchanged. If the input array key element contains NULL, the output sorted array will place NULL first.

example

```
mysql [test]>select array_sortby(['a','b','c'],[3,2,1]);
+-----------------------------------------------+
| array_sortby(ARRAY('a', 'b', 'c'), ARRAY(3, 2, 1)) |
+-----------------------------------------------+
| ['c', 'b', 'a']                               |
+-----------------------------------------------+

mysql [test]>select array_sortby([1,2,3,4,5],[10,5,1,20,80]);
+-------------------------------------------------------+
| array_sortby(ARRAY(1, 2, 3, 4, 5), ARRAY(10, 5, 1, 20, 80)) |
+-------------------------------------------------------+
| [3, 2, 1, 4, 5]                                       |
+-------------------------------------------------------+

mysql [test]>select *,array_sortby(c_array1,c_array2) from test_array_sortby order by id;
+------+-----------------+-------------------------+--------------------------------------+
| id   | c_array1        | c_array2                | array_sortby(`c_array1`, `c_array2`) |
+------+-----------------+-------------------------+--------------------------------------+
|    0 | NULL            | [2]                     | NULL                                 |
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [5, 3, 1, 2, 4]                      |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [6, 7, 8]                            |
|    3 | [1]             | [-100]                  | [1]                                  |
|    4 | NULL            | NULL                    | NULL                                 |
|    5 | [3]             | NULL                    | [3]                                  |
|    6 | [1, 2]          | [2, 1]                  | [2, 1]                               |
|    7 | [NULL]          | [NULL]                  | [NULL]                               |
|    8 | [1, 2, 3]       | [3, 2, 1]               | [3, 2, 1]                            |
+------+-----------------+-------------------------+--------------------------------------+

mysql [test]>select *, array_map((x,y)->(y+x),c_array1,c_array2) as arr_sum,array_sortby((x,y)->(y+x),c_array1,c_array2) as arr_sort
from array_test2;
+------+-----------------+--------------+----------------+-----------------+
| id   | c_array1        | c_array2     | arr_sum        | arr_sort        |
```

```
+------+----------------+-------------+---------------+-----------------+
|    1 | [1, 2, 3]      | [10, 11, 12] | [11, 13, 15]  | [1, 2, 3]       |
|    2 | [4, 3, 5]      | [10, 20, 30] | [14, 23, 35]  | [4, 3, 5]       |
|    3 | [-40, 30, -100] | [30, 10, 20] | [-10, 40, -80] | [-100, -40, 30] |
+------+----------------+-------------+---------------+-----------------+
```

keywords

ARRAY, SORT, ARRAY_SORTBY

ARRAY_POSITION

array_position

array_position

description

Syntax

```
BIGINT array_position(ARRAY<T> arr, T value)
```

Returns a position/index of first occurrence of the `value` in the given array.

```
position - value position in array (starts with 1);
0        - if value does not present in the array;
NULL     - when array is NULL.
```

example

```
mysql> SELECT id,c_array,array_position(c_array, 5) FROM `array_test`;
+------+-----------------+-----------------------------+
| id   | c_array         | array_position(`c_array`, 5) |
+------+-----------------+-----------------------------+
|    1 | [1, 2, 3, 4, 5] |                           5 |
|    2 | [6, 7, 8]       |                           0 |
|    3 | []              |                           0 |
|    4 | NULL            |                        NULL |
+------+-----------------+-----------------------------+

mysql> select array_position([1, null], null);
+---------------------------------------+
| array_position(ARRAY(1, NULL), NULL) |
+---------------------------------------+
|                                     2 |
+---------------------------------------+
1 row in set (0.01 sec)
```

keywords

ARRAY,POSITION,ARRAY_POSITION

ARRAY_CONTAINS

array_contains

array_contains

description

Syntax

```
BOOLEAN array_contains(ARRAY<T> arr, T value)
```

Check if a value presents in an array column. Return below values:

```
1    - if value presents in an array;
0    - if value does not present in an array;
NULL - when array is NULL;
```

example

```
mysql> SELECT id,c_array,array_contains(c_array, 5) FROM `array_test`;
+------+-----------------+-----------------------------+
| id   | c_array         | array_contains(`c_array`, 5) |
+------+-----------------+-----------------------------+
|    1 | [1, 2, 3, 4, 5] |                           1 |
|    2 | [6, 7, 8]       |                           0 |
|    3 | []              |                           0 |
|    4 | NULL            |                        NULL |
+------+-----------------+-----------------------------+

mysql> select array_contains([null, 1], null);
+--------------------------------------+
| array_contains(ARRAY(NULL, 1), NULL) |
+--------------------------------------+
|                                    1 |
+--------------------------------------+
1 row in set (0.00 sec)
```

keywords

ARRAY,CONTAIN,CONTAINS,ARRAY_CONTAINS

ARRAY_EXCEPT

array_except

array_except

description

Syntax

```
ARRAY<T> array_except(ARRAY<T> array1, ARRAY<T> array2)
```

Returns an array of the elements in array1 but not in array2, without duplicates. If the input parameter is null, null is returned.

example

```
mysql> select k1,k2,k3,array_except(k2,k3) from array_type_table;
+------+-----------------+--------------+--------------------------+
| k1   | k2              | k3           | array_except(`k2`, `k3`) |
+------+-----------------+--------------+--------------------------+
|    1 | [1, 2, 3]       | [2, 4, 5]    | [1, 3]                   |
|    2 | [2, 3]          | [1, 5]       | [2, 3]                   |
|    3 | [1, 1, 1]       | [2, 2, 2]    | [1]                      |
+------+-----------------+--------------+--------------------------+

mysql> select k1,k2,k3,array_except(k2,k3) from array_type_table_nullable;
+------+-----------------+--------------+--------------------------+
| k1   | k2              | k3           | array_except(`k2`, `k3`) |
+------+-----------------+--------------+--------------------------+
|    1 | [1, NULL, 3]    | [1, 3, 5]    | [NULL]                   |
|    2 | [NULL, NULL, 2] | [2, NULL, 4] | []                       |
|    3 | NULL            | [1, 2, 3]    | NULL                     |
+------+-----------------+--------------+--------------------------+

mysql> select k1,k2,k3,array_except(k2,k3) from array_type_table_varchar;
+------+---------------------------+---------------------------------+--------------------------+
| k1   | k2                        | k3                              | array_except(`k2`, `k3`) |
+------+---------------------------+---------------------------------+--------------------------+
|    1 | ['hello', 'world', 'c++'] | ['I', 'am', 'c++']              | ['hello', 'world']       |
|    2 | ['a1', 'equals', 'b1']    | ['a2', 'equals', 'b2']          | ['a1', 'b1']             |
|    3 | ['hasnull', NULL, 'value'] | ['nohasnull', 'nonull', 'value'] | ['hasnull', NULL]        |
|    3 | ['hasnull', NULL, 'value'] | ['hasnull', NULL, 'value']      | []                       |
+------+---------------------------+---------------------------------+--------------------------+

mysql> select k1,k2,k3,array_except(k2,k3) from array_type_table_decimal;
+------+-----------------+------------------+--------------------------+
| k1   | k2              | k3               | array_except(`k2`, `k3`) |
+------+-----------------+------------------+--------------------------+
|    1 | [1.1, 2.1, 3.44] | [2.1, 3.4, 5.4]  | [1.1, 3.44]              |
|    2 | [NULL, 2, 5]    | [NULL, NULL, 5.4] | [2, 5]                   |
|    1 | [1, NULL, 2, 5] | [1, 3.1, 5.4]    | [NULL, 2, 5]             |
+------+-----------------+------------------+--------------------------+
```

keywords

ARRAY,EXCEPT,ARRAY_EXCEPT

ARRAY_PRODUCT

array_product

array_product

description

Syntax

```
T array_product(ARRAY<T> arr)
```

Get the product of all elements in an array (`NULL` values are skipped). When the array is empty or all elements in the array are `NULL` values, the function returns `NULL`.

example

```
mysql> create table array_type_table(k1 INT, k2 Array<int>) duplicate key (k1)
    -> distributed by hash(k1) buckets 1 properties('replication_num' = '1');
mysql> insert into array_type_table values (0, []), (1, [NULL]), (2, [1, 2, 3]), (3, [1, NULL, 3]);
mysql> select k2, array_product(k2) from array_type_table;
+--------------+---------------------+
| k2           | array_product(`k2`) |
+--------------+---------------------+
| []           |                NULL |
| [NULL]       |                NULL |
| [1, 2, 3]    |                   6 |
| [1, NULL, 3] |                   3 |
+--------------+---------------------+
4 rows in set (0.01 sec)
```

keywords

ARRAY,PRODUCT,ARRAY_PRODUCT

ARRAY_INTERSECT

array_intersect

array_intersect

description

Syntax

```
ARRAY<T> array_intersect(ARRAY<T> array1, ARRAY<T> array2)
```

Returns an array of the elements in the intersection of array1 and array2, without duplicates. If the input parameter is null, null is returned.

example

```
mysql> select k1,k2,k3,array_intersect(k2,k3) from array_type_table;
+------+-----------------+--------------+----------------------------+
| k1   | k2              | k3           | array_intersect(`k2`, `k3`) |
+------+-----------------+--------------+----------------------------+
|    1 | [1, 2, 3]       | [2, 4, 5]    | [2]                        |
|    2 | [2, 3]          | [1, 5]       | []                         |
|    3 | [1, 1, 1]       | [2, 2, 2]    | []                         |
+------+-----------------+--------------+----------------------------+

mysql> select k1,k2,k3,array_intersect(k2,k3) from array_type_table_nullable;
+------+-----------------+--------------+----------------------------+
| k1   | k2              | k3           | array_intersect(`k2`, `k3`) |
+------+-----------------+--------------+----------------------------+
|    1 | [1, NULL, 3]    | [1, 3, 5]    | [1, 3]                     |
|    2 | [NULL, NULL, 2] | [2, NULL, 4] | [NULL, 2]                  |
|    3 | NULL            | [1, 2, 3]    | NULL                       |
+------+-----------------+--------------+----------------------------+

mysql> select k1,k2,k3,array_intersect(k2,k3) from array_type_table_varchar;
+------+---------------------------+---------------------------------+----------------------------+
| k1   | k2                        | k3                              | array_intersect(`k2`, `k3`) |
+------+---------------------------+---------------------------------+----------------------------+
|    1 | ['hello', 'world', 'c++'] | ['I', 'am', 'c++']              | ['c++']                    |
|    2 | ['a1', 'equals', 'b1']    | ['a2', 'equals', 'b2']          | ['equals']                 |
|    3 | ['hasnull', NULL, 'value'] | ['nohasnull', 'nonull', 'value'] | [NULL, 'value']            |
|    3 | ['hasnull', NULL, 'value'] | ['hasnull', NULL, 'value']      | ['hasnull', 'value']       |
+------+---------------------------+---------------------------------+----------------------------+

mysql> select k1,k2,k3,array_intersect(k2,k3) from array_type_table_decimal;
+------+-----------------+------------------+----------------------------+
| k1   | k2              | k3               | array_intersect(`k2`, `k3`) |
+------+-----------------+------------------+----------------------------+
|    1 | [1.1, 2.1, 3.44] | [2.1, 3.4, 5.4]  | [2.1]                      |
|    2 | [NULL, 2, 5]    | [NULL, NULL, 5.4] | [NULL]                     |
|    3 | [1, NULL, 2, 5] | [1, 3.1, 5.4]    | [1]                        |
+------+-----------------+------------------+----------------------------+
```

keywords

ARRAY,INTERSECT,ARRAY_INTERSECT

ARRAY_RANGE

array_range

array_range

description

Syntax

```
ARRAY<Int> array_range(Int end)
ARRAY<Int> array_range(Int start, Int end)
ARRAY<Int> array_range(Int start, Int end, Int step)
```

The parameters are all positive integers. start default value is 0, and step default value is 1. Return the array which numbers from start to end - 1 by step.

example

```
mysql> select array_range(10);
+--------------------------------+
| array_range(10)                |
+--------------------------------+
| [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] |
+--------------------------------+

mysql> select array_range(10,20);
+----------------------------------------+
| array_range(10, 20)                    |
+----------------------------------------+
| [10, 11, 12, 13, 14, 15, 16, 17, 18, 19] |
+----------------------------------------+

mysql> select array_range(0,20,2);
+-------------------------------------+
| array_range(0, 20, 2)               |
+-------------------------------------+
| [0, 2, 4, 6, 8, 10, 12, 14, 16, 18] |
+-------------------------------------+
```

keywords

ARRAY, RANGE, ARRAY_RANGE

ARRAY_DISTINCT

array_distinct

array_distinct

description

Syntax

```
ARRAY<T> array_distinct(ARRAY<T> arr)
```

Return the array which has been removed duplicate values. Return NULL for NULL input.

example

```
mysql> select k1, k2, array_distinct(k2) from array_test;
+------+-----------------------------+--------------------------+
| k1   | k2                          | array_distinct(k2)       |
+------+-----------------------------+--------------------------+
| 1    | [1, 2, 3, 4, 5]             | [1, 2, 3, 4, 5]          |
| 2    | [6, 7, 8]                   | [6, 7, 8]                |
| 3    | []                          | []                       |
| 4    | NULL                        | NULL                     |
| 5    | [1, 2, 3, 4, 5, 4, 3, 2, 1] | [1, 2, 3, 4, 5]          |
| 6    | [1, 2, 3, NULL]             | [1, 2, 3, NULL]          |
| 7    | [1, 2, 3, NULL, NULL]       | [1, 2, 3, NULL]    |
+------+-----------------------------+--------------------------+

mysql> select k1, k2, array_distinct(k2) from array_test01;
+------+----------------------------------------+--------------------------+
| k1   | k2                                     | array_distinct(`k2`)     |
+------+----------------------------------------+--------------------------+
| 1    | ['a', 'b', 'c', 'd', 'e']              | ['a', 'b', 'c', 'd', 'e'] |
| 2    | ['f', 'g', 'h']                        | ['f', 'g', 'h']          |
| 3    | ['']                                   | ['']                     |
| 3    | [NULL]                                 | [NULL]                   |
| 5    | ['a', 'b', 'c', 'd', 'e', 'a', 'b', 'c'] | ['a', 'b', 'c', 'd', 'e'] |
| 6    | NULL                                   | NULL                     |
| 7    | ['a', 'b', NULL]                       | ['a', 'b', NULL]         |
| 8    | ['a', 'b', NULL, NULL]                 | ['a', 'b', NULL]    |
+------+----------------------------------------+--------------------------+
```

keywords

ARRAY, DISTINCT, ARRAY_DISTINCT

ARRAY_DIFFERENCE

array_difference

array_difference

description

Syntax

```
ARRAY<T> array_difference(ARRAY<T> arr)
```

Calculates the difference between adjacent array elements. Returns an array where the first element will be 0, the second is the difference between a[1] - a[0]. need notice that NULL will be return NULL

example

```
mysql> select *,array_difference(k2) from array_type_table;
+------+---------------------------+---------------------------------+
| k1   | k2                        | array_difference(`k2`)          |
+------+---------------------------+---------------------------------+
|    0 | []                        | []                              |
|    1 | [NULL]                    | [NULL]                          |
|    2 | [1, 2, 3]                 | [0, 1, 1]                       |
|    3 | [1, NULL, 3]              | [0, NULL, NULL]                 |
|    4 | [0, 1, 2, 3, NULL, 4, 6]  | [0, 1, 1, 1, NULL, NULL, 2]     |
|    5 | [1, 2, 3, 4, 5, 4, 3, 2, 1] | [0, 1, 1, 1, 1, -1, -1, -1, -1] |
|    6 | [6, 7, 8]                 | [0, 1, 1]                       |
+------+---------------------------+---------------------------------+
```

keywords

ARRAY, DIFFERENCE, ARRAY_DIFFERENCE

ARRAY_UNION

array_union

array_union

description

Syntax

```
ARRAY<T> array_union(ARRAY<T> array1, ARRAY<T> array2)
```

Returns an array of the elements in the union of array1 and array2, without duplicates. If the input parameter is null, null is returned.

example

```
mysql> select k1,k2,k3,array_union(k2,k3) from array_type_table;
+------+-----------------+--------------+------------------------+
| k1   | k2              | k3           | array_union(`k2`, `k3`) |
+------+-----------------+--------------+------------------------+
|    1 | [1, 2, 3]       | [2, 4, 5]    | [1, 2, 3, 4, 5]         |
|    2 | [2, 3]          | [1, 5]       | [2, 3, 1, 5]            |
|    3 | [1, 1, 1]       | [2, 2, 2]    | [1, 2]                  |
+------+-----------------+--------------+------------------------+

mysql> select k1,k2,k3,array_union(k2,k3) from array_type_table_nullable;
+------+-----------------+--------------+------------------------+
| k1   | k2              | k3           | array_union(`k2`, `k3`) |
+------+-----------------+--------------+------------------------+
|    1 | [1, NULL, 3]    | [1, 3, 5]    | [1, NULL, 3, 5]         |
|    2 | [NULL, NULL, 2] | [2, NULL, 4] | [NULL, 2, 4]            |
|    3 | NULL            | [1, 2, 3]    | NULL                    |
+------+-----------------+--------------+------------------------+

mysql> select k1,k2,k3,array_union(k2,k3) from array_type_table_varchar;
+------+----------------------------+---------------------------------+-------------------------------------------------+
| k1   | k2                         | k3                              | array_union(`k2`, `k3`)                         |
+------+----------------------------+---------------------------------+-------------------------------------------------+
|    1 | ['hello', 'world', 'c++']  | ['I', 'am', 'c++']              | ['hello', 'world', 'c++', 'I', 'am']            |
|    2 | ['a1', 'equals', 'b1']     | ['a2', 'equals', 'b2']          | ['a1', 'equals', 'b1', 'a2', 'b2']              |
|    3 | ['hasnull', NULL, 'value'] | ['nohasnull', 'nonull', 'value'] | ['hasnull', NULL, 'value', 'nohasnull', 'nonull'] |
|    4 | ['hasnull', NULL, 'value'] | ['hasnull', NULL, 'value']      | ['hasnull', NULL, 'value']                      |
+------+----------------------------+---------------------------------+-------------------------------------------------+

mysql> select k1,k2,k3,array_union(k2,k3) from array_type_table_decimal;
+------+-----------------+------------------+--------------------------+
| k1   | k2              | k3               | array_union(`k2`, `k3`)  |
+------+-----------------+------------------+--------------------------+
|    1 | [1.1, 2.1, 3.44] | [2.1, 3.4, 5.4]  | [1.1, 2.1, 3.44, 3.4, 5.4] |
|    2 | [NULL, 2, 5]     | [NULL, NULL, 5.4] | [NULL, 2, 5, 5.4]          |
|    4 | [1, NULL, 2, 5]  | [1, 3.1, 5.4]    | [1, NULL, 2, 5, 3.1, 5.4]  |
+------+-----------------+------------------+--------------------------+
```

keywords

ARRAY,UNION,ARRAY_UNION

ARRAY_JOIN

array_join

array_join

description

Syntax

```
VARCHAR array_join(ARRAY<T> arr, VARCHAR sep[, VARCHAR null_replace])
```

Combines all elements in the array to generate a new string according to the separator (sep) and the string to replace NULL (null_replace). If sep is NULL, return NULL. If null_replace is NULL, return NULL. If sep is an empty string, no delimiter is applied. If null_replace is an empty string or not specified, the NULL elements in the array are discarded directly.

example

```
mysql> select k1, k2, array_join(k2, '_', 'null') from array_test order by k1;
+------+----------------------------+-----------------------------------+
| k1   | k2                         | array_join(`k2`, '_', 'null')     |
+------+----------------------------+-----------------------------------+
|  1   | [1, 2, 3, 4, 5]            | 1_2_3_4_5                         |
|  2   | [6, 7, 8]                  | 6_7_8                             |
|  3   | []                         |                                   |
|  4   | NULL                       | NULL                              |
|  5   | [1, 2, 3, 4, 5, 4, 3, 2, 1] | 1_2_3_4_5_4_3_2_1                 |
|  6   | [1, 2, 3, NULL]            | 1_2_3_null                        |
|  7   | [4, 5, 6, NULL, NULL]      | 4_5_6_null_null                   |
+------+----------------------------+-----------------------------------+

mysql> select k1, k2, array_join(k2, '_', 'null') from array_test01 order by k1;
+------+----------------------------------+-----------------------------------+
| k1   | k2                               | array_join(`k2`, '_', 'null')     |
+------+----------------------------------+-----------------------------------+
|  1   | ['a', 'b', 'c', 'd']             | a_b_c_d                           |
|  2   | ['e', 'f', 'g', 'h']             | e_f_g_h                           |
|  3   | [NULL, 'a', NULL, 'b', NULL, 'c'] | null_a_null_b_null_c              |
|  4   | ['d', 'e', NULL, ' ']            | d_e_null_                         |
|  5   | [' ', NULL, 'f', 'g']            |  _null_f_g                        |
+------+----------------------------------+-----------------------------------+

mysql> select k1, k2, array_join(k2, '_') from array_test order by k1;
+------+----------------------------+---------------------------+
| k1   | k2                         | array_join(`k2`, '_')     |
+------+----------------------------+---------------------------+
|  1   | [1, 2, 3, 4, 5]            | 1_2_3_4_5                 |
|  2   | [6, 7, 8]                  | 6_7_8                     |
|  3   | []                         |                           |
|  4   | NULL                       | NULL                      |
|  5   | [1, 2, 3, 4, 5, 4, 3, 2, 1] | 1_2_3_4_5_4_3_2_1         |
|  6   | [1, 2, 3, NULL]            | 1_2_3                     |
|  7   | [4, 5, 6, NULL, NULL]      | 4_5_6                     |
```

```
+------+-----------------------------+---------------------------+

mysql> select k1, k2, array_join(k2, '_') from array_test01 order by k1;
+------+---------------------------------+---------------------------+
| k1   | k2                              | array_join(`k2`, '_')     |
+------+---------------------------------+---------------------------+
|  1   | ['a', 'b', 'c', 'd']            | a_b_c_d                   |
|  2   | ['e', 'f', 'g', 'h']            | e_f_g_h                   |
|  3   | [NULL, 'a', NULL, 'b', NULL, 'c'] | a_b_c                   |
|  4   | ['d', 'e', NULL, ' ']           | d_e_                      |
|  5   | [' ', NULL, 'f', 'g']           |  _f_g                     |
+------+---------------------------------+---------------------------+
```

keywords

ARRAY, JOIN, ARRAY_JOIN

ARRAY_WITH_CONSTANT

array_with_constant

array_with_constant

description

Syntax

```
ARRAY<T> array_with_constant(n, T)
ARRAY<T> array_repeat(T, n)
```

get array of constants with n length, array_repeat has the same function as array_with_constant and is used to be compatible with the hive syntax format #### example

```
mysql> select array_with_constant(2, "hello"), array_repeat("hello", 2);
+---------------------------------+--------------------------+
| array_with_constant(2, 'hello') | array_repeat('hello', 2) |
+---------------------------------+--------------------------+
| ['hello', 'hello']              | ['hello', 'hello']       |
+---------------------------------+--------------------------+
1 row in set (0.04 sec)

mysql> select array_with_constant(3, 12345), array_repeat(12345, 3);
+-------------------------------+-----------------------+
| array_with_constant(3, 12345) | array_repeat(12345, 3) |
+-------------------------------+-----------------------+
| [12345, 12345, 12345]         | [12345, 12345, 12345]  |
+-------------------------------+-----------------------+
1 row in set (0.01 sec)

mysql> select array_with_constant(3, null), array_repeat(null, 3);
+-----------------------------+----------------------+
| array_with_constant(3, NULL) | array_repeat(NULL, 3) |
+-----------------------------+----------------------+
| [NULL, NULL, NULL]           |  [NULL, NULL, NULL]   |
+-----------------------------+----------------------+
1 row in set (0.01 sec)

mysql> select array_with_constant(null, 3), array_repeat(3, null);
+-----------------------------+----------------------+
| array_with_constant(NULL, 3) | array_repeat(3, NULL) |
+-----------------------------+----------------------+
| []                          | []                   |
+-----------------------------+----------------------+
1 row in set (0.01 sec)
```

keywords

ARRAY,WITH_CONSTANT,ARRAY_WITH_CONSTANT,ARRAY_REPEAT

ARRAY_ENUMERATE

array_enumerate

array_enumerate

description

Syntax

```
ARRAY<T> array_enumerate(ARRAY<T> arr)
```

Returns array sub item indexes eg. [1, 2, 3, …, length (arr)]

example

```
mysql> create table array_type_table(k1 INT, k2 Array<STRING>) duplicate key (k1)
    -> distributed by hash(k1) buckets 1 properties('replication_num' = '1');
mysql> insert into array_type_table values (0, []), ("1", [NULL]), ("2", ["1", "2", "3"]), ("3", ["1", NULL, "3"]), ("4", NULL);
mysql> select k2, array_enumerate(k2) from array_type_table;
+------------------+-----------------------+
| k2               | array_enumerate(`k2`) |
+------------------+-----------------------+
| []               | []                    |
| [NULL]           | [1]                   |
| ['1', '2', '3']  | [1, 2, 3]             |
| ['1', NULL, '3'] | [1, 2, 3]             |
| NULL             | NULL                  |
+------------------+-----------------------+
5 rows in set (0.01 sec)
```

keywords

ARRAY,ENUMERATE,ARRAY_ENUMERATE

ARRAY_ENUMERATE_UNIQ

array_enumerate_uniq

array_enumerate_uniq

description

Syntax

```
ARRAY<T> array_enumerate_uniq(ARRAY<T> arr)
```

Returns an array the same size as the source array, indicating for each element what its position is among elements with the same value. For example, array_enumerate_uniq([1, 2, 1, 4]) = [1, 1, 2, 1]. The array_enumerate_uniq function can take multiple arrays of the same size as arguments. In this case, uniqueness is considered for tuples of elements in the same positions in all the arrays. For example, array_enumerate_uniq([1, 2, 1, 1, 2], [2, 1, 2, 2, 1]) = [1, 1, 2, 3, 2].

example

```
mysql> select k2, array_enumerate_uniq([1, 2, 3, 1, 2, 3]);
+-----------------------------------------------------+
| array_enumerate_uniq(ARRAY(1, 2, 3, 1, 2, 3))       |
+-----------------------------------------------------+
| [1, 1, 1, 2, 2, 2]                                  |
+-----------------------------------------------------+
mysql> select array_enumerate_uniq([1, 1, 1, 1, 1], [2, 1, 2, 1, 2], [3, 1, 3, 1, 3]);
+-------------------------------------------------------------------------------------+
| array_enumerate_uniq(ARRAY(1, 1, 1, 1, 1), ARRAY(2, 1, 2, 1, 2), ARRAY(3, 1, 3, 1, 3)) |
+-------------------------------------------------------------------------------------+
| [1, 1, 2, 1, 3]                                                                     |
+-------------------------------------------------------------------------------------+
```

keywords

ARRAY,ENUMERATE_UNIQ,ARRAY_ENUMERATE_UNIQ

ARRAY_POPBACK

array_popback

array_popback

description

Syntax

```
ARRAY<T> array_popback(ARRAY<T> arr)
```

Remove the last element from array.

example

```
mysql> select array_popback(['test', NULL, 'value']);
+-----------------------------------------------------+
| array_popback(ARRAY('test', NULL, 'value'))         |
+-----------------------------------------------------+
| [test, NULL]                                        |
+-----------------------------------------------------+
```

keywords

ARRAY,POPBACK,ARRAY_POPBACK

ARRAY_POPFRONT

array_popfront

array_popfront

description

Syntax

```
ARRAY<T> array_popfront(ARRAY<T> arr)
```

Remove the first element from array.

example

```
mysql> select array_popfront(['test', NULL, 'value']);
+-----------------------------------------------------+
| array_popfront(ARRAY('test', NULL, 'value'))        |
+-----------------------------------------------------+
| [NULL, value]                                       |
+-----------------------------------------------------+
```

keywords

ARRAY,POPFRONT,ARRAY_POPFRONT

ARRAY_PUSHFRONT

array_pushfront

array_pushfront

description

Syntax

```
Array<T> array_pushfront(Array<T> arr, T value)
```

Add the value to the beginning of the array.

Returned value

The array after adding the value.

Type: Array.

example

```
mysql> select array_pushfront([1, 2], 3);
+---------------------------------+
| array_pushfront(ARRAY(1, 2), 3) |
+---------------------------------+
| [3, 1, 2]                       |
+---------------------------------+

mysql> select col3, array_pushfront(col3, 6) from array_test;
+-----------+---------------------------+
| col3      | array_pushfront(`col3`, 6) |
+-----------+---------------------------+
| [3, 4, 5] | [6, 3, 4, 5]              |
| [NULL]    | [6, NULL]                 |
| NULL      | NULL                      |
| []        | [6]                       |
+-----------+---------------------------+

mysql> select col1, col3, array_pushfront(col3, col1) from array_test;
+------+-----------+--------------------------------+
| col1 | col3      | array_pushfront(`col3`, `col1`) |
+------+-----------+--------------------------------+
|    0 | [3, 4, 5] | [0, 3, 4, 5]                   |
|    1 | [NULL]    | [1, NULL]                      |
|    2 | NULL      | NULL                           |
|    3 | []        | [3]                            |
+------+-----------+--------------------------------+
```

keywords

ARRAY,PUSHFRONT,ARRAY_PUSHFRONT

ARRAY_PUSHBACK

array_pushback

array_pushback

description

Syntax

```
Array<T> array_pushback(Array<T> arr, T value)
```

Add the value to the end of the array.

Returned value

The array after adding the value.

Type: Array.

example

```
mysql> select array_pushback([1, 2], 3);
+---------------------------------+
| array_pushback(ARRAY(1, 2), 3)  |
+---------------------------------+
| [1, 2, 3]                       |
+---------------------------------+

mysql> select col3, array_pushback(col3, 6) from array_test;
+-----------+---------------------------+
| col3      | array_pushback(`col3`, 6)  |
+-----------+---------------------------+
| [3, 4, 5] | [3, 4, 5, 6]              |
| [NULL]    | [NULL, 6]                 |
| NULL      | NULL                      |
| []        | [6]                       |
+-----------+---------------------------+

mysql> select col1, col3, array_pushback(col3, col1) from array_test;
+------+-----------+--------------------------------+
| col1 | col3      | array_pushback(`col3`, `col1`)  |
+------+-----------+--------------------------------+
|    0 | [3, 4, 5] | [3, 4, 5, 0]                   |
|    1 | [NULL]    | [NULL, 1]                      |
|    2 | NULL      | NULL                           |
|    3 | []        | [3]                            |
+------+-----------+--------------------------------+
```

keywords

ARRAY,PUSHBACK,ARRAY_PUSHBACK

ARRAY_COMPACT

array_compact

array_compact

description

Removes consecutive duplicate elements from an array. The order of result values is determined by the order in the source array.

Syntax

```
Array<T> array_compact(arr)
```

Arguments

`arr` — The array to inspect.

Returned value

The array without continuous duplicate.

Type: Array.

example

```
select array_compact([1, 2, 3, 3, null, null, 4, 4]);

+--------------------------------------------------+
| array_compact(ARRAY(1, 2, 3, 3, NULL, NULL, 4, 4)) |
+--------------------------------------------------+
| [1, 2, 3, NULL, 4]                               |
+--------------------------------------------------+

select array_compact(['aaa','aaa','bbb','ccc','ccccc',null, null,'dddd']);

+-------------------------------------------------------------------------------+
| array_compact(ARRAY('aaa', 'aaa', 'bbb', 'ccc', 'ccccc', NULL, NULL, 'dddd')) |
+-------------------------------------------------------------------------------+
| ['aaa', 'bbb', 'ccc', 'ccccc', NULL, 'dddd']                                  |
+-------------------------------------------------------------------------------+

select array_compact(['2015-03-13','2015-03-13']);

+--------------------------------------------------+
| array_compact(ARRAY('2015-03-13', '2015-03-13')) |
+--------------------------------------------------+
| ['2015-03-13']                                   |
+--------------------------------------------------+
```

keywords

ARRAY,COMPACT,ARRAY_COMPACT

ARRAY_CONCAT

array_concat

array_concat

description

Concat all arrays passed in the arguments

Syntax

```
Array<T> array_concat(Array<T>, ...)
```

Returned value

The concated array.

Type: Array.

example

```
mysql> select array_concat([1, 2], [7, 8], [5, 6]);
+-----------------------------------------------+
| array_concat(ARRAY(1, 2), ARRAY(7, 8), ARRAY(5, 6)) |
+-----------------------------------------------+
| [1, 2, 7, 8, 5, 6]                            |
+-----------------------------------------------+
1 row in set (0.02 sec)

mysql> select col2, col3, array_concat(col2, col3) from array_test;
+--------------+-----------+-----------------------------+
| col2         | col3      | array_concat(`col2`, `col3`) |
+--------------+-----------+-----------------------------+
| [1, 2, 3]    | [3, 4, 5] | [1, 2, 3, 3, 4, 5]          |
| [1, NULL, 2] | [NULL]    | [1, NULL, 2, NULL]          |
| [1, 2, 3]    | NULL      | NULL                        |
| []           | []        | []                          |
+--------------+-----------+-----------------------------+
```

keywords

ARRAY,CONCAT,ARRAY_CONCAT

ARRAY_ZIP

array_zip

array_zip

description

Combines all all arrays into a single array. The resulting array contains the corresponding elements of the source arrays grouped into structs in the listed order of arguments.

Syntax

```
Array<Struct<T1, T2,...>> array_zip(Array<T1>, Array<T2>, ...)
```

Returned value

Array with elements from the source arrays grouped into tuples. Data types in the tuple are the same as types of the input arrays and in the same order as arrays are passed.

example

```
mysql> select array_zip(['a', 'b', 'c'], [1, 2, 3]);
+-----------------------------------------------+
| array_zip(ARRAY('a', 'b', 'c'), ARRAY(1, 2, 3)) |
+-----------------------------------------------+
| [{'a', 1}, {'b', 2}, {'c', 3}]                |
+-----------------------------------------------+
1 row in set (0.01 sec)
```

keywords

ARRAY,ZIP,ARRAY_ZIP

ARRAY_SHUFFLE

array_shuffle

array_shuffle shuffle

description

Syntax

```
ARRAY<T> array_shuffle(ARRAY<T> array1, [INT seed])
ARRAY<T> shuffle(ARRAY<T> array1, [INT seed])
```

Randomly arrange the elements in the array. Among them, the parameter array1 is the array to be randomly arranged, and the optional parameter seed is to set the initial value used by the pseudo-random number generator to generate pseudo-random numbers. Shuffle has the same function as array_shuffle.

```
array_shuffle(array1);
array_shuffle(array1, 0);
shuffle(array1);
shuffle(array1, 0);
```

example

```
mysql [test]> select c_array1, array_shuffle(c_array1) from array_test;
+----------------------+--------------------------+
| c_array1             | array_shuffle(`c_array1`) |
+----------------------+--------------------------+
| [1, 2, 3, 4, 5, NULL] | [2, NULL, 5, 3, 4, 1]     |
| [6, 7, 8, NULL]       | [7, NULL, 8, 6]           |
| [1, NULL]             | [1, NULL]                 |
| NULL                  | NULL                      |
+----------------------+--------------------------+
4 rows in set (0.01 sec)

MySQL [test]> select c_array1, array_shuffle(c_array1, 0) from array_test;
+----------------------+-----------------------------+
| c_array1             | array_shuffle(`c_array1`, 0) |
+----------------------+-----------------------------+
| [1, 2, 3, 4, 5, NULL] | [1, 3, 2, NULL, 4, 5]        |
| [6, 7, 8, NULL]       | [6, 8, 7, NULL]              |
| [1, NULL]             | [1, NULL]                    |
| NULL                  | NULL                         |
+----------------------+-----------------------------+
4 rows in set (0.01 sec)
```

keywords

ARRAY,ARRAY_SHUFFLE,SHUFFLE

ARRAY_CUM_SUM

array_cum_sum

array_cum_sum

description

Get the cumulative sum of an array (`NULL` values are skipped). If the array contains `NULL` values, then `NULL` is set at the same position in the result array.

Syntax

```
Array<T> array_cum_sum(Array<T>)
```

example

```
mysql> create table array_type_table(k1 INT, k2 Array<int>) duplicate key (k1) distributed by hash(k1) buckets 1
properties('replication_num' = '1');
mysql> insert into array_type_table values (0, []), (1, [NULL]), (2, [1, 2, 3, 4]), (3, [1, NULL, 3, NULL, 5]);
mysql> select k2, array_cum_sum(k2) from array_type_table;
+-----------------------+-----------------------+
| k2                    | array_cum_sum(`k2`)   |
+-----------------------+-----------------------+
| []                    | []                    |
| [NULL]                | [NULL]                |
| [1, 2, 3, 4]          | [1, 3, 6, 10]         |
| [1, NULL, 3, NULL, 5] | [1, NULL, 4, NULL, 9] |
+-----------------------+-----------------------+

4 rows in set
Time: 0.122s
```

keywords

ARRAY,CUM_SUM,ARRAY_CUM_SUM

ARRAY_EXISTS

array_exists

array_exists(lambda,array1,array2….) array_exists(array1)

description

Syntax

```
BOOLEAN array_exists(lambda, ARRAY<T> arr1, ARRAY<T> arr2, ... )
BOOLEAN array_exists(ARRAY<T> arr)
```

Use an optional lambda expression as an input parameter to perform corresponding expression calculations on the internal data of other input ARRAY parameters. Returns 1 when the calculation returns something other than 0; otherwise returns 0. There are one or more parameters input in the lambda expression, which must be consistent with the number of input array columns later. Legal scalar functions can be executed in lambda, aggregate functions, etc. are not supported. When lambda expression is not used as a parameter, array1 is used as the calculation result.

```
array_exists(x->x, array1);
array_exists(x->(x%2 = 0), array1);
array_exists(x->(abs(x)-1), array1);
array_exists((x,y)->(x = y), array1, array2);
array_exists(array1);
```

example

```
mysql [test]>select *, array_exists(x->x>1,[1,2,3]) from array_test2 order by id;
+------+-----------------+-------------------------+-----------------------------------------------+
| id   | c_array1        | c_array2                | array_exists([x] -> x(0) > 1, ARRAY(1, 2, 3)) |
+------+-----------------+-------------------------+-----------------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [0, 1, 1]                                     |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [0, 1, 1]                                     |
|    3 | [1]             | [-100]                  | [0, 1, 1]                                     |
|    4 | NULL            | NULL                    | [0, 1, 1]                                     |
+------+-----------------+-------------------------+-----------------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select c_array1, c_array2, array_exists(x->x%2=0,[1,2,3]) from array_test2 order by id;
+-----------------+-------------------------+-------------------------------------------------+
| c_array1        | c_array2                | array_exists([x] -> x(0) % 2 = 0, ARRAY(1, 2, 3)) |
+-----------------+-------------------------+-------------------------------------------------+
| [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [0, 1, 0]                                       |
| [6, 7, 8]       | [10, 12, 13]            | [0, 1, 0]                                       |
| [1]             | [-100]                  | [0, 1, 0]                                       |
| NULL            | NULL                    | [0, 1, 0]                                       |
+-----------------+-------------------------+-------------------------------------------------+
4 rows in set (0.02 sec)
```

```
mysql [test]>select c_array1, c_array2, array_exists(x->abs(x)-1,[1,2,3]) from array_test2 order by id;
+-----------------+------------------------+--------------------------------------------------+
| c_array1        | c_array2               | array_exists([x] -> abs(x(0)) - 1, ARRAY(1, 2, 3)) |
+-----------------+------------------------+--------------------------------------------------+
| [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [0, 1, 1, 1, 1]                                  |
| [6, 7, 8]       | [10, 12, 13]            | [1, 1, 1]                                        |
| [1, NULL]       | [-100]                  | [0, NULL]                                        |
| NULL            | NULL                    | NULL                                             |
+-----------------+------------------------+--------------------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select c_array1, c_array2, array_exists((x,y)->x>y,c_array1,c_array2) from array_test2 order by id;
+-----------------+------------------------+-----------------------------------------------------------------+
| c_array1        | c_array2               | array_exists([x, y] -> x(0) > y(1), `c_array1`, `c_array2`) |
+-----------------+------------------------+-----------------------------------------------------------------+
| [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] | [0, 0, 1, 0, 1]                                                 |
| [6, 7, 8]       | [10, 12, 13]            | [0, 0, 0]                                                       |
| [1]             | [-100]                  | [1]                                                             |
| NULL            | NULL                    | NULL                                                            |
+-----------------+------------------------+-----------------------------------------------------------------+
4 rows in set (0.02 sec)

mysql [test]>select *, array_exists(c_array1) from array_test2 order by id;
+------+-----------------+------------------------+-------------------------+
| id   | c_array1        | c_array2               | array_exists(`c_array1`) |
+------+-----------------+------------------------+-------------------------+
|    1 | [1, 2, 3, 0, 5] | [10, 20, -40, 80, -100] | [1, 1, 1, 0, 1]         |
|    2 | [6, 7, 8]       | [10, 12, 13]            | [1, 1, 1]               |
|    3 | [0, NULL]       | [-100]                  | [0, NULL]               |
|    4 | NULL            | NULL                    | NULL                    |
+------+-----------------+------------------------+-------------------------+
4 rows in set (0.02 sec)
```

keywords

ARRAY,ARRAY_EXISTS

ARRAY_FIRST_INDEX

array_first_index

array_first_index

description

Syntax

```
ARRAY<T> array_first_index(lambda, ARRAY<T> array1, ...)
```

Use an lambda expression as an input parameter to perform corresponding expression calculations on the internal data of other input ARRAY parameters. Returns the first index such that the return value of `lambda(array1[i], ...)` is not 0. Return 0 if such index is not found.

There are one or more parameters input in the lambda expression, and the number of elements of all input arrays must be the same. Legal scalar functions can be executed in lambda, aggregate functions, etc. are not supported.

```
array_first_index(x->x>1, array1);
array_first_index(x->(x%2 = 0), array1);
array_first_index(x->(abs(x)-1), array1);
array_first_index((x,y)->(x = y), array1, array2);
```

example

```
mysql> select array_first_index(x->x+1>3, [2, 3, 4]);
+-----------------------------------------------------------------+
| array_first_index(array_map([x] -> x(0) + 1 > 3, ARRAY(2, 3, 4))) |
+-----------------------------------------------------------------+
|                                                               2 |
+-----------------------------------------------------------------+

mysql> select array_first_index(x -> x is null, [null, 1, 2]);
+--------------------------------------------------------------------+
| array_first_index(array_map([x] -> x(0) IS NULL, ARRAY(NULL, 1, 2))) |
+--------------------------------------------------------------------+
|                                                                  1 |
+--------------------------------------------------------------------+

mysql> select array_first_index(x->power(x,2)>10, [1, 2, 3, 4]);
+-----------------------------------------------------------------------------+
| array_first_index(array_map([x] -> power(x(0), 2.0) > 10.0, ARRAY(1, 2, 3, 4))) |
+-----------------------------------------------------------------------------+
|                                                                           4 |
+-----------------------------------------------------------------------------+

mysql> select col2, col3, array_first_index((x,y)->x>y, col2, col3) from array_test;
+--------------+--------------+-------------------------------------------------------------------------+
| col2         | col3         | array_first_index(array_map([x, y] -> x(0) > y(1), `col2`, `col3`)) |
+--------------+--------------+-------------------------------------------------------------------------+
```

```
| [1, 2, 3]    | [3, 4, 5]    |                                                                    0 |
| [1, NULL, 2] | [NULL, 3, 1] |                                                                    3 |
| [1, 2, 3]    | [9, 8, 7]    |                                                                    0 |
| NULL         | NULL         |                                                                    0 |
+--------------+--------------+----------------------------------------------------------------------+
```

keywords

ARRAY,FIRST_INDEX,ARRAY_FIRST_INDEX

ARRAY_LAST_INDEX

array_last_index

array_last_index

description

Syntax

```
ARRAY<T> array_last_index(lambda, ARRAY<T> array1, ...)
```

Use an lambda expression as an input parameter to perform corresponding expression calculations on the internal data of other input ARRAY parameters. Returns the last index such that the return value of `lambda(array1[i], ...)` is not 0. Return 0 if such index is not found.

There are one or more parameters input in the lambda expression, and the number of elements of all input arrays must be the same. Legal scalar functions can be executed in lambda, aggregate functions, etc. are not supported.

```
array_last_index(x->x>1, array1);
array_last_index(x->(x%2 = 0), array1);
array_last_index(x->(abs(x)-1), array1);
array_last_index((x,y)->(x = y), array1, array2);
```

example

```
mysql> select array_last_index(x -> x is null, [null, null, 1, 2]);
+-----------------------------------------------------------------------+
| array_last_index(array_map([x] -> x IS NULL, ARRAY(NULL, NULL, 1, 2))) |
+-----------------------------------------------------------------------+
|                                                                     2 |
+-----------------------------------------------------------------------+


mysql> select array_last_index(x->x='s', ['a', 's', 's', 's', 'b']);
+-------------------------------------------------------------------------------+
| array_last_index(array_map([x] -> x = 's', ARRAY('a', 's', 's', 's', 'b'))) |
+-------------------------------------------------------------------------------+
|                                                                             4 |
+-------------------------------------------------------------------------------+

mysql> select array_last_index(x->power(x,2)>10, [1, 4, 3, 4]);
+-------------------------------------------------------------------------------+
| array_last_index(array_map([x] -> power(x, 2.0) > 10.0, ARRAY(1, 4, 3, 4))) |
+-------------------------------------------------------------------------------+
|                                                                             4 |
+-------------------------------------------------------------------------------+

mysql> select col2, col3, array_last_index((x,y)->x>y, col2, col3) from array_test;
+--------------+--------------+-------------------------------------------------------------------------+
| col2         | col3         | array_last_index(array_map([x, y] -> x(0) > y(1), `col2`, `col3`)) |
```

```
+--------------+--------------+------------------------------------------------------------------+
| [1, 2, 3]    | [3, 4, 5]    |                                                                0 |
| [1, NULL, 2] | [NULL, 3, 1] |                                                                3 |
| [1, 2, 3]    | [9, 8, 7]    |                                                                0 |
| NULL         | NULL         |                                                                0 |
+--------------+--------------+------------------------------------------------------------------+
```

keywords

ARRAY,FIRST_INDEX,array_last_index

ARRAY_FIRST

array_first

array_first

description

Returns the first element in the array for which func(arr1[i]) returns something other than 0.

Syntax

```
T array_first(lambda, ARRAY<T>)
```

Use a lambda bool expression and an array as the input parameters, the lambda expression is used to evaluate the internal data of other input ARRAY parameters.

example

```
mysql> select array_first(x->x>2, [1,2,3,0]) ;
+-----------------------------------------------------------------------------------------------+
| array_first(array_filter(ARRAY(1, 2, 3, 0), array_map([x] -> x(0) > 2, ARRAY(1, 2, 3, 0))), -1) |
+-----------------------------------------------------------------------------------------------+
|                                                                                             3 |
+-----------------------------------------------------------------------------------------------+


mysql> select array_first(x->x>4, [1,2,3,0]) ;
+-----------------------------------------------------------------------------------------------+
| array_first(array_filter(ARRAY(1, 2, 3, 0), array_map([x] -> x(0) > 4, ARRAY(1, 2, 3, 0))), -1) |
+-----------------------------------------------------------------------------------------------+
|                                                                                          NULL |
+-----------------------------------------------------------------------------------------------+


mysql> select array_first(x->x>1, [1,2,3,0]) ;
+-----------------------------------------------------------------------------------------------+
| array_first(array_filter(ARRAY(1, 2, 3, 0), array_map([x] -> x > 1, ARRAY(1, 2, 3, 0))), 1) |
+-----------------------------------------------------------------------------------------------+
|                                                                                             2 |
+-----------------------------------------------------------------------------------------------+
```

keywords

ARRAY, LAST, array_first

ARRAY_LAST

array_last

array_last

description

Returns the last element in the array for which func(arr1[i]) returns something other than 0.

Syntax

```
T array_last(lambda, ARRAY<T>)
```

Use a lambda bool expression and an array as the input parameters, the lambda expression is used to evaluate the internal data of other input ARRAY parameters.

example

```
mysql> select array_last(x->x>2, [1,2,3,0]) ;
+-----------------------------------------------------------------------------------------------+
| array_last(array_filter(ARRAY(1, 2, 3, 0), array_map([x] -> x(0) > 2, ARRAY(1, 2, 3, 0))), -1) |
+-----------------------------------------------------------------------------------------------+
|                                                                                             3 |
+-----------------------------------------------------------------------------------------------+


mysql> select array_last(x->x>4, [1,2,3,0]) ;
+-----------------------------------------------------------------------------------------------+
| array_last(array_filter(ARRAY(1, 2, 3, 0), array_map([x] -> x(0) > 4, ARRAY(1, 2, 3, 0))), -1) |
+-----------------------------------------------------------------------------------------------+
|                                                                                          NULL |
+-----------------------------------------------------------------------------------------------+
```

keywords

ARRAY, LAST, ARRAY_LAST

ARRAYS_OVERLAP

arrays_overlap

arrays_overlap

description

Syntax

```
BOOLEAN arrays_overlap(ARRAY<T> left, ARRAY<T> right)
```

Check if there is any common element for left and right array. Return below values:

```
1    - if any common element inside left and right array;
0    - if no common element inside left and right array;
NULL - when left or right array is NULL; OR any element inside left and right array is NULL;
```

example

```
mysql> select c_left,c_right,arrays_overlap(c_left,c_right) from array_test;
+--------------+-----------+-----------------------------------+
| c_left       | c_right   | arrays_overlap(`c_left`, `c_right`) |
+--------------+-----------+-----------------------------------+
| [1, 2, 3]    | [3, 4, 5] |                                 1 |
| [1, 2, 3]    | [5, 6]    |                                 0 |
| [1, 2, NULL] | [1]       |                              NULL |
| NULL         | [1, 2]    |                              NULL |
| [1, 2, 3]    | [1, 2]    |                                 1 |
+--------------+-----------+-----------------------------------+
```

keywords

ARRAY,ARRAYS,OVERLAP,ARRAYS_OVERLAP

ARRAY_COUNT

array_count

array_count

description

```
array_count(lambda, array1, ...)
```

Use lambda expressions as input parameters to perform corresponding expression calculations on the internal data of other input ARRAY parameters. Returns the number of elements such that the return value of `lambda(array1[i], ...)` is not 0. Returns 0 if no element is found that satisfies this condition.

There are one or more parameters are input in the lambda expression, which must be consistent with the number of input array columns later.The number of elements of all input arrays must be the same. Legal scalar functions can be executed in lambda, aggregate functions, etc. are not supported.

```
array_count(x->x, array1);
array_count(x->(x%2 = 0), array1);
array_count(x->(abs(x)-1), array1);
array_count((x,y)->(x = y), array1, array2);
```

example

```
mysql> select array_count(x -> x, [0, 1, 2, 3]);
+-----------------------------------------------------+
| array_count(array_map([x] -> x(0), ARRAY(0, 1, 2, 3))) |
+-----------------------------------------------------+
|                                                   3 |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> select array_count(x -> x > 2, [0, 1, 2, 3]);
+-------------------------------------------------------+
| array_count(array_map([x] -> x(0) > 2, ARRAY(0, 1, 2, 3))) |
+-------------------------------------------------------+
|                                                     1 |
+-------------------------------------------------------+
1 row in set (0.01 sec)

mysql> select array_count(x -> x is null, [null, null, null, 1, 2]);
+-------------------------------------------------------------------------+
| array_count(array_map([x] -> x(0) IS NULL, ARRAY(NULL, NULL, NULL, 1, 2))) |
+-------------------------------------------------------------------------+
|                                                                       3 |
+-------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql> select array_count(x -> power(x,2)>10, [1, 2, 3, 4, 5]);
+-----------------------------------------------------------------------------+
| array_count(array_map([x] -> power(x(0), 2.0) > 10.0, ARRAY(1, 2, 3, 4, 5))) |
```

```
+--------------------------------------------------------------------------------+
|                                                                              2 |
+--------------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql> select *, array_count((x, y) -> x>y, c_array1, c_array2) from array_test;
+------+-----------------+-------------------------+--------------------------------------------------------------------------------+
| id   | c_array1        | c_array2                | array_count(array_map([x, y] -> x(0) > y(1), `c_array1`, `c_array2`)) |
+------+-----------------+-------------------------+--------------------------------------------------------------------------------+
|    1 | [1, 2, 3, 4, 5] | [10, 20, -40, 80, -100] |                                                                              2 |
|    2 | [6, 7, 8]       | [10, 12, 13]            |                                                                              0 |
|    3 | [1]             | [-100]                  |                                                                              1 |
|    4 | [1, NULL, 2]    | [NULL, 3, 1]            |                                                                              1 |
|    5 | []              | []                      |                                                                              0 |
|    6 | NULL            | NULL                    |                                                                              0 |
+------+-----------------+-------------------------+--------------------------------------------------------------------------------+
6 rows in set (0.02 sec)
```

keywords

ARRAY, COUNT, ARRAY_COUNT

ARRAY_APPLY

array_apply

array_apply

description

Filter array to match specific binary condition

Syntax

```
array_apply(arr, op, val)
```

Arguments

`arr` — The array to inspect. If it null, null will be returned. `op` — The compare operation, op includes =, >=, <=, >, <, !=. Support const value only. `val` — The compared value.If it null, null will be returned. Support const value only.

Returned value

The filtered array matched with condition.

Type: Array.

example

```
mysql> select array_apply([1, 2, 3, 4, 5], ">=", 2);
+-------------------------------------------+
| array_apply(ARRAY(1, 2, 3, 4, 5), '>=', 2) |
+-------------------------------------------+
| [2, 3, 4, 5]                              |
+-------------------------------------------+
1 row in set (0.01 sec)

mysql> select array_apply([1000000, 1000001, 1000002], "=", "1000002");
+-----------------------------------------------------------+
| array_apply(ARRAY(1000000, 1000001, 1000002), '=', 1000002) |
+-----------------------------------------------------------+
| [1000002]                                                 |
+-----------------------------------------------------------+
1 row in set (0.01 sec)
```

keywords

ARRAY,APPLY,ARRAY_APPLY

COUNTEQUAL

countequal

countequal

description

Syntax

```
BIGINT countequal(ARRAY<T> arr, T value)
```

Returns a number of the `value` in the given array.

```
num      - how many the value number in array;
0        - if value does not present in the array;
NULL     - when array is NULL.
```

example

```
mysql> select *, countEqual(c_array,5) from array_test;
+------+-----------------+--------------------------+
| id   | c_array         | countequal(`c_array`, 5) |
+------+-----------------+--------------------------+
|    1 | [1, 2, 3, 4, 5] |                        1 |
|    2 | [6, 7, 8]       |                        0 |
|    3 | []              |                        0 |
|    4 | NULL            |                     NULL |
+------+-----------------+--------------------------+

mysql> select *,countEqual(c_array, 1),countEqual(c_array, 5),countEqual(c_array, NULL) from array_test;
+------+----------------------+--------------------------+--------------------------+-----------------------------+
| id   | c_array              | countequal(`c_array`, 1) | countequal(`c_array`, 5) | countequal(`c_array`, NULL) |
+------+----------------------+--------------------------+--------------------------+-----------------------------+
|    1 | [1, 2, 3, 4, 5]      |                        1 |                        1 |                           0 |
|    2 | [6, 7, 8]            |                        0 |                        0 |                           0 |
|    3 | []                   |                        0 |                        0 |                           0 |
|    4 | NULL                 |                     NULL |                     NULL |                        NULL |
|    5 | [66, 77]             |                        0 |                        0 |                           0 |
|    5 | [66, 77]             |                        0 |                        0 |                           0 |
|    6 | NULL                 |                     NULL |                     NULL |                        NULL |
|    7 | [NULL, NULL, NULL]   |                        0 |                        0 |                           3 |
|    8 | [1, 2, 3, 4, 5, 5, 5] |                        1 |                        3 |                           0 |
+------+----------------------+--------------------------+--------------------------+-----------------------------+
```

keywords

ARRAY,COUNTEQUAL,

ELEMENT_AT

element_at

element_at

description

Syntax

```
T element_at(ARRAY<T> arr, BIGINT position)
T arr[position]
```

Returns an element of an array located at the input position. If there is no element at the position, return NULL.

`position` is 1-based and support negative number.

example

positive `position` example:

```
mysql> SELECT id,c_array,element_at(c_array, 5) FROM `array_test`;
+------+-----------------+--------------------------+
| id   | c_array         | element_at(`c_array`, 5) |
+------+-----------------+--------------------------+
|    1 | [1, 2, 3, 4, 5] |                        5 |
|    2 | [6, 7, 8]       |                     NULL |
|    3 | []              |                     NULL |
|    4 | NULL            |                     NULL |
+------+-----------------+--------------------------+
```

negative `position` example:

```
mysql> SELECT id,c_array,c_array[-2] FROM `array_test`;
+------+-----------------+---------------------------------+
| id   | c_array         | %element_extract%(`c_array`, -2) |
+------+-----------------+---------------------------------+
|    1 | [1, 2, 3, 4, 5] |                               4 |
|    2 | [6, 7, 8]       |                               7 |
|    3 | []              |                            NULL |
|    4 | NULL            |                            NULL |
+------+-----------------+---------------------------------+
```

keywords

ELEMENT_AT, SUBSCRIPT

CONVERT_TZ

convert_tz

Description

Syntax

```
DATETIME CONVERT_TZ(DATETIME dt, VARCHAR from_tz, VARCHAR to_tz)
```

Convert datetime value. Go from the given input time zone to the specified time zone and return the result value. If the argument is invalid, the function returns null.

Example

```
mysql> select convert_tz('2019-08-01 13:21:03', 'Asia/Shanghai', 'America/Los_Angeles');
+------------------------------------------------------------------------+
| convert_tz('2019-08-01 13:21:03', 'Asia/Shanghai', 'America/Los_Angeles') |
+------------------------------------------------------------------------+
| 2019-07-31 22:21:03                                                    |
+------------------------------------------------------------------------+

mysql> select convert_tz('2019-08-01 13:21:03', '+08:00', 'America/Los_Angeles');
+---------------------------------------------------------------------+
| convert_tz('2019-08-01 13:21:03', '+08:00', 'America/Los_Angeles') |
+---------------------------------------------------------------------+
| 2019-07-31 22:21:03                                                 |
+---------------------------------------------------------------------+
```

keywords

```
CONVERT_TZ
```

CURDATE,CURRENT_DATE

curdate,current_date

Description

Syntax

```
DATE CURDATE()
```

Get the current date and return it in Date type

example

```
mysql> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2019-12-20 |
+------------+

mysql> SELECT CURDATE() + 0;
+---------------+
| CURDATE() + 0 |
+---------------+
|      20191220 |
+---------------+
```

keywords

```
CURDATE,CURRENT_DATE
```

CURTIME,CURRENT_TIME

curtime,current_time

Description

Syntax

```
TIME CURTIME()
```

Get the current date and return it in Time type

Examples

```
mysql> select current_time();
+----------------+
| current_time() |
+----------------+
| 15:25:47       |
+----------------+
```

keywords

```
CURTIME,CURRENT_TIME
```

CURRENT_TIMESTAMP

Description

The function retrieves the current system time and returns it as a datetime value (`DATETIME`). An optional precision can be specified to adjust the number of digits in the fractional seconds part of the return value.

Syntax

```
CURRENT_TIMESTAMP([<precision>])
```

Parameters

Parameter	Description
<precision>	Optional parameter specifying the precision of the fractional seconds part in the return value. The range is 0 to 6, and the default is 0 (no fractional seconds). Limited by the JDK implementation: if FE is built with JDK8, the precision supports up to milliseconds (3 fractional digits), and higher precision digits will be filled with 0. If higher precision is required, please use JDK11.

Return Value

- Returns the current system time as a DATETIME type.
- If the specified `<precision>` is out of range (e.g., negative or greater than 6), the function will return an error.

Example

```
select CURRENT_TIMESTAMP(),CURRENT_TIMESTAMP(3),CURRENT_TIMESTAMP(6);
```

```
+---------------------+-------------------------+----------------------------+
| now()               | now(3)                  | now(6)                     |
```

```
+---------------------+-------------------------+----------------------------+
| 2025-01-23 11:26:01 | 2025-01-23 11:26:01.771 | 2025-01-23 11:26:01.771000 |
+---------------------+-------------------------+----------------------------+
```

LOCALTIME,LOCALTIMESTAMP

Description

The function retrieves the current system time and returns it as a datetime value (`DATETIME`). An optional precision can be specified to adjust the number of digits in the fractional seconds part of the return value.

Syntax

```
LOCALTIME([<precision>])
LOCALTIMESTAMP([<precision>]))
```

Parameters

Parameter	Description
<precision>	Optional parameter specifying the precision of the fractional seconds part in the return value. The range is 0 to 6, and the default is 0 (no fractional seconds). Limited by the JDK implementation: if FE is built with JDK8, the precision supports up to milliseconds (3 fractional digits), and higher precision digits will be filled with 0. If higher precision is required, please use JDK11.

Return Value

- Returns the current system time as a DATETIME type.
- If the specified `<precision>` is out of range (e.g., negative or greater than 6), the function will return an error.

Example

```
select LOCALTIME(),LOCALTIME(3),LOCALTIME(6);
```

```
+---------------------+-------------------------+----------------------------+
| now()               | now(3)                  | now(6)                     |
+---------------------+-------------------------+----------------------------+
| 2025-01-23 11:50:18 | 2025-01-23 11:50:18.883 | 2025-01-23 11:50:18.883000 |
+---------------------+-------------------------+----------------------------+
```

NOW

Description

The function retrieves the current system time and returns it as a datetime value (`DATETIME`). An optional precision can be specified to adjust the number of digits in the fractional seconds part of the return value.

Syntax

```
NOW([<precision>])
```

Parameters

Parameter	Description
<precision>	Optional parameter specifying the precision of the fractional seconds part in the return value. The range is 0 to 6, and the default is 0 (no fractional seconds). Limited by the JDK implementation: if FE is built with JDK8, the precision supports up to milliseconds (3 fractional digits), and higher precision digits will be filled with 0. If higher precision is required, please use JDK11.

Return Value

- Returns the current system time as a DATETIME type.
- If the specified <precision> is out of range (e.g., negative or greater than 6), the function will return an error.

Example

```
select NOW(), NOW(3), NOW(6);
```

```
+---------------------+-------------------------+----------------------------+
| now()               | now(3)                  | now(6)                     |
```

```
+---------------------+-------------------------+----------------------------+
| 2025-01-23 11:08:35 | 2025-01-23 11:08:35.561 | 2025-01-23 11:08:35.562000 |
+---------------------+-------------------------+----------------------------+
```

YEAR

year

Description

Syntax

```
INT YEAR(DATETIME date)
```

Returns the year part of the date type, ranging from 1000 to 9999

The parameter is Date or Datetime type

example

```
mysql> select year('1987-01-01');
+-----------------------------+
| year('1987-01-01 00:00:00') |
+-----------------------------+
|                        1987 |
+-----------------------------+
```

keywords

```
YEAR
```

QUARTER

Description

The function returns the quarter (1 to 4) of the given date. Each quarter includes three months: - Q1: January to March - Q2: April to June - Q3: July to September - Q4: October to December

Syntax

```
QUARTER(<datetime>)
```

Parameters

Parameter	Description
<datetime>	A valid DATE or DATETIME value to determine the quarter.

Return Value

- Returns an integer representing the quarter of the input date, ranging from 1 to 4.
- If the input is NULL, the function returns NULL.
- If the input is an invalid date (e.g., 0000-00-00), the function returns NULL.

Example

```
SELECT QUARTER('2025-01-16'),QUARTER('2025-01-16 01:11:10');
```

```
+----------------------------------------+-------------------------------------------------+
| quarter(cast('2025-01-16' as DATETIME)) | quarter(cast('2025-01-16 01:11:10' as DATETIME)) |
+----------------------------------------+-------------------------------------------------+
|                                      1 |                                               1 |
+----------------------------------------+-------------------------------------------------+
```

MONTH

month

Description

Syntax

```
INT MONTH (DATETIME date)
```

Returns month information in the time type, ranging from 1,12

The parameter is Date or Datetime type

example

```
mysql> select month('1987-01-01');
+-----------------------------+
| month('1987-01-01 00:00:00') |
+-----------------------------+
|                           1 |
+-----------------------------+
```

keywords

```
MONTH
```

DAY

day

Description

Syntax

```
INT DAY(DATETIME date)
```

Get the day information in the date, and return values range from 1 to 31.

The parameter is Date or Datetime type

example

```
mysql> select day('1987-01-31');
+----------------------------+
| day('1987-01-31 00:00:00') |
+----------------------------+
|                         31 |
+----------------------------+
```

keywords

```
DAY
```

DAYOFYEAR

dayofyear

Description

Syntax

```
INT DAYOFYEAR (DATETIME date)
```

The date of acquisition is the date of the corresponding year.

The parameter is Date or Datetime type

example

```
mysql> select dayofyear('2007-02-03 00:00:00');
+----------------------------------+
| dayofyear('2007-02-03 00:00:00') |
+----------------------------------+
|                               34 |
+----------------------------------+
```

keywords

```
DAYOFYEAR
```

DAYOFMONTH

dayofmonth

Description

Syntax

```
INT DAYOFMONTH (DATETIME date)
```

Get the day information in the date, and return values range from 1 to 31.

The parameter is Date or Datetime type

example

```
mysql> select dayofmonth('1987-01-31');
+-----------------------------------+
| dayofmonth('1987-01-31 00:00:00') |
+-----------------------------------+
|                                31 |
+-----------------------------------+
```

keywords

```
DAYOFMONTH
```

DAYOFWEEK

dayofweek

Description

Syntax

```
INT DAYOFWEEK (DATETIME date)
```

The DAYOFWEEK function returns the index value of the working day of the date, that is, 1 on Sunday, 2 on Monday, and 7 on Saturday.

The parameter is Date or Datetime type

example

```
mysql> select dayofweek('2019-06-25');
+----------------------------------+
| dayofweek('2019-06-25 00:00:00') |
+----------------------------------+
|                                3 |
+----------------------------------+

mysql> select dayofweek(cast(20190625 as date));
+-----------------------------------+
| dayofweek(CAST(20190625 AS DATE)) |
+-----------------------------------+
|                                 3 |
+-----------------------------------+
```

keywords

```
DAYOFWEEK
```

WEEK

week

Description

Syntax

```
INT WEEK(DATE date[, INT mode])
```

Returns the week number for date.The value of the mode argument defaults to 0. The following table describes how the mode argument works.

Mode	First day of week	Range	Week 1 is the first week …
0	Sunday	0-53	with a Sunday in this year
1	Monday	0-53	with 4 or more days this year
2	Sunday	1-53	with a Sunday in this year
3	Monday	1-53	with 4 or more days this year
4	Sunday	0-53	with 4 or more days this year
5	Monday	0-53	with a Monday in this year
6	Sunday	1-53	with 4 or more days this year
7	Monday	1-53	with a Monday in this year

The parameter is Date or Datetime type

example

```
mysql> select week('2020-1-1');
+------------------+
| week('2020-1-1') |
+------------------+
|                0 |
+------------------+
```

```
mysql> select week('2020-7-1',1);
+---------------------+
| week('2020-7-1', 1) |
+---------------------+
|                  27 |
+---------------------+
```

keywords

```
WEEK
```

WEEKDAY

weekday

Description

Syntax

```
INT WEEKDAY (DATETIME date)
```

The WEEKDAY function returns the index value of the working day of the date, that is, 0 on Monday, 1 on Tuesday, and 6 on Sunday.

The parameter is Date or Datetime type

Notice the difference between WEEKDAY and DAYOFWEEK:

```
          +-----+-----+-----+-----+-----+-----+-----+
          | Sun | Mon | Tues| Wed | Thur| Fri | Sat |
          +-----+-----+-----+-----+-----+-----+-----+
  weekday |  6  |  0  |  1  |  2  |  3  |  4  |  5  |
          +-----+-----+-----+-----+-----+-----+-----+
dayofweek |  1  |  2  |  3  |  4  |  5  |  6  |  7  |
          +-----+-----+-----+-----+-----+-----+-----+
```

example

```
mysql> select weekday('2019-06-25');
+---------------------------------+
| weekday('2019-06-25 00:00:00') |
+---------------------------------+
|                               1 |
+---------------------------------+

mysql> select weekday(cast(20190625 as date));
+----------------------------------+
| weekday(CAST(20190625 AS DATE)) |
+----------------------------------+
|                                1 |
+----------------------------------+
```

keywords

```
WEEKDAY
```

WEEKOFYEAR

weekofyear

Description

Syntax

```
INT WEEKOFYEAR (DATETIME DATE)
```

Get the Weeks of the Year

The parameter is Date or Datetime type

example

```
mysql> select weekofyear('2008-02-20 00:00:00');
+----------------------------------+
| weekofyear('2008-02-20 00:00:00') |
+----------------------------------+
|                                8 |
+----------------------------------+
```

keywords

```
WEEKOFYEAR
```

YEARWEEK

yearweek

Description

Syntax

```
INT YEARWEEK(DATE date[, INT mode])
```

Returns year and week for a date.The value of the mode argument defaults to 0. When the week of the date belongs to the previous year, the year and week of the previous year are returned; when the week of the date belongs to the next year, the year of the next year is returned and the week is 1.

The following table describes how the mode argument works.

Mode	First day of week	Range	Week 1 is the first week …
0	Sunday	1-53	with a Sunday in this year
1	Monday	1-53	with 4 or more days this year
2	Sunday	1-53	with a Sunday in this year
3	Monday	1-53	with 4 or more days this year
4	Sunday	1-53	with 4 or more days this year
5	Monday	1-53	with a Monday in this year
6	Sunday	1-53	with 4 or more days this year
7	Monday	1-53	with a Monday in this year

The parameter is Date or Datetime type

example

```
mysql> select yearweek('2021-1-1');
+----------------------+
| yearweek('2021-1-1') |
+----------------------+
|               202052 |
+----------------------+
```

```
mysql> select yearweek('2020-7-1');
+----------------------+
| yearweek('2020-7-1') |
+----------------------+
|               202026 |
+----------------------+
```

```
mysql> select yearweek('2024-12-30',1);
+-----------------------------------+
| yearweek('2024-12-30 00:00:00', 1) |
+-----------------------------------+
|                            202501 |
+-----------------------------------+
```

keywords

```
YEARWEEK
```

DAYNAME

dayname

Description

Syntax

```
VARCHAR DAYNAME (DATE)
```

Date name corresponding to return date

The parameter is Date or Datetime type

example

```
mysql> select dayname('2007-02-03 00:00:00');
+--------------------------------+
| dayname('2007-02-03 00:00:00') |
+--------------------------------+
| Saturday                       |
+--------------------------------+
```

keywords

```
DAYNAME
```

MONTHNAME

monthname

Description

Syntax

```
VARCHAR MONTHNAME (DATE)
```

Month name corresponding to return date

The parameter is Date or Datetime type

example

```
mysql> select monthname('2008-02-03 00:00:00');
+----------------------------------+
| monthname('2008-02-03 00:00:00') |
+----------------------------------+
| February                         |
+----------------------------------+
```

keywords

```
MONTHNAME
```

HOUR

hour

description

Syntax

```
INT HOUR(DATETIME date)
```

Returns hour information in the time type, ranging from 0,23

The parameter is Date or Datetime type

example

```
mysql> select hour('2018-12-31 23:59:59');
+-----------------------------+
| hour('2018-12-31 23:59:59') |
+-----------------------------+
|                          23 |
+-----------------------------+
```

keywords

```
HOUR
```

MINUTE

minute

description

Syntax

```
INT MINUTE(DATETIME date)
```

Returns minute information in the time type, ranging from 0,59

The parameter is Date or Datetime type

example

```
mysql> select minute('2018-12-31 23:59:59');
+-----------------------------+
| minute('2018-12-31 23:59:59') |
+-----------------------------+
|                          59 |
+-----------------------------+
```

keywords

```
MINUTE
```

SECOND

Description

The function returns the second part of the specified datetime value. The range of seconds is 0 to 59.

Syntax

```
SECOND(<datetime>)
```

Parameters

Parameter	Description
<datetime>	The input date or datetime value. Supports DATE or DATETIME types.

Return Value

- Returns an integer representing the second part of the input datetime value, ranging from 0 to 59.
- If the input is NULL, the function returns NULL.
- If the input is an invalid date (e.g., 0000-00-00 00:00:00), the function returns NULL.

Example

```
select second('2018-12-31 23:59:59');
```

```
+-------------------------------------------+
| second(cast('2018-12-30' as DATETIMEV2(0))) |
+-------------------------------------------+
|                                         0 |
+-------------------------------------------+
```

FROM_DAYS

from_days

Description

Syntax

```
DATE FROM_DAYS(INT N)
```

Given a day number, returns a DATE. Note that to maintain consistent behavior with MySQL, the date 0000-02-29 does not exist.

example

```
mysql> select from_days(730669);
+-------------------+
| from_days(730669) |
+-------------------+
| 2000-07-03        |
+-------------------+

mysql> select from_days (5);
+--------------+
| from_days(5) |
+--------------+
| 0000-01-05   |
+--------------+

mysql> select from_days (59);
+---------------+
| from_days(59) |
+---------------+
| 0000-02-28    |
+---------------+

mysql> select from_days (60);
+---------------+
| from_days(60) |
+---------------+
| 0000-03-01    |
+---------------+
```

keywords

```
FROM_DAYS,FROM,DAYS
```

LAST_DAY

last_day

Description

Syntax

```
DATE last_day(DATETIME date)
```

Return the last day of the month, the return day may be : ‘28’(February and not a leap year), ‘29’(February and a leap year), ‘30’(April, June, September, November), ‘31’(January, March, May, July, August, October, December)

example

```
mysql > select last_day('2000-02-03');
+-------------------+
| last_day('2000-02-03 00:00:00') |
+-------------------+
| 2000-02-29        |
+-------------------+
```

keywords

```
LAST_DAY,DAYS
```

TO_MONDAY

to_monday

Description

Syntax

```
DATE to_monday(DATETIME date)
```

Round a date or datetime down to the nearest Monday, return type is Date or DateV2. Specially, input 1970-01-01, 1970-01-02, 1970-01-03 and 1970-01-04 will return '1970-01-01'

example

```
MySQL [(none)]> select to_monday('2022-09-10');
+---------------------------------+
| to_monday('2022-09-10 00:00:00') |
+---------------------------------+
| 2022-09-05                      |
+---------------------------------+
```

keywords

```
MONDAY
```

FROM_SECOND

Description

The function converts a Unix timestamp (in seconds) into a `DATETIME` value.

Syntax

```
FROM_SECOND(<unix_timestamp>)
```

Parameters

Parameter	Description
`<unix_timestamp>`	Required. The Unix timestamp representing the number of seconds elapsed since 1970-01-01 00:00:00 UTC.

Return Value

- Returns a DATETIME value representing the date and time corresponding to the given Unix timestamp.
- If `<unix_timestamp>` is NULL, the function returns NULL.
- If `<unix_timestamp>` is out of valid range, the function returns an error.

Example

```
SELECT FROM_SECOND(1700000000);
```

```
+-------------------------+
| from_second(1700000000) |
+-------------------------+
| 2023-11-15 06:13:20     |
+-------------------------+
```

FROM_UNIXTIME

from_unixtime

description

syntax

```
DATETIME FROM UNIXTIME (BIGINT unix timestamp [, VARCHAR string format]
```

Convert the UNIX timestamp to the corresponding time format of bits, and the format returned is specified by `string_format`

Input is an big integer and return is a string type

Support `date_format`'s format, and default is `%Y-%m-%d %H:%i:%s`

The current supported range for `unix_timestamp` is `[0, 32536771199]`. `unix_timestamp` values that fall outside of this range will be returned as NULL

example

```
mysql> select from_unixtime(1196440219);
+---------------------------+
| from_unixtime(1196440219) |
+---------------------------+
| 2007-12-01 00:30:19       |
+---------------------------+

mysql> select from_unixtime(1196440219, '%Y-%m-%d');
+----------------------------------------+
| from_unixtime(1196440219, '%Y-%m-%d')   |
+----------------------------------------+
| 2007-12-01                             |
+----------------------------------------+

mysql> select from_unixtime(1196440219, '%Y-%m-%d %H:%i:%s');
+-------------------------------------------------+
|From unixtime (1196440219,'%Y-%m-%d %H:%i:%s')    |
+-------------------------------------------------+
| 2007-12-01 00:30:19                             |
+-------------------------------------------------+
```

For timestamps that exceed the range, you can use the “from_second” function. `DATETIME FROM_SECOND(BIGINT unix_timestamp)`

```
mysql> select from_second(21474836470);
+--------------------------+
| from_second(21474836470) |
+--------------------------+
```

```
| 2650-07-06 16:21:10      |
+--------------------------+
```

keywords

```
FROM_UNIXTIME,FROM,UNIXTIME
```

UNIX_TIMESTAMP

unix_timestamp

Description

Syntax

```
INT UNIX_TIMESTAMP([DATETIME date[, STRING fmt]])
```

Converting a Date or Datetime type to a UNIX timestamp.

If there are no parameters, the current time is converted into a timestamp.

The parameter needs to be Date or Datetime type.

Any date before 1970-01-01 00:00:00 or after 2038-01-19 03:14:07 will return 0.

See `date_format` function to get Format explanation.

This function is affected by time zone.

example

```
mysql> select unix_timestamp();
+------------------+
| unix_timestamp() |
+------------------+
|       1558589570 |
+------------------+

mysql> select unix_timestamp('2007-11-30 10:30:19');
+---------------------------------------+
| unix_timestamp('2007-11-30 10:30:19') |
+---------------------------------------+
|                            1196389819 |
+---------------------------------------+

mysql> select unix_timestamp('2007-11-30 10:30-19', '%Y-%m-%d %H:%i-%s');
+---------------------------------------+
| unix_timestamp('2007-11-30 10:30-19') |
+---------------------------------------+
|                            1196389819 |
+---------------------------------------+

mysql> select unix_timestamp('2007-11-30 10:30%3A19', '%Y-%m-%d %H:%i%%3A%s');
+---------------------------------------+
| unix_timestamp('2007-11-30 10:30%3A19') |
+---------------------------------------+
|                            1196389819 |
+---------------------------------------+

mysql> select unix_timestamp('1969-01-01 00:00:00');
+---------------------------------------+
```

```
| unix_timestamp('1969-01-01 00:00:00') |
+---------------------------------------+
|                                     0 |
+---------------------------------------+
```

keywords

```
UNIX_TIMESTAMP,UNIX,TIMESTAMP
```

UTC_TIMESTAMP

utc_timestamp

Description

Syntax

```
DATETIME UTC_TIMESTAMP()
```

Returns the current UTC date and time in “YYYY-MM-DD HH: MM: SS” or

A Value of “YYYYMMDDHMMSS” Format

Depending on whether the function is used in a string or numeric context

example

```
mysql> select utc_timestamp(),utc_timestamp() + 1;
+---------------------+---------------------+
| utc_timestamp()     | utc_timestamp() + 1 |
+---------------------+---------------------+
| 2019-07-10 12:31:18 |      20190710123119 |
+---------------------+---------------------+
```

keywords

```
UTC_TIMESTAMP,UTC,TIMESTAMP
```

TO_DATE

to_date

description

Syntax

```
DATE TO_DATE(DATETIME)
```

Return the DATE part of DATETIME value.

example

```
mysql> select to_date("2020-02-02 00:00:00");
+---------------------------------+
| to_date('2020-02-02 00:00:00') |
+---------------------------------+
| 2020-02-02                      |
+---------------------------------+
```

keywords

```
TO_DATE
```

TO_DAYS

to_days

Description

Syntax

```
INT TO DAYS
```

Days of returning date distance 0000-01-01

The parameter is Date or Datetime type

example

```
mysql> select to_days('2007-10-07');
+-----------------------+
| to_days('2007-10-07') |
+-----------------------+
|                733321 |
+-----------------------+
```

keywords

```
TO_DAYS,TO,DAYS
```

TIME_TO_SEC

Description

The function converts an input `TIME` or `DATETIME` value into the total time in seconds. If the input is of `DATETIME` type, the function automatically extracts the time part (`HH:MM:SS`).

Syntax

```
TIME_TO_SEC(<time>)
```

Parameters

Parameter	Description
`<time>`	Required. Supports TIME or DATETIME values. If the input is DATETIME, the function extracts the time part for calculation.

Return Value

- Returns an integer representing the total seconds of the input time value.
- If `<time>` is NULL, the function returns NULL.

Example

```
SELECT TIME_TO_SEC('16:32:18'),TIME_TO_SEC('2025-01-01 16:32:18');
```

```
+--------------------------------------+-------------------------------------------------+
| time_to_sec(cast('16:32:18' as TIME)) | time_to_sec(cast('2025-01-01 16:32:18' as TIME)) |
+--------------------------------------+-------------------------------------------------+
|                                59538 |                                           59538 |
+--------------------------------------+-------------------------------------------------+
```

SEC_TO_TIME

Description

The `SEC_TO_TIME` function converts a value in seconds into a `TIME` type, returning the result in the format `HH:MM:SS`. The input seconds represent the time elapsed since the start of a day (`00:00:00`).

Syntax

```
SEC_TO_TIME(<seconds>)
```

Parameters

Parameter	Description
`<seconds>`	Required. The input number of seconds, representing the time elapsed since the start of a day (00:00:00). Supports positive or negative integers.

Return Value

- Returns a TIME value in the format `HH:MM:SS`, representing the time calculated from the start of a day (00:00:00).
- If `<seconds>` is NULL, the function returns NULL.

Example

```
SELECT SEC_TO_TIME(59738);
```

```
+--------------------+
| sec_to_time(59738) |
+--------------------+
| 16:35:38           |
+--------------------+
```

EXTRACT

extract

description

Syntax

```
INT extract(unit FROM DATETIME)
```

Extract DATETIME The value of a specified unit. The unit can be year, day, hour, minute, second or microsecond

Example

```
mysql> select extract(year from '2022-09-22 17:01:30') as year,
    -> extract(month from '2022-09-22 17:01:30') as month,
    -> extract(day from '2022-09-22 17:01:30') as day,
    -> extract(hour from '2022-09-22 17:01:30') as hour,
    -> extract(minute from '2022-09-22 17:01:30') as minute,
    -> extract(second from '2022-09-22 17:01:30') as second,
    -> extract(microsecond from cast('2022-09-22 17:01:30.000123' as datetimev2(6))) as microsecond;
+------+-------+------+------+--------+--------+-------------+
| year | month | day  | hour | minute | second | microsecond |
+------+-------+------+------+--------+--------+-------------+
| 2022 |     9 |   22 |   17 |      1 |     30 |         123 |
+------+-------+------+------+--------+--------+-------------+
```

keywords

```
extract
```

MAKEDATE

makedate

Description

Syntax

```
DATE MAKEDATE(INT year, INT dayofyear)
```

Returns a date, given year and day-of-year values. dayofyear must be greater than 0 or the result is NULL.

example

```
mysql> select makedate(2021,1), makedate(2021,100), makedate(2021,400);
+------------------+--------------------+--------------------+
| makedate(2021, 1) | makedate(2021, 100) | makedate(2021, 400) |
+------------------+--------------------+--------------------+
| 2021-01-01        | 2021-04-10          | 2022-02-04          |
+------------------+--------------------+--------------------+
```

keywords

```
MAKEDATE
```

STR_TO_DATE

Description

The function converts the input datetime string into a DATETIME value based on the specified format.

Syntax

```
STR_TO_DATE(<datetime_str>, <format>)
```

Parameters

Parameter	Description
<datetime_str>	Required. The input datetime string to be converted.
<format>	Required. The specified datetime format string, such as %Y-%m-%d %H:%i:%s, etc. for specific format parameters, see the DATE_FORMAT documentation.

In addition, the <format> supports the following several alternative formats and interprets them to the regular format:

Alternative Input	Interpret as
yyyyMMdd	%Y%m%d
yyyy-MM-dd	%Y-%m-%d
yyyy-MM-dd HH:mm:ss	%Y-%m-%d %H:%i:%s

Return Value

- Returns a DATETIME value representing the converted datetime.
- If the input <datetime_str> or <format> is invalid, the function returns NULL.

Example

Convert common datetime strings to DATETIME

```
SELECT STR_TO_DATE('2025-01-23 12:34:56', '%Y-%m-%d %H:%i:%s'),STR_TO_DATE('2025-01-23 12:34:56', 'yyyy-MM-dd HH:mm:ss');
```

```
+-----------------------------------------------------+-----------------------------------------------------------+
| str_to_date('2025-01-23 12:34:56', '%Y-%m-%d %H:%i:%s') | str_to_date('2025-01-23 12:34:56', 'yyyy-MM-dd HH:mm:ss') |
+-----------------------------------------------------+-----------------------------------------------------------+
| 2025-01-23 12:34:56.000000                          | 2025-01-23 12:34:56.000000                                |
+-----------------------------------------------------+-----------------------------------------------------------+
```

Others

```
select STR_TO_DATE('200442 Monday', '%X%V %W'),STR_TO_DATE('2023','%Y');
```

```
+----------------------------------------+--------------------------+
| str_to_date('200442 Monday', '%X%V %W') | str_to_date('2023', '%Y') |
+----------------------------------------+--------------------------+
| 2004-10-18                             | 2023-01-01               |
+----------------------------------------+--------------------------+
```

TIME_ROUND

time_round

description

Syntax

```
DATETIME TIME_ROUND(DATETIME expr)
DATETIME TIME_ROUND(DATETIME expr, INT period)
DATETIME TIME_ROUND(DATETIME expr, DATETIME origin)
DATETIME TIME_ROUND(DATETIME expr, INT period, DATETIME origin)
```

The function name `TIME_ROUND` consists of two parts, Each part consists of the following optional values. - `TIME`: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR` - `ROUND`: `FLOOR`, `CEIL`

Returns the upper/lower bound of `expr`.

- `period` specifies how many `TIME` units, the default is `1`.
- `origin` specifies the start time of the period, the default is `1970-01-01T00:00:00`, the start time of `WEEK` is Sunday, which is `1970-01-04T00:00:00`. Could be larger than `expr`.
- Please try to choose common `period`, such as 3 `MONTH`, 90 `MINUTE`. If you set a uncommon `period`, please also specify `origin`.

example

```
MySQL> SELECT YEAR_FLOOR('20200202000000');
+-----------------------------+
| year_floor('20200202000000') |
+-----------------------------+
| 2020-01-01 00:00:00          |
+-----------------------------+


MySQL> SELECT MONTH_CEIL(CAST('2020-02-02 13:09:20' AS DATETIME), 3); --quarter
+-------------------------------------------------------+
| month_ceil(CAST('2020-02-02 13:09:20' AS DATETIME), 3) |
+-------------------------------------------------------+
| 2020-04-01 00:00:00                                    |
+-------------------------------------------------------+


MySQL> SELECT WEEK_CEIL('2020-02-02 13:09:20', '2020-01-06'); --monday
+-------------------------------------------------------+
| week_ceil('2020-02-02 13:09:20', '2020-01-06 00:00:00') |
+-------------------------------------------------------+
| 2020-02-03 00:00:00                                    |
+-------------------------------------------------------+


MySQL> SELECT MONTH_CEIL(CAST('2020-02-02 13:09:20' AS DATETIME), 3, CAST('1970-01-09 00:00:00' AS DATETIME)); --next rent day
```

```
+-----------------------------------------------------------------------------------------------+
| month_ceil(CAST('2020-02-02 13:09:20' AS DATETIME), 3, CAST('1970-01-09 00:00:00' AS DATETIME)) |
+-----------------------------------------------------------------------------------------------+
| 2020-04-09 00:00:00                                                                           |
+-----------------------------------------------------------------------------------------------+
```

keywords

```
TIME_ROUND
```

TIMEDIFF

Description

The TIMEDIFF function calculates the difference between two datetime values. This function accepts two arguments and returns the difference as a TIME type.

Syntax

```
TIMEDIFF(<end_datetime>, <start_datetime>)
```

Parameters

Parameter	Description
<end_datetime>	The ending datetime value.
<start_datetime>	The starting datetime value.

Return Value

Returns a TIME type value representing the difference between the two inputs: - If <end_datetime> is later than <start_datetime>, it returns a positive time difference. - If <end_datetime> is earlier than <start_datetime>, it returns a negative time difference. - If <end_datetime> and <start_datetime> are equal, it returns 00:00:00. - If <end_datetime> or <start_datetime> is NULL, the function returns NULL. - If <end_datetime> or <start_datetime> is an invalid datetime value (e.g., 0000-00-00 00:00:00), the function returns NULL.

Example

```
SELECT TIMEDIFF('2024-07-20 16:59:30','2024-07-11 16:35:21');
```

```
+-----------------------------------------------------------------------------------------------------+
| timediff(cast('2024-07-20 16:59:30' as DATETIMEV2(0)), cast('2024-07-11 16:35:21' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------------------+
| 216:24:09                                                                                           |
+-----------------------------------------------------------------------------------------------------+
```

TIMESTAMPADD

timestampadd

description

Syntax

```
DATETIME TIMESTAMPADD(unit, interval, DATETIME datetime_expr)
```

Adds the integer expression interval to the date or datetime expression datetime_expr.

The unit for interval is given by the unit argument, which should be one of the following values:

SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, or YEAR.

example

```
mysql> SELECT TIMESTAMPADD(MINUTE,1,'2019-01-02');
+-----------------------------------------------+
| timestampadd(MINUTE, 1, '2019-01-02 00:00:00') |
+-----------------------------------------------+
| 2019-01-02 00:01:00                           |
+-----------------------------------------------+

mysql> SELECT TIMESTAMPADD(WEEK,1,'2019-01-02');
+-----------------------------------------------+
| timestampadd(WEEK, 1, '2019-01-02 00:00:00') |
+-----------------------------------------------+
| 2019-01-09 00:00:00                           |
+-----------------------------------------------+
```

keywords

```
TIMESTAMPADD
```

TIMESTAMPDIFF

timestampdiff

description

Syntax

```
INT TIMESTAMPDIFF(unit, DATETIME datetime_expr1, DATETIME datetime_expr2)
```

Returns datetime_expr2 − datetime_expr1, where datetime_expr1 and datetime_expr2 are date or datetime expressions.

The unit for the result (an integer) is given by the unit argument.

The legal values for unit are the same as those listed in the description of the TIMESTAMPADD() function.

example

```
MySQL> SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');
+-------------------------------------------------------------------+
| timestampdiff(MONTH, '2003-02-01 00:00:00', '2003-05-01 00:00:00') |
+-------------------------------------------------------------------+
|                                                                 3 |
+-------------------------------------------------------------------+

MySQL> SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');
+-------------------------------------------------------------------+
| timestampdiff(YEAR, '2002-05-01 00:00:00', '2001-01-01 00:00:00') |
+-------------------------------------------------------------------+
|                                                                -1 |
+-------------------------------------------------------------------+


MySQL> SELECT TIMESTAMPDIFF(MINUTE,'2003-02-01','2003-05-01 12:05:55');
+---------------------------------------------------------------------+
| timestampdiff(MINUTE, '2003-02-01 00:00:00', '2003-05-01 12:05:55') |
+---------------------------------------------------------------------+
|                                                              128885 |
+---------------------------------------------------------------------+
```

keywords

```
TIMESTAMPDIFF
```

DATE_ADD

date_add

Description

Syntax

```
DATETIME DATE_ADD(DATETIME date, INTERVAL expr type)
```

Adds a specified time interval to the date.

The date parameter is a valid date expression.

The expr parameter is the interval you want to add.

The type parameter can be the following values: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

example

```
mysql> select date_add('2010-11-30 23:59:59', INTERVAL 2 DAY);
+-------------------------------------------------+
| date_add('2010-11-30 23:59:59', INTERVAL 2 DAY) |
+-------------------------------------------------+
| 2010-12-02 23:59:59                             |
+-------------------------------------------------+
```

keywords

```
DATE_ADD,DATE,ADD
```

DATE_SUB

date_sub

Description

Syntax

```
DATETIME DATE_SUB(DATETIME date, INTERVAL expr type)
```

Subtract the specified time interval from the date

The date parameter is a valid date expression.

The expr parameter is the interval you want to add.

The type parameter can be the following values: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

example

```
mysql> select date_sub('2010-11-30 23:59:59', INTERVAL 2 DAY);
+------------------------------------------------+
| date_sub('2010-11-30 23:59:59', INTERVAL 2 DAY) |
+------------------------------------------------+
| 2010-11-28 23:59:59                            |
+------------------------------------------------+
```

keywords

```
Date, date, date
```

DATE_TRUNC

date_trunc

date_trunc

Description

Syntax

```
DATETIME DATE_TRUNC(DATETIME datetime, VARCHAR unit)
```

Truncates datetime in the specified time unit.

datetime is a legal date expression.

unit is the time unit you want to truncate. The optional values are as follows: [`second`,`minute`,`hour`,`day`,`week`,`month`,`quarter`,`year`].

example

```
mysql> select date_trunc('2010-12-02 19:28:30', 'second');
+-------------------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'second')     |
+-------------------------------------------------+
| 2010-12-02 19:28:30                             |
+-------------------------------------------------+

mysql> select date_trunc('2010-12-02 19:28:30', 'minute');
+-------------------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'minute')     |
+-------------------------------------------------+
| 2010-12-02 19:28:00                             |
+-------------------------------------------------+

mysql> select date_trunc('2010-12-02 19:28:30', 'hour');
+-------------------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'hour')       |
+-------------------------------------------------+
| 2010-12-02 19:00:00                             |
+-------------------------------------------------+

mysql> select date_trunc('2010-12-02 19:28:30', 'day');
+-------------------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'day')        |
+-------------------------------------------------+
| 2010-12-02 00:00:00                             |
+-------------------------------------------------+

mysql> select date_trunc('2010-12-02 19:28:30', 'week');
+-----------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'week') |
+-----------------------------------------+
| 2010-11-29 00:00:00                     |
```

```
+---------------------------------------------+

mysql> select date_trunc('2010-12-02 19:28:30', 'month');
+-------------------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'month')      |
+-------------------------------------------------+
| 2010-12-01 00:00:00                             |
+-------------------------------------------------+

mysql> select date_trunc('2010-12-02 19:28:30', 'quarter');
+-------------------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'quarter')    |
+-------------------------------------------------+
| 2010-10-01 00:00:00                             |
+-------------------------------------------------+

mysql> select date_trunc('2010-12-02 19:28:30', 'year');
+-------------------------------------------------+
| date_trunc('2010-12-02 19:28:30', 'year')       |
+-------------------------------------------------+
| 2010-01-01 00:00:00                             |
+-------------------------------------------------+
```

keywords

DATE_TRUNC,DATE,DATETIME

DATE_FORMAT

date_format

Description

Syntax

```
VARCHAR DATE_FORMAT (DATETIME DATE, VARCHAR Format)
```

Convert the date type to a string according to the format type. Convert the date type to a string according to the format type. Currently, it supports a maximum of 128 bytes for the string. If the return value length exceeds 128 bytes, then it returns NULL.

The date parameter is the valid date. Format specifies the date/time output format.

The formats available are:

Format	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, …)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM)
%S	Seconds (00..59)

Format	Description
%s	Seconds (00..59)
%T	Time, 24-hour (hh:mm:ss)
%U	Week (00..53), where Sunday is the first day of the week; WEEK mode 0
%u	Week (00..53), where Monday is the first day of the week; WEEK mode 1
%V	Week (01..53), where Sunday is the first day of the week; WEEK mode 2; used with %X
%v	Week (01..53), where Monday is the first day of the week; WEEK mode 3; used with %x
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)
%X	Year for the week where Sunday is the first day of the week, numeric, four digits; used with %V
%x	Year for the week, where Monday is the first day of the week, numeric, four digits; used with %v
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal % character
%**x**	**x**, for any “**x**” not listed above

Also support 3 formats:

yyyyMMdd

yyyy-MM-dd

yyyy-MM-dd HH:mm:ss

example

```
mysql> select date_format('2009-10-04 22:23:00', '%W %M %Y');
+------------------------------------------------+
| date_format('2009-10-04 22:23:00', '%W %M %Y') |
+------------------------------------------------+
| Sunday October 2009                            |
+------------------------------------------------+

mysql> select date_format('2007-10-04 22:23:00', '%H:%i:%s');
+------------------------------------------------+
| date_format('2007-10-04 22:23:00', '%H:%i:%s') |
+------------------------------------------------+
| 22:23:00                                       |
+------------------------------------------------+

mysql> select date_format('1900-10-04 22:23:00', '%D %y %a %d %m %b %j');
+------------------------------------------------------------+
| date_format('1900-10-04 22:23:00', '%D %y %a %d %m %b %j') |
+------------------------------------------------------------+
| 4th 00 Thu 04 10 Oct 277                                   |
+------------------------------------------------------------+

mysql> select date_format('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');
+------------------------------------------------------------+
```

```
| date_format('1997-10-04 22:23:00', '%H %k %I %r %T %S %w') |
+------------------------------------------------------------+
| 22 22 10 10:23:00 PM 22:23:00 00 6                         |
+------------------------------------------------------------+

mysql> select date_format('1999-01-01 00:00:00', '%X %V');
+---------------------------------------------+
| date_format('1999-01-01 00:00:00', '%X %V') |
+---------------------------------------------+
| 1998 52                                     |
+---------------------------------------------+

mysql> select date_format('2006-06-01', '%d');
+----------------------------------------+
| date_format('2006-06-01 00:00:00', '%d') |
+----------------------------------------+
| 01                                     |
+----------------------------------------+

mysql> select date_format('2006-06-01', '%%d');
+-----------------------------------------+
| date_format('2006-06-01 00:00:00', '%%d') |
+-----------------------------------------+
| %01                                     |
+-----------------------------------------+
```

keywords

```
DATE_FORMAT,DATE,FORMAT
```

DATEDIFF

datediff

Description

Syntax

```
INT DATEDIFF (DATETIME expr1, DATETIME expr2)
```

Calculate expr1 - expr2, the result is accurate to the unit of a day.

Expr1 and expr2 parameters are valid date or date/time expressions.

Note: Only the date part of the value participates in the calculation.

example

```
mysql> select datediff(CAST('2007-12-31 23:59:59' AS DATETIME), CAST('2007-12-30' AS DATETIME));
+----------------------------------------------------------------------------------+
| datediff(CAST('2007-12-31 23:59:59' AS DATETIME), CAST('2007-12-30' AS DATETIME)) |
+----------------------------------------------------------------------------------+
|                                                                                1 |
+----------------------------------------------------------------------------------+

mysql> select datediff(CAST('2010-11-30 23:59:59' AS DATETIME), CAST('2010-12-31' AS DATETIME));
+----------------------------------------------------------------------------------+
| datediff(CAST('2010-11-30 23:59:59' AS DATETIME), CAST('2010-12-31' AS DATETIME)) |
+----------------------------------------------------------------------------------+
|                                                                              -31 |
+----------------------------------------------------------------------------------+
```

keywords

```
DATEDIFF
```

SECOND_FLOOR

Description

The function aligns the input datetime value upwards to the nearest second boundary based on the specified period and returns the aligned datetime value.

Syntax

```
SECOND_FLOOR(<datetime>[, <period>][, <origin_datetime>])
```

Parameters

Parameter	Description
<datetime>	Required. The input datetime value. Supports the DATETIME type.
<period>	Optional. Specifies the number of seconds in each period. Supports positive integers (INT). Defaults to 1 second.
<origin_datetime>	Optional. The starting point for alignment. Supports the DATETIME type. Defaults to 0001-01-01T00:00:00 if not specified.

Return Value

- Returns a datetime value representing the input datetime aligned upwards to the nearest specified second boundary.
- If <datetime> is NULL, the function returns NULL.
- If <datetime> is an invalid date (e.g., 0000-00-00T00:00:00), the function returns NULL.

Example

Only specifying <datetime>

```
SELECT SECOND_FLOOR('2025-01-23 12:34:56');
```

```
+-----------------------------------------------------------+
| second_floor(cast('2025-01-23 12:34:56' as DATETIMEV2(0))) |
+-----------------------------------------------------------+
| 2025-01-23 12:34:56                                       |
+-----------------------------------------------------------+
```

Specifying <datetime> and <origin_datetime>

```
SELECT SECOND_FLOOR('2025-01-23 12:34:56', '2025-01-01 00:00:00');
```

```
+-----------------------------------------------------------------------------------------------------------+
| second_floor(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), cast('2025-01-01 00:00:00' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------------------------+
| 2025-01-23 12:34:56                                                                                       |
+-----------------------------------------------------------------------------------------------------------+
```

Specifying <datetime> and <period>

```
SELECT SECOND_FLOOR('2025-01-23 12:34:56', 5)
```

```
+-------------------------------------------------------------+
| second_floor(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), 5) |
+-------------------------------------------------------------+
| 2025-01-23 12:34:55                                         |
+-------------------------------------------------------------+
```

Specifying <datetime>, <period> and <origin_datetime>

```
SELECT SECOND_FLOOR('2025-01-23 12:34:56', 10, '2025-01-23 12:00:00');
```

```
+-----------------------------------------------------------------------------------------------------------------+
| second_floor(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), 10, cast('2025-01-23 12:00:00' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------------------------------+
| 2025-01-23 12:34:50                                                                                             |
+-----------------------------------------------------------------------------------------------------------------+
```

MINUTE_FLOOR

minute_floor

description

Syntax

```
DATETIME MINUTE_FLOOR(DATETIME datetime)
DATETIME MINUTE_FLOOR(DATETIME datetime, DATETIME origin)
DATETIME MINUTE_FLOOR(DATETIME datetime, INT period)
DATETIME MINUTE_FLOOR(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding down time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many minutes each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select minute_floor("2023-07-13 22:28:18", 5);
+------------------------------------------------------------+
| minute_floor(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+------------------------------------------------------------+
| 2023-07-13 22:25:00                                        |
+------------------------------------------------------------+
1 row in set (0.06 sec)
```

keywords

```
MINUTE_FLOOR, MINUTE, FLOOR
```

Best Practice

See also date_floor

HOUR_FLOOR

hour_floor

description

Syntax

```
DATETIME HOUR_FLOOR(DATETIME datetime)
DATETIME HOUR_FLOOR(DATETIME datetime, DATETIME origin)
DATETIME HOUR_FLOOR(DATETIME datetime, INT period)
DATETIME HOUR_FLOOR(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding down time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many hours each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select hour_floor("2023-07-13 22:28:18", 5);
+------------------------------------------------------------+
| hour_floor(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+------------------------------------------------------------+
| 2023-07-13 21:00:00                                        |
+------------------------------------------------------------+
1 row in set (0.23 sec)
```

keywords

```
HOUR_FLOOR, HOUR, FLOOR
```

Best Practice

See also date_floor

DAY_FLOOR

day_floor

Description

Syntax

```
DATETIME DAY_FLOOR(DATETIME datetime)
DATETIME DAY_FLOOR(DATETIME datetime, DATETIME origin)
DATETIME DAY_FLOOR(DATETIME datetime, INT period)
DATETIME DAY_FLOOR(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding down time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many days each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

Example

```
mysql> select day_floor("2023-07-13 22:28:18", 5);
+-----------------------------------------------------------+
| day_floor(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+-----------------------------------------------------------+
| 2023-07-12 00:00:00                                       |
+-----------------------------------------------------------+
1 row in set (0.07 sec)
```

Keywords

```
DAY_FLOOR,DAY,FLOOR
```

Best Practices

See also date_floor

MONTH_FLOOR

month_floor

description

Syntax

```
DATETIME MONTH_FLOOR(DATETIME datetime)
DATETIME MONTH_FLOOR(DATETIME datetime, DATETIME origin)
DATETIME MONTH_FLOOR(DATETIME datetime, INT period)
DATETIME MONTH_FLOOR(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding down time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many months each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select month_floor("2023-07-13 22:28:18", 5);
+------------------------------------------------------------+
| month_floor(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+------------------------------------------------------------+
| 2023-05-01 00:00:00                                        |
+------------------------------------------------------------+
1 row in set (0.12 sec)
```

keywords

```
MONTH_FLOOR, MONTH, FLOOR
```

Best Practice

See also date_floor

YEAR_FLOOR

year_floor

description

Syntax

```
DATETIME YEAR_FLOOR(DATETIME datetime)
DATETIME YEAR_FLOOR(DATETIME datetime, DATETIME origin)
DATETIME YEAR_FLOOR(DATETIME datetime, INT period)
DATETIME YEAR_FLOOR(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding down time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many years each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select year_floor("2023-07-13 22:28:18", 5);
+-------------------------------------------------------------+
| year_floor(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+-------------------------------------------------------------+
| 2020-01-01 00:00:00                                         |
+-------------------------------------------------------------+
1 row in set (0.11 sec)
```

keywords

```
YEAR_FLOOR, YEAR, FLOOR
```

Best Practice

See also date_floor

DATE_CEIL

date_ceil

description

Syntax

```
DATETIME DATE_CEIL(DATETIME datetime, INTERVAL period type)
```

Convert the date to the nearest rounding up time of the specified time interval period.

The datetime parameter is a valid date expression.

The period parameter specifies how many units each cycle consists of, starting from 0001-01-01T00:00:00

type： YEAR, MONTH, DAY, HOUR, MINUTE, SECOND.

example

```
mysql [(none)]>select date_ceil("2023-07-13 22:28:18",interval 5 second);
+-------------------------------------------------------------+
| second_ceil('2023-07-13 22:28:18', 5, '0001-01-01 00:00:00') |
+-------------------------------------------------------------+
| 2023-07-13 22:28:20                                         |
+-------------------------------------------------------------+
1 row in set (0.01 sec)

mysql [(none)]>select date_ceil("2023-07-13 22:28:18",interval 5 minute);
+-------------------------------------------------------------+
| minute_ceil('2023-07-13 22:28:18', 5, '0001-01-01 00:00:00') |
+-------------------------------------------------------------+
| 2023-07-13 22:30:00                                         |
+-------------------------------------------------------------+
1 row in set (0.01 sec)

mysql [(none)]>select date_ceil("2023-07-13 22:28:18",interval 5 hour);
+-----------------------------------------------------------+
| hour_ceil('2023-07-13 22:28:18', 5, '0001-01-01 00:00:00') |
+-----------------------------------------------------------+
| 2023-07-13 23:00:00                                       |
+-----------------------------------------------------------+
1 row in set (0.01 sec)

mysql [(none)]>select date_ceil("2023-07-13 22:28:18",interval 5 day);
+----------------------------------------------------------+
| day_ceil('2023-07-13 22:28:18', 5, '0001-01-01 00:00:00') |
+----------------------------------------------------------+
| 2023-07-15 00:00:00                                      |
+----------------------------------------------------------+
1 row in set (0.00 sec)

mysql [(none)]>select date_ceil("2023-07-13 22:28:18",interval 5 month);
+------------------------------------------------------------+
| month_ceil('2023-07-13 22:28:18', 5, '0001-01-01 00:00:00') |
```

```
+------------------------------------------------------------+
| 2023-12-01 00:00:00                                        |
+------------------------------------------------------------+
1 row in set (0.01 sec)

mysql [(none)]>select date_ceil("2023-07-13 22:28:18",interval 5 year);
+------------------------------------------------------------+
| year_ceil('2023-07-13 22:28:18', 5, '0001-01-01 00:00:00') |
+------------------------------------------------------------+
| 2026-01-01 00:00:00                                        |
+------------------------------------------------------------+
1 row in set (0.00 sec)
```

keywords

```
DATE_CEIL,DATE,CEIL
```

Best Practice

See also - second_ceil - minute_ceil - hour_ceil - day_ceil - month_ceil - year_ceil

SECOND_CEIL

Description

The function aligns the input datetime value upwards to the nearest second boundary based on the specified period and returns the aligned datetime value.

Syntax

```
SECOND_CEIL(<datetime>[, <period>][, <origin_datetime>])
```

Parameters

Parameter	Description
<datetime>	Required. The input datetime value. Supports the DATETIME type.
<period>	Optional. Specifies the number of seconds in each period. Supports positive integers (INT). Defaults to 1 second.
<origin_datetime>	Optional. The starting point for alignment. Supports the DATETIME type. Defaults to 0001-01-01T00:00:00 if not specified.

Return Value

- Returns a datetime value representing the input datetime aligned upwards to the nearest specified second boundary.
- If <datetime> is NULL, the function returns NULL.
- If <datetime> is an invalid date (e.g., 0000-00-00T00:00:00), the function returns NULL.

Example

Only specifying <datetime>

```
SELECT SECOND_CEIL('2025-01-23 12:34:56');
```

```
+-----------------------------------------------------------+
| second_ceil(cast('2025-01-23 12:34:56' as DATETIMEV2(0))) |
+-----------------------------------------------------------+
| 2025-01-23 12:34:56                                       |
+-----------------------------------------------------------+
```

Specifying <datetime> and <origin_datetime>

```
SELECT SECOND_CEIL('2025-01-23 12:34:56', '2025-01-01 00:00:00');
```

```
+-----------------------------------------------------------------------------------------------------------+
| second_ceil(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), cast('2025-01-01 00:00:00' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------------------------+
| 2025-01-23 12:34:56                                                                                       |
+-----------------------------------------------------------------------------------------------------------+
```

Specifying <datetime> and <period>

```
SELECT SECOND_CEIL('2025-01-23 12:34:56', 5)
```

```
+-----------------------------------------------------------+
| second_ceil(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), 5) |
+-----------------------------------------------------------+
| 2025-01-23 12:35:00                                       |
+-----------------------------------------------------------+
```

Specifying <datetime>, <period> and <origin_datetime>

```
SELECT SECOND_CEIL('2025-01-23 12:34:56', 10, '2025-01-23 12:00:00');
```

```
+-----------------------------------------------------------------------------------------------------------------+
| second_ceil(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), 10, cast('2025-01-23 12:00:00' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------------------------------+
| 2025-01-23 12:35:00                                                                                             |
+-----------------------------------------------------------------------------------------------------------------+
```

MINUTE_CEIL

minute_ceil

description

Syntax

```
DATETIME MINUTE_CEIL(DATETIME datetime)
DATETIME MINUTE_CEIL(DATETIME datetime, DATETIME origin)
DATETIME MINUTE_CEIL(DATETIME datetime, INT period)
DATETIME MINUTE_CEIL(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding up time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many minutes each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select minute_ceil("2023-07-13 22:28:18", 5);
+-----------------------------------------------------------+
| minute_ceil(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+-----------------------------------------------------------+
| 2023-07-13 22:30:00                                       |
+-----------------------------------------------------------+
1 row in set (0.21 sec)
```

keywords

```
MINUTE_CEIL, MINUTE, CEIL
```

Best Practice

See also date_ceil

HOUR_CEIL

hour_ceil

description

Syntax

```
DATETIME HOUR_CEIL(DATETIME datetime)
DATETIME HOUR_CEIL(DATETIME datetime, DATETIME origin)
DATETIME HOUR_CEIL(DATETIME datetime, INT period)
DATETIME HOUR_CEIL(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding up time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many hours each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select hour_ceil("2023-07-13 22:28:18", 5);
+-----------------------------------------------------------+
| hour_ceil(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+-----------------------------------------------------------+
| 2023-07-14 02:00:00                                       |
+-----------------------------------------------------------+
1 row in set (0.03 sec)
```

keywords

```
HOUR_CEIL, HOUR, CEIL
```

Best Practice

See also date_ceil

DAY_CEIL

day_ceil

Description

Syntax

```
DATETIME DAY_CEIL(DATETIME datetime)
DATETIME DAY_CEIL(DATETIME datetime, DATETIME origin)
DATETIME DAY_CEIL(DATETIME datetime, INT period)
DATETIME DAY_CEIL(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding up time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many days each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

Example

```
mysql> select day_ceil("2023-07-13 22:28:18", 5);
+-------------------------------------------------------------+
| day_ceil(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+-------------------------------------------------------------+
| 2023-07-17 00:00:00                                         |
+-------------------------------------------------------------+
1 row in set (0.01 sec)
```

Keywords

```
DAY_CEIL, DAY, CEIL
```

Best Practices

See also date_ceil

MONTH_CEIL

month_ceil

description

Syntax

```
DATETIME MONTH_CEIL(DATETIME datetime)
DATETIME MONTH_CEIL(DATETIME datetime, DATETIME origin)
DATETIME MONTH_CEIL(DATETIME datetime, INT period)
DATETIME MONTH_CEIL(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding up time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many months each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select month_ceil("2023-07-13 22:28:18", 5);
+------------------------------------------------------------+
| month_ceil(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+------------------------------------------------------------+
| 2023-10-01 00:00:00                                        |
+------------------------------------------------------------+
1 row in set (0.02 sec)
```

keywords

```
MONTH_CEIL, MONTH, CEIL
```

Best Practice

See also date_ceil

YEAR_CEIL

year_ceil

description

Syntax

```
DATETIME YEAR_CEIL(DATETIME datetime)
DATETIME YEAR_CEIL(DATETIME datetime, DATETIME origin)
DATETIME YEAR_CEIL(DATETIME datetime, INT period)
DATETIME YEAR_CEIL(DATETIME datetime, INT period, DATETIME origin)
```

Convert the date to the nearest rounding up time of the specified time interval period.

- datetime: a valid date expression.
- period: specifies how many years each cycle consists of.
- origin: starting from 0001-01-01T00:00:00.

example

```
mysql> select year_ceil("2023-07-13 22:28:18", 5);
+-----------------------------------------------------------+
| year_ceil(cast('2023-07-13 22:28:18' as DATETIMEV2(0)), 5) |
+-----------------------------------------------------------+
| 2025-01-01 00:00:00                                       |
+-----------------------------------------------------------+
1 row in set (0.02 sec)
```

keywords

```
YEAR_CEIL, YEAR, CEIL
```

Best Practice

See also date_ceil

MICROSECOND

microsecond

description

Syntax

```
INT MICROSECOND(DATETIMEV2 date)
```

Returns microsecond information in the time type.

The parameter is Datetime type

example

```
mysql> select microsecond(cast('1999-01-02 10:11:12.000123' as datetimev2(6))) as microsecond;
+-------------+
| microsecond |
+-------------+
|         123 |
+-------------+
```

keywords

```
MICROSECOND
```

MICROSECONDS_ADD

microseconds_add

description

Syntax

`DATETIMEV2 microseconds_add(DATETIMEV2 basetime, INT delta)` - basetime: Base time whose type is DATETIMEV2 - delta: Microseconds to add to basetime - Return type of this function is DATETIMEV2

example

```
mysql> select now(3), microseconds_add(now(3), 100000);
+-------------------------+---------------------------------+
| now(3)                  | microseconds_add(now(3), 100000) |
+-------------------------+---------------------------------+
| 2023-02-21 11:35:56.556 | 2023-02-21 11:35:56.656          |
+-------------------------+---------------------------------+
```

`now(3)` returns current time as type DATETIMEV2 with precision 3d，`microseconds_add(now(3), 100000)` means 100000 microseconds after current time

keywords

```
microseconds_add
```

MICROSECONDS_DIFF

microseconds_diff

description

Syntax

```
INT microseconds_diff(DATETIME enddate, DATETIME startdate)
```

How many microseconds is the difference between the start time and the end time.

example

```
mysql> select microseconds_diff('2020-12-25 21:00:00.623000','2020-12-25 21:00:00.123000');
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| microseconds_diff(cast('2020-12-25 21:00:00.623000' as DATETIMEV2(6)), cast('2020-12-25 21:00:00.123000' as DATETIMEV2(6))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
|                                                                                                                                        500000 |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.12 sec)
```

keywords

```
microseconds_diff
```

MICROSECONDS_SUB

microseconds_sub

description

Syntax

`DATETIMEV2 microseconds_sub(DATETIMEV2 basetime, INT delta)` - basetime: Base time whose type is DATETIMEV2 - delta: Microseconds to subtract from basetime - Return type of this function is DATETIMEV2

example

```
mysql> select now(3), microseconds_sub(now(3), 100000);
+-------------------------+---------------------------------+
| now(3)                  | microseconds_sub(now(3), 100000) |
+-------------------------+---------------------------------+
| 2023-02-25 02:03:05.174 | 2023-02-25 02:03:05.074          |
+-------------------------+---------------------------------+
```

`now(3)` returns current time as type DATETIMEV2 with precision `3`, `microseconds_sub(now(3), 100000)` means 100000 microseconds before current time

keywords

```
microseconds_sub
```

MILLISECONDS_ADD

milliseconds_add

description

Syntax

`DATETIMEV2 milliseconds_add(DATETIMEV2 basetime, INT delta)` - basetime: Base time whose type is DATETIMEV2 - delta:Milliseconds to add to basetime - Return type of this function is DATETIMEV2

example

```
mysql> select milliseconds_add('2023-09-08 16:02:08.435123', 1);
+----------------------------------------------------------------------------+
| milliseconds_add(cast('2023-09-08 16:02:08.435123' as DATETIMEV2(6)), 1) |
+----------------------------------------------------------------------------+
| 2023-09-08 16:02:08.436123                                                 |
+----------------------------------------------------------------------------+
1 row in set (0.04 sec)
```

keywords

```
milliseconds_add
```

MILLISECONDS_DIFF

milliseconds_diff

description

Syntax

```
INT milliseconds_diff(DATETIME enddate, DATETIME startdate)
```

How many milliseconds is the difference between the start time and the end time?

example

```
mysql> select milliseconds_diff('2020-12-25 21:00:00.623000','2020-12-25 21:00:00.123000');
+-------------------------------------------------------------------------------------------------------------------------------------------+
| milliseconds_diff(cast('2020-12-25 21:00:00.623000' as DATETIMEV2(6)), cast('2020-12-25 21:00:00.123000' as DATETIMEV2(6))) |
+-------------------------------------------------------------------------------------------------------------------------------------------+
|                                                                                                                                       500 |
+-------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.03 sec)
```

keywords

```
milliseconds_diff
```

MILLISECONDS_SUB

milliseconds_sub

description

Syntax

`DATETIMEV2 milliseconds_sub(DATETIMEV2 basetime, INT delta)` - basetime: Base time whose type is DATETIMEV2 - delta: Milliseconds to subtract from basetime - Return type of this function is DATETIMEV2

example

```
mysql> select milliseconds_sub('2023-09-08 16:02:08.435123', 1);
+----------------------------------------------------------------------------+
| milliseconds_sub(cast('2023-09-08 16:02:08.435123' as DATETIMEV2(6)), 1) |
+----------------------------------------------------------------------------+
| 2023-09-08 16:02:08.434123                                                 |
+----------------------------------------------------------------------------+
1 row in set (0.11 sec)
```

keywords

```
milliseconds_sub
```

MINUTES_ADD

minutes_add

description

Syntax

```
DATETIME MINUTES_ADD(DATETIME date, INT minutes)
```

Add specified minutes from date time or date

The parameter date can be DATETIME or DATE, and the return type is DATETIME.

example

```
mysql> select minutes_add("2020-02-02", 1);
+---------------------------------------+
| minutes_add('2020-02-02 00:00:00', 1) |
+---------------------------------------+
| 2020-02-02 00:01:00                   |
+---------------------------------------+
```

keywords

```
MINUTES_ADD
```

MINUTES_DIFF

minutes_diff

description

Syntax

```
INT minutes_diff(DATETIME enddate, DATETIME startdate)
```

The difference between the start time and the end time is a few minutes

example

```
mysql> select minutes_diff('2020-12-25 22:00:00','2020-12-25 21:00:00');
+-----------------------------------------------------------+
| minutes_diff('2020-12-25 22:00:00', '2020-12-25 21:00:00') |
+-----------------------------------------------------------+
|                                                        60 |
+-----------------------------------------------------------+
```

keywords

```
minutes_diff
```

MINUTES_SUB

minutes_sub

description

Syntax

```
DATETIME MINUTES_SUB(DATETIME date, INT minutes)
```

Subtracts a specified number of minutes from a datetime or date

The parameter date can be DATETIME or DATE, and the return type is DATETIME.

example

```
mysql> select minutes_sub("2020-02-02 02:02:02", 1);
+---------------------------------------+
| minutes_sub('2020-02-02 02:02:02', 1) |
+---------------------------------------+
| 2020-02-02 02:01:02                   |
+---------------------------------------+
```

keywords

```
MINUTES_SUB
```

SECONDS_ADD

Description

The function adds or subtracts a specified number of seconds to/from a given datetime value and returns the resulting datetime.

Syntax

```
SECONDS_ADD(<datetime>, <seconds>)
```

Parameters

Parameter	Description
<datetime>	Required. The input datetime value. Supports the DATETIME and DATE type.
<seconds>	Required. The number of seconds to add or subtract. Supports integers (INT). Positive numbers add seconds, while negative numbers subtract seconds.

Return Value

- Returns a datetime value of the same type as the input .
- If <datetime> is NULL, the function returns NULL.
- If <datetime> is an invalid date (e.g., 0000-00-00T00:00:00), the function returns NULL.

Examples

```
SELECT SECONDS_ADD('2025-01-23 12:34:56', 30),SECONDS_ADD('2025-01-23 12:34:56', -30);
```

```
+-----------------------------------------------------------------+-----------------------------------------------------------------+
| seconds_add(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), 30) | seconds_add(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), -30) |
+-----------------------------------------------------------------+-----------------------------------------------------------------+
| 2025-01-23 12:35:26                                             | 2025-01-23 12:34:26                                             |
+-----------------------------------------------------------------+-----------------------------------------------------------------+
```

SECONDS_DIFF

Description

The function calculates the time difference between two datetime values and returns the difference in seconds.

Syntax

```
SECONDS_DIFF(<end_datetime>, <start_datetime>)
```

Parameters

Parameter	Description
<end_datetime>	Required. The ending datetime value. Supports the DATETIME and DATE type.
<start_datetime>	Required. The starting datetime value. Supports the DATETIME and DATE type.

Return Value

- Returns an integer representing the difference in seconds between two datetime values:
  - If <end_datetime> is later than <start_datetime>, returns a positive value.
  - If <end_datetime> is earlier than <start_datetime>, returns a negative value.
  - If <end_datetime> and <start_datetime> are equal, returns 0.
- If either parameter is NULL, the function returns NULL.
- If the input datetime values are invalid (e.g., 0000-00-00T00:00:00), the function returns NULL.

Example

```
SELECT SECONDS_DIFF('2025-01-23 12:35:56', '2025-01-23 12:34:56');
```

```
+-----------------------------------------------------------------------------------------------------------+
| seconds_diff(cast('2025-01-23 12:35:56' as DATETIMEV2(0)), cast('2025-01-23 12:34:56' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------------------------+
|                                                                                                        60 |
+-----------------------------------------------------------------------------------------------------------+
```

SECONDS_SUB

Description

The function subtracts or adds a specified number of seconds to/from a given datetime value and returns the resulting datetime.

Syntax

```
SECONDS_SUB(<datetime>, <seconds>)
```

Parameters

Parameter	Description
<datetime>	Required. The input datetime value. Supports the DATETIME and DATE type.
<seconds>	Required. The number of seconds to subtract or add. Supports integers (INT). Positive numbers add seconds, while negative numbers subtract seconds.

Return Value

- Returns a datetime value of the same type as the input .
- If <datetime> is NULL, the function returns NULL.
- If <datetime> is an invalid date (e.g., 0000-00-00T00:00:00), the function returns NULL.

Examples

```
SELECT SECONDS_SUB('2025-01-23 12:34:56', 30),SECONDS_SUB('2025-01-23 12:34:56', -30);
```

```
+-----------------------------------------------------------------+-----------------------------------------------------------------+
| seconds_sub(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), 30) | seconds_sub(cast('2025-01-23 12:34:56' as DATETIMEV2(0)), -30) |
+-----------------------------------------------------------------+-----------------------------------------------------------------+
| 2025-01-23 12:34:26                                             | 2025-01-23 12:35:26                                             |
+-----------------------------------------------------------------+-----------------------------------------------------------------+
```

HOURS_ADD

hours_add

description

Syntax

```
DATETIME HOURS_ADD(DATETIME date, INT hours)
```

Add specified hours from date time or date

The parameter date can be DATETIME or DATE, and the return type is DATETIME.

example

```
mysql> select hours_add("2020-02-02 02:02:02", 1);
+-------------------------------------+
| hours_add('2020-02-02 02:02:02', 1) |
+-------------------------------------+
| 2020-02-02 03:02:02                 |
+-------------------------------------+
```

keywords

```
HOURS_ADD
```

HOURS_DIFF

hours_diff

description

Syntax

```
INT hours_diff(DATETIME enddate, DATETIME startdate)
```

The difference between the start time and the end time is a few hours

example

```
mysql> select hours_diff('2020-12-25 22:00:00','2020-12-25 21:00:00');
+-----------------------------------------------------------+
| hours_diff('2020-12-25 22:00:00', '2020-12-25 21:00:00') |
+-----------------------------------------------------------+
|                                                         1 |
+-----------------------------------------------------------+
```

keywords

```
hours_diff
```

HOURS_SUB

hours_sub

description

Syntax

```
DATETIME HOURS_SUB(DATETIME date, INT hours)
```

Subtracts a specified number of hours from a datetime or date

The parameter date can be DATETIME or DATE, and the return type is DATETIME.

example

```
mysql> select hours_sub("2020-02-02 02:02:02", 1);
+-------------------------------------+
| hours_sub('2020-02-02 02:02:02', 1) |
+-------------------------------------+
| 2020-02-02 01:02:02                 |
+-------------------------------------+
```

keywords

```
HOURS_SUB
```

DAYS_ADD

days_add

description

Syntax

```
DATETIME DAYS_ADD(DATETIME date, INT days)
```

From date time or date plus specified days

The parameter date can be DATETIME or DATE, and the return type is consistent with that of the parameter date.

example

```
mysql> select days_add(to_date("2020-02-02 02:02:02"), 1);
+-----------------------------------------+
| days_add(to_date('2020-02-02 02:02:02'), 1) |
+-----------------------------------------+
| 2020-02-03                              |
+-----------------------------------------+
```

keywords

```
DAYS_ADD
```

DAYS_DIFF

days_diff

description

Syntax

```
INT days_diff(DATETIME enddate, DATETIME startdate)
```

The number of days between end date and start date, the date judgment is accurate to seconds and rounded down to the nearest integer. Different from the date_diff function, the datediff function judges the date with precision to the day.”

example

```
mysql> select days_diff('2020-12-25 22:00:00','2020-12-24 22:00:00');
+-----------------------------------------------------+
| days_diff('2020-12-25 22:00:00', '2020-12-24 22:00:00') |
+-----------------------------------------------------+
|                                                   1 |
+-----------------------------------------------------+

mysql> select days_diff('2020-12-25 22:00:00','2020-12-24 22:00:01');
+-----------------------------------------------------+
| days_diff('2020-12-24 22:00:01', '2020-12-25 22:00:00') |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
```

keywords

```
days_diff
```

DAYS_SUB

days_sub

description

Syntax

```
DATETIME DAYS_SUB(DATETIME date, INT days)
```

Subtract a specified number of days from a datetime or date

The parameter date can be DATETIME or DATE, and the return type is consistent with that of the parameter date.

example

```
mysql> select days_sub("2020-02-02 02:02:02", 1);
+-----------------------------------+
| days_sub('2020-02-02 02:02:02', 1) |
+-----------------------------------+
| 2020-02-01 02:02:02               |
+-----------------------------------+
```

keywords

```
DAYS_SUB
```

WEEKS_ADD

weeks_add

description

Syntax

```
DATETIME WEEKS_ADD(DATETIME date, INT weeks)
```

ADD a specified number of weeks from a datetime or date

The parameter date can be DATETIME or DATE, and the return type is consistent with that of the parameter date.

example

```
mysql> select weeks_add("2020-02-02 02:02:02", 1);
+-------------------------------------+
| weeks_add('2020-02-02 02:02:02', 1) |
+-------------------------------------+
| 2020-02-09 02:02:02                 |
+-------------------------------------+
```

keywords

```
WEEKS_ADD
```

WEEKS_DIFF

weeks_diff

description

Syntax

```
INT weeks_diff(DATETIME enddate, DATETIME startdate)
```

The difference between the start time and the end time is weeks

example

```
mysql> select weeks_diff('2020-12-25','2020-10-25');
+--------------------------------------------------------+
| weeks_diff('2020-12-25 00:00:00', '2020-10-25 00:00:00') |
+--------------------------------------------------------+
|                                                      8 |
+--------------------------------------------------------+
```

keywords

```
weeks_diff
```

WEEKS_SUB

weeks_sub

description

Syntax

```
DATETIME WEEKS_SUB(DATETIME date, INT weeks)
```

Subtracts a specified number of weeks from a datetime or date

The parameter date can be DATETIME or DATE, and the return type is consistent with that of the parameter date.

example

```
mysql> select weeks_sub("2020-02-02 02:02:02", 1);
+-------------------------------------+
| weeks_sub('2020-02-02 02:02:02', 1) |
+-------------------------------------+
| 2020-01-26 02:02:02                 |
+-------------------------------------+
```

keywords

```
WEEKS_SUB
```

MONTHS_ADD

Description

The MONTHS_ADD function is used to add or subtract a specified number of months to a given date and returns the resulting date.

Syntax

```
MONTHS_ADD(<datetime/date>, <nums>)
```

Parameters

Parameter	Description
<datetime/date>	The date value to which months will be added or subtracted
<nums>	The number of months to add or subtract

Return Value

The return value is of the same type as the input . Special cases: - If the input is 0000-00-00 or 0000-00-00 00:00:00, the function returns NULL. - If the input is NULL, the function returns NULL. - If the input is MONTHS_ADD(“9999-12-31”, 1), the function will return NULL.

Example

```
select months_add("2020-01-31 02:02:02", 1),months_add("2020-01-31", 1),months_add("2020-01-31", -1);
```

```
+-----------------------------------------------------------+-------------------------------------------+-----------------------
-----------------------+
| months_add(cast('2020-01-31 02:02:02' as DATETIMEV2(0)), 1) | months_add(cast('2020-01-31' as DATEV2), 1) | months_add(cast('2020-
01-31' as DATEV2), -1) |
+-----------------------------------------------------------+-------------------------------------------+-----------------------
-----------------------+
| 2020-02-29 02:02:02                                       | 2020-02-29                                | 2019-12-31
|
+-----------------------------------------------------------+-------------------------------------------+-----------------------
-----------------------+
```

MONTHS_DIFF

Description

The MONTHS_DIFF function calculates the number of complete months between two dates. It accepts two date arguments and returns the difference in months as an integer.

Syntax

```
MONTHS_DIFF(<enddate>, <startdate>)
```

Parameters

参数	说明
<enddate>	The ending date, representing the later date in the difference calculation. Supports DATE (e.g., YYYY-MM-DD) or DATETIME (e.g., YYYY-MM-DD HH:MM:SS) types.
<startdate>	The starting date, representing the earlier date in the difference calculation. Supports DATE (e.g., YYYY-MM-DD) or DATETIME (e.g., YYYY-MM-DD HH:MM:SS) types.

Return Value

returns the number of months resulting from <enddate> minus <startdate> - When either <enddate> or <startdate> is NULL, or both are NULL, it returns NULL

Example

```
select months_diff('2020-12-25','2020-10-25'),months_diff('2020-10-25 10:00:00','2020-12-25 11:00:00');
```

```
+-----------------------------------------------------------------------------------------+-----------------------------------------
-------------------------------------------------------------+
| months_diff(cast('2020-12-25' as DATETIMEV2(0)), cast('2020-10-25' as DATETIMEV2(0))) | months_diff(cast('2020-10-25 10:00:00' as
DATETIMEV2(0)), cast('2020-12-25 11:00:00' as DATETIMEV2(0))) |
+-----------------------------------------------------------------------------------------+-----------------------------------------
-------------------------------------------------------------+
|                                                                                       2 |
-2 |
+-----------------------------------------------------------------------------------------+-----------------------------------------
-------------------------------------------------------------+
```

MONTHS_SUB

Description

The MONTHS_SUB function is used to add or subtract a specified number of months to a given date and returns the resulting date.

Syntax

```
MONTHS_SUB(<datetime/date>,  <nums>)
```

Parameters

Parameter	Description
<datetime/date>	The date value to which months will be added or subtracted
<nums>	The number of months to add or subtract

Return Value

The return value is of the same type as the input . Special cases: - If the input is 0000-00-00 or 0000-00-00 00:00:00, the function returns NULL. - If the input is NULL, the function returns NULL. - If the input is MONTHS_SUB(“9999-12-31”, -1), the function will return NULL.

Example

```
select months_sub("2020-01-31 02:02:02", 1),months_sub("2020-01-31", 1),months_sub("2020-01-31", -1);
```

```
+-----------------------------------------------------------+-------------------------------------------+-----------------------
-----------------------+
| months_sub(cast('2020-01-31 02:02:02' as DATETIMEV2(0)), 1) | months_sub(cast('2020-01-31' as DATEV2), 1) | months_sub(cast('2020-
01-31' as DATEV2), -1) |
+-----------------------------------------------------------+-------------------------------------------+-----------------------
-----------------------+
| 2019-12-31 02:02:02                                       | 2019-12-31                                | 2020-02-29
|
+-----------------------------------------------------------+-------------------------------------------+-----------------------
-----------------------+
```

YEARS_ADD

Description

Returns a new datetime value that is the result of adding a specified number of years to the input datetime.

Syntax

```
YEARS_ADD(<date>, <years>)
```

Parameters

Parameter	Description
<date>	The input datetime value, which can be of type DATETIME or DATE
<years>	The number of years to add, of type INT

Return Value

Returns a value with the same type as the input <date> (DATETIME or DATE), representing the time value after adding the specified number of years to the input datetime.

Example

```
SELECT YEARS_ADD('2020-01-31 02:02:02', 1);
```

```
+-----------------------------------+
| years_add('2020-01-31 02:02:02', 1) |
+-----------------------------------+
| 2021-01-31 02:02:02               |
+-----------------------------------+
```

YEARS_DIFF

Description

Calculates the difference in years between two datetime values.

Syntax

```
YEARS_DIFF(<enddate>, <startdate>)
```

Parameters

Parameter	Description
<enddate>	The end date, which can be of type DATETIME or DATE
<startdate>	The start date, which can be of type DATETIME or DATE

Return Value

Returns a value of type INT, representing the number of years between the two dates.

Example

```
SELECT YEARS_DIFF('2020-12-25', '2019-10-25');
```

```
+-----------------------------------------------------------+
| years_diff('2020-12-25 00:00:00', '2019-10-25 00:00:00') |
+-----------------------------------------------------------+
|                                                         1 |
+-----------------------------------------------------------+
```

YEARS_SUB

Description

Returns a new datetime value that is the result of subtracting a specified number of years from the input datetime.

Syntax

```
YEARS_SUB(<date>, <years>)
```

Parameters

Parameter	Description
<date>	The input datetime value, which can be of type DATETIME or DATE
<years>	The number of years to subtract, of type INT

Return Value

Returns a value with the same type as the input <date> (DATETIME or DATE), representing the time value after subtracting the specified number of years from the input datetime.

Example

```
SELECT YEARS_SUB('2020-02-02 02:02:02', 1);
```

```
+-----------------------------------+
| years_sub('2020-02-02 02:02:02', 1) |
+-----------------------------------+
| 2019-02-02 02:02:02               |
+-----------------------------------+
```

ST_X

Description

When point is a valid POINT type, return the corresponding x-coordinate value

Syntax

```
ST_X( <point>)
```

Parameters

Parameters	Instructions
<point>	The geometric coordinates of a two-dimensional point

Return Value

X value in geometric coordinates

Examples

```
SELECT ST_X(ST_Point(24.7, 56.7));
```

```
+----------------------------+
| st_x(st_point(24.7, 56.7)) |
+----------------------------+
|                       24.7 |
+----------------------------+
```

ST_Y

Description

When point is a valid POINT type, return the corresponding y-coordinate value

```
ST_Y( <point>)
```

Parameters

Parameters	Instructions
<point>	The geometric coordinates of a two-dimensional point

Return Value

Y value in geometric coordinates

Examples

```
SELECT ST_Y(ST_Point(24.7, 56.7));
```

```
+---------------------------+
| ST_Y(ST_Point(24.7, 56.7)) |
+---------------------------+
| 56.7                      |
+---------------------------+
```

ST_CIRCLE

Description

Convert a WKT (Well Known Text) to a circle on the sphere of the Earth.

Syntax

```
ST_CIRCLE( <center_lng>, <center_lat>, <radius>)
```

Parameters

Parameters	Instructions
<center_lng>	Longitude of the center of the circle
<center_lat>	The latitude of the center of the circle
<radius>	Radius of a circle

- The unit of radius is meters. A maximum of 9999999 RADIUS is supported

Return Value

A circle on a sphere based on basic information about the circle

Examples

```
SELECT ST_AsText(ST_Circle(111, 64, 10000));
```

```
+-----------------------------------------+
| st_astext(st_circle(111.0, 64.0, 10000.0)) |
+-----------------------------------------+
| CIRCLE ((111 64), 10000)                |
+-----------------------------------------+
```

ST_DISTANCE_SPHERE

Description

Calculate the spherical distance in meters between two points on the Earth. The parameters passed in are the longitude of point X, latitude of point X, longitude of point Y, and latitude of point Y.

Syntax

```
ST_DISTANCE_SPHERE( <x_lng>, <x_lat>, <y_lng>, <y_lat>)
```

Parameters

Parameters	Instructions
<x_lng>	Longitude data, reasonable value range is [-180, 180]
<y_lng>	Longitude data, reasonable value range is [-180, 180]
<x_lat>	Latitude data, reasonable value range is [-90, 90]
<y_lat>	Latitude data, reasonable value range is [-90, 90]

Return Value

The spherical distance between two points

Examples

```
select st_distance_sphere(116.35620117, 39.939093, 116.4274406433, 39.9020987219);
```

```
+-----------------------------------------------------------------------------+
| st_distance_sphere(116.35620117, 39.939093, 116.4274406433, 39.9020987219) |
+-----------------------------------------------------------------------------+
|                                                          7336.9135549995917 |
+-----------------------------------------------------------------------------+
```

ST_ANGLE

Description

Enter three points that represent two intersecting lines. Return the Angle between these lines.

Syntax

```
ST_ANGLE( <point1>, <point2>, <point3>)
```

Parameters

Parameters	Instructions
<point1>	The first point of the first line
<point2>	The second point of the first line and the first point of the second line
<point3>	The second point of the second line

Return Value

The Angle between these lines is expressed in radians and ranges from [0, 2pi]. The Angle is measured clockwise from the first line to the second line.

ST_ANGLE has the following edge cases:

- If point 2 and point 3 are the same, NULL is returned
- If point 2 and point 1 are the same, NULL is returned
- NULL is returned if points 2 and 3 are perfect Antipodes
- NULL is returned if points 2 and 1 are perfect AntipodesL
- If any input geography is not a single point or is an empty geography, an error is thrown

Examples

```
SELECT ST_Angle(ST_Point(1, 0),ST_Point(0, 0),ST_Point(0, 1));
```

```
+-----------------------------------------------------------------------+
| st_angle(st_point(1.0, 0.0), st_point(0.0, 0.0), st_point(0.0, 1.0)) |
+-----------------------------------------------------------------------+
|                                                 4.71238898038469 |
+-----------------------------------------------------------------------+
```

```
SELECT ST_Angle(ST_Point(0, 0),ST_Point(1, 0),ST_Point(0, 1));
```

```
+-----------------------------------------------------------------------+
| st_angle(st_point(0.0, 0.0), st_point(1.0, 0.0), st_point(0.0, 1.0)) |
+-----------------------------------------------------------------------+
|                                              0.78547432161873854 |
+-----------------------------------------------------------------------+
```

```
SELECT ST_Angle(ST_Point(1, 0),ST_Point(0, 0),ST_Point(1, 0));
```

```
+-----------------------------------------------------------------------+
| st_angle(st_point(1.0, 0.0), st_point(0.0, 0.0), st_point(1.0, 0.0)) |
+-----------------------------------------------------------------------+
|                                                                     0 |
+-----------------------------------------------------------------------+
```

```
SELECT ST_Angle(ST_Point(1, 0),ST_Point(0, 0),ST_Point(0, 0));
```

```
+-----------------------------------------------------------------------+
| st_angle(st_point(1.0, 0.0), st_point(0.0, 0.0), st_point(0.0, 0.0)) |
+-----------------------------------------------------------------------+
|                                                                  NULL |
+-----------------------------------------------------------------------+
```

```
SELECT ST_Angle(ST_Point(0, 0),ST_Point(-30, 0),ST_Point(150, 0));
```

```
+-------------------------------------------------------------------------+
| st_angle(st_point(0.0, 0.0), st_point(-30.0, 0.0), st_point(150.0, 0.0)) |
+-------------------------------------------------------------------------+
|                                                                    NULL |
+-------------------------------------------------------------------------+
```

ST_AZIMUTH

Description

Enter two points and return the azimuth of the line segment formed by points 1 and 2. Azimuth is the arc of the Angle between the true north line of point 1 and the line segment formed by points 1 and 2.

Syntax

```
ST_AZIMUTH( <point1>, <point2>)
```

Parameters

Parameters	Instructions
<point1>	The first point used to calculate the azimuth
<point2>	The second point used to calculate the azimuth

Return Value

Positive angles are measured clockwise on the sphere. For example, the azimuth of a line segment:

- North is 0
- East is PI/2
- The guide is PI
- The west is 3PI/2

ST_Azimuth has the following edge cases:

- Return NULL if both input points are the same.
- NULL is returned if the two input points are perfect mapping points.
- An error is thrown if any of the input geographies are not a single point or are empty geographies

Examples

```
SELECT st_azimuth(ST_Point(1, 0),ST_Point(0, 0));
```

```
+---------------------------------------------------+
| st_azimuth(st_point(1.0, 0.0), st_point(0.0, 0.0)) |
+---------------------------------------------------+
|                                 4.71238898038469 |
+---------------------------------------------------+
```

```
SELECT st_azimuth(ST_Point(0, 0),ST_Point(1, 0));
```

```
+-----------------------------------------------------+
| st_azimuth(st_point(0.0, 0.0), st_point(1.0, 0.0)) |
+-----------------------------------------------------+
|                                1.5707963267948966 |
+-----------------------------------------------------+
```

```
SELECT st_azimuth(ST_Point(0, 0),ST_Point(0, 1));
```

```
+-----------------------------------------------------+
| st_azimuth(st_point(0.0, 0.0), st_point(0.0, 1.0)) |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
```

```
SELECT st_azimuth(ST_Point(-30, 0),ST_Point(150, 0));
```

```
+-------------------------------------------------------+
| st_azimuth(st_point(-30.0, 0.0), st_point(150.0, 0.0)) |
+-------------------------------------------------------+
|                                                  NULL |
+-------------------------------------------------------+
```

ST_ANGLE_SPHERE

Description

Calculate the center Angle between two points on the Earth's surface in degrees. The parameters passed in are the longitude of point X, latitude of point X, longitude of point Y, and latitude of point Y.

Syntax

```
ST_ANGLE_SPHERE( <x_lng>, <x_lat>, <y_lng>, <y_lat>)
```

Parameters

Parameters	Instructions
<x_lng>	Longitude data, reasonable value range is [-180, 180]
<y_lng>	Longitude data, reasonable value range is [-180, 180]
<x_lat>	Latitude data, reasonable value range is [-90, 90]
<y_lat>	Latitude data, reasonable value range is [-90, 90]

Return Value

The Angle of the center between two points

Examples

```
select ST_Angle_Sphere(116.35620117, 39.939093, 116.4274406433, 39.9020987219);
```

```
+-------------------------------------------------------------------------+
| st_angle_sphere(116.35620117, 39.939093, 116.4274406433, 39.9020987219) |
+-------------------------------------------------------------------------+
|                                                  0.0659823452409903 |
+-------------------------------------------------------------------------+
```

```
select ST_Angle_Sphere(0, 0, 45, 0);
```

```
+--------------------------------------+
| st_angle_sphere(0.0, 0.0, 45.0, 0.0) |
+--------------------------------------+
|                                   45 |
+--------------------------------------+
```

ST_AREA_SQUARE_METERS

Description

Calculate the area of the region on the sphere of the Earth

Syntax

```
ST_AREA_SQUARE_METERS( <geo>)
```

Parameters

Parameters	Instructions
<geo>	The spherical position of the earth

Return Value

ST_Area_Square_Meters( ): the units returned are square meters

Examples

```
SELECT ST_Area_Square_Meters(ST_Circle(0, 0, 1));
```

```
+-------------------------------------------------+
| st_area_square_meters(st_circle(0.0, 0.0, 1.0)) |
+-------------------------------------------------+
|                              3.1415926535897869 |
+-------------------------------------------------+
```

```
SELECT ST_Area_Square_Meters(ST_Point(0, 1));
```

```
+-----------------------------------------+
| st_area_square_meters(st_point(0.0, 1.0)) |
+-----------------------------------------+
|                                       0 |
+-----------------------------------------+
```

```
SELECT ST_Area_Square_Meters(ST_LineFromText("LINESTRING (1 1, 2 2)"));
```

```
+-----------------------------------------------------------------+
| st_area_square_meters(st_linefromtext('LINESTRING (1 1, 2 2)')) |
+-----------------------------------------------------------------+
|                                                               0 |
+-----------------------------------------------------------------+
```

ST_AREA_SQUARE_KM

Description

Calculate the area of the region on the sphere of the Earth

Syntax

```
ST_AREA_SQUARE_KM( <geo>)
```

Parameters

Parameters	Instructions
<geo>	The spherical position of the earth

Return Value

ST_Area_Square_Km( ): the units returned are square kilometers

Examples

```
SELECT ST_Area_Square_Km(ST_Polygon("POLYGON ((0 0, 1 0, 1 1, 0 1, 0 0))"));
```

```
+-----------------------------------------------------------------------+
| st_area_square_km(st_polygon('POLYGON ((0 0, 1 0, 1 1, 0 1, 0 0))')) |
+-----------------------------------------------------------------------+
|                                              12364.036567076409 |
+-----------------------------------------------------------------------+
```

ST_POINT

Description

With the given X coordinate value, the Y coordinate value returns the corresponding Point.

The current value is only meaningful in the sphere set, X/Y corresponds to longitude/latitude;

Syntax

```
ST_POINT( <x>, <y>)
```

Parameters

Parameters	Instructions
<x>	x-coordinate
<y>	y-coordinate

Return Value

Given horizontal coordinate and vertical coordinate corresponding position information

Examples

```
SELECT ST_AsText(ST_Point(24.7, 56.7));
```

```
+--------------------------------+
| st_astext(st_point(24.7, 56.7)) |
+--------------------------------+
| POINT (24.7 56.7)              |
+--------------------------------+
```

ST_POLYGON

Description

Convert a WKT (Well Known Text) to the corresponding polygonal memory form

Alias

- ST_POLYGONFROMTEXT
- ST_POLYFROMTEXT

Syntax

```
ST_POLYGON( <wkt>)
```

Parameters

Parameters	Instructions
<wkt>	A POLYGON generated by the polygon function

Return Value

Memory form of polygon

Examples

```
SELECT ST_AsText(ST_Polygon("POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))"));
```

```
+-----------------------------------------------------------------+
| st_astext(st_polygon('POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))')) |
+-----------------------------------------------------------------+
| POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))                         |
+-----------------------------------------------------------------+
```

ST_ASTEXT

Description

Convert a geometric figure to a representation of WKT (Well Known Text)

Alias

- ST_ASWKT

Syntax

```
ST_ASTEXT( <geo>)
```

Parameters

Parameters	Instructions
<geo>	The graph that needs to be converted

Return Value

The WKT representation of the geometry:

Examples

```
SELECT ST_AsText(ST_Point(24.7, 56.7));
```

```
+---------------------------------+
| st_astext(st_point(24.7, 56.7)) |
+---------------------------------+
| POINT (24.7 56.7)               |
+---------------------------------+
```

ST_CONTAINS

Description

Determines whether the geometry shape1 is fully capable of containing the geometry shape2

Syntax

```
ST_CONTAINS( <shape1>, <shape2>)
```

Parameters

Parameters	Instructions
<shape1>	The passed geometry used to determine whether shape2 is included
<shape2>	The passed geometry is used to determine whether shape1 is included

Return Value

Return 1:shape1 The graph can contain the graph shape2

Return 0:shape1 Graph cannot contain graph shape2

Examples

```
SELECT ST_Contains(ST_Polygon("POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))"), ST_Point(5, 5));
```

```
+-----------------------------------------------------------------------------------+
| st_contains(st_polygon('POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))'), st_point(5.0, 5.0)) |
+-----------------------------------------------------------------------------------+
|                                                                                 1 |
+-----------------------------------------------------------------------------------+
```

```
SELECT ST_Contains(ST_Polygon("POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))"), ST_Point(50, 50));
```

```
+-------------------------------------------------------------------------------------+
| st_contains(st_polygon('POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))'), st_point(50.0, 50.0)) |
+-------------------------------------------------------------------------------------+
|                                                                                   0 |
+-------------------------------------------------------------------------------------+
```

ST_GEOMETRYFROMTEXT

Description

Convert a linear WKT (Well Known Text) to the corresponding memory geometry

Alias

- ST_GEOMFROMTEXT

Syntax

```
ST_GEOMETRYFROMTEXT( <wkt>)
```

Parameters

Parameters	Instructions
<wkt>	The memory form of the graph

Return Value

The corresponding geometric storage form of WKB

Examples

```
SELECT ST_AsText(ST_GeometryFromText("LINESTRING (1 1, 2 2)"));
```

```
+----------------------------------------------------------+
| st_astext(st_geometryfromtext('LINESTRING (1 1, 2 2)')) |
+----------------------------------------------------------+
| LINESTRING (1 1, 2 2)                                    |
+----------------------------------------------------------+
```

ST_LINEFROMTEXT

Description

Converts a WKT (Well Known Text) to a memory representation in the form of a Line

Alias

- ST_LINESTRINGFROMTEXT

Syntax

```
ST_LINEFROMTEXT( <wkt>)
```

Parameters

Parameters	Instructions
<wkt>	A line segment consisting of two coordinates

Return Value

The memory form of a line segment.

Examples

```
SELECT ST_AsText(ST_LineFromText("LINESTRING (1 1, 2 2)"));
```

```
+-----------------------------------------------------+
| st_astext(st_geometryfromtext('LINESTRING (1 1, 2 2)')) |
+-----------------------------------------------------+
| LINESTRING (1 1, 2 2)                               |
+-----------------------------------------------------+
```

ST_ASBINARY

Description

Convert a geometric figure to a standard WKB (Well-known binary) representation.

At present, the supported geometry is: Point, LineString, Polygon.

Syntax

```
ST_ASBINARY( <geo>)
```

Parameters

Parameters	Instructions
<geo>	The graph that needs to be converted

Return Value

The WKB representation of the geometry

Examples

```
select ST_AsBinary(st_point(24.7, 56.7));
```

```
+--------------------------------------------+
| st_asbinary(st_point(24.7, 56.7))          |
+--------------------------------------------+
| \x0101000000333333333b338409a99999999594c40 |
+--------------------------------------------+
```

```
select ST_AsBinary(ST_GeometryFromText("LINESTRING (1 1, 2 2)"));
```

```
+-------------------------------------------------------------------------------------+
| st_asbinary(st_geometryfromtext('LINESTRING (1 1, 2 2)'))                           |
+-------------------------------------------------------------------------------------+
| \x010200000002000000000000000000f03f000000000000f03f0000000000000040000000000000040 |
+-------------------------------------------------------------------------------------+
```

```
select ST_AsBinary(ST_Polygon("POLYGON ((114.104486 22.547119,114.093758 22.547753,114.096504 22.532057,114.104229
22.539826,114.106203 22.542680,114.104486 22.547119))"));
```

```
+-------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+
```

```
| st_asbinary(st_polygon('POLYGON ((114.104486 22.547119,114.093758 22.547753,114.096504 22.532057,114.104229 22.539826,114.106203
22.542680,114.104486 22.547119))'))                                                                                                |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|
\x0103000000010000000600000f3380ce6af865c402d05a4fd0f8c364041ef8d2100865c403049658a398c3640b9fb1c1f2d865c409d9b36e334883640de921cb0
ab865c40cf876709328a36402cefaa07cc865c407b319413ed8a3640f3380ce6af865c402d05a4fd0f8c3640 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

ST_GEOMETRYFROMWKB

Description

Converts a standard figure WKB (Well-known binary) to the corresponding memory geometry

Alias

- ST_GEOMFROMWKB

Syntax

```
ST_GEOMETRYFROMWKB( <wkb>)
```

Parameters

Parameters	Instructions
<wkb>	The memory form of the graph

Return Value

The corresponding geometric storage form of WKB

Examples

```
select ST_AsText(ST_GeometryFromWKB(ST_AsBinary(ST_Point(24.7, 56.7))));
```

```
+-------------------------------------------------------------------+
| st_astext(st_geometryfromwkb(st_asbinary(st_point(24.7, 56.7)))) |
+-------------------------------------------------------------------+
| POINT (24.7 56.7)                                                 |
+-------------------------------------------------------------------+
```

```
select ST_AsText(ST_GeomFromWKB(ST_AsBinary(ST_Point(24.7, 56.7))));
```

```
+-----------------------------------------------------------+
| st_astext(st_geomfromwkb(st_asbinary(st_point(24.7, 56.7)))) |
+-----------------------------------------------------------+
| POINT (24.7 56.7)                                         |
+-----------------------------------------------------------+
```

```
select ST_AsText(ST_GeometryFromWKB(ST_AsBinary(ST_GeometryFromText("LINESTRING (1 1, 2 2)"))));
```

```
+-------------------------------------------------------------------------------------+
| st_astext(st_geometryfromwkb(st_asbinary(st_geometryfromtext('LINESTRING (1 1, 2 2)')))) |
+-------------------------------------------------------------------------------------+
| LINESTRING (1 1, 2 2)                                                               |
+-------------------------------------------------------------------------------------+
```

```
select ST_AsText(ST_GeometryFromWKB(ST_AsBinary(ST_Polygon("POLYGON ((114.104486 22.547119,114.093758 22.547753,114.096504
22.532057,114.104229 22.539826,114.106203 22.542680,114.104486 22.547119))"))));
```

```
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+
| st_astext(st_geometryfromwkb(st_asbinary(st_polygon('POLYGON ((114.104486 22.547119,114.093758 22.547753,114.096504
22.532057,114.104229 22.539826,114.106203 22.542680,114.104486 22.547119))')))) |
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+
| POLYGON ((114.104486 22.547119, 114.093758 22.547753, 114.096504 22.532057, 114.104229 22.539826, 114.106203 22.54268, 114.104486
22.547119))                                                  |
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+
```

```
select ST_AsText(ST_GeomFromWKB(ST_AsBinary(ST_Polygon("POLYGON ((114.104486 22.547119,114.093758 22.547753,114.096504
22.532057,114.104229 22.539826,114.106203 22.542680,114.104486 22.547119))"))));
```

```
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+
| st_astext(st_geomfromwkb(st_asbinary(st_polygon('POLYGON ((114.104486 22.547119,114.093758 22.547753,114.096504
22.532057,114.104229 22.539826,114.106203 22.542680,114.104486 22.547119))')))) |
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+
| POLYGON ((114.104486 22.547119, 114.093758 22.547753, 114.096504 22.532057, 114.104229 22.539826, 114.106203 22.54268, 114.104486
22.547119))                                                  |
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+
```

TO_BASE64

to_base64

description

Syntax

```
VARCHAR to_base64(VARCHAR str)
```

Returns the result of Base64 encoding the input string

example

```
mysql> select to_base64('1');
+----------------+
| to_base64('1') |
+----------------+
| MQ==           |
+----------------+

mysql> select to_base64('234');
+------------------+
| to_base64('234') |
+------------------+
| MjM0             |
+------------------+
```

keywords

```
to_base64
```

FROM_BASE64

from_base64

description

Syntax

```
VARCHAR from_base64(VARCHAR str)
```

Returns the result of Base64 decoding the input string

example

```
mysql> select from_base64('MQ==');
+---------------------+
| from_base64('MQ==') |
+---------------------+
| 1                   |
+---------------------+

mysql> select from_base64('MjM0');
+---------------------+
| from_base64('MjM0') |
+---------------------+
| 234                 |
+---------------------+
```

keywords

```
from_base64
```

ASCII

ascii

Description

Syntax

```
INT AXES (WARCHAR STR)
```

Returns the ASCII code corresponding to the first character of the string

example

```
mysql> select ascii('1');
+------------+
| ascii('1') |
+------------+
|         49 |
+------------+

mysql> select ascii('234');
+--------------+
| ascii('234') |
+--------------+
|           50 |
+--------------+
```

keywords

```
ASCII
```

LENGTH

length

Description

Syntax

```
INT length (VARCHAR str)
```

Returns the length of the string in byte size.

example

```
mysql> select length("abc");
+---------------+
| length('abc') |
+---------------+
|             3 |
+---------------+

mysql> select length("中国");
+------------------+
| length('中国')   |
+------------------+
|                6 |
+------------------+
```

keywords

```
LENGTH
```

BIT_LENGTH

bit_length

Description

Syntax

```
INT bit_length (VARCHAR str)
```

Return length of argument in bits.

example

```
mysql> select bit_length("abc");
+------------------+
| bit_length('abc') |
+------------------+
|                24 |
+------------------+

mysql> select bit_length("中国");
+----------------------+
| bit_length('中国')    |
+----------------------+
|                   48 |
+----------------------+
```

keywords

```
BIT_LENGTH
```

CHAR_LENGTH

char_length

Description

Syntax

INT char_length(VARCHAR str)

Returns the length of the string, and the number of characters returned for multi-byte characters. For example, five two-byte width words return a length of 5, only utf8 encoding is support at the current version. `character_length` is the alias for this function.

example

```
mysql> select char_length("abc");
+--------------------+
| char_length('abc') |
+--------------------+
|                  3 |
+--------------------+

mysql> select char_length("中国");
+------------------- ---+
| char_length('中国')   |
+-----------------------+
|                     2 |
+-----------------------+
```

keywords

```
CHAR_LENGTH, CHARACTER_LENGTH
```

LPAD

lpad

Description

Syntax

```
VARCHAR lpad (VARCHAR str, INT len, VARCHAR pad)
```

Returns a string of length len in str, starting with the initials. If len is longer than str, pad characters are added to STR until the length of the string reaches len. If len is less than str's length, the function is equivalent to truncating STR strings and returning only strings of len's length. The len is character length not the bye size.

Except when it contains a NULL in parameters, when pad's length is 0, the return value is the empty string.

example

```
mysql> SELECT lpad("hi", 5, "xy");
+---------------------+
| lpad('hi', 5, 'xy') |
+---------------------+
| xyxhi               |
+---------------------+

mysql> SELECT lpad("hi", 1, "xy");
+---------------------+
| lpad('hi', 1, 'xy') |
+---------------------+
| h                   |
+---------------------+

mysql> SELECT lpad("", 0, "");
+-----------------+
| lpad('', 0, '') |
+-----------------+
|                 |
+-----------------+
```

keywords

```
LPAD
```

RPAD

rpad

Description

Syntax

```
VARCHAR rpad (VARCHAR str, INT len, VARCHAR pad)
```

Returns a string of length len in str, starting with the initials. If len is longer than str, pad characters are added to the right of STR until the length of the string reaches len. If len is less than str's length, the function is equivalent to truncating STR strings and returning only strings of len's length. The len is character length not the bye size.

Except when it contains a NULL in parameters, when pad's length is 0, the return value is the empty string.

example

```
mysql> SELECT rpad("hi", 5, "xy");
+---------------------+
| rpad('hi', 5, 'xy') |
+---------------------+
| hixyx               |
+---------------------+

mysql> SELECT rpad("hi", 1, "xy");
+---------------------+
| rpad('hi', 1, 'xy') |
+---------------------+
| h                   |
+---------------------+

mysql> SELECT rpad("", 0, "");
+-----------------+
| rpad('', 0, '') |
+-----------------+
|                 |
+-----------------+
```

keywords

```
RPAD
```

LOWER

lower

Description

Syntax

```
VARCHAR lower (VARCHAR str)
```

Convert all strings in parameters to lowercase. Another alias for this function is lcase.

example

```
mysql> SELECT lower("AbC123");
+-----------------+
| lower('AbC123') |
+-----------------+
| abc123          |
+-----------------+
```

keywords

```
LOWER
```

LCASE

lcase

Description

Syntax

```
INT lcase (VARCHAR str)
```

Convert all strings in parameters to lowercase. Another alias for this function is lower.

keywords

```
LCASE
```

UPPER

upper

description

Syntax

```
VARCHAR upper(VARCHAR str)
```

Convert all strings in parameters to uppercase. Another alias for this function is ucase.

example

```
mysql> SELECT upper("aBc123");
+-----------------+
| upper('aBc123') |
+-----------------+
| ABC123          |
+-----------------+
```

keywords

```
UPPER
```

UCASE

ucase

description

Syntax

```
VARCHAR ucase(VARCHAR str)
```

Convert all strings in parameters to uppercase. Another alias for this function is upper.

keywords

```
UCASE
```

INITCAP

initcap

description

Syntax

```
VARCHAR initcap(VARCHAR str)
```

Convert the first letter of each word to upper case and the rest to lower case. Words are sequences of alphanumeric characters separated by non-alphanumeric characters.

example

```
mysql> select initcap('hello hello.,HELLO123HELlo');
+---------------------------------------+
| initcap('hello hello.,HELLO123HELlo') |
+---------------------------------------+
| Hello Hello.,Hello123hello            |
+---------------------------------------+
```

keywords

```
INITCAP
```

REPEAT

repeat

Description

Syntax

```
VARCHAR repeat (VARCHAR str, INT count)
```

Repeat the str of the string count times, return empty string when count is less than 1, return NULL when str, count is any NULL

example

```
mysql> SELECT repeat("a", 3);
+----------------+
| repeat('a', 3) |
+----------------+
| aaa            |
+----------------+

mysql> SELECT repeat("a", -1);
+-----------------+
| repeat('a', -1) |
+-----------------+
|                 |
+-----------------+
```

keywords

```
REPEAT
```

REVERSE

reverse

description

Syntax

```
VARCHAR reverse(VARCHAR str)
ARRAY<T> reverse(ARRAY<T> arr)
```

The REVERSE() function reverses a string or array and returns the result.

example

```
mysql> SELECT REVERSE('hello');
+------------------+
| REVERSE('hello') |
+------------------+
| olleh            |
+------------------+
1 row in set (0.00 sec)

mysql> SELECT REVERSE('你好');
+------------------+
| REVERSE('你好')   |
+------------------+
| 好你              |
+------------------+
1 row in set (0.00 sec)

mysql> select k1, k2, reverse(k2) from array_test order by k1;
+------+-----------------------------+-----------------------------+
| k1   | k2                          | reverse(`k2`)               |
+------+-----------------------------+-----------------------------+
|  1   | [1, 2, 3, 4, 5]             | [5, 4, 3, 2, 1]             |
|  2   | [6, 7, 8]                   | [8, 7, 6]                   |
|  3   | []                          | []                          |
|  4   | NULL                        | NULL                        |
|  5   | [1, 2, 3, 4, 5, 4, 3, 2, 1] | [1, 2, 3, 4, 5, 4, 3, 2, 1] |
|  6   | [1, 2, 3, NULL]             | [NULL, 3, 2, 1]             |
|  7   | [4, 5, 6, NULL, NULL]       | [NULL, NULL, 6, 5, 4]       |
+------+-----------------------------+-----------------------------+

mysql> select k1, k2, reverse(k2) from array_test01 order by k1;
+------+---------------------------------+---------------------------------+
| k1   | k2                              | reverse(`k2`)                   |
+------+---------------------------------+---------------------------------+
|  1   | ['a', 'b', 'c', 'd']            | ['d', 'c', 'b', 'a']            |
|  2   | ['e', 'f', 'g', 'h']            | ['h', 'g', 'f', 'e']            |
|  3   | [NULL, 'a', NULL, 'b', NULL, 'c'] | ['c', NULL, 'b', NULL, 'a', NULL] |
|  4   | ['d', 'e', NULL, ' ']           | [' ', NULL, 'e', 'd']           |
|  5   | [' ', NULL, 'f', 'g']           | ['g', 'f', NULL, ' ']           |
+------+---------------------------------+---------------------------------+
```

keywords

```
REVERSE, ARRAY
```

CHAR

> **Tips**
>
> This function is supported since the Apache Doris 1.2.

function char

description

Syntax

```
VARCHAR char(INT,..., [USING charset_name])
```

Interprets each argument as an integer and returns a string consisting of the characters given by the code values of those integers. `NULL` values are skipped.

If the result string is illegal for the given character set, the result from `CHAR()` becomes `NULL`.

Arguments larger than `255` are converted into multiple result bytes. For example, `char(15049882)` is equivalent to `char(229, 164, 154)`.

Currently only `utf8` is supported for `charset_name`.

example

```
mysql> select char(68, 111, 114, 105, 115);
+-------------------------------------+
| char('utf8', 68, 111, 114, 105, 115) |
+-------------------------------------+
| Doris                               |
+-------------------------------------+

mysql> select char(15049882, 15179199, 14989469);
+-------------------------------------------+
| char('utf8', 15049882, 15179199, 14989469) |
+-------------------------------------------+
| 多睿丝                                    |
+-------------------------------------------+

mysql> select char(255);
+------------------+
| char('utf8', 255) |
+------------------+
| NULL             |
+------------------+
```

keywords

```
CHAR
```

CONCAT

concat

Description

Syntax

```
VARCHAR concat (VARCHAR,...)
```

Connect multiple strings and return NULL if any of the parameters is NULL

example

```
mysql> select concat("a", "b");
+------------------+
| concat('a', 'b') |
+------------------+
| ab               |
+------------------+

mysql> select concat("a", "b", "c");
+-----------------------+
| concat('a', 'b', 'c') |
+-----------------------+
| abc                   |
+-----------------------+

mysql> select concat("a", null, "c");
+------------------------+
| concat('a', NULL, 'c') |
+------------------------+
| NULL                   |
+------------------------+
```

keywords

```
CONCAT
```

CONCAT_WS

concat_ws

Description

Syntax

```
VARCHAR concat_ws(VARCHAR sep, VARCHAR str,...)
VARCHAR concat_ws(VARCHAR sep, ARRAY array)
```

Using the first parameter SEP as a connector, the second parameter and all subsequent parameters(or all string in an ARRAY) are spliced into a string. If the separator is NULL, return NULL. The `concat_ws` function does not skip empty strings, it skips NULL values.

example

```
mysql> select concat_ws("or", "d", "is");
+----------------------------+
| concat_ws('or', 'd', 'is') |
+----------------------------+
| doris                      |
+----------------------------+

mysql> select concat_ws(NULL, "d", "is");
+----------------------------+
| concat_ws(NULL, 'd', 'is') |
+----------------------------+
| NULL                       |
+----------------------------+

mysql> select concat_ws("or", "d", NULL,"is");
+---------------------------------+
| concat_ws("or", "d", NULL,"is") |
+---------------------------------+
| doris                           |
+---------------------------------+

mysql> select concat_ws("or", ["d", "is"]);
+----------------------------------+
| concat_ws('or', ARRAY('d', 'is')) |
+----------------------------------+
| doris                            |
+----------------------------------+

mysql> select concat_ws(NULL, ["d", "is"]);
+----------------------------------+
| concat_ws(NULL, ARRAY('d', 'is')) |
+----------------------------------+
| NULL                             |
+----------------------------------+

mysql> select concat_ws("or", ["d", NULL,"is"]);
+---------------------------------------+
```

```
| concat_ws('or', ARRAY('d', NULL, 'is')) |
+-----------------------------------------+
| doris                                   |
+-----------------------------------------+
```

keywords

```
CONCAT_WS,CONCAT,WS,ARRAY
```

SUBSTR

substr

Description

Syntax

```
VARCHAR substr(VARCHAR content, INT start, INT length)
```

Find a substring, return the part of the string described by the first parameter starting from start and having a length of len. The index of the first letter is 1.

example

```
mysql> select substr("Hello doris", 2, 1);
+-----------------------------+
| substr('Hello doris', 2, 1) |
+-----------------------------+
| e                           |
+-----------------------------+
mysql> select substr("Hello doris", 1, 2);
+-----------------------------+
| substr('Hello doris', 1, 2) |
+-----------------------------+
| He                          |
+-----------------------------+
```

keywords

```
SUBSTR
```

SUBSTRING

substring

description

Syntax

```
VARCHAR substring(VARCHAR str, INT pos[, INT len])
```

The forms without a `len` argument return a substring from string `str` starting at position `pos`. The forms with a `len` argument return a substring len characters long from string `str`, starting at position pos. It is also possible to use a negative value for `pos`. In this case, the beginning of the substring is `pos` characters from the end of the string, rather than the beginning. A negative value may be used for `pos` in any of the forms of this function. A value of 0 for `pos` returns an empty string.

For all forms of SUBSTRING(), the position of the first character in the string from which the substring is to be extracted is reckoned as 1.

If len is less than 1, the result is the empty string.

example

```
mysql> select substring('abc1', 2);
+-----------------------------+
| substring('abc1', 2)        |
+-----------------------------+
| bc1                         |
+-----------------------------+

mysql> select substring('abc1', -2);
+-----------------------------+
| substring('abc1', -2)       |
+-----------------------------+
| c1                          |
+-----------------------------+

mysql> select substring('abc1', 0);
+----------------------+
| substring('abc1', 0) |
+----------------------+
|                      |
+----------------------+

mysql> select substring('abc1', 5);
+-----------------------------+
| substring('abc1', 5)        |
+-----------------------------+
| NULL                        |
+-----------------------------+

mysql> select substring('abc1def', 2, 2);
+-----------------------------+
```

```
| substring('abc1def', 2, 2)  |
+-----------------------------+
| bc                          |
+-----------------------------+
```

keywords

```
SUBSTRING, STRING
```

SUB_REPLACE

sub_replace

Description

Syntax

```
VARCHAR sub_replace(VARCHAR str, VARCHAR new_str, INT start[, INT len])
```

Return with new_str replaces the str with length and starting position from start. When start and len are negative integers, return NULL. and the default value of len is the length of new_str.

example

```
mysql> select sub_replace("this is origin str","NEW-STR",1);
+-------------------------------------------------+
| sub_replace('this is origin str', 'NEW-STR', 1) |
+-------------------------------------------------+
| tNEW-STRorigin str                              |
+-------------------------------------------------+

mysql> select sub_replace("doris","***",1,2);
+-----------------------------------+
| sub_replace('doris', '***', 1, 2) |
+-----------------------------------+
| d***is                            |
+-----------------------------------+
```

keywords

```
SUB_REPLACE
```

APPEND_TRAILING_CHAR_IF_ABSENT

append_trailing_char_if_absent

description

Syntax

```
VARCHAR append_trailing_char_if_absent(VARCHAR str, VARCHAR trailing_char)
```

If the @str string is non-empty and does not contain the @trailing_char character at the end, it appends the @trailing_char character to the end. @trailing_char contains only one character, and it will return NULL if contains more than one character

example

```
MySQL [test]> select append_trailing_char_if_absent('a','c');
+-----------------------------------------+
| append_trailing_char_if_absent('a', 'c') |
+-----------------------------------------+
| ac                                      |
+-----------------------------------------+
1 row in set (0.02 sec)

MySQL [test]> select append_trailing_char_if_absent('ac','c');
+------------------------------------------+
| append_trailing_char_if_absent('ac', 'c') |
+------------------------------------------+
| ac                                       |
+------------------------------------------+
1 row in set (0.00 sec)
```

keywords

```
APPEND_TRAILING_CHAR_IF_ABSENT
```

ENDS_WITH

ends_with

Description

Syntax

```
BOOLEAN ENDS_WITH(VARCHAR str, VARCHAR suffix)
```

It returns true if the string ends with the specified suffix, otherwise it returns false. If any parameter is NULL, it returns NULL.

example

```
mysql> select ends_with("Hello doris", "doris");
+-----------------------------------+
| ends_with('Hello doris', 'doris') |
+-----------------------------------+
|                                 1 |
+-----------------------------------+

mysql> select ends_with("Hello doris", "Hello");
+-----------------------------------+
| ends_with('Hello doris', 'Hello') |
+-----------------------------------+
|                                 0 |
+-----------------------------------+
```

keywords

```
ENDS_WITH
```

STARTS_WITH

starts_with

Description

Syntax

```
BOOLEAN STARTS_WITH(VARCHAR str, VARCHAR prefix)
```

It returns true if the string starts with the specified prefix, otherwise it returns false. If any parameter is NULL, it returns NULL.

example

```
MySQL [(none)]> select starts_with("hello world","hello");
+-------------------------------------+
| starts_with('hello world', 'hello') |
+-------------------------------------+
|                                   1 |
+-------------------------------------+

MySQL [(none)]> select starts_with("hello world","world");
+-------------------------------------+
| starts_with('hello world', 'world') |
+-------------------------------------+
|                                   0 |
+-------------------------------------+
```

keywords

```
STARTS_WITH
```

TRIM

trim

description

Syntax

```
VARCHAR trim(VARCHAR str[, VARCHAR rhs])
```

When the ‘rhs’ parameter is not present, remove the continuous spaces that appear from the starting and ending of the ‘str’ parameter. Otherwise, remove ‘rhs’.

example

```
mysql> SELECT trim('   ab d   ') str;
+------+
| str  |
+------+
| ab d |
+------+

mysql> SELECT trim('ababccaab','ab') str;
+------+
| str  |
+------+
| cca  |
+------+
```

keywords

```
TRIM
```

LTRIM

ltrim

Description

Syntax

```
VARCHAR ltrim(VARCHAR str[, VARCHAR rhs])
```

When the ‘rhs’ parameter is not present, remove the continuous spaces that appear from the beginning of the ‘str’ parameter. Otherwise, remove ‘rhs’.

example

```
mysql> SELECT ltrim('   ab d');
+------------------+
| ltrim('   ab d') |
+------------------+
| ab d             |
+------------------+

mysql> SELECT ltrim('ababccaab','ab') str;
+-------+
| str   |
+-------+
| ccaab |
+-------+
```

keywords

```
LTRIM
```

RTRIM

rtrim

description

Syntax

```
VARCHAR rtrim(VARCHAR str[, VARCHAR rhs])
```

When the ‘rhs’ parameter is not present, remove the continuous spaces that appear from the ending of the ‘str’ parameter. Otherwise, remove ‘rhs’.

example

```
mysql> SELECT rtrim('ab d   ') str;
+------+
| str  |
+------+
| ab d |
+------+

mysql> SELECT rtrim('ababccaab','ab') str;
+---------+
| str     |
+---------+
| ababcca |
+---------+
```

keywords

```
RTRIM
```

NULL_OR_EMPTY

Description

The `null_or_empty` function checks if the given value is NULL or an empty string. It returns true if the input value is NULL or an empty string, otherwise, it returns false.

Syntax

```
NULL_OR_EMPTY (<str>)
```

Parameters

Parameter	Description
<str>	The string to check.

Return Value

Returns true if the string is an empty string or NULL, otherwise returns false.

Examples

```
select null_or_empty(null);
```

```
+---------------------+
| null_or_empty(NULL) |
+---------------------+
|                   1 |
+---------------------+
```

```
select null_or_empty("");
```

```
+-------------------+
| null_or_empty('') |
+-------------------+
|                 1 |
+-------------------+
```

```
select null_or_empty("a");
```

```
+--------------------+
| null_or_empty('a') |
+--------------------+
```

```
|                   0 |
+---------------------+
```

NOT_NULL_OR_EMPTY

Description

The `not_null_or_empty` function checks if the given value is neither NULL nor empty. It returns true if the input value is not NULL and not an empty string, otherwise, it returns false.

Syntax

```
NOT_NULL_OR_EMPTY (<str>)
```

Parameters

Parameter	Description
<str>	The string to check

Return Value

Returns false if the string is an empty string or NULL, otherwise returns true.

Examples

```
select not_null_or_empty(null);
```

```
+-------------------------+
| not_null_or_empty(NULL) |
+-------------------------+
|                       0 |
+-------------------------+
```

```
select not_null_or_empty("");
```

```
+----------------------+
| not_null_or_empty('') |
+----------------------+
|                    0 |
+----------------------+
```

```
select not_null_or_empty("a");
```

```
+-----------------------+
| not_null_or_empty('a') |
+-----------------------+
```

```
|                    1 |
+----------------------+
```

HEX

hex

description

Syntax

```
VARCHAR hex(VARCHAR str)

VARCHAR hex(BIGINT num)
```

If the input parameter is a number, the string representation of the hexadecimal value is returned;

If the input parameter is a string, each character will be converted into two hexadecimal characters, and all the characters after the conversion will be spliced into a string for output

example

```
input string

mysql> select hex('1');
+----------+
| hex('1') |
+----------+
| 31       |
+----------+

mysql> select hex('@');
+----------+
| hex('@') |
+----------+
| 40       |
+----------+

mysql> select hex('12');
+-----------+
| hex('12') |
+-----------+
| 3132      |
+-----------+
```

```
intput num

mysql> select hex(12);
+---------+
| hex(12) |
+---------+
| C       |
+---------+

mysql> select hex(-1);
+------------------+
| hex(-1)          |
```

```
+------------------+
| FFFFFFFFFFFFFFF |
+------------------+
```

keywords

```
HEX
```

UNHEX

unhex

description

Syntax

```
VARCHAR unhex(VARCHAR str)
```

Enter a string, if the length of the string is 0 or an odd number, an empty string is returned; If the string contains characters other than `[0-9], [a-f], [A-F]`, an empty string is returned; In other cases, every two characters are a group of characters converted into hexadecimal, and then spliced into a string for output.

example

```
mysql> select unhex('@');
+------------+
| unhex('@') |
+------------+
|            |
+------------+

mysql> select unhex('41');
+-------------+
| unhex('41') |
+-------------+
| A           |
+-------------+

mysql> select unhex('4142');
+---------------+
| unhex('4142') |
+---------------+
| AB            |
+---------------+
```

keywords

```
UNHEX
```

ELT

elt

Description

Syntax

```
VARCHAR elt(INT, VARCHAR,...)
```

Returns the string at specified index. Returns NULL if there is no string at specified index.

example

```
mysql> select elt(1, 'aaa', 'bbb');
+----------------------+
| elt(1, 'aaa', 'bbb') |
+----------------------+
| aaa                  |
+----------------------+
mysql> select elt(2, 'aaa', 'bbb');
+----------------------+
| elt(2, 'aaa', 'bbb')  |
+----------------------+
| bbb                   |
+----------------------+
mysql> select elt(0, 'aaa', 'bbb');
+----------------------+
| elt(0, 'aaa', 'bbb') |
+----------------------+
| NULL                 |
+----------------------+
mysql> select elt(2, 'aaa', 'bbb');
+----------------------+
| elt(3, 'aaa', 'bbb')  |
+----------------------+
| NULL                  |
+----------------------+
```

keywords

```
ELT
```

INSTR

instr

Description

Syntax

```
INSTR (VARCHAR STR, VARCHAR substrate)
```

Returns the location where substr first appeared in str (counting from 1). If substr does not appear in str, return 0.

example

```
mysql> select instr("abc", "b");
+-------------------+
| instr('abc', 'b') |
+-------------------+
|                 2 |
+-------------------+

mysql> select instr("abc", "d");
+-------------------+
| instr('abc', 'd') |
+-------------------+
|                 0 |
+-------------------+
```

keywords

```
INSTR
```

LOCATE

locate

Description

Syntax

```
INT LOCATION (VARCHAR substrate, VARCHAR str [, INT pos]]
```

Returns where substr appears in str (counting from 1). If the third parameter POS is specified, the position where substr appears is found from the string where STR starts with POS subscript. If not found, return 0

example

```
mysql> SELECT LOCATE('bar', 'foobarbar');
+----------------------------+
| locate('bar', 'foobarbar') |
+----------------------------+
|                          4 |
+----------------------------+

mysql> SELECT LOCATE('xbar', 'foobar');
+--------------------------+
| locate('xbar', 'foobar') |
+--------------------------+
|                        0 |
+--------------------------+

mysql> SELECT LOCATE('bar', 'foobarbar', 5);
+-------------------------------+
| locate('bar', 'foobarbar', 5) |
+-------------------------------+
|                             7 |
+-------------------------------+
```

keywords

```
LOCATE
```

FIELD

Description

Returns the position of the first occurrence of <expr> in the list of values <param> [, ...].
If <expr> is not found, the function returns 0. This function is commonly used in ORDER BY to implement custom sorting.

Syntax

```
FIELD(<expr>, <param> [, ...])
```

Parameters

Parameter	Description
<expr>	The value to be searched in the list of parameters.
<param>	A sequence of values to compare against <expr>.

Return Value

- Returns the position (1-based index) of <expr> in the list of <param> values.
- If <expr> is not found, returns 0.
- If <expr> is NULL, returns 0.

Examples

```
SELECT FIELD(2, 3, 1, 2, 5);
```

```
+----------------------+
| FIELD(2, 3, 1, 2, 5) |
+----------------------+
|                    3 |
+----------------------+
```

```
SELECT k1, k7 FROM baseall WHERE k1 IN (1,2,3) ORDER BY FIELD(k1, 2, 1, 3);
```

```
+------+------------+
| k1   | k7         |
+------+------------+
|    2 | wangyu14   |
|    1 | wangjing04 |
|    3 | yuanyuan06 |
+------+------------+
```

```
SELECT class_name FROM class_test ORDER BY FIELD(class_name, 'Suzi', 'Ben', 'Henry');
```

```
+------------+
| class_name |
+------------+
| Suzi       |
| Suzi       |
| Ben        |
| Ben        |
| Henry      |
| Henry      |
+------------+
```

FIND_IN_SET

find_in_set

description

Syntax

```
INT find_in_set(VARCHAR str, VARCHAR strlist)
```

“NOT found in set (VARCHAR str., VARCHAR strlist)”

Return to the location where the str first appears in strlist (counting from 1). Strlist is a comma-separated string. If not, return 0. Any parameter is NULL, returning NULL.

example

```
mysql> select find_in_set("b", "a,b,c");
+--------------------------+
| find_in_set('b', 'a,b,c') |
+--------------------------+
|                        2 |
+--------------------------+
```

keywords

```
FIND_IN_SET,FIND,IN,SET
```

REPLACE

replace

description

Syntax

```
VARCHAR REPLACE (VARCHAR str, VARCHAR old, VARCHAR new)
```

replace all old substring with new substring in str

example

```
mysql> select replace("http://www.baidu.com:9090", "9090", "");
+-----------------------------------------------------+
| replace('http://www.baidu.com:9090', '9090', '') |
+-----------------------------------------------------+
| http://www.baidu.com:                               |
+-----------------------------------------------------+
```

keywords

```
REPLACE
```

LEFT

left

Description

Syntax

```
VARCHAR left (VARCHAR str, INT len)
```

It returns the left part of a string of specified length, length is char length not the byte size. Another alias for this function is strleft.

example

```
mysql> select left("Hello doris",5);
+-----------------------+
| left('Hello doris', 5) |
+-----------------------+
| Hello                 |
+-----------------------+
```

keywords

```
LEFT
```

RIGHT

right

Description

Syntax

```
VARCHAR RIGHT (VARCHAR str, INT len)
```

It returns the right part of a string of specified length, length is char length not the byte size. Another alias for this function is strright.

example

```
mysql> select right("Hello doris",5);
+-------------------------+
| right('Hello doris', 5) |
+-------------------------+
| doris                   |
+-------------------------+
```

keywords

```
RIGHT
```

STRLEFT

strleft

Description

Syntax

```
VARCHAR STRLEFT (VARCHAR str, INT len)
```

It returns the left part of a string of specified length, length is char length not the byte size. Another alias for this function is left.

example

```
mysql> select strleft("Hello doris",5);
+-----------------------+
| strleft('Hello doris', 5) |
+-----------------------+
| Hello                 |
+-----------------------+
```

keywords

```
STRLEFT
```

STRRIGHT

strright

Description

Syntax

```
VARCHAR strright (VARCHAR str, INT len)
```

It returns the right part of a string of specified length, length is char length not the byte size. Another alias for this function is right.

example

```
mysql> select strright("Hello doris",5);
+-------------------------+
| strright('Hello doris', 5) |
+-------------------------+
| doris                   |
+-------------------------+
```

keywords

```
STRRIGHT
```

SPLIT_PART

split_part

Description

Syntax

```
VARCHAR split_part(VARCHAR content, VARCHAR delimiter, INT field)
```

Returns the specified partition by splitting the string according to the delimiter. If field is positive, splitting and counting from the beginning of content, otherwise from the ending.

`delimiter` and `field` parameter should be constant.

example

```
mysql> select split_part("hello world", " ", 1);
+----------------------------------+
| split_part('hello world', ' ', 1) |
+----------------------------------+
| hello                            |
+----------------------------------+


mysql> select split_part("hello world", " ", 2);
+----------------------------------+
| split_part('hello world', ' ', 2) |
+----------------------------------+
| world                             |
+----------------------------------+

mysql> select split_part("2019年7月8号", "月", 1);
+---------------------------------------+
| split_part('2019年7月8号', '月', 1)     |
+---------------------------------------+
| 2019年7                                |
+---------------------------------------+

mysql> select split_part("abca", "a", 1);
+---------------------------+
| split_part('abca', 'a', 1) |
+---------------------------+
|                           |
+---------------------------+

mysql> select split_part("prefix_string", "_", -1);
+-------------------------------------+
| split_part('prefix_string', '_', -1) |
+-------------------------------------+
| string                               |
+-------------------------------------+


mysql> select split_part("prefix_string", "_", -2);
```

```
+-------------------------------------+
| split_part('prefix_string', '_', -2) |
+-------------------------------------+
| prefix                              |
+-------------------------------------+

mysql> select split_part("abc##123###234", "##", -1);
+---------------------------------------+
| split_part('abc##123###234', '##', -1) |
+---------------------------------------+
| 234                                   |
+---------------------------------------+

mysql> select split_part("abc##123###234", "##", -2);
+---------------------------------------+
| split_part('abc##123###234', '##', -2) |
+---------------------------------------+
| 123#                                  |
+---------------------------------------+
```

keywords

```
SPLIT_PART,SPLIT,PART
```

SPLIT_BY_STRING

split_by_string

description

Syntax

```
ARRAY<STRING> split_by_string(STRING s, STRING separator)
```

Splits a string into substrings separated by a string. It uses a constant string separator of multiple characters as the separator. If the string separator is empty, it will split the string s into an array of single characters.

Arguments

`separator` — The separator. Type: `String`

`s` — The string to split. Type: `String`

Returned value(s)

Returns an array of selected substrings. Empty substrings may be selected when:

A non-empty separator occurs at the beginning or end of the string;

There are multiple consecutive separators;

The original string s is empty.

Type: `Array(String)`

example

```
select split_by_string('a1b1c1d','1');
+---------------------------------+
| split_by_string('a1b1c1d', '1') |
+---------------------------------+
| ['a', 'b', 'c', 'd']            |
+---------------------------------+

select split_by_string(',,a,b,c,',',');
+----------------------------------+
| split_by_string(',,a,b,c,', ',') |
+----------------------------------+
| ['', '', 'a', 'b', 'c', '']      |
+----------------------------------+

SELECT split_by_string(NULL,',');
+---------------------------+
| split_by_string(NULL, ',') |
+---------------------------+
```

```
| NULL                       |
+----------------------------+

select split_by_string('a,b,c,abcde',',');
+-----------------------------------+
| split_by_string('a,b,c,abcde', ',') |
+-----------------------------------+
| ['a', 'b', 'c', 'abcde']            |
+-----------------------------------+

select split_by_string('1,,2,3,,4,5,,abcde', ',,');
+--------------------------------------------+
| split_by_string('1,,2,3,,4,5,,abcde', ',,') |
+--------------------------------------------+
| ['1', '2,3', '4,5', 'abcde']               |
+--------------------------------------------+

select split_by_string(',,,,',',,');
+-------------------------------+
| split_by_string(',,,,', ',,') |
+-------------------------------+
| ['', '', '']                  |
+-------------------------------+

select split_by_string(',,a,,b,,c,,',',,');
+-------------------------------------+
| split_by_string(',,a,,b,,c,,', ',,') |
+-------------------------------------+
| ['', 'a', 'b', 'c', '']             |
+-------------------------------------+
```

keywords

SPLIT_BY_STRING,SPLIT

SUBSTRING_INDEX

substring_index

Name

> **Tips**
>
> This function is supported since the Apache Doris 1.2. version

SUBSTRING_INDEX

description

Syntax

```
VARCHAR substring_index(VARCHAR content, VARCHAR delimiter, INT field)
```

Split `content` to two parts at position where the `field`s of `delimiter` stays, return one of them according to below rules: if `field` is positive, return the left part; else if `field` is negative, return the right part; if `field` is zero, return an empty string when `content` is not null, else will return null.

- `delimiter` is case sensitive and multi-byte safe.
- `delimiter` and `field` parameter should be constant.

example

```
mysql> select substring_index("hello world", " ", 1);
+---------------------------------------+
| substring_index("hello world", " ", 1) |
+---------------------------------------+
| hello                                 |
+---------------------------------------+
mysql> select substring_index("hello world", " ", 2);
+---------------------------------------+
| substring_index("hello world", " ", 2) |
+---------------------------------------+
| hello world                           |
+---------------------------------------+
mysql> select substring_index("hello world", " ", -1);
+----------------------------------------+
| substring_index("hello world", " ", -1) |
+----------------------------------------+
| world                                  |
+----------------------------------------+
mysql> select substring_index("hello world", " ", -2);
+----------------------------------------+
| substring_index("hello world", " ", -2) |
+----------------------------------------+
| hello world                            |
```

```
+---------------------------------------+
mysql> select substring_index("hello world", " ", -3);
+---------------------------------------+
| substring_index("hello world", " ", -3) |
+---------------------------------------+
| hello world                           |
+---------------------------------------+
mysql> select substring_index("hello world", " ", 0);
+---------------------------------------+
| substring_index("hello world", " ", 0) |
+---------------------------------------+
|                                       |
+---------------------------------------+
```

keywords

```
SUBSTRING_INDEX, SUBSTRING
```

MONEY_FORMAT

money_format

Description

Syntax

```
VARCHAR money format (Number)
```

The number is output in currency format, the integer part is separated by commas every three bits, and the decimal part is reserved for two bits.

example

```
mysql> select money_format(17014116);
+------------------------+
| money_format(17014116) |
+------------------------+
| 17,014,116.00          |
+------------------------+

mysql> select money_format(1123.456);
+------------------------+
| money_format(1123.456) |
+------------------------+
| 1,123.46               |
+------------------------+

mysql> select money_format(1123.4);
+----------------------+
| money_format(1123.4) |
+----------------------+
| 1,123.40             |
+----------------------+
```

keywords

```
MONEY_FORMAT,MONEY,FORMAT
```

PARSE_URL

parse_url

description

Syntax

```
VARCHAR parse_url(VARCHAR url, VARCHAR name)
```

From the URL, the field corresponding to name is resolved. The name options are as follows: ‘PROTOCOL’, ‘HOST’, ‘PATH’, ‘REF’, ‘AUTHORITY’, ‘FILE’, ‘USERINFO’, ‘PORT’, ‘QUERY’, and the result is returned.

example

```
mysql> SELECT parse_url ('https://doris.apache.org/', 'HOST');
+-------------------------------------------------+
| parse_url('https://doris.apache.org/', 'HOST') |
+-------------------------------------------------+
| doris.apache.org                                |
+-------------------------------------------------+
```

If you want to get parameter in QUERY, you can use extract_url_parameter.

keywords

```
PARSE URL
```

CONVERT_TO

Description

Converts the character encoding of a column to a specified target character set. This function is typically used in the ORDER BY clause to ensure that columns containing Chinese characters are sorted according to their pinyin order. Currently, only conversion to `'gbk'` is supported.

Syntax

```
CONVERT_TO(<column>, <character>)
```

Parameters

Parameters	Description
`<column>`	The VARCHAR column whose encoding is to be converted.
`<character>`	The target character set. Currently, only `'gbk'` is supported.

Return Value

Returns a VARCHAR value with the converted encoding, allowing proper pinyin-based ordering when used in the ORDER BY clause.

Examples

```
SELECT * FROM class_test ORDER BY class_name;
```

```
+----------+------------+-------------+
| class_id | class_name | student_ids |
+----------+------------+-------------+
|        6 | asd        | [6]         |
|        7 | qwe        | [7]         |
|        8 | z          | [8]         |
|        2 | 哈         | [2]         |
|        3 | 哦         | [3]         |
|        1 | 啊         | [1]         |
|        4 | 张         | [4]         |
|        5 | 我         | [5]         |
+----------+------------+-------------+
```

```
SELECT * FROM class_test ORDER BY CONVERT_TO(class_name, 'gbk');
```

```
+----------+------------+-------------+
| class_id | class_name | student_ids |
+----------+------------+-------------+
```

```
|        6 | asd        | [6]         |
|        7 | qwe        | [7]         |
|        8 | z          | [8]         |
|        1 | 啊         | [1]         |
|        2 | 哈         | [2]         |
|        3 | 哦         | [3]         |
|        5 | 我         | [5]         |
|        4 | 张         | [4]         |
+----------+------------+-------------+
```

EXTRACT_URL_PARAMETER

extract_url_parameter

description

Syntax

```
VARCHAR extract_url_parameter(VARCHAR url, VARCHAR name)
```

Returns the value of the “name” parameter in the URL, if present. Otherwise an empty string. If there are many parameters with this name, the first occurrence is returned. This function works assuming that the parameter name is encoded in the URL exactly as it was in the passed parameter.

```
mysql> SELECT extract_url_parameter ("http://doris.apache.org?k1=aa&k2=bb&test=cc#999", "k2");
+-----------------------------------------------------------------------------+
| extract_url_parameter('http://doris.apache.org?k1=aa&k2=bb&test=cc#999', 'k2') |
+-----------------------------------------------------------------------------+
| bb                                                                          |
+-----------------------------------------------------------------------------+
```

If you want to get other part of URL, you can use parse_url.

keywords

```
EXTRACT URL PARAMETER
```

UUID

uuid

uuid

description

Syntax

```
VARCHAR uuid()
```

return a random uuid string

example

```
mysql> select uuid();
+--------------------------------------+
| uuid()                               |
+--------------------------------------+
| 29077778-fc5e-4603-8368-6b5f8fd55c24 |
+--------------------------------------+
```

keywords

```
UUID
```

SPACE

space

Description

Syntax

```
VARCHAR space(Int num)
```

Returns a string consisting of num spaces.

example

```
mysql> select length(space(10));
+-------------------+
| length(space(10)) |
+-------------------+
|                10 |
+-------------------+
1 row in set (0.01 sec)

mysql> select length(space(-10));
+--------------------+
| length(space(-10)) |
+--------------------+
|                  0 |
+--------------------+
1 row in set (0.02 sec)
```

keywords

```
space
```

ESQUERY

Description

Use the `ESQUERY(<field>, <query_dsl>)` function to push down queries that cannot be expressed in SQL to Elasticsearch for filtering. The first parameter, `<field>`, is used to associate indexes, while the second parameter, `<query_dsl>`, is a JSON expression representing the basic Elasticsearch Query DSL. The JSON must be enclosed in curly brackets `{}` and contain exactly one root key (e.g., `match_phrase`, `geo_shape`, `bool`).

Syntax

```
ESQUERY(<field>, <query_dsl>)
```

Parameters

Parameter	Description
`<field>`	The field to be queried; used to associate indexes.
`<query_dsl>`	A JSON expression representing the Elasticsearch Query DSL. It must be enclosed in `{}` and contain exactly one root key (e.g., `match_phrase`, `geo_shape`, `bool`).

Return Value

Returns a boolean value indicating whether the document matches the provided Elasticsearch query DSL.

Examples

```
-- match_phrase SQL:
SELECT * FROM es_table
WHERE ESQUERY(k4, '{
    "match_phrase": {
       "k4": "doris on es"
    }
}');
```

```
-- geo_shape SQL:
SELECT * FROM es_table
WHERE ESQUERY(k4, '{
  "geo_shape": {
     "location": {
        "shape": {
           "type": "envelope",
           "coordinates": [
              [13, 53],
              [14, 52]
           ]
        },
        "relation": "within"
```

```
    }
  }
}');
```

MULTI_SEARCH_ALL_POSITIONS

multi_search_all_positions

Description

Syntax

```
ARRAY<INT> multi_search_all_positions(VARCHAR haystack, ARRAY<VARCHAR> needles)
```

Returns an `ARRAY` where the `i`-th element is the position of the `i`-th element in `needles`(i.e. `needle`)'s **first** occurrence in the string `haystack`. Positions are counted from 1, with 0 meaning the element was not found. **Case-sensitive**.

example

```
mysql> select multi_search_all_positions('Hello, World!', ['hello', '!', 'world']);
+-----------------------------------------------------------------------+
| multi_search_all_positions('Hello, World!', ['hello', '!', 'world']) |
+-----------------------------------------------------------------------+
| [0,13,0]                                                              |
+-----------------------------------------------------------------------+

select multi_search_all_positions("Hello, World!", ['hello', '!', 'world', 'Hello', 'World']);
+-----------------------------------------------------------------------------------------------+
| multi_search_all_positions('Hello, World!', ARRAY('hello', '!', 'world', 'Hello', 'World')) |
+-----------------------------------------------------------------------------------------------+
| [0, 13, 0, 1, 8]                                                                              |
+-----------------------------------------------------------------------------------------------+
```

keywords

```
MULTI_SEARCH,SEARCH,POSITIONS
```

MULTI_MATCH_ANY

multi_match_any

Description

Syntax

```
TINYINT multi_match_any(VARCHAR haystack, ARRAY<VARCHAR> patterns)
```

Checks whether the string `haystack` matches the regular expressions `patterns` in re2 syntax. returns 0 if none of the regular expressions are matched and 1 if any of the patterns matches.

example

```
mysql> select multi_match_any('Hello, World!', ['hello', '!', 'world']);
+-----------------------------------------------------------+
| multi_match_any('Hello, World!', ['hello', '!', 'world']) |
+-----------------------------------------------------------+
| 1                                                         |
+-----------------------------------------------------------+

mysql> select multi_match_any('abc', ['A', 'bcd']);
+--------------------------------------+
| multi_match_any('abc', ['A', 'bcd']) |
+--------------------------------------+
| 0                                    |
+--------------------------------------+
```

keywords

```
MULTI_MATCH,MATCH,ANY
```

MASK

mask

description

syntax

```
VARCHAR mask(VARCHAR str[, VARCHAR upper[, VARCHAR lower[, VARCHAR number]]])
```

Returns a masked version of str . By default, upper case letters are converted to “X”, lower case letters are converted to “x” and numbers are converted to “n”. For example mask(“abcd-EFGH-8765-4321”) results in xxxx-XXXX-nnnn-nnnn. You can override the characters used in the mask by supplying additional arguments: the second argument controls the mask character for upper case letters, the third argument for lower case letters and the fourth argument for numbers. For example, mask(“abcd-EFGH-8765-4321”, “U”, “l”, “#”) results in llll-UUUU-####-####.

example

```
// table test
+-----------+
| name      |
+-----------+
| abc123EFG |
| NULL      |
| 456AbCdEf |
+-----------+

mysql> select mask(name) from test;
+--------------+
| mask(`name`) |
+--------------+
| xxxnnnXXX    |
| NULL         |
| nnnXxXxXx    |
+--------------+

mysql> select mask(name, '*', '#', '$') from test;
+-----------------------------+
| mask(`name`, '*', '#', '$') |
+-----------------------------+
| ###$$$***                   |
| NULL                        |
| $$$*#*#*#                   |
+-----------------------------+
```

keywords

```
mask
```

MASK_FIRST_N

mask_first_n

description

syntax

```
VARCHAR mask_first_n(VARCHAR str[, INT n])
```

Returns a masked version of str with the first n values masked. Upper case letters are converted to “X”, lower case letters are converted to “x” and numbers are converted to “n”. For example, mask_first_n(“1234-5678-8765-4321”, 4) results in nnnn-5678-8765-4321.

example

```
// table test
+-----------+
| name      |
+-----------+
| abc123EFG |
| NULL      |
| 456AbCdEf |
+-----------+

mysql> select mask_first_n(name, 5) from test;
+-------------------------+
| mask_first_n(`name`, 5) |
+-------------------------+
| xxxnn3EFG               |
| NULL                    |
| nnnXxCdEf               |
+-------------------------+
```

keywords

```
mask_first_n
```

MASK_LAST_N

mask_last_n

description

syntax

```
VARCHAR mask_last_n(VARCHAR str[, INT n])
```

Returns a masked version of str with the last n values masked. Upper case letters are converted to “X”, lower case letters are converted to “x” and numbers are converted to “n”. For example, mask_last_n(“1234-5678-8765-4321”, 4) results in 1234-5678-8765-nnnn.

example

```
// table test
+-----------+
| name      |
+-----------+
| abc123EFG |
| NULL      |
| 456AbCdEf |
+-----------+

mysql> select mask_last_n(name, 5) from test;
+------------------------+
| mask_last_n(`name`, 5) |
+------------------------+
| abc1nnXXX              |
| NULL                   |
| 456AxXxXx              |
+------------------------+
```

keywords

```
mask_last_n
```

LIKE

like

description

syntax

```
BOOLEAN like(VARCHAR str, VARCHAR pattern)
```

Perform fuzzy matching on the string str, return true if it matches, and false if it doesn't match.

like match/fuzzy match, will be used in combination with % and _.

the percent sign ('%') represents zero, one, or more characters.

the underscore ('_') represents a single character.

```
'a'   // Precise matching, the same effect as `=`
'%a'  // data ending with a
'a%'  // data starting with a
'%a%' // data containing a
'_a_' // three digits and the middle letter is a
'_a'  // two digits and the ending letter is a
'a_'  // two digits and the initial letter is a
'a__b'  // four digits, starting letter is a and ending letter is b
```

example

```
// table test
+-------+
| k1    |
+-------+
| b     |
| bb    |
| bab   |
| a     |
+-------+

// Return the data containing a in the k1 string
mysql> select k1 from test where k1 like '%a%';
+-------+
| k1    |
+-------+
| a     |
| bab   |
+-------+

// Return the data equal to a in the k1 string
mysql> select k1 from test where k1 like 'a';
+-------+
| k1    |
+-------+
```

```
| a     |
+-------+
```

keywords

```
LIKE
```

NOT LIKE

not like

description

syntax

```
BOOLEAN not like(VARCHAR str, VARCHAR pattern)
```

Perform fuzzy matching on the string str, return false if it matches, and return true if it doesn't match.

like match/fuzzy match, will be used in combination with % and _.

the percent sign ('%') represents zero, one, or more characters.

the underscore ('_') represents a single character.

```
'a'   // Precise matching, the same effect as `=`
'%a'  // data ending with a
'a%'  // data starting with a
'%a%' // data containing a
'_a_' // three digits and the middle letter is a
'_a'  // two digits and the ending letter is a
'a_'  // two digits and the initial letter is a
'a__b'  // four digits, starting letter is a and ending letter is b
```

example

```
// table test
+-------+
| k1    |
+-------+
| b     |
| bb    |
| bab   |
| a     |
+-------+

// Return data that does not contain a in the k1 string
mysql> select k1 from test where k1 not like '%a%';
+-------+
| k1    |
+-------+
| b     |
| bb    |
+-------+

// Return the data that is not equal to a in the k1 string
mysql> select k1 from test where k1 not like 'a';
+-------+
| k1    |
+-------+
```

```
| b     |
| bb    |
| bab   |
+-------+
```

keywords

```
LIKE, NOT, NOT LIKE
```

REGEXP

Description

Perform regular matching on the string str, return true if it matches, return false or not. Pattern is a regular expression.

- Character set matching requires the use of Unicode standard character classes. For example, to match Chinese, use `\p{Han}`.

Syntax

```
REGEXP(<str>, <pattern>)
```

Parameters

Parameter	Description
<str>	The column need to do regular matching.
<pattern>	Target pattern.

Return Value

A `BOOLEAN` value indicating whether the match was successful

Example

Lets prepare some data.

```
CREATE TABLE test ( k1 VARCHAR(255) ) properties("replication_num"="1")

INSERT INTO test (k1) VALUES ('billie eillish'), ('It\'s ok'), ('billie jean'), ('hello world');
```

Do `REGEXP` now

```
--- Find all data starting with 'billie' in the k1 field
SELECT k1 FROM test WHERE k1 REGEXP '^billie'
--------------

+----------------+
| k1             |
+----------------+
| billie eillish |
| billie jean    |
+----------------+
2 rows in set (0.02 sec)

--- Find all data ending with 'ok' in the k1 field:
```

```
SELECT k1 FROM test WHERE k1 REGEXP 'ok$'
--------------

+---------+
| k1      |
+---------+
| It's ok |
+---------+
1 row in set (0.03 sec)
```

Example for Chinese character

```
mysql> select regexp('这是一段中文This is a passage in English 1234567', '\\p{Han}');
+-------------------------------------------------------------------------------+
| ('这是一段中文This is a passage in English 1234567' regexp '\p{Han}')         |
+-------------------------------------------------------------------------------+
|                                                                             1 |
+-------------------------------------------------------------------------------+
```

REGEXP_EXTRACT

Description

The string STR is matched regularly and the POS matching part which conforms to pattern is extracted. Patterns need to match exactly some part of the STR to return to the matching part of the pattern.

- If there is no match, return an empty string.
- Character set matching requires the use of Unicode standard character classes. For example, to match Chinese, use `\p{Han}`.

Syntax

```
REGEXP_EXTRACT(<str>, <pattern>, <pos>)
```

Parameters

Parameter	Description
`<str>`	The column need to do regular matching.
`<pattern>`	Target pattern.
`<pos>`	The parameter used to specify the position in the string from which to start searching for the regular expression match. It is an integer value representing the character position in the string (starting from 1). `pos` must be specified.

Return Value

Matching part of the partern. It is `Varchar` type.

Example

```
mysql> SELECT regexp_extract('AbCdE', '([[:lower:]]+)C([[:lower:]]+)', 1);
+-----------------------------------------------------------+
| regexp_extract('AbCdE', '([[:lower:]]+)C([[:lower:]]+)', 1) |
+-----------------------------------------------------------+
| b                                                         |
+-----------------------------------------------------------+

mysql> SELECT regexp_extract('AbCdE', '([[:lower:]]+)C([[:lower:]]+)', 2);
+-----------------------------------------------------------+
| regexp_extract('AbCdE', '([[:lower:]]+)C([[:lower:]]+)', 2) |
+-----------------------------------------------------------+
| d                                                         |
+-----------------------------------------------------------+

mysql> select regexp_extract('这是一段中文This is a passage in English 1234567', '(\\p{Han}+)(.+)', 2);
```

```
+-----------------------------------------------------------------------------------------------+
| regexp_extract('这是一段中文This is a passage in English 1234567', '(\p{Han}+)(.+)', 2)       |
+-----------------------------------------------------------------------------------------------+
| This is a passage in English 1234567                                                          |
+-----------------------------------------------------------------------------------------------+
```

REGEXP_EXTRACT_ALL

Description

Regularly matches a string `str` and extracts the first sub-pattern matching part of `pattern`. The pattern needs to exactly match a part of `str` in order to return an array of strings for the part of the pattern that needs to be matched. If there is no match or the pattern has no sub-pattern, the empty string is returned.

- Character set matching requires the use of Unicode standard character classes. For example, to match Chinese, use `\p{Han}`.

Syntax

```
REGEXP_EXTRACT_ALL(<str>, <pattern>)
```

Parameters

Parameter	Description
`<str>`	The column need to do regular matching.
`<pattern>`	Target pattern.

Return value

Value after extraction. It is `String` type.

Example

```
mysql> SELECT regexp_extract_all('AbCdE', '([[:lower:]]+)C([[:lower:]]+)');
+-----------------------------------------------------------+
| regexp_extract_all('AbCdE', '([[:lower:]]+)C([[:lower:]]+)') |
+-----------------------------------------------------------+
| ['b']                                                     |
+-----------------------------------------------------------+

mysql> SELECT regexp_extract_all('AbCdEfCg', '([[:lower:]]+)C([[:lower:]]+)');
+-----------------------------------------------------------+
| regexp_extract_all('AbCdEfCg', '([[:lower:]]+)C([[:lower:]]+)') |
+-----------------------------------------------------------+
| ['b','f']                                                 |
+-----------------------------------------------------------+

mysql> SELECT regexp_extract_all('abc=111, def=222, ghi=333','("[^"]+"|\\w+)=("[^"]+"|\\w+)');
+-------------------------------------------------------------------------------+
| regexp_extract_all('abc=111, def=222, ghi=333', '("[^"]+"|\w+)=("[^"]+"|\w+)') |
+-------------------------------------------------------------------------------+
| ['abc','def','ghi']                                                           |
+-------------------------------------------------------------------------------+
```

```
mysql> select regexp_extract_all('这是一段中文This is a passage in English 1234567', '(\\p{Han}+)(.+)');
+-----------------------------------------------------------------------------------------------+
| regexp_extract_all('这是一段中文This is a passage in English 1234567', '(\p{Han}+)(.+)')       |
+-----------------------------------------------------------------------------------------------+
| ['这是一段中文']                                                                              |
+-----------------------------------------------------------------------------------------------+
```

REGEXP_REPLACE

Description

Regular matching of STR strings, replacing the part hitting pattern with a new string.

- Character set matching requires the use of Unicode standard character classes. For example, to match Chinese, use `\p{Han}`.

Syntax

```
REGEXP_REPLACE(<str>, <pattern>, <repl>)
```

Parameters

Parameter	Description
`<str>`	The column need to do regular matching.
`<pattern>`	Target pattern.
`<repl>`	The string to replace the matched pattern.

Return Value

Result after doing replacement. It is `Varchar` type.

Example

```
mysql> SELECT regexp_replace('a b c', ' ', '-');
+-----------------------------------+
| regexp_replace('a b c', ' ', '-') |
+-----------------------------------+
| a-b-c                             |
+-----------------------------------+

mysql> SELECT regexp_replace('a b c', '(b)', '<\\1>');
+----------------------------------------+
| regexp_replace('a b c', '(b)', '<\1>') |
+----------------------------------------+
| a <b> c                                |
+----------------------------------------+

mysql> select regexp_replace('这是一段中文This is a passage in English 1234567', '\\p{Han}+', '123');
+-------------------------------------------------------------------------------------------+
| regexp_replace('这是一段中文This is a passage in English 1234567', '\p{Han}+', '123')      |
+-------------------------------------------------------------------------------------------+
| 123This is a passage in English 1234567                                                   |
+-------------------------------------------------------------------------------------------+
```

REGEXP_REPLACE_ONE

描述

对字符串 STR 进行正则匹配, 将命中 pattern 的部分使用 repl 来进行替换，仅替换第一个匹配项。

- 字符集匹配需要使用 Unicode 标准字符类型。例如，匹配中文请使用 \p{Han}。

语法

```
REGEXP_REPLACE_ONE(<str>, <pattern>, <repl>)
```

参数

参数	描述
<str>	需要进行正则匹配的列。
<pattern>	目标模式。
<repl>	用于替换匹配模式的字符串。

返回值

替换之后的结果，类型是 Varchar。

举例

```
mysql> SELECT regexp_replace_one('a b c', ' ', '-');
+-----------------------------------+
| regexp_replace_one('a b c', ' ', '-') |
+-----------------------------------+
| a-b c                             |
+-----------------------------------+

mysql> SELECT regexp_replace_one('a b b', '(b)', '<\\1>');
+---------------------------------------+
| regexp_replace_one('a b b', '(b)', '<\1>') |
+---------------------------------------+
| a <b> b                               |
+---------------------------------------+

mysql> select regexp_replace_one('这是一段中文This is a passage in English 1234567', '\\p{Han}', '123');
+-----------------------------------------------------------------------------------------------+
| regexp_replace_one('这是一段中文This is a passage in English 1234567', '\p{Han}', '123')       |
+-----------------------------------------------------------------------------------------------+
| 123是一段中文This is a passage in English 1234567                                             |
+-----------------------------------------------------------------------------------------------+
```

STRUCT

struct()

> **Tips**
>
> This function is supported since the Apache Doris 2.0 version

struct()

description

Syntax

```
STRUCT<T1, T2, T3, ...> struct(T1, T2, T3, ...)
```

construct an struct with variadic elements and return it, Tn could be column or literal

example

```
mysql> select struct(1, 'a', "abc");
+-----------------------+
| struct(1, 'a', 'abc') |
+-----------------------+
| {1, 'a', 'abc'}       |
+-----------------------+
1 row in set (0.03 sec)

mysql> select struct(null, 1, null);
+-----------------------+
| struct(NULL, 1, NULL) |
+-----------------------+
| {NULL, 1, NULL}       |
+-----------------------+
1 row in set (0.02 sec)

mysql> select struct(cast('2023-03-16' as datetime));
+----------------------------------------+
| struct(CAST('2023-03-16' AS DATETIME)) |
+----------------------------------------+
| {2023-03-16 00:00:00}                  |
+----------------------------------------+
1 row in set (0.01 sec)

mysql> select struct(k1, k2, null) from test_tb;
+--------------------------+
| struct(`k1`, `k2`, NULL) |
+--------------------------+
| {1, 'a', NULL}           |
+--------------------------+
1 row in set (0.04 sec)
```

keywords

STRUCT,CONSTRUCTOR

NAMED_STRUCT

named_struct

named_struct

description

Syntax

```
STRUCT<T1, T2, T3, ...> named_struct({VARCHAR, T1}, {VARCHAR, T2}, ...)
```

Construct a struct with the given field names and values.

The number of parameters must be non zero and even. With odd digits being the name of the field and could be string literal, with even digits being the value of the field and could be column or literal.

example

```
mysql> select named_struct('f1', 1, 'f2', 'a', 'f3', "abc");
+-----------------------------------------------+
| named_struct('f1', 1, 'f2', 'a', 'f3', 'abc') |
+-----------------------------------------------+
| {1, 'a', 'abc'}                               |
+-----------------------------------------------+
1 row in set (0.01 sec)

mysql> select named_struct('a', null, 'b', "v");
+----------------------------------+
| named_struct('a', NULL, 'b', 'v') |
+----------------------------------+
| {NULL, 'v'}                      |
+----------------------------------+
1 row in set (0.01 sec)

mysql> select named_struct('f1', k1, 'f2', k2, 'f3', null) from test_tb;
+-------------------------------------------------+
| named_struct('f1', `k1`, 'f2', `k2`, 'f3', NULL) |
+-------------------------------------------------+
| {1, 'a', NULL}                                  |
+-------------------------------------------------+
1 row in set (0.02 sec)
```

keywords

NAMED, STRUCT, NAMED_STRUCT

STRUCT_ELEMENT

struct_element

struct_element

description

Function allows getting a field from a struct.

Syntax

```
struct_element(struct, n/s)
```

Arguments

```
struct - The input struct column. If null, null will be returned.
n - The position of field, starting from 1, only supports constants.
s - The name of field, only supports constants.
```

Returned value

Returns the specified field column, of any type.

example

```
mysql> select struct_element(named_struct('f1', 1, 'f2', 'a'), 'f2');
+-------------------------------------------------------+
| struct_element(named_struct('f1', 1, 'f2', 'a'), 'f2') |
+-------------------------------------------------------+
| a                                                     |
+-------------------------------------------------------+
1 row in set (0.03 sec)

mysql> select struct_element(named_struct('f1', 1, 'f2', 'a'), 1);
+-------------------------------------------------------+
| struct_element(named_struct('f1', 1, 'f2', 'a'), 1) |
+-------------------------------------------------------+
|                                                     1 |
+-------------------------------------------------------+
1 row in set (0.02 sec)

mysql> select struct_col, struct_element(struct_col, 'f1') from test_struct;
+-----------------------------------------------+-----------------------------------+
| struct_col                                    | struct_element(`struct_col `, 'f1') |
+-----------------------------------------------+-----------------------------------+
| {1, 2, 3, 4, 5}                               |                                 1 |
| {1, 1000, 10000000, 100000000000, 100000000000} |                                 1 |
| {5, 4, 3, 2, 1}                               |                                 5 |
```

```
| NULL                                          |                               NULL |
| {1, NULL, 3, NULL, 5}                         |                                  1 |
+-----------------------------------------------+------------------------------------+
9 rows in set (0.01 sec)
```

keywords

STRUCT, ELEMENT, STRUCT_ELEMENT

STATE

description

Returns the intermediate result of the aggregation function, which can be used for subsequent aggregation or to obtain the actual calculation result through the merge combiner, or can be directly written into the agg_state type table and saved. The type of the result is agg_state, and the function signature in agg_state is `AGGREGATE_FUNCTION(arg...)`.

Syntax

```
AGGREGATE_FUNCTION_STATE(arg...)
```

example

```
mysql [test]>select avg_merge(t) from (select avg_union(avg_state(1)) as t from d_table group by k1)p;
+----------------+
| avg_merge(`t`) |
+----------------+
|              1 |
+----------------+
```

keywords

AGG_STATE,STATE

MERGE

Description

The aggregated intermediate results are aggregated and calculated to obtain the actual result. The type of the result is consistent with `AGGREGATE_FUNCTION`.

Syntax

```
AGGREGATE_FUNCTION_MERGE(agg_state)
```

Example

```
mysql [test]>select avg_merge(avg_state(1)) from d_table;
+-------------------------+
| avg_merge(avg_state(1)) |
+-------------------------+
|                       1 |
+-------------------------+
```

Keywords

AGG_STATE, MERGE

UNION

description

Aggregate multiple aggregation intermediate results into one. The type of the result is agg_state, and the function signature is consistent with the input parameter.

Syntax

```
AGGREGATE_FUNCTION_UNION(agg_state)
```

example

```
mysql [test]>select avg_merge(t) from (select avg_union(avg_state(1)) as t from d_table group by k1)p;
+----------------+
| avg_merge(`t`) |
+----------------+
|              1 |
+----------------+
```

keywords

AGG_STATE, UNION

COLLECT_SET

COLLECT_SET

COLLECT_SET

description

Syntax

```
ARRAY<T> collect_set(expr[,max_size])
```

Creates an array containing distinct elements from `expr`,with the optional `max_size` parameter limits the size of the resulting array to `max_size` elements. It has an alias `group_uniq_array`.

example

```
mysql> select k1,k2,k3 from collect_set_test order by k1;
+------+------------+-------+
| k1   | k2         | k3    |
+------+------------+-------+
|    1 | 2023-01-01 | hello |
|    2 | 2023-01-01 | NULL  |
|    2 | 2023-01-02 | hello |
|    3 | NULL       | world |
|    3 | 2023-01-02 | hello |
|    4 | 2023-01-02 | doris |
|    4 | 2023-01-03 | sql   |
+------+------------+-------+

mysql> select collect_set(k1),collect_set(k1,2) from collect_set_test;
+-------------------------+-------------------------+
| collect_set(`k1`)       | collect_set(`k1`,2)     |
+-------------------------+-------------------------+
| [4,3,2,1]               | [1,2]                   |
+-------------------------------------------------+

mysql> select k1,collect_set(k2),collect_set(k3,1) from collect_set_test group by k1 order by k1;
+------+-------------------------+-------------------------+
| k1   | collect_set(`k2`)       | collect_set(`k3`,1)     |
+------+-------------------------+-------------------------+
|    1 | [2023-01-01]            | [hello]                 |
|    2 | [2023-01-01,2023-01-02] | [hello]                 |
|    3 | [2023-01-02]            | [world]                 |
|    4 | [2023-01-02,2023-01-03] | [sql]                   |
+------+-------------------------+-------------------------+
```

keywords

COLLECT_SET,GROUP_UNIQ_ARRAY,COLLECT_LIST,ARRAY

MIN

MIN

Description

Syntax

```
MIN(expr)
```

Returns the minimum value of an expr expression

example

```
MySQL > select min(scan_rows) from log_statis group by datetime;
+------------------+
| min(`scan_rows`) |
+------------------+
|                0 |
+------------------+
```

keywords

MIN

STDDEV_SAMP

STDDEV_SAMP

Description

Syntax

```
STDDEV SAMP (expr)
```

Returns the sample standard deviation of the expr expression

example

```
MySQL > select stddev_samp(scan_rows) from log_statis group by datetime;
+--------------------------+
| stddev_samp(`scan_rows`) |
+--------------------------+
|        2.372044195280762 |
+--------------------------+
```

keywords

STDDEV SAMP,STDDEV,SAMP

AVG

AVG

Description

Syntax

```
AVG([DISTINCT] expr)
```

Used to return the average value of the selected field

Optional field DISTINCT parameters can be used to return the weighted average

example

```
mysql> SELECT datetime, AVG(cost_time) FROM log_statis group by datetime;
+---------------------+--------------------+
| datetime            | avg(`cost_time`)   |
+---------------------+--------------------+
| 2019-07-03 21:01:20 | 25.827794561933533 |
+---------------------+--------------------+

mysql> SELECT datetime, AVG(distinct cost_time) FROM log_statis group by datetime;
+---------------------+---------------------------+
| datetime            | avg(DISTINCT `cost_time`) |
+---------------------+---------------------------+
| 2019-07-04 02:23:24 |        20.666666666666668 |
+---------------------+---------------------------+
```

keywords

AVG

AVG_WEIGHTED

AVG_WEIGHTED

Description

Syntax

```
double avg_weighted(x, weight)
```

Calculate the weighted arithmetic mean, which is the sum of the products of all corresponding values and weights, divided the total weight sum. If the sum of all weights equals 0, NaN will be returned.

example

```
mysql> select avg_weighted(k2,k1) from baseall;
+--------------------------+
| avg_weighted(`k2`, `k1`) |
+--------------------------+
|                  495.675 |
+--------------------------+
1 row in set (0.02 sec)
```

keywords

AVG_WEIGHTED

PERCENTILE

Description

Syntax

```
PERCENTILE(expr, DOUBLE p)
```

Calculates the exact percentile for small amounts of data. The specified columns are sorted in descending order and then the exact `p` percentile is taken. The value of `p` is between `0` and `1`. If `p` does not point to an exact position, the linear interpolation of the values adjacent to either side of the pointed position at the position pointed to by `p` is returned. Note that this is not an average of the two numbers.

Parameters: `expr`: required. The value is an integer (`bigint` at most). `p`: required. The const value is `[0.0,1.0]`.

Example

```
MySQL > select `table`, percentile(cost_time,0.99) from log_statis group by `table`;
+----------------------+--------------------------+
| table    |         percentile(`cost_time`, 0.99)|
+----------+--------------------------------------+
| test     |                                54.22 |
+----------+--------------------------------------+

MySQL > select percentile(NULL,0.3) from table1;
+-----------------------+
| percentile(NULL, 0.3) |
+-----------------------+
|                  NULL |
+-----------------------+
```

Keywords

```
PERCENTILE
```

PERCENTILE_ARRAY

PERCENTILE_ARRAY

Description

Syntax

```
ARRAY_DOUBLE PERCENTILE_ARRAY(BIGINT, ARRAY_DOUBLE p)
```

Calculate exact percentiles, suitable for small data volumes. Sorts the specified column in descending order first, then takes the exact pth percentile. The return value is the result of sequentially taking the specified percentages in the array p. Parameter Description: expr: Required. Columns whose values are of type integer (up to bigint). p: The exact percentile is required, an array of constants, taking the value [0.0, 1.0].

example

```
mysql> select percentile_array(k1,[0.3,0.5,0.9]) from baseall;
+----------------------------------------------+
| percentile_array(`k1`, ARRAY(0.3, 0.5, 0.9)) |
+----------------------------------------------+
| [5.2, 8, 13.6]                               |
+----------------------------------------------+
1 row in set (0.02 sec)
```

keywords

PERCENTILE_ARRAY

HLL_UNION_AGG

HLL_UNION_AGG

description

Syntax

```
HLL_UNION_AGG(hll)
```

HLL is an engineering implementation based on HyperLog algorithm, which is used to save the intermediate results of HyperLog calculation process.

It can only be used as the value column type of the table and reduce the amount of data through aggregation to achieve the purpose of speeding up the query.

Based on this, we get an estimate with an error of about 1%. The HLL column is generated by other columns or data imported into the data.

When importing, hll_hash function is used to specify which column in data is used to generate HLL column. It is often used to replace count distinct, and to calculate UV quickly in business by combining rollup.

example

```
MySQL > select HLL_UNION_AGG(uv_set) from test_uv;;
+-------------------------+
THE COURT OF JUSTICE OF THE EUROPEAN COMMUNITIES,
+-------------------------+
| 17721                   |
+-------------------------+
```

keywords

HLL_UNION_AGG,HLL,UNION,AGG

TOPN

TOPN

description

Syntax

```
topn(expr, INT top_num[, INT space_expand_rate])
```

The topn function uses the Space-Saving algorithm to calculate the top_num frequent items in expr, and the result is the frequent items and their occurrence times, which is an approximation

The space_expand_rate parameter is optional and is used to set the number of counters used in the Space-Saving algorithm

```
counter numbers = top_num * space_expand_rate
```

The higher value of space_expand_rate, the more accurate result will be. The default value is 50

example

```
MySQL [test]> select topn(keyword,10) from keyword_table where date>= '2020-06-01' and date <= '2020-06-19' ;
+-----------------------------------------------------------------------------------------------------------+
| topn(`keyword`, 10)                                                                                       |
+-----------------------------------------------------------------------------------------------------------+
| a:157, b:138, c:133, d:133, e:131, f:127, g:124, h:122, i:117, k:117                                      |
+-----------------------------------------------------------------------------------------------------------+

MySQL [test]> select date,topn(keyword,10,100) from keyword_table where date>= '2020-06-17' and date <= '2020-06-19' group by date;
+------------+------------------------------------------------------------------------------------------------+
| date       | topn(`keyword`, 10, 100)                                                                       |
+------------+------------------------------------------------------------------------------------------------+
| 2020-06-19 | a:11, b:8, c:8, d:7, e:7, f:7, g:7, h:7, i:7, j:7                                              |
| 2020-06-18 | a:10, b:8, c:7, f:7, g:7, i:7, k:7, l:7, m:6, d:6                                              |
| 2020-06-17 | a:9, b:8, c:8, j:8, d:7, e:7, f:7, h:7, i:7, k:7                                               |
+------------+------------------------------------------------------------------------------------------------+
```

keywords

TOPN

TOPN_ARRAY

TOPN_ARRAY

description

Syntax

```
ARRAY<T> topn_array(expr, INT top_num[, INT space_expand_rate])
```

The topn function uses the Space-Saving algorithm to calculate the top_num frequent items in expr, and return an array about the top n nums, which is an approximation

The space_expand_rate parameter is optional and is used to set the number of counters used in the Space-Saving algorithm

```
counter numbers = top_num * space_expand_rate
```

The higher value of space_expand_rate, the more accurate result will be. The default value is 50

example

```
mysql> select topn_array(k3,3) from baseall;
+-------------------------+
| topn_array(`k3`, 3)     |
+-------------------------+
| [3021, 2147483647, 5014] |
+-------------------------+
1 row in set (0.02 sec)

mysql> select topn_array(k3,3,100) from baseall;
+-------------------------+
| topn_array(`k3`, 3, 100) |
+-------------------------+
| [3021, 2147483647, 5014] |
+-------------------------+
1 row in set (0.02 sec)
```

keywords

TOPN, TOPN_ARRAY

TOPN_WEIGHTED

TOPN_WEIGHTED

description

Syntax

```
ARRAY<T> topn_weighted(expr, BigInt weight, INT top_num[, INT space_expand_rate])
```

The topn_weighted function is calculated using the Space-Saving algorithm, and the sum of the weights in expr is the result of the top n numbers, which is an approximate value

The space_expand_rate parameter is optional and is used to set the number of counters used in the Space-Saving algorithm

```
counter numbers = top_num * space_expand_rate
```

The higher value of space_expand_rate, the more accurate result will be. The default value is 50

example

```
mysql> select topn_weighted(k5,k1,3) from baseall;
+-----------------------------+
| topn_weighted(`k5`, `k1`, 3) |
+-----------------------------+
| [0, 243.325, 100.001]       |
+-----------------------------+
1 row in set (0.02 sec)

mysql> select topn_weighted(k5,k1,3,100) from baseall;
+----------------------------------+
| topn_weighted(`k5`, `k1`, 3, 100) |
+----------------------------------+
| [0, 243.325, 100.001]            |
+----------------------------------+
1 row in set (0.02 sec)
```

keywords

TOPN, TOPN_WEIGHTED

COUNT

COUNT

Description

Syntax

```
COUNT([DISTINCT] expr)
```

Number of rows used to return the required rows

example

```
MySQL > select count(*) from log_statis group by datetime;
+----------+
| count(*) |
+----------+
| 28515903 |
+----------+

MySQL > select count(datetime) from log_statis group by datetime;
+-------------------+
| count(`datetime`) |
+-------------------+
|         28521682  |
+-------------------+

MySQL > select count(distinct datetime) from log_statis group by datetime;
+-------------------------------+
| count(DISTINCT `datetime`)    |
+-------------------------------+
|                       71045   |
+-------------------------------+
```

keywords

COUNT

SUM

SUM

Description

Syntax

```
Sum (Expr)
```

Used to return the sum of all values of the selected field

example

```
MySQL > select sum(scan_rows) from log_statis group by datetime;
+------------------+
| sum(`scan_rows`) |
+------------------+
|       8217360135 |
+------------------+
```

keywords

SUM

MAX_BY

MAX_BY

description

Syntax

```
MAX_BY(expr1, expr2)
```

Returns the value of an expr1 associated with the maximum value of expr2 in a group.

example

```
MySQL > select * from tbl;
+------+------+------+------+
| k1   | k2   | k3   | k4   |
+------+------+------+------+
|    0 | 3    | 2    |  100 |
|    1 | 2    | 3    |    4 |
|    4 | 3    | 2    |    1 |
|    3 | 4    | 2    |    1 |
+------+------+------+------+

MySQL > select max_by(k1, k4) from tbl;
+--------------------+
| max_by(`k1`, `k4`) |
+--------------------+
|                  0 |
+--------------------+
```

keywords

MAX_BY

BITMAP_UNION

BITMAP_UNION

description

example

Create table

The aggregation model needs to be used when creating the table. The data type is bitmap and the aggregation function is bitmap_union.

```
CREATE TABLE `pv_bitmap` (
  `dt` int (11) NULL COMMENT" ",
  `page` varchar (10) NULL COMMENT" ",
  `user_id` bitmap BITMAP_UNION NULL COMMENT" "
) ENGINE = OLAP
AGGREGATE KEY (`dt`,` page`)
COMMENT "OLAP"
DISTRIBUTED BY HASH (`dt`) BUCKETS 2;
```

Note: When the amount of data is large, it is best to create a corresponding rollup table for high-frequency bitmap_union queries

```
ALTER TABLE pv_bitmap ADD ROLLUP pv (page, user_id);
```

Data Load

`TO_BITMAP (expr)`: Convert 0 ~ 18446744073709551615 unsigned bigint to bitmap

`BITMAP_EMPTY ()`: Generate empty bitmap columns, used for insert or import to fill the default value

`BITMAP_HASH (expr)` or `BITMAP_HASH64 (expr)`: Convert any type of column to a bitmap by hashing

Stream Load

```
cat data | curl --location-trusted -u user: passwd -T--H "columns: dt, page, user_id, user_id = to_bitmap (user_id)" http: // host: 8410 / api / test / testDb / _stream_load
```

```
cat data | curl --location-trusted -u user: passwd -T--H "columns: dt, page, user_id, user_id = bitmap_hash (user_id)" http: // host: 8410 / api / test / testDb / _stream_load
```

```
cat data | curl --location-trusted -u user: passwd -T--H "columns: dt, page, user_id, user_id = bitmap_empty ()" http: // host: 8410 / api / test / testDb / _stream_load
```

Insert Into

id2's column type is bitmap

```
insert into bitmap_table1 select id, id2 from bitmap_table2;
```

id2's column type is bitmap

```
INSERT INTO bitmap_table1 (id, id2) VALUES (1001, to_bitmap (1000)), (1001, to_bitmap (2000));
```

id2's column type is bitmap

```
insert into bitmap_table1 select id, bitmap_union (id2) from bitmap_table2 group by id;
```

id2's column type is int

```
insert into bitmap_table1 select id, to_bitmap (id2) from table;
```

id2's column type is String

```
insert into bitmap_table1 select id, bitmap_hash (id_string) from table;
```

Data Query

Syntax

`BITMAP_UNION (expr)`: Calculate the union of two Bitmaps. The return value is the new Bitmap value.

`BITMAP_UNION_COUNT (expr)`: Calculate the cardinality of the union of two Bitmaps, equivalent to BITMAP_COUNT (BITMAP_UNION (expr)). It is recommended to use the BITMAP_UNION_COUNT function first, its performance is better than BITMAP_COUNT (BITMAP_UNION (expr)).

`BITMAP_UNION_INT (expr)`: Count the number of different values in columns of type TINYINT, SMALLINT and INT, return the sum of COUNT (DISTINCT expr) same

`INTERSECT_COUNT (bitmap_column_to_count, filter_column, filter_values ...)`: The calculation satisfies filter_column The cardinality of the intersection of multiple bitmaps of the filter. bitmap_column_to_count is a column of type bitmap, filter_column is a column of varying dimensions, and filter_values is a list of dimension values.

Example

The following SQL uses the pv_bitmap table above as an example:

Calculate the deduplication value for user_id:

```
select bitmap_union_count (user_id) from pv_bitmap;

select bitmap_count (bitmap_union (user_id)) from pv_bitmap;
```

Calculate the deduplication value of id:

```
select bitmap_union_int (id) from pv_bitmap;
```

Calculate the retention of user_id:

```
select intersect_count (user_id, page, 'meituan') as meituan_uv,
intersect_count (user_id, page, 'waimai') as waimai_uv,
intersect_count (user_id, page, 'meituan', 'waimai') as retention // Number of users appearing on both 'meituan' and 'waimai' pages
from pv_bitmap
where page in ('meituan', 'waimai');
```

keywords

BITMAP, BITMAP_COUNT, BITMAP_EMPTY, BITMAP_UNION, BITMAP_UNION_INT, TO_BITMAP, BITMAP_UNION_COUNT, INTERSECT_COUNT

GROUP_BITMAP_XOR

group_bitmap_xor

description

Syntax

```
BITMAP GROUP_BITMAP_XOR(expr)
```

Perform an xor calculation on expr, and return a new bitmap.

example

```
mysql>  select page, bitmap_to_string(user_id) from pv_bitmap;
+------+----------------------------+
| page | bitmap_to_string(`user_id`) |
+------+----------------------------+
| m    | 4,7,8                      |
| m    | 1,3,6,15                   |
| m    | 4,7                        |
+------+----------------------------+

mysql> select page, bitmap_to_string(group_bitmap_xor(user_id)) from pv_bitmap group by page;
+------+-----------------------------------------------+
| page | bitmap_to_string(group_bitmap_xor(`user_id`)) |
+------+-----------------------------------------------+
| m    | 1,3,6,8,15                                    |
+------+-----------------------------------------------+
```

keywords

```
GROUP_BITMAP_XOR,BITMAP
```

GROUP_BIT_AND

group_bit_and

description

Syntax

```
expr GROUP_BIT_AND(expr)
```

Perform an and calculation on expr, and return a new expr. All ints are supported

example

```
mysql> select * from group_bit;
+-------+
| value |
+-------+
|     3 |
|     1 |
|     2 |
|     4 |
+-------+
4 rows in set (0.02 sec)

mysql> select group_bit_and(value) from group_bit;
+------------------------+
| group_bit_and(`value`) |
+------------------------+
|                      0 |
+------------------------+
```

keywords

```
GROUP_BIT_AND,BIT
```

GROUP_BIT_OR

group_bit_or

description

Syntax

```
expr GROUP_BIT_OR(expr)
```

Perform an or calculation on expr, and return a new expr. All ints are supported

example

```
mysql> select * from group_bit;
+-------+
| value |
+-------+
|     3 |
|     1 |
|     2 |
|     4 |
+-------+
4 rows in set (0.02 sec)

mysql> select group_bit_or(value) from group_bit;
+-----------------------+
| group_bit_or(`value`) |
+-----------------------+
|                     7 |
+-----------------------+
```

keywords

```
GROUP_BIT_OR,BIT
```

GROUP_BIT_XOR

group_bit_xor

description

Syntax

```
expr GROUP_BIT_XOR(expr)
```

Perform an xor calculation on expr, and return a new expr. All ints are supported

example

```
mysql> select * from group_bit;
+-------+
| value |
+-------+
|     3 |
|     1 |
|     2 |
|     4 |
+-------+
4 rows in set (0.02 sec)

mysql> select group_bit_xor(value) from group_bit;
+------------------------+
| group_bit_xor(`value`) |
+------------------------+
|                      4 |
+------------------------+
```

keywords

```
GROUP_BIT_XOR,BIT
```

PERCENTILE_APPROX

PERCENTILE_APPROX

Description

Syntax

```
PERCENTILE_APPROX(expr, DOUBLE p[, DOUBLE compression])
```

Return the approximation of the point p, where the value of P is between 0 and 1.

Compression param is optional and can be setted to a value in the range of [2048, 10000]. The bigger compression you set, the more precise result and more time cost you will get. If it is not setted or not setted in the correct range, PERCENTILE_APPROX function will run with a default compression param of 10000.

This function uses fixed size memory, so less memory can be used for columns with high cardinality, and can be used to calculate statistics such as tp99.

example

```
MySQL > select `table`, percentile_approx(cost_time,0.99) from log_statis group by `table`;
+----------------------+--------------------------+
| table    | percentile_approx(`cost_time`, 0.99) |
+----------+--------------------------------------+
| test     |                                54.22 |
+----------+--------------------------------------+

MySQL > select `table`, percentile_approx(cost_time,0.99, 4096) from log_statis group by `table`;
+----------------------+--------------------------+
| table    | percentile_approx(`cost_time`, 0.99, 4096.0) |
+----------+--------------------------------------+
| test     |                                54.21 |
+----------+--------------------------------------+
```

keywords

PERCENTILE_APPROX,PERCENTILE,APPROX

STDDEV,STDDEV_POP

STDDEV,STDDEV_POP

Description

Syntax

```
stddev (expl)
```

Returns the standard deviation of the expr expression

example

```
MySQL > select stddev(scan_rows) from log_statis group by datetime;
+---------------------+
| stddev(`scan_rows`) |
+---------------------+
|  2.3736656687790934 |
+---------------------+

MySQL > select stddev_pop(scan_rows) from log_statis group by datetime;
+-------------------------+
| stddev_pop(`scan_rows`) |
+-------------------------+
|      2.3722760595994914 |
+-------------------------+
```

keywords

STDDEV,STDDEV_POP,POP

GROUP_CONCAT

GROUP_CONCAT

description

Syntax

```
VARCHAR GROUP_CONCAT([DISTINCT] VARCHAR str[, VARCHAR sep] [ORDER BY { col_name | expr} [ASC | DESC]])
```

This function is an aggregation function similar to sum (), and group_concat links multiple rows of results in the result set to a string. The second parameter, sep, is a connection symbol between strings, which can be omitted. This function usually needs to be used with group by statements.

Support Order By for sorting multi-row results, sorting and aggregation columns can be different.

> `group_concat` **don't support using** `distinct` **with** `order by` **together.**

example

```
mysql> select value from test;
+-------+
| value |
+-------+
| a     |
| b     |
| c     |
| c     |
+-------+

mysql> select GROUP_CONCAT(value) from test;
+----------------------+
| GROUP_CONCAT(`value`) |
+----------------------+
| a, b, c, c           |
+----------------------+

mysql> select GROUP_CONCAT(value, " ") from test;
+---------------------------+
| GROUP_CONCAT(`value`, ' ') |
+---------------------------+
| a b c c                   |
+---------------------------+

mysql> select GROUP_CONCAT(DISTINCT value) from test;
+----------------------+
| GROUP_CONCAT(`value`) |
+----------------------+
| a, b, c              |
+----------------------+

mysql> select GROUP_CONCAT(value, NULL) from test;
```

```
+----------------------------+
| GROUP_CONCAT(`value`, NULL)|
+----------------------------+
| NULL                       |
+----------------------------+
```

keywords

GROUP_CONCAT,GROUP,CONCAT

COLLECT_LIST

COLLECT_LIST

description

Syntax

```
ARRAY<T> collect_list(expr)
```

Returns an array consisting of all values in expr within the group, and ,with the optional `max_size` parameter limits the size of the resulting array to `max_size` elements.The order of elements in the array is non-deterministic. NULL values are excluded. It has an alias `group_array`.

example

```
mysql> select k1,k2,k3 from collect_list_test order by k1;
+------+------------+-------+
| k1   | k2         | k3    |
+------+------------+-------+
|    1 | 2023-01-01 | hello |
|    2 | 2023-01-02 | NULL  |
|    2 | 2023-01-02 | hello |
|    3 | NULL       | world |
|    3 | 2023-01-02 | hello |
|    4 | 2023-01-02 | sql   |
|    4 | 2023-01-03 | sql   |
+------+------------+-------+

mysql> select collect_list(k1),collect_list(k1,3) from collect_list_test;
+-------------------------+-------------------------+
| collect_list(`k1`)      | collect_list(`k1`,3)    |
+-------------------------+-------------------------+
| [1,2,2,3,3,4,4]         | [1,2,2]                 |
+-------------------------+-------------------------+

mysql> select k1,collect_list(k2),collect_list(k3,1) from collect_list_test group by k1 order by k1;
+------+-------------------------+-------------------------+
| k1   | collect_list(`k2`)      | collect_list(`k3`,1)    |
+------+-------------------------+-------------------------+
|    1 | [2023-01-01]            | [hello]                 |
|    2 | [2023-01-02,2023-01-02] | [hello]                 |
|    3 | [2023-01-02]            | [world]                 |
|    4 | [2023-01-02,2023-01-03] | [sql]                   |
+------+-------------------------+-------------------------+
```

keywords

COLLECT_LIST,GROUP_ARRAY,COLLECT_SET,ARRAY

MIN_BY

MIN_BY

description

Syntax

```
MIN_BY(expr1, expr2)
```

Returns the value of an expr1 associated with the minimum value of expr2 in a group.

example

```
MySQL > select * from tbl;
+------+------+------+------+
| k1   | k2   | k3   | k4   |
+------+------+------+------+
|    0 | 3    | 2    |  100 |
|    1 | 2    | 3    |    4 |
|    4 | 3    | 2    |    1 |
|    3 | 4    | 2    |    1 |
+------+------+------+------+

MySQL > select min_by(k1, k4) from tbl;
+--------------------+
| min_by(`k1`, `k4`) |
+--------------------+
|                  4 |
+--------------------+
```

keywords

MIN_BY

MAX

MAX

description

Syntax

```
MAX(expr)
```

Returns the maximum value of an expr expression

example

```
MySQL > select max(scan_rows) from log_statis group by datetime;
+------------------+
| max(`scan_rows`) |
+------------------+
|          4671587 |
+------------------+
```

keywords

MAX

ANY_VALUE

ANY_VALUE

ANY_VALUE

description

Syntax

```
ANY_VALUE(expr)
```

If there is a non NULL value in expr, any non NULL value is returned; otherwise, NULL is returned.

Alias function: `ANY(expr)`

example

```
mysql> select id, any_value(name) from cost2 group by id;
+------+------------------+
| id   | any_value(`name`) |
+------+------------------+
|    3 | jack             |
|    2 | jack             |
+------+------------------+
```

keywords

ANY_VALUE, ANY

VARIANCE_SAMP,VARIANCE_SAMP

VARIANCE_SAMP,VARIANCE_SAMP

Description

Syntax

```
VAR SAMP (expr)
```

Returns the sample variance of the expr expression

example

```
MySQL > select var_samp(scan_rows) from log_statis group by datetime;
+----------------------+
| var_samp(`scan_rows`) |
+----------------------+
|    5.6227132145741789 |
+----------------------+
```

keywords

VAR SAMP, VARIANCE SAMP,VAR,SAMP,VARIANCE

APPROX_COUNT_DISTINCT

Description

Syntax

```
APPROX_COUNT_DISTINCT(expr)
```

Returns an approximate aggregation function similar to the result of `COUNT(DISTINCT col)`.

It is implemented based on the HyperLogLog algorithm, which uses a fixed size of memory to estimate the column base. The algorithm is based on the assumption of a null distribution in the tails, and the accuracy depends on the data distribution. Based on the fixed bucket size used by Doris, the relative standard error of the algorithm is 0.8125%.

For a more detailed and specific analysis, see related paper

Example

```
MySQL > select approx_count_distinct(query_id) from log_statis group by datetime;
+-----------------+
| approx_count_distinct(`query_id`) |
+-----------------+
| 17721           |
+-----------------+
```

Keywords

APPROX_COUNT_DISTINCT

VARIANCE,VAR_POP,VARIANCE_POP

VARIANCE,VAR_POP,VARIANCE_POP

Description

Syntax

```
VARIANCE(expr)
```

Returns the variance of the expr expression

example

```
MySQL > select variance(scan_rows) from log_statis group by datetime;
+----------------------+
| variance(`scan_rows`) |
+----------------------+
|    5.6183332881176211 |
+----------------------+

MySQL > select var_pop(scan_rows) from log_statis group by datetime;
+----------------------+
| var_pop(`scan_rows`) |
+----------------------+
|   5.6230744719006163 |
+----------------------+
```

keywords

VARIANCE,VAR_POP,VARIANCE_POP,VAR,POP

RETENTION

RETENTION

RETENTION

Description

Syntax

```
retention(event1, event2, ... , eventN);
```

The `retention` function takes as arguments a set of conditions from 1 to 32 arguments of type `UInt8` that indicate whether a certain condition was met for the event. Any condition can be specified as an argument.

The conditions, except the first, apply in pairs: the result of the second will be true if the first and second are true, of the third if the first and third are true, etc.

To put it simply, the first digit of the return value array indicates whether `event1` is true or false, the second digit represents the truth and falseness of `event1` and `event2`, and the third digit represents whether `event1` is true or false and `event3` is true False and, and so on. If `event1` is false, return an array full of zeros.

Arguments

`event` — An expression that returns a `UInt8` result (1 or 0).

Returned value

An array of 1s and 0s with a maximum length of 32 bits, the final output array has the same length as the input parameter.

1 — Condition was met for the event.

0 — Condition wasn't met for the event.

example

```
DROP TABLE IF EXISTS retention_test;

CREATE TABLE retention_test(
                `uid` int COMMENT 'user id',
                `date` datetime COMMENT 'date time'
                )
DUPLICATE KEY(uid)
DISTRIBUTED BY HASH(uid) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);
```

```
INSERT into retention_test (uid, date) values (0, '2022-10-12'),
                                      (0, '2022-10-13'),
                                      (0, '2022-10-14'),
                                      (1, '2022-10-12'),
                                      (1, '2022-10-13'),
                                      (2, '2022-10-12');

SELECT * from retention_test;

+------+---------------------+
| uid  | date                |
+------+---------------------+
|    0 | 2022-10-14 00:00:00 |
|    0 | 2022-10-13 00:00:00 |
|    0 | 2022-10-12 00:00:00 |
|    1 | 2022-10-13 00:00:00 |
|    1 | 2022-10-12 00:00:00 |
|    2 | 2022-10-12 00:00:00 |
+------+---------------------+

SELECT
    uid,
    retention(date = '2022-10-12')
        AS r
            FROM retention_test
            GROUP BY uid
            ORDER BY uid ASC;

+------+------+
| uid  | r    |
+------+------+
|    0 | [1]  |
|    1 | [1]  |
|    2 | [1]  |
+------+------+

SELECT
    uid,
    retention(date = '2022-10-12', date = '2022-10-13')
        AS r
            FROM retention_test
            GROUP BY uid
            ORDER BY uid ASC;

+------+--------+
| uid  | r      |
+------+--------+
|    0 | [1, 1] |
|    1 | [1, 1] |
|    2 | [1, 0] |
+------+--------+

SELECT
    uid,
    retention(date = '2022-10-12', date = '2022-10-13', date = '2022-10-14')
        AS r
            FROM retention_test
            GROUP BY uid
            ORDER BY uid ASC;

+------+-----------+
| uid  | r         |
+------+-----------+
|    0 | [1, 1, 1] |
```

```
|    1 | [1, 1, 0] |
|    2 | [1, 0, 0] |
+------+-----------+
```

keywords

RETENTION

SEQUENCE_MATCH

SEQUENCE-MATCH

Description

Syntax

```
sequence_match(pattern, timestamp, cond1, cond2, ...);
```

Checks whether the sequence contains an event chain that matches the pattern.

WARNING!

Events that occur at the same second may lay in the sequence in an undefined order affecting the result.

Arguments

`pattern` — Pattern string.

Pattern syntax

`(?N)` — Matches the condition argument at position N. Conditions are numbered in the `[1, 32]` range. For example, `(?1)` matches the argument passed to the `cond1` parameter.

`.*` — Matches any number of events. You do not need conditional arguments to match this element of the pattern.

`(?t operator value)` — Sets the time in seconds that should separate two events.

We define `t` as the difference in seconds between two times, For example, pattern `(?1)(?t>1800)(?2)` matches events that occur more than 1800 seconds from each other. pattern `(?1)(?t>10000)(?2)` matches events that occur more than 10000 seconds from each other. An arbitrary number of any events can lay between these events. You can use the `>=`, `>`, `<`, `<=`, `==` operators.

`timestamp` — Column considered to contain time data. Typical data types are `Date` and `DateTime`. You can also use any of the supported UInt data types.

`cond1`, `cond2` — Conditions that describe the chain of events. Data type: `UInt8`. You can pass up to 32 condition arguments. The function takes only the events described in these conditions into account. If the sequence contains data that isn't described in a condition, the function skips them.

Returned value

1, if the pattern is matched.

0, if the pattern isn't matched.

example

match examples

```
DROP TABLE IF EXISTS sequence_match_test1;

CREATE TABLE sequence_match_test1(
                `uid` int COMMENT 'user id',
                `date` datetime COMMENT 'date time',
                `number` int NULL COMMENT 'number'
                )
DUPLICATE KEY(uid)
DISTRIBUTED BY HASH(uid) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT INTO sequence_match_test1(uid, date, number) values (1, '2022-11-02 10:41:00', 1),
                                                    (2, '2022-11-02 13:28:02', 2),
                                                    (3, '2022-11-02 16:15:01', 1),
                                                    (4, '2022-11-02 19:05:04', 2),
                                                    (5, '2022-11-02 20:08:44', 3);

SELECT * FROM sequence_match_test1 ORDER BY date;

+------+---------------------+--------+
| uid  | date                | number |
+------+---------------------+--------+
|    1 | 2022-11-02 10:41:00 |      1 |
|    2 | 2022-11-02 13:28:02 |      2 |
|    3 | 2022-11-02 16:15:01 |      1 |
|    4 | 2022-11-02 19:05:04 |      2 |
|    5 | 2022-11-02 20:08:44 |      3 |
+------+---------------------+--------+

SELECT sequence_match('(?1)(?2)', date, number = 1, number = 3) FROM sequence_match_test1;

+-------------------------------------------------------------------+
| sequence_match('(?1)(?2)', `date`, `number` = 1, `number` = 3) |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+

SELECT sequence_match('(?1)(?2)', date, number = 1, number = 2) FROM sequence_match_test1;

+-------------------------------------------------------------------+
| sequence_match('(?1)(?2)', `date`, `number` = 1, `number` = 2) |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+

SELECT sequence_match('(?1)(?t>=3600)(?2)', date, number = 1, number = 2) FROM sequence_match_test1;

+-----------------------------------------------------------------------------+
| sequence_match('(?1)(?t>=3600)(?2)', `date`, `number` = 1, `number` = 2) |
+-----------------------------------------------------------------------------+
|                                                                           1 |
+-----------------------------------------------------------------------------+
```

not match examples

```
DROP TABLE IF EXISTS sequence_match_test2;

CREATE TABLE sequence_match_test2(
                `uid` int COMMENT 'user id',
                `date` datetime COMMENT 'date time',
                `number` int NULL COMMENT 'number'
                )
DUPLICATE KEY(uid)
DISTRIBUTED BY HASH(uid) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT INTO sequence_match_test2(uid, date, number) values (1, '2022-11-02 10:41:00', 1),
                                                    (2, '2022-11-02 11:41:00', 7),
                                                    (3, '2022-11-02 16:15:01', 3),
                                                    (4, '2022-11-02 19:05:04', 4),
                                                    (5, '2022-11-02 21:24:12', 5);

SELECT * FROM sequence_match_test2 ORDER BY date;

+------+---------------------+--------+
| uid  | date                | number |
+------+---------------------+--------+
|    1 | 2022-11-02 10:41:00 |      1 |
|    2 | 2022-11-02 11:41:00 |      7 |
|    3 | 2022-11-02 16:15:01 |      3 |
|    4 | 2022-11-02 19:05:04 |      4 |
|    5 | 2022-11-02 21:24:12 |      5 |
+------+---------------------+--------+

SELECT sequence_match('(?1)(?2)', date, number = 1, number = 2) FROM sequence_match_test2;

+-------------------------------------------------------------+
| sequence_match('(?1)(?2)', `date`, `number` = 1, `number` = 2) |
+-------------------------------------------------------------+
|                                                           0 |
+-------------------------------------------------------------+

SELECT sequence_match('(?1)(?2).*', date, number = 1, number = 2) FROM sequence_match_test2;

+--------------------------------------------------------------+
| sequence_match('(?1)(?2).*', `date`, `number` = 1, `number` = 2) |
+--------------------------------------------------------------+
|                                                            0 |
+--------------------------------------------------------------+

SELECT sequence_match('(?1)(?t>3600)(?2)', date, number = 1, number = 7) FROM sequence_match_test2;

+-----------------------------------------------------------------------+
| sequence_match('(?1)(?t>3600)(?2)', `date`, `number` = 1, `number` = 7) |
+-----------------------------------------------------------------------+
|                                                                     0 |
+-----------------------------------------------------------------------+
```

special examples

```
DROP TABLE IF EXISTS sequence_match_test3;

CREATE TABLE sequence_match_test3(
                `uid` int COMMENT 'user id',
                `date` datetime COMMENT 'date time',
                `number` int NULL COMMENT 'number'
```

```
               )
DUPLICATE KEY(uid)
DISTRIBUTED BY HASH(uid) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT INTO sequence_match_test3(uid, date, number) values (1, '2022-11-02 10:41:00', 1),
                                                           (2, '2022-11-02 11:41:00', 7),
                                                           (3, '2022-11-02 16:15:01', 3),
                                                           (4, '2022-11-02 19:05:04', 4),
                                                           (5, '2022-11-02 21:24:12', 5);

SELECT * FROM sequence_match_test3 ORDER BY date;

+------+---------------------+--------+
| uid  | date                | number |
+------+---------------------+--------+
|    1 | 2022-11-02 10:41:00 |      1 |
|    2 | 2022-11-02 11:41:00 |      7 |
|    3 | 2022-11-02 16:15:01 |      3 |
|    4 | 2022-11-02 19:05:04 |      4 |
|    5 | 2022-11-02 21:24:12 |      5 |
+------+---------------------+--------+
```

Perform the query:

```
SELECT sequence_match('(?1)(?2)', date, number = 1, number = 5) FROM sequence_match_test3;

+-------------------------------------------------------------------+
| sequence_match('(?1)(?2)', `date`, `number` = 1, `number` = 5) |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+
```

This is a very simple example. The function found the event chain where number 5 follows number 1. It skipped number 7,3,4 between them, because the number is not described as an event. If we want to take this number into account when searching for the event chain given in the example, we should make a condition for it.

Now, perform this query:

```
SELECT sequence_match('(?1)(?2)', date, number = 1, number = 5, number = 4) FROM sequence_match_test3;

+-----------------------------------------------------------------------------------+
| sequence_match('(?1)(?2)', `date`, `number` = 1, `number` = 5, `number` = 4) |
+-----------------------------------------------------------------------------------+
|                                                                                 0 |
+-----------------------------------------------------------------------------------+
```

The result is kind of confusing. In this case, the function couldn't find the event chain matching the pattern, because the event for number 4 occurred between 1 and 5. If in the same case we checked the condition for number 6, the sequence would match the pattern.

```
SELECT sequence_match('(?1)(?2)', date, number = 1, number = 5, number = 6) FROM sequence_match_test3;

+-----------------------------------------------------------------------------------+
| sequence_match('(?1)(?2)', `date`, `number` = 1, `number` = 5, `number` = 6) |
+-----------------------------------------------------------------------------------+
```

```
|                                                                              1 |
+--------------------------------------------------------------------------------+
```

keywords

SEQUENCE_MATCH

SEQUENCE-COUNT

SEQUENCE-COUNT

Description

Syntax

```
sequence_count(pattern, timestamp, cond1, cond2, ...);
```

Counts the number of event chains that matched the pattern. The function searches event chains that do not overlap. It starts to search for the next chain after the current chain is matched.

WARNING!

Events that occur at the same second may lay in the sequence in an undefined order affecting the result.

Arguments

`pattern` — Pattern string.

Pattern syntax

`(?N)` — Matches the condition argument at position N. Conditions are numbered in the `[1, 32]` range. For example, `(?1)` matches the argument passed to the `cond1` parameter.

`.*` — Matches any number of events. You do not need conditional arguments to count this element of the pattern.

`(?t operator value)` — Sets the time in seconds that should separate two events.

We define `t` as the difference in seconds between two times, For example, pattern `(?1)(?t>1800)(?2)` matches events that occur more than 1800 seconds from each other. pattern `(?1)(?t>10000)(?2)` matches events that occur more than 10000 seconds from each other. An arbitrary number of any events can lay between these events. You can use the `>=`, `>`, `<`, `<=`, `==` operators.

`timestamp` — Column considered to contain time data. Typical data types are `Date` and `DateTime`. You can also use any of the supported UInt data types.

`cond1`, `cond2` — Conditions that describe the chain of events. Data type: `UInt8`. You can pass up to 32 condition arguments. The function takes only the events described in these conditions into account. If the sequence contains data that isn't described in a condition, the function skips them.

Returned value

Number of non-overlapping event chains that are matched.

example

count examples

```
DROP TABLE IF EXISTS sequence_count_test2;

CREATE TABLE sequence_count_test2(
                `uid` int COMMENT 'user id',
                `date` datetime COMMENT 'date time',
                `number` int NULL COMMENT 'number'
                )
DUPLICATE KEY(uid)
DISTRIBUTED BY HASH(uid) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT INTO sequence_count_test2(uid, date, number) values (1, '2022-11-02 10:41:00', 1),
                                                (2, '2022-11-02 13:28:02', 2),
                                                (3, '2022-11-02 16:15:01', 1),
                                                (4, '2022-11-02 19:05:04', 2),
                                                (5, '2022-11-02 20:08:44', 3);

SELECT * FROM sequence_count_test2 ORDER BY date;

+------+---------------------+--------+
| uid  | date                | number |
+------+---------------------+--------+
|    1 | 2022-11-02 10:41:00 |      1 |
|    2 | 2022-11-02 13:28:02 |      2 |
|    3 | 2022-11-02 16:15:01 |      1 |
|    4 | 2022-11-02 19:05:04 |      2 |
|    5 | 2022-11-02 20:08:44 |      3 |
+------+---------------------+--------+

SELECT sequence_count('(?1)(?2)', date, number = 1, number = 3) FROM sequence_count_test2;

+-------------------------------------------------------------------+
| sequence_count('(?1)(?2)', `date`, `number` = 1, `number` = 3) |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+

SELECT sequence_count('(?1)(?2)', date, number = 1, number = 2) FROM sequence_count_test2;

+-------------------------------------------------------------------+
| sequence_count('(?1)(?2)', `date`, `number` = 1, `number` = 2) |
+-------------------------------------------------------------------+
|                                                                 2 |
+-------------------------------------------------------------------+

SELECT sequence_count('(?1)(?t>=10000)(?2)', date, number = 1, number = 2) FROM sequence_count_test1;

+-------------------------------------------------------------------------------+
| sequence_count('(?1)(?t>=3600)(?2)', `date`, `number` = 1, `number` = 2) |
+-------------------------------------------------------------------------------+
|                                                                             2 |
+-------------------------------------------------------------------------------+
```

not count examples

```
DROP TABLE IF EXISTS sequence_count_test1;

CREATE TABLE sequence_count_test1(
                `uid` int COMMENT 'user id',
                `date` datetime COMMENT 'date time',
                `number` int NULL COMMENT 'number'
                )
DUPLICATE KEY(uid)
DISTRIBUTED BY HASH(uid) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT INTO sequence_count_test1(uid, date, number) values (1, '2022-11-02 10:41:00', 1),
                                                (2, '2022-11-02 11:41:00', 7),
                                                (3, '2022-11-02 16:15:01', 3),
                                                (4, '2022-11-02 19:05:04', 4),
                                                (5, '2022-11-02 21:24:12', 5);

SELECT * FROM sequence_count_test1 ORDER BY date;

+------+---------------------+--------+
| uid  | date                | number |
+------+---------------------+--------+
|    1 | 2022-11-02 10:41:00 |      1 |
|    2 | 2022-11-02 11:41:00 |      7 |
|    3 | 2022-11-02 16:15:01 |      3 |
|    4 | 2022-11-02 19:05:04 |      4 |
|    5 | 2022-11-02 21:24:12 |      5 |
+------+---------------------+--------+

SELECT sequence_count('(?1)(?2)', date, number = 1, number = 2) FROM sequence_count_test1;

+-------------------------------------------------------------+
| sequence_count('(?1)(?2)', `date`, `number` = 1, `number` = 2) |
+-------------------------------------------------------------+
|                                                           0 |
+-------------------------------------------------------------+

SELECT sequence_count('(?1)(?2).*', date, number = 1, number = 2) FROM sequence_count_test1;

+-------------------------------------------------------------+
| sequence_count('(?1)(?2).*', `date`, `number` = 1, `number` = 2) |
+-------------------------------------------------------------+
|                                                           0 |
+-------------------------------------------------------------+

SELECT sequence_count('(?1)(?t>3600)(?2)', date, number = 1, number = 7) FROM sequence_count_test1;

+-------------------------------------------------------------------------+
| sequence_count('(?1)(?t>3600)(?2)', `date`, `number` = 1, `number` = 7) |
+-------------------------------------------------------------------------+
|                                                                       0 |
+-------------------------------------------------------------------------+
```

special examples

```
DROP TABLE IF EXISTS sequence_count_test3;

CREATE TABLE sequence_count_test3(
                `uid` int COMMENT 'user id',
                `date` datetime COMMENT 'date time',
                `number` int NULL COMMENT 'number'
```

```
               )
DUPLICATE KEY(uid)
DISTRIBUTED BY HASH(uid) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT INTO sequence_count_test3(uid, date, number) values (1, '2022-11-02 10:41:00', 1),
                                                           (2, '2022-11-02 11:41:00', 7),
                                                           (3, '2022-11-02 16:15:01', 3),
                                                           (4, '2022-11-02 19:05:04', 4),
                                                           (5, '2022-11-02 21:24:12', 5);

SELECT * FROM sequence_count_test3 ORDER BY date;

+------+---------------------+--------+
| uid  | date                | number |
+------+---------------------+--------+
|    1 | 2022-11-02 10:41:00 |      1 |
|    2 | 2022-11-02 11:41:00 |      7 |
|    3 | 2022-11-02 16:15:01 |      3 |
|    4 | 2022-11-02 19:05:04 |      4 |
|    5 | 2022-11-02 21:24:12 |      5 |
+------+---------------------+--------+
```

Perform the query:

```
SELECT sequence_count('(?1)(?2)', date, number = 1, number = 5) FROM sequence_count_test3;

+-------------------------------------------------------------------+
| sequence_count('(?1)(?2)', `date`, `number` = 1, `number` = 5) |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+
```

This is a very simple example. The function found the event chain where number 5 follows number 1. It skipped number 7,3,4 between them, because the number is not described as an event. If we want to take this number into account when searching for the event chain given in the example, we should make a condition for it.

Now, perform this query:

```
SELECT sequence_count('(?1)(?2)', date, number = 1, number = 5, number = 4) FROM sequence_count_test3;

+-----------------------------------------------------------------------------+
| sequence_count('(?1)(?2)', `date`, `number` = 1, `number` = 5, `number` = 4) |
+-----------------------------------------------------------------------------+
|                                                                           0 |
+-----------------------------------------------------------------------------+
```

The result is kind of confusing. In this case, the function couldn't find the event chain matching the pattern, because the event for number 4 occurred between 1 and 5. If in the same case we checked the condition for number 6, the sequence would count the pattern.

```
SELECT sequence_count('(?1)(?2)', date, number = 1, number = 5, number = 6) FROM sequence_count_test3;

+-----------------------------------------------------------------------------+
| sequence_count('(?1)(?2)', `date`, `number` = 1, `number` = 5, `number` = 6) |
+-----------------------------------------------------------------------------+
```

```
|                                                                              1 |
+--------------------------------------------------------------------------------+
```

keywords

SEQUENCE_COUNT

GROUPING

Description

Indicates whether a specified column expression in a GROUP BY list is aggregated or not. GROUPING returns 1 if the column is aggregated (i.e., coming from a summary row generated by ROLLUP, CUBE, or GROUPING SETS) and 0 if it is not aggregated. This function is useful for distinguishing the NULL values produced by these operations from actual NULL values in the data.

Syntax

```
GROUPING( <column_expression> )
```

Parameters

Parameter	Description
<column_expression>	A column or an expression that appears in the GROUP BY clause.

Return Value

Returns a BIGINT value. The function returns 1 if the specified column expression is aggregated and 0 if it is not.

Examples

The following example groups the `camp` column and aggregates the count of `occupation`. The GROUPING function is applied to the `camp` column to differentiate between the summary row added by the ROLLUP operation and actual NULL values in the data.

```
CREATE TABLE `roles` (
  role_id       INT,
  occupation    VARCHAR(32),
  camp          VARCHAR(32),
  register_time DATE
)
UNIQUE KEY(role_id)
DISTRIBUTED BY HASH(role_id) BUCKETS 1
PROPERTIES (
  "replication_allocation" = "tag.location.default: 1"
);

INSERT INTO `roles` VALUES
(0, 'who am I', NULL, NULL),
(1, 'mage', 'alliance', '2018-12-03 16:11:28'),
(2, 'paladin', 'alliance', '2018-11-30 16:11:28'),
(3, 'rogue', 'horde', '2018-12-01 16:11:28'),
(4, 'priest', 'alliance', '2018-12-02 16:11:28'),
(5, 'shaman', 'horde', NULL),
```

```
(6, 'warrior', 'alliance', NULL),
(7, 'warlock', 'horde', '2018-12-04 16:11:28'),
(8, 'hunter', 'horde', NULL);

SELECT
  camp,
  COUNT(occupation) AS occ_cnt,
  GROUPING(camp) AS grouping
FROM
  `roles`
GROUP BY
  ROLLUP(camp);
```

The result set shows two NULL values under the `camp` column. The first NULL, with a grouping value of 1, is the summary row added by the ROLLUP operation representing the aggregate of all `camp` groups. The second NULL, with a grouping value of 0, represents an actual NULL value from the data.

```
+----------+---------+----------+
| camp     | occ_cnt | grouping |
+----------+---------+----------+
| NULL     |       9 |        1 |
| NULL     |       1 |        0 |
| alliance |       4 |        0 |
| horde    |       4 |        0 |
+----------+---------+----------+
```

GROUPING_ID

Description

Computes the level of grouping for rows in a GROUP BY query. The GROUPING_ID function returns an integer bitmap indicating which columns in the GROUP BY list are not aggregated for a given output row. It can be used in the SELECT list, HAVING, or ORDER BY clauses when a GROUP BY is specified.

Syntax

```
GROUPING_ID(<column_expression> [, ...])
```

Parameters

Parameters	Description
<column_expression>	A column expression from the GROUP BY clause.

Return Value

Returns a BIGINT value representing the grouping bitmap for the given columns.

Examples

Example A: Identify grouping levels

```
SELECT
  department,
  CASE
    WHEN GROUPING_ID(department, level) = 0 THEN level
    WHEN GROUPING_ID(department, level) = 1 THEN CONCAT('Total: ', department)
    WHEN GROUPING_ID(department, level) = 3 THEN 'Total: Company'
    ELSE 'Unknown'
  END AS `Job Title`,
  COUNT(uid) AS `Employee Count`
FROM employee
GROUP BY ROLLUP(department, level)
ORDER BY GROUPING_ID(department, level) ASC;
```

Expected Output:

```
+--------------------+--------------------------+----------------+
| department         | Job Title                | Employee Count |
+--------------------+--------------------------+----------------+
| Board of Directors | Senior                   |              2 |
| Technology         | Senior                   |              3 |
| Sales              | Senior                   |              1 |
| Sales              | Assistant                |              2 |
```

```
| Sales              | Trainee                  |             1 |
| Marketing          | Senior                   |             1 |
| Marketing          | Trainee                  |             2 |
| Marketing          | Assistant                |             1 |
| Board of Directors | Total: Board of Directors |             2 |
| Technology         | Total: Technology        |             3 |
| Sales              | Total: Sales             |             4 |
| Marketing          | Total: Marketing         |             4 |
| NULL               | Total: Company           |            13 |
+--------------------+--------------------------+---------------+
```

Example B: Filter result set using GROUPING_ID

```
SELECT
  department,
  CASE
    WHEN GROUPING_ID(department, level) = 0 THEN level
    WHEN GROUPING_ID(department, level) = 1 THEN CONCAT('Total: ', department)
    WHEN GROUPING_ID(department, level) = 3 THEN 'Total: Company'
    ELSE 'Unknown'
  END AS `Job Title`,
  COUNT(uid) AS `Count`
FROM employee
GROUP BY ROLLUP(department, level)
HAVING `Job Title` = 'Senior';
```

Expected Output:

```
+--------------------+-----------+-------+
| department         | Job Title | Count |
+--------------------+-----------+-------+
| Board of Directors | Senior    |     2 |
| Technology         | Senior    |     3 |
| Sales              | Senior    |     1 |
| Marketing          | Senior    |     1 |
+--------------------+-----------+-------+
```

COUNT_BY_ENUM

COUNT_BY_ENUM

COUNT_BY_ENUM

Description

Syntax

```
count_by_enum(expr1, expr2, ... , exprN);
```

Treats the data in a column as an enumeration and counts the number of values in each enumeration. Returns the number of enumerated values for each column, and the number of non-null values versus the number of null values.

Arguments

`expr1` — At least one input must be specified. The value is a column of type `STRING`.

Returned value

Returns a JSONArray string.

For example:

```
[{
  "cbe": {
    "F": 100,
    "M": 99
  },
  "notnull": 199,
  "null": 1,
  "all": 200
}, {
  "cbe": {
    "20": 10,
    "30": 5,
    "35": 1
  },
  "notnull": 16,
  "null": 184,
  "all": 200
}, {
  "cbe": {
    "China": 10,
    "United States": 9,
    "England": 20,
    "Germany": 30
  },
  "notnull": 69,
```

```
  "null": 131,
  "all": 200
}]
```

Description: The return value is a JSON array string and the order of the internal objects is the order of the input parameters. * cbe: count of non-null values based on enumeration values * notnull: number of non-null values. * null: number of null values * all: total number, including both null and non-null values.

example

```
DROP TABLE IF EXISTS count_by_enum_test;

CREATE TABLE count_by_enum_test(
                                    `id` varchar(1024) NULL,
                                    `f1` text REPLACE_IF_NOT_NULL NULL,
                                    `f2` text REPLACE_IF_NOT_NULL NULL,
                                    `f3` text REPLACE_IF_NOT_NULL NULL
)
AGGREGATE KEY(`id`)
DISTRIBUTED BY HASH(id) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT into count_by_enum_test (id, f1, f2, f3) values
                                                (1, "F", "10", "China"),
                                                (2, "F", "20", "China"),
                                                (3, "M", NULL, "United States"),
                                                (4, "M", NULL, "United States"),
                                                (5, "M", NULL, "England");

SELECT * from count_by_enum_test;

+------+------+------+---------------+
| id   | f1   | f2   | f3            |
+------+------+------+---------------+
| 1    | F    | 10   | China         |
| 2    | F    | 20   | China         |
| 3    | M    | NULL | United States |
| 4    | M    | NULL | United States |
| 5    | M    | NULL | England       |
+------+------+------+---------------+

select count_by_enum(f1) from count_by_enum_test;

+-----------------------------------------------------+
| count_by_enum(`f1`)                                 |
+-----------------------------------------------------+
| [{"cbe":{"M":3,"F":2},"notnull":5,"null":0,"all":5}] |
+-----------------------------------------------------+

select count_by_enum(f2) from count_by_enum_test;

+-------------------------------------------------------+
| count_by_enum(`f2`)                                   |
+-------------------------------------------------------+
| [{"cbe":{"10":1,"20":1},"notnull":2,"null":3,"all":5}] |
+-------------------------------------------------------+

select count_by_enum(f1,f2,f3) from count_by_enum_test;
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
| count_by_enum(`f1`, `f2`, `f3`)
|
+-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
| [{"cbe":{"M":3,"F":2},"notnull":5,"null":0,"all":5},{"cbe":{"20":1,"10":1},"notnull":2,"null":3,"all":5},{"cbe":
{"England":1,"United States":2,"China":2},"notnull":5,"null":0,"all":5}] |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
```

keywords

COUNT_BY_ENUM

HISTOGRAM

HISTOGRAM

description

Syntax

```
histogram(expr[, INT num_buckets])
```

The histogram function is used to describe the distribution of the data. It uses an “equal height” bucking strategy, and divides the data into buckets according to the value of the data. It describes each bucket with some simple data, such as the number of values that fall in the bucket. It is mainly used by the optimizer to estimate the range query.

The result of the function returns an empty or Json string.

Parameter description： - num_buckets：Optional. Limit the number of histogram buckets. The default value is 128.

Alias function: `hist(expr[, INT num_buckets])`

example

```
MySQL [test]> SELECT histogram(c_float) FROM histogram_test;
+-----------------------------------------------------------------------------------------------------------------------------------------------------
--+
| histogram(`c_float`)
|
+-----------------------------------------------------------------------------------------------------------------------------------------------------
--+
| {"num_buckets":3,"buckets":[{"lower":"0.1","upper":"0.1","count":1,"pre_sum":0,"ndv":1},...]} |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
--+

MySQL [test]> SELECT histogram(c_string, 2) FROM histogram_test;
+-----------------------------------------------------------------------------------------------------------------------------------------------------
--+
| histogram(`c_string`)
|
+-----------------------------------------------------------------------------------------------------------------------------------------------------
--+
| {"num_buckets":2,"buckets":[{"lower":"str1","upper":"str7","count":4,"pre_sum":0,"ndv":3},...]} |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
--+
```

Query result description：

```
{
    "num_buckets": 3,
    "buckets": [
        {
```

```
            "lower": "0.1",
            "upper": "0.2",
            "count": 2,
            "pre_sum": 0,
            "ndv": 2
        },
        {
            "lower": "0.8",
            "upper": "0.9",
            "count": 2,
            "pre_sum": 2,
            "ndv": 2
        },
        {
            "lower": "1.0",
            "upper": "1.0",
            "count": 2,
            "pre_sum": 4,
            "ndv": 1
        }
    ]
}
```

Field description： - num_buckets：The number of buckets - buckets：All buckets - lower：Upper bound of the bucket - upper：Lower bound of the bucket - count：The number of elements contained in the bucket - pre_sum：The total number of elements in the front bucket - ndv：The number of different values in the bucket

> Total number of histogram elements = number of elements in the last bucket(count) + total number of elements in the previous bucket(pre_sum).

keywords

HISTOGRAM, HIST

MAP_AGG

MAP_AGG

description

Syntax

```
MAP_AGG(expr1, expr2)
```

Returns a map consists of expr1 as the key and expr2 as the corresponding value.

example

```
MySQL > select `n_nationkey`, `n_name`, `n_regionkey` from `nation`;
+-------------+----------------+-------------+
| n_nationkey | n_name         | n_regionkey |
+-------------+----------------+-------------+
|           0 | ALGERIA        |           0 |
|           1 | ARGENTINA      |           1 |
|           2 | BRAZIL         |           1 |
|           3 | CANADA         |           1 |
|           4 | EGYPT          |           4 |
|           5 | ETHIOPIA       |           0 |
|           6 | FRANCE         |           3 |
|           7 | GERMANY        |           3 |
|           8 | INDIA          |           2 |
|           9 | INDONESIA      |           2 |
|          10 | IRAN           |           4 |
|          11 | IRAQ           |           4 |
|          12 | JAPAN          |           2 |
|          13 | JORDAN         |           4 |
|          14 | KENYA          |           0 |
|          15 | MOROCCO        |           0 |
|          16 | MOZAMBIQUE     |           0 |
|          17 | PERU           |           1 |
|          18 | CHINA          |           2 |
|          19 | ROMANIA        |           3 |
|          20 | SAUDI ARABIA   |           4 |
|          21 | VIETNAM        |           2 |
|          22 | RUSSIA         |           3 |
|          23 | UNITED KINGDOM |           3 |
|          24 | UNITED STATES  |           1 |
+-------------+----------------+-------------+

MySQL > select `n_regionkey`, map_agg(`n_nationkey`, `n_name`) from `nation` group by `n_regionkey`;
+-------------+-------------------------------------------------------------------------------+
| n_regionkey | map_agg(`n_nationkey`, `n_name`)                                              |
+-------------+-------------------------------------------------------------------------------+
|           1 | {1:"ARGENTINA", 2:"BRAZIL", 3:"CANADA", 17:"PERU", 24:"UNITED STATES"}        |
|           0 | {0:"ALGERIA", 5:"ETHIOPIA", 14:"KENYA", 15:"MOROCCO", 16:"MOZAMBIQUE"}        |
|           3 | {6:"FRANCE", 7:"GERMANY", 19:"ROMANIA", 22:"RUSSIA", 23:"UNITED KINGDOM"}     |
|           4 | {4:"EGYPT", 10:"IRAN", 11:"IRAQ", 13:"JORDAN", 20:"SAUDI ARABIA"}             |
|           2 | {8:"INDIA", 9:"INDONESIA", 12:"JAPAN", 18:"CHINA", 21:"VIETNAM"}              |
+-------------+-------------------------------------------------------------------------------+
```

```
MySQL > select n_regionkey, map_agg(`n_name`, `n_nationkey` % 5) from `nation` group by `n_regionkey`;
+-------------+-------------------------------------------------------------------------+
| n_regionkey | map_agg(`n_name`, (`n_nationkey` % 5))                                  |
+-------------+-------------------------------------------------------------------------+
|           2 | {"INDIA":3, "INDONESIA":4, "JAPAN":2, "CHINA":3, "VIETNAM":1}           |
|           0 | {"ALGERIA":0, "ETHIOPIA":0, "KENYA":4, "MOROCCO":0, "MOZAMBIQUE":1}     |
|           3 | {"FRANCE":1, "GERMANY":2, "ROMANIA":4, "RUSSIA":2, "UNITED KINGDOM":3} |
|           1 | {"ARGENTINA":1, "BRAZIL":2, "CANADA":3, "PERU":2, "UNITED STATES":4}    |
|           4 | {"EGYPT":4, "IRAN":0, "IRAQ":1, "JORDAN":3, "SAUDI ARABIA":0}           |
+-------------+-------------------------------------------------------------------------+
```

keywords

MAP_AGG

BITMAP_AGG

BITMAP_AGG

description

Syntax

```
BITMAP_AGG(expr)
```

This aggregating function returns a bitmap that aggregates the values of expr, excluding any null values. The type of expr needs to be TINYINT, SMALLINT, INT, or BIGINT.

example

```
MySQL > select `n_nationkey`, `n_name`, `n_regionkey` from `nation`;
+-------------+----------------+-------------+
| n_nationkey | n_name         | n_regionkey |
+-------------+----------------+-------------+
|           0 | ALGERIA        |           0 |
|           1 | ARGENTINA      |           1 |
|           2 | BRAZIL         |           1 |
|           3 | CANADA         |           1 |
|           4 | EGYPT          |           4 |
|           5 | ETHIOPIA       |           0 |
|           6 | FRANCE         |           3 |
|           7 | GERMANY        |           3 |
|           8 | INDIA          |           2 |
|           9 | INDONESIA      |           2 |
|          10 | IRAN           |           4 |
|          11 | IRAQ           |           4 |
|          12 | JAPAN          |           2 |
|          13 | JORDAN         |           4 |
|          14 | KENYA          |           0 |
|          15 | MOROCCO        |           0 |
|          16 | MOZAMBIQUE     |           0 |
|          17 | PERU           |           1 |
|          18 | CHINA          |           2 |
|          19 | ROMANIA        |           3 |
|          20 | SAUDI ARABIA   |           4 |
|          21 | VIETNAM        |           2 |
|          22 | RUSSIA         |           3 |
|          23 | UNITED KINGDOM |           3 |
|          24 | UNITED STATES  |           1 |
+-------------+----------------+-------------+

MySQL > select n_regionkey, bitmap_to_string(bitmap_agg(n_nationkey)) from nation group by n_regionkey;
+-------------+-----------------------------------------------+
| n_regionkey | bitmap_to_string(bitmap_agg(`n_nationkey`)) |
+-------------+-----------------------------------------------+
|           4 | 4,10,11,13,20                                 |
|           2 | 8,9,12,18,21                                  |
|           1 | 1,2,3,17,24                                   |
|           0 | 0,5,14,15,16                                  |
|           3 | 6,7,19,22,23                                  |
```

```
+-------------+-------------------------------------------+

MySQL > select bitmap_count(bitmap_agg(n_nationkey))  from nation;
+----------------------------------------+
| bitmap_count(bitmap_agg(`n_nationkey`)) |
+----------------------------------------+
|                                     25 |
+----------------------------------------+
```

keywords

BITMAP_AGG

TO_BITMAP

to_bitmap

description

Syntax

```
BITMAP TO_BITMAP(expr)
```

Convert an unsigned bigint (ranging from 0 to 18446744073709551615) to a bitmap containing that value. Null will be return when the input value is not in this range. Mainly be used to load integer value into bitmap column, e.g.,

```
cat data | curl --location-trusted -u user:passwd -T - -H "columns: dt,page,user_id, user_id=to_bitmap(user_id)"
http://host:8410/api/test/testDb/_stream_load
```

example

```
mysql> select bitmap_count(to_bitmap(10));
+-----------------------------+
| bitmap_count(to_bitmap(10)) |
+-----------------------------+
|                           1 |
+-----------------------------+

MySQL> select bitmap_to_string(to_bitmap(-1));
+---------------------------------+
| bitmap_to_string(to_bitmap(-1)) |
+---------------------------------+
|                                 |
+---------------------------------+
```

keywords

```
TO_BITMAP,BITMAP
```

BITMAP_HASH

bitmap_hash

Name

BITMAP_HASH

Description

Calculating hash value for what your input and return a BITMAP which contain the hash value. MurMur3 is used for this function because it is high-performance with low collision rate. More important, the MurMur3 distribution is “simili-random”; the Chi-Square distribution test is used to prove it. By the way, Different hardware platforms and different SEED may change the result of MurMur3. For more information about its performance, see Smhasher.

Syntax

```
BITMAP BITMAP_HASH(<any_value>)
```

Arguments

`<any_value>` any value or expression.

Return Type

BITMAP

Remarks

Generally, MurMurHash 32 is friendly to random, short STRING with low collision rate about one-billionth. But for longer STRING, such as your path of system, can cause more frequent collision. If you indexed your system path, you will find a lot of collisions.

The following two values are the same.

```
SELECT
bitmap_to_string(bitmap_hash('/System/Volumes/Data/Library/Developer/CommandLineTools/SDKs/MacOSX12.3.sdk/System/Library/Frameworks/KernelManagement.framework/KernelManagement.tbd')) AS a ,

bitmap_to_string(bitmap_hash('/System/Library/PrivateFrameworks/Install.framework/Versions/Current/Resources/es_419.lproj/Architectures.strings')) AS b;
```

Here is the result.

```
+-----------+-----------+
| a         | b         |
+-----------+-----------+
| 282251871 | 282251871 |
+-----------+-----------+
```

Example

If you want to calculate MurMur3 of a certain value, you can

```
select bitmap_to_array(bitmap_hash('hello'))[1];
```

Here is the result.

```
+-------------------------------------------------------------+
| %element_extract%(bitmap_to_array(bitmap_hash('hello')), 1) |
+-------------------------------------------------------------+
|                                                  1321743225 |
+-------------------------------------------------------------+
```

If you want to `count distinct` some columns, using bitmap has higher performance in some scenes.

```
select bitmap_count(bitmap_union(bitmap_hash(`word`))) from `words`;
```

Here is the result.

```
+-------------------------------------------------+
| bitmap_count(bitmap_union(bitmap_hash(`word`))) |
+-------------------------------------------------+
|                                        33263478 |
+-------------------------------------------------+
```

Keywords

```
BITMAP_HASH,BITMAP
```

Best Practice

For more information, see also: - BITMAP_HASH64

BITMAP_FROM_STRING

bitmap_from_string

description

Syntax

```
BITMAP BITMAP_FROM_STRING(VARCHAR input)
```

Convert a string into a bitmap. The input string should be a comma separated unsigned bigint (ranging from 0 to 18446744073709551615). For example: input string “0, 1, 2” will be converted to a Bitmap with bit 0, 1, 2 set. If input string is invalid, return NULL.

example

```
mysql> select bitmap_to_string(bitmap_from_string("0, 1, 2"));
+------------------------------------------------+
| bitmap_to_string(bitmap_from_string('0, 1, 2')) |
+------------------------------------------------+
| 0,1,2                                          |
+------------------------------------------------+

mysql> select bitmap_from_string("-1, 0, 1, 2");
+----------------------------------+
| bitmap_from_string('-1, 0, 1, 2') |
+----------------------------------+
| NULL                             |
+----------------------------------+

mysql> select bitmap_to_string(bitmap_from_string("0, 1, 18446744073709551615"));
+-------------------------------------------------------------------+
| bitmap_to_string(bitmap_from_string('0, 1, 18446744073709551615')) |
+-------------------------------------------------------------------+
| 0,1,18446744073709551615                                          |
+-------------------------------------------------------------------+
```

keywords

```
BITMAP_FROM_STRING,BITMAP
```

BITMAP_TO_STRING

bitmap_to_string

description

Syntax

```
VARCHAR BITMAP_TO_STRING(BITMAP input)
```

Convert a input BITMAP to a string. The string is a separated string, contains all set bits in Bitmap. If input is null, return null.

example

```
mysql> select bitmap_to_string(null);
+------------------------+
| bitmap_to_string(NULL) |
+------------------------+
| NULL                   |
+------------------------+

mysql> select bitmap_to_string(bitmap_empty());
+----------------------------------+
| bitmap_to_string(bitmap_empty()) |
+----------------------------------+
|                                  |
+----------------------------------+

mysql> select bitmap_to_string(to_bitmap(1));
+--------------------------------+
| bitmap_to_string(to_bitmap(1)) |
+--------------------------------+
|  1                             |
+--------------------------------+

mysql> select bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2)));
+----------------------------------------------------------+
| bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2))) |
+----------------------------------------------------------+
|  1,2                                                     |
+----------------------------------------------------------+
```

keywords

```
BITMAP_TO_STRING,BITMAP
```

BITMAP_TO_ARRAY

bitmap_to_array

description

Syntax

```
ARRAY_BIGINT bitmap_to_array(BITMAP input)
```

Convert a input BITMAP to Array. If input is null, return null.

example

```
mysql> select bitmap_to_array(null);
+------------------------+
| bitmap_to_array(NULL)  |
+------------------------+
| NULL                   |
+------------------------+

mysql> select bitmap_to_array(bitmap_empty());
+---------------------------------+
| bitmap_to_array(bitmap_empty()) |
+---------------------------------+
| []                              |
+---------------------------------+

mysql> select bitmap_to_array(to_bitmap(1));
+------------------------------+
| bitmap_to_array(to_bitmap(1)) |
+------------------------------+
| [1]                          |
+------------------------------+

mysql> select bitmap_to_array(bitmap_from_string('1,2,3,4,5'));
+-------------------------------------------------+
| bitmap_to_array(bitmap_from_string('1,2,3,4,5')) |
+-------------------------------------------------+
| [1, 2, 3, 4, 5]                                 |
+-------------------------------------------------
```

keywords

```
BITMAP_TO_ARRAY,BITMAP
```

BITMAP_FROM_ARRAY

bitmap_from_array

description

Syntax

```
BITMAP BITMAP_FROM_ARRAY(ARRAY input)
```

Convert a TINYINT/SMALLINT/INT/BIGINT array to a BITMAP When the input field is illegal, the result returns NULL

example

```
mysql> select *, bitmap_to_string(bitmap_from_array(c_array)) from array_test;
+------+----------------------+-------------------------------------------------+
| id   | c_array              | bitmap_to_string(bitmap_from_array(`c_array`)) |
+------+----------------------+-------------------------------------------------+
|    1 | [NULL]               | NULL                                            |
|    2 | [1, 2, 3, NULL]      | NULL                                            |
|    2 | [1, 2, 3, -10]       | NULL                                            |
|    3 | [1, 2, 3, 4, 5, 6, 7] | 1,2,3,4,5,6,7                                  |
|    4 | [100, 200, 300, 300]  | 100,200,300                                    |
+------+----------------------+-------------------------------------------------+
5 rows in set (0.02 sec)
```

keywords

```
BITMAP_FROM_ARRAY,BITMAP
```

BITMAP_EMPTY

bitmap_empty

description

Syntax

```
BITMAP BITMAP_EMPTY()
```

Return an empty bitmap. Mainly be used to supply default value for bitmap column when loading, e.g.,

```
cat data | curl --location-trusted -u user:passwd -T - -H "columns: dt,page,v1,v2=bitmap_empty()"
http://host:8410/api/test/testDb/_stream_load
```

example

```
mysql> select bitmap_count(bitmap_empty());
+------------------------------+
| bitmap_count(bitmap_empty()) |
+------------------------------+
|                            0 |
+------------------------------+

mysql> select bitmap_to_string(bitmap_empty());
+----------------------------------+
| bitmap_to_string(bitmap_empty()) |
+----------------------------------+
|                                  |
+----------------------------------+
```

keywords

```
BITMAP_EMPTY,BITMAP
```

BITMAP_OR

bitmap_or

description

Syntax

```
BITMAP BITMAP_OR(BITMAP lhs, BITMAP rhs, ...)
```

Compute union of two or more input bitmaps, returns the new bitmap.

example

```
mysql> select bitmap_count(bitmap_or(to_bitmap(1), to_bitmap(1))) cnt;
+------+
| cnt  |
+------+
|    1 |
+------+

mysql> select bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(1))) ;
+-----------------------------------------------------+
| bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(1))) |
+-----------------------------------------------------+
| 1                                                   |
+-----------------------------------------------------+

mysql> select bitmap_count(bitmap_or(to_bitmap(1), to_bitmap(2))) cnt;
+------+
| cnt  |
+------+
|    2 |
+------+

mysql> select bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2)));
+-----------------------------------------------------+
| bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2))) |
+-----------------------------------------------------+
| 1,2                                                 |
+-----------------------------------------------------+

mysql> select bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2), to_bitmap(10), to_bitmap(0), NULL));
+-----------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2), to_bitmap(10), to_bitmap(0), NULL)) |
+-----------------------------------------------------------------------------------------+
| 0,1,2,10                                                                                |
+-----------------------------------------------------------------------------------------+

mysql> select bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2),to_bitmap(10),to_bitmap(0),bitmap_empty()));
+-----------------------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_or(to_bitmap(1), to_bitmap(2), to_bitmap(10), to_bitmap(0), bitmap_empty())) |
+-----------------------------------------------------------------------------------------------------+
| 0,1,2,10                                                                                            |
+-----------------------------------------------------------------------------------------------------+
```

```
mysql> select bitmap_to_string(bitmap_or(to_bitmap(10), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'))) ;
+-----------------------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_or(to_bitmap(10), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'))) |
+-----------------------------------------------------------------------------------------------------+
| 1,2,3,4,5,10                                                                                        |
+-----------------------------------------------------------------------------------------------------+
```

keywords

```
BITMAP_OR,BITMAP
```

BITMAP_AND

bitmap_and

description

Syntax

```
BITMAP BITMAP_AND(BITMAP lhs, BITMAP rhs, ...)
```

Compute intersection of two or more input bitmaps, return the new bitmap.

example

```
mysql> select bitmap_count(bitmap_and(to_bitmap(1), to_bitmap(2))) cnt;
+------+
| cnt  |
+------+
|    0 |
+------+

mysql> select bitmap_to_string(bitmap_and(to_bitmap(1), to_bitmap(2)));
+-----------------------------------------------------------+
| bitmap_to_string(bitmap_and(to_bitmap(1), to_bitmap(2))) |
+-----------------------------------------------------------+
|                                                           |
+-----------------------------------------------------------+

mysql> select bitmap_count(bitmap_and(to_bitmap(1), to_bitmap(1))) cnt;
+------+
| cnt  |
+------+
|    1 |
+------+

MySQL> select bitmap_to_string(bitmap_and(to_bitmap(1), to_bitmap(1)));
+-----------------------------------------------------------+
| bitmap_to_string(bitmap_and(to_bitmap(1), to_bitmap(1))) |
+-----------------------------------------------------------+
| 1                                                         |
+-----------------------------------------------------------+

MySQL> select bitmap_to_string(bitmap_and(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5')));
+-----------------------------------------------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_and(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'))) |
+-----------------------------------------------------------------------------------------------------------------------------+
| 1,2                                                                                                                         |
+-----------------------------------------------------------------------------------------------------------------------------+

MySQL> select bitmap_to_string(bitmap_and(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'),
bitmap_from_string('1,2,3,4,5'),bitmap_empty()));
+-----------------------------------------------------------------------------------------------------------------------------------------------
----+
| bitmap_to_string(bitmap_and(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'),
bitmap_empty())) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
```

```
----+
|
|
+-----------------------------------------------------------------------------------------------------------------------------------
----+

MySQL> select bitmap_to_string(bitmap_and(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'),
bitmap_from_string('1,2,3,4,5'),NULL));
+-----------------------------------------------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_and(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'), NULL)) |
+-----------------------------------------------------------------------------------------------------------------------------+
| NULL                                                                                                                        |
+-----------------------------------------------------------------------------------------------------------------------------+
```

keywords

```
BITMAP_AND,BITMAP
```

BITMAP_UNION

bitmap_union function

description

Aggregate function, used to calculate the grouped bitmap union. Common usage scenarios such as: calculating PV, UV.

Syntax

```
BITMAP BITMAP_UNION(BITMAP value)
```

Enter a set of bitmap values, find the union of this set of bitmap values, and return.

example

```
mysql> select page_id, bitmap_union(user_id) from table group by page_id;
```

Combined with the bitmap_count function, the PV data of the web page can be obtained

```
mysql> select page_id, bitmap_count(bitmap_union(user_id)) from table group by page_id;
```

When the user_id field is int, the above query semantics is equivalent to

```
mysql> select page_id, count(distinct user_id) from table group by page_id;
```

keywords

```
BITMAP_UNION, BITMAP
```

BITMAP_XOR

bitmap_xor

description

Syntax

```
BITMAP BITMAP_XOR(BITMAP lhs, BITMAP rhs, ...)
```

Compute the symmetric union of two or more input bitmaps, return the new bitmap.

example

```
mysql> select bitmap_count(bitmap_xor(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4'))) cnt;
+------+
| cnt  |
+------+
|    2 |
+------+

mysql> select bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4')));
+-----------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'))) |
+-----------------------------------------------------------------------------------+
| 1,4                                                                               |
+-----------------------------------------------------------------------------------+

MySQL> select bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4'),bitmap_from_string('3,4,5')));
+-----------------------------------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'), bitmap_from_string('3,4,5'))) |
+-----------------------------------------------------------------------------------------------------------------+
| 1,3,5                                                                                                           |
+-----------------------------------------------------------------------------------------------------------------+

MySQL> select
bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4'),bitmap_from_string('3,4,5'),bitmap_empty()));
+-------------------------------------------------------------------------------------------------------------------------------
--+
| bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'), bitmap_from_string('3,4,5'),
bitmap_empty())) |
+-------------------------------------------------------------------------------------------------------------------------------
--+
| 1,3,5
|
+-------------------------------------------------------------------------------------------------------------------------------
--+

MySQL> select
bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4'),bitmap_from_string('3,4,5'),NULL));
+-----------------------------------------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_xor(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'), bitmap_from_string('3,4,5'), NULL)) |
+-----------------------------------------------------------------------------------------------------------------------+
| NULL                                                                                                                  |
+-----------------------------------------------------------------------------------------------------------------------+
```

keywords

```
BITMAP_XOR,BITMAP
```

BITMAP_NOT

bitmap_not

description

Syntax

```
BITMAP BITMAP_NOT(BITMAP lhs, BITMAP rhs)
```

Calculate the set after lhs minus rhs, return the new bitmap.

example

```
mysql> select bitmap_to_string(bitmap_not(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4')));
+-----------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_not(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'))) |
+-----------------------------------------------------------------------------------+
|                                                                                   |
+-----------------------------------------------------------------------------------+

mysql> select bitmap_to_string(bitmap_not(bitmap_from_string('2,3,5'),bitmap_from_string('1,2,3,4')));
+-----------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_not(bitmap_from_string('2,3,5'), bitmap_from_string('1,2,3,4'))) |
+-----------------------------------------------------------------------------------+
| 5                                                                                 |
+-----------------------------------------------------------------------------------+
```

keywords

```
BITMAP_NOT,BITMAP
```

BITMAP_AND_NOT,BITMAP_ANDNOT

bitmap_and_not,bitmap_andnot

description

Syntax

```
BITMAP BITMAP_AND_NOT(BITMAP lhs, BITMAP rhs)
```

Calculate the set after lhs minus intersection of two input bitmaps, return the new bitmap.

example

```
mysql> select bitmap_count(bitmap_and_not(bitmap_from_string('1,2,3'),bitmap_from_string('3,4,5'))) cnt;
+------+
| cnt  |
+------+
|    2 |
+------+

mysql> select bitmap_to_string(bitmap_and_not(bitmap_from_string('1,2,3'),bitmap_from_string('3,4,5')));
+-----------------------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_and_not(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5'))) |
+-----------------------------------------------------------------------------------------------+
| 1,2                                                                                           |
+-----------------------------------------------------------------------------------------------+

mysql> select bitmap_to_string(bitmap_and_not(bitmap_from_string('1,2,3'),bitmap_empty())) ;
+-------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_and_not(bitmap_from_string('1,2,3'), bitmap_empty())) |
+-------------------------------------------------------------------------------+
| 1,2,3                                                                         |
+-------------------------------------------------------------------------------+

mysql> select bitmap_to_string(bitmap_and_not(bitmap_from_string('1,2,3'),NULL));
+-----------------------------------------------------------------+
| bitmap_to_string(bitmap_and_not(bitmap_from_string('1,2,3'), NULL)) |
+-----------------------------------------------------------------+
| NULL                                                            |
+-----------------------------------------------------------------+
```

keywords

```
BITMAP_AND_NOT,BITMAP_ANDNOT,BITMAP
```

BITMAP_SUBSET_LIMIT

bitmap_subset_limit

Description

Syntax

```
BITMAP BITMAP_SUBSET_LIMIT(BITMAP src, BIGINT range_start, BIGINT cardinality_limit)
```

Create subset of the BITMAP, begin with range from range_start, limit by cardinality_limit range_start: start value for the range cardinality_limit: subset upper limit

example

```
mysql> select bitmap_to_string(bitmap_subset_limit(bitmap_from_string('1,2,3,4,5'), 0, 3)) value;
+-----------+
| value     |
+-----------+
| 1,2,3 |
+-----------+

mysql> select bitmap_to_string(bitmap_subset_limit(bitmap_from_string('1,2,3,4,5'), 4, 3)) value;
+-------+
| value |
+-------+
| 4,5     |
+-------+
```

keywords

```
BITMAP_SUBSET_LIMIT,BITMAP_SUBSET,BITMAP
```

BITMAP_SUBSET_IN_RANGE

bitmap_subset_in_range

Description

Syntax

```
BITMAP BITMAP_SUBSET_IN_RANGE(BITMAP src, BIGINT range_start, BIGINT range_end)
```

Return subset in specified range (not include the range_end).

example

```
mysql> select bitmap_to_string(bitmap_subset_in_range(bitmap_from_string('1,2,3,4,5'), 0, 9)) value;
+-----------+
| value     |
+-----------+
| 1,2,3,4,5 |
+-----------+

mysql> select bitmap_to_string(bitmap_subset_in_range(bitmap_from_string('1,2,3,4,5'), 2, 3)) value;
+-------+
| value |
+-------+
| 2     |
+-------+
```

keywords

```
BITMAP_SUBSET_IN_RANGE,BITMAP_SUBSET,BITMAP
```

SUB_BITMAP

sub_bitmap

description

Syntax

```
BITMAP SUB_BITMAP(BITMAP src, BIGINT offset, BIGINT cardinality_limit)
```

Starting from the position specified by offset, intercept cardinality_limit bitmap elements and return a bitmap subset.

example

```
mysql> select bitmap_to_string(sub_bitmap(bitmap_from_string('1,0,1,2,3,1,5'), 0, 3)) value;
+-------+
| value |
+-------+
| 0,1,2 |
+-------+

mysql> select bitmap_to_string(sub_bitmap(bitmap_from_string('1,0,1,2,3,1,5'), -3, 2)) value;
+-------+
| value |
+-------+
| 2,3   |
+-------+

mysql> select bitmap_to_string(sub_bitmap(bitmap_from_string('1,0,1,2,3,1,5'), 2, 100)) value;
+-------+
| value |
+-------+
| 2,3,5 |
+-------+
```

keywords

```
SUB_BITMAP,BITMAP_SUBSET,BITMAP
```

BITMAP_COUNT

bitmap_count

description

Syntax

```
BITMAP BITMAP_COUNT(BITMAP lhs)
```

Returns the number of input bitmaps。

example

```
mysql> select bitmap_count(to_bitmap(1)) cnt;
+------+
| cnt  |
+------+
|    1 |
+------+

mysql> select bitmap_count(bitmap_and(to_bitmap(1), to_bitmap(1))) cnt;
+------+
| cnt  |
+------+
|    1 |
+------+
```

keywords

```
BITMAP_COUNT
```

BITMAP_AND_COUNT

bitmap_and_count

description

Syntax

```
BigIntVal bitmap_and_count(BITMAP lhs, BITMAP rhs, ...)
```

Calculate the intersection of two or more input bitmaps and return the number of intersections.

example

```
MySQL> select bitmap_and_count(bitmap_from_string('1,2,3'),bitmap_empty());
+-----------------------------------------------------------+
| bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_empty()) |
+-----------------------------------------------------------+
|                                                         0 |
+-----------------------------------------------------------+


MySQL> select bitmap_and_count(bitmap_from_string('1,2,3'),bitmap_from_string('1,2,3'));
+-------------------------------------------------------------------------+
| bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2,3')) |
+-------------------------------------------------------------------------+
|                                                                       3 |
+-------------------------------------------------------------------------+

MySQL> select bitmap_and_count(bitmap_from_string('1,2,3'),bitmap_from_string('3,4,5'));
+-------------------------------------------------------------------------+
| bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5')) |
+-------------------------------------------------------------------------+
|                                                                       1 |
+-------------------------------------------------------------------------+

MySQL> select bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'));
+-----------------------------------------------------------------------------------------------------------+
| (bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'))) |
+-----------------------------------------------------------------------------------------------------------+
|                                                                                                         2 |
+-----------------------------------------------------------------------------------------------------------+

MySQL> select bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'),
bitmap_from_string('1,2,3,4,5'),bitmap_empty());
+-----------------------------------------------------------------------------------------------------------------------------+
| (bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'), bitmap_empty())) |
+-----------------------------------------------------------------------------------------------------------------------------+
|                                                                                                                           0 |
+-----------------------------------------------------------------------------------------------------------------------------+

MySQL> select bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'), NULL);
+-----------------------------------------------------------------------------------------------------------+
| (bitmap_and_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2'), bitmap_from_string('1,2,3,4,5'), NULL)) |
+-----------------------------------------------------------------------------------------------------------+
```

```
|                                                
```

BITMAP_AND_NOT_COUNT,BITMAP_ANDNOT_COUNT

bitmap_and_not_count,bitmap_andnot_count

description

Syntax

```
BITMAP BITMAP_AND_NOT_COUNT(BITMAP lhs, BITMAP rhs)
```

Calculate the set after lhs minus intersection of two input bitmaps, return the new bitmap size.

example

```
mysql> select bitmap_and_not_count(bitmap_from_string('1,2,3'),bitmap_from_string('3,4,5')) cnt;
+------+
| cnt  |
+------+
|    2 |
+------+
```

keywords

```
BITMAP_AND_NOT_COUNT,BITMAP_ANDNOT_COUNT,BITMAP
```

BITMAP_XOR_COUNT

bitmap_xor_count

description

Syntax

```
BIGINT BITMAP_XOR_COUNT(BITMAP lhs, BITMAP rhs, ...)
```

XOR two or more bitmap sets and return the size of the result set.

example

```
mysql> select bitmap_xor_count(bitmap_from_string('1,2,3'),bitmap_from_string('3,4,5'));
+-----------------------------------------------------------------------+
| bitmap_xor_count(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5')) |
+-----------------------------------------------------------------------+
|                                                                     4 |
+-----------------------------------------------------------------------+

mysql> select bitmap_xor_count(bitmap_from_string('1,2,3'),bitmap_from_string('1,2,3'));
+-----------------------------------------------------------------------+
| bitmap_xor_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2,3')) |
+-----------------------------------------------------------------------+
|                                                                     0 |
+-----------------------------------------------------------------------+

mysql> select bitmap_xor_count(bitmap_from_string('1,2,3'),bitmap_from_string('4,5,6'));
+-----------------------------------------------------------------------+
| bitmap_xor_count(bitmap_from_string('1,2,3'), bitmap_from_string('4,5,6')) |
+-----------------------------------------------------------------------+
|                                                                     6 |
+-----------------------------------------------------------------------+

MySQL> select (bitmap_xor_count(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4'),bitmap_from_string('3,4,5')));
+-------------------------------------------------------------------------------------------------------+
| (bitmap_xor_count(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'), bitmap_from_string('3,4,5'))) |
+-------------------------------------------------------------------------------------------------------+
|                                                                                                     3 |
+-------------------------------------------------------------------------------------------------------+

MySQL> select
(bitmap_xor_count(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4'),bitmap_from_string('3,4,5'),bitmap_empty()));
+-------------------------------------------------------------------------------------------------------------------------+
| (bitmap_xor_count(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'), bitmap_from_string('3,4,5'), bitmap_empty())) |
+-------------------------------------------------------------------------------------------------------------------------+
|                                                                                                                       3 |
+-------------------------------------------------------------------------------------------------------------------------+

MySQL> select (bitmap_xor_count(bitmap_from_string('2,3'),bitmap_from_string('1,2,3,4'),bitmap_from_string('3,4,5'),NULL));
+-------------------------------------------------------------------------------------------------------+
| (bitmap_xor_count(bitmap_from_string('2,3'), bitmap_from_string('1,2,3,4'), bitmap_from_string('3,4,5'), NULL)) |
+-------------------------------------------------------------------------------------------------------+
```

```
|                                                                                                                  NULL |
+-------------------------------------------------------------------------------------------------------------------+
```

keywords

```
BITMAP_XOR_COUNT,BITMAP
```

BITMAP_OR_COUNT

bitmap_or_count

description

Syntax

```
BigIntVal bitmap_or_count(BITMAP lhs, BITMAP rhs)
```

Calculates the union of two or more input bitmaps and returns the number of union sets.

example

```
MySQL> select bitmap_or_count(bitmap_from_string('1,2,3'),bitmap_empty());
+-----------------------------------------------------------+
| bitmap_or_count(bitmap_from_string('1,2,3'), bitmap_empty()) |
+-----------------------------------------------------------+
|                                                         3 |
+-----------------------------------------------------------+


MySQL> select bitmap_or_count(bitmap_from_string('1,2,3'),bitmap_from_string('1,2,3'));
+-------------------------------------------------------------------------+
| bitmap_or_count(bitmap_from_string('1,2,3'), bitmap_from_string('1,2,3')) |
+-------------------------------------------------------------------------+
|                                                                       3 |
+-------------------------------------------------------------------------+

MySQL> select bitmap_or_count(bitmap_from_string('1,2,3'),bitmap_from_string('3,4,5'));
+-------------------------------------------------------------------------+
| bitmap_or_count(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5')) |
+-------------------------------------------------------------------------+
|                                                                       5 |
+-------------------------------------------------------------------------+

MySQL> select bitmap_or_count(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5'), to_bitmap(100), bitmap_empty());
+-----------------------------------------------------------------------------------------------------------+
| bitmap_or_count(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5'), to_bitmap(100), bitmap_empty()) |
+-----------------------------------------------------------------------------------------------------------+
|                                                                                                         6 |
+-----------------------------------------------------------------------------------------------------------+

MySQL> select bitmap_or_count(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5'), to_bitmap(100), NULL);
+-----------------------------------------------------------------------------------------------+
| bitmap_or_count(bitmap_from_string('1,2,3'), bitmap_from_string('3,4,5'), to_bitmap(100), NULL) |
+-----------------------------------------------------------------------------------------------+
|                                                                                          NULL |
+-----------------------------------------------------------------------------------------------+
```

keywords

```
BITMAP_OR_COUNT,BITMAP
```

BITMAP_CONTAINS

bitmap_contains

description

Syntax

```
BOOLEAN BITMAP_CONTAINS(BITMAP bitmap, BIGINT input)
```

Calculates whether the input value is in the Bitmap column and returns a Boolean value.

example

```
mysql> select bitmap_contains(to_bitmap(1),2) cnt;
+------+
| cnt  |
+------+
|    0 |
+------+

mysql> select bitmap_contains(to_bitmap(1),1) cnt;
+------+
| cnt  |
+------+
|    1 |
+------+
```

keywords

```
BITMAP_CONTAINS,BITMAP
```

BITMAP_HAS_ALL

bitmap_has_all

description

Syntax

```
BOOLEAN BITMAP_HAS_ALL(BITMAP lhs, BITMAP rhs)
```

Returns true if the first bitmap contains all the elements of the second bitmap. Returns true if the second bitmap contains an empty element.

example

```
mysql> select bitmap_has_all(bitmap_from_string("0, 1, 2"), bitmap_from_string("1, 2"));
+-----------------------------------------------------------------------------+
| bitmap_has_all(bitmap_from_string('0, 1, 2'), bitmap_from_string('1, 2')) |
+-----------------------------------------------------------------------------+
|                                                                           1 |
+-----------------------------------------------------------------------------+

mysql> select bitmap_has_all(bitmap_empty(), bitmap_from_string("1, 2"));
+-----------------------------------------------------------+
| bitmap_has_all(bitmap_empty(), bitmap_from_string('1, 2')) |
+-----------------------------------------------------------+
|                                                         0 |
+-----------------------------------------------------------+
```

keywords

```
BITMAP_HAS_ALL,BITMAP
```

BITMAP_HAS_ANY

bitmap_has_any

description

Syntax

```
BOOLEAN BITMAP_HAS_ANY(BITMAP lhs, BITMAP rhs)
```

Calculate whether there are intersecting elements in the two Bitmap columns. The return value is Boolean.

example

```
mysql> select bitmap_has_any(to_bitmap(1),to_bitmap(2));
+-------------------------------------------+
| bitmap_has_any(to_bitmap(1), to_bitmap(2)) |
+-------------------------------------------+
|                                         0 |
+-------------------------------------------+

mysql> select bitmap_has_any(to_bitmap(1),to_bitmap(1));
+-------------------------------------------+
| bitmap_has_any(to_bitmap(1), to_bitmap(1)) |
+-------------------------------------------+
|                                         1 |
+-------------------------------------------+
```

keywords

```
BITMAP_HAS_ANY,BITMAP
```

BITMAP_MAX

bitmap_max

description

Syntax

```
BIGINT BITMAP_MAX(BITMAP input)
```

Calculate and return the max values of a bitmap.

example

```
mysql> select bitmap_max(bitmap_from_string('')) value;
+-------+
| value |
+-------+
|  NULL |
+-------+

mysql> select bitmap_max(bitmap_from_string('1,9999999999')) value;
+------------+
| value      |
+------------+
| 9999999999 |
+------------+
```

keywords

```
BITMAP_MAX,BITMAP
```

BITMAP_MIN

bitmap_min

description

Syntax

```
BIGINT BITMAP_MIN(BITMAP input)
```

Calculate and return the min values of a bitmap.

example

```
mysql> select bitmap_min(bitmap_from_string('')) value;
+-------+
| value |
+-------+
|  NULL |
+-------+

mysql> select bitmap_min(bitmap_from_string('1,9999999999')) value;
+-------+
| value |
+-------+
|     1 |
+-------+
```

keywords

```
BITMAP_MIN,BITMAP
```

INTERSECT_COUNT

intersect_count

description

Syntax

`BITMAP INTERSECT_COUNT(bitmap_column, column_to_filter, filter_values)` Calculate the intersection of two or more bitmaps Usage: intersect_count(bitmap_column_to_count, filter_column, filter_values ...) Example: intersect_count(user_id, event, 'A', 'B', 'C'), meaning find the intersect count of user_id in all A/B/C 3 bitmaps

example

```
MySQL [test_query_qa]> select dt,bitmap_to_string(user_id) from pv_bitmap where dt in (3,4);
+------+----------------------------+
| dt   | bitmap_to_string(`user_id`) |
+------+----------------------------+
|    4 | 1,2,3                      |
|    3 | 1,2,3,4,5                  |
+------+----------------------------+
2 rows in set (0.012 sec)

MySQL [test_query_qa]> select intersect_count(user_id,dt,3,4) from pv_bitmap;
+---------------------------------------+
| intersect_count(`user_id`, `dt`, 3, 4) |
+---------------------------------------+
|                                     3 |
+---------------------------------------+
1 row in set (0.014 sec)
```

keywords

```
INTERSECT_COUNT,BITMAP
```

BITMAP_INTERSECT

bitmap_intersect

description

Aggregation function, used to calculate the bitmap intersection after grouping. Common usage scenarios such as: calculating user retention rate.

Syntax

```
BITMAP BITMAP_INTERSECT(BITMAP value)
```

Enter a set of bitmap values, find the intersection of the set of bitmap values, and return.

example

Table schema

```
KeysType: AGG_KEY
Columns: tag varchar, date datetime, user_id bitmap bitmap_union
```

```
Find the retention of users between 2020-05-18 and 2020-05-19 under different tags.
mysql> select tag, bitmap_intersect(user_id) from (select tag, date, bitmap_union(user_id) user_id from table where date in ('2020-
05-18', '2020-05-19') group by tag, date) a group by tag;
```

Used in combination with the bitmap_to_string function to obtain the specific data of the intersection

```
Who are the users retained under different tags between 2020-05-18 and 2020-05-19?
mysql> select tag, bitmap_to_string(bitmap_intersect(user_id)) from (select tag, date, bitmap_union(user_id) user_id from table
where date in ('2020-05-18', '2020-05-19') group by tag, date) a group by tag;
```

keywords

```
BITMAP_INTERSECT, BITMAP
```

BITMAP_HASH64

bitmap_hash64

description

Syntax

```
BITMAP BITMAP_HASH64(expr)
```

Compute the 64-bits hash value of a expr of any type, then return a bitmap containing that hash value. Mainly be used to load non-integer value into bitmap column, e.g.,

```
cat data | curl --location-trusted -u user:passwd -T - -H "columns: dt,page,device_id, device_id=bitmap_hash64(device_id)"
http://host:8410/api/test/testDb/_stream_load
```

example

```
mysql> select bitmap_to_string(bitmap_hash64('hello'));
+-----------------------------------------+
| bitmap_to_string(bitmap_hash64('hello')) |
+-----------------------------------------+
| 15231136565543391023                    |
+-----------------------------------------+
```

keywords

```
BITMAP_HASH,BITMAP
```

BITMAP_FROM_BASE64

bitmap_from_base64

description

Syntax

```
BITMAP BITMAP_FROM_BASE64(VARCHAR input)
```

Convert a base64 string(result of function `bitmap_to_base64`) into a bitmap. If input string is invalid, return NULL.

example

```
mysql> select bitmap_to_string(bitmap_from_base64("AA=="));
+---------------------------------------------+
| bitmap_to_string(bitmap_from_base64("AA==")) |
+---------------------------------------------+
|                                             |
+---------------------------------------------+

mysql> select bitmap_to_string(bitmap_from_base64("AQEAAAA="));
+-----------------------------------+
| bitmap_to_string(bitmap_from_base64("AQEAAAA=")) |
+-----------------------------------+
| 1                                 |
+-----------------------------------+

mysql> select bitmap_to_string(bitmap_from_base64("AjowAAACAAAAAAAAAJgAAAAYAAAAGgAAAAEAf5Y="));
+-------------------------------------------------------------------------------+
| bitmap_to_string(bitmap_from_base64("AjowAAACAAAAAAAAAJgAAAAYAAAAGgAAAAEAf5Y=")) |
+-------------------------------------------------------------------------------+
| 1,9999999                                                                     |
+-------------------------------------------------------------------------------+
```

keywords

```
BITMAP_FROM_BASE64,BITMAP
```

BITMAP_TO_BASE64

bitmap_to_base64

description

Syntax

```
VARCHAR BITMAP_TO_BASE64(BITMAP input)
```

Convert an input BITMAP to a base64 string. If input is null, return null. Since BE config item `enable_set_in_bitmap_value` will change the format of bitmap value in memory, it also affect the result of this function.

example

```
mysql> select bitmap_to_base64(null);
+------------------------+
| bitmap_to_base64(NULL) |
+------------------------+
| NULL                   |
+------------------------+

mysql> select bitmap_to_base64(bitmap_empty());
+----------------------------------+
| bitmap_to_base64(bitmap_empty()) |
+----------------------------------+
| AA==                             |
+----------------------------------+

mysql> select bitmap_to_base64(to_bitmap(1));
+--------------------------------+
| bitmap_to_base64(to_bitmap(1)) |
+--------------------------------+
| AQEAAAA=                       |
+--------------------------------+

mysql> select bitmap_to_base64(bitmap_from_string("1,9999999"));
+-----------------------------------------------------------+
| bitmap_to_base64(bitmap_from_string("1,9999999"))         |
+-----------------------------------------------------------+
| AjowAAACAAAAAAAAAJgAAAAYAAAAGgAAAAEAf5Y=                  |
+-----------------------------------------------------------+
```

keywords

```
BITMAP_TO_BASE64,BITMAP
```

BITMAP_REMOVE

bitmap_remove

description

Syntax

```
BITMAP BITMAP_REMOVE(BITMAP bitmap, BIGINT input)
```

Remove the specified value from bitmap.

example

```
mysql [(none)]>select bitmap_to_string(bitmap_remove(bitmap_from_string('1, 2, 3'), 3)) res;
+------+
| res  |
+------+
| 1,2  |
+------+

mysql [(none)]>select bitmap_to_string(bitmap_remove(bitmap_from_string('1, 2, 3'), null)) res;
+------+
| res  |
+------+
| NULL |
+------+
```

keywords

```
BITMAP_REMOVE,BITMAP
```

BITAND

bitand

description

Syntax

```
BITAND(Integer-type lhs, Integer-type rhs)
```

Returns the result of the AND operation of two integers.

Integer range: TINYINT、SMALLINT、INT、BIGINT、LARGEINT

example

```
mysql> select bitand(3,5) ans;
+------+
| ans  |
+------+
|    1 |
+------+

mysql> select bitand(4,7) ans;
+------+
| ans  |
+------+
|    4 |
+------+
```

keywords

```
BITAND
```

BITOR

bitor

description

Syntax

```
BITOR(Integer-type lhs, Integer-type rhs)
```

Returns the result of the OR operation of two integers.

Integer range: TINYINT、SMALLINT、INT、BIGINT、LARGEINT

example

```
mysql> select bitor(3,5) ans;
+------+
| ans  |
+------+
|    7 |
+------+

mysql> select bitor(4,7) ans;
+------+
| ans  |
+------+
|    7 |
+------+
```

keywords

```
BITOR
```

BITXOR

bitxor

description

Syntax

```
BITXOR(Integer-type lhs, Integer-type rhs)
```

Returns the result of the XOR operation of two integers.

Integer range: TINYINT、SMALLINT、INT、BIGINT、LARGEINT

example

```
mysql> select bitxor(3,5) ans;
+------+
| ans  |
+------+
|    7 |
+------+

mysql> select bitxor(1,7) ans;
+------+
| ans  |
+------+
|    6 |
+------+
```

keywords

```
BITXOR
```

BITNOT

bitnot

description

Syntax

```
BITNOT(Integer-type value)
```

Returns the result of the NOT operation of one integer.

Integer range: TINYINT、SMALLINT、INT、BIGINT、LARGEINT

example

```
mysql> select bitnot(7) ans;
+------+
| ans  |
+------+
|   -8 |
+------+

mysql> select bitxor(-127) ans;
+------+
| ans  |
+------+
|  126 |
+------+
```

keywords

```
BITNOT
```

CASE

case

description

Syntax

```
CASE expression
    WHEN condition1 THEN result1
    [WHEN condition2 THEN result2]
    ...
    [WHEN conditionN THEN resultN]
    [ELSE result]
END
```

OR

```
CASE WHEN condition1 THEN result1
    [WHEN condition2 THEN result2]
    ...
    [WHEN conditionN THEN resultN]
    [ELSE result]
END
```

Compare the expression with multiple possible values, and return the corresponding results when matching

example

```
mysql> select user_id, case user_id when 1 then 'user_id = 1' when 2 then 'user_id = 2' else 'user_id not exist' end test_case from
test;
+---------+-------------+
| user_id | test_case   |
+---------+-------------+
| 1       | user_id = 1 |
| 2       | user_id = 2 |
+---------+-------------+

mysql> select user_id, case when user_id = 1 then 'user_id = 1' when user_id = 2 then 'user_id = 2' else 'user_id not exist' end
test_case from test;
+---------+-------------+
| user_id | test_case   |
+---------+-------------+
| 1       | user_id = 1 |
| 2       | user_id = 2 |
+---------+-------------+
```

keywords

CASE

COALESCE

Description

Returns the first non-null expression from left to right in the parameter list. If all arguments are NULL, returns NULL.

Syntax

```
COALESCE(<expr> [, ...])
```

Parameters

Parameter	Description
<expr>	A sequence of expressions to evaluate. All expressions must be compatible data types.

Return Value

The first non-null expression in the parameter list. Returns NULL if all arguments are NULL.

Examples

```
SELECT COALESCE(NULL, '1111', '0000');
```

```
+--------------------------------+
| coalesce(NULL, '1111', '0000') |
+--------------------------------+
| 1111                           |
+--------------------------------+
```

IF

Description

Returns `valueTrue` when the condition is true, and returns `valueFalseOrNull` otherwise. The return type is determined by the result of the `valueTrue`/`valueFalseOrNull` expression.

Syntax

```
IF(<condition>, <value_true>, <value_false_or_null>)
```

Parameters

Parameter	Description
`<condition>`	The boolean condition to evaluate.
`<value_true>`	The value to return if `<condition>` evaluates to true.
`<value_false_or_null>`	The value to return if `<condition>` evaluates to false.

Return Value

The result of the IF expression: - Returns `valueTrue` when the condition is true. - Returns `valueFalseOrNull` when the condition is false.

Examples

```
SELECT user_id, IF(user_id = 1, 'true', 'false') AS test_if FROM test;
```

```
+---------+---------+
| user_id | test_if |
+---------+---------+
| 1       | true    |
| 2       | false   |
+---------+---------+
```

IFNULL

Description

Returns <expr1> if it is not NULL; otherwise, returns <expr2>.

Alias

- NVL

Syntax

```
IFNULL(<expr1>, <expr2>)
```

Parameters

Parameter	Description
<expr1>	The first expression to check for NULL.
<expr2>	The value to return if <expr1> is NULL.

Return Value

- Returns <expr1> if it is not NULL.
- Otherwise, returns <expr2>.

Examples

```
SELECT IFNULL(1, 0);
```

```
+--------------+
| IFNULL(1, 0) |
+--------------+
|            1 |
+--------------+
```

```
SELECT IFNULL(NULL, 10);
```

```
+------------------+
| IFNULL(NULL, 10) |
+------------------+
|               10 |
+------------------+
```

NVL

nvl

nvl

description

Syntax

```
nvl(expr1, expr2)
```

If the value of expr1 is not null, expr1 is returned, otherwise expr2 is returned

example

```
mysql> select nvl(1,0);
+--------------+
| nvl(1, 0) |
+--------------+
|            1 |
+--------------+

mysql> select nvl(null,10);
+------------------+
| nvl(NULL, 10) |
+------------------+
|               10 |
+------------------+
```

keywords

NVL

NULLIF

Description

Returns `NULL` if the two input values are equal; otherwise, returns the first input value. This function is equivalent to the following `CASE WHEN` expression:

```
CASE
    WHEN <expr1> = <expr2> THEN NULL
    ELSE <expr1>
END
```

Syntax

```
NULLIF(<expr1>, <expr2>)
```

Parameters

Parameter	Description
`<expr1>`	The first input value to compare.
`<expr2>`	The second input value to compare against the first.

Return Value

- Returns `NULL` if `<expr1>` is equal to `<expr2>`.
- Otherwise, returns the value of `<expr1>`.

Examples

```
SELECT NULLIF(1, 1);
```

```
+--------------+
| NULLIF(1, 1) |
+--------------+
|         NULL |
+--------------+
```

```
SELECT NULLIF(1, 0);
```

```
+--------------+
| NULLIF(1, 0) |
+--------------+
```

```
|           1 |
+-------------+
```

JSON_PARSE

json_parse

description

Syntax

```
JSON json_parse(VARCHAR json_str)
JSON json_parse_error_to_null(VARCHAR json_str)
JSON json_parse_error_to_value(VARCHAR json_str, VARCHAR default_json_str)
```

json_parse functions parse JSON string to binary format. A series of functions are provided to satisfy different demand for exception handling. - all return NULL if json_str is NULL - if json_str is not valid - json_parse will report error - json_parse_error_to_null will return NULL - json_parse_error_to_value will return the value specified by default_json_str

example

1. parse valid JSON string

```
mysql> SELECT json_parse('{"k1":"v31","k2":300}');
+---------------------------------------+
| json_parse('{"k1":"v31","k2":300}') |
+---------------------------------------+
| {"k1":"v31","k2":300}                 |
+---------------------------------------+
1 row in set (0.01 sec)
```

2. parse invalid JSON string

```
mysql> SELECT json_parse('invalid json');
ERROR 1105 (HY000): errCode = 2, detailMessage = json parse error: Invalid document: document must be an object or an array for
value: invalid json

mysql> SELECT json_parse_error_to_null('invalid json');
+-----------------------------------------+
| json_parse_error_to_null('invalid json') |
+-----------------------------------------+
| NULL                                    |
+-----------------------------------------+
1 row in set (0.01 sec)

mysql> SELECT json_parse_error_to_value('invalid json', '{}');
+-------------------------------------------------+
| json_parse_error_to_value('invalid json', '{}') |
+-------------------------------------------------+
| {}                                              |
+-------------------------------------------------+
1 row in set (0.00 sec)
```

refer to json tutorial for more.

keywords

JSONB, JSON, json_parse, json_parse_error_to_null, json_parse_error_to_value

JSON_EXTRACT

json_extract

description

Syntax

```
`VARCHAR json_extract(VARCHAR json_str, VARCHAR path[, VARCHAR path] ...))`
JSON jsonb_extract(JSON j, VARCHAR json_path)
BOOLEAN json_extract_isnull(JSON j, VARCHAR json_path)
BOOLEAN json_extract_bool(JSON j, VARCHAR json_path)
INT json_extract_int(JSON j, VARCHAR json_path)
BIGINT json_extract_bigint(JSON j, VARCHAR json_path)
LARGEINT json_extract_largeint(JSON j, VARCHAR json_path)
DOUBLE json_extract_double(JSON j, VARCHAR json_path)
STRING json_extract_string(JSON j, VARCHAR json_path)
```

json_extract functions extract field specified by json_path from JSON. A series of functions are provided for different datatype. - json_extract with VARCHAR argument, extract and return VARCHAR datatype - jsonb_extract extract and return JSON datatype - json_extract_isnull check if the field is json null and return BOOLEAN datatype - json_extract_bool extract and return BOOLEAN datatype - json_extract_int extract and return INT datatype - json_extract_bigint extract and return BIGINT datatype - json_extract_largeint extract and return LARGEINT datatype - json_extract_double extract and return DOUBLE datatype - json_extract_STRING extract and return STRING datatype

json path syntax: - ‘$' for json document root - '.k1' for element of json object with key 'k1' - If the key column value contains ".", double quotes are required in json_path, For example: SELECT json_extract('{"k1.a":"abc","k2":300}', '$.”k1.a”'); - ‘[i]’ for element of json array at index i - Use ‘ $[last]' to get the last element of json_array, and' [last-1]’ to get the penultimate element, and so on.

Exception handling is as follows: - if the field specified by json_path does not exist, return NULL - if datatype of the field specified by json_path is not the same with type of json_extract_t, return t if it can be cast to t else NULL

json_exists_path and json_type

description

Syntax

```
BOOLEAN json_exists_path(JSON j, VARCHAR json_path)
STRING json_type(JSON j, VARCHAR json_path)
```

There are two extra functions to check field existence and type - json_exists_path check the existence of the field specified by json_path, return TRUE or FALS - json_type get the type as follows of the field specified by

json_path, return NULL if it does not exist - object - array - null - bool - int - bigint - largeint - double - string

example

refer to json tutorial for more.

```
mysql> SELECT json_extract('{"id": 123, "name": "doris"}', '$.id');
+-----------------------------------------------------+
| json_extract('{"id": 123, "name": "doris"}', '$.id') |
+-----------------------------------------------------+
| 123                                                 |
+-----------------------------------------------------+
1 row in set (0.01 sec)

mysql> SELECT json_extract('[1, 2, 3]', '$.[1]');
+-----------------------------------+
| json_extract('[1, 2, 3]', '$.[1]') |
+-----------------------------------+
| 2                                 |
+-----------------------------------+
1 row in set (0.01 sec)

mysql> SELECT json_extract('{"k1": "v1", "k2": { "k21": 6.6, "k22": [1, 2] } }', '$.k1', '$.k2.k21', '$.k2.k22', '$.k2.k22[1]');
+-----------------------------------------------------------------------------------------------------------------------------+
| json_extract('{"k1": "v1", "k2": { "k21": 6.6, "k22": [1, 2] } }', '$.k1', '$.k2.k21', '$.k2.k22', '$.k2.k22[1]') |
+-----------------------------------------------------------------------------------------------------------------------------+
| ["v1",6.6,[1,2],2]                                                                                                          |
+-----------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql> SELECT json_extract('{"id": 123, "name": "doris"}', '$.aaa', '$.name');
+-----------------------------------------------------------------+
| json_extract('{"id": 123, "name": "doris"}', '$.aaa', '$.name') |
+-----------------------------------------------------------------+
| [null,"doris"]                                                  |
+-----------------------------------------------------------------+
1 row in set (0.01 sec)
```

keywords

JSONB, JSON, json_extract, json_extract_isnull, json_extract_bool, json_extract_int, json_extract_bigint, json_extract_largeint,json_extract_double, json_extract_string, json_exists_path, json_type

JSON_EXISTS_PATH

json_exists_path

description

It is used to judge whether the field specified by json_path exists in the JSON data. If it exists, it returns TRUE, and if it does not exist, it returns FALSE

Syntax

```
BOOLEAN json_exists_path(JSON j, VARCHAR json_path)
```

example

Refer to json tutorial

keywords

json_exists_path

JSON_TYPE

json_type

description

It is used to determine the type of the field specified by json_path in JSON data. If the field does not exist, return NULL. If it exists, return one of the following types

- object
- array
- null
- bool
- int
- bigint
- largeint
- double
- string

Syntax

```
STRING json_type(JSON j, VARCHAR json_path)
```

example

Refer to json tutorial

keywords

json_type

JSON_ARRAY

json_array

Description

Syntax

```
VARCHAR json_array(VARCHAR,...)
```

Generate a json array containing the specified values, return empty if no values

example

```
MySQL> select json_array();
+--------------+
| json_array() |
+--------------+
| []           |
+--------------+

MySQL> select json_array(null);
+--------------------+
| json_array('NULL') |
+--------------------+
| [NULL]             |
+--------------------+


MySQL> SELECT json_array(1, "abc", NULL, TRUE, CURTIME());
+-----------------------------------------------+
| json_array(1, 'abc', 'NULL', TRUE, curtime()) |
+-----------------------------------------------+
| [1, "abc", NULL, TRUE, "10:41:15"]            |
+-----------------------------------------------+


MySQL> select json_array("a", null, "c");
+-----------------------------+
| json_array('a', 'NULL', 'c') |
+-----------------------------+
| ["a", NULL, "c"]            |
+-----------------------------+
```

keywords

json,array,json_array

JSON_OBJECT

json_object

Description

Syntax

```
VARCHAR json_object(VARCHAR,...)
```

Generate a json object containing the specified Key-Value, an exception error is returned when Key is NULL or the number of parameters are odd.

example

```
MySQL> select json_object();
+---------------+
| json_object() |
+---------------+
| {}            |
+---------------+

MySQL> select json_object('time',curtime());
+---------------------------------+
| json_object('time', curtime()) |
+---------------------------------+
| {"time": "10:49:18"}           |
+---------------------------------+


MySQL> SELECT json_object('id', 87, 'name', 'carrot');
+----------------------------------------+
| json_object('id', 87, 'name', 'carrot') |
+----------------------------------------+
| {"id": 87, "name": "carrot"}            |
+----------------------------------------+


MySQL> select json_object('username',null);
+---------------------------------+
| json_object('username', 'NULL') |
+---------------------------------+
| {"username": NULL}              |
+---------------------------------+
```

keywords

json,object,json_object

JSON_QUOTE

json_quote

Description

Syntax

```
VARCHAR json_quote(VARCHAR)
```

Enclose json_value in double quotes (“), escape special characters contained.

example

```
MySQL> SELECT json_quote('null'), json_quote('"null"');
+--------------------+----------------------+
| json_quote('null') | json_quote('"null"') |
+--------------------+----------------------+
| "null"             | "\"null\""           |
+--------------------+----------------------+


MySQL> SELECT json_quote('[1, 2, 3]');
+-------------------------+
| json_quote('[1, 2, 3]') |
+-------------------------+
| "[1, 2, 3]"             |
+-------------------------+


MySQL> SELECT json_quote(null);
+------------------+
| json_quote(null) |
+------------------+
| NULL             |
+------------------+

MySQL> select json_quote("\n\b\r\t");
+------------------------+
| json_quote('\n\b\r\t') |
+------------------------+
| "\n\b\r\t"             |
+------------------------+
```

keywords

json,quote,json_quote

JSON_UNQUOTE

json_unquote

Description

Syntax

```
VARCHAR json_unquote(VARCHAR)
```

This function unquotes a JSON value and returns the result as a utf8mb4 string. If the argument is NULL, it will return NULL.

Escape sequences within a string as shown in the following table will be recognized. Backslashes will be ignored for all other escape sequences.

Escape Sequence	Character Represented by Sequence
"	A double quote (“) character
A backspace character	
A formfeed character	
	A newline (linefeed) character
A carriage return character	
A tab character	
\	A backslash () character
	UTF-8 bytes for Unicode value XXXX

example

```
mysql> SELECT json_unquote('"doris"');
+-------------------------+
| json_unquote('"doris"') |
+-------------------------+
| doris                   |
+-------------------------+

mysql> SELECT json_unquote('[1, 2, 3]');
+--------------------------+
| json_unquote('[1, 2, 3]') |
+--------------------------+
| [1, 2, 3]                |
+--------------------------+


mysql> SELECT json_unquote(null);
+-------------------+
| json_unquote(NULL) |
+-------------------+
```

```
| NULL              |
+-------------------+

mysql> SELECT json_unquote('"\\ttest"');
+--------------------------+
| json_unquote('"\ttest"') |
+--------------------------+
|       test                   |
+--------------------------+
```

keywords

json,unquote,json_unquote

JSON_VALID

json_valid

description

json_valid functions returns 0 or 1 to indicate whether a value is valid JSON and Returns NULL if the argument is NULL.

Syntax

```
JSONB json_valid(VARCHAR json_str)
```

example

1. parse valid JSON string

```
MySQL > SELECT json_valid('{"k1":"v31","k2":300}');
+--------------------------------------+
| json_valid('{"k1":"v31","k2":300}') |
+--------------------------------------+
|                                    1 |
+--------------------------------------+
1 row in set (0.02 sec)
```

2. parse invalid JSON string

```
MySQL > SELECT json_valid('invalid json');
+----------------------------+
| json_valid('invalid json') |
+----------------------------+
|                          0 |
+----------------------------+
1 row in set (0.02 sec)
```

3. parse NULL

```
MySQL > select json_valid(NULL);
+------------------+
| json_valid(NULL) |
+------------------+
|             NULL |
+------------------+
1 row in set (0.02 sec)
```

keywords

JSON, VALID, JSON_VALID

JSON_CONTAINS

json_contains

description

Syntax

`BOOLEAN json_contains(JSON json_str, JSON candidate) BOOLEAN json_contains(JSON json_str, JSON candidate, VARCHAR json_path) BOOLEAN json_contains(VARCHAR json_str, VARCHAR candidate, VARCHAR json_path)`

Indicates by returning 1 or 0 whether a given candidate JSON document is contained at a specific path within the json_str JSON document

example

```
mysql> SET @j = '{"a": 1, "b": 2, "c": {"d": 4}}';
mysql> SET @j2 = '1';
mysql> SELECT JSON_CONTAINS(@j, @j2, '$.a');
+-------------------------------+
| JSON_CONTAINS(@j, @j2, '$.a') |
+-------------------------------+
|                             1 |
+-------------------------------+
mysql> SELECT JSON_CONTAINS(@j, @j2, '$.b');
+-------------------------------+
| JSON_CONTAINS(@j, @j2, '$.b') |
+-------------------------------+
|                             0 |
+-------------------------------+

mysql> SET @j2 = '{"d": 4}';
mysql> SELECT JSON_CONTAINS(@j, @j2, '$.a');
+-------------------------------+
| JSON_CONTAINS(@j, @j2, '$.a') |
+-------------------------------+
|                             0 |
+-------------------------------+
mysql> SELECT JSON_CONTAINS(@j, @j2, '$.c');
+-------------------------------+
| JSON_CONTAINS(@j, @j2, '$.c') |
+-------------------------------+
|                             1 |
+-------------------------------+

mysql> SELECT json_contains('[1, 2, {"x": 3}]', '1');
+---------------------------------------+
| json_contains('[1, 2, {"x": 3}]', '1') |
+---------------------------------------+
|                                     1 |
+---------------------------------------+
1 row in set (0.04 sec)
```

keywords

json,json_contains

JSON_LENGTH

json_length

description

Syntax

```
INT json_length(JSON json_str) INT json_length(JSON json_str, VARCHAR json_path)
```

If specified path, the JSON_LENGTH() function returns the length of the data matching the path in the JSON document, otherwise it returns the length of the JSON document. The function calculates the length of the JSON document according to the following rules:

- The length of a scalar is 1. For example, the length of 1, “‘x”’, true, false, null is all 1.
- The length of an array is the number of array elements. For example, the length of [1, 2] is 2.
- The length of an object is the number of object members. For example, the length of {“x”: 1} is 1.

example

```
mysql> SELECT json_length('{"k1":"v31","k2":300}');
+--------------------------------------+
| json_length('{"k1":"v31","k2":300}') |
+--------------------------------------+
|                                    2 |
+--------------------------------------+
1 row in set (0.26 sec)

mysql> SELECT json_length('"abc"');
+----------------------+
| json_length('"abc"') |
+----------------------+
|                    1 |
+----------------------+
1 row in set (0.17 sec)

mysql> SELECT json_length('{"x": 1, "y": [1, 2]}', '$.y');
+---------------------------------------------+
| json_length('{"x": 1, "y": [1, 2]}', '$.y') |
+---------------------------------------------+
|                                           2 |
+---------------------------------------------+
1 row in set (0.07 sec)
```

keywords

json,json_length

GET_JSON_DOUBLE

get_json_double

description

Syntax

```
DOUBLE get_json_double(VARCHAR json_str, VARCHAR json_path)
```

Parse and get the floating-point content of the specified path in the JSON string. Where json_path must start with the $symbol and use. as the path splitter. If the path contains…, double quotation marks can be used to surround it. Use [] to denote array subscripts, starting at 0. The content of path cannot contain “,[and]. If the json_string format is incorrect, or the json_path format is incorrect, or matches cannot be found, NULL is returned.

In addition, it is recommended to use the jsonb type and jsonb_extract_XXX function performs the same function.

Exception handling is as follows: - if the field specified by json_path does not exist, return NULL - if datatype of the field specified by json_path is not the same with type of json_extract_t, return t if it can be cast to t else NULL

example

1. Get the value of key as “k1”

```
mysql> SELECT get_json_double('{"k1":1.3, "k2":"2"}', "$.k1");
+-----------------------------------------------+
| get_json_double('{"k1":1.3, "k2":"2"}', '$.k1') |
+-----------------------------------------------+
|                                           1.3 |
+-----------------------------------------------+
```

2. Get the second element of the array whose key is “my. key”

```
mysql> SELECT get_json_double('{"k1":"v1", "my.key":[1.1, 2.2, 3.3]}', '$."my.key"[1]');
+-----------------------------------------------------------------------+
| get_json_double('{"k1":"v1", "my.key":[1.1, 2.2, 3.3]}', '$."my.key"[1]') |
+-----------------------------------------------------------------------+
|                                                                   2.2 |
+-----------------------------------------------------------------------+
```

3. Get the first element in an array whose secondary path is k1. key - > K2

```
mysql> SELECT get_json_double('{"k1.key":{"k2":[1.1, 2.2]}}', '$."k1.key".k2[0]');
+-----------------------------------------------------------------+
| get_json_double('{"k1.key":{"k2":[1.1, 2.2]}}', '$."k1.key".k2[0]') |
+-----------------------------------------------------------------+
```

```
|                                                                 1.1 |
+---------------------------------------------------------------------+
```

keywords

GET_JSON_DOUBLE,GET,JSON,DOUBLE

GET_JSON_INT

get_json_int

Description

Syntax

```
INT get_json_int(VARCHAR json_str, VARCHAR json_path)
```

Parse and retrieve the integer content of the specified path in the JSON string. Where json_path must start with the $symbol and use. as the path splitter. If the path contains…, double quotation marks can be used to surround it. Use [] to denote array subscripts, starting at 0. The content of path cannot contain “,[and]. If the json_string format is incorrect, or the json_path format is incorrect, or matches cannot be found, NULL is returned.

In addition, it is recommended to use the jsonb type and jsonb_extract_XXX function performs the same function.

Exception handling is as follows: - if the field specified by json_path does not exist, return NULL - if datatype of the field specified by json_path is not the same with type of json_extract_t, return t if it can be cast to t else NULL

example

1. Get the value of key as “k1”

```
mysql> SELECT get_json_int('{"k1":1, "k2":"2"}', "$.k1");
+-------------------------------------------+
| get_json_int('{"k1":1, "k2":"2"}', '$.k1') |
+-------------------------------------------+
|                                         1 |
+-------------------------------------------+
```

2. Get the second element of the array whose key is “my. key”

```
mysql> SELECT get_json_int('{"k1":"v1", "my.key":[1, 2, 3]}', '$."my.key"[1]');
+-----------------------------------------------------------------+
| get_json_int('{"k1":"v1", "my.key":[1, 2, 3]}', '$."my.key"[1]') |
+-----------------------------------------------------------------+
|                                                               2 |
+-----------------------------------------------------------------+
```

3. Get the first element in an array whose secondary path is k1. key - > K2

```
mysql> SELECT get_json_int('{"k1.key":{"k2":[1, 2]}}', '$."k1.key".k2[0]');
+-----------------------------------------------------------+
| get_json_int('{"k1.key":{"k2":[1, 2]}}', '$."k1.key".k2[0]') |
+-----------------------------------------------------------+
```

```
|                                                          1 |
+------------------------------------------------------------+
```

keywords

GET_JSON_INT,GET,JSON,INT

GET_JSON_BIGINT

get_json_bigint

Description

Syntax

```
INT get_json_bigint(VARCHAR json_str, VARCHAR json_path)
```

Parse and retrieve the big integer content of the specified path in the JSON string. Where json_path must start with the $symbol and use. as the path splitter. If the path contains…, double quotation marks can be used to surround it. Use [] to denote array subscripts, starting at 0. The content of path cannot contain “,[and]. If the json_string format is incorrect, or the json_path format is incorrect, or matches cannot be found, NULL is returned.

In addition, it is recommended to use the jsonb type and jsonb_extract_XXX function performs the same function.

Exception handling is as follows: - if the field specified by json_path does not exist, return NULL - if datatype of the field specified by json_path is not the same with type of json_extract_t, return t if it can be cast to t else NULL

example

1. Get the value of key as “k1”

```
mysql> SELECT get_json_bigint('{"k1":1, "k2":"2"}', "$.k1");
+-----------------------------------------------+
| get_json_bigint('{"k1":1, "k2":"2"}', '$.k1') |
+-----------------------------------------------+
|                                             1 |
+-----------------------------------------------+
```

2. Get the second element of the array whose key is “my. key”

```
mysql> SELECT get_json_bigint('{"k1":"v1", "my.key":[1, 1678708107000, 3]}', '$."my.key"[1]');
+-------------------------------------------------------------------------------+
| get_json_bigint('{"k1":"v1", "my.key":[1, 1678708107000, 3]}', '$."my.key"[1]') |
+-------------------------------------------------------------------------------+
|                                                                 1678708107000 |
+-------------------------------------------------------------------------------+
```

3. Get the first element in an array whose secondary path is k1. key - > K2

```
mysql> SELECT get_json_bigint('{"k1.key":{"k2":[1678708107000, 2]}}', '$."k1.key".k2[0]');
+-----------------------------------------------------------------------------+
| get_json_bigint('{"k1.key":{"k2":[1678708107000, 2]}}', '$."k1.key".k2[0]') |
+-----------------------------------------------------------------------------+
```

```
|                                                                          1678708107000 |
+--------------------------------------------------------------------------------------+
```

keywords

GET_JSON_BIGINT,GET,JSON,BIGINT

GET_JSON_STRING

get_json_string

description

Syntax

```
VARCHAR get_json_string (VARCHAR json str, VARCHAR json path)
```

Parse and retrieve the string content of the specified path in the JSON string. Where json_path must start with the $symbol and use. as the path splitter. If the path contains…, double quotation marks can be used to surround it. Use [] to denote array subscripts, starting at 0. The content of path cannot contain “,[and]. If the json_string format is incorrect, or the json_path format is incorrect, or matches cannot be found, NULL is returned.

In addition, it is recommended to use the jsonb type and jsonb_extract_XXX function performs the same function.

Exception handling is as follows: - if the field specified by json_path does not exist, return NULL - if datatype of the field specified by json_path is not the same with type of json_extract_t, return t if it can be cast to t else NULL

example

1. Get the value of key as “k1”

```
mysql> SELECT get_json_string('{"k1":"v1", "k2":"v2"}', "$.k1");
+-----------------------------------------------------+
| get_json_string('{"k1":"v1", "k2":"v2"}', '$.k1') |
+-----------------------------------------------------+
| v1                                                  |
+-----------------------------------------------------+
```

2. Get the second element of the array whose key is “my. key”

```
mysql> SELECT get_json_string('{"k1":"v1", "my.key":["e1", "e2", "e3"]}', '$."my.key"[1]');
+-----------------------------------------------------------------------------------+
| get_json_string('{"k1":"v1", "my.key":["e1", "e2", "e3"]}', '$."my.key"[1]') |
+-----------------------------------------------------------------------------------+
| e2                                                                                |
+-----------------------------------------------------------------------------------+
```

3. Get the first element in an array whose secondary path is k1. key - > K2

```
mysql> SELECT get_json_string('{"k1.key":{"k2":["v1", "v2"]}}', '$."k1.key".k2[0]');
+-------------------------------------------------------------------+
| get_json_string('{"k1.key":{"k2":["v1", "v2"]}}', '$."k1.key".k2[0]') |
+-------------------------------------------------------------------+
```

```
| v1                                                                  |
+---------------------------------------------------------------------+
```

4. Get all the values in the array where the key is “k1”

```
mysql> SELECT get_json_string('[{"k1":"v1"}, {"k2":"v2"}, {"k1":"v3"}, {"k1":"v4"}]', '$.k1');
+-----------------------------------------------------------------------------------+
| get_json_string('[{"k1":"v1"}, {"k2":"v2"}, {"k1":"v3"}, {"k1":"v4"}]', '$.k1') |
+-----------------------------------------------------------------------------------+
| ["v1","v3","v4"]                                                                  |
+-----------------------------------------------------------------------------------+
```

keywords

GET_JSON_STRING,GET,JSON,STRING

JSON_INSERT

json_insert

Description

Syntax

```
VARCHAR json_insert(VARCHAR json_str, VARCHAR path, VARCHAR val[, VARCHAR path, VARCHAR val] ...)
```

`json_set` function inserts data in a JSON and returns the result.Returns NULL if `json_str` or `path` is NULL. Otherwise, an error occurs if the `json_str` argument is not a valid JSON or any path argument is not a valid path expression or contains a * wildcard.

The path-value pairs are evaluated left to right.

A path-value pair for a nonexisting path in the json adds the value to the json if the path identifies one of these types of values:

- A member not present in an existing object. The member is added to the object and associated with the new value.
- A position past the end of an existing array. The array is extended with the new value. If the existing value is not an array, it is autowrapped as an array, then extended with the new value.

Otherwise, a path-value pair for a nonexisting path in the json is ignored and has no effect.

example

```
MySQL> select json_insert(null, null, null);
+---------------------------------+
| json_insert(NULL, NULL, 'NULL') |
+---------------------------------+
| NULL                            |
+---------------------------------+

MySQL> select json_insert('{"k": 1}', "$.k", 2);
+---------------------------------------+
| json_insert('{\"k\": 1}', '$.k', '2') |
+---------------------------------------+
| {"k":1}                               |
+---------------------------------------+

MySQL> select json_insert('{"k": 1}', "$.j", 2);
+---------------------------------------+
| json_insert('{\"k\": 1}', '$.j', '2') |
+---------------------------------------+
| {"k":1,"j":2}                         |
+---------------------------------------+
```

keywords

JSON, json_insert

JSON_REPLACE

json_replace

Description

Syntax

```
VARCHAR json_replace(VARCHAR json_str, VARCHAR path, VARCHAR val[, VARCHAR path, VARCHAR val] ...)
```

`json_replace` function updates data in a JSON and returns the result.Returns NULL if `json_str` or `path` is NULL. Otherwise, an error occurs if the `json_str` argument is not a valid JSON or any path argument is not a valid path expression or contains a * wildcard.

The path-value pairs are evaluated left to right.

A path-value pair for an existing path in the json overwrites the existing json value with the new value.

Otherwise, a path-value pair for a nonexisting path in the json is ignored and has no effect.

example

```
MySQL> select json_replace(null, null, null);
+----------------------------------+
| json_replace(NULL, NULL, 'NULL') |
+----------------------------------+
| NULL                             |
+----------------------------------+

MySQL> select json_replace('{"k": 1}', "$.k", 2);
+---------------------------------------+
| json_replace('{\"k\": 1}', '$.k', '2') |
+---------------------------------------+
| {"k":2}                               |
+---------------------------------------+

MySQL> select json_replace('{"k": 1}', "$.j", 2);
+---------------------------------------+
| json_replace('{\"k\": 1}', '$.j', '2') |
+---------------------------------------+
| {"k":1}                               |
+---------------------------------------+
```

keywords

JSON, json_replace

JSON_SET

json_set

Description

Syntax

```
VARCHAR json_set(VARCHAR json_str, VARCHAR path, VARCHAR val[, VARCHAR path, VARCHAR val] ...)
```

`json_set` function inserts or updates data in a JSON and returns the result.Returns NULL if `json_str` or `path` is NULL. Otherwise, an error occurs if the `json_str` argument is not a valid JSON or any path argument is not a valid path expression or contains a * wildcard.

The path-value pairs are evaluated left to right.

A path-value pair for an existing path in the json overwrites the existing json value with the new value. A path-value pair for a nonexisting path in the json adds the value to the json if the path identifies one of these types of values:

- A member not present in an existing object. The member is added to the object and associated with the new value.
- A position past the end of an existing array. The array is extended with the new value. If the existing value is not an array, it is autowrapped as an array, then extended with the new value.

Otherwise, a path-value pair for a nonexisting path in the json is ignored and has no effect.

example

```
MySQL> select json_set(null, null, null);
+------------------------------+
| json_set(NULL, NULL, 'NULL') |
+------------------------------+
| NULL                         |
+------------------------------+

MySQL> select json_set('{"k": 1}', "$.k", 2);
+-----------------------------------+
| json_set('{\"k\": 1}', '$.k', '2') |
+-----------------------------------+
| {"k":2}                           |
+-----------------------------------+

MySQL> select json_set('{"k": 1}', "$.j", 2);
+-----------------------------------+
| json_set('{\"k\": 1}', '$.j', '2') |
+-----------------------------------+
| {"k":1,"j":2}                     |
+-----------------------------------+
```

keywords

JSON, json_set

MAP

Description

Syntax

```
MAP<K, V> map(K key0, V value0, K key1, V value1, ..., K keyn, V valuen)
```

Construct a type-specific `Map<K, V>` using a set of key-value pairs.

Example

```
mysql> select map(1, "100", 0.1, 2);
+-----------------------------------------------------------------------------------+
| map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)) |
+-----------------------------------------------------------------------------------+
| {1.0:"100", 0.1:"2"}                                                              |
+-----------------------------------------------------------------------------------+
1 row in set (0.16 sec)

mysql> select map(1, "100", 0.1, 2)[1];
+-----------------------------------------------------------------------------------------------------------------------------------+
| element_at(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)), cast(1 as DECIMALV3(2, 1))) |
+-----------------------------------------------------------------------------------------------------------------------------------+
| 100                                                                                                                               |
+-----------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.16 sec)
```

Keywords

MAP

MAP_KEYS

Description

Syntax

```
ARRAY<K> map_keys(Map<K, V> map)
```

Extracts the keys of a given `map` into an `ARRAY` of the corresponding type.

Example

```
mysql> select map_keys(map(1, "100", 0.1, 2));
+-----------------------------------------------------------------------------------------------+
| map_keys(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT))) |
+-----------------------------------------------------------------------------------------------+
| [1.0, 0.1]                                                                                    |
+-----------------------------------------------------------------------------------------------+
1 row in set (0.15 sec)

mysql> select map_keys(map());
+-----------------+
| map_keys(map()) |
+-----------------+
| []              |
+-----------------+
1 row in set (0.12 sec)
```

Keywords

MAP, KEYS, MAP_KEYS

MAP_VALUES

Description

Syntax

```
ARRAY<K> map_values(Map<K, V> map)
```

Extracts the value of a given `map` into an `ARRAY` of the corresponding type.

Example

```
mysql> select map_values(map(1, "100", 0.1, 2));
+-----------------------------------------------------------------------------------------------------+
| map_values(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT))) |
+-----------------------------------------------------------------------------------------------------+
| ["100", "2"]                                                                                        |
+-----------------------------------------------------------------------------------------------------+
1 row in set (0.12 sec)

mysql> select map_values(map());
+-------------------+
| map_values(map()) |
+-------------------+
| []                |
+-------------------+
1 row in set (0.11 sec)
```

Keywords

MAP, VALUES, MAP_VALUES

MAP_CONTAINS_KEY

Description

Syntax

```
BOOLEAN map_contains_key(Map<K, V> map, K key)
```

Determines whether the given `map` contains a particular key `key`.

Example

```
mysql> select map_contains_key(map(1, "100", 0.1, 2), "1");
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
| map_contains_key(cast(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)) as MAP<DECIMALV3(38,
9),VARCHAR(3)>), cast('1' as DECIMALV3(38, 9))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
|
1 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
1 row in set (0.17 sec)

mysql> select map_contains_key(map(1, "100", 0.1, 2), "abc");
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
| map_contains_key(cast(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)) as MAP<DECIMALV3(38,
9),VARCHAR(3)>), cast('abc' as DECIMALV3(38, 9))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
|
0 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
1 row in set (0.16 sec)

mysql> select map_contains_key(map(1, "100", 0.1, 2), 0.11);
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
| map_contains_key(cast(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)) as MAP<DECIMALV3(3,
2),VARCHAR(3)>), cast(0.11 as DECIMALV3(3, 2))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
|
0 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
1 row in set (0.15 sec)

mysql> select map_contains_key(map(null, 1, 2, null), null);
+---------------------------------------------+
| map_contains_key(map(NULL, 1, 2, NULL), NULL) |
+---------------------------------------------+
|                                           1 |
```

```
+-----------------------------------------------+
1 row in set (0.14 sec)
```

Keywords

MAP, CONTAINS, KEY, MAP_CONTAINS_KEY

MAP_CONTAINS_VALUE

Description

Syntax

```
BOOLEAN map_contains_value(Map<K, V> map, V value)
```

Determines whether the given `map` contains a particular value `value`.

Example

```
mysql> select map_contains_key(map(1, "100", 0.1, 2), "1");
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
| map_contains_key(cast(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)) as MAP<DECIMALV3(38,
9),VARCHAR(3)>), cast('1' as DECIMALV3(38, 9))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
|
1 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
1 row in set (0.17 sec)

mysql> select map_contains_key(map(1, "100", 0.1, 2), "abc");
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
| map_contains_key(cast(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)) as MAP<DECIMALV3(38,
9),VARCHAR(3)>), cast('abc' as DECIMALV3(38, 9))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
|
0 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
1 row in set (0.16 sec)

mysql> select map_contains_key(map(1, "100", 0.1, 2), 0.11);
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
| map_contains_key(cast(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)) as MAP<DECIMALV3(3,
2),VARCHAR(3)>), cast(0.11 as DECIMALV3(3, 2))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
|
0 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------+
1 row in set (0.15 sec)

mysql> select map_contains_value(map(1, "100", 0.1, 2), 100.0);
+-----------------------------------------------------------------------------------------------------------------------------------------------
---+
| map_contains_value(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)), cast(100.0 as
VARCHAR(3))) |
+-----------------------------------------------------------------------------------------------------------------------------------------------
```

```
---+
|
1 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
---+
1 row in set (0.16 sec)

mysql> select map_contains_value(map(1, "100", 0.1, 2), 101);
+-----------------------------------------------------------------------------------------------------------------------------------------------
-+
| map_contains_value(map(cast(1 as DECIMALV3(2, 1)), '100', cast(0.1 as DECIMALV3(2, 1)), cast(2 as TEXT)), cast(101 as VARCHAR(3)))
|
+-----------------------------------------------------------------------------------------------------------------------------------------------
-+
|                                                                                                                                             0
|
+-----------------------------------------------------------------------------------------------------------------------------------------------
-+
1 row in set (0.14 sec)

mysql> select map_contains_value(map(null, 1, 2, null), null);
+-------------------------------------------------+
| map_contains_value(map(NULL, 1, 2, NULL), NULL) |
+-------------------------------------------------+
|                                               1 |
+-------------------------------------------------+
1 row in set (0.15 sec)
```

Keywords

MAP, CONTAINS, VALUE, MAP_CONTAINS_VALUE

MURMUR_HASH3_32

murmur_hash3_32

description

Syntax

```
INT MURMUR_HASH3_32(VARCHAR input, ...)
```

Return the 32 bits murmur3 hash of input string.

example

```
mysql> select murmur_hash3_32(null);
+-----------------------+
| murmur_hash3_32(NULL) |
+-----------------------+
|                  NULL |
+-----------------------+

mysql> select murmur_hash3_32("hello");
+--------------------------+
| murmur_hash3_32('hello') |
+--------------------------+
|               1321743225 |
+--------------------------+

mysql> select murmur_hash3_32("hello", "world");
+-----------------------------------+
| murmur_hash3_32('hello', 'world') |
+-----------------------------------+
|                         984713481 |
+-----------------------------------+
```

keywords

```
MURMUR_HASH3_32,HASH
```

MURMUR_HASH3_64

murmur_hash3_64

description

Syntax

```
BIGINT MURMUR_HASH3_64(VARCHAR input, ...)
```

Return the 64 bits murmur3 hash of input string.

example

```
mysql> select murmur_hash3_64(null);
+----------------------+
| murmur_hash3_64(NULL) |
+----------------------+
|                 NULL |
+----------------------+

mysql> select murmur_hash3_64("hello");
+--------------------------+
| murmur_hash3_64('hello') |
+--------------------------+
|     -3215607508166160593 |
+--------------------------+

mysql> select murmur_hash3_64("hello", "world");
+-----------------------------------+
| murmur_hash3_64('hello', 'world') |
+-----------------------------------+
|               3583109472027628045 |
+-----------------------------------+
```

keywords

```
MURMUR_HASH3_64,HASH
```

HLL_CARDINALITY

Description

HLL_CARDINALITY calculates the cardinality of a HyperLogLog (HLL) type value. It is an approximate counting algorithm suitable for estimating the number of distinct elements in large datasets.

Syntax

```
HLL_CARDINALITY(<hll>)
```

Parameters

Parameter	Description
<hll>	The HLL type value representing the dataset whose cardinality needs to be estimated.

Return Value

Returns the estimated cardinality of the HLL type value, representing the number of distinct elements in the dataset.

Example

```
select HLL_CARDINALITY(uv_set) from test_uv;
```

```
+--------------------------+
| hll_cardinality(`uv_set`) |
+--------------------------+
|                        3 |
+--------------------------+
```

HLL_EMPTY

Description

`HLL_EMPTY` returns an empty HLL (HyperLogLog) value, representing a data set with no elements.

Syntax

```
HLL_EMPTY()
```

Return Value

Returns an empty HLL type value, representing a data set with no elements.

Example

```
select hll_cardinality(hll_empty());
```

```
+-----------------------------+
| hll_cardinality(hll_empty()) |
+-----------------------------+
|                           0 |
+-----------------------------+
```

HLL_HASH

Description

Converts a given value to the HLL (HyperLogLog) type. This function is typically used during data loading to create an HLL from raw data.

Syntax

```
HLL_HASH(<value>)
```

Parameters

Parameter	Description
<value>	The value to be converted to HLL type. This can be a string, number, or any data type.

Examples

```
SELECT HLL_CARDINALITY(HLL_HASH('abc'));
```

```
+----------------------------------+
| hll_cardinality(HLL_HASH('abc')) |
+----------------------------------+
|                                1 |
+----------------------------------+
```

CONV

Description

Do radix conversion for input parameter.

Syntax

```
CONV(<input>, <from_base>, <to_base>)
```

Parameters

Parameter	Description
<input>	Parameters to be converted, either as strings or integers
<from_base>	Numeric, the source base, within [2,36].
<to_base>	Numeric, the target base, within [2,36].

Return Value

The number under the converted target binary <to_base> is returned as a string.

Examples

```
SELECT CONV(15,10,2);
```

```
+-----------------+
| conv(15, 10, 2) |
+-----------------+
| 1111            |
+-----------------+
```

```
SELECT CONV('ff',16,10);
```

```
+--------------------+
| conv('ff', 16, 10) |
+--------------------+
| 255                |
+--------------------+
```

```
SELECT CONV(230,10,16);
```

```
+-------------------+
| conv(230, 10, 16) |
```

```
+-------------------+
| E6                |
+-------------------+
```

BIN

Description

Converts decimal numbers to binary text.

Syntax

```
BIN(<a>)
```

Parameters

Parameter	Description
<a>	Decimal value to be converted

Return Value

The binary representation of the parameter <a>. When <a> is negative, the result is its 64-bit complement.

Examples

```
select bin(0);
```

```
+--------+
| bin(0) |
+--------+
| 0      |
+--------+
```

```
select bin(-1);
```

```
+------------------------------------------------------------------+
| bin(-1)                                                          |
+------------------------------------------------------------------+
| 1111111111111111111111111111111111111111111111111111111111111111 |
+------------------------------------------------------------------+
```

```
select bin(123);
```

```
+----------+
| bin(123) |
+----------+
| 1111011  |
+----------+
```

SIN

Description

Calculate the sine of the parameter

Syntax

```
SIN(<a>)
```

Parameters

Parameter	Description
<a>	floating point number, the radian value of the parameter to calculate

Return Value

The sine of the parameter <a>, expressed in radians.

Examples

```
select sin(1);
```

```
+------------------------+
| sin(cast(1 as DOUBLE)) |
+------------------------+
|     0.8414709848078965 |
+------------------------+
```

```
select sin(0);
```

```
+------------------------+
| sin(cast(0 as DOUBLE)) |
+------------------------+
|                    0.0 |
+------------------------+
```

```
select sin(Pi());
```

```
+-----------------------------------+
| sin(pi())                         |
+-----------------------------------+
```

```
| 0.00000000000000012246467991473532 |
+------------------------------------+
```

COS

Description

Calculate the cosine of the parameter

Syntax

```
COS(<a>)
```

Parameters

Parameter	Description
<a>	floating point number, the radian value of the parameter to calculate

Return Value

The cosine of the parameter <a>, expressed in radians.

Examples

```
select cos(1);
```

```
+--------------------+
| cos(1.0)           |
+--------------------+
| 0.54030230586813977 |
+--------------------+
```

```
select cos(0);
```

```
+------------------------+
| cos(cast(0 as DOUBLE)) |
+------------------------+
|                    1.0 |
+------------------------+
```

```
select cos(Pi());
```

```
+-----------+
| cos(pi()) |
+-----------+
```

```
|        -1 |
+-----------+
```

TAN

tan

description

Syntax

`DOUBLE tan(DOUBLE x)` Returns the tangent of `x`, where `x` is in radians.

example

```
mysql> select tan(0);
+----------+
| tan(0.0) |
+----------+
|        0 |
+----------+
mysql> select tan(1);
+--------------------+
| tan(1.0)           |
+--------------------+
| 1.5574077246549023 |
+--------------------+
```

keywords

```
TAN
```

ASIN

asin

description

Syntax

`DOUBLE asin(DOUBLE x)` Returns the arc sine of `x`, or `nan` if `x` is not in the range `-1` to `1`.

example

```
mysql> select asin(0.5);
+--------------------+
| asin(0.5)          |
+--------------------+
| 0.52359877559829893 |
+--------------------+
mysql> select asin(2);
+-----------+
| asin(2.0) |
+-----------+
|       nan |
+-----------+
```

keywords

```
ASIN
```

ACOS

acos

description

Syntax

`DOUBLE acos(DOUBLE x)` Returns the arc cosine of `x`, or `nan` if `x` is not in the range `-1` to `1`.

example

```
mysql> select acos(1);
+-----------+
| acos(1.0) |
+-----------+
|         0 |
+-----------+
mysql> select acos(0);
+--------------------+
| acos(0.0)          |
+--------------------+
| 1.5707963267948966 |
+--------------------+
mysql> select acos(-2);
+------------+
| acos(-2.0) |
+------------+
|        nan |
+------------+
```

keywords

```
ACOS
```

ATAN

atan

description

Syntax

`DOUBLE atan(DOUBLE x)` Returns the arctangent of `x`, where `x` is in radians.

example

```
mysql> select atan(0);
+-----------+
| atan(0.0) |
+-----------+
|         0 |
+-----------+
mysql> select atan(2);
+--------------------+
| atan(2.0)          |
+--------------------+
| 1.1071487177940904 |
+--------------------+
```

keywords

```
ATAN
```

E

e

description

Syntax

`DOUBLE e()` Returns the constant `e` value.

example

```
mysql> select e();
+-------------------+
| e()               |
+-------------------+
| 2.7182818284590451 |
+-------------------+
```

keywords

```
E
```

PI

Pi

description

Syntax

`DOUBLE Pi()` Returns the constant `Pi` value.

example

```
mysql> select Pi();
+-------------------+
| pi()              |
+-------------------+
| 3.1415926535897931 |
+-------------------+
```

keywords

```
PI
```

EXP

exp

description

Syntax

`DOUBLE exp(DOUBLE x)` Returns `x` raised to the base `e`.

> **Another alias for this function is `dexp`.**

example

```
mysql> select exp(2);
+------------------+
| exp(2.0)         |
+------------------+
| 7.38905609893065 |
+------------------+
mysql> select exp(3.4);
+--------------------+
| exp(3.4)           |
+--------------------+
| 29.964100047397011 |
+--------------------+
```

keywords

```
EXP, DEXP
```

LOG

log

description

Syntax

`DOUBLE log(DOUBLE b, DOUBLE x)` Returns the logarithm of `x` based on base `b`.

example

```
mysql> select log(5,1);
+---------------+
| log(5.0, 1.0) |
+---------------+
|             0 |
+---------------+
mysql> select log(3,20);
+--------------------+
| log(3.0, 20.0)     |
+--------------------+
| 2.7268330278608417 |
+--------------------+
mysql> select log(2,65536);
+-------------------+
| log(2.0, 65536.0) |
+-------------------+
|                16 |
+-------------------+
```

keywords

```
LOG
```

LOG2

log2

description

Syntax

`DOUBLE log2(DOUBLE x)` Returns the natural logarithm of `x` to base `2`.

example

```
mysql> select log2(1);
+-----------+
| log2(1.0) |
+-----------+
|         0 |
+-----------+
mysql> select log2(2);
+-----------+
| log2(2.0) |
+-----------+
|         1 |
+-----------+
mysql> select log2(10);
+--------------------+
| log2(10.0)         |
+--------------------+
| 3.3219280948873622 |
+--------------------+
```

keywords

```
LOG2
```

LOG10

log10

description

Syntax

`DOUBLE log10(DOUBLE x)` Returns the natural logarithm of `x` to base `10`.

> **Another alias for this function is `dlog10`.**

example

```
mysql> select log10(1);
+------------+
| log10(1.0) |
+------------+
|          0 |
+------------+
mysql> select log10(10);
+-------------+
| log10(10.0) |
+-------------+
|           1 |
+-------------+
mysql> select log10(16);
+--------------------+
| log10(16.0)        |
+--------------------+
| 1.2041199826559248 |
+--------------------+
```

keywords

```
LOG10, DLOG10
```

LN

ln

description

Syntax

`DOUBLE ln(DOUBLE x)` Returns the natural logarithm of `x` to base `e`.

> **Another alias for this function is `dlog1`.**

example

```
mysql> select ln(1);
+---------+
| ln(1.0) |
+---------+
|       0 |
+---------+
mysql> select ln(e());
+---------+
| ln(e()) |
+---------+
|       1 |
+---------+
mysql> select ln(10);
+--------------------+
| ln(10.0)           |
+--------------------+
| 2.3025850929940459 |
+--------------------+
```

keywords

```
LN, DLOG1
```

CEIL

Description

Round up floating-point and fixed-point decimals to a specific number of places and return the rounded floating-point or fixed-point number.

Syntax

```
CEIL(<a>[, <d>])
```

Parameters

Parameter	Description
<a>	Floating-point (Double) or fixed-point (Decimal) parameter indicating the parameter to be rounded
<d>	Optional, integer, indicates rounding to the target number of digits, a positive number means rounding to the next decimal point, a negative number means rounding to the next decimal point, and 0 indicates rounding to an integer. When not filled, it is equivalent to <d> = 0.

Return Value

Returns the smallest rounded number greater than or equal to <a> according to the following rules.

Round to 1/(10^d) digit, i.e., make the result divisible by 1/(10^d). If 1/(10^d) is not exact, the rounding digit is the nearest number of the corresponding data type.

For an entry <a> of type Decimal, assuming it is of type Decimal(p, s), the return value is:

- Decimal(p, 0), if <d> <= 0
- Decimal(p, <d>), if 0 < <d> <= s
- Decimal(p, s), if <d> > s

Alias

- DCEIL
- CEILING

Examples

```
select ceil(123.456);
```

```
+---------------+
| ceil(123.456) |
+---------------+
|           124 |
+---------------+
```

```
select ceil(123.456, 2);
```

```
+------------------+
| ceil(123.456, 2) |
+------------------+
|           123.46 |
+------------------+
```

```
select ceil(123.456, -2);
```

```
+-------------------+
| ceil(123.456, -2) |
+-------------------+
|               200 |
+-------------------+
```

```
select ceil(123.45, 1), ceil(123.45), ceil(123.45, 0), ceil(123.45, -1);
```

```
+-----------------+--------------+-----------------+------------------+
| ceil(123.45, 1) | ceil(123.45) | ceil(123.45, 0) | ceil(123.45, -1) |
+-----------------+--------------+-----------------+------------------+
|           123.5 |          124 |             124 |              130 |
+-----------------+--------------+-----------------+------------------+
```

```
select ceil(x, 2) from ( select cast(123.456 as decimal(6,3)) as x from numbers("number"="5") )t;
```

```
+------------+
| ceil(x, 2) |
+------------+
|     123.46 |
|     123.46 |
|     123.46 |
|     123.46 |
|     123.46 |
+------------+
```

FLOOR

Description

Round down floating-point and fixed-point decimals to a specific number of digits and return the rounded floating-point or fixed-point number.

Syntax

```
FLOOR(<a>[, <d>])
```

Parameters

Parameter	Description
<a>	Floating-point (Double) or fixed-point (Decimal) parameter indicating the parameter to be rounded
<d>	Optional, integer, indicates rounding to the target number of digits, a positive number means rounding to the next decimal point, a negative number means rounding to the next decimal point, and 0 indicates rounding to an integer. When not filled, it is equivalent to <d> = 0.

Return Value

Returns the largest rounded number less than or equal to <a> according to the following rules.

Round to 1/(10^d) digit, i.e., make the result divisible by 1/(10^d). If 1/(10^d) is not exact, the rounding digit is the nearest number of the corresponding data type.

For an entry <a> of type Decimal, assuming it is of type Decimal(p, s), the return value is:

- Decimal(p, 0), if <d> <= 0
- Decimal(p, <d>), if 0 < <d> <= s
- Decimal(p, s), if <d> > s

Alias

- DFLOOR

Examples

```
select floor(123.456);
```

```
+----------------+
| floor(123.456) |
+----------------+
|            123 |
+----------------+
```

```
select floor(123.456, 2);
```

```
+-------------------+
| floor(123.456, 2) |
+-------------------+
|            123.45 |
+-------------------+
```

```
select floor(123.456, -2);
```

```
+--------------------+
| floor(123.456, -2) |
+--------------------+
|                100 |
+--------------------+
```

```
select floor(123.45, 1), floor(123.45), floor(123.45, 0), floor(123.45, -1);
```

```
+------------------+---------------+------------------+-------------------+
| floor(123.45, 1) | floor(123.45) | floor(123.45, 0) | floor(123.45, -1) |
+------------------+---------------+------------------+-------------------+
|            123.4 |           123 |              123 |               120 |
+------------------+---------------+------------------+-------------------+
```

```
select floor(x, 2) from ( select cast(123.456 as decimal(6,3)) as x from numbers("number"="5") )t;
```

```
+-------------+
| floor(x, 2) |
+-------------+
|      123.45 |
|      123.45 |
|      123.45 |
|      123.45 |
|      123.45 |
+-------------+
```

PMOD

pmod

description

Syntax

```
BIGINT PMOD(BIGINT x, BIGINT y)
DOUBLE PMOD(DOUBLE x, DOUBLE y)
```

Returns the positive result of x mod y in the residue systems. Formally, return (x%y+y)%y.

example

```
MySQL [test]> SELECT PMOD(13,5);
+-------------+
| pmod(13, 5) |
+-------------+
|           3 |
+-------------+

MySQL [test]> SELECT PMOD(-13,5);
+-------------+
| pmod(-13, 5) |
+-------------+
|           2 |
+-------------+
```

keywords

```
PMOD
```

ROUND

round

description

Syntax

`T round(T x[, d])` Rounds the argument `x` to `d` decimal places. `d` defaults to 0 if not specified. If d is negative, the left d digits of the decimal point are 0. If x or d is null, null is returned. 2.5 will round up to 3. If you want to round down to 2, please use the round_bankers function.

> **Another alias for this function is `dround`.**

example

```
mysql> select round(2.4);
+------------+
| round(2.4) |
+------------+
|          2 |
+------------+
mysql> select round(2.5);
+------------+
| round(2.5) |
+------------+
|          3 |
+------------+
mysql> select round(-3.4);
+-------------+
| round(-3.4) |
+-------------+
|          -3 |
+-------------+
mysql> select round(-3.5);
+-------------+
| round(-3.5) |
+-------------+
|          -4 |
+-------------+
mysql> select round(1667.2725, 2);
+---------------------+
| round(1667.2725, 2) |
+---------------------+
|             1667.27 |
+---------------------+
mysql> select round(1667.2725, -2);
+----------------------+
| round(1667.2725, -2) |
+----------------------+
```

```
|                1700 |
+---------------------+
```

keywords

```
ROUND, DROUND
```

ROUND_BANKERS

round_bankers

description

Syntax

`T round_bankers(T x[, d])` Rounds the argument `x` to `d` specified decimal places. `d` defaults to 0 if not specified. If d is negative, the left d digits of the decimal point are 0. If x or d is null, null is returned.

- If the rounding number is halfway between two numbers, the function uses banker's rounding.
- In other cases, the function rounds numbers to the nearest integer.

example

```
mysql> select round_bankers(0.4);
+-------------------+
| round_bankers(0.4) |
+-------------------+
|                  0 |
+-------------------+
mysql> select round_bankers(-3.5);
+--------------------+
| round_bankers(-3.5) |
+--------------------+
|                 -4 |
+--------------------+
mysql> select round_bankers(-3.4);
+--------------------+
| round_bankers(-3.4) |
+--------------------+
|                 -3 |
+--------------------+
mysql> select round_bankers(10.755, 2);
+-------------------------+
| round_bankers(10.755, 2) |
+-------------------------+
|                   10.76 |
+-------------------------+
mysql> select round_bankers(1667.2725, 2);
+----------------------------+
| round_bankers(1667.2725, 2) |
+----------------------------+
|                    1667.27 |
+----------------------------+
mysql> select round_bankers(1667.2725, -2);
+-----------------------------+
| round_bankers(1667.2725, -2) |
+-----------------------------+
|                        1700 |
+-----------------------------+
```

keywords

```
round_bankers
```

TRUNCATE

truncate

description

Syntax

`DOUBLE truncate(DOUBLE x, INT d)` Numerically truncate `x` according to the number of decimal places `d`.

The rules are as follows: When `d > 0`: keep `d` decimal places of `x` When `d = 0`: remove the fractional part of `x` and keep only the integer part When `d < 0`: Remove the fractional part of `x`, and replace the integer part with the number `0` according to the number of digits specified by `d`

example

```
mysql> select truncate(124.3867, 2);
+-----------------------+
| truncate(124.3867, 2) |
+-----------------------+
|                124.38 |
+-----------------------+
mysql> select truncate(124.3867, 0);
+-----------------------+
| truncate(124.3867, 0) |
+-----------------------+
|                   124 |
+-----------------------+
mysql> select truncate(-124.3867, -2);
+-------------------------+
| truncate(-124.3867, -2) |
+-------------------------+
|                    -100 |
+-------------------------+
```

keywords

```
TRUNCATE
```

ABS

abs

description

Syntax

```
SMALLINT abs(TINYINT x)
INT abs(SMALLINT x)
BIGINT abs(INT x)
LARGEINT abs(BIGINT x)
LARGEINT abs(LARGEINT x)
DOUBLE abs(DOUBLE x)
FLOAT abs(FLOAT x)
DECIMAL abs(DECIMAL x)`
```

Returns the absolute value of x.

example

```
mysql> select abs(-2);
+---------+
| abs(-2) |
+---------+
|       2 |
+---------+
mysql> select abs(3.254655654);
+------------------+
| abs(3.254655654) |
+------------------+
|      3.254655654 |
+------------------+
mysql> select abs(-3254654236547654354654767);
+---------------------------------+
| abs(-3254654236547654354654767) |
+---------------------------------+
| 3254654236547654354654767       |
+---------------------------------+
```

keywords

```
ABS
```

SQRT

sqrt

description

Syntax

`DOUBLE sqrt(DOUBLE x)` Returns the square root of `x`.`x` is required to be greater than or equal to `0`.

> **Another alias for this function is `dsqrt`.**

example

```
mysql> select sqrt(9);
+-----------+
| sqrt(9.0) |
+-----------+
|         3 |
+-----------+
mysql> select sqrt(2);
+--------------------+
| sqrt(2.0)          |
+--------------------+
| 1.4142135623730951 |
+--------------------+
mysql> select sqrt(100.0);
+-------------+
| sqrt(100.0) |
+-------------+
|          10 |
+-------------+
```

keywords

```
SQRT, DSQRT
```

CBRT

Description

Calculate the cube root of the parameter

Syntax

```
CBRT(<a>)
```

Parameters

Parameter	Description
<a>	Floating point parameter

Return Value

Cubic root of parameter <a>, a floating point number.

Examples

```
select cbrt(0);
```

```
+-------------------------+
| cbrt(cast(0 as DOUBLE)) |
+-------------------------+
|                     0.0 |
+-------------------------+
```

```
select cbrt(-111);
```

```
+----------------------------+
| cbrt(cast(-111 as DOUBLE)) |
+----------------------------+
|         -4.805895533705333 |
+----------------------------+
```

```
select cbrt(1234);
```

```
+----------------------------+
| cbrt(cast(1234 as DOUBLE)) |
+----------------------------+
|         10.726014668827325 |
+----------------------------+
```

POW

pow

description

Returns the value of the first argument raised to the power of the second argument.

> **The other aliases for this function are `power`, `fpow` and `dpow`.**

Syntax

`DOUBLE pow(DOUBLE a, DOUBLE b)` Returns `a` raised to the `b` power.

example

```
mysql> select pow(2,0);
+---------------+
| pow(2.0, 0.0) |
+---------------+
|             1 |
+---------------+
mysql> select pow(2,3);
+---------------+
| pow(2.0, 3.0) |
+---------------+
|             8 |
+---------------+
mysql> select pow(3,2.4);
+--------------------+
| pow(3.0, 2.4)      |
+--------------------+
| 13.966610165238235 |
+--------------------+
```

keywords

```
POW, POWER, FPOW, DPOW
```

DEGREES

Description

Input a double-precision floating-point number and convert it from radians to degrees.

- When the parameter is NULL, return NULL.

Syntax

```
DEGREES(<a>)
```

Parameters

parameter	explain
<a>	The value that needs to be converted from radians to degrees.

Return Value

The angle of parameter a.

- When the parameter is NULL, return NULL.

Examples

```
select degrees(3.14),degrees(1),degrees(-1),degrees(NULL)
```

```
+------------------------------+---------------------------+----------------------------+--------------+
| degrees(cast(3.14 as DOUBLE)) | degrees(cast(1 as DOUBLE)) | degrees(cast(-1 as DOUBLE)) | degrees(NULL) |
+------------------------------+---------------------------+----------------------------+--------------+
|             179.9087476710785 |          57.29577951308232 |          -57.29577951308232 |          NULL |
+------------------------------+---------------------------+----------------------------+--------------+
```

RADIANS

radians

description

Syntax

`DOUBLE radians(DOUBLE x)` Returns the value of `x` in radians, converted from degrees to radians.

example

```
mysql> select radians(0);
+--------------+
| radians(0.0) |
+--------------+
|            0 |
+--------------+
mysql> select radians(30);
+--------------------+
| radians(30.0)      |
+--------------------+
| 0.52359877559829882 |
+--------------------+
mysql> select radians(90);
+-------------------+
| radians(90.0)     |
+-------------------+
| 1.5707963267948966 |
+-------------------+
```

keywords

```
RADIANS
```

SIGN

sign

description

Syntax

`TINYINT sign(DOUBLE x)` Returns the sign of `x`. Negative, zero or positive numbers correspond to -1, 0 or 1 respectively.

example

```
mysql> select sign(3);
+-----------+
| sign(3.0) |
+-----------+
|         1 |
+-----------+
mysql> select sign(0);
+-----------+
| sign(0.0) |
+-----------+
|         0 |
mysql> select sign(-10.0);
+-------------+
| sign(-10.0) |
+-------------+
|          -1 |
+-------------+
1 row in set (0.01 sec)
```

keywords

```
SIGN
```

POSITIVE

positive

description

Syntax

```
BIGINT positive(BIGINT x)
DOUBLE positive(DOUBLE x)
DECIMAL positive(DECIMAL x)
```

Return x.

example

```
mysql> SELECT positive(-10);
+---------------+
| positive(-10) |
+---------------+
|           -10 |
+---------------+
mysql> SELECT positive(12);
+--------------+
| positive(12) |
+--------------+
|           12 |
+--------------+
```

keywords

```
POSITIVE
```

NEGATIVE

negative

description

Syntax

```
BIGINT negative(BIGINT x)
DOUBLE negative(DOUBLE x)
DECIMAL negative(DECIMAL x)
```

Return -x.

example

```
mysql> SELECT negative(-10);
+---------------+
| negative(-10) |
+---------------+
|            10 |
+---------------+
mysql> SELECT negative(12);
+--------------+
| negative(12) |
+--------------+
|          -12 |
+--------------+
```

keywords

```
NEGATIVE
```

GREATEST

Description

Compares multiple expressions and returns the greatest value among them. If any argument is `NULL`, the function returns `NULL`.

Syntax

```
GREATEST(<expr> [, ...])
```

Parameters

Parameter	Description
`<expr>`	A list of expressions to compare. Supports `TINYINT`, `SMALLINT`, `INT`, `BIGINT`, `LARGEINT`, `FLOAT`, `DOUBLE`, `STRING`, `DATETIME`, and `DECIMAL` types.

Return Value

- Returns the largest value among the given expressions.
- If any argument is `NULL`, returns `NULL`.

Examples

```
SELECT GREATEST(-1, 0, 5, 8);
```

```
+----------------------+
| GREATEST(-1, 0, 5, 8) |
+----------------------+
|                    8 |
+----------------------+
```

```
SELECT GREATEST(-1, 0, 5, NULL);
```

```
+-------------------------+
| GREATEST(-1, 0, 5, NULL) |
+-------------------------+
| NULL                    |
+-------------------------+
```

```
SELECT GREATEST(6.3, 4.29, 7.6876);
```

```
+----------------------------+
| GREATEST(6.3, 4.29, 7.6876) |
```

```
+-----------------------------+
|                      7.6876 |
+-----------------------------+
```

```
SELECT GREATEST('2022-02-26 20:02:11', '2020-01-23 20:02:11', '2020-06-22 20:02:11');
```

```
+---------------------------------------------------------------------------+
| GREATEST('2022-02-26 20:02:11', '2020-01-23 20:02:11', '2020-06-22 20:02:11') |
+---------------------------------------------------------------------------+
| 2022-02-26 20:02:11                                                       |
+---------------------------------------------------------------------------+
```

LEAST

Description

Compares multiple expressions and returns the smallest value among them. If any parameter is `NULL`, the function returns `NULL`.

Syntax

```
LEAST(<expr> [, ...])
```

Parameters

Parameter	Description
`<expr>`	The expressions to be compared. Supported types include `TINYINT`, `SMALLINT`, `INT`, `BIGINT`, `LARGEINT`, `FLOAT`, `DOUBLE`, `STRING`, `DATETIME`, and `DECIMAL`.

Return Value

- Returns the smallest value among the given expressions.
- If any parameter is `NULL`, the function returns `NULL`.

Examples

```
SELECT LEAST(-1, 0, 5, 8);
```

```
+--------------------+
| LEAST(-1, 0, 5, 8) |
+--------------------+
|                 -1 |
+--------------------+
```

```
SELECT LEAST(-1, 0, 5, NULL);
```

```
+-----------------------+
| LEAST(-1, 0, 5, NULL) |
+-----------------------+
| NULL                  |
+-----------------------+
```

```
SELECT LEAST(6.3, 4.29, 7.6876);
```

```
+--------------------------+
| LEAST(6.3, 4.29, 7.6876) |
+--------------------------+
|                     4.29 |
+--------------------------+
```

```
SELECT LEAST('2022-02-26 20:02:11', '2020-01-23 20:02:11', '2020-06-22 20:02:11');
```

```
+---------------------------------------------------------------------------+
| LEAST('2022-02-26 20:02:11', '2020-01-23 20:02:11', '2020-06-22 20:02:11') |
+---------------------------------------------------------------------------+
| 2020-01-23 20:02:11                                                       |
+---------------------------------------------------------------------------+
```

RANDOM

random

description

Syntax

`DOUBLE random()` Returns a random number between 0-1.

example

```
mysql> select random();
+--------------------+
| random()           |
+--------------------+
| 0.35446706030596947 |
+--------------------+
```

keywords

```
RANDOM
```

MOD

mod

description

Syntax

```
mod(col_a, col_b)
```

`column` support type：`TINYINT SMALLINT INT BIGINT LARGEINT FLOAT DOUBLE DECIMAL`

Find the remainder of a/b. For floating-point types, use the fmod function.

example

```
mysql> select mod(10, 3);
+------------+
| mod(10, 3) |
+------------+
|          1 |
+------------+

mysql> select fmod(10.1, 3.2);
+-----------------+
| fmod(10.1, 3.2) |
+-----------------+
|      0.50000024 |
+-----------------+
```

keywords

```
MOD, FMOD
```

RUNNING_DIFFERENCE

running_difference

description

Syntax

```
T running_difference(T x)
```

Calculates the difference between successive row values in the data block. The result of the function depends on the affected data blocks and the order of data in the block.

The rows order used during the calculation of running_difference can differ from the order of rows returned to the user. The function will be deprecated in the future. Please use window function instead, below is the example:

```
-- running difference(x)
SELECT running_difference(x) FROM t ORDER BY k;

-- window function
SELECT x - lag(x, 1, 0) OVER (ORDER BY k) FROM t;
```

Arguments

x - A list of data.TINYINT,SMALLINT,INT,BIGINT,LARGEINT,FLOAT,DOUBLE,DATE,DATETIME,DECIMAL

Returned value

Returns 0 for the first row and the difference from the previous row for each subsequent row

example

```
DROP TABLE IF EXISTS running_difference_test;

CREATE TABLE running_difference_test (
    `id` int NOT NULL COMMENT 'id',
    `day` date COMMENT 'day',
    `time_val` datetime COMMENT 'time_val',
    `doublenum` double NULL COMMENT 'doublenum'
)
DUPLICATE KEY(id)
DISTRIBUTED BY HASH(id) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT into running_difference_test (id, day, time_val,doublenum) values ('1', '2022-10-28', '2022-03-12 10:41:00', null),
                                                ('2','2022-10-27', '2022-03-12 10:41:02', 2.6),
```

```
                                                ('3','2022-10-28', '2022-03-12 10:41:03', 2.5),
                                                ('4','2022-9-29', '2022-03-12 10:41:03', null),
                                                ('5','2022-10-31', '2022-03-12 10:42:01', 3.3),
                                                ('6', '2022-11-08', '2022-03-12 11:05:04', 4.7);

SELECT * from running_difference_test ORDER BY id ASC;

+------+------------+---------------------+-----------+
| id   | day        | time_val            | doublenum |
+------+------------+---------------------+-----------+
|    1 | 2022-10-28 | 2022-03-12 10:41:00 |      NULL |
|    2 | 2022-10-27 | 2022-03-12 10:41:02 |       2.6 |
|    3 | 2022-10-28 | 2022-03-12 10:41:03 |       2.5 |
|    4 | 2022-09-29 | 2022-03-12 10:41:03 |      NULL |
|    5 | 2022-10-31 | 2022-03-12 10:42:01 |       3.3 |
|    6 | 2022-11-08 | 2022-03-12 11:05:04 |       4.7 |
+------+------------+---------------------+-----------+

SELECT
    id,
    running_difference(id) AS delta
FROM
(
    SELECT
        id,
        day,
        time_val,
        doublenum
    FROM running_difference_test
)as runningDifference ORDER BY id ASC;

+------+-------+
| id   | delta |
+------+-------+
|    1 |     0 |
|    2 |     1 |
|    3 |     1 |
|    4 |     1 |
|    5 |     1 |
|    6 |     1 |
+------+-------+

SELECT
    day,
    running_difference(day) AS delta
FROM
(
    SELECT
        id,
        day,
        time_val,
        doublenum
    FROM running_difference_test
)as runningDifference ORDER BY id ASC;

+------------+-------+
| day        | delta |
+------------+-------+
| 2022-10-28 |     0 |
| 2022-10-27 |    -1 |
| 2022-10-28 |     1 |
| 2022-09-29 |   -29 |
| 2022-10-31 |    32 |
| 2022-11-08 |     8 |
+------------+-------+
```

```
SELECT
    time_val,
    running_difference(time_val) AS delta
FROM
(
    SELECT
        id,
        day,
        time_val,
        doublenum
    FROM running_difference_test
)as runningDifference ORDER BY id ASC;

+---------------------+-------+
| time_val            | delta |
+---------------------+-------+
| 2022-03-12 10:41:00 |     0 |
| 2022-03-12 10:41:02 |     2 |
| 2022-03-12 10:41:03 |     1 |
| 2022-03-12 10:41:03 |     0 |
| 2022-03-12 10:42:01 |    58 |
| 2022-03-12 11:05:04 |  1383 |
+---------------------+-------+

SELECT
    doublenum,
    running_difference(doublenum) AS delta
FROM
(
    SELECT
        id,
        day,
        time_val,
        doublenum
    FROM running_difference_test
)as runningDifference ORDER BY id ASC;

+-----------+----------------------+
| doublenum | delta                |
+-----------+----------------------+
|      NULL |                 NULL |
|       2.6 |                 NULL |
|       2.5 | -0.10000000000000009 |
|      NULL |                 NULL |
|       3.3 |                 NULL |
|       4.7 |   1.4000000000000004 |
+-----------+----------------------+
```

keywords

running_difference

UUID_NUMERIC

uuid_numeric

description

Syntax

```
LARGEINT uuid_numeric()
```

Return a uuid in type `LARGEINT`.

Note that `LARGEINT` has type Int128, so we could get a negative number from `uuid_numeric()`.

example

```
mysql> select uuid_numeric();
+--------------------------------------+
| uuid_numeric()                       |
+--------------------------------------+
| 82218484683747862468445277894131281464 |
+--------------------------------------+
```

keywords

```
UUID UUID-NUMERIC
```

AES_ENCRYPT

Description

AES encryption function. This function behaves like the `AES_ENCRYPT` function in MySQL. By default, it uses the `AES_128_ECB` algorithm with `PKCS7` padding mode.

Syntax

```
VARCHAR AES_ENCRYPT(VARCHAR str, VARCHAR key_str [, VARCHAR init_vector [, VARCHAR encryption_mode]])
```

Parameters

- `str` is the text to be encrypted;
- `key_str` is the key. Note that this key is not a hexadecimal encoding, but a string representation of the encoded key. For example, for 128-bit key encryption, `key_str` should be 16-length. If the key is not long enough, use **zero padding** to make it up. If it is longer than that, the final key is found using a cyclic xor method. For example, if the 128-bit key used by the algorithm finally is `key`, then `key[i] = key_str[i] ^ key_str[i+128] ^ key_str[i+256] ^ ...`
- `init_vector` is the initial vector to be used in the algorithm, this is only valid for some algorithms, if not specified then Doris will use the built-in value;
- `encryption_mode` is the encryption algorithm, optionally available in variable.

> **Function with two arguments will ignore session variable `block_encryption_mode` and always use `AES_128_ECB` to do encryption. So it's not recommended to use it.**

Notice

For the incoming key, the AES_ENCRYPT function not directly uses, but will further process it. The specific steps are as follows:

1. According to the encryption algorithm used, determine the number of bytes of the key, for example, if you use the AES_128_ECB algorithm, the number of bytes of the key is `128 / 8 = 16` (if you use the AES_256_ECB algorithm, the number of bytes of the key is `256 / 8 = 32`). 2;
2. then for the key entered by the user, bits `i` and `16*k+i` are used to perform an isomorphism, followed by a zero if the key entered by the user is less than 16 bits. 3. finally, the newly generated key is used to generate a new key;
3. finally, the newly generated key is used for encryption.

Example

```
set block_encryption_mode='';
```

```
select to_base64(aes_encrypt('text','F3229A0B371ED2D9441B830D21A390C3'));
```

```
+------------------------------------------------------------+
| to_base64(aes_encrypt('text', '***', '', 'AES_128_ECB')) |
+------------------------------------------------------------+
| wr2JEDVXzL9+2XtRhgIloA==                                   |
+------------------------------------------------------------+
```

```
set block_encryption_mode="AES_256_CBC";
select to_base64(aes_encrypt('text','F3229A0B371ED2D9441B830D21A390C3'));
```

```
+------------------------------------------------------------+
| to_base64(aes_encrypt('text', '***', '', 'AES_256_CBC')) |
+------------------------------------------------------------+
| 3dymOE7/+1zbrLIaBVNHSw==                                   |
+------------------------------------------------------------+
```

```
select to_base64(aes_encrypt('text','F3229A0B371ED2D9441B830D21A390C3', '0123456789'));
```

```
+------------------------------------------------------------------+
| to_base64(aes_encrypt('text', '***', '0123456789', 'AES_256_CBC')) |
+------------------------------------------------------------------+
| tsmK1HzbpnEdR2//WhO+MA==                                         |
+------------------------------------------------------------------+
```

AES_DECRYPT

Description

AES decryption function. This function behaves like the `AES_DECRYPT` function in MySQL. By default, it uses the `AES_128_ECB` algorithm with `PKCS7` padding mode.

Syntax

```
VARCHAR AES_DECRYPT(VARCHAR str, VARCHAR key_str [, VARCHAR init_vector [, VARCHAR encryption_mode]])
```

Parameters

- `str` is the text to be decrypted;
- `key_str` is the key. Note that this key is not a hexadecimal encoding, but a string representation of the encoded key. For example, for 128-bit key encryption, `key_str` should be 16-length. If the key is not long enough, use **zero padding** to make it up. If it is longer than that, the final key is found using a cyclic xor method. For example, if the 128-bit key used by the algorithm finally is `key`, then `key[i] = key_str[i] ^ key_str[i+128] ^ key_str[i+256] ^ ...`
- `init_vector` is the initial vector to be used in the algorithm, this is only valid for some algorithms, if not specified then Doris will use the built-in value;
- `encryption_mode` is the encryption algorithm, optionally available in variable.

> **Function with two arguments will ignore session variable `block_encryption_mode` and always use `AES_128_ECB` to do decryption. So it's not recommended to use it.**

Example

```
set block_encryption_mode='';
select aes_decrypt(from_base64('wr2JEDVXzL9+2XtRhgIloA=='),'F3229A0B371ED2D9441B830D21A390C3');
```

```
+------------------------------------------------------------------------------+
| aes_decrypt(from_base64('wr2JEDVXzL9+2XtRhgIloA=='), '***', '', 'AES_128_ECB') |
+------------------------------------------------------------------------------+
| text                                                                         |
+------------------------------------------------------------------------------+
```

```
set block_encryption_mode="AES_256_CBC";
select aes_decrypt(from_base64('3dym0E7/+1zbrLIaBVNHSw=='),'F3229A0B371ED2D9441B830D21A390C3');
```

```
+------------------------------------------------------------------------------+
| aes_decrypt(from_base64('3dym0E7/+1zbrLIaBVNHSw=='), '***', '', 'AES_256_CBC') |
+------------------------------------------------------------------------------+
```

```
| text                                                                        |
+-----------------------------------------------------------------------------+
```

```
select AES_DECRYPT(FROM_BASE64('tsmK1HzbpnEdR2//WhO+MA=='),'F3229AOB371ED2D9441B830D21A390C3', '0123456789');
```

```
+-----------------------------------------------------------------------------------------+
| aes_decrypt(from_base64('tsmK1HzbpnEdR2//WhO+MA=='), '***', '0123456789', 'AES_256_CBC') |
+-----------------------------------------------------------------------------------------+
| text                                                                                    |
+-----------------------------------------------------------------------------------------+
```

MD5

description

Calculates an MD5 128-bit checksum for the string

Syntax

```
MD5(str)
```

example

```
select md5("abc");
```

```
+----------------------------------+
| md5('abc')                       |
+----------------------------------+
| 900150983cd24fb0d6963f7d28e17f72 |
+----------------------------------+
```

MD5SUM

description

Calculates an MD5 128-bit checksum for the strings

Syntax

```
MD5SUM(str[,str])
```

example

```
select md5("abcd");
```

```
+----------------------------------+
| md5('abcd')                      |
+----------------------------------+
| e2fc714c4727ee9395f324cd2e7f331f |
+----------------------------------+
```

```
select md5sum("ab","cd");
```

```
+----------------------------------+
| md5sum('ab', 'cd')               |
+----------------------------------+
| e2fc714c4727ee9395f324cd2e7f331f |
+----------------------------------+
```

SM4_ENCRYPT

Description

SM4 is a China's national standard symmetric key encryption algorithm, widely used in finance, communications, e-commerce, and other fields. The SM4_ENCRYPT function is used to encrypt data using the SM4 algorithm. By default, it uses the `SM4_128_ECB` algorithm.

Syntax

```
VARCHAR SM4_ENCRYPT(VARCHAR str, VARCHAR key_str[, VARCHAR init_vector][, VARCHAR encryption_mode])
```

Parameters

- `str` is the text to be encrypted;
- `key_str` is the key. Note that this key is not a hexadecimal encoding, but a string representation of the encoded key. For example, for 128-bit key encryption, `key_str` should be 16-length. If the key is not long enough, use **zero padding** to make it up. If it is longer than that, the final key is found using a cyclic xor method. For example, if the 128-bit key used by the algorithm finally is `key`, then `key[i] = key_str[i] ^ key_str[i+128] ^ key_str[i+256] ^ ...`
- `init_vector` is the initial vector to be used in the algorithm, this is only valid for some algorithms, if not specified then Doris will use the built-in value;
- `encryption_mode` is the encryption algorithm, optionally available in variable.

> **Function with two arguments will ignore session variable `block_encryption_mode` and always use `SM4_128_ECB` to do encryption. This is different from the 1.2 version behavior and must be taken into account when upgrading or downgrading.**

Example

```
set block_encryption_mode='';
select TO_BASE64(SM4_ENCRYPT('text','F3229A0B371ED2D9441B830D21A390C3'));
```

```
+-----------------------------------------------------------+
| to_base64(sm4_encrypt('text', '***', '', 'SM4_128_ECB')) |
+-----------------------------------------------------------+
| aDjwRflBrDjhBZIOFNw3Tg==                                  |
+-----------------------------------------------------------+
```

```
select to_base64(SM4_ENCRYPT('text','F3229A0B371ED2D9441B830D21A390C3', '0123456789'));
```

```
+----------------------------------------------------------------------------------+
| to_base64(sm4_encrypt('text', 'F3229A0B371ED2D9441B830D21A390C3', '0123456789')) |
+----------------------------------------------------------------------------------+
| G7yqOKfEyxdagboz6Qf01A==                                                         |
+----------------------------------------------------------------------------------+
```

SM4_DECRYPT

Description

SM4 is a China's national standard symmetric key encryption algorithm, widely used in finance, communications, e-commerce, and other fields. The SM4_DECRYPT function is used to decrypt data using the SM4 algorithm. By default, it uses the `SM4_128_ECB` algorithm.

Syntax

```
VARCHAR SM4_DECRYPT(VARCHAR str, VARCHAR key_str [, VARCHAR init_vector [, VARCHAR encryption_mode]])
```

Parameters

- `str` is the text to be decrypted;
- `key_str` is the key. Note that this key is not a hexadecimal encoding, but a string representation of the encoded key. For example, for 128-bit key encryption, `key_str` should be 16-length. If the key is not long enough, use **zero padding** to make it up. If it is longer than that, the final key is found using a cyclic xor method. For example, if the 128-bit key used by the algorithm finally is `key`, then `key[i] = key_str[i] ^ key_str[i+128] ^ key_str[i+256] ^ ...`
- `init_vector` is the initial vector to be used in the algorithm, this is only valid for some algorithms, if not specified then Doris will use the built-in value;
- `encryption_mode` is the encryption algorithm, optionally available in variable.

> **Function with two arguments will ignore session variable `block_encryption_mode` and always use `SM4_128_ECB` to do decryption. This is different from the 1.2 version behavior and must be taken into account when upgrading or downgrading.**

Example

```
set block_encryption_mode='';
select SM4_DECRYPT(FROM_BASE64('aDjwRflBrDjhBZIOFNw3Tg=='),'F3229A0B371ED2D9441B830D21A390C3');
```

```
+------------------------------------------------------------------------------+
| sm4_decrypt(from_base64('aDjwRflBrDjhBZIOFNw3Tg=='), '***', '', 'SM4_128_ECB') |
+------------------------------------------------------------------------------+
| text                                                                         |
+------------------------------------------------------------------------------+
```

```
select SM4_DECRYPT(FROM_BASE64('G7yqOKfEyxdagboz6Qf01A=='),'F3229A0B371ED2D9441B830D21A390C3', '0123456789');
```

```
+-----------------------------------------------------------------------------------------------------+
| sm4_decrypt(from_base64('G7yqOKfEyxdagboz6Qf01A=='), 'F3229A0B371ED2D9441B830D21A390C3', '0123456789') |
+-----------------------------------------------------------------------------------------------------+
| text                                                                                                |
+-----------------------------------------------------------------------------------------------------+
```

SM3

description

Calculates an SM3 256-bit checksum for the string

Syntax

```
SM3(str)
```

example

```
select sm3("abcd");
```

```
+------------------------------------------------------------------+
| sm3('abcd')                                                      |
+------------------------------------------------------------------+
| 82ec580fe6d36ae4f81cae3c73f4a5b3b5a09c943172dc9053c69fd8e18dca1e |
+------------------------------------------------------------------+
```

SM3SUM

description

Calculates an SM3 128-bit checksum for the strings

Syntax

```
SM3SUM(str[,str])
```

example

```
select sm3("abcd");
```

```
+------------------------------------------------------------------+
| sm3('abcd')                                                      |
+------------------------------------------------------------------+
| 82ec580fe6d36ae4f81cae3c73f4a5b3b5a09c943172dc9053c69fd8e18dca1e |
+------------------------------------------------------------------+
```

```
select sm3sum("ab","cd");
```

```
+------------------------------------------------------------------+
| sm3sum('ab', 'cd')                                               |
+------------------------------------------------------------------+
| 82ec580fe6d36ae4f81cae3c73f4a5b3b5a09c943172dc9053c69fd8e18dca1e |
+------------------------------------------------------------------+
```

SHA

description

Use SHA1 to digest the message.

Syntax

```
{ SHA | SHA1 }(str)
```

example

```
select sha("123");
```

```
+------------------------------------------+
| sha1('123')                              |
+------------------------------------------+
| 40bd001563085fc35165329ea1ff5c5ecbdbbeef |
+------------------------------------------+
```

SHA2

description

Use SHA2 to digest the message.

Syntax

```
SHA2(str, digest_length)
```

Paramters

- `str`: content to be encrypted
- `digest_length`: the length of the digest. Support 224, 256, 384, 512

example

```
select sha2('abc', 224);
```

```
+----------------------------------------------------------+
| sha2('abc', 224)                                         |
+----------------------------------------------------------+
| 23097d223405d8228642a477bda255b32aadbce4bda0b3f7e36c9da7 |
+----------------------------------------------------------+
```

```
select sha2('abc', 384);
```

```
+------------------------------------------------------------------------------------------------+
| sha2('abc', 384)                                                                               |
+------------------------------------------------------------------------------------------------+
| cb00753f45a35e8bb5a03d699ac65007272c32ab0eded1631a8b605a43ff5bed8086072ba1e7cc2358baeca134c825a7 |
+------------------------------------------------------------------------------------------------+
```

```
select sha2(NULL, 512);
```

```
+-----------------+
| sha2(NULL, 512) |
+-----------------+
| NULL            |
+-----------------+
```

EXPLODE_JSON_ARRAY

explode_json_array

description

Table functions must be used in conjunction with Lateral View.

Expand a json array. According to the array element type, there are three function names. Corresponding to integer, floating point and string arrays respectively.

syntax

```
explode_json_array_int(json_str)
explode_json_array_double(json_str)
explode_json_array_string(json_str)
explode_json_array_json(json_str)
```

example

Original table data:

```
mysql> select k1 from example1 order by k1;
+------+
| k1   |
+------+
|    1 |
|    2 |
|    3 |
+------+
```

Lateral View:

```
mysql> select k1, e1 from example1 lateral view explode_json_array_int('[]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    2 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_int('[1,2,3]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 |    1 |
|    1 |    2 |
|    1 |    3 |
|    2 |    1 |
|    2 |    2 |
```

```
|    2 |    3 |
|    3 |    1 |
|    3 |    2 |
|    3 |    3 |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_int('[1,"b",3]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    1 |    1 |
|    1 |    3 |
|    2 | NULL |
|    2 |    1 |
|    2 |    3 |
|    3 | NULL |
|    3 |    1 |
|    3 |    3 |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_int('["a","b","c"]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    1 | NULL |
|    1 | NULL |
|    2 | NULL |
|    2 | NULL |
|    2 | NULL |
|    3 | NULL |
|    3 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_int('{"a": 3}') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    2 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_double('[]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    2 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_double('[1,2,3]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    1 | NULL |
|    1 | NULL |
|    2 | NULL |
|    2 | NULL |
|    2 | NULL |
|    3 | NULL |
```

```
|    3 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_double('[1,"b",3]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    1 | NULL |
|    1 | NULL |
|    2 | NULL |
|    2 | NULL |
|    2 | NULL |
|    3 | NULL |
|    3 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_double('[1.0,2.0,3.0]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 |    1 |
|    1 |    2 |
|    1 |    3 |
|    2 |    1 |
|    2 |    2 |
|    2 |    3 |
|    3 |    1 |
|    3 |    2 |
|    3 |    3 |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_double('[1,"b",3]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    1 | NULL |
|    1 | NULL |
|    2 | NULL |
|    2 | NULL |
|    2 | NULL |
|    3 | NULL |
|    3 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_double('["a","b","c"]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    1 | NULL |
|    1 | NULL |
|    2 | NULL |
|    2 | NULL |
|    2 | NULL |
|    3 | NULL |
|    3 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_double('{"a": 3}') tmp1 as e1 order by k1, e1;
```

```
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    2 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_string('[]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    2 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_string('[1.0,2.0,3.0]') tmp1 as e1 order by k1, e1;
+------+----------+
| k1   | e1       |
+------+----------+
|    1 | 1.000000 |
|    1 | 2.000000 |
|    1 | 3.000000 |
|    2 | 1.000000 |
|    2 | 2.000000 |
|    2 | 3.000000 |
|    3 | 1.000000 |
|    3 | 2.000000 |
|    3 | 3.000000 |
+------+----------+

mysql> select k1, e1 from example1 lateral view explode_json_array_string('[1,"b",3]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | 1    |
|    1 | 3    |
|    1 | b    |
|    2 | 1    |
|    2 | 3    |
|    2 | b    |
|    3 | 1    |
|    3 | 3    |
|    3 | b    |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_string('["a","b","c"]') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | a    |
|    1 | b    |
|    1 | c    |
|    2 | a    |
|    2 | b    |
|    2 | c    |
|    3 | a    |
|    3 | b    |
|    3 | c    |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_string('{"a": 3}') tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
```

```
+------+------+
|    1 | NULL |
|    2 | NULL |
|    3 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_json_array_json('[{"id":1,"name":"John"},{"id":2,"name":"Mary"},
{"id":3,"name":"Bob"}]') tmp1 as e1 order by k1, e1;
+------+-----------------------+
| k1   | e1                    |
+------+-----------------------+
|    1 | {"id":1,"name":"John"} |
|    1 | {"id":2,"name":"Mary"} |
|    1 | {"id":3,"name":"Bob"}  |
|    2 | {"id":1,"name":"John"} |
|    2 | {"id":2,"name":"Mary"} |
|    2 | {"id":3,"name":"Bob"}  |
|    3 | {"id":1,"name":"John"} |
|    3 | {"id":2,"name":"Mary"} |
|    3 | {"id":3,"name":"Bob"}  |
+------+-----------------------+
```

keywords

explode,json,array,json_array,explode_json,explode_json_array

EXPLODE

explode

description

Table functions must be used in conjunction with Lateral View.

explode array column to rows. `explode_outer` will return NULL, while `array` is NULL or empty. `explode` and `explode_outer` both keep the nested NULL elements of array.

syntax

```
explode(expr)
explode_outer(expr)
```

example

```
mysql> select e1 from (select 1 k1) as t lateral view explode([1,2,3]) tmp1 as e1;
+------+
| e1   |
+------+
|    1 |
|    2 |
|    3 |
+------+

mysql> select e1 from (select 1 k1) as t lateral view explode_outer(null) tmp1 as e1;
+------+
| e1   |
+------+
| NULL |
+------+

mysql> select e1 from (select 1 k1) as t lateral view explode([]) tmp1 as e1;
Empty set (0.010 sec)

mysql> select e1 from (select 1 k1) as t lateral view explode([null,1,null]) tmp1 as e1;
+------+
| e1   |
+------+
| NULL |
|    1 |
| NULL |
+------+

mysql> select e1 from (select 1 k1) as t lateral view explode_outer([null,1,null]) tmp1 as e1;
+------+
| e1   |
+------+
| NULL |
|    1 |
```

```
| NULL |
+------+
```

keywords

EXPLODE,EXPLODE_OUTER,ARRAY

EXPLODE_SPLIT

explode_split

description

syntax

```
explode_split(str, delimiter)
```

Table functions must be used in conjunction with Lateral View.

Split a string into multiple substrings according to the specified delimiter.

grammar:

```
explode_split(str, delimiter)
```

example

Original table data:

```
mysql> select * from example1 order by k1;
+------+---------+
| k1   | k2      |
+------+---------+
|    1 |         |
|    2 | NULL    |
|    3 | ,       |
|    4 | 1       |
|    5 | 1,2,3   |
|    6 | a, b, c |
+------+---------+
```

Lateral View:

```
mysql> select k1, e1 from example1 lateral view explode_split(k2, ',') tmp1 as e1 where k1 = 1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 |      |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_split(k2, ',') tmp1 as e1 where k1 = 2 order by k1, e1;
Empty set

mysql> select k1, e1 from example1 lateral view explode_split(k2, ',') tmp1 as e1 where k1 = 3 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    3 |      |
+------+------+
```

```
mysql> select k1, e1 from example1 lateral view explode_split(k2, ',') tmp1 as e1 where k1 = 4 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    4 | 1    |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_split(k2, ',') tmp1 as e1 where k1 = 5 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    5 | 1    |
|    5 | 2    |
|    5 | 3    |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_split(k2, ',') tmp1 as e1 where k1 = 6 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    6 |  b   |
|    6 |  c   |
|    6 | a    |
+------+------+
```

keywords

explode,split,explode_split

EXPLODE_BITMAP

explode_bitmap

description

Table functions must be used in conjunction with Lateral View.

Expand a bitmap type.

syntax

```
explode_bitmap(bitmap)
```

example

Original table data:

```
mysql> select k1 from example1 order by k1;
+------+
| k1   |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+
```

Lateral View:

```
mysql> select k1, e1 from example1 lateral view explode_bitmap(bitmap_empty()) tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 | NULL |
|    2 | NULL |
|    3 | NULL |
|    4 | NULL |
|    5 | NULL |
|    6 | NULL |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_bitmap(bitmap_from_string("1")) tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 |    1 |
|    2 |    1 |
|    3 |    1 |
|    4 |    1 |
|    5 |    1 |
```

```
|    6 |    1 |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_bitmap(bitmap_from_string("1,2")) tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 |    1 |
|    1 |    2 |
|    2 |    1 |
|    2 |    2 |
|    3 |    1 |
|    3 |    2 |
|    4 |    1 |
|    4 |    2 |
|    5 |    1 |
|    5 |    2 |
|    6 |    1 |
|    6 |    2 |
+------+------+

mysql> select k1, e1 from example1 lateral view explode_bitmap(bitmap_from_string("1,1000")) tmp1 as e1 order by k1, e1;
+------+------+
| k1   | e1   |
+------+------+
|    1 |    1 |
|    1 | 1000 |
|    2 |    1 |
|    2 | 1000 |
|    3 |    1 |
|    3 | 1000 |
|    4 |    1 |
|    4 | 1000 |
|    5 |    1 |
|    5 | 1000 |
|    6 |    1 |
|    6 | 1000 |
+------+------+

mysql> select k1, e1, e2 from example1
lateral view explode_bitmap(bitmap_from_string("1,1000")) tmp1 as e1
lateral view explode_split("a,b", ",") tmp2 as e2 order by k1, e1, e2;
+------+------+------+
| k1   | e1   | e2   |
+------+------+------+
|    1 |    1 | a    |
|    1 |    1 | b    |
|    1 | 1000 | a    |
|    1 | 1000 | b    |
|    2 |    1 | a    |
|    2 |    1 | b    |
|    2 | 1000 | a    |
|    2 | 1000 | b    |
|    3 |    1 | a    |
|    3 |    1 | b    |
|    3 | 1000 | a    |
|    3 | 1000 | b    |
|    4 |    1 | a    |
|    4 |    1 | b    |
|    4 | 1000 | a    |
|    4 | 1000 | b    |
|    5 |    1 | a    |
|    5 |    1 | b    |
|    5 | 1000 | a    |
|    5 | 1000 | b    |
```

```
|    6 |    1 | a    |
|    6 |    1 | b    |
|    6 | 1000 | a    |
|    6 | 1000 | b    |
+------+------+------+
```

keywords

explode,bitmap,explode_bitmap

NUMBERS

numbers

description

Table function that generates a temporary table containing only one column with the column name number and all element values are const_value if const_value is specified, otherwise they are [0,number) incremented.

syntax

```
numbers(
  "number" = "n"
  <, "const_value" = "x">
  );
```

parameter： - number: Line number. - const_value: the constant value.

example

```
mysql> select * from numbers("number" = "5");
+--------+
| number |
+--------+
|      0 |
|      1 |
|      2 |
|      3 |
|      4 |
+--------+
5 rows in set (0.11 sec)

mysql> select * from numbers("number" = "5", "const_value" = "-123");
+--------+
| number |
+--------+
|   -123 |
|   -123 |
|   -123 |
|   -123 |
|   -123 |
+--------+
5 rows in set (0.12 sec)
```

keywords

```
numbers, const_value
```

EXPLODE_NUMBERS

explode_numbers

description

Table functions must be used in conjunction with Lateral View.

Get a number sequence [0,n).

syntax

```
explode_numbers(n)
```

example

```
mysql> select e1 from (select 1 k1) as t lateral view explode_numbers(5) tmp1 as e1;
+------+
| e1   |
+------+
|    0 |
|    1 |
|    2 |
|    3 |
|    4 |
+------+
```

keywords

explode,numbers,explode_numbers

EXPLODE_NUMBERS_OUTER

outer combinator

description

syntax

```
explode_numbers(INT x)
```

Adding the `_outer` suffix after the function name of the table function changes the function behavior from `non-outer` to `outer`, and adds a row of `Null` data when the table function generates 0 rows of data.

example

```
mysql> select e1 from (select 1 k1) as t lateral view explode_numbers(0) tmp1 as e1;
Empty set

mysql> select e1 from (select 1 k1) as t lateral view explode_numbers_outer(0) tmp1 as e1;
+------+
| e1   |
+------+
| NULL |
+------+
```

keywords

```
outer
```

S3

S3

Name

S3

description

S3 table-valued-function(tvf), allows users to read and access file contents on S3-compatible object storage, just like accessing relational table. Currently supports `csv/csv_with_names/csv_with_names_and_types/json/parquet/orc` file format.

syntax

```
s3(
  "uri" = "..",
  "s3.access_key" = "...",
  "s3.secret_key" = "...",
  "s3.region" = "...",
  "format" = "csv",
  "keyn" = "valuen",
  ...
  );
```

parameter description

Each parameter in S3 tvf is a pair of `"key"="value"`.

Related parameters for accessing S3:

- `uri`: (required) The S3 tvf will decide whether to use the path style access method according to the `use_path_style` parameter, and the default access method is the virtual-hosted style method.
- `s3.access_key`： (required)
- `s3.secret_key`： (required)
- `s3.region`： (optional). Mandatory if the Minio has set another region. Otherwise, `us-east-1` is used by default.
- `s3.session_token`： (optional)
- `use_path_style`: (optional) default `false` . The S3 SDK uses the virtual-hosted style by default. However, some object storage systems may not be enabled or support virtual-hosted style access. At this time, we can add the `use_path_style` parameter to force the use of path style access method.
- `force_parsing_by_standard_uri`: (optional) default `false` . We can add `force_parsing_by_standard_uri` parameter to force parsing unstandard uri as standard uri.

> Note: For AWS S3, standard uri styles should be:

> 1. AWS Client Style(Hadoop S3 Style): `s3://my-bucket/path/to/file?versionId=abc123&partNumber=77&partNumber=88`
> 2. Virtual Host Style: `https://my-bucket.s3.us-west-1.amazonaws.com/resources/doc.txt?versionId=abc123&partNumber=77&partNumber=88`
> 3. Path Style: `https://s3.us-west-1.amazonaws.com/my-bucket/resources/doc.txt?versionId=abc123&partNumber=77&partNumber=88`
>
> In addition to supporting the common uri styles of the above three standards, it also supports some other uri styles (maybe not common, but there may be):
>
> 1. Virtual Host AWS Client (Hadoop S3) Mixed Style: `s3://my-bucket.s3.us-west-1.amazonaws.com/resources/doc.txt?versionId=abc123&partNumber=77&partNumber=88`
> 2. Path AWS Client (Hadoop S3) Mixed Style: `s3://s3.us-west-1.amazonaws.com/my-bucket/resources/doc.txt?versionId=abc123&partNumber=77&partNumber=88`
>
> For detailed use cases, you can refer to Best Practice at the bottom.

file format parameter:

- `format`: (required) Currently support `csv/csv_with_names/csv_with_names_and_types/json/parquet/orc`
- `column_separator`: (optional) default `\t`.
- `line_delimiter`: (optional) default `\n`.
- `compress_type`: (optional) Currently support `UNKNOWN/PLAIN/GZ/LZO/BZ2/LZ4FRAME/DEFLATE/SNAPPYBLOCK`. Default value is `UNKNOWN`, it will automatically infer the type based on the suffix of `uri`.

The following 6 parameters are used for loading in json format. For specific usage methods, please refer to: Json Load

- `read_json_by_line`: (optional) default `"true"`
- `strip_outer_array`: (optional) default `"false"`
- `json_root`: (optional) default `""`
- `jsonpaths`: (optional) default `""`
- `num_as_string`: (optional) default `"false"`
- `fuzzy_parse`: (optional) default `"false"`

The following 2 parameters are used for loading in csv format

- `trim_double_quotes`: Boolean type (optional), the default value is `false`. True means that the outermost double quotes of each field in the csv file are trimmed.
- `skip_lines`: Integer type (optional), the default value is 0. It will skip some lines in the head of csv file. It will be disabled when the format is `csv_with_names` or `csv_with_names_and_types`.

other parameter:

- `path_partition_keys`: (optional) Specifies the column names carried in the file path. For example, if the file path is /path/to/city=beijing/date="2023-07-09", you should fill in `path_partition_keys="city,date"`. It will automatically read the corresponding column names and values from the path during load process.
- `resource`：(optional) Specify the resource name. S3 tvf can use the existing S3 resource to directly access S3. You can refer to the method for creating an S3 resource: CREATE-RESOURCE. This property is supported starting from version 2.1.4 .

> **Tip**
>
> To directly query a TVF or create a VIEW based on that TVF, you need to have usage permission for that resource. To query a VIEW created based on TVF, you only need select permission for that VIEW.

Example

Read and access csv format files on S3-compatible object storage.

```
select * from s3("uri" = "http://127.0.0.1:9312/test2/student1.csv",
                 "s3.access_key"= "minioadmin",
                 "s3.secret_key" = "minioadmin",
                 "format" = "csv",
                 "use_path_style" = "true") order by c1;
```

Can be used with `desc function`

```
MySQL [(none)]> Desc function s3("uri" = "http://127.0.0.1:9312/test2/student1.csv",
                  "s3.access_key"= "minioadmin",
                  "s3.secret_key" = "minioadmin",
                  "format" = "csv",
                  "use_path_style" = "true");
```

Keywords

```
s3, table-valued-function, tvf
```

Best Practice

Since the S3 table-valued-function does not know the table schema in advance, it will read the file first to parse out the table schema.

Usage of different uri schemas Example of http:// 、https://

```
// Note how to write your bucket of URI and set the 'use_path_style' parameter, as well as http://.
// Because of "use_path_style"="true", s3 will be accessed in 'path style'.
select * from s3(
    "URI" = "https://endpoint/bucket/file/student.csv",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "csv",
    "use_path_style"="true");

// Note how to write your bucket of URI and set the 'use_path_style' parameter, as well as http://.
// Because of "use_path_style"="false", s3 will be accessed in 'virtual-hosted style'.
select * from s3(
    "URI" = "https://bucket.endpoint/file/student.csv",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "csv",
    "use_path_style"="false");
```

```
// The OSS on Alibaba Cloud and The COS on Tencent Cloud will use 'virtual-hosted style' to access s3.
// OSS
select * from s3(
    "URI" = "http://example-bucket.oss-cn-beijing.aliyuncs.com/your-folder/file.parquet",
    "s3.access_key" = "ak",
    "s3.secret_key" = "sk",
    "region" = "oss-cn-beijing",
    "format" = "parquet",
    "use_path_style" = "false");
// COS
select * from s3(
    "URI" = "https://example-bucket.cos.ap-hongkong.myqcloud.com/your-folder/file.parquet",
    "s3.access_key" = "ak",
    "s3.secret_key" = "sk",
    "region" = "ap-hongkong",
    "format" = "parquet",
    "use_path_style" = "false");

// The BOS on Baidu Cloud will use 'virtual-hosted style' compatible with the S3 protocol to access s3.
// BOS
select * from s3(
    "uri" = "https://example-bucket.s3.bj.bcebos.com/your-folder/file.parquet",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "s3.region" = "bj",
    "format" = "parquet",
    "use_path_style" = "false");
```

// MinIO select * from s3(“uri” = “s3://bucket/file.csv”, “s3.endpoint” = “http://172.21.0.101:9000”, “s3.access_key”= “ak”, “s3.secret_key” = “sk”, “s3.region” = “us-east-1”, “format” = “csv”);

Example of s3://:

```
// Note how to write your bucket of URI, no need to set 'use_path_style'.
// s3 will be accessed in 'virtual-hosted style'.
select * from s3(
    "URI" = "s3://bucket/file/student.csv",
    "s3.endpoint"= "endpont",
    "s3.region" = "region",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "csv");
```

Example of other uri styles:

```
// Virtual Host AWS Client (Hadoop S3) Mixed Style. Used by setting `use_path_style = false` and `force_parsing_by_standard_uri =
true`.
select * from s3(
    "URI" = "s3://my-bucket.s3.us-west-1.amazonaws.com/resources/doc.txt?versionId=abc123&partNumber=77&partNumber=88",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
    "format" = "csv",
    "use_path_style"="false",
    "force_parsing_by_standard_uri"="true");

// Path AWS Client (Hadoop S3) Mixed Style. Used by setting `use_path_style = true` and `force_parsing_by_standard_uri = true`.
select * from s3(
    "URI" = "s3://s3.us-west-1.amazonaws.com/my-bucket/resources/doc.txt?versionId=abc123&partNumber=77&partNumber=88",
    "s3.access_key"= "ak",
    "s3.secret_key" = "sk",
```

```
    "format" = "csv",
    "use_path_style"="true",
    "force_parsing_by_standard_uri"="true");
```

csv format `csv` format: Read the file on S3 and process it as a csv file, read the first line in the file to parse out the table schema. The number of columns in the first line of the file `n` will be used as the number of columns in the table schema, and the column names of the table schema will be automatically named `c1, c2, ..., cn`, and the column type is set to `String` , for example:

The file content of student1.csv:

```
1,ftw,12
2,zs,18
3,ww,20
```

use S3 tvf

```
MySQL [(none)]> select * from s3("uri" = "http://127.0.0.1:9312/test2/student1.csv",
->                 "s3.access_key"= "minioadmin",
->                 "s3.secret_key" = "minioadmin",
->                 "format" = "csv",
->                 "use_path_style" = "true") order by c1;
+------+------+------+
| c1   | c2   | c3   |
+------+------+------+
| 1    | ftw  | 12   |
| 2    | zs   | 18   |
| 3    | ww   | 20   |
+------+------+------+
```

use `desc function S3()` to view the table schema

```
MySQL [(none)]> Desc function s3("uri" = "http://127.0.0.1:9312/test2/student1.csv",
->                 "s3.access_key"= "minioadmin",
->                 "s3.secret_key" = "minioadmin",
->                 "format" = "csv",
->                 "use_path_style" = "true");
+-------+------+------+-------+---------+-------+
| Field | Type | Null | Key   | Default | Extra |
+-------+------+------+-------+---------+-------+
| c1    | TEXT | Yes  | false | NULL    | NONE  |
| c2    | TEXT | Yes  | false | NULL    | NONE  |
| c3    | TEXT | Yes  | false | NULL    | NONE  |
+-------+------+------+-------+---------+-------+
```

csv_with_names format `csv_with_names` format: The first line of the file is used as the number and name of the columns of the table schema, and the column type is set to `String`, for example:

The file content of student_with_names.csv:

```
id,name,age
1,ftw,12
2,zs,18
3,ww,20
```

use S3 tvf

```
MySQL [(none)]> select * from s3("uri" = "http://127.0.0.1:9312/test2/student_with_names.csv",
->                 "s3.access_key"= "minioadmin",
->                 "s3.secret_key" = "minioadmin",
->                 "format" = "csv_with_names",
->                 "use_path_style" = "true") order by id;
+------+------+------+
| id   | name | age  |
+------+------+------+
| 1    | ftw  | 12   |
| 2    | zs   | 18   |
| 3    | ww   | 20   |
+------+------+------+
```

```
MySQL [(none)]> Desc function s3("uri" = "http://127.0.0.1:9312/test2/student_with_names.csv",
->                 "s3.access_key"= "minioadmin",
->                 "s3.secret_key" = "minioadmin",
->                 "format" = "csv_with_names",
->                 "use_path_style" = "true");
+-------+------+------+-------+---------+-------+
| Field | Type | Null | Key   | Default | Extra |
+-------+------+------+-------+---------+-------+
| id    | TEXT | Yes  | false | NULL    | NONE  |
| name  | TEXT | Yes  | false | NULL    | NONE  |
| age   | TEXT | Yes  | false | NULL    | NONE  |
+-------+------+------+-------+---------+-------+
```

csv_with_names_and_types format

`csv_with_names_and_types` format: Currently, it does not support parsing the column type from a csv file. When using this format, S3 tvf will parse the first line of the file as the number and name of the columns of the table schema, and set the column type to String. Meanwhile, the second line of the file is ignored.

The file content of student_with_names_and_types.csv:

```
id,name,age
INT,STRING,INT
1,ftw,12
2,zs,18
3,ww,20
```

use S3 tvf

```
MySQL [(none)]> select * from s3("uri" = "http://127.0.0.1:9312/test2/student_with_names_and_types.csv",
->                 "s3.access_key"= "minioadmin",
->                 "s3.secret_key" = "minioadmin",
->                 "format" = "csv_with_names_and_types",
->                 "use_path_style" = "true") order by id;
+------+------+------+
| id   | name | age  |
+------+------+------+
| 1    | ftw  | 12   |
| 2    | zs   | 18   |
| 3    | ww   | 20   |
+------+------+------+
```

```
MySQL [(none)]> Desc function s3("uri" = "http://127.0.0.1:9312/test2/student_with_names_and_types.csv",
->                 "s3.access_key"= "minioadmin",
->                 "s3.secret_key" = "minioadmin",
->                 "format" = "csv_with_names_and_types",
->                 "use_path_style" = "true");
+-------+------+------+-------+---------+-------+
| Field | Type | Null | Key   | Default | Extra |
+-------+------+------+-------+---------+-------+
| id    | TEXT | Yes  | false | NULL    | NONE  |
| name  | TEXT | Yes  | false | NULL    | NONE  |
| age   | TEXT | Yes  | false | NULL    | NONE  |
+-------+------+------+-------+---------+-------+
```

json format

`json` format: The json format involves many optional parameters, and the meaning of each parameter can be referred to: Json Load. When S3 tvf queries the json format file, it locates a json object according to the `json_root` and `jsonpaths` parameters, and uses the `key` in the object as the column name of the table schema, and sets the column type to String. For example:

The file content of data.json:

```
[{"id":1, "name":"ftw", "age":18}]
[{"id":2, "name":"xxx", "age":17}]
[{"id":3, "name":"yyy", "age":19}]
```

use S3 tvf:

```
MySQL [(none)]> select * from s3(
    "URI" = "http://127.0.0.1:9312/test2/data.json",
    "s3.access_key"= "minioadmin",
    "s3.secret_key" = "minioadmin",
    "format" = "json",
    "strip_outer_array" = "true",
    "read_json_by_line" = "true",
    "use_path_style"="true");
+------+------+------+
| id   | name | age  |
+------+------+------+
| 1    | ftw  | 18   |
| 2    | xxx  | 17   |
| 3    | yyy  | 19   |
+------+------+------+

MySQL [(none)]> select * from s3(
    "URI" = "http://127.0.0.1:9312/test2/data.json",
    "s3.access_key"= "minioadmin",
    "s3.secret_key" = "minioadmin",
    "format" = "json",
    "strip_outer_array" = "true",
    "jsonpaths" = "[\"$.id\", \"$.age\"]",
    "use_path_style"="true");
+------+------+
| id   | age  |
+------+------+
| 1    | 18   |
| 2    | 17   |
```

```
| 3    | 19   |
+------+------+
```

parquet format

`parquet` format: S3 tvf supports parsing the column names and column types of the table schema from the parquet file. Example:

```
MySQL [(none)]> select * from s3(
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "minioadmin",
    "s3.secret_key" = "minioadmin",
    "format" = "parquet",
    "use_path_style"="true") limit 5;
+-----------+-----------------------------------------+----------------+----------+-------------------------+--------+-------------
+---------------+---------------------+
| p_partkey | p_name                                  | p_mfgr         | p_brand  | p_type                  | p_size | p_container
| p_retailprice | p_comment           |
+-----------+-----------------------------------------+----------------+----------+-------------------------+--------+-------------
+---------------+---------------------+
|         1 | goldenrod lavender spring chocolate lace | Manufacturer#1 | Brand#13 | PROMO BURNISHED COPPER  |      7 | JUMBO PKG
|           901 | ly. slyly ironi     |
|         2 | blush thistle blue yellow saddle         | Manufacturer#1 | Brand#13 | LARGE BRUSHED BRASS     |      1 | LG CASE
|           902 | lar accounts amo    |
|         3 | spring green yellow purple cornsilk      | Manufacturer#4 | Brand#42 | STANDARD POLISHED BRASS |     21 | WRAP CASE
|           903 | egular deposits hag |
|         4 | cornflower chocolate smoke green pink    | Manufacturer#3 | Brand#34 | SMALL PLATED BRASS      |     14 | MED DRUM
|           904 | p furiously r       |
|         5 | forest brown coral puff cream            | Manufacturer#3 | Brand#32 | STANDARD POLISHED TIN   |     15 | SM PKG
|           905 |  wake carefully     |
+-----------+-----------------------------------------+----------------+----------+-------------------------+--------+-------------
+---------------+---------------------+
```

```
MySQL [(none)]> desc function s3(
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "s3.access_key"= "minioadmin",
    "s3.secret_key" = "minioadmin",
    "format" = "parquet",
    "use_path_style"="true");
+---------------+--------------+------+-------+---------+-------+
| Field         | Type         | Null | Key   | Default | Extra |
+---------------+--------------+------+-------+---------+-------+
| p_partkey     | INT          | Yes  | false | NULL    | NONE  |
| p_name        | TEXT         | Yes  | false | NULL    | NONE  |
| p_mfgr        | TEXT         | Yes  | false | NULL    | NONE  |
| p_brand       | TEXT         | Yes  | false | NULL    | NONE  |
| p_type        | TEXT         | Yes  | false | NULL    | NONE  |
| p_size        | INT          | Yes  | false | NULL    | NONE  |
| p_container   | TEXT         | Yes  | false | NULL    | NONE  |
| p_retailprice | DECIMAL(9,0) | Yes  | false | NULL    | NONE  |
| p_comment     | TEXT         | Yes  | false | NULL    | NONE  |
+---------------+--------------+------+-------+---------+-------+
```

orc format

`orc` format: Same as `parquet` format, set `format` parameter to orc.

```
MySQL [(none)]> select * from s3(
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.orc",
```

```
    "s3.access_key"= "minioadmin",
    "s3.secret_key" = "minioadmin",
    "format" = "orc",
    "use_path_style"="true") limit 5;
+-----------+----------------------------------------+---------------+----------+-------------------------+--------+-------------
+---------------+--------------------+
| p_partkey | p_name                                 | p_mfgr         | p_brand  | p_type                  | p_size | p_container
| p_retailprice | p_comment           |
+-----------+----------------------------------------+---------------+----------+-------------------------+--------+-------------
+---------------+--------------------+
|         1 | goldenrod lavender spring chocolate lace | Manufacturer#1 | Brand#13 | PROMO BURNISHED COPPER  |      7 | JUMBO PKG
|           901 | ly. slyly ironi     |
|         2 | blush thistle blue yellow saddle         | Manufacturer#1 | Brand#13 | LARGE BRUSHED BRASS     |      1 | LG CASE
|           902 | lar accounts amo    |
|         3 | spring green yellow purple cornsilk      | Manufacturer#4 | Brand#42 | STANDARD POLISHED BRASS |     21 | WRAP CASE
|           903 | egular deposits hag |
|         4 | cornflower chocolate smoke green pink    | Manufacturer#3 | Brand#34 | SMALL PLATED BRASS      |     14 | MED DRUM
|           904 | p furiously r       |
|         5 | forest brown coral puff cream            | Manufacturer#3 | Brand#32 | STANDARD POLISHED TIN   |     15 | SM PKG
|           905 |  wake carefully     |
+-----------+----------------------------------------+---------------+----------+-------------------------+--------+-------------
+---------------+--------------------+
```

avro format

`avro` format: S3 tvf supports parsing the column names and column types of the table schema from the avro file. Example:

```
select * from s3(
         "uri" = "http://127.0.0.1:9312/test2/person.avro",
         "ACCESS_KEY" = "ak",
         "SECRET_KEY" = "sk",
         "FORMAT" = "avro");
+--------+--------------+-------------+-----------------+
| name   | boolean_type | double_type | long_type       |
+--------+--------------+-------------+-----------------+
| Alyssa |            1 |     10.0012 | 100000000221133 |
| Ben    |            0 |    5555.999 |      4009990000 |
| lisi   |            0 | 5992225.999 |      9099933330 |
+--------+--------------+-------------+-----------------+
```

uri contains wildcards

uri can use wildcards to read multiple files. Note: If wildcards are used, the format of each file must be consistent (especially csv/csv_with_names/csv_with_names_and_types count as different formats), S3 tvf uses the first file to parse out the table schema. For example:

The following two csv files:

```
// file1.csv
1,aaa,18
2,qqq,20
3,qwe,19

// file2.csv
5,cyx,19
6,ftw,21
```

You can use wildcards on the uri to query.

```
MySQL [(none)]> select * from s3(
        "URI" = "http://127.0.0.1:9312/test2/file*.csv",
        "s3.access_key"= "minioadmin",
        "s3.secret_key" = "minioadmin",
        "format" = "csv",
        "use_path_style"="true");
+------+------+------+
| c1   | c2   | c3   |
+------+------+------+
| 1    | aaa  | 18   |
| 2    | qqq  | 20   |
| 3    | qwe  | 19   |
| 5    | cyx  | 19   |
| 6    | ftw  | 21   |
+------+------+------+
```

Using `S3` tvf with `insert into` and `cast`

```
// Create doris internal table
CREATE TABLE IF NOT EXISTS ${testTable}
    (
        id int,
        name varchar(50),
        age int
    )
    COMMENT "my first table"
    DISTRIBUTED BY HASH(id) BUCKETS 32
    PROPERTIES("replication_num" = "1");

// Insert data using S3
insert into ${testTable} (id,name,age)
select cast (id as INT) as id, name, cast (age as INT) as age
from s3(
    "uri" = "${uri}",
    "s3.access_key"= "${ak}",
    "s3.secret_key" = "${sk}",
    "format" = "${format}",
    "strip_outer_array" = "true",
    "read_json_by_line" = "true",
    "use_path_style" = "true");
```

HDFS

HDFS

Name

hdfs

Description

HDFS table-valued-function(tvf), allows users to read and access file contents on S3-compatible object storage, just like accessing relational table. Currently supports `csv/csv_with_names/csv_with_names_and_types/json/parquet/orc` file format.

syntax

```
hdfs(
  "uri" = "..",
  "fs.defaultFS" = "...",
  "hadoop.username" = "...",
  "format" = "csv",
  "keyn" = "valuen"
  ...
  );
```

parameter description

Related parameters for accessing hdfs:

- `uri`: (required) hdfs uri. If the uri path does not exist or the files are empty files, hdfs tvf will return an empty result set.
- `fs.defaultFS`: (required)
- `hadoop.username`: (required) Can be any string, but cannot be empty.
- `hadoop.security.authentication`: (optional)
- `hadoop.username`: (optional)
- `hadoop.kerberos.principal`: (optional)
- `hadoop.kerberos.keytab`: (optional)
- `dfs.client.read.shortcircuit`: (optional)
- `dfs.domain.socket.path`: (optional)

Related parameters for accessing HDFS in HA mode:

- `dfs.nameservices`: (optional)
- `dfs.ha.namenodes.your-nameservices`: (optional)
- `dfs.namenode.rpc-address.your-nameservices.your-namenode`: (optional)
- `dfs.client.failover.proxy.provider.your-nameservices`: (optional)

File format parameters:

- `format`: (required) Currently support `csv/csv_with_names/csv_with_names_and_types/json/parquet/orc/avro`
- `column_separator`: (optional) default `\t`.
- `line_delimiter`: (optional) default `\n`.
- `compress_type`: (optional) Currently support `UNKNOWN/PLAIN/GZ/LZO/BZ2/LZ4FRAME/DEFLATE/SNAPPYBLOCK`. Default value is `UNKNOWN`, it will automatically infer the type based on the suffix of `uri`.

  The following 6 parameters are used for loading in json format. For specific usage methods, please refer to: Json Load

- `read_json_by_line`: (optional) default `"true"`
- `strip_outer_array`: (optional) default `"false"`
- `json_root`: (optional) default `""`
- `json_paths`: (optional) default `""`
- `num_as_string`: (optional) default `false`
- `fuzzy_parse`: (optional) default `false`

  The following 2 parameters are used for loading in csv format:

- `trim_double_quotes`: Boolean type (optional), the default value is `false`. True means that the outermost double quotes of each field in the csv file are trimmed.
- `skip_lines`: Integer type (optional), the default value is 0. It will skip some lines in the head of csv file. It will be disabled when the format is `csv_with_names` or `csv_with_names_and_types`.

other kinds of parameters:

- `path_partition_keys`: (optional) Specifies the column names carried in the file path. For example, if the file path is /path/to/city=beijing/date="2023-07-09", you should fill in `path_partition_keys="city,date"`. It will automatically read the corresponding column names and values from the path during load process.
- `resource`:（optional）Specify the resource name. Hdfs Tvf can use the existing Hdfs resource to directly access Hdfs. You can refer to the method for creating an Hdfs resource: CREATE-RESOURCE. This property is supported starting from version 2.1.4.

> **Tip**
>
> To directly query a TVF or create a VIEW based on that TVF, you need to have usage permission for that resource. To query a VIEW created based on TVF, you only need select permission for that VIEW.

Examples

Read and access csv format files on hdfs storage.

```
MySQL [(none)]> select * from hdfs(
            "uri" = "hdfs://127.0.0.1:842/user/doris/csv_format_test/student.csv",
            "fs.defaultFS" = "hdfs://127.0.0.1:8424",
            "hadoop.username" = "doris",
            "format" = "csv");
+------+---------+------+
| c1   | c2      | c3   |
+------+---------+------+
| 1    | alice   | 18   |
| 2    | bob     | 20   |
| 3    | jack    | 24   |
| 4    | jackson | 19   |
| 5    | liming  | 18   |
+------+---------+------+
```

Read and access csv format files on hdfs storage in HA mode.

```
MySQL [(none)]> select * from hdfs(
            "uri" = "hdfs://127.0.0.1:842/user/doris/csv_format_test/student.csv",
            "fs.defaultFS" = "hdfs://127.0.0.1:8424",
            "hadoop.username" = "doris",
            "format" = "csv",
            "dfs.nameservices" = "my_hdfs",
            "dfs.ha.namenodes.my_hdfs" = "nn1,nn2",
            "dfs.namenode.rpc-address.my_hdfs.nn1" = "nanmenode01:8020",
            "dfs.namenode.rpc-address.my_hdfs.nn2" = "nanmenode02:8020",
            "dfs.client.failover.proxy.provider.my_hdfs" =
"org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider");
+------+---------+------+
| c1   | c2      | c3   |
+------+---------+------+
| 1    | alice   | 18   |
| 2    | bob     | 20   |
| 3    | jack    | 24   |
| 4    | jackson | 19   |
| 5    | liming  | 18   |
+------+---------+------+
```

Can be used with `desc function` :

```
MySQL [(none)]> desc function hdfs(
            "uri" = "hdfs://127.0.0.1:8424/user/doris/csv_format_test/student_with_names.csv",
            "fs.defaultFS" = "hdfs://127.0.0.1:8424",
            "hadoop.username" = "doris",
            "format" = "csv_with_names");
```

Keywords

```
hdfs, table-valued-function, tvf
```

Best Practice

For more detailed usage of HDFS tvf, please refer to S3 tvf, The only difference between them is the way of accessing the storage system.

LOCAL

local

Name

local

Description

Local table-valued-function(tvf), allows users to read and access local file contents on be node, just like accessing relational table. Currently supports `csv/csv_with_names/csv_with_names_and_types/json/parquet/orc` file format.

It needs `ADMIN` privilege to use.

syntax

```
local(
  "file_path" = "path/to/file.txt",
  "backend_id" = "be_id",
  "format" = "csv",
  "keyn" = "valuen"
  ...
  );
```

parameter description

- Related parameters for accessing local file on be node:
    - `file_path`:

      (required) The path of the file to be read, which is a relative path to the `user_files_secure_path` directory, where `user_files_secure_path` parameter can be configured on be.

      Can not contains `..` in path. Support using glob syntax to match multi files, such as `log/*.log`
- Related to execution method:

  In versions prior to 2.1.1, Doris only supported specifying a BE node to read local data files on that node.
    - `backend_id`:

      The be id where the file is located. `backend_id` can be obtained through the `show backends` command.

  Starting from version 2.1.2, Doris adds a new parameter `shared_storage`.
    - `shared_storage`

Default is false. If true, the specified file exists on shared storage (such as NAS). Shared storage must be compatible with the POXIS file interface and mounted on all BE nodes at the same time.

When `shared_storage` is true, you do not need to set `backend_id`, Doris may use all BE nodes for data access. If `backend_id` is set, still only executes on the specified BE node.

- File format parameters:
  - `format`: (required) Currently support `csv/csv_with_names/csv_with_names_and_types/json/parquet/orc`
  - `column_separator`: (optional) default `\t`.
  - `line_delimiter`: (optional) default `\n`.
  - `compress_type`: (optional) Currently support `UNKNOWN/PLAIN/GZ/LZO/BZ2/LZ4FRAME/DEFLATE/SNAPPYBLOCK`. Default value is `UNKNOWN`, it will automatically infer the type based on the suffix of `uri`.
- The following parameters are used for loading in json format. For specific usage methods, please refer to: Json Load
  - `read_json_by_line`: (optional) default `"true"`
  - `strip_outer_array`: (optional) default `"false"`
  - `json_root`: (optional) default `""`
  - `json_paths`: (optional) default `""`
  - `num_as_string`: (optional) default `false`
  - `fuzzy_parse`: (optional) default `false`
- The following parameters are used for loading in csv format
  - `trim_double_quotes`: Boolean type (optional), the default value is `false`. True means that the outermost double quotes of each field in the csv file are trimmed.
  - `skip_lines`: Integer type (optional), the default value is 0. It will skip some lines in the head of csv file. It will be disabled when the format is `csv_with_names` or `csv_with_names_and_types`.
- other parameter:
  - `path_partition_keys`: (optional) Specifies the column names carried in the file path. For example, if the file path is /path/to/city=beijing/date="2023-07-09", you should fill in `path_partition_keys="city,date"`. It will automatically read the corresponding column names and values from the path during load process.

Examples

Analyze the log file on specified BE:

```
mysql> select * from local(
        "file_path" = "log/be.out",
        "backend_id" = "10006",
        "format" = "csv")
       where c1 like "%start_time%" limit 10;
+-----------------------------------------------------+
| c1                                                  |
+-----------------------------------------------------+
| start time: 2023年 08月 07日 星期一 23:20:32 CST       |
| start time: 2023年 08月 07日 星期一 23:32:10 CST       |
| start time: 2023年 08月 08日 星期二 00:20:50 CST       |
```

```
| start time: 2023年 08月 08日 星期二 00:29:15 CST       |
+-----------------------------------------------------+
```

Read and access csv format files located at path `${DORIS_HOME}/student.csv`:

```
mysql> select * from local(
      "file_path" = "student.csv",
      "backend_id" = "10003",
      "format" = "csv");
+------+---------+--------+
| c1   | c2      | c3     |
+------+---------+--------+
| 1    | alice   | 18     |
| 2    | bob     | 20     |
| 3    | jack    | 24     |
| 4    | jackson | 19     |
| 5    | liming  | d18    |
+------+---------+--------+
```

Query files on NAS:

```
mysql> select * from local(
        "file_path" = "/mnt/doris/prefix_*.txt",
        "format" = "csv",
        "column_separator" =",",
        "shared_storage" = "true");
+------+------+------+
| c1   | c2   | c3   |
+------+------+------+
| 1    | 2    | 3    |
| 1    | 2    | 3    |
| 1    | 2    | 3    |
| 1    | 2    | 3    |
| 1    | 2    | 3    |
+------+------+------+
```

Can be used with `desc function` :

```
mysql> desc function local(
      "file_path" = "student.csv",
      "backend_id" = "10003",
      "format" = "csv");
+-------+------+------+-------+---------+-------+
| Field | Type | Null | Key   | Default | Extra |
+-------+------+------+-------+---------+-------+
| c1    | TEXT | Yes  | false | NULL    | NONE  |
| c2    | TEXT | Yes  | false | NULL    | NONE  |
| c3    | TEXT | Yes  | false | NULL    | NONE  |
+-------+------+------+-------+---------+-------+
```

Keywords

local, table-valued-function, tvf

Best Practice

- For more detailed usage of local tvf, please refer to S3 tvf, The only difference between them is the way of accessing the storage system.
- Access data on NAS through local tvf

  NAS shared storage allows to be mounted to multiple nodes at the same time. Each node can access files in the shared storage just like local files. Therefore, the NAS can be thought of as a local file system, accessed through local tvf.

  When setting `"shared_storage" = "true"`, Doris will think that the specified file can be accessed from any BE node. When a set of files is specified using wildcards, Doris will distribute requests to access files to multiple BE nodes, so that multiple nodes can be used to perform distributed file scanning and improve query performance.

ICEBERG_META

iceberg_meta

Name

iceberg_meta

description

iceberg_meta table-valued-function(tvf), Use for read iceberg metadata, operation history, snapshots of table, file metadata etc.

syntax

```
iceberg_meta(
  "table" = "ctl.db.tbl",
  "query_type" = "snapshots"
  ...
  );
```

parameter description

Each parameter in iceberg_meta tvf is a pair of `"key"="value"`.

Related parameters: - `table`： (required) Use iceberg table name the format `catlog.database.table`. - `query_type`： (required) The type of iceberg metadata. Only `snapshots` is currently supported.

Example

Read and access the iceberg tabular metadata for snapshots.

```
select * from iceberg_meta("table" = "ctl.db.tbl", "query_type" = "snapshots");
```

Can be used with `desc function` :

```
desc function iceberg_meta("table" = "ctl.db.tbl", "query_type" = "snapshots");
```

Keywords

```
iceberg_meta, table-valued-function, tvf
```

Best Prac

Inspect the iceberg table snapshots :

```
select * from iceberg_meta("table" = "iceberg_ctl.test_db.test_tbl", "query_type" = "snapshots");
+-----------------------+---------------+--------------+-----------+------------------+----------------------------+
|      committed_at     |  snapshot_id  |   parent_id  | operation |   manifest_list  |            summary         |
+-----------------------+---------------+--------------+-----------+------------------+----------------------------+
|  2022-09-20 11:14:29  |  64123452344  |       -1     |  append   | hdfs:/path/to/m1 | {"flink.job-id":"xxm1", ...} |
|  2022-09-21 10:36:35  |  98865735822  |  64123452344 | overwrite | hdfs:/path/to/m2 | {"flink.job-id":"xxm2", ...} |
|  2022-09-21 21:44:11  |  51232845315  |  98865735822 | overwrite | hdfs:/path/to/m3 | {"flink.job-id":"xxm3", ...} |
+-----------------------+---------------+--------------+-----------+------------------+----------------------------+
```

Filtered by snapshot_id :

```
select * from iceberg_meta("table" = "iceberg_ctl.test_db.test_tbl", "query_type" = "snapshots")
where snapshot_id = 98865735822;
+-----------------------+---------------+--------------+-----------+------------------+----------------------------+
|      committed_at     |  snapshot_id  |   parent_id  | operation |   manifest_list  |            summary         |
+-----------------------+---------------+--------------+-----------+------------------+----------------------------+
|  2022-09-21 10:36:35  |  98865735822  |  64123452344 | overwrite | hdfs:/path/to/m2 | {"flink.job-id":"xxm2", ...} |
+-----------------------+---------------+--------------+-----------+------------------+----------------------------+
```

BACKENDS

backends

Name

backends

description

Table-Value-Function, generate a temporary table named `backends`. This tvf is used to view the information of BE nodes in the doris cluster.

This function is used in `FROM` clauses.

syntax

`backends()`

The table schema of `backends()` tvf:

```
mysql> desc function backends();
+-------------------------+---------+------+-------+---------+-------+
| Field                   | Type    | Null | Key   | Default | Extra |
+-------------------------+---------+------+-------+---------+-------+
| BackendId               | BIGINT  | No   | false | NULL    | NONE  |
| Host                    | TEXT    | No   | false | NULL    | NONE  |
| HeartbeatPort           | INT     | No   | false | NULL    | NONE  |
| BePort                  | INT     | No   | false | NULL    | NONE  |
| HttpPort                | INT     | No   | false | NULL    | NONE  |
| BrpcPort                | INT     | No   | false | NULL    | NONE  |
| LastStartTime           | TEXT    | No   | false | NULL    | NONE  |
| LastHeartbeat           | TEXT    | No   | false | NULL    | NONE  |
| Alive                   | BOOLEAN | No   | false | NULL    | NONE  |
| SystemDecommissioned    | BOOLEAN | No   | false | NULL    | NONE  |
| TabletNum               | BIGINT  | No   | false | NULL    | NONE  |
| DataUsedCapacity        | BIGINT  | No   | false | NULL    | NONE  |
| AvailCapacity           | BIGINT  | No   | false | NULL    | NONE  |
| TotalCapacity           | BIGINT  | No   | false | NULL    | NONE  |
| UsedPct                 | DOUBLE  | No   | false | NULL    | NONE  |
| MaxDiskUsedPct          | DOUBLE  | No   | false | NULL    | NONE  |
| RemoteUsedCapacity      | BIGINT  | No   | false | NULL    | NONE  |
| Tag                     | TEXT    | No   | false | NULL    | NONE  |
| ErrMsg                  | TEXT    | No   | false | NULL    | NONE  |
| Version                 | TEXT    | No   | false | NULL    | NONE  |
| Status                  | TEXT    | No   | false | NULL    | NONE  |
| HeartbeatFailureCounter | INT     | No   | false | NULL    | NONE  |
| NodeRole                | TEXT    | No   | false | NULL    | NONE  |
+-------------------------+---------+------+-------+---------+-------+
23 rows in set (0.002 sec)
```

The information displayed by the `backends` tvf is basically consistent with the information displayed by the `show backends` statement. However, the types of each field in the `backends` tvf are more specific, and you can use the

backends tvf to perform operations such as filtering and joining.

The information displayed by the backends tvf is authenticated, which is consistent with the behavior of show backends, user must have ADMIN/OPERATOR privelege.

example

```
mysql> select * from backends()\G
*************************** 1. row ***************************
              BackendId: 10002
                   Host: 10.xx.xx.90
          HeartbeatPort: 9053
                 BePort: 9063
               HttpPort: 8043
               BrpcPort: 8069
          LastStartTime: 2023-06-15 16:51:02
          LastHeartbeat: 2023-06-15 17:09:58
                  Alive: 1
   SystemDecommissioned: 0
             TabletNum: 21
       DataUsedCapacity: 0
          AvailCapacity: 5187141550081
          TotalCapacity: 7750977622016
                UsedPct: 33.077583202570978
        MaxDiskUsedPct: 33.077583202583881
     RemoteUsedCapacity: 0
                    Tag: {"location" : "default"}
                 ErrMsg:
                Version: doris-0.0.0-trunk-4b18cde0c7
                 Status: {"lastSuccessReportTabletsTime":"2023-06-15
17:09:02","lastStreamLoadTime":-1,"isQueryDisabled":false,"isLoadDisabled":false}
HeartbeatFailureCounter: 0
               NodeRole: mix
1 row in set (0.038 sec)
```

keywords

```
backends
```

FRONTENDS

`frontends`

Name

frontends

description

Table-Value-Function, generate a temporary table named `frontends`. This tvf is used to view the information of BE nodes in the doris cluster.

This function is used in `FROM` clauses.

syntax

`frontends()`

The table schema of `frontends()` tvf:

```
mysql> desc function frontends();
+-------------------+------+------+-------+---------+-------+
| Field             | Type | Null | Key   | Default | Extra |
+-------------------+------+------+-------+---------+-------+
| Name              | TEXT | No   | false | NULL    | NONE  |
| Host              | TEXT | No   | false | NULL    | NONE  |
| EditLogPort       | TEXT | No   | false | NULL    | NONE  |
| HttpPort          | TEXT | No   | false | NULL    | NONE  |
| QueryPort         | TEXT | No   | false | NULL    | NONE  |
| RpcPort           | TEXT | No   | false | NULL    | NONE  |
| ArrowFlightSqlPort| TEXT | No   | false | NULL    | NONE  |
| Role              | TEXT | No   | false | NULL    | NONE  |
| IsMaster          | TEXT | No   | false | NULL    | NONE  |
| ClusterId         | TEXT | No   | false | NULL    | NONE  |
| Join              | TEXT | No   | false | NULL    | NONE  |
| Alive             | TEXT | No   | false | NULL    | NONE  |
| ReplayedJournalId | TEXT | No   | false | NULL    | NONE  |
| LastHeartbeat     | TEXT | No   | false | NULL    | NONE  |
| IsHelper          | TEXT | No   | false | NULL    | NONE  |
| ErrMsg            | TEXT | No   | false | NULL    | NONE  |
| Version           | TEXT | No   | false | NULL    | NONE  |
| CurrentConnected  | TEXT | No   | false | NULL    | NONE  |
+-------------------+------+------+-------+---------+-------+
17 rows in set (0.022 sec)
```

The information displayed by the `frontends` tvf is basically consistent with the information displayed by the `show frontends` statement. However, the types of each field in the `frontends` tvf are more specific, and you can use the `frontends` tvf to perform operations such as filtering and joining.

The information displayed by the `frontends` tvf is authenticated, which is consistent with the behavior of `show frontends`, user must have ADMIN/OPERATOR privelege.

example

```
mysql> select * from frontends()\G
*************************** 1. row ***************************
             Name: fe_5fa8bf19_fd6b_45cb_89c5_25a5ebc45582
               IP: 10.xx.xx.14
      EditLogPort: 9013
         HttpPort: 8034
        QueryPort: 9033
          RpcPort: 9023
ArrowFlightSqlPort: 9040
             Role: FOLLOWER
         IsMaster: true
        ClusterId: 1258341841
             Join: true
            Alive: true
ReplayedJournalId: 186
    LastHeartbeat: 2023-06-15 16:53:12
         IsHelper: true
           ErrMsg:
          Version: doris-0.0.0-trunk-4b18cde0c7
 CurrentConnected: Yes
1 row in set (0.060 sec)
```

keywords

```
frontends
```

FRONTENDS_DISKS

`frontends`

Name

frontends

description

Table-Value-Function, generate a temporary table named `frontends_disks`. This tvf is used to view the information of FE nodes 's disks in the doris cluster.

This function is used in `FROM` clauses.

syntax

`frontends_disks()`

The table schema of `frontends_disks()` tvf:

```
mysql> desc function frontends_disks();
+-------------+------+------+-------+---------+-------+
| Field       | Type | Null | Key   | Default | Extra |
+-------------+------+------+-------+---------+-------+
| Name        | TEXT | No   | false | NULL    | NONE  |
| Host        | TEXT | No   | false | NULL    | NONE  |
| DirType     | TEXT | No   | false | NULL    | NONE  |
| Dir         | TEXT | No   | false | NULL    | NONE  |
| Filesystem  | TEXT | No   | false | NULL    | NONE  |
| Capacity    | TEXT | No   | false | NULL    | NONE  |
| Used        | TEXT | No   | false | NULL    | NONE  |
| Available   | TEXT | No   | false | NULL    | NONE  |
| UseRate     | TEXT | No   | false | NULL    | NONE  |
| MountOn     | TEXT | No   | false | NULL    | NONE  |
+-------------+------+------+-------+---------+-------+
11 rows in set (0.14 sec)
```

The information displayed by the `frontends_disks` tvf is basically consistent with the information displayed by the `show frontends disks` statement. However, the types of each field in the `frontends_disks` tvf are more specific, and you can use the `frontends_disks` tvf to perform operations such as filtering and joining.

The information displayed by the `frontends_disks` tvf is authenticated, which is consistent with the behavior of `show frontends disks`, user must have ADMIN/OPERATOR privelege.

example

```
mysql> select * from frontends_disk()\G
*************************** 1. row ***************************
       Name: fe_fe1d5bd9_d1e5_4ccc_9b03_ca79b95c9941
```

```
       Host: 172.XX.XX.1
    DirType: log
        Dir: /data/doris/fe-github/log
 Filesystem: /dev/sdc5
   Capacity: 366G
       Used: 119G
  Available: 228G
    UseRate: 35%
    MountOn: /data
......
12 row in set (0.03 sec)
```

keywords

```
frontends_disks
```

CATALOGS

`catalogs`

Name

catalogs

description

The table function generates a temporary table of catalogs to view the information of the catalogs created in the current Doris.

This function is used in the from clause.

syntax

`catalogs()`

Catalogs () table structure:

```
mysql> desc function catalogs();
+-------------+--------+------+-------+---------+-------+
| Field       | Type   | Null | Key   | Default | Extra |
+-------------+--------+------+-------+---------+-------+
| CatalogId   | BIGINT | No   | false | NULL    | NONE  |
| CatalogName | TEXT   | No   | false | NULL    | NONE  |
| CatalogType | TEXT   | No   | false | NULL    | NONE  |
| Property    | TEXT   | No   | false | NULL    | NONE  |
| Value       | TEXT   | No   | false | NULL    | NONE  |
+-------------+--------+------+-------+---------+-------+
5 rows in set (0.04 sec)
```

The information presented by `catalogs()` tvf is the result of synthesizing `show catalogs` and `show catalog xxx` statements.

The table generated by tvf can be used for filtering, join and other operations.

example

```
mysql> select * from catalogs();
+-----------+-------------+-------------+-------------------------------------------+-------------------------------------------
----------------------------+
| CatalogId | CatalogName | CatalogType | Property                                  | Value
|
+-----------+-------------+-------------+-------------------------------------------+-------------------------------------------
----------------------------+
|     16725 | hive        | hms         | dfs.client.failover.proxy.provider.HANN   |
org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider |
|     16725 | hive        | hms         | dfs.ha.namenodes.HANN                     | nn1,nn2
|
```

```
|     16725 | hive        | hms         | create_time                                | 2023-07-13 16:24:38.968
|
|     16725 | hive        | hms         | ipc.client.fallback-to-simple-auth-allowed | true
|
|     16725 | hive        | hms         | dfs.namenode.rpc-address.HANN.nn1          | nn1_host:rpc_port
|
|     16725 | hive        | hms         | hive.metastore.uris                        | thrift://127.0.0.1:7004
|
|     16725 | hive        | hms         | dfs.namenode.rpc-address.HANN.nn2          | nn2_host:rpc_port
|
|     16725 | hive        | hms         | type                                       | hms
|
|     16725 | hive        | hms         | dfs.nameservices                           | HANN
|
|         0 | internal    | internal    | NULL                                       | NULL
|
|     16726 | es          | es          | create_time                                | 2023-07-13 16:24:44.922
|
|     16726 | es          | es          | type                                       | es
|
|     16726 | es          | es          | hosts                                      | http://127.0.0.1:9200
|
+-----------+-------------+-------------+--------------------------------------------+-----------------------------------------------
------------------------------+
13 rows in set (0.01 sec)
```

keywords

```
catalogs
```

WORKLOAD_GROUPS

`workload_groups`

Name

workload_groups

> **Deprecated. Since 2.1.1, this table function has been moved to the information_schema.workload_groups.**

description

Table-Value-Function, generate a temporary table named `workload_groups`. This tvf is used to view information about workload groups for which current user has permission.

This function is used in `FROM` clauses.

syntax

`workload_groups()`

The table schema of `workload_groups()` tvf:

```
mysql> desc function workload_groups();
+-------+-------------+------+-------+---------+-------+
| Field | Type        | Null | Key   | Default | Extra |
+-------+-------------+------+-------+---------+-------+
| Id    | BIGINT      | No   | false | NULL    | NONE  |
| Name  | STRING      | No   | false | NULL    | NONE  |
| Item  | STRING      | No   | false | NULL    | NONE  |
| Value | STRING      | No   | false | NULL    | NONE  |
+-------+-------------+------+-------+---------+-------+
```

example

```
mysql> select * from workload_groups()\G
+-------+--------+--------------+-------+
| Id    | Name   | Item         | Value |
+-------+--------+--------------+-------+
| 11001 | normal | memory_limit | 100%  |
| 11001 | normal | cpu_share    | 10    |
+-------+--------+--------------+-------+
```

keywords

```
workload_groups
```

WINDOW FUNCTIONS OVERVIEW

description

Analytical functions(windown function) are a special class of built-in functions. Similar to aggregate functions, analytic functions also perform calculations on multiple input rows to obtain a data value. The difference is that the analytic function processes the input data within a specific window, rather than grouping calculations by group by. The data within each window can be sorted and grouped using the over() clause. The analytic function computes a single value for each row of the result set, rather than one value per group by grouping. This flexible approach allows the user to add additional columns to the select clause, giving the user more opportunities to reorganize and filter the result set. Analytic functions can only appear in select lists and in the outermost order by clause. During the query process, the analytical function will take effect at the end, that is, after the join, where and group by operations are performed. Analytical functions are often used in financial and scientific computing to analyze trends, calculate outliers, and perform bucket analysis on large amounts of data.

The syntax of the analytic function:

```
function(args) OVER(partition_by_clause order_by_clause [window_clause])
partition_by_clause ::= PARTITION BY expr [, expr ...]
order_by_clause ::= ORDER BY expr [ASC | DESC] [, expr [ASC | DESC] ...]
```

Function

Support Functions: AVG(), COUNT(), DENSE_RANK(), FIRST_VALUE(), LAG(), LAST_VALUE(), LEAD(), MAX(), MIN(), RANK(), ROW_NUMBER(), SUM() and all aggregate functions

PARTITION BY clause

The Partition By clause is similar to Group By. It groups the input rows according to the specified column or columns, and rows with the same value will be grouped together.

ORDER BY clause

The Order By clause is basically the same as the outer Order By. It defines the order in which the input rows are sorted, and if Partition By is specified, Order By defines the order within each Partition grouping. The only difference from the outer Order By is that the Order By n (n is a positive integer) in the OVER clause is equivalent to doing nothing, while the outer Order By n means sorting according to the nth column.

Example:

This example shows adding an id column to the select list with values 1, 2, 3, etc., sorted by the date_and_time column in the events table.

```
SELECT
row_number() OVER (ORDER BY date_and_time) AS id,
```

```
c1, c2, c3, c4
FROM events;
```

Window clause

The Window clause is used to specify an operation range for the analytical function, the current row is the criterion, and several rows before and after are used as the object of the analytical function operation. The methods supported by the Window clause are: AVG(), COUNT(), FIRST_VALUE(), LAST_VALUE() and SUM(). For MAX() and MIN(), the window clause can specify the starting range UNBOUNDED PRECEDING

syntax:

```
ROWS BETWEEN [ { m | UNBOUNDED } PRECEDING | CURRENT ROW] [ AND [CURRENT ROW | { UNBOUNDED | n } FOLLOWING] ]
```

Example

Suppose we have the following stock data, the stock symbol is JDR, and the closing price is the closing price of each day.

```
create table stock_ticker (stock_symbol string, closing_price decimal(8,2), closing_date datetime);
...load some data...
select * from stock_ticker order by stock_symbol, closing_date
 | stock_symbol | closing_price | closing_date        |
 |--------------|---------------|---------------------|
 | JDR          | 12.86         | 2014-10-02 00:00:00 |
 | JDR          | 12.89         | 2014-10-03 00:00:00 |
 | JDR          | 12.94         | 2014-10-04 00:00:00 |
 | JDR          | 12.55         | 2014-10-05 00:00:00 |
 | JDR          | 14.03         | 2014-10-06 00:00:00 |
 | JDR          | 14.75         | 2014-10-07 00:00:00 |
 | JDR          | 13.98         | 2014-10-08 00:00:00 |
```

This query uses the analytic function to generate the column moving_average, whose value is the 3-day average price of the stock, that is, the three-day average price of the previous day, the current day, and the next day. The first day has no value for the previous day, and the last day does not have the value for the next day, so these two lines only calculate the average of the two days. Partition By does not play a role here, because all the data are JDR data, but if there is other stock information, Partition By will ensure that the analysis function value acts within this Partition.

```
select stock_symbol, closing_date, closing_price,
avg(closing_price) over (partition by stock_symbol order by closing_date
rows between 1 preceding and 1 following) as moving_average
from stock_ticker;
 | stock_symbol | closing_date        | closing_price | moving_average |
 |--------------|---------------------|---------------|----------------|
 | JDR          | 2014-10-02 00:00:00 | 12.86         | 12.87          |
 | JDR          | 2014-10-03 00:00:00 | 12.89         | 12.89          |
 | JDR          | 2014-10-04 00:00:00 | 12.94         | 12.79          |
 | JDR          | 2014-10-05 00:00:00 | 12.55         | 13.17          |
 | JDR          | 2014-10-06 00:00:00 | 14.03         | 13.77          |
 | JDR          | 2014-10-07 00:00:00 | 14.75         | 14.25          |
 | JDR          | 2014-10-08 00:00:00 | 13.98         | 14.36          |
```

Keywords

WINDOW, FUNCTION

WINDOW_FUNCTION_SUM

WINDOW FUNCTION SUM

description

Calculate the sum of the data in the window

```
SUM([ALL] expression) [OVER (analytic_clause)]
```

example

Group by property, and calculate the sum of the x columns of the current row and the previous row within the group.

```
select x, property,
sum(x) over
(
partition by property
order by x
rows between 1 preceding and 1 following
) as 'moving total'
from int_t where property in ('odd','even');

| x  | property | moving total |
|----|----------|--------------|
| 2  | even     | 6            |
| 4  | even     | 12           |
| 6  | even     | 18           |
| 8  | even     | 24           |
| 10 | even     | 18           |
| 1  | odd      | 4            |
| 3  | odd      | 9            |
| 5  | odd      | 15           |
| 7  | odd      | 21           |
| 9  | odd      | 16           |
```

keywords

```
WINDOW,FUNCTION,SUM
```

WINDOW_FUNCTION_AVG

WINDOW FUNCTION AVG

description

Calculate the mean of the data within the window

```
AVG([ALL] *expression*) [OVER (*analytic_clause*)]
```

example

Calculate the x-average of the current row and the rows before and after it

```
select x, property,
avg(x) over
(
partition by property
order by x
rows between 1 preceding and 1 following
) as 'moving average'
from int_t where property in ('odd','even');

 | x  | property | moving average |
 |----|----------|----------------|
 | 2  | even     | 3              |
 | 4  | even     | 4              |
 | 6  | even     | 6              |
 | 8  | even     | 8              |
 | 10 | even     | 9              |
 | 1  | odd      | 2              |
 | 3  | odd      | 3              |
 | 5  | odd      | 5              |
 | 7  | odd      | 7              |
 | 9  | odd      | 8              |
```

keywords

```
WINDOW,FUNCTION,AVG
```

WINDOW_FUNCTION_MAX

WINDOW FUNCTION MAX

description

The LEAD() method is used to calculate the maximum value within the window.

```
MAX([DISTINCT | ALL] expression) [OVER (analytic_clause)]
```

example

Calculate the maximum value from the first row to the row after the current row

```
select x, property,
max(x) over
(
order by property, x
rows between unbounded preceding and 1 following
) as 'local maximum'
from int_t where property in ('prime','square');

| x | property | local maximum |
|---|----------|---------------|
| 2 | prime    | 3             |
| 3 | prime    | 5             |
| 5 | prime    | 7             |
| 7 | prime    | 7             |
| 1 | square   | 7             |
| 4 | square   | 9             |
| 9 | square   | 9             |
```

keywords

```
WINDOW,FUNCTION,MAX
```

WINDOW_FUNCTION_MIN

WINDOW FUNCTION MIN

description

The LEAD() method is used to calculate the minimum value within the window.

```
MAX([DISTINCT | ALL] expression) [OVER (analytic_clause)]
```

example

Calculate the minimum value from the first row to the row after the current row

```
select x, property,
min(x) over
(
order by property, x desc
rows between unbounded preceding and 1 following
) as 'local minimum'
from int_t where property in ('prime','square');
| x | property | local minimum |
|---|----------|---------------|
| 7 | prime    | 5             |
| 5 | prime    | 3             |
| 3 | prime    | 2             |
| 2 | prime    | 2             |
| 9 | square   | 2             |
| 4 | square   | 1             |
| 1 | square   | 1             |
```

keywords

```
WINDOW,FUNCTION,MIN
```

WINDOW_FUNCTION_COUNT

WINDOW FUNCTION COUNT

description

Count the number of occurrences of data in the window

```
COUNT(expression) [OVER (analytic_clause)]
```

example

Count the number of occurrences of x from the current row to the first row.

```
select x, property,
count(x) over
(
partition by property
order by x
rows between unbounded preceding and current row
) as 'cumulative total'
from int_t where property in ('odd','even');

 | x  | property | cumulative count |
 |----|----------|------------------|
 | 2  | even     | 1                |
 | 4  | even     | 2                |
 | 6  | even     | 3                |
 | 8  | even     | 4                |
 | 10 | even     | 5                |
 | 1  | odd      | 1                |
 | 3  | odd      | 2                |
 | 5  | odd      | 3                |
 | 7  | odd      | 4                |
 | 9  | odd      | 5                |
```

keywords

```
WINDOW,FUNCTION,COUNT
```

WINDOW_FUNCTION_RANK

WINDOW FUNCTION RANK

description

The RANK() function is used to represent rankings. Unlike DENSE_RANK(), RANK() will have vacancies. For example, if there are two 1s in a row, the third number in RANK() is 3, not 2.

```
RANK() OVER(partition_by_clause order_by_clause)
```

example

rank by x

```
select x, y, rank() over(partition by x order by y) as rank from int_t;

| x  | y    | rank     |
|----|------|----------|
| 1  | 1    | 1        |
| 1  | 2    | 2        |
| 1  | 2    | 2        |
| 2  | 1    | 1        |
| 2  | 2    | 2        |
| 2  | 3    | 3        |
| 3  | 1    | 1        |
| 3  | 1    | 1        |
| 3  | 2    | 3        |
```

keywords

```
WINDOW,FUNCTION,RANK
```

WINDOW_FUNCTION_DENSE_RANK

WINDOW FUNCTION DENSE_RANK

description

The DENSE_RANK() function is used to represent rankings. Unlike RANK(), DENSE_RANK() does not have vacancies. For example, if there are two parallel 1s, the third number of DENSE_RANK() is still 2, and the third number of RANK() is 3.

```
DENSE_RANK() OVER(partition_by_clause order_by_clause)
```

example

Group by the property column to rank column x:

```
select x, y, dense_rank() over(partition by x order by y) as rank from int_t;

| x  | y    | rank     |
|----|------|----------|
| 1  | 1    | 1        |
| 1  | 2    | 2        |
| 1  | 2    | 2        |
| 2  | 1    | 1        |
| 2  | 2    | 2        |
| 2  | 3    | 3        |
| 3  | 1    | 1        |
| 3  | 1    | 1        |
| 3  | 2    | 2        |
```

keywords

```
WINDOW,FUNCTION,DENSE_RANK
```

WINDOW_FUNCTION_FIRST_VALUE

WINDOW FUNCTION FIRST_VALUE

description

> **Note**
>
> Since 2.0.9 , the ignore_null usage is supported

FIRST_VALUE() returns the first value in the window's range , ignore_null determines whether to ignore null values , the ignore_null of default value is false .

```
FIRST_VALUE(expr[, ignore_null]) OVER(partition_by_clause order_by_clause [window_clause])
```

example

We have the following data

```
select id, myday, time_col, state from t;

| id   | myday | time_col    | state |
|------|-------|-------------|-------|
|    3 |    21 | 04-21-13    |     3 |
|    7 |    22 | 04-22-10-24 |  NULL |
|    8 |    22 | 04-22-10-25 |     9 |
|   10 |    23 | 04-23-12    |    10 |
|    4 |    22 | 04-22-10-21 |  NULL |
|    9 |    23 | 04-23-11    |  NULL |
|    1 |    21 | 04-21-11    |  NULL |
|    5 |    22 | 04-22-10-22 |  NULL |
|   12 |    24 | 02-24-10-21 |  NULL |
|    2 |    21 | 04-21-12    |     2 |
|    6 |    22 | 04-22-10-23 |     5 |
|   11 |    23 | 04-23-13    |  NULL |
```

Use FIRST_VALUE() to group by myday and return the value of the first state in each group:

```
select * ,
first_value(`state`, 1) over(partition by `myday` order by `time_col` rows between 1 preceding and 1 following) as ignore_null,
first_value(`state`, 0) over(partition by `myday` order by `time_col` rows between 1 preceding and 1 following) as not_ignore_null,
first_value(`state`) over(partition by `myday` order by `time_col` rows between 1 preceding and 1 following) as ignore_null_default
from t order by `id`, `myday`, `time_col`;

| id   | myday | time_col    | state | ignore_null | not_ignore_null | ignore_null_default |
|------|-------|-------------|-------|-------------|-----------------|---------------------|
|    1 |    21 | 04-21-11    |  NULL |           2 |            NULL |                NULL |
|    2 |    21 | 04-21-12    |     2 |           2 |            NULL |                NULL |
|    3 |    21 | 04-21-13    |     3 |           2 |               2 |                   2 |
|    4 |    22 | 04-22-10-21 |  NULL |        NULL |            NULL |                NULL |
```

```
|    5 |    22 | 04-22-10-22 |  NULL |           5 |           NULL |               NULL |
|    6 |    22 | 04-22-10-23 |     5 |           5 |           NULL |               NULL |
|    7 |    22 | 04-22-10-24 |  NULL |           5 |              5 |                  5 |
|    8 |    22 | 04-22-10-25 |     9 |           9 |           NULL |               NULL |
|    9 |    23 | 04-23-11    |  NULL |          10 |           NULL |               NULL |
|   10 |    23 | 04-23-12    |    10 |          10 |           NULL |               NULL |
|   11 |    23 | 04-23-13    |  NULL |          10 |             10 |                 10 |
|   12 |    24 | 02-24-10-21 |  NULL |        NULL |           NULL |               NULL |
```

keywords

```
WINDOW,FUNCTION,FIRST_VALUE
```

WINDOW_FUNCTION_LAST_VALUE

WINDOW FUNCTION LAST_VALUE

description

> **Note**
>
> Since 2.0.9 , the ignore_null usage is supported

LAST_VALUE() returns the last value in the window range . Opposite of FIRST_VALUE() . ignore_null determines whether to ignore null values , the ignore_null of default value is false .

```
LAST_VALUE(expr[, ignore_null]) OVER(partition_by_clause order_by_clause [window_clause])
```

example

Using the data from the FIRST_VALUE() example:

```
select * ,
last_value(`state`, 1) over(partition by `myday` order by `time_col` DESC rows between 1 preceding and 1 following) as ignore_null,
last_value(`state`, 0) over(partition by `myday` order by `time_col` DESC rows between 1 preceding and 1 following) as
not_ignore_null,
last_value(`state`) over(partition by `myday` order by `time_col` DESC rows between 1 preceding and 1 following) as
ignore_null_default
from t order by `id`, `myday`, `time_col`;

| id   | myday | time_col    | state | ignore_null | not_ignore_null | ignore_null_default |
|------|-------|-------------|-------|-------------|-----------------|---------------------|
|    1 |    21 | 04-21-11    |  NULL |           2 |            NULL |                NULL |
|    2 |    21 | 04-21-12    |     2 |           2 |            NULL |                NULL |
|    3 |    21 | 04-21-13    |     3 |           2 |               2 |                   2 |
|    4 |    22 | 04-22-10-21 |  NULL |        NULL |            NULL |                NULL |
|    5 |    22 | 04-22-10-22 |  NULL |           5 |            NULL |                NULL |
|    6 |    22 | 04-22-10-23 |     5 |           5 |            NULL |                NULL |
|    7 |    22 | 04-22-10-24 |  NULL |           5 |               5 |                   5 |
|    8 |    22 | 04-22-10-25 |     9 |           9 |            NULL |                NULL |
|    9 |    23 | 04-23-11    |  NULL |          10 |            NULL |                NULL |
|   10 |    23 | 04-23-12    |    10 |          10 |            NULL |                NULL |
|   11 |    23 | 04-23-13    |  NULL |          10 |              10 |                  10 |
|   12 |    24 | 02-24-10-21 |  NULL |        NULL |            NULL |                NULL |
```

keywords

```
WINDOW,FUNCTION,LAST_VALUE
```

WINDOW_FUNCTION_LEAD

WINDOW FUNCTION LEAD

description

The LEAD() method is used to calculate the value of the current line several lines backwards.

```
LEAD(expr, offset, default) OVER (partition_by_clause order_by_clause)
```

example

Calculate the trend of the closing price of the second day compared with the closing price of the day, that is, the closing price of the second day is higher or lower than that of the day.

```
select stock_symbol, closing_date, closing_price,
case
(lead(closing_price,1, 0)
over (partition by stock_symbol order by closing_date)-closing_price) > 0
when true then "higher"
when false then "flat or lower"
end as "trending"
from stock_ticker
order by closing_date;

| stock_symbol | closing_date        | closing_price | trending      |
|--------------|---------------------|---------------|---------------|
| JDR          | 2014-09-13 00:00:00 | 12.86         | higher        |
| JDR          | 2014-09-14 00:00:00 | 12.89         | higher        |
| JDR          | 2014-09-15 00:00:00 | 12.94         | flat or lower |
| JDR          | 2014-09-16 00:00:00 | 12.55         | higher        |
| JDR          | 2014-09-17 00:00:00 | 14.03         | higher        |
| JDR          | 2014-09-18 00:00:00 | 14.75         | flat or lower |
| JDR          | 2014-09-19 00:00:00 | 13.98         | flat or lower |
```

keywords

```
WINDOW,FUNCTION,LEAD
```

WINDOW_FUNCTION_LAG

WINDOW FUNCTION LAG

description

The LAG() method is used to calculate the value of the current line several lines ahead.

```
LAG(expr, offset, default) OVER (partition_by_clause order_by_clause)
```

example

Calculate the previous day's closing price

```
select stock_symbol, closing_date, closing_price,
lag(closing_price,1, 0) over (partition by stock_symbol order by closing_date) as "yesterday closing"
from stock_ticker
order by closing_date;

| stock_symbol | closing_date        | closing_price | yesterday closing |
|--------------|---------------------|---------------|-------------------|
| JDR          | 2014-09-13 00:00:00 | 12.86         | 0                 |
| JDR          | 2014-09-14 00:00:00 | 12.89         | 12.86             |
| JDR          | 2014-09-15 00:00:00 | 12.94         | 12.89             |
| JDR          | 2014-09-16 00:00:00 | 12.55         | 12.94             |
| JDR          | 2014-09-17 00:00:00 | 14.03         | 12.55             |
| JDR          | 2014-09-18 00:00:00 | 14.75         | 14.03             |
| JDR          | 2014-09-19 00:00:00 | 13.98         | 14.75             |
```

keywords

```
WINDOW,FUNCTION,LAG
```

WINDOW_FUNCTION_ROW_NUMBER

WINDOW FUNCTION ROW_NUMBER

description

Returns a continuously increasing integer starting from 1 for each row of each Partition. Unlike RANK() and DENSE_RANK(), the value returned by ROW_NUMBER() does not repeat or appear vacant, and is continuously incremented.

```
ROW_NUMBER() OVER(partition_by_clause order_by_clause)
```

example

```
select x, y, row_number() over(partition by x order by y) as rank from int_t;

| x | y    | rank     |
|---|------|----------|
| 1 | 1    | 1        |
| 1 | 2    | 2        |
| 1 | 2    | 3        |
| 2 | 1    | 1        |
| 2 | 2    | 2        |
| 2 | 3    | 3        |
| 3 | 1    | 1        |
| 3 | 1    | 2        |
| 3 | 2    | 3        |
```

keywords

```
WINDOW,FUNCTION,ROW_NUMBER
```

WINDOW_FUNCTION_NTILE

WINDOW FUNCTION NTILE

description

For NTILE(n), this function will divides rows in a sorted partition into a specific number of groups(in this case, n buckets). Each group is assigned a bucket number starting at one. For the case that cannot be distributed evenly, rows are preferentially allocated to the bucket with the smaller number. The number of rows in all buckets cannot differ by more than 1. For now, n must be constant positive integer.

```
NTILE(n) OVER(partition_by_clause order_by_clause)
```

example

```
select x, y, ntile(2) over(partition by x order by y) as ntile from int_t;

| x | y    | rank     |
|---|------|----------|
| 1 | 1    | 1        |
| 1 | 2    | 1        |
| 1 | 2    | 2        |
| 2 | 1    | 1        |
| 2 | 2    | 1        |
| 2 | 3    | 2        |
| 3 | 1    | 1        |
| 3 | 1    | 1        |
| 3 | 2    | 2        |
```

keywords

```
WINDOW,FUNCTION,NTILE
```

WINDOW_FUNCTION_WINDOW_FUNNEL

WINDOW FUNCTION WINDOW_FUNNEL

description

Searches the longest event chain happened in order (event1, event2, … , eventN) along the timestamp_column with length of window.

- window is the length of time window in seconds.
- mode can be one of the followings:
    - “default”: Defualt mode.
    - “deduplication”: If the same event holds for the sequence of events, then such repeating event interrupts further processing. E.g. the array parameter is [event1=‘A’, event2=‘B’, event3=‘C’, event4=‘D’], and the original event chain is “A-B-C-B-D”. Since event B repeats, the filtered event chain can only be “A-B-C” and the max event level is 3.
    - “fixed”: Don’t allow interventions of other events. E.g. the array parameter is [event1=‘A’, event2=‘B’, event3=‘C’, event4=‘D’], and the original event chain is A->B->D->C, it stops finding A->B->C at the D and the max event level is 2.
    - “increase”: Apply conditions only to events with strictly increasing timestamps.
- timestamp_column specifies column of DATETIME type, sliding time window works on it.
- evnetN is boolean expression like eventID = 1004.

The function works according to the algorithm:

- The function searches for data that triggers the first condition in the chain and sets the event counter to 1. This is the moment when the sliding window starts.
- If events from the chain occur sequentially within the window, the counter is incremented. If the sequence of events is disrupted, the counter is not incremented.
- If the data has multiple event chains at varying points of completion, the function will only output the size of the longest chain.

```
window_funnel(window, mode, timestamp_column, event1, event2, ... , eventN)
```

example

```
CREATE TABLE windowfunnel_test (
                `xwho` varchar(50) NULL COMMENT 'xwho',
                `xwhen` datetime COMMENT 'xwhen',
                `xwhat` int NULL COMMENT 'xwhat'
                )
DUPLICATE KEY(xwho)
DISTRIBUTED BY HASH(xwho) BUCKETS 3
PROPERTIES (
    "replication_num" = "1"
);

INSERT into windowfunnel_test (xwho, xwhen, xwhat) values ('1', '2022-03-12 10:41:00', 1),
```

```
                                                ('1', '2022-03-12 13:28:02', 2),
                                                ('1', '2022-03-12 16:15:01', 3),
                                                ('1', '2022-03-12 19:05:04', 4);

select window_funnel(3600 * 3, 'default', t.xwhen, t.xwhat = 1, t.xwhat = 2 ) AS level from windowfunnel_test t;

| level |
|---|
| 2 |
```

keywords

```
WINDOW,FUNCTION,WINDOW_FUNNEL
```

IPV4_NUM_TO_STRING

IPV4_NUM_TO_STRING

IPV4_NUM_TO_STRING

description

Syntax

```
VARCHAR IPV4_NUM_TO_STRING(BIGINT ipv4_num)
```

Takes a Int16, Int32, Int64 number. Interprets it as an IPv4 address in big endian. Returns a string containing the corresponding IPv4 address in the format A.B.C.d (dot-separated numbers in decimal form). #### notice

```
will return NULL if the input parameter is negative or larger than 4294967295(num value of '255.255.255.255')
```

example

```
mysql> select IPV4_NUM_TO_STRING(3232235521);
+------------------------------+
| IPV4_NUM_TO_STRING(3232235521) |
+------------------------------+
| 192.168.0.1                  |
+------------------------------+
1 row in set (0.01 sec)

mysql> select num,IPV4_NUM_TO_STRING(num) from ipv4_bi;
+------------+------------------------+
| num        | IPV4_NUM_TO_STRING(`num`) |
+------------+------------------------+
|         -1 | NULL                   |
|          0 | 0.0.0.0                |
| 2130706433 | 127.0.0.1              |
| 4294967295 | 255.255.255.255        |
| 4294967296 | NULL                   |
+------------+------------------------+
7 rows in set (0.01 sec)
```

keywords

IPV4_NUM_TO_STRING, IP

INET_NTOA

INET_NTOA

INET_NTOA

description

Syntax

```
VARCHAR INET_NTOA(BIGINT ipv4_num)
```

Takes a Int16, Int32, Int64 number. Interprets it as an IPv4 address in big endian. Returns a string containing the corresponding IPv4 address in the format A.B.C.d (dot-separated numbers in decimal form). #### notice

```
will return NULL if the input parameter is negative or larger than 4294967295(num value of '255.255.255.255')
```

example

```
mysql> select inet_ntoa(3232235521);
+-----------------------------+
| IPV4_NUM_TO_STRING(3232235521) |
+-----------------------------+
| 192.168.0.1                 |
+-----------------------------+
1 row in set (0.01 sec)

mysql> select num,inet_ntoa(num) from ipv4_bi;
+------------+------------------------+
| num        | IPV4_NUM_TO_STRING(`num`) |
+------------+------------------------+
|         -1 | NULL                   |
|          0 | 0.0.0.0                |
| 2130706433 | 127.0.0.1              |
| 4294967295 | 255.255.255.255        |
| 4294967296 | NULL                   |
+------------+------------------------+
7 rows in set (0.01 sec)
```

keywords

INET_NTOA, IP

COSINE_DISTANCE

cosine_distance

description

Syntax

```
DOUBLE cosine_distance(ARRAY<T> array1, ARRAY<T> array2)
```

Calculates the cosine distance between two vectors (the values of the vectors are the coordinates). Return NULL if input array is NULL or any element of array is NULL.

Notice

- nested type of input array support: TINYINT, SMALLINT, INT, BIGINT, LARGEINT, FLOAT, DOUBLE
- input array1 and array2 should have the same element size

example

```
sql> SELECT cosine_distance([1, 2], [2, 3]);
+-----------------------------------------+
| cosine_distance(ARRAY(1, 2), ARRAY(2, 3)) |
+-----------------------------------------+
|                     0.0077221232863322609 |
+-----------------------------------------+
```

keywords

```
COSINE_DISTANCE,DISTANCE,COSINE,ARRAY
```

INNER_PRODUCT

inner_product

description

Syntax

```
DOUBLE inner_product(ARRAY<T> array1, ARRAY<T> array2)
```

Calculates the scalar product of two vectors of the same size. Return NULL if input array is NULL or any element of array is NULL.

Notice

- nested type of input array support: TINYINT, SMALLINT, INT, BIGINT, LARGEINT, FLOAT, DOUBLE
- input array1 and array2 should have the same element size

example

```
sql> SELECT inner_product([1, 2], [2, 3]);
+---------------------------------------+
| inner_product(ARRAY(1, 2), ARRAY(2, 3)) |
+---------------------------------------+
|                                     8 |
+---------------------------------------+
```

keywords

```
INNER_PRODUCT,DISTANCE,ARRAY
```

L1_DISTANCE

l1_distance

description

Syntax

```
DOUBLE l1_distance(ARRAY<T> array1, ARRAY<T> array2)
```

Calculates the distance between two points (the values of the vectors are the coordinates) in L1 space. Return NULL if input array is NULL or any element of array is NULL.

Notice

- nested type of input array support: TINYINT, SMALLINT, INT, BIGINT, LARGEINT, FLOAT, DOUBLE
- input array1 and array2 should have the same element size

example

```
sql> SELECT l1_distance([1, 2], [2, 3]);
+---------------------------------------+
| l1_distance(ARRAY(1, 2), ARRAY(2, 3)) |
+---------------------------------------+
|                                     2 |
+---------------------------------------+
```

keywords

```
L1_DISTANCE,DISTANCE,L1,ARRAY
```

L2_DISTANCE

l2_distance

description

Syntax

```
DOUBLE l2_distance(ARRAY<T> array1, ARRAY<T> array2)
```

Calculates the distance between two points (the values of the vectors are the coordinates) in Euclidean space. Return NULL if input array is NULL or any element of array is NULL.

Notice

- nested type of input array support: TINYINT, SMALLINT, INT, BIGINT, LARGEINT, FLOAT, DOUBLE
- input array1 and array2 should have the same element size

example

```
sql> SELECT l2_distance([1, 2], [2, 3]);
+---------------------------------------+
| l2_distance(ARRAY(1, 2), ARRAY(2, 3)) |
+---------------------------------------+
|                    1.4142135623730951 |
+---------------------------------------+
```

keywords

```
L2_DISTANCE,DISTANCE,L2,ARRAY
```

CAST

CAST

Description

Syntax

```
T cast (input as Type)
```

Converts input to the specified type

example

1. Turn constant, or a column in a table

```
mysql> select cast (1 as BIGINT);
+-------------------+
| CAST(1 AS BIGINT) |
+-------------------+
|                 1 |
+-------------------+
```

2. Transferred raw data

```
curl --location-trusted -u root: -T ~/user_data/bigint -H "columns: tmp_k1, k1=cast(tmp_k1 as BIGINT)"
http://host:port/api/test/bigint/_stream_load
```

- Note: In the import, because the original type is String, when the original data with floating point value is cast, the data will be converted to NULL, such as 12.0. Doris is currently not truncating raw data. *

If you want to force this type of raw data cast to int. Look at the following words:

```
curl --location-trusted -u root: -T ~/user_data/bigint -H "columns: tmp_k1, k1=cast(cast(tmp_k1 as DOUBLE) as BIGINT)"
http://host:port/api/test/bigint/_stream_load

mysql> select cast(cast ("11.2" as double) as bigint);
+----------------------------------------+
| CAST(CAST('11.2' AS DOUBLE) AS BIGINT) |
+----------------------------------------+
|                                     11 |
+----------------------------------------+
1 row in set (0.00 sec)

For the DECIMALV3 ,DATETIME type, the cast operation performs rounding half up.
mysql> select cast (1.115 as DECIMALV3(16, 2));
+---------------------------------+
| cast(1.115 as DECIMALV3(16, 2)) |
+---------------------------------+
|                            1.12 |
+---------------------------------+
```

```
mysql> select cast('2024-12-29-20:40:50.123500' as datetime(3));
+-----------------------------------------------------+
| cast('2024-12-29-20:40:50.123500' as DATETIMEV2(3)) |
+-----------------------------------------------------+
| 2024-12-29 20:40:50.124                             |
+-----------------------------------------------------+

mysql> select cast('2024-12-29-20:40:50.123499' as datetime(3));
+-----------------------------------------------------+
| cast('2024-12-29-20:40:50.123499' as DATETIMEV2(3)) |
+-----------------------------------------------------+
| 2024-12-29 20:40:50.123                             |
+-----------------------------------------------------+
```

keywords

CAST

DIGITAL_MASKING

description

An alias function with the original function being `CONCAT(LEFT(id,3),'****',RIGHT(id,4))`. Performs data masking on the input `digital_number` and returns the masked result.

Syntax

```
digital_masking(digital_number)
```

example

Desensitize the cell phone number

```
select digital_masking(13812345678);
```

```
+-----------------------------+
| digital_masking(13812345678) |
+-----------------------------+
| 138****5678                 |
+-----------------------------+
```

WIDTH_BUCKET

width_bucket

Description

Constructs equi-width histograms, in which the histogram range is divided into intervals of identical size, and returns the bucket number into which the value of an expression falls, after it has been evaluated. The function returns an integer value or null (if any input is null).

Syntax

```
INT width_bucket(Expr expr, T min_value, T max_value, INT num_buckets)
```

Arguments

`expr` - The expression for which the histogram is created. This expression must evaluate to a numeric value or to a value that can be implicitly converted to a numeric value.

The value must be within the range of `-(2^53 - 1)` to `2^53 - 1` (inclusive).

`min_value` and `max_value` - The low and high end points of the acceptable range for the expression. The end points must also evaluate to numeric values and not be equal.

The low and high end points must be within the range of `-(2^53 - 1)` to `2^53 - 1` (inclusive). In addition, the difference between these points must be less than `2^53` (i.e. `abs(max_value - min_value) < 2^53`).

`num_buckets` - The desired number of buckets; must be a positive integer value. A value from the expression is assigned to each bucket, and the function then returns the corresponding bucket number.

Returned value

It returns the bucket number into which the value of an expression falls.

When an expression falls outside the range, the function returns:

`0` if the expression is less than `min_value`.

`num_buckets + 1` if the expression is greater than or equal to `max_value`.

`null` if any input is `null`.

example

```
DROP TABLE IF EXISTS width_bucket_test;

CREATE TABLE IF NOT EXISTS width_bucket_test (
```

```
              `k1` int NULL COMMENT "",
              `v1` date NULL COMMENT "",
              `v2` double NULL COMMENT "",
              `v3` bigint NULL COMMENT ""
            ) ENGINE=OLAP
            DUPLICATE KEY(`k1`)
            DISTRIBUTED BY HASH(`k1`) BUCKETS 1
            PROPERTIES (
            "replication_allocation" = "tag.location.default: 1",
            "storage_format" = "V2"
            );

INSERT INTO width_bucket_test VALUES (1, "2022-11-18", 290000.00, 290000),
                                     (2, "2023-11-18", 320000.00, 320000),
                                     (3, "2024-11-18", 399999.99, 399999),
                                     (4, "2025-11-18", 400000.00, 400000),
                                     (5, "2026-11-18", 470000.00, 470000),
                                     (6, "2027-11-18", 510000.00, 510000),
                                     (7, "2028-11-18", 610000.00, 610000),
                                     (8, null, null, null);

SELECT * FROM width_bucket_test ORDER BY k1;

+------+------------+-----------+--------+
| k1   | v1         | v2        | v3     |
+------+------------+-----------+--------+
|    1 | 2022-11-18 |    290000 | 290000 |
|    2 | 2023-11-18 |    320000 | 320000 |
|    3 | 2024-11-18 | 399999.99 | 399999 |
|    4 | 2025-11-18 |    400000 | 400000 |
|    5 | 2026-11-18 |    470000 | 470000 |
|    6 | 2027-11-18 |    510000 | 510000 |
|    7 | 2028-11-18 |    610000 | 610000 |
|    8 | NULL       |      NULL |   NULL |
+------+------------+-----------+--------+

SELECT k1, v1, v2, v3, width_bucket(v1, date('2023-11-18'), date('2027-11-18'), 4) AS w FROM width_bucket_test ORDER BY k1;

+------+------------+-----------+--------+------+
| k1   | v1         | v2        | v3     | w    |
+------+------------+-----------+--------+------+
|    1 | 2022-11-18 |    290000 | 290000 |    0 |
|    2 | 2023-11-18 |    320000 | 320000 |    1 |
|    3 | 2024-11-18 | 399999.99 | 399999 |    2 |
|    4 | 2025-11-18 |    400000 | 400000 |    3 |
|    5 | 2026-11-18 |    470000 | 470000 |    4 |
|    6 | 2027-11-18 |    510000 | 510000 |    5 |
|    7 | 2028-11-18 |    610000 | 610000 |    5 |
|    8 | NULL       |      NULL |   NULL | NULL |
+------+------------+-----------+--------+------+

SELECT k1, v1, v2, v3, width_bucket(v2, 200000, 600000, 4) AS w FROM width_bucket_test ORDER BY k1;

+------+------------+-----------+--------+------+
| k1   | v1         | v2        | v3     | w    |
+------+------------+-----------+--------+------+
|    1 | 2022-11-18 |    290000 | 290000 |    1 |
|    2 | 2023-11-18 |    320000 | 320000 |    2 |
|    3 | 2024-11-18 | 399999.99 | 399999 |    2 |
|    4 | 2025-11-18 |    400000 | 400000 |    3 |
|    5 | 2026-11-18 |    470000 | 470000 |    3 |
|    6 | 2027-11-18 |    510000 | 510000 |    4 |
|    7 | 2028-11-18 |    610000 | 610000 |    5 |
|    8 | NULL       |      NULL |   NULL | NULL |
+------+------------+-----------+--------+------+
```

```
SELECT k1, v1, v2, v3, width_bucket(v3, 200000, 600000, 4) AS w FROM width_bucket_test ORDER BY k1;

+------+------------+-----------+--------+------+
| k1   | v1         | v2        | v3     | w    |
+------+------------+-----------+--------+------+
|    1 | 2022-11-18 |    290000 | 290000 |    1 |
|    2 | 2023-11-18 |    320000 | 320000 |    2 |
|    3 | 2024-11-18 | 399999.99 | 399999 |    2 |
|    4 | 2025-11-18 |    400000 | 400000 |    3 |
|    5 | 2026-11-18 |    470000 | 470000 |    3 |
|    6 | 2027-11-18 |    510000 | 510000 |    4 |
|    7 | 2028-11-18 |    610000 | 610000 |    5 |
|    8 | NULL       |      NULL |   NULL | NULL |
+------+------------+-----------+--------+------+
```

keywords

WIDTH_BUCKET

ALTER-SYSTEM-ADD-FOLLOWER

ALTER-SYSTEM-ADD-FOLLOWER

Name

ALTER SYSTEM ADD FOLLOWER

Description

This statement is to increase the node of the FOLLOWER role of FRONTEND, (only for administrators!)

grammar:

```
ALTER SYSTEM ADD FOLLOWER "follower_host:edit_log_port"
```

illustrate:

1. host can be a hostname or an ip address
2. edit_log_port : edit_log_port in its configuration file fe.conf

Example

1. Add a FOLLOWER node

   ```
   ALTER SYSTEM ADD FOLLOWER "host_ip:9010"
   ```

Keywords

```
ALTER, SYSTEM, ADD, FOLLOWER, ALTER SYSTEM
```

Best Practice

ALTER-SYSTEM-ADD-OBSERVER

ALTER-SYSTEM-ADD-OBSERVER

Name

ALTER SYSTEM ADD OBSERVER

Description

This statement is to increase the node of the OBSERVER role of FRONTEND, (only for administrators!)

grammar:

```
ALTER SYSTEM ADD OBSERVER "follower_host:edit_log_port"
```

illustrate:

1. host can be a hostname or an ip address
2. edit_log_port : edit_log_port in its configuration file fe.conf

Example

1. Add an OBSERVER node

   ```
   ALTER SYSTEM ADD OBSERVER "host_ip:9010"
   ```

Keywords

```
ALTER, SYSTEM, ADD, OBSERVER, ALTER SYSTEM
```

Best Practice

ALTER-SYSTEM-ADD-BACKEND

ALTER-SYSTEM-ADD-BACKEND

Name

ALTER SYSTEM ADD BACKEND

Description

The ADD BACKEND command is used to add one or more backend nodes to a Doris OLAP database cluster. This command allows administrators to specify the host and port of the new backend nodes, along with optional properties that configure their behavior.

grammar:

```
-- Add nodes (add this method if you do not use the multi-tenancy function)
   ALTER SYSTEM ADD BACKEND "host:heartbeat_port"[,"host:heartbeat_port"...] [PROPERTIES ("key"="value", ...)];
```

Parameters

- `host` can be a hostname or an ip address of the backend node while `heartbeat_port` is the heartbeat port of the node
- `PROPERTIES ("key"="value", ...)`: (Optional) A set of key-value pairs that define additional properties for the backend nodes. These properties can be used to customize the configuration of the backends being added. Available properties include:
    - tag.location: Specifies the resource group where the backend node belongs. For example, PROPERTIES (“tag.location” = “groupb”).

Example

1. Adding Backends Without Additional Properties

   ```
   ALTER SYSTEM ADD BACKEND "host1:9050,host2:9050";
   ```

   This command adds two backend nodes to the cluster:

    - host1 with port 9050
    - host2 with port 9050

   No additional properties are specified, so the default settings will apply to these backends.

2. Adding Backends With Resource Group

```
ALTER SYSTEM ADD BACKEND "host3:9050" PROPERTIES ("tag.location" = "groupb");
```

This command adds a single backend node (host3 with port 9050) to the cluster in resource group `groupb`:

Keywords

```
ALTER, SYSTEM, ADD, BACKEND, PROPERTIES
```

Best Practice

ALTER-SYSTEM-ADD-BROKER

ALTER-SYSTEM-ADD-BROKER

Name

ALTER SYSTEM ADD BROKER

Description

This statement is used to add a BROKER node. (Administrator only!)

grammar:

```
ALTER SYSTEM ADD BROKER broker_name "broker_host1:broker_ipc_port1","broker_host2:broker_ipc_port2",...;
```

Example

1. Add two brokers

   ```
   ALTER SYSTEM ADD BROKER "host1:port", "host2:port";
   ```

2. When fe enable fqdn(fqdn),add one Broker

   ```
   ALTER SYSTEM ADD BROKER "broker_fqdn1:port";
   ```

Keywords

```
ALTER, SYSTEM, ADD, FOLLOWER, ALTER SYSTEM
```

Best Practice

ALTER-SYSTEM-MODIFY-BACKEND

ALTER-SYSTEM-MODIFY-BACKEND

Name

ALTER SYSTEM MKDIFY BACKEND

Description

Modify BE node properties (administrator only!)

grammar:

- Find backend through host and port

```
ALTER SYSTEM MODIFY BACKEND "host:heartbeat_port" SET ("key" = "value"[, ...]);
```

- Find backend through backend_id

```
ALTER SYSTEM MODIFY BACKEND "id1" SET ("key" = "value"[, ...]);
```

illustrate:

1. host can be a hostname or an ip address
2. heartbeat_port is the heartbeat port of the node
3. Modify BE node properties The following properties are currently supported:

- tag.xxxx: resource tag
- disable_query: query disable attribute
- disable_load: import disable attribute

Note: 1. A backend can be set multi resource tags. But must contain “tag.location” type.

Example

1. Modify the resource tag of BE

```
ALTER SYSTEM MODIFY BACKEND "host1:heartbeat_port" SET ("tag.location" = "group_a");
ALTER SYSTEM MODIFY BACKEND "host1:heartbeat_port" SET ("tag.location" = "group_a", "tag.compute" = "c1");
```

```
ALTER SYSTEM MODIFY BACKEND "id1" SET ("tag.location" = "group_a");
ALTER SYSTEM MODIFY BACKEND "id1" SET ("tag.location" = "group_a", "tag.compute" = "c1");
```

2. Modify the query disable property of BE

```
ALTER SYSTEM MODIFY BACKEND "host1:heartbeat_port" SET ("disable_query" = "true");
```

```
ALTER SYSTEM MODIFY BACKEND "id1" SET ("disable_query" = "true");
```

3. Modify the import disable property of BE

```
ALTER SYSTEM MODIFY BACKEND "host1:heartbeat_port" SET ("disable_load" = "true");
```

```
ALTER SYSTEM MODIFY BACKEND "id1" SET ("disable_load" = "true");
```

Keywords

```
ALTER, SYSTEM, ADD, BACKEND, ALTER SYSTEM
```

Best Practice

ALTER-SYSTEM-DECOMMISSION-BACKEND

ALTER-SYSTEM-DECOMMISSION-BACKEND

Name

ALTER SYSTEM DECOMMISSION BACKEND

Description

The node offline operation is used to safely log off the node. The operation is asynchronous. If successful, the node is eventually removed from the metadata. If it fails, the logout will not be done (only for admins!)

grammar:

- Find backend through host and port

```
ALTER SYSTEM DECOMMISSION BACKEND "host:heartbeat_port"[,"host:heartbeat_port"...];
```

- Find backend through backend_id

```
ALTER SYSTEM DECOMMISSION BACKEND "id1","id2"...;
```

illustrate:

1. host can be a hostname or an ip address
2. heartbeat_port is the heartbeat port of the node
3. The node offline operation is used to safely log off the node. The operation is asynchronous. If successful, the node is eventually removed from the metadata. If it fails, the logout will not be completed.
4. You can manually cancel the node offline operation. See CANCEL DECOMMISSION

Example

1. Offline two nodes

   ```
   ALTER SYSTEM DECOMMISSION BACKEND "host1:port", "host2:port";
   ```

   ```
   ALTER SYSTEM DECOMMISSION BACKEND "id1", "id2";
   ```

Keywords

```
ALTER, SYSTEM, DECOMMISSION, BACKEND, ALTER SYSTEM
```

Best Practice

ALTER-SYSTEM-DROP-FOLLOWER

Name

ALTER SYSTEM DROP FOLLOWER

Description

This statement is to delete the node of the FOLLOWER role of FRONTEND, (only for administrators!)

Grammar:

```
ALTER SYSTEM DROP FOLLOWER "follower_host:edit_log_port"
```

Illustration:

1. host can be a hostname or an ip address
2. edit_log_port : edit_log_port in its configuration file fe.conf

Example

1. Delete a FOLLOWER node

   ```
   ALTER SYSTEM DROP FOLLOWER "host_ip:9010"
   ```

Keywords

ALTER, SYSTEM, DROP, FOLLOWER, ALTER SYSTEM

ALTER-SYSTEM-DROP-OBSERVER

ALTER-SYSTEM-DROP-OBSERVER

Name

ALTER SYSTEM DROP OBSERVER

Description

This statement is to delete the node of the OBSERVER role of FRONTEND, (only for administrators!)

grammar:

```
ALTER SYSTEM DROP OBSERVER "follower_host:edit_log_port"
```

illustrate:

1. host can be a hostname or an ip address
2. edit_log_port : edit_log_port in its configuration file fe.conf

Example

1. Add a FOLLOWER node

   ```
   ALTER SYSTEM DROP OBSERVER "host_ip:9010"
   ```

Keywords

```
ALTER, SYSTEM, DROP, OBSERVER, ALTER SYSTEM
```

Best Practice

ALTER-SYSTEM-DROP-BACKEND

ALTER-SYSTEM-DROP-BACKEND

Name

ALTER SYSTEM DROP BACKEND

Description

This statement is used to delete the BACKEND node (administrator only!)

grammar:

- Find backend through host and port

```
ALTER SYSTEM DROP BACKEND "host:heartbeat_port"[,"host:heartbeat_port"...]
```

- Find backend through backend_id

```
ALTER SYSTEM DROP BACKEND "id1","id2"...;
```

illustrate:

1. host can be a hostname or an ip address
2. heartbeat_port is the heartbeat port of the node
3. Adding and deleting nodes is a synchronous operation. These two operations do not consider the existing data on the node, and the node is directly deleted from the metadata, please use it with caution.

Example

1. Delete two nodes

   ```
   ALTER SYSTEM DROP BACKEND "host1:port", "host2:port";
   ```

   ```
   ALTER SYSTEM DROP BACKEND "ids1", "ids2";
   ```

Keywords

```
ALTER, SYSTEM, DROP, BACKEND, ALTER SYSTEM
```

Best Practice

ALTER-SYSTEM-DROP-BROKER

ALTER-SYSTEM-DROP-BROKER

Name

ALTER SYSTEM DROP BROKER

Description

This statement is to delete the BROKER node, (administrator only)

grammar:

```
-- Delete all brokers
ALTER SYSTEM DROP ALL BROKER broker_name
-- Delete a Broker node
ALTER SYSTEM DROP BROKER broker_name "host:port"[,"host:port"...];
```

Example

1. Delete all brokers

   ```
   ALTER SYSTEM DROP ALL BROKER broker_name
   ```

2. Delete a Broker node

   ```
   ALTER SYSTEM DROP BROKER broker_name "host:port"[,"host:port"...];
   ```

Keywords

```
ALTER, SYSTEM, DROP, FOLLOWER, ALTER SYSTEM
```

Best Practice

CANCEL-ALTER-SYSTEM

CANCEL-ALTER-SYSTEM

Name

CANCEL DECOMMISSION

Description

This statement is used to undo a node offline operation. (Administrator only!)

grammar:

- Find backend through host and port

```
CANCEL DECOMMISSION BACKEND "host:heartbeat_port"[,"host:heartbeat_port"...];
```

- Find backend through backend_id

```
CANCEL DECOMMISSION BACKEND "id1","id2","id3...";
```

Example

1. Cancel the offline operation of both nodes:

   ```
   CANCEL DECOMMISSION BACKEND "host1:port", "host2:port";
   ```

2. Cancel the offline operation of the node with backend_id 1:

   ```
       CANCEL DECOMMISSION BACKEND "1","2";
   ```

Keywords

```
CANCEL, DECOMMISSION, CANCEL ALTER
```

Best Practice

CREATE-ROLE

CREATE-ROLE

Name

CREATE ROLE

Description

The statement user creates a role

```
CREATE ROLE rol_name;
```

This statement creates an unprivileged role, which can be subsequently granted with the GRANT command.

Example

1. Create a character

```
CREATE ROLE role1;
```

Keywords

```
CREATE, ROLE
```

Best Practice

CREATE-USER

CREATE-USER

Name

CREATE USER

Description

The CREATE USER command is used to create a Doris user.

```
CREATE USER [IF EXISTS] user_identity [IDENTIFIED BY 'password']
[DEFAULT ROLE 'role_name']
[password_policy]

user_identity:
    'user_name'@'host'

password_policy:

    1. PASSWORD_HISTORY [n|DEFAULT]
    2. PASSWORD_EXPIRE [DEFAULT|NEVER|INTERVAL n DAY/HOUR/SECOND]
    3. FAILED_LOGIN_ATTEMPTS n
    4. PASSWORD_LOCK_TIME [n DAY/HOUR/SECOND|UNBOUNDED]
```

In Doris, a user_identity uniquely identifies a user. user_identity consists of two parts, user_name and host, where username is the username. host Identifies the host address where the client connects. The host part can use % for fuzzy matching. If no host is specified, it defaults to '%', which means the user can connect to Doris from any host.

The host part can also be specified as a domain, the syntax is: 'user_name'@['domain'], even if it is surrounded by square brackets, Doris will think this is a domain and try to resolve its ip address. .

If a role (ROLE) is specified, the newly created user will be automatically granted the permissions of the role. If not specified, the user has no permissions by default. The specified ROLE must already exist.

`password_policy` is a clause used to specify policies related to password authentication login. Currently, the following policies are supported:

1. `PASSWORD_HISTORY`

   Whether to allow the current user to use historical passwords when resetting their passwords. For example, `PASSWORD_HISTORY 10` means that it is forbidden to use the password set in the past 10 times as a new password. If set to `PASSWORD_HISTORY DEFAULT`, the value in the global variable `password_history` will be used. `0` means do not enable this feature. Default is 0.

2. `PASSWORD_EXPIRE`

Set the expiration time of the current user's password. For example `PASSWORD_EXPIRE INTERVAL 10 DAY` means the password will expire in 10 days. `PASSWORD_EXPIRE NEVER` means that the password does not expire. If set to `PASSWORD_EXPIRE DEFAULT`, the value in the global variable `default_password_lifetime` is used. Defaults to NEVER (or 0), which means it will not expire.

3. `FAILED_LOGIN_ATTEMPTS` and `PASSWORD_LOCK_TIME`

   When the current user logs in, if the user logs in with the wrong password for n times, the account will be locked, and the lock time is set. For example, `FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 1 DAY` means that if you log in wrongly for 3 times, the account will be locked for one day.

   A locked account can be actively unlocked through the `ALTER USER` statement.

Example

1. Create a passwordless user (if host is not specified, it is equivalent to jack@'%')

```
CREATE USER 'jack';
```

2. Create a user with a password to allow login from '172.10.1.10'

```
CREATE USER jack@'172.10.1.10' IDENTIFIED BY '123456';
```

3. In order to avoid passing plaintext, use case 2 can also be created in the following way

```
CREATE USER jack@'172.10.1.10' IDENTIFIED BY PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9';
The encrypted content can be obtained through PASSWORD(), for example:
SELECT PASSWORD('123456');
```

4. Create a user that is allowed to log in from the '192.168' subnet, and specify its role as example_role

```
CREATE USER 'jack'@'192.168.%' DEFAULT ROLE 'example_role';
```

5. Create a user that is allowed to log in from the domain 'example_domain'

```
CREATE USER 'jack'@['example_domain'] IDENTIFIED BY '12345';
```

6. Create a user and assign a role

```
CREATE USER 'jack'@'%' IDENTIFIED BY '12345' DEFAULT ROLE 'my_role';
```

7. Create a user, set the password to expire after 10 days, and set the account to be locked for one day if you log in failed for 3 times.

```
CREATE USER 'jack' IDENTIFIED BY '12345' PASSWORD_EXPIRE INTERVAL 10 DAY FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 1 DAY;
```

8. Create a user and restrict non-resetable passwords to the last 8 passwords used.

```
CREATE USER 'jack' IDENTIFIED BY '12345' PASSWORD_HISTORY 8;
```

Keywords

```
CREATE, USER
```

Best Practice

ALTER-USER

ALTER USER

Name

ALTER USER

Description

The ALTER USER command is used to modify a user's account attributes, including passwords, and password policies, etc.

> Note that.
>
> This command give over supports modifying user roles from versions 2.0. Please use GRANT and REVOKE for related operations

```
ALTER USER [IF EXISTS] user_identity [IDENTIFIED BY 'password']
[password_policy]

user_identity:
    'user_name'@'host'

password_policy:

    1. PASSWORD_HISTORY [n|DEFAULT]
    2. PASSWORD_EXPIRE [DEFAULT|NEVER|INTERVAL n DAY/HOUR/SECOND]
    3. FAILED_LOGIN_ATTEMPTS n
    4. PASSWORD_LOCK_TIME [n DAY/HOUR/SECOND|UNBOUNDED]
    5. ACCOUNT_UNLOCK
```

About `user_identity` and `password_policy`, Please refer to `CREATE USER`.

`ACCOUNT_UNLOCK` is used to unlock a locked user.

In an ALTER USER command, only one of the following account attributes can be modified at the same time:

1. Change password
2. Modify `PASSWORD_HISTORY`
3. Modify `PASSWORD_EXPIRE`
4. Modify `FAILED_LOGIN_ATTEMPTS` and `PASSWORD_LOCK_TIME`
5. Unlock users

Example

1. Change the user's password

```
ALTER USER jack@'%' IDENTIFIED BY "12345";
```

2. Modify the user’s password policy

```
ALTER USER jack@'%' FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 1 DAY;
```

3. Unlock a user

```
ALTER USER jack@'%' ACCOUNT_UNLOCK
```

Keywords

```
ALTER, USER
```

Best Practice

1. Modify the password policy
    1. Modify `PASSWORD_EXPIRE` will reset the timing of password expiration time.
    2. Modify `FAILED_LOGIN_ATTEMPTS` or `PASSWORD_LOCK_TIME` will unlock the user.

SET-PASSWORD

SET-PASSWORD

Name

SET PASSWORD

Description

The SET PASSWORD command can be used to modify a user's login password. If the [FOR user_identity] field does not exist, then change the current user's password

```
SET PASSWORD [FOR user_identity] =
    [PASSWORD('plain password')]|['hashed password']
```

Note that the user_identity here must exactly match the user_identity specified when creating a user with CREATE USER, otherwise an error will be reported that the user does not exist. If user_identity is not specified, the current user is 'username'@'ip', which may not match any user_identity. Current users can be viewed through SHOW GRANTS.

The plaintext password is input in the PASSWORD() method; when using a string directly, the encrypted password needs to be passed. To modify the passwords of other users, administrator privileges are required.

Example

1. Modify the current user's password

   ```
   SET PASSWORD = PASSWORD('123456')
   SET PASSWORD = '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'
   ```

2. Modify the specified user password

   ```
   SET PASSWORD FOR 'jack'@'192.%' = PASSWORD('123456')
   SET PASSWORD FOR 'jack'@['domain'] = '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'
   ```

Keywords

```
SET, PASSWORD
```

Best Practice

SET-PROPERTY

SET-PROPERTY

Name

SET PROPERTY

Description

Set user attributes, including resources assigned to users, importing clusters, etc.

```
SET PROPERTY [FOR 'user'] 'key' = 'value' [, 'key' = 'value']
```

The user attribute set here is for user, not user_identity. That is, if two users ‘jack’@‘%’ and ‘jack’@‘192.%’ are created through the CREATE USER statement, the SET PROPERTY statement can only be used for the user jack, not ‘jack’@‘%’ or ‘jack’@‘192.%’

key:

Super user privileges:

max_user_connections: The maximum number of connections.

max_query_instances: The number of instances that a user can use to execute a query at the same time.

sql_block_rules: Set sql block rules. Once set, queries sent by this user will be rejected if they match the rules.

cpu_resource_limit: Limit the cpu resources for queries. See the introduction to the session variable `cpu_resource_limit` for details. -1 means not set.

exec_mem_limit: Limit the memory usage of the query. See the introduction to the session variable `exec_mem_limit` for details. -1 means not set.

resource_tags: Specifies the user’s resource tag permissions.

query_timeout: Specifies the user’s query timeout permissions.

```
Note: If the attributes `cpu_resource_limit`, `exec_mem_limit` are not set, the value in the session variable will be used by default.
```

Example

1. Modify the maximum number of user jack connections to 1000

   ```
   SET PROPERTY FOR 'jack' 'max_user_connections' = '1000';
   ```

2. Modify the number of available instances for user jack's query to 3000

```
SET PROPERTY FOR 'jack' 'max_query_instances' = '3000';
```

3. Modify the sql block rule of user jack

```
SET PROPERTY FOR 'jack' 'sql_block_rules' = 'rule1, rule2';
```

4. Modify the cpu usage limit of user jack

```
SET PROPERTY FOR 'jack' 'cpu_resource_limit' = '2';
```

5. Modify the user's resource tag permissions

```
SET PROPERTY FOR 'jack' 'resource_tags.location' = 'group_a, group_b';
```

6. Modify the user's query memory usage limit, in bytes

```
SET PROPERTY FOR 'jack' 'exec_mem_limit' = '2147483648';
```

7. Modify the user's query timeout limit, in second

```
SET PROPERTY FOR 'jack' 'query_timeout' = '500';
```

Keywords

```
SET, PROPERTY
```

Best Practice

LDAP

LDAP

Name

LDAP

Description

SET LDAP_ADMIN_PASSWORD

```
  SET LDAP_ADMIN_PASSWORD = PASSWORD('plain password')
```

The SET LDAP_ADMIN_PASSWORD command is used to set the LDAP administrator password. When using LDAP authentication, doris needs to use the administrator account and password to query the LDAP service for login user information.

Example

1. Set the LDAP administrator password

```
SET LDAP_ADMIN_PASSWORD = PASSWORD('123456')
```

Keywords

```
 LDAP, PASSWORD, LDAP_ADMIN_PASSWORD
```

Best Practice

GRANT

GRANT

Name

GRANT

Description

The GRANT command has the following functions:

1. Grant the specified permissions to a user or role.
2. Grant the specified role to a user.

> Note that.
>
> “Grant specified roles to user” is supported in versions 2.0 and later

```
GRANT privilege_list ON priv_level TO user_identity [ROLE role_name]

GRANT privilege_list ON RESOURCE resource_name TO user_identity [ROLE role_name]

GRANT role_list TO user_identity
```

GRANT privilege_list ON WORKLOAD GROUP workload_group_name TO user_identity [ROLE role_name]

privilege_list is a list of privileges to be granted, separated by commas. Currently Doris supports the following permissions:

```
NODE_PRIV: Cluster node operation permissions, including node online and offline operations. User who has NODE_PRIV and GRANT_PRIV
permission, can grant NODE_PRIV to other users.
ADMIN_PRIV: All privileges except NODE_PRIV.
GRANT_PRIV: Privilege for operation privileges. Including creating and deleting users, roles, authorization and revocation, setting
passwords, etc.
SELECT_PRIV: read permission on the specified database or table
LOAD_PRIV: Import privileges on the specified database or table
ALTER_PRIV: Schema change permission for the specified database or table
CREATE_PRIV: Create permission on the specified database or table
DROP_PRIV: drop privilege on the specified database or table
USAGE_PRIV: access to the specified resource
SHOW_VIEW_PRIV: View permission to `view` creation statements (starting from version 2.0.3, 'SELECT_PRIV' and 'LOAD_PRIV'
permissions cannot be 'SHOW CREATE TABLE view_name', has one of `CREATE_PRIV`, `ALTER_PRIV`, `DROP_PRIV`, `SHOW_VIEW_PRIV` can `SHOW
CREATE TABLE view_name`)

ALL and READ_WRITE in legacy permissions will be converted to: SELECT_PRIV,LOAD_PRIV,ALTER_PRIV,CREATE_PRIV,DROP_PRIV;
READ_ONLY is converted to SELECT_PRIV.
```

Permission classification:

```
1. Node Privilege: NODE_PRIV
2. database table permissions: SELECT_PRIV, LOAD_PRIV, ALTER_PRIV, CREATE_PRIV, DROP_PRIV
3. Resource  and workload groups Privilege: USAGE_PRIV
```

Priv_level supports the following four forms:

```
1. *.*.* permissions can be applied to all catalogs, all databases and all tables in them
2. catalog_name.*.* permissions can be applied to all databases and all tables in them
3. catalog_name.db.* permissions can be applied to all tables under the specified database
4. catalog_name.db.tbl permission can be applied to the specified table under the specified database

The catalog or database, table specified here may be not exist.
```

resource_name supports the following two forms:

```
1. * Permissions apply to all resources
2. The resource permission applies to the specified resource

The resource specified here can be a non-existing resource. In addition, please distinguish the resources here from external tables,
and use catalog as an alternative if you use external tables.
```

workload_group_name specifies the workload group name and supports % and _ match characters, % can match any string and _ matches any single character.

user_identity:

```
The user_identity syntax here is the same as CREATE USER. And must be a user_identity created with CREATE USER. The host in
user_identity can be a domain name. If it is a domain name, the effective time of the authority may be delayed by about 1 minute.

You can also assign permissions to the specified ROLE, if the specified ROLE does not exist, it will be created automatically.
```

role_list is the list of roles to be assigned, separated by commas,the specified role must exist.

Example

1. Grant permissions to all catalog and databases and tables to the user

   ```
   GRANT SELECT_PRIV ON *.*.* TO 'jack'@'%';
   ```

2. Grant permissions to the specified database table to the user

   ```
   GRANT SELECT_PRIV,ALTER_PRIV,LOAD_PRIV ON ctl1.db1.tbl1 TO 'jack'@'192.8.%';
   ```

3. Grant permissions to the specified database table to the role

   ```
   GRANT LOAD_PRIV ON ctl1.db1.* TO ROLE 'my_role';
   ```

4. Grant access to all resources to users

```
GRANT USAGE_PRIV ON RESOURCE * TO 'jack'@'%';
```

5. Grant the user permission to use the specified resource

```
GRANT USAGE_PRIV ON RESOURCE 'spark_resource' TO 'jack'@'%';
```

6. Grant access to specified resources to roles

```
GRANT USAGE_PRIV ON RESOURCE 'spark_resource' TO ROLE 'my_role';
```

> **Tips**
>
> This feature is supported since the Apache Doris 2.0 version

7. Grant the specified role to a user

```
GRANT 'role1','role2' TO 'jack'@'%';
```

8. Grant the specified workload group ‘g1’ to user jack

```
GRANT USAGE_PRIV ON WORKLOAD GROUP 'g1' TO 'jack'@'%'.
```

9. match all workload groups granted to user jack

```
GRANT USAGE_PRIV ON WORKLOAD GROUP '%' TO 'jack'@'%'.
```

10. grant the workload group ‘g1’ to the role my_role

```
GRANT USAGE_PRIV ON WORKLOAD GROUP 'g1' TO ROLE 'my_role'.
```

11. Allow jack to view the creation statement of view1 under db1

```
GRANT SHOW_VIEW_PRIV ON db1.view1 TO 'jack'@'%';
```

Keywords

```
GRANT
```

Best Practice

REVOKE

REVOKE

Name

REVOKE

Description

The REVOKE command has the following functions:

1. Revoke the specified permission of a user or a role.
2. Revoke the specified role previously granted to a user.

> Note that.
>
> “Revoke the specified roles previously granted to a user” is supported in versions 2.0 and later

```
REVOKE privilege_list ON db_name[.tbl_name] FROM user_identity [ROLE role_name]

REVOKE privilege_list ON RESOURCE resource_name FROM user_identity [ROLE role_name]

REVOKE role_list FROM user_identity
```

user_identity:

The user_identity syntax here is the same as CREATE USER. And must be a user_identity created with CREATE USER. The host in user_identity can be a domain name. If it is a domain name, the revocation time of permissions may be delayed by about 1 minute.

It is also possible to revoke the permissions of the specified ROLE, the executed ROLE must exist.

role_list is the list of roles to be revoked, separated by commas. The specified roles must exist.

Example

1. Revoke the permission of user jack database testDb

   ```
   REVOKE SELECT_PRIV ON db1.* FROM 'jack'@'192.%';
   ```

2. Revoke user jack resource spark_resource permission

   ```
   REVOKE USAGE_PRIV ON RESOURCE 'spark_resource' FROM 'jack'@'192.%';
   ```

3. Revoke the roles role1 and role2 previously granted to jack

```
REVOKE 'role1','role2' FROM 'jack'@'192.%';
```

Keywords

```
REVOKE
```

Best Practice

DROP-ROLE

DROP-ROLE

Description

The statement user removes a role

```
DROP ROLE [IF EXISTS] role1;
```

Deleting a role does not affect the permissions of users who previously belonged to the role. It is only equivalent to decoupling the role from the user. The permissions that the user has obtained from the role will not change

Example

1. Drop a role1

```
DROP ROLE role1;
```

Keywords

```
DROP, ROLE
```

Best Practice

DROP-USER

DROP-USER

Name

DROP USER

Description

Delete a user

```
DROP USER 'user_identity'

   `user_identity`:

       user@'host'
       user@['domain']
```

Delete the specified user identitiy.

Example

1. Delete user jack@‘192.%’

```
DROP USER 'jack'@'192.%'
```

Keywords

```
DROP, USER
```

Best Practice

ADMIN-SHOW-CONFIG

ADMIN-SHOW-CONFIG

Name

ADMIN SHOW CONFIG

Description

This statement is used to display the configuration of the current cluster (currently only the configuration items of FE are supported)

grammar:

```
ADMIN SHOW FRONTEND CONFIG [LIKE "pattern"];
```

The columns in the results have the following meanings:

1. Key: Configuration item name
2. Value: Configuration item value
3. Type: Configuration item type
4. IsMutable: Whether it can be set by ADMIN SET CONFIG command
5. MasterOnly: Is it only applicable to Master FE
6. Comment: Configuration item description

Example

1. View the configuration of the current FE node

   ```
   ADMIN SHOW FRONTEND CONFIG;
   ```

2. Use the like predicate to search the configuration of the current Fe node

   ```
   mysql> ADMIN SHOW FRONTEND CONFIG LIKE '%check_java_version%';
   +--------------------+-------+---------+---------- -+------------+---------+
   | Key | Value | Type | IsMutable | MasterOnly | Comment |
   +--------------------+-------+---------+---------- -+------------+---------+
   | check_java_version | true | boolean | false | false | |
   +--------------------+-------+---------+---------- -+------------+---------+
   1 row in set (0.01 sec)
   ```

Keywords

```
ADMIN, SHOW, CONFIG, ADMIN SHOW
```

Best Practice

ADMIN-SET-CONFIG

ADMIN-SET-CONFIG

Name

ADMIN SET CONFIG

Description

This statement is used to set the configuration items of the cluster (currently only the configuration items of FE are supported). The settable configuration items can be viewed through the ADMIN SHOW FRONTEND CONFIG; command.

grammar:

```
ADMIN SET FRONTEND CONFIG ("key" = "value") [ALL];
ADMIN SET ALL FRONTENDS CONFIG ("key" = "value");
```

illustrate:

1. If `ALL` keyword is used, this config will be applied to all FE(except master_only configuration)

Example

1. Set ‘disable_balance’ to true

   ```
   ADMIN SET FRONTEND CONFIG ("disable_balance" = "true");
   ```

Keywords

```
ADMIN, SET, CONFIG
```

Best Practice

SET-VARIABLE

SET-VARIABLE

Name

SET VARIABLE

Description

This statement is mainly used to modify Doris system variables. These system variables can be modified at the global and session level, and some can also be modified dynamically. You can also view these system variables with `SHOW VARIABLE`.

grammar:

```
SET variable_assignment [, variable_assignment] ...
```

illustrate:

1. variable_assignment: user_var_name = expr | [GLOBAL | SESSION] system_var_name = expr

> Note:
>
> 1. Only ADMIN users can set variables to take effect globally
> 2. The globally effective variable affects the current session and new sessions thereafter, but does not affect other sessions that currently exist.

Example

1. Set the time zone to Dongba District

   ```
   SET time_zone = "Asia/Shanghai";
   ```

2. Set the global execution memory size

   ```
   SET GLOBAL exec_mem_limit = 137438953472
   ```

Keywords

```
SET, VARIABLE
```

INSTALL-PLUGIN

INSTALL-PLUGIN

Name

INSTALL PLUGIN

Description

This statement is used to install a plugin.

grammar:

```
INSTALL PLUGIN FROM [source] [PROPERTIES ("key"="value", ...)]
```

source supports three types:

1. An absolute path to a zip file.
2. An absolute path to a plugin directory.
3. Point to a zip file download path with http or https protocol

Example

1. Install a local zip file plugin:

   ```
   INSTALL PLUGIN FROM "/home/users/doris/auditdemo.zip";
   ```

2. Install the plugin in a local directory:

   ```
   INSTALL PLUGIN FROM "/home/users/doris/auditdemo/";
   ```

3. Download and install a plugin:

   ```
   INSTALL PLUGIN FROM "http://mywebsite.com/plugin.zip";
   ```

   Note than an md5 file with the same name as the `.zip` file needs to be placed, such as `http://mywebsite.com/plugin.zip.md5` . The content is the MD5 value of the .zip file.

4. Download and install a plugin, and set the md5sum value of the zip file at the same time:

   ```
   INSTALL PLUGIN FROM "http://mywebsite.com/plugin.zip" PROPERTIES("md5sum" = "73877f6029216f4314d712086a146570");
   ```

Keywords

```
INSTALL, PLUGIN
```

Best Practice

UNINSTALL-PLUGIN

UNINSTALL-PLUGIN

Name

UNINSTALL PLUGIN

Description

This statement is used to uninstall a plugin.

grammar:

```
UNINSTALL PLUGIN plugin_name;
```

plugin_name can be viewed with the `SHOW PLUGINS;` command.

Only non-builtin plugins can be uninstalled.

Example

1. Uninstall a plugin:

   ```
   UNINSTALL PLUGIN auditdemo;
   ```

Keywords

```
UNINSTALL, PLUGIN
```

Best Practice

ADMIN SET REPLICA STATUS

Description

This statement is used to set the status of a specified replica. Currently, it is only used for manually setting certain replica statuses to `BAD`, `DROP`, or `OK`, allowing the system to automatically repair these replicas.

Syntax

```
ADMIN SET REPLICA STATUS
PROPERTIES ("tablet_id"="<tablet_id>","backend_id"="<backend_id>","status"="<status>")
```

Required Parameters

1. `<tablet_id>`

The ID of the tablet whose replica status needs to be set.

2. `<backend_id>`

The ID of the BE node where the replica is located.

3. `<status>`

Currently, only the values “drop”, “bad”, and “ok” are supported. If the specified replica does not exist or is already in a bad state, the operation will be ignored.

Notes：

- Setting a replica to BAD status

  A replica marked as BAD cannot be read or written. However, setting a replica to BAD may not always take effect. If the replica’s actual data is correct, and the BE reports the replica status as OK, the FE will automatically restore the replica back to OK. This operation may immediately delete the replica, so proceed with caution.

- Setting a replica to DROP status

  A replica marked as DROP can still be read and written. The system will first add a healthy replica on another machine before deleting the DROP replica. Compared to BAD, setting a replica to DROP is a safer operation.

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

- Set the replica of tablet 10003 on BE 10001 to bad

```
ADMIN SET REPLICA STATUS PROPERTIES("tablet_id" = "10003", "backend_id" = "10001", "status" = "bad");
```

- Set the replica of tablet 10003 on BE 10001 to drop

```
ADMIN SET REPLICA STATUS PROPERTIES("tablet_id" = "10003", "backend_id" = "10001", "status" = "drop");
```

- Set the replica of tablet 10003 on BE 10001 to ok

```
ADMIN SET REPLICA STATUS PROPERTIES("tablet_id" = "10003", "backend_id" = "10001", "status" = "ok");
```

SHOW REPLICA DISTRIBUTION

Description

This statement is used to display the replica status information for a table or partition.

Syntax

```
SHOW REPLICA DISTRIBUTION FROM [ <database_name>.]<table_name> [<partition_list>]
[where_clause]
```

Where:

```
partition_list
  : PARTITION (<partition_name>[ , ... ])
```

Where:

```
where_clause
: WHERE <output_column_name> = <value>
```

Required Parameters

1. `<table_name>`

> The identifier (i.e., name) of the table, which must be unique within the database (Database).
>
> The identifier must start with a letter character (or any character from supported languages if Unicode name support is enabled), and cannot contain spaces or special characters unless the entire identifier string is enclosed in backticks (e.g., My Object).
>
> Identifiers cannot use reserved keywords.
>
> For more details, refer to the identifier requirements and reserved keywords.

Optional Parameters

1. `<db_name>`

> The identifier (i.e., name) of the database, which must be unique within the cluster (Cluster).
>
> The identifier must start with a letter character (or any character from supported languages if Unicode name support is enabled), and cannot contain spaces or special characters unless the entire identifier

> string is enclosed in backticks (e.g., My Object).
>
> Identifiers cannot use reserved keywords.
>
> For more details, refer to the identifier requirements and reserved keywords.

2. `<partition_list>`

> A comma-separated list of partition identifiers (i.e., names), which must be unique within the table (Table).
>
> The identifier must start with a letter character (or any character from supported languages if Unicode name support is enabled), and cannot contain spaces or special characters unless the entire identifier string is enclosed in backticks (e.g., My Object).
>
> Identifiers cannot use reserved keywords.
>
> For more details, refer to the identifier requirements and reserved keywords.

Return Value

Column	DataType	Note
BackendId	Int	The ID of the BE (Backend) node where the replica is located.
ReplicaNum	Int	The number of replicas on this BE node.
ReplicaSize	Int	The total storage size occupied by all replicas on this BE node (in bytes).
NumGraph	String	A visual representation (using > symbols) of the replica count proportion.
NumPercent	String	The percentage representation of the replica count, e.g., 100.00%.
SizeGraph	String	A visual representation (using > symbols) of the replica storage size proportion.
SizePercent	String	The percentage representation of the replica storage size, e.g., 100.00%.

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

- 查看表的副本分布

```
SHOW REPLICA DISTRIBUTION FROM sell_user;
```

```
*************************** 1. row ***************************
BackendId: 10009
 ReplicaNum: 4
ReplicaSize: 8857
   NumGraph: >>>>
```

SHOW REPLICA STATUS

Description

This statement is used to display the replica status information for a table or partition.

Syntax

```
SHOW REPLICA STATUS FROM [ <database_name>.]<table_name> [<partition_list>]
[where_clause]
```

Where:

```
partition_list
  : PARTITION (<partition_name>[ , ... ])
```

Where:

```
where_clause
: WHERE <output_column_name> = <value>
```

Required Parameters

1. `<table_name>`

> The identifier (i.e., name) of the table, which must be unique within the database (Database).
>
> The identifier must start with a letter character (or any character from supported languages if Unicode name support is enabled), and cannot contain spaces or special characters unless the entire identifier string is enclosed in backticks (e.g., My Object).
>
> Identifiers cannot use reserved keywords.
>
> For more details, refer to the identifier requirements and reserved keywords.

Optional Parameters

1. `<db_name>`

> The identifier (i.e., name) of the database, which must be unique within the cluster (Cluster).
>
> The identifier must start with a letter character (or any character from supported languages if Unicode name support is enabled), and cannot contain spaces or special characters unless the entire identifier

string is enclosed in backticks (e.g., My Object).

Identifiers cannot use reserved keywords.

For more details, refer to the identifier requirements and reserved keywords.

2. `<partition_list>`

A comma-separated list of partition identifiers (i.e., names), which must be unique within the table (Table).

The identifier must start with a letter character (or any character from supported languages if Unicode name support is enabled), and cannot contain spaces or special characters unless the entire identifier string is enclosed in backticks (e.g., My Object).

Identifiers cannot use reserved keywords.

For more details, refer to the identifier requirements and reserved keywords.

3. `WHERE <output_column_name> = <value>`

Specifies the filtering condition for the output. The output_column_name must be part of the output field list.

When output_column_name is STATUS, the value can be one of the following:

- DEAD: The backend where the replica resides is unavailable.
- VERSION_ERROR: The replica's data version is incomplete.
- SCHEMA_ERROR: The schema hash of the replica is incorrect.
- MISSING: The replica does not exist.

Return Value

Column	DataType	Note
TabletId	Int	Unique identifier for the tablet.
ReplicaId	Int	Unique identifier for the replica.
BackendId	Int	ID of the Backend (BE) node where the replica is located.
Version	Int	The current version of the replica.
LastFailedVersion	Int	The version when the replica last failed. A value of -1 means no failure.
LastSuccessVersion	Int	The version when the replica last succeeded.
CommittedVersion	Int	The committed version of the replica.
SchemaHash	Int	A hash value representing the schema of the replica.
VersionNum	Int	The number of versions the replica has gone through.

Column	DataType	Note
IsBad	Boolean	Indicates whether the replica is in a bad state (true/false).
IsUserDrop	Boolean	Indicates if the replica has been marked for user-driven deletion.
State	String	The current state of the replica (e.g., NORMAL).
Status	String	The health status of the replica (e.g., OK).

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

- Display the replica status for all replicas of a table

```
SHOW REPLICA STATUS FROM db1.tbl1;
```

```
+----------+-----------+-----------+---------+-------------------+--------------------+------------------+------------+----
--------+-------+------------+--------+--------+
| TabletId | ReplicaId | BackendId | Version | LastFailedVersion | LastSuccessVersion | CommittedVersion | SchemaHash |
VersionNum | IsBad | IsUserDrop | State  | Status |
+----------+-----------+-----------+---------+-------------------+--------------------+------------------+------------+----
--------+-------+------------+--------+--------+
| 10145    | 10146     | 10009     | 14      | -1                | 14                 | 14               | 182881783  | 1
| false | false      | NORMAL | OK     |
| 10147    | 10148     | 10009     | 14      | -1                | 14                 | 14               | 182881783  | 1
| false | false      | NORMAL | OK     |
| 10149    | 10150     | 10009     | 14      | -1                | 14                 | 14               | 182881783  | 1
| false | false      | NORMAL | OK     |
| 10151    | 10152     | 10009     | 14      | -1                | 14                 | 14               | 182881783  | 1
| false | false      | NORMAL | OK     |
+----------+-----------+-----------+---------+-------------------+--------------------+------------------+------------+----
--------+-------+------------+--------+--------+
```

- Display the replicas of specific partitions with a VERSION_ERROR status

```
SHOW REPLICA STATUS FROM tbl1 PARTITION (p1, p2)
WHERE STATUS = "VERSION_ERROR";
```

- Display all replicas of a table that are in unhealthy states

```
SHOW REPLICA STATUS FROM tbl1
WHERE STATUS != "OK";
```

ADMIN-REPAIR-TABLE

ADMIN-REPAIR-TABLE

Name

ADMIN REPAIR TABLE

Description

statement used to attempt to preferentially repair the specified table or partition

grammar:

```
ADMIN REPAIR TABLE table_name[ PARTITION (p1,...)]
```

illustrate:

1. This statement only means to let the system try to repair the shard copy of the specified table or partition with high priority, and does not guarantee that the repair can be successful. Users can view the repair status through the ADMIN SHOW REPLICA STATUS command.
2. The default timeout is 14400 seconds (4 hours). A timeout means that the system will no longer repair shard copies of the specified table or partition with high priority. Need to re-use this command to set

Example

1. Attempt to repair the specified table

   ```
   ADMIN REPAIR TABLE tbl1;
   ```

2. Try to repair the specified partition

   ```
   ADMIN REPAIR TABLE tbl1 PARTITION (p1, p2);
   ```

Keywords

```
ADMIN, REPAIR, TABLE
```

Best Practice

ADMIN-CANCEL-REPAIR

ADMIN-CANCEL-REPAIR

Name

ADMIN CANCEL REPAIR

Description

This statement is used to cancel the repair of the specified table or partition with high priority

grammar:

```
ADMIN CANCEL REPAIR TABLE table_name[ PARTITION (p1,...)];
```

illustrate:

1. This statement simply means that the system will no longer repair shard copies of the specified table or partition with high priority. Replicas are still repaired with the default schedule.

Example

1. Cancel high priority repair

   ```
   ADMIN CANCEL REPAIR TABLE tbl PARTITION(p1);
   ```

Keywords

```
ADMIN, CANCEL, REPAIR
```

Best Practice

ADMIN CHECK TABLET

Description

This statement is used to perform the specified check operation on a set of tablets.

Syntax

```
ADMIN CHECK TABLET ( <tablet_id> [,...] ) PROPERTIES("type" = "<type_value>")
```

Required Parameters

1. `<tablet_id>`

The ID of the tablet on which the specified check operation will be performed.

Optional Parameters

1. `<type_value>`

Currently, only consistency is supported.

- consistency:

  Checks the replica data consistency of the tablet. This command is asynchronous, meaning that after sending it, Doris will start executing the consistency check job for the corresponding tablet.

Return Value

The final result of the statement will be reflected in the InconsistentNum column of the result from `SHOW PROC "/cluster_health/tablet_health";`.

Column	DataType	Note
InconsistentNum	Int	Number of tablets with inconsistency

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

- Perform a replica data consistency check on a specified set of tablets

```
admin check tablet (10000, 10001) PROPERTIES("type" = "consistency");
```

ADMIN DIAGNOSE TABLET

ADMIN DIAGNOSE TABLET

Description

```
This statement is used to diagnose the specified tablet. The results will show information about the tablet and some potential
problems.

grammar:

    ADMIN DIAGNOSE TABLET tblet_id

illustrate:

    The lines of information in the result are as follows:
    1. TabletExist:                         Whether the Tablet exists
    2. TabletId:                            Tablet ID
    3. Database:                            The DB to which the Tablet belongs and its ID
    4. Table:                               The Table to which Tablet belongs and its ID
    5. Partition:                           The Partition to which the Tablet belongs and its ID
    6. MaterializedIndex:                   The materialized view to which the Tablet belongs and its ID
    7. Replicas(ReplicaId -> BackendId):    Tablet replicas and their BE.
    8. ReplicasNum:                         Whether the number of replicas is correct.
    9. ReplicaBackendStatus:                Whether the BE node where the replica is located is normal.
    10.ReplicaVersionStatus:                Whether the version number of the replica is normal.
    11.ReplicaStatus:                       Whether the replica status is normal.
    12.ReplicaCompactionStatus:             Whether the replica Compaction status is normal.
```

Example

```
1. Diagnose tablet 10001

    ADMIN DIAGNOSE TABLET 10001;
```

Keywords

```
ADMIN,DIAGNOSE,TABLET
```

ADMIN COPY TABLET

Description

This statement is used to make a snapshot for the specified tablet, mainly used to load the tablet locally to reproduce the problem.

Syntax

```
ADMIN COPY TABLET <tablet_id> PROPERTIES ("<key>"="<value>" [,...]).
```

Required Parameters

1. `<tablet_id>`

The ID of the tablet to be copied.

Optional Parameters

```
[ PROPERTIES ("<key>"="<value>" [, ... ]) ]
```

The PROPERTIES clause allows specifying additional parameters:

1. `<backend_id>`

Specifies the id of the BE node where the replica is located. If not specified, a replica is randomly selected.

2. `<version>`

Specifies the version of the snapshot. The version must be less than or equal to the largest version of the replica. If not specified, the largest version is used.

3. `<expiration_minutes>`

Snapshot retention time. The default is 1 hour. It will automatically clean up after a timeout. Unit minutes.

Return Value

Column	Data Type	Note
TabletId	string	The ID of the tablet for which the snapshot was created.
BackendId	string	The ID of the BE node where the snapshot is stored.

Column	Data Type	Note
Ip	string	The IP address of the BE node storing the snapshot.
Path	string	The storage path where the snapshot is saved on the BE node.
ExpirationMinutes	string	The duration (in minutes) after which the snapshot will be automatically deleted.
CreateTableStmt	string	The table creation statement for the table corresponding to the tablet. This statement is not the original table-building statement, but a simplified table-building statement for later loading the tablet locally.

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

- Take a snapshot of the replica on the specified BE node

```
ADMIN COPY TABLET 10020 PROPERTIES("backend_id" = "10003");
```

```
         TabletId: 10020
        BackendId: 10003
               Ip: 192.168.10.1
             Path: /path/to/be/storage/snapshot/20220830101353.2.3600
ExpirationMinutes: 60
  CreateTableStmt: CREATE TABLE `tbl1` (
  `k1` int(11) NULL,
  `k2` int(11) NULL
) ENGINE=OLAP
DUPLICATE KEY(`k1`, `k2`)
DISTRIBUTED BY HASH(k1) BUCKETS 1
PROPERTIES (
"replication_num" = "1",
"version_info" = "2"
);
```

- Take a snapshot of the specified version of the replica on the specified BE node

```
ADMIN COPY TABLET 10010 PROPERTIES("backend_id" = "10003", "version" = "10");
```

```
         TabletId: 10010
        BackendId: 10003
               Ip: 192.168.10.1
             Path: /path/to/be/storage/snapshot/20220830101353.2.3600
ExpirationMinutes: 60
  CreateTableStmt: CREATE TABLE `tbl1` (
  `k1` int(11) NULL,
  `k2` int(11) NULL
) ENGINE=OLAP
DUPLICATE KEY(`k1`, `k2`)
DISTRIBUTED BY HASH(k1) BUCKETS 1
PROPERTIES (
"replication_num" = "1",
"version_info" = "2"
);
```

SHOW TABLET STORAGE FORMAT

Description

This statement is used to display storage format information on the Backend.

Syntax

```
SHOW TABLET STORAGE FORMAT [VERBOSE]
```

Optional Parameters

** 1. `VERBOSE` **

Displays more detailed information.

Return Value

Column	DataType	Note
BackendId	Int	The ID of the BE (Backend) node where the tablet replica is located.
V1Count	Int	Number of V1 version tablets.
V2Count	Int	Number of V2 version tablets.
TabletId	Int	The unique identifier of the tablet.
StorageFormat	String	The version of the tablet, either V1 or V2.

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

- Execute the statement without the verbose parameter

```
show tablet storage format;
```

```
+-----------+---------+---------+
| BackendId | V1Count | V2Count |
```

```
+-----------+---------+---------+
| 10002     | 0       | 2867    |
+-----------+---------+---------+
```

- Execute the statement with the verbose parameter

```
show tablet storage format verbose;
```

```
+-----------+----------+---------------+
| BackendId | TabletId | StorageFormat |
+-----------+----------+---------------+
| 10002     | 39227    | V2            |
| 10002     | 39221    | V2            |
| 10002     | 39215    | V2            |
| 10002     | 39199    | V2            |
+-----------+----------+---------------+
```

ADMIN CLEAN TRASH

Description

This statement is used to clear garbage data in backend.

Syntax

```
ADMIN CLEAN TRASH [ON ("<be_host>:<be_heartbeat_port>" [, ...])]
```

Optional Parameters

1. `[ON ("<be_host>:<be_heartbeat_port>" [, ...])]`

Specify the backend to be cleaned up. If you do not add ON, all backend is cleared by default.

Access Control Requirements

Users executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
ADMIN_PRIV	User or Role	Only users or roles with the ADMIN_PRIV privilege can perform the CLEAN TRASH operation.

Examples

```
-- Clean up the junk data of all be nodes.
ADMIN CLEAN TRASH;
```

```
-- Clean up garbage data for '192.168.0.1:9050' and '192.168.0.2:9050'.
ADMIN CLEAN TRASH ON ("192.168.0.1:9050", "192.168.0.2:9050");
```

RECOVER

RECOVER

Name

RECOVER

Description

This statement is used to restore a previously deleted database, table or partition. It supports recover meta information by name or id, and you can set new name for recovered meta information.

You can get all meta informations that can be recovered by statement `SHOW CATALOG RECYCLE BIN`.

grammar:

1. restore database by name

```
RECOVER DATABASE db_name;
```

2. restore table by name

```
RECOVER TABLE [db_name.]table_name;
```

3. restore partition by name

```
RECOVER PARTITION partition_name FROM [db_name.]table_name;
```

4. restore database by name and id

```
RECOVER DATABASE db_name db_id;
```

5. restore table by name and id

```
RECOVER TABLE [db_name.]table_name table_id;
```

6. restore partition by name and id

```
RECOVER PARTITION partition_name partition_id FROM [db_name.]table_name;
```

7. restore database by name, and set new db name

```
RECOVER DATABASE db_name AS new_db_name;
```

8. restore table by name and id, and set new table name

```
RECOVER TABLE [db_name.]table_name table_id AS new_table_name;
```

9. restore partition by name and id, and set new partition name

```
RECOVER PARTITION partition_name partition_id AS new_partition_name FROM [db_name.]table_name;
```

illustrate:

- This operation can only restore meta information that was deleted in the previous period. Default is 1 day. (Configurable through the `catalog_trash_expire_second` parameter in fe.conf)
- If you recover a meta information by name without id, it will recover the last dropped one which has same name.
- You can get all meta informations that can be recovered by statement `SHOW CATALOG RECYCLE BIN`.

Example

1. Restore the database named example_db

```
RECOVER DATABASE example_db;
```

2. Restore the table named example_tbl

```
RECOVER TABLE example_db.example_tbl;
```

3. Restore the partition named p1 in table example_tbl

```
RECOVER PARTITION p1 FROM example_tbl;
```

4. Restore the database named example_db with id example_db_id

```
RECOVER DATABASE example_db example_db_id;
```

5. Restore the table named example_tbl with id example_tbl_id

```
RECOVER TABLE example_db.example_tbl example_tbl_id;
```

6. Restore the partition named p1 with id p1_id in table example_tbl

```
RECOVER PARTITION p1 p1_id FROM example_tbl;
```

7. Restore the database named example_db with id example_db_id, and set new name new_example_db

```
RECOVER DATABASE example_db example_db_id AS new_example_db;
```

8. Restore the table named example_tbl, and set new name new_example_tbl

```
RECOVER TABLE example_db.example_tbl AS new_example_tbl;
```

9. Restore the partition named p1 with id p1_id in table example_tbl, and new name new_p1

```
RECOVER PARTITION p1 p1_id AS new_p1 FROM example_tbl;
```

Keywords

```
RECOVER
```

Best Practice

KILL

KILL

Name

KILL

Description

Each Doris connection runs in a separate thread. You can kill a thread with the KILL processlist_id statement.

The thread process list identifier can be determined from the ID column of the INFORMATION_SCHEMA PROCESSLIST table, the Id column of the SHOW PROCESSLIST output, and the PROCESSLIST_ID column of the Performance Schema thread table.

grammar:

```
KILL [CONNECTION] processlist_id
```

In addition, you can also use processlist_id or query_id terminates the executing query command

grammar:

```
KILL QUERY processlist_id | query_id
```

Example

Keywords

```
KILL
```

Best Practice

ADMIN-REBALANCE-DISK

ADMIN-REBALANCE-DISK

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Name

ADMIN REBALANCE DISK

Description

This statement is used to try to rebalance disks of the specified backends first, no matter if the cluster is balanced

Grammar:

```
ADMIN REBALANCE DISK [ON ("BackendHost1:BackendHeartBeatPort1", "BackendHost2:BackendHeartBeatPort2", ...)];
```

Explain:

1. This statement only means that the system attempts to rebalance disks of specified backends with high priority, no matter if the cluster is balanced.
2. The default timeout is 24 hours. Timeout means that the system will no longer rebalance disks of specified backends with high priority. The command settings need to be reused.

Example

1. Attempt to rebalance disks of all backends

```
ADMIN REBALANCE DISK;
```

2. Attempt to rebalance disks oof the specified backends

```
ADMIN REBALANCE DISK ON ("192.168.1.1:1234", "192.168.1.2:1234");
```

Keywords

ADMIN,REBALANCE,DISK

Best Practice

ADMIN-CANCEL-REBALANCE-DISK

ADMIN-CANCEL-REBALANCE-DISK

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Name

ADMIN CANCEL REBALANCE DISK

Description

This statement is used to cancel rebalancing disks of specified backends with high priority

Grammar:

ADMIN CANCEL REBALANCE DISK [ON (“BackendHost1:BackendHeartBeatPort1”, “BackendHost2:BackendHeartBeatPort2”, …)];

Explain:

1. This statement only indicates that the system no longer rebalance disks of specified backends with high priority. The system will still rebalance disks by default scheduling.

Example

1. Cancel High Priority Disk Rebalance of all of backends of the cluster

ADMIN CANCEL REBALANCE DISK;

2. Cancel High Priority Disk Rebalance of specified backends

ADMIN CANCEL REBALANCE DISK ON (“192.168.1.1:1234”, “192.168.1.2:1234”);

Keywords

ADMIN,CANCEL,REBALANCE DISK

Best Practice

ENABLE FEATURE

ENABLE-FEATURE

Description

Example

Keywords

```
ENABLE, FEATURE
```

Best Practice

CREATE-CATALOG

CREATE-CATALOG

Name

CREATE CATALOG

Description

This statement is used to create an external catalog

Syntax:

```
CREATE CATALOG [IF NOT EXISTS] catalog_name [comment]
    PROPERTIES ("key"="value", ...);
```

Example

1. Create catalog hive

```
CREATE CATALOG hive comment 'hive catalog' PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://127.0.0.1:7004',
    'dfs.nameservices'='HANN',
    'dfs.ha.namenodes.HANN'='nn1,nn2',
    'dfs.namenode.rpc-address.HANN.nn1'='nn1_host:rpc_port',
    'dfs.namenode.rpc-address.HANN.nn2'='nn2_host:rpc_port',
    'dfs.client.failover.proxy.provider.HANN'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

2. Create catalog es

```
CREATE CATALOG es PROPERTIES (
    "type"="es",
    "hosts"="http://127.0.0.1:9200"
);
```

3. Create catalog jdbc

   mysql

```
CREATE CATALOG jdbc PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="123456",
    "jdbc_url" = "jdbc:mysql://127.0.0.1:3316/doris_test?useSSL=false",
    "driver_url" = "https://doris-community-test-1308700295.cos.ap-hongkong.myqcloud.com/jdbc_driver/mysql-connector-java-
8.0.25.jar",
```

```
    "driver_class" = "com.mysql.cj.jdbc.Driver"
);
```

postgresql

```
CREATE CATALOG jdbc PROPERTIES (
    "type"="jdbc",
    "user"="postgres",
    "password"="123456",
    "jdbc_url" = "jdbc:postgresql://127.0.0.1:5432/demo",
    "driver_url" = "file:///path/to/postgresql-42.5.1.jar",
    "driver_class" = "org.postgresql.Driver"
);
```

clickhouse

```
CREATE CATALOG jdbc PROPERTIES (
    "type"="jdbc",
    "user"="default",
    "password"="123456",
    "jdbc_url" = "jdbc:clickhouse://127.0.0.1:8123/demo",
    "driver_url" = "file:///path/to/clickhouse-jdbc-0.3.2-patch11-all.jar",
    "driver_class" = "com.clickhouse.jdbc.ClickHouseDriver"
)
```

oracle `sql CREATE CATALOG jdbc PROPERTIES ( "type"="jdbc", "user"="doris", "password"="123456", "jdbc_url" = "jdbc:oracle:thin:@127.0.0.1:1521:helowin", "driver_url" = "file:///path/to/ojdbc8.jar", "driver_class" = "oracle.jdbc.driver.OracleDriver" );`

SQLServer `sql CREATE CATALOG sqlserver_catalog PROPERTIES ("type"="jdbc", "user"="SA", "password"="Doris123456", "jdbc_url" = "jdbc:sqlserver://localhost:1433;DataBaseName=doris_test", "driver_url" = "file:///path/to/mssql-jdbc-11.2.3.jre8.jar", "driver_class" = "com.microsoft.sqlserver.jdbc.SQLServerDriver");`

SAP HANA `sql CREATE CATALOG saphana_catalog PROPERTIES ( "type"="jdbc", "user"="SYSTEM", "password"="SAPHANA", "jdbc_url" = "jdbc:sap://localhost:31515/TEST", "driver_url" = "file:///path/to/ngdbc.jar", "driver_class" = "com.sap.db.jdbc.Driver" );`

Trino `sql CREATE CATALOG trino_catalog PROPERTIES ( "type"="jdbc", "user"="hadoop", "password"="", "jdbc_url" = "jdbc:trino://localhost:8080/hive", "driver_url" = "file:///path/to/trino-jdbc-389.jar", "driver_class" = "io.trino.jdbc.TrinoDriver" );`

OceanBase `sql CREATE CATALOG oceanbase_catalog PROPERTIES ( "type"="jdbc", "user"="root", "password"="", "jdbc_url" = "jdbc:oceanbase://localhost:2881/demo", "driver_url" = "file:///path/to/oceanbase-client-2.4.2.jar", "driver_class" = "com.oceanbase.jdbc.Driver" );`

Keywords

CREATE, CATALOG

Best Practice

CREATE-DATABASE

CREATE-DATABASE

Name

CREATE DATABASE

Description

This statement is used to create a new database (database)

grammar:

```
CREATE DATABASE [IF NOT EXISTS] db_name
    [PROPERTIES ("key"="value", ...)];
```

`PROPERTIES` Additional information about the database, which can be defaulted.

- If you create an Iceberg database, you need to provide the following information in properties:

  ```
  PROPERTIES (
    "iceberg.database" = "iceberg_db_name",
    "iceberg.hive.metastore.uris" = "thrift://127.0.0.1:9083",
    "iceberg.catalog.type" = "HIVE_CATALOG"
  )
  ```

  illustrate:

  - `ceberg.database` : the library name corresponding to Iceberg;
  - `iceberg.hive.metastore.uris` : hive metastore service address;
  - `iceberg.catalog.type`: The default is `HIVE_CATALOG`; currently only `HIVE_CATALOG` is supported, and more Iceberg catalog types will be supported in the future.

- If you want to specify the default replica distribution for tables in db, you need to specify `replication_allocation` (the `replication_allocation` attribute of table will have higher priority than db)

  ```
  PROPERTIES (
    "replication_allocation" = "tag.location.default:3"
  )
  ```

Example

1. Create a new database db_test

   ```
   CREATE DATABASE db_test;
   ```

2. Create a new Iceberg database iceberg_test

```
CREATE DATABASE `iceberg_test`
PROPERTIES (
"iceberg.database" = "doris",
"iceberg.hive.metastore.uris" = "thrift://127.0.0.1:9083",
"iceberg.catalog.type" = "HIVE_CATALOG"
);
```

Keywords

```
CREATE, DATABASE
```

Best Practice

CREATE-TABLE

CREATE-TABLE

Description

This command is used to create a table. The subject of this document describes the syntax for creating Doris self-maintained tables. For external table syntax, please refer to the CREATE-EXTERNAL-TABLE document.

```
CREATE TABLE [IF NOT EXISTS] [database.]table
(
    column_definition_list
    [, index_definition_list]
)
[engine_type]
[keys_type]
[table_comment]
[partition_info]
distribution_desc
[rollup_list]
[properties]
[extra_properties]
```

column_definition_list

Column definition list:

`column_definition[, column_definition]`

- `column_definition`

  Column definition:

  `column_name column_type [KEY] [aggr_type] [NULL] [default_value] [column_comment]`

  - `column_type`

    Column type, the following types are supported:

    ```
    TINYINT (1 byte)
        Range: -2^7 + 1 ~ 2^7-1
    SMALLINT (2 bytes)
        Range: -2^15 + 1 ~ 2^15-1
    INT (4 bytes)
        Range: -2^31 + 1 ~ 2^31-1
    BIGINT (8 bytes)
        Range: -2^63 + 1 ~ 2^63-1
    LARGEINT (16 bytes)
        Range: -2^127 + 1 ~ 2^127-1
    FLOAT (4 bytes)
        Support scientific notation
    DOUBLE (12 bytes)
        Support scientific notation
    ```

```
DECIMAL[(precision, scale)] (16 bytes)
    The decimal type with guaranteed precision. The default is DECIMAL(10, 0)
    precision: 1 ~ 27
    scale: 0 ~ 9
    Where the integer part is 1 ~ 18
    Does not support scientific notation
DATE (3 bytes)
    Range: 0000-01-01 ~ 9999-12-31
DATETIME (8 bytes)
    Range: 0000-01-01 00:00:00 ~ 9999-12-31 23:59:59
CHAR[(length)]
    Fixed-length character string. Length range: 1 ~ 255. Default is 1
VARCHAR[(length)]
    Variable length character string. Length range: 1 ~ 65533. Default is 65533
HLL (1~16385 bytes)
    HyperLogLog column type, do not need to specify the length and default value. The length is controlled within
the system according to the degree of data aggregation.
    Must be used with HLL_UNION aggregation type.
BITMAP
    The bitmap column type does not need to specify the length and default value. Represents a collection of
integers, and the maximum number of elements supported is 2^64-1.
    Must be used with BITMAP_UNION aggregation type.
```

- `aggr_type`

  Aggregation type, the following aggregation types are supported:

```
SUM: Sum. Applicable numeric types.
MIN: Find the minimum value. Suitable for numeric types.
MAX: Find the maximum value. Suitable for numeric types.
REPLACE: Replace. For rows with the same dimension column, the index column will be imported in the order of
import, and the last imported will replace the first imported.
REPLACE_IF_NOT_NULL: non-null value replacement. The difference with REPLACE is that there is no replacement for
null values. It should be noted here that the default value should be NULL, not an empty string. If it is an empty
string, you should replace it with an empty string.
HLL_UNION: The aggregation method of HLL type columns, aggregated by HyperLogLog algorithm.
BITMAP_UNION: The aggregation mode of BIMTAP type columns, which performs the union aggregation of bitmaps.
```

- `default_value`

  Default value of the column. If the load data does not specify a value for this column, the system will assign a default value to this column.

  The syntax is: `default default_value`。

  Currently, the default value supports two forms:

  1. The user specifies a fixed value, such as:

```
    k1 INT DEFAULT '1',
    k2 CHAR(10) DEFAULT 'aaaa'
```

  2. Keywords are provided by the system. Currently, the following keywords are supported:

```
    // This keyword is used only for DATETIME type. If the value is missing, the system assigns the current
timestamp.
    dt DATETIME DEFAULT CURRENT_TIMESTAMP
```

Example:

```
```
k1 TINYINT,
k2 DECIMAL(10,2) DEFAULT "10.5",
k4 BIGINT NULL DEFAULT "1000" COMMENT "This is column k4",
v1 VARCHAR(10) REPLACE NOT NULL,
v2 BITMAP BITMAP_UNION,
v3 HLL HLL_UNION,
v4 INT SUM NOT NULL DEFAULT "1" COMMENT "This is column v4"
```
```

index_definition_list

Index list definition:

`index_definition[, index_definition]`

- `index_definition`

  Index definition:

  ```
  INDEX index_name (col_name) [USING INVERTED] COMMENT'xxxxxx'
  ```

  Example:

  ```
  INDEX idx1 (k1) USING INVERTED COMMENT "This is a inverted index1",
  INDEX idx2 (k2) USING INVERTED COMMENT "This is a inverted index2",
  ...
  ```

engine_type

Table engine type. All types in this document are OLAP. For other external table engine types, see CREATE EXTERNAL TABLE document. Example:

```
`ENGINE=olap`
```

keys_type

Data model.

`key_type(col1, col2, ...)`

`key_type` supports the following models:

- DUPLICATE KEY (default): The subsequent specified column is the sorting column.
- AGGREGATE KEY: The specified column is the dimension column.
- UNIQUE KEY: The subsequent specified column is the primary key column.

NOTE: when set table property `"enable_duplicate_without_keys_by_default" = "true"`, will create a duplicate model without sorting columns and prefix indexes by default.

Example:

```
DUPLICATE KEY(col1, col2),
AGGREGATE KEY(k1, k2, k3),
UNIQUE KEY(k1, k2)
```

table_comment

Table notes. Example:

```
```
COMMENT "This is my first DORIS table"
```
```

partition_info

Partition information supports three writing methods:

1. LESS THAN: Only define the upper boundary of the partition. The lower bound is determined by the upper bound of the previous partition.

   ```
   PARTITION BY RANGE(col1[, col2, ...])
   (
       PARTITION partition_name1 VALUES LESS THAN MAXVALUE|("value1", "value2", ...),
       PARTITION partition_name2 VALUES LESS THAN MAXVALUE|("value1", "value2", ...)
   )
   ```

2. FIXED RANGE: Define the left closed and right open interval of the zone.

   ```
   PARTITION BY RANGE(col1[, col2, ...])
   (
       PARTITION partition_name1 VALUES [("k1-lower1", "k2-lower1", "k3-lower1",...), ("k1-upper1", "k2-upper1", "k3-upper1",
   ... )),
       PARTITION partition_name2 VALUES [("k1-lower1-2", "k2-lower1-2", ...), ("k1-upper1-2", MAXVALUE, ))
   )
   ```

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

3. MULTI RANGE: Multi build RANGE partitions,Define the left closed and right open interval of the zone, Set the time unit and step size, the time unit supports year, month, day, week and hour.

   ```
   PARTITION BY RANGE(col)
   (
      FROM ("2000-11-14") TO ("2021-11-14") INTERVAL 1 YEAR,
   ```

```
    FROM ("2021-11-14") TO ("2022-11-14") INTERVAL 1 MONTH,
    FROM ("2022-11-14") TO ("2023-01-03") INTERVAL 1 WEEK,
    FROM ("2023-01-03") TO ("2023-01-14") INTERVAL 1 DAY
)
```

4. MULTI RANGE: Multi build integer RANGE partitions,Define the left closed and right open interval of the zone, and step size.

```
PARTITION BY RANGE(int_col)
(
    FROM (1) TO (100) INTERVAL 10
)
```

distribution_desc

Define the data bucketing method.

1. Hash Syntax: `DISTRIBUTED BY HASH (k1[,k2 ...]) [BUCKETS num|auto]` Explain: Hash bucketing using the specified key column.
2. Random Syntax: `DISTRIBUTED BY RANDOM [BUCKETS num|auto]` Explain: Use random numbers for bucketing.

rollup_list

Multiple materialized views (ROLLUP) can be created at the same time as the table is built.

`ROLLUP (rollup_definition[, rollup_definition, ...])`

- `rollup_definition`

  `rollup_name (col1[, col2, ...]) [DUPLICATE KEY(col1[, col2, ...])] [PROPERTIES("key" = "value")]`

  Example:

```
ROLLUP (
    r1 (k1, k3, v1, v2),
    r2 (k1, v1)
)
```

properties

Set table properties. The following attributes are currently supported:

- `replication_num`

  Number of copies. The default number of copies is 3. If the number of BE nodes is less than 3, you need to specify that the number of copies is less than or equal to the number of BE nodes.

  After version 0.15, this attribute will be automatically converted to the `replication_allocation` attribute, such as:

`"replication_num" = "3"` will be automatically converted to `"replication_allocation" = "tag.location.default:3"`

- `replication_allocation`

  Set the copy distribution according to Tag. This attribute can completely cover the function of the `replication_num` attribute.

- `is_being_synced`

  Used to identify whether this table is copied by CCR and is being synchronized by syncer. The default is `false`.

  If set to `true`:
  `colocate_with`, `storage_policy` properties will be erased
  `dynamic partition`, `auto bucket` features will be disabled, that is, they will be displayed as enabled in `show create table`, but will not actually take effect. When `is_being_synced` is set to `false`, these features will resume working.

  This property is for CCR peripheral modules only and should not be manually set during CCR synchronization.

- `storage_medium/storage_cooldown_time`

  Data storage medium. `storage_medium` is used to declare the initial storage medium of the table data, and `storage_cooldown_time` is used to set the expiration time. Example:

  ```
  "storage_medium" = "SSD",
  "storage_cooldown_time" = "2020-11-20 00:00:00"
  ```

  This example indicates that the data is stored in the SSD and will be automatically migrated to the HDD storage after the expiration of 2020-11-20 00:00:00.

- `colocate_with`

  When you need to use the Colocation Join function, use this parameter to set the Colocation Group.

  `"colocate_with" = "group1"`

- `bloom_filter_columns`

  The user specifies the list of column names that need to be added to the Bloom Filter index. The Bloom Filter index of each column is independent, not a composite index.

  `"bloom_filter_columns" = "k1, k2, k3"`

- `in_memory`

  Deprecated.

- `function_column.sequence_col`

When using the UNIQUE KEY model, you can specify a sequence column. When the KEY columns are the same, REPLACE will be performed according to the sequence column (the larger value replaces the smaller value, otherwise it cannot be replaced)

The `function_column.sequence_col` is used to specify the mapping of the sequence column to a column in the table, which can be integral and time (DATE, DATETIME). The type of this column cannot be changed after creation. If `function_column.sequence_col` is set, `function_column.sequence_type` is ignored.

```
"function_column.sequence_col" ='column_name'
```

- `function_column.sequence_type`

  When using the UNIQUE KEY model, you can specify a sequence column. When the KEY columns are the same, REPLACE will be performed according to the sequence column (the larger value replaces the smaller value, otherwise it cannot be replaced)

  Here we only need to specify the type of sequence column, support time type or integer type. Doris will create a hidden sequence column.

  ```
  "function_column.sequence_type" ='Date'
  ```

- `compression`

  The default compression method for Doris tables is LZ4. After version 1.1, it is supported to specify the compression method as ZSTD to obtain a higher compression ratio.

  ```
  "compression"="zstd"
  ```

- `light_schema_change`

  Whether to use the Light Schema Change optimization.

  If set to true, the addition and deletion of value columns can be done more quickly and synchronously.

  ```
  "light_schema_change"="true"
  ```

  This feature is enabled by default after v2.0.0.

- `disable_auto_compaction`

  Whether to disable automatic compaction for this table.

  If this property is set to ‘true’, the background automatic compaction process will skip all the tables of this table.

  ```
  "disable_auto_compaction" = "false"
  ```

- `enable_single_replica_compaction`

  Whether to enable single replica compaction for this table.

  If this property is set to ‘true’, all replicas of the tablet will only have one replica performing compaction, while the others fetch rowsets from that replica.

  ```
  "enable_single_replica_compaction" = "false"
  ```

- `enable_duplicate_without_keys_by_default`

  When `true`, if Unique, Aggregate, or Duplicate is not specified when creating a table, a Duplicate model table without sorting columns and prefix indexes will be created by default.

  `"enable_duplicate_without_keys_by_default" = "false"`

- `skip_write_index_on_load`

  Whether to enable skip inverted index on load for this table.

  If this property is set to 'true', skip writting index (only inverted index now) on first time load and delay writting index to compaction. It can reduce CPU and IO resource usage for high throughput load.

  `"skip_write_index_on_load" = "false"`

- `compaction_policy`

  Configure the compaction strategy in the compression phase. Only support configuring the compaction policy as "time_series" or "size_based".

  time_series: When the disk size of a rowset accumulates to a certain threshold, version merging takes place. The merged rowset is directly promoted to the base compaction stage. This approach effectively reduces the write amplification rate of compaction, especially in scenarios with continuous imports in a time series context.

  In the case of time series compaction, the execution of compaction is adjusted using parameters that have the prefix time_series_compaction.

  `"compaction_policy" = ""`

- `time_series_compaction_goal_size_mbytes`

  Time series compaction policy will utilize this parameter to adjust the size of input files for each compaction. The output file size will be approximately equal to the input file size.

  `"time_series_compaction_goal_size_mbytes" = "1024"`

- `time_series_compaction_file_count_threshold`

  Time series compaction policy will utilize this parameter to adjust the minimum number of input files for each compaction.

  If the number of files in a tablet exceeds the configured threshold, it will trigger a compaction process.

  `"time_series_compaction_file_count_threshold" = "2000"`

- `time_series_compaction_time_threshold_seconds`

  When time series compaction policy is applied, a significant duration passes without a compaction being executed, a compaction will be triggered.

  `"time_series_compaction_time_threshold_seconds" = "3600"`

- `time_series_compaction_level_threshold`

When time series compaction policy is applied, This parameter defaults to 1. When set to 2, it is used to control the re-merging of segments that have been

merged once, ensuring that the segment size reaches the time_series_compaction_goal_size_mbytes, which can achieve the effect of reducing the number of

segments.

`"time_series_compaction_level_threshold" = "2"`

- Dynamic partition related

  The relevant parameters of dynamic partition are as follows:

- `dynamic_partition.enable`: Used to specify whether the dynamic partition function at the table level is enabled. The default is true.
- `dynamic_partition.time_unit`: is used to specify the time unit for dynamically adding partitions, which can be selected as DAY (day), WEEK (week), MONTH (month), YEAR (year), HOUR (hour).
- `dynamic_partition.start`: Used to specify how many partitions to delete forward. The value must be less than 0. The default is Integer.MIN_VALUE.
- `dynamic_partition.end`: Used to specify the number of partitions created in advance. The value must be greater than 0.
- `dynamic_partition.prefix`: Used to specify the partition name prefix to be created. For example, if the partition name prefix is p, the partition name will be automatically created as p20200108.
- `dynamic_partition.buckets`: Used to specify the number of partition buckets that are automatically created.
- `dynamic_partition.create_history_partition`: Whether to create a history partition.
- `dynamic_partition.history_partition_num`: Specify the number of historical partitions to be created.
- `dynamic_partition.reserved_history_periods`: Used to specify the range of reserved history periods.

Example

1. Create a detailed model table

```
CREATE TABLE example_db.table_hash
(
    k1 TINYINT,
    k2 DECIMAL(10, 2) DEFAULT "10.5",
    k3 CHAR(10) COMMENT "string column",
    k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
)
COMMENT "my first table"
DISTRIBUTED BY HASH(k1) BUCKETS 32
```

2. Create a detailed model table, partition, specify the sorting column, and set the number of copies to 1

```
CREATE TABLE example_db.table_hash
(
    k1 DATE,
    k2 DECIMAL(10, 2) DEFAULT "10.5",
    k3 CHAR(10) COMMENT "string column",
    k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
)
DUPLICATE KEY(k1, k2)
COMMENT "my first table"
PARTITION BY RANGE(k1)
(
    PARTITION p1 VALUES LESS THAN ("2020-02-01"),
    PARTITION p2 VALUES LESS THAN ("2020-03-01"),
    PARTITION p3 VALUES LESS THAN ("2020-04-01")
)
DISTRIBUTED BY HASH(k1) BUCKETS 32
PROPERTIES (
    "replication_num" = "1"
);
```

3. Create a table with a unique model of the primary key, set the initial storage medium and cooling time

```
CREATE TABLE example_db.table_hash
(
    k1 BIGINT,
    k2 LARGEINT,
    v1 VARCHAR(2048),
    v2 SMALLINT DEFAULT "10"
)
UNIQUE KEY(k1, k2)
DISTRIBUTED BY HASH (k1, k2) BUCKETS 32
PROPERTIES(
    "storage_medium" = "SSD",
    "storage_cooldown_time" = "2015-06-04 00:00:00"
);
```

4. Create an aggregate model table, using a fixed range partition description

```
CREATE TABLE table_range
(
    k1 DATE,
    k2 INT,
    k3 SMALLINT,
    v1 VARCHAR(2048) REPLACE,
    v2 INT SUM DEFAULT "1"
)
AGGREGATE KEY(k1, k2, k3)
PARTITION BY RANGE (k1, k2, k3)
(
    PARTITION p1 VALUES [("2014-01-01", "10", "200"), ("2014-01-01", "20", "300")),
    PARTITION p2 VALUES [("2014-06-01", "100", "200"), ("2014-07-01", "100", "300"))
)
DISTRIBUTED BY HASH(k2) BUCKETS 32
```

5. Create an aggregate model table with HLL and BITMAP column types

```
CREATE TABLE example_db.example_table
(
    k1 TINYINT,
```

```
    k2 DECIMAL(10, 2) DEFAULT "10.5",
    v1 HLL HLL_UNION,
    v2 BITMAP BITMAP_UNION
)
ENGINE=olap
AGGREGATE KEY(k1, k2)
DISTRIBUTED BY HASH(k1) BUCKETS 32
```

6. Create two self-maintained tables of the same Colocation Group.

```
CREATE TABLE t1 (
    id int(11) COMMENT "",
    value varchar(8) COMMENT ""
)
DUPLICATE KEY(id)
DISTRIBUTED BY HASH(id) BUCKETS 10
PROPERTIES (
    "colocate_with" = "group1"
);

CREATE TABLE t2 (
    id int(11) COMMENT "",
    value1 varchar(8) COMMENT "",
    value2 varchar(8) COMMENT ""
)
DUPLICATE KEY(`id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 10
PROPERTIES (
    "colocate_with" = "group1"
);
```

7. Create a table with inverted index and bloom filter index

```
CREATE TABLE example_db.table_hash
(
    k1 TINYINT,
    k2 DECIMAL(10, 2) DEFAULT "10.5",
    v1 CHAR(10) REPLACE,
    v2 INT SUM,
    INDEX k1_idx (k1) USING INVERTED COMMENT'my first index'
)
AGGREGATE KEY(k1, k2)
DISTRIBUTED BY HASH(k1) BUCKETS 32
PROPERTIES (
    "bloom_filter_columns" = "k2"
);
```

8. Create a dynamic partition table.

   The table creates partitions 3 days in advance every day, and deletes the partitions 3 days ago. For example, if today is `2020-01-08`, partitions named `p20200108`, `p20200109`, `p20200110`, `p20200111` will be created. The partition ranges are:

```
[types: [DATE]; keys: [2020-01-08]; ··types: [DATE]; keys: [2020-01-09];)
[types: [DATE]; keys: [2020-01-09]; ··types: [DATE]; keys: [2020-01-10];)
[types: [DATE]; keys: [2020-01-10]; ··types: [DATE]; keys: [2020-01-11];)
[types: [DATE]; keys: [2020-01-11]; ··types: [DATE]; keys: [2020-01-12];)
```

```
CREATE TABLE example_db.dynamic_partition
(
    k1 DATE,
    k2 INT,
    k3 SMALLINT,
    v1 VARCHAR(2048),
    v2 DATETIME DEFAULT "2014-02-04 15:36:00"
)
DUPLICATE KEY(k1, k2, k3)
PARTITION BY RANGE (k1) ()
DISTRIBUTED BY HASH(k2) BUCKETS 32
PROPERTIES(
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.start" = "-3",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "32"
);
```

9. Create a table with a materialized view (ROLLUP).

```
CREATE TABLE example_db.rolup_index_table
(
    event_day DATE,
    siteid INT DEFAULT '10',
    citycode SMALLINT,
    username VARCHAR(32) DEFAULT'',
    pv BIGINT SUM DEFAULT '0'
)
AGGREGATE KEY(event_day, siteid, citycode, username)
DISTRIBUTED BY HASH(siteid) BUCKETS 10
ROLLUP (
    r1(event_day,siteid),
    r2(event_day,citycode),
    r3(event_day)
)
PROPERTIES("replication_num" = "3");
```

10. Set the replica of the table through the `replication_allocation` property.

```
CREATE TABLE example_db.table_hash
(
  k1 TINYINT,
    k2 DECIMAL(10, 2) DEFAULT "10.5"
)
DISTRIBUTED BY HASH(k1) BUCKETS 32
PROPERTIES (
    "replication_allocation"="tag.location.group_a:1, tag.location.group_b:2"
);
```

```
CREATE TABLE example_db.dynamic_partition
(
    k1 DATE,
    k2 INT,
    k3 SMALLINT,
    v1 VARCHAR(2048),
    v2 DATETIME DEFAULT "2014-02-04 15:36:00"
)
PARTITION BY RANGE (k1) ()
DISTRIBUTED BY HASH(k2) BUCKETS 32
```

```
PROPERTIES(
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.start" = "-3",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "32",
    "dynamic_partition.replication_allocation" = "tag.location.group_a:3"
 );
```

11. Set the table hot and cold separation policy through the `storage_policy` property.

```
    CREATE TABLE IF NOT EXISTS create_table_use_created_policy
    (
        k1 BIGINT,
        k2 LARGEINT,
        v1 VARCHAR(2048)
    )
    UNIQUE KEY(k1)
    DISTRIBUTED BY HASH (k1) BUCKETS 3
    PROPERTIES(
        "storage_policy" = "test_create_table_use_policy",
        "replication_num" = "1"
    );
```

NOTE: Need to create the s3 resource and storage policy before the table can be successfully associated with the migration policy

12. Add a hot and cold data migration strategy for the table partition

```
    CREATE TABLE create_table_partion_use_created_policy
    (
        k1 DATE,
        k2 INT,
        V1 VARCHAR(2048) REPLACE
    ) PARTITION BY RANGE (k1) (
        PARTITION p1 VALUES LESS THAN ("2022-01-01") ("storage_policy" = "test_create_table_partition_use_policy_1"
,"replication_num"="1"),
        PARTITION p2 VALUES LESS THAN ("2022-02-01") ("storage_policy" = "test_create_table_partition_use_policy_2"
,"replication_num"="1")
    ) DISTRIBUTED BY HASH(k2) BUCKETS 1;
```

NOTE: Need to create the s3 resource and storage policy before the table can be successfully associated with the migration policy

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

13. Multi Partition by a partition desc

```
    CREATE TABLE create_table_multi_partion_date
    (
        k1 DATE,
        k2 INT,
        V1 VARCHAR(20)
```

```
    ) PARTITION BY RANGE (k1) (
        FROM ("2000-11-14") TO ("2021-11-14") INTERVAL 1 YEAR,
        FROM ("2021-11-14") TO ("2022-11-14") INTERVAL 1 MONTH,
        FROM ("2022-11-14") TO ("2023-01-03") INTERVAL 1 WEEK,
        FROM ("2023-01-03") TO ("2023-01-14") INTERVAL 1 DAY,
        PARTITION p_20230114 VALUES [('2023-01-14'), ('2023-01-15'))
    ) DISTRIBUTED BY HASH(k2) BUCKETS 1
    PROPERTIES(
        "replication_num" = "1"
    );
```

```
    CREATE TABLE create_table_multi_partion_date_hour
    (
        k1 DATETIME,
        k2 INT,
        V1 VARCHAR(20)
    ) PARTITION BY RANGE (k1) (
        FROM ("2023-01-03 12") TO ("2023-01-14 22") INTERVAL 1 HOUR
    ) DISTRIBUTED BY HASH(k2) BUCKETS 1
    PROPERTIES(
        "replication_num" = "1"
    );
```

```
    CREATE TABLE create_table_multi_partion_integer
    (
        k1 BIGINT,
        k2 INT,
        V1 VARCHAR(20)
    ) PARTITION BY RANGE (k1) (
        FROM (1) TO (100) INTERVAL 10
    ) DISTRIBUTED BY HASH(k2) BUCKETS 1
    PROPERTIES(
        "replication_num" = "1"
    );
```

NOTE: Multi Partition can be mixed with conventional manual creation of partitions. When using, you need to limit the partition column to only one, The default maximum number of partitions created in multi partition is 4096, This parameter can be adjusted in fe configuration `max_multi_partition_num`.

1. Add a duplicate without sorting column table

```
CREATE TABLE example_db.table_hash
(
    k1 DATE,
    k2 DECIMAL(10, 2) DEFAULT "10.5",
    k3 CHAR(10) COMMENT "string column",
    k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
)
COMMENT "duplicate without keys"
PARTITION BY RANGE(k1)
(
    PARTITION p1 VALUES LESS THAN ("2020-02-01"),
    PARTITION p2 VALUES LESS THAN ("2020-03-01"),
    PARTITION p3 VALUES LESS THAN ("2020-04-01")
)
DISTRIBUTED BY HASH(k1) BUCKETS 32
PROPERTIES (
    "replication_num" = "1",
```

```
    "enable_duplicate_without_keys_by_default" = "true"
);
```

Keywords

```
CREATE, TABLE
```

Best Practice

Partitioning and bucketing

A table must specify the bucket column, but it does not need to specify the partition. For the specific introduction of partitioning and bucketing, please refer to the Data Division document.

Tables in Doris can be divided into partitioned tables and non-partitioned tables. This attribute is determined when the table is created and cannot be changed afterwards. That is, for partitioned tables, you can add or delete partitions in the subsequent use process, and for non-partitioned tables, you can no longer perform operations such as adding partitions afterwards.

At the same time, partitioning columns and bucketing columns cannot be changed after the table is created. You can neither change the types of partitioning and bucketing columns, nor do any additions or deletions to these columns.

Therefore, it is recommended to confirm the usage method to build the table reasonably before building the table.

Dynamic Partition

The dynamic partition function is mainly used to help users automatically manage partitions. By setting certain rules, the Doris system regularly adds new partitions or deletes historical partitions. Please refer to Dynamic Partition document for more help.

Materialized View

Users can create multiple materialized views (ROLLUP) while building a table. Materialized views can also be added after the table is built. It is convenient for users to create all materialized views at one time by writing in the table creation statement.

If the materialized view is created when the table is created, all subsequent data import operations will synchronize the data of the materialized view to be generated. The number of materialized views may affect the efficiency of data import.

If you add a materialized view in the subsequent use process, if there is data in the table, the creation time of the materialized view depends on the current amount of data.

For the introduction of materialized views, please refer to the document materialized views.

Index

Users can create indexes on multiple columns while building a table. Indexes can also be added after the table is built.

If you add an index in the subsequent use process, if there is data in the table, you need to rewrite all the data, so the creation time of the index depends on the current data volume.

CREATE-TABLE-LIKE

CREATE-TABLE-LIKE

Name

CREATE TABLE LIKE

Description

This statement is used to create an empty table with the exact same table structure as another table, and can optionally replicate some rollups.

grammar:

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [database.]table_name LIKE [database.]table_name [WITH ROLLUP (r1,r2,r3,...)]
```

illustrate:

- The copied table structure includes Column Definition, Partitions, Table Properties, etc.
- The user needs to have `SELECT` permission on the copied original table
- Support for copying external tables such as MySQL
- Support the rollup of copying OLAP Table

Example

1. Create an empty table with the same table structure as table1 under the test1 library, the table name is table2

   ```
   CREATE TABLE test1.table2 LIKE test1.table1
   ```

2. Create an empty table with the same table structure as test1.table1 under the test2 library, the table name is table2

   ```
   CREATE TABLE test2.table2 LIKE test1.table1
   ```

3. Create an empty table with the same table structure as table1 under the test1 library, the table name is table2, and copy the two rollups of r1 and r2 of table1 at the same time

   ```
   CREATE TABLE test1.table2 LIKE test1.table1 WITH ROLLUP (r1,r2)
   ```

4. Create an empty table with the same table structure as table1 under the test1 library, the table name is table2, and copy all the rollups of table1 at the same time

```
CREATE TABLE test1.table2 LIKE test1.table1 WITH ROLLUP
```

5. Create an empty table with the same table structure as test1.table1 under the test2 library, the table name is table2, and copy the two rollups of r1 and r2 of table1 at the same time

```
CREATE TABLE test2.table2 LIKE test1.table1 WITH ROLLUP (r1,r2)
```

6. Create an empty table with the same table structure as test1.table1 under the test2 library, the table name is table2, and copy all the rollups of table1 at the same time

```
CREATE TABLE test2.table2 LIKE test1.table1 WITH ROLLUP
```

7. Create an empty table under the test1 library with the same table structure as the MySQL outer table1, the table name is table2

```
CREATE TABLE test1.table2 LIKE test1.table1
```

Keywords

```
CREATE, TABLE, LIKE
```

Best Practice

CREATE-TABLE-AS-SELECT

CREATE-TABLE-AS-SELECT

Name

CREATE TABLE AS SELECT

Description

This statement creates the table structure by returning the results from the Select statement and imports the data at the same time

grammar:

```
CREATE TABLE table_name [( column_name_list )]
    opt_engine:engineName
    opt_keys:keys
    opt_comment:tableComment
    opt_partition:partition
    opt_distribution:distribution
    opt_rollup:index
    opt_properties:tblProperties
    opt_ext_properties:extProperties
    KW_AS query_stmt:query_def
```

illustrate:

- The user needs to have`SELECT`permission for the source table and`CREATE`permission for the target database
- After a table is created, data is imported. If the import fails, the table is deleted
- You can specify the key type. The default key type is `Duplicate Key`

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

- All columns of type string (varchar/var/string) are created as type “string”.
- If the created source is an external table and the first column is of type String, the first column is automatically set to VARCHAR(65533). Because of Doris internal table, String column is not allowed as first column.

Example

1. Using the field names in the SELECT statement

```
create table `test`.`select_varchar`
PROPERTIES("replication_num" = "1")
as select * from `test`.`varchar_table`
```

2. Custom field names (need to match the number of fields returned) `sql create table `test`.`select_name`(user, testname, userstatus) PROPERTIES("replication_num" = "1") as select vt.userId, vt.username, jt.status from `test`.`varchar_table` vt join `test`.`join_table` jt on vt.userId=jt.userId`
3. Specify table model, partitions, and buckets `sql CREATE TABLE t_user(dt, id, name) ENGINE=OLAP UNIQUE KEY(dt, id) COMMENT "OLAP" PARTITION BY RANGE(dt) ( FROM ("2020-01-01") TO ("2021-12-31") INTERVAL 1 YEAR ) DISTRIBUTED BY HASH(id) BUCKETS 1 PROPERTIES("replication_num"="1") AS SELECT cast('2020-05-20' as date) as dt, 1 as id, 'Tom' as name;`

Keywords

```
CREATE, TABLE, AS, SELECT
```

Best Practice

CREATE-EXTERNAL-TABLE

CREATE-EXTERNAL-TABLE

Name

CREATE EXTERNAL TABLE

Description

This statement is used to create an external table, see CREATE TABLE for the specific syntax.

Which type of external table is mainly identified by the ENGINE type, currently MYSQL, BROKER, HIVE, ICEBERG, HUDI are optional

1. If it is mysql, you need to provide the following information in properties:

```
PROPERTIES (
"host" = "mysql_server_host",
"port" = "mysql_server_port",
"user" = "your_user_name",
"password" = "your_password",
"database" = "database_name",
"table" = "table_name"
)
```

   and there is an optional propertiy “charset” which can set character fom mysql connection, default value is “utf8”. You can set another value “utf8mb4” instead of “utf8” when you need.

   Notice:

   - “table_name” in “table” entry is the real table name in mysql. The table_name in the CREATE TABLE statement is the name of the mysql table in Doris, which can be different.
   - The purpose of creating a mysql table in Doris is to access the mysql database through Doris. Doris itself does not maintain or store any mysql data.

2. If it is a broker, it means that the access to the table needs to pass through the specified broker, and the following information needs to be provided in properties:

```
PROPERTIES (
"broker_name" = "broker_name",
"path" = "file_path1[,file_path2]",
"column_separator" = "value_separator"
"line_delimiter" = "value_delimiter"
)
```

   In addition, you need to provide the Property information required by the Broker, and pass it through the BROKER PROPERTIES, for example, HDFS needs to pass in

```
BROKER PROPERTIES(
  "username" = "name",
  "password" = "password"
)
```

According to different Broker types, the content that needs to be passed in is also different.

Notice:

- If there are multiple files in “path”, separate them with comma [,]. If the filename contains a comma, use %2c instead. If the filename contains %, use %25 instead
- Now the file content format supports CSV, and supports GZ, BZ2, LZ4, LZO (LZOP) compression formats.

3. If it is hive, you need to provide the following information in properties:

```
PROPERTIES (
"database" = "hive_db_name",
"table" = "hive_table_name",
"hive.metastore.uris" = "thrift://127.0.0.1:9083"
)
```

Where database is the name of the library corresponding to the hive table, table is the name of the hive table, and hive.metastore.uris is the address of the hive metastore service.

4. In case of iceberg, you need to provide the following information in properties:

```
PROPERTIES (
"iceberg.database" = "iceberg_db_name",
"iceberg.table" = "iceberg_table_name",
"iceberg.hive.metastore.uris" = "thrift://127.0.0.1:9083",
"iceberg.catalog.type" = "HIVE_CATALOG"
)
```

Where database is the library name corresponding to Iceberg; table is the corresponding table name in Iceberg; hive.metastore.uris is the hive metastore service address; catalog.type defaults to HIVE_CATALOG. Currently only HIVE_CATALOG is supported, more Iceberg catalog types will be supported in the future.

5. In case of hudi, you need to provide the following information in properties:

```
PROPERTIES (
"hudi.database" = "hudi_db_in_hive_metastore",
"hudi.table" = "hudi_table_in_hive_metastore",
"hudi.hive.metastore.uris" = "thrift://127.0.0.1:9083"
)
```

Where hudi.database is the corresponding database name in HiveMetaStore; hudi.table is the corresponding table name in HiveMetaStore; hive.metastore.uris is the hive metastore service address;

Example

1. Create a MYSQL external table

   Create mysql table directly from outer table information

```
CREATE EXTERNAL TABLE example_db.table_mysql
(
k1 DATE,
k2 INT,
k3 SMALLINT,
k4 VARCHAR(2048),
k5 DATETIME
)
ENGINE=mysql
PROPERTIES
(
"host" = "127.0.0.1",
"port" = "8239",
"user" = "mysql_user",
"password" = "mysql_passwd",
"database" = "mysql_db_test",
"table" = "mysql_table_test",
"charset" = "utf8mb4"
)
```

   Create mysql table through External Catalog Resource

```
# Create Resource first
CREATE EXTERNAL RESOURCE "mysql_resource"
PROPERTIES
(
  "type" = "odbc_catalog",
  "user" = "mysql_user",
  "password" = "mysql_passwd",
  "host" = "127.0.0.1",
   "port" = "8239"
);

# Then create mysql external table through Resource
CREATE EXTERNAL TABLE example_db.table_mysql
(
k1 DATE,
k2 INT,
k3 SMALLINT,
k4 VARCHAR(2048),
k5 DATETIME
)
ENGINE=mysql
PROPERTIES
(
"odbc_catalog_resource" = "mysql_resource",
"database" = "mysql_db_test",
"table" = "mysql_table_test"
)
```

2. Create a broker external table with data files stored on HDFS, the data is split with “|”, and “” is newline

```
CREATE EXTERNAL TABLE example_db.table_broker (
k1 DATE,
```

```
k2 INT,
k3 SMALLINT,
k4 VARCHAR(2048),
k5 DATETIME
)
ENGINE=broker
PROPERTIES (
"broker_name" = "hdfs",
"path" = "hdfs://hdfs_host:hdfs_port/data1,hdfs://hdfs_host:hdfs_port/data2,hdfs://hdfs_host:hdfs_port/data3%2c4",
"column_separator" = "|",
"line_delimiter" = "\n"
)
BROKER PROPERTIES (
"username" = "hdfs_user",
"password" = "hdfs_password"
)
```

3. Create a hive external table

```
CREATE TABLE example_db.table_hive
(
  k1 TINYINT,
  k2 VARCHAR(50),
  v INT
)
ENGINE=hive
PROPERTIES
(
  "database" = "hive_db_name",
  "table" = "hive_table_name",
  "hive.metastore.uris" = "thrift://127.0.0.1:9083"
);
```

4. Create an Iceberg skin

```
CREATE TABLE example_db.t_iceberg
ENGINE=ICEBERG
PROPERTIES (
"iceberg.database" = "iceberg_db",
"iceberg.table" = "iceberg_table",
"iceberg.hive.metastore.uris" = "thrift://127.0.0.1:9083",
"iceberg.catalog.type" = "HIVE_CATALOG"
);
```

5. Create an Hudi external table

   create hudi table without schema(recommend)

```
CREATE TABLE example_db.t_hudi
ENGINE=HUDI
PROPERTIES (
"hudi.database" = "hudi_db_in_hive_metastore",
"hudi.table" = "hudi_table_in_hive_metastore",
"hudi.hive.metastore.uris" = "thrift://127.0.0.1:9083"
);
```

   create hudi table with schema

```
CREATE TABLE example_db.t_hudi (
   `id` int NOT NULL COMMENT "id number",
   `name` varchar(10) NOT NULL COMMENT "user name"
)
ENGINE=HUDI
PROPERTIES (
"hudi.database" = "hudi_db_in_hive_metastore",
"hudi.table" = "hudi_table_in_hive_metastore",
"hudi.hive.metastore.uris" = "thrift://127.0.0.1:9083"
);
```

Keywords

```
CREATE, EXTERNAL, TABLE
```

Best Practice

CREATE-INDEX

CREATE-INDEX

Name

CREATE INDEX

Description

This statement is used to create an index grammar:

```
CREATE INDEX [IF NOT EXISTS] index_name ON table_name (column [, ...],) [USING INVERTED] [COMMENT 'balabala'];
```

Notice: - INVERTED indexes are only created on a single column

Example

1. Create a inverted index for siteid on table1

   ```
   CREATE INDEX [IF NOT EXISTS] index_name ON table1 (siteid) USING INVERTED COMMENT 'balabala';
   ```

Keywords

```
CREATE, INDEX
```

Best Practice

CREATE-VIEW

CREATE-VIEW

Name

CREATE VIEW

Description

This statement is used to create a logical view grammar:

```
CREATE VIEW [IF NOT EXISTS]
 [db_name.]view_name
 (column1[ COMMENT "col comment"][, column2, ...])
AS query_stmt
```

illustrate:

- Views are logical views and have no physical storage. All queries on the view are equivalent to the sub-queries corresponding to the view.
- query_stmt is any supported SQL

Example

1. Create the view example_view on example_db

   ```
   CREATE VIEW example_db.example_view (k1, k2, k3, v1)
   AS
   SELECT c1 as k1, k2, k3, SUM(v1) FROM example_table
   WHERE k1 = 20160112 GROUP BY k1,k2,k3;
   ```

2. Create a view with a comment

   ```
   CREATE VIEW example_db.example_view
   (
       k1 COMMENT "first key",
       k2 COMMENT "second key",
       k3 COMMENT "third key",
       v1 COMMENT "first value"
   )
   COMMENT "my first view"
   AS
   SELECT c1 as k1, k2, k3, SUM(v1) FROM example_table
   WHERE k1 = 20160112 GROUP BY k1,k2,k3;
   ```

Keywords

```
CREATE, VIEW
```

Best Practice

CREATE-MATERIALIZED-VIEW

CREATE-MATERIALIZED-VIEW

Name

CREATE MATERIALIZED VIEW

Description

This statement is used to create a materialized view.

This operation is an asynchronous operation. After the submission is successful, you need to view the job progress through SHOW ALTER TABLE MATERIALIZED VIEW. After displaying FINISHED, you can use the `desc [table_name] all` command to view the schema of the materialized view.

grammar:

```
CREATE MATERIALIZED VIEW < MV name > as < query >
[PROPERTIES ("key" = "value")]
```

illustrate:

- `MV name`: The name of the materialized view, required. Materialized view names for the same table cannot be repeated.
- `query`: The query statement used to construct the materialized view, the result of the query statement is the data of the materialized view. Currently supported query formats are:

  ```
  SELECT select_expr[, select_expr ...]
  FROM [Base view name]
  GROUP BY column_name[, column_name ...]
  ORDER BY column_name[, column_name ...]
  ```

  The syntax is the same as the query syntax.

  - `select_expr`: All columns in the schema of the materialized view.
    - Contains at least one single column.
  - `base view name`: The original table name of the materialized view, required.
    - Must be a single table and not a subquery
  - `group by`: The grouping column of the materialized view, optional.
    - If not filled, the data will not be grouped.
  - `order by`: the sorting column of the materialized view, optional.
    - The declaration order of the sort column must be the same as the column declaration order in select_expr.
    - If order by is not declared, the sorting column is automatically supplemented according to the rules. If the materialized view is an aggregate type, all grouping columns are

automatically supplemented as sort columns. If the materialized view is of a non-aggregate type, the first 36 bytes are automatically supplemented as the sort column.
    - If the number of auto-supplemented sorts is less than 3, the first three are used as the sort sequence. If query contains a grouping column, the sorting column must be the same as the grouping column.

- properties

  Declare some configuration of the materialized view, optional.

  ```
  PROPERTIES ("key" = "value", "key" = "value" ...)
  ```

  The following configurations can be declared here:

  ```
  short_key: The number of sorting columns.
  timeout: The timeout for materialized view construction.
  ```

Example

Base table structure is

```
mysql> desc duplicate_table;
+-------+--------+------+------+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+--------+------+------+---------+-------+
| k1 | INT | Yes | true | N/A | |
| k2 | INT | Yes | true | N/A | |
| k3 | BIGINT | Yes | true | N/A | |
| k4 | BIGINT | Yes | true | N/A | |
+-------+--------+------+------+---------+-------+
```

```
create table duplicate_table(
    k1 int null,
    k2 int null,
    k3 bigint null,
    k4 bigint null
)
duplicate key (k1,k2,k3,k4)
distributed BY hash(k4) buckets 3
properties("replication_num" = "1");
```

attention：If the materialized view contains partitioned and distributed columns of the Base table, these columns must be used as key columns in the materialized view

1. Create a materialized view that contains only the columns of the original table (k1, k2)

   ```
   create materialized view k2_k1 as
   select k2, k1 from duplicate_table;
   ```

   The schema of the materialized view is as follows, the materialized view contains only two columns k1, k2 without any aggregation

```
+-------+-------+--------+------+------+ ---------+-------+
| IndexName | Field | Type | Null | Key | Default | Extra |
+-------+-------+--------+------+------+ ---------+-------+
| k2_k1 | k2 | INT | Yes | true | N/A | |
| | k1 | INT | Yes | true | N/A | |
+-------+-------+--------+------+------+ ---------+-------+
```

2. Create a materialized view with k2 as the sort column

```
create materialized view k2_order as
select k2, k1 from duplicate_table order by k2;
```

The schema of the materialized view is shown in the figure below. The materialized view contains only two columns k2, k1, where k2 is the sorting column without any aggregation.

```
+-------+-------+--------+------+------- +---------+-------+
| IndexName | Field | Type | Null | Key | Default | Extra |
+-------+-------+--------+------+------- +---------+-------+
| k2_order | k2 | INT | Yes | true | N/A | |
| | k1 | INT | Yes | false | N/A | NONE |
+-------+-------+--------+------+------- +---------+-------+
```

3. Create a materialized view with k1, k2 grouping and k3 column aggregated by SUM

```
create materialized view k1_k2_sumk3 as
select k1, k2, sum(k3) from duplicate_table group by k1, k2;
```

The schema of the materialized view is shown in the figure below. The materialized view contains two columns k1, k2, sum(k3) where k1, k2 are the grouping columns, and sum(k3) is the sum value of the k3 column grouped by k1, k2.

Since the materialized view does not declare a sorting column, and the materialized view has aggregated data, the system defaults to supplement the grouping columns k1 and k2 as sorting columns.

```
+-------+-------+--------+------+------- +---------+-------+
| IndexName | Field | Type | Null | Key | Default | Extra |
+-------+-------+--------+------+------- +---------+-------+
| k1_k2_sumk3 | k1 | INT | Yes | true | N/A | |
| | k2 | INT | Yes | true | N/A | |
| | k3 | BIGINT | Yes | false | N/A | SUM |
+-------+-------+--------+------+------- +---------+-------+
```

4. Create a materialized view that removes duplicate rows

```
create materialized view deduplicate as
select k1, k2, k3, k4 from duplicate_table group by k1, k2, k3, k4;
```

The materialized view schema is as shown below. The materialized view contains columns k1, k2, k3, and k4, and there are no duplicate rows.

```
+-------+-------+--------+------+------- +---------+-------+
| IndexName | Field | Type | Null | Key | Default | Extra |
+-------+-------+--------+------+------- +---------+-------+
| deduplicate | k1 | INT | Yes | true | N/A | |
| | k2 | INT | Yes | true | N/A | |
| | k3 | BIGINT | Yes | true | N/A | |
| | k4 | BIGINT | Yes | true | N/A | |
+-------+-------+--------+------+------- +---------+-------+
```

5. Create a non-aggregate materialized view that does not declare a sort column

   The schema of all_type_table is as follows

```
+-------+--------------+------+-------+---------+- ------+
| Field | Type | Null | Key | Default | Extra |
+-------+--------------+------+-------+---------+- ------+
| k1 | TINYINT | Yes | true | N/A | |
| k2 | SMALLINT | Yes | true | N/A | |
| k3 | INT | Yes | true | N/A | |
| k4 | BIGINT | Yes | true | N/A | |
| k5 | DECIMAL(9,0) | Yes | true | N/A | |
| k6 | DOUBLE | Yes | false | N/A | NONE |
| k7 | VARCHAR(20) | Yes | false | N/A | NONE |
+-------+--------------+------+-------+---------+- ------+
```

   The materialized view contains k3, k4, k5, k6, k7 columns, and does not declare a sort column, the creation statement is as follows:

```
create materialized view mv_1 as
select k3, k4, k5, k6, k7 from all_type_table;
```

   The default added sorting column of the system is k3, k4, k5 three columns. The sum of the bytes of these three column types is 4(INT) + 8(BIGINT) + 16(DECIMAL) = 28 < 36. So the addition is that these three columns are used as sorting columns. The schema of the materialized view is as follows, you can see that the key field of the k3, k4, k5 columns is true, that is, the sorting column. The key field of the k6, k7 columns is false, which is a non-sorted column.

```
+----------------+-------+--------------+------+-- -----+---------+-------+
| IndexName | Field | Type | Null | Key | Default | Extra |
+----------------+-------+--------------+------+-- -----+---------+-------+
| mv_1 | k3 | INT | Yes | true | N/A | |
| | k4 | BIGINT | Yes | true | N/A | |
| | k5 | DECIMAL(9,0) | Yes | true | N/A | |
| | k6 | DOUBLE | Yes | false | N/A | NONE |
| | k7 | VARCHAR(20) | Yes | false | N/A | NONE |
+----------------+-------+--------------+------+-- -----+---------+-------+
```

Keywords

```
CREATE, MATERIALIZED, VIEW
```

Best Practice

CREATE-FUNCTION

CREATE-FUNCTION

Name

CREATE FUNCTION

Description

This statement creates a custom function. Executing this command requires the user to have `ADMIN` privileges.

If `function_name` contains the database name, then the custom function will be created in the corresponding database, otherwise the function will be created in the database where the current session is located. The name and parameters of the new function cannot be the same as the existing functions in the current namespace, otherwise the creation will fail. But only with the same name and different parameters can be created successfully.

grammar:

```
CREATE [GLOBAL] [AGGREGATE] [ALIAS] FUNCTION function_name
    (arg_type [, ...])
    [RETURNS ret_type]
    [INTERMEDIATE inter_type]
    [WITH PARAMETER(param [,...]) AS origin_function]
    [PROPERTIES ("key" = "value" [, ...]) ]
```

Parameter Description:

- `GLOBAL`: If there is this item, it means that the created function is a global function.
- `AGGREGATE`: If there is this item, it means that the created function is an aggregate function.
- `ALIAS`: If there is this item, it means that the created function is an alias function.

If the above two items are absent, it means that the created function is a scalar function

- `function_name`: The name of the function to be created, which can include the name of the database. For example: `db1.my_func`.
- `arg_type`: The parameter type of the function, which is the same as the type defined when creating the table. Variable-length parameters can be represented by `, ...`. If it is a variable-length type, the type of the variable-length parameter is the same as that of the last non-variable-length parameter.

  NOTE: `ALIAS FUNCTION` does not support variable-length arguments and must have at least one argument.
- `ret_type`: Required for creating new functions. If you are aliasing an existing function, you do not need to fill in this parameter.
- `inter_type`: The data type used to represent the intermediate stage of the aggregation function.

- `param`: used to represent the parameter of the alias function, including at least one.
- `origin_function`: used to represent the original function corresponding to the alias function.
- `properties`: Used to set function-related properties, the properties that can be set include:
    - `file`: Indicates the jar package containing the user UDF. In a multi-machine environment, you can also use http to download the jar package. This parameter is mandatory.
    - `symbol`: Indicates the name of the class containing the UDF class. This parameter must be set
    - `type`: Indicates the UDF call type, the default is Native, and JAVA_UDF is passed when using Java UDF.
    - `always_nullable`: Indicates whether NULL values may appear in the UDF return result, is an optional parameter, and the default value is true.

Example

1. Create a custom UDF function

```
CREATE FUNCTION java_udf_add_one(int) RETURNS int PROPERTIES (
    "file"="file:///path/to/java-udf-demo-jar-with-dependencies.jar",
    "symbol"="org.apache.doris.udf.AddOne",
    "always_nullable"="true",
    "type"="JAVA_UDF"
);
```

2. Create a custom UDAF function

```
CREATE AGGREGATE FUNCTION simple_sum(INT) RETURNS INT PROPERTIES (
    "file"="file:///pathTo/java-udaf.jar",
    "symbol"="org.apache.doris.udf.demo.SimpleDemo",
    "always_nullable"="true",
    "type"="JAVA_UDF"
);
```

3. Create a custom alias function

```
CREATE ALIAS FUNCTION id_masking(INT) WITH PARAMETER(id) AS CONCAT(LEFT(id, 3), '****', RIGHT(id, 4));
```

4. Create a global custom alias function

```
CREATE GLOBAL ALIAS FUNCTION id_masking(INT) WITH PARAMETER(id) AS CONCAT(LEFT(id, 3), '****', RIGHT(id, 4));
```

Keywords

```
CREATE, FUNCTION
```

Best Practice

CREATE-FILE

CREATE-FILE

Name

CREATE FILE

Description

This statement is used to create and upload a file to the Doris cluster. This function is usually used to manage files that need to be used in some other commands, such as certificates, public and private keys, and so on.

This command can only be executed by users with `admin` privileges. A certain file belongs to a certain database. This file can be used by any user with access rights to database.

A single file size is limited to 1MB. A Doris cluster can upload up to 100 files.

grammar:

```
CREATE FILE "file_name" [IN database]
PROPERTIES("key"="value", ...)
```

illustrate:

- file_name: custom file name.
- database: The file belongs to a certain db, if not specified, the db of the current session is used.
- properties supports the following parameters:
    - url: Required. Specifies the download path for a file. Currently only unauthenticated http download paths are supported. After the command executes successfully, the file will be saved in doris and the url will no longer be needed.
    - catalog: Required. The classification name of the file can be customized. However, in some commands, files in the specified catalog are looked up. For example, in the routine import, when the data source is kafka, the file under the catalog name kafka will be searched.
    - md5: optional. md5 of the file. If specified, verification will be performed after the file is downloaded.

Example

1. Create a file ca.pem , classified as kafka

```
CREATE FILE "ca.pem"
PROPERTIES
(
    "url" = "https://test.bj.bcebos.com/kafka-key/ca.pem",
    "catalog" = "kafka"
);
```

2. Create a file client.key, classified as my_catalog

```
CREATE FILE "client.key"
IN my_database
PROPERTIES
(
    "url" = "https://test.bj.bcebos.com/kafka-key/client.key",
    "catalog" = "my_catalog",
    "md5" = "b5bb901bf10f99205b39a46ac3557dd9"
);
```

Keywords

```
CREATE, FILE
```

Best Practice

1. This command can only be executed by users with amdin privileges. A certain file belongs to a certain database. This file can be used by any user with access rights to database.

2. File size and quantity restrictions.

   This function is mainly used to manage some small files such as certificates. So a single file size is limited to 1MB. A Doris cluster can upload up to 100 files.

CREATE-POLICY

CREATE-POLICY

Name

CREATE POLICY

Description

Create policies,such as: 1. Create security policies(ROW POLICY) and explain to view the rewritten SQL. 2. Create storage migration policy(STORAGE POLICY), used for cold and hot data transform

Grammar:

1. ROW POLICY

```
CREATE ROW POLICY test_row_policy_1 ON test.table1
AS {RESTRICTIVE|PERMISSIVE} TO test USING (id in (1, 2));
```

illustrate：

- filterType：It is usual to constrict a set of policies through AND. PERMISSIVE to constrict a set of policies through OR
- Configure multiple policies. First, merge the RESTRICTIVE policy with the PERMISSIVE policy
- It is connected with AND between RESTRICTIVE AND PERMISSIVE
- It cannot be created for users root and admin

2. STORAGE POLICY

```
CREATE STORAGE POLICY test_storage_policy_1
PROPERTIES ("key"="value", ...);
```

illustrate： - PROPERTIES has such keys: 1. storage_resource：storage resource name for policy 2. cooldown_datetime：cool down time for tablet, can't be set with cooldown_ttl. 3. cooldown_ttl：hot data stay time. The time cost between the time of tablet created and the time of migrated to cold data, formatted as：1d：1 day 1h：1 hour 50000: 50000 second

Example

1. Create a set of row security policies

```
CREATE ROW POLICY test_row_policy_1 ON test.table1
AS RESTRICTIVE TO test USING (c1 = 'a');
```

```
CREATE ROW POLICY test_row_policy_2 ON test.table1
AS RESTRICTIVE TO test USING (c2 = 'b');
```

```
CREATE ROW POLICY test_row_policy_3 ON test.table1
AS PERMISSIVE TO test USING (c3 = 'c');
```

```
CREATE ROW POLICY test_row_policy_3 ON test.table1
AS PERMISSIVE TO test USING (c4 = 'd');
```

When we execute the query on Table1, the rewritten SQL is

```
select * from (select * from table1 where c1 = 'a' and c2 = 'b' or c3 = 'c' or c4 = 'd')
```

2. Create policy for storage

    1. NOTE
        - To create a cold hot separation policy, you must first create a resource, and then associate the created resource name when creating a migration policy
        - Currently, the drop data migration policy is not supported to prevent data from being migrated. If the policy has been deleted, then the system cannot retrieve the data
    2. Create policy on cooldown_datetime

```
CREATE STORAGE POLICY testPolicy
PROPERTIES(
  "storage_resource" = "s3",
  "cooldown_datetime" = "2022-06-08 00:00:00"
);
```

    3. Create policy on cooldown_ttl

```
CREATE STORAGE POLICY testPolicy
PROPERTIES(
  "storage_resource" = "s3",
  "cooldown_ttl" = "1d"
);
```

    Relevant parameters are as follows:

    - `storage_resource`: the storage resource of create
    - `cooldown_datetime`: Data migration time
    - `cooldown_ttl`: Countdown of the distance between the migrated data and the current time

Keywords

```
CREATE, POLICY
```

Best Practice

CREATE-ENCRYPT-KEY

CREATE-ENCRYPT-KEY

Name

CREATE ENCRYPTKEY

Description

This statement creates a custom key. Executing this command requires the user to have `ADMIN` privileges.

grammar:

```
CREATE ENCRYPTKEY key_name AS "key_string"
```

illustrate:

`key_name`: The name of the key to be created, may contain the name of the database. For example: `db1.my_key`.

`key_string`: The string to create the key with.

If `key_name` contains the database name, then the custom key will be created in the corresponding database, otherwise this function will create the database in the current session. The name of the new key cannot be the same as the existing key in the corresponding database, otherwise the creation will fail.

Example

1. Create a custom key

   ```
   CREATE ENCRYPTKEY my_key AS "ABCD123456789";
   ```

2. Use a custom key

   To use a custom key, you need to add the keyword `KEY/key` before the key, separated from the `key_name` space.

   ```
   mysql> SELECT HEX(AES_ENCRYPT("Doris is Great", KEY my_key));
   +-----------------------------------------------+
   | hex(aes_encrypt('Doris is Great', key my_key)) |
   +-----------------------------------------------+
   | D26DB38579D6A343350EDDC6F2AD47C6 |
   +-----------------------------------------------+
   1 row in set (0.02 sec)

   mysql> SELECT AES_DECRYPT(UNHEX('D26DB38579D6A343350EDDC6F2AD47C6'), KEY my_key);
   +------------------------------------------------ -------------------+
   | aes_decrypt(unhex('D26DB38579D6A343350EDDC6F2AD47C6'), key my_key) |
   +------------------------------------------------ -------------------+
   ```

```
| Doris is Great |
+----------------------------------------------- ------------------+
1 row in set (0.01 sec)
```

Keywords

```
CREATE, ENCRYPTKEY
```

Best Practice

CREATE-RESOURCE

CREATE-RESOURCE

Name

CREATE RESOURCE

Description

This statement is used to create a resource. Only the root or admin user can create resources. Currently supports Spark, ODBC, S3 external resources. In the future, other external resources may be added to Doris for use, such as Spark/GPU for query, HDFS/S3 for external storage, MapReduce for ETL, etc.

grammar:

```
CREATE [EXTERNAL] RESOURCE "resource_name"
PROPERTIES ("key"="value", ...);
```

illustrate:

- The type of resource needs to be specified in PROPERTIES “type” = “[spark|odbc_catalog|s3]”, currently supports spark, odbc_catalog, s3.
- PROPERTIES differs depending on the resource type, see the example for details.

Example

1. Create a Spark resource named spark0 in yarn cluster mode.

   ```
   CREATE EXTERNAL RESOURCE "spark0"
   PROPERTIES
   (
     "type" = "spark",
     "spark.master" = "yarn",
     "spark.submit.deployMode" = "cluster",
     "spark.jars" = "xxx.jar,yyy.jar",
     "spark.files" = "/tmp/aaa,/tmp/bbb",
     "spark.executor.memory" = "1g",
     "spark.yarn.queue" = "queue0",
     "spark.hadoop.yarn.resourcemanager.address" = "127.0.0.1:9999",
     "spark.hadoop.fs.defaultFS" = "hdfs://127.0.0.1:10000",
     "working_dir" = "hdfs://127.0.0.1:10000/tmp/doris",
     "broker" = "broker0",
     "broker.username" = "user0",
     "broker.password" = "password0"
   );
   ```

   Spark related parameters are as follows:

   - spark.master: Required, currently supports yarn, spark://host:port.

- spark.submit.deployMode: The deployment mode of the Spark program, required, supports both cluster and client.
- spark.hadoop.yarn.resourcemanager.address: Required when master is yarn.
- spark.hadoop.fs.defaultFS: Required when master is yarn.
- Other parameters are optional, refer to here

Working_dir and broker need to be specified when Spark is used for ETL. described as follows:

- working_dir: The directory used by the ETL. Required when spark is used as an ETL resource. For example: hdfs://host:port/tmp/doris.
- broker: broker name. Required when spark is used as an ETL resource. Configuration needs to be done in advance using the `ALTER SYSTEM ADD BROKER` command.
- broker.property_key: The authentication information that the broker needs to specify when reading the intermediate file generated by ETL.

2. Create an ODBC resource

```
CREATE EXTERNAL RESOURCE `oracle_odbc`
PROPERTIES (
"type" = "odbc_catalog",
"host" = "192.168.0.1",
"port" = "8086",
"user" = "test",
"password" = "test",
"database" = "test",
"odbc_type" = "oracle",
"driver" = "Oracle 19 ODBC driver"
);
```

The relevant parameters of ODBC are as follows:

- hosts: IP address of the external database
- driver: The driver name of the ODBC appearance, which must be the same as the Driver name in be/conf/odbcinst.ini.
- odbc_type: the type of the external database, currently supports oracle, mysql, postgresql
- user: username of the foreign database
- password: the password information of the corresponding user
- charset: connection charset
- There is also support for implementing custom parameters per ODBC Driver, see the description of the corresponding ODBC Driver

3. Create S3 resource

```
CREATE RESOURCE "remote_s3"
PROPERTIES
(
   "type" = "s3",
   "s3.endpoint" = "bj.s3.com",
   "s3.region" = "bj",
   "s3.access_key" = "bbb",
   "s3.secret_key" = "aaaa",
   -- the followings are optional
   "s3.connection.maximum" = "50",
   "s3.connection.request.timeout" = "3000",
```

```
   "s3.connection.timeout" = "1000"
);
```

If S3 resource is used for cold hot separation, we should add more required fields.

```
CREATE RESOURCE "remote_s3"
PROPERTIES
(
   "type" = "s3",
   "s3.endpoint" = "bj.s3.com",
   "s3.region" = "bj",
   "s3.access_key" = "bbb",
   "s3.secret_key" = "aaaa",
   -- required by cooldown
   "s3.root.path" = "/path/to/root",
   "s3.bucket" = "test-bucket"
);
```

S3 related parameters are as follows:

- Required parameters
  - `s3.endpoint`: s3 endpoint
  - `s3.region`:s3 region
  - `s3.root.path`: s3 root directory
  - `s3.access_key`: s3 access key
  - `s3.secret_key`: s3 secret key
  - `s3.bucket`：s3 bucket
- optional parameter
  - `s3.connection.maximum`: the maximum number of s3 connections, the default is 50
  - `s3.connection.request.timeout`: s3 request timeout, in milliseconds, the default is 3000
  - `s3.connection.timeout`: s3 connection timeout, in milliseconds, the default is 1000

4. Create JDBC resource

```
CREATE RESOURCE mysql_resource PROPERTIES (
   "type"="jdbc",
   "user"="root",
   "password"="123456",
   "jdbc_url" = "jdbc:mysql://127.0.0.1:3316/doris_test?useSSL=false",
   "driver_url" = "https://doris-community-test-1308700295.cos.ap-hongkong.myqcloud.com/jdbc_driver/mysql-connector-java-
8.0.25.jar",
"driver_class" = "com.mysql.cj.jdbc.Driver"
);
```

JDBC related parameters are as follows:

- user：The username used to connect to the database
- password：The password used to connect to the database
- jdbc_url: The identifier used to connect to the specified database
- driver_url: The url of JDBC driver package
- driver_class: The class of JDBC driver

5. Create HDFS resource

```
CREATE RESOURCE hdfs_resource PROPERTIES (
   "type"="hdfs",
   "hadoop.username"="user",
   "dfs.nameservices" = "my_ha",
   "dfs.ha.namenodes.my_ha" = "my_namenode1, my_namenode2",
   "dfs.namenode.rpc-address.my_ha.my_namenode1" = "nn1_host:rpc_port",
   "dfs.namenode.rpc-address.my_ha.my_namenode2" = "nn2_host:rpc_port",
   "dfs.client.failover.proxy.provider.my_ha" = "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"
);
```

HDFS related parameters are as follows:

- fs.defaultFS: namenode address and port
- hadoop.username: hdfs username
- dfs.nameservices: if hadoop enable HA, please set fs nameservice. See hdfs-site.xml
- dfs.ha.namenodes.[nameservice ID]：unique identifiers for each NameNode in the nameservice. See hdfs-site.xml
- dfs.namenode.rpc-address.[nameservice ID].[name node ID]`：the fully-qualified RPC address for each NameNode to listen on. See hdfs-site.xml
- dfs.client.failover.proxy.provider.[nameservice ID]：the Java class that HDFS clients use to contact the Active NameNode, usually it is org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider

6. Create HMS resource

   HMS resource is used to create hms catalog

```
CREATE RESOURCE hms_resource PROPERTIES (
   'type'='hms',
   'hive.metastore.uris' = 'thrift://127.0.0.1:7004',
   'dfs.nameservices'='HANN',
   'dfs.ha.namenodes.HANN'='nn1,nn2',
   'dfs.namenode.rpc-address.HANN.nn1'='nn1_host:rpc_port',
   'dfs.namenode.rpc-address.HANN.nn2'='nn2_host:rpc_port',
   'dfs.client.failover.proxy.provider.HANN'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider'
);
```

   HMS related parameters are as follows:

   - hive.metastore.uris: hive metastore server address Optional:
   - dfs.*: If hive data is on hdfs, HDFS resource parameters should be added, or copy hive-site.xml into fe/conf.
   - s3.*: If hive data is on s3, S3 resource parameters should be added. If using Aliyun Data Lake Formation, copy hive-site.xml into fe/conf.

7. Create ES resource

```
CREATE RESOURCE es_resource PROPERTIES (
   "type"="es",
   "hosts"="http://127.0.0.1:29200",
   "nodes_discovery"="false",
   "enable_keyword_sniff"="true"
);
```

ES related parameters are as follows:

- hosts: ES Connection Address, maybe one or more node, load-balance is also accepted
- user: username for ES
- password: password for the user
- enable_docvalue_scan: whether to enable ES/Lucene column storage to get the value of the query field, the default is true
- enable_keyword_sniff: Whether to probe the string segmentation type text.fields in ES, query by keyword (the default is true, false matches the content after the segmentation)
- nodes_discovery: Whether or not to enable ES node discovery, the default is true. In network isolation, set this parameter to false. Only the specified node is connected
- http_ssl_enabled: Whether ES cluster enables https access mode, the current FE/BE implementation is to trust all

Keywords

```
CREATE, RESOURCE
```

Best Practice

CREATE-WORKLOAD-GORUP

CREATE-WORKLOAD-GROUP

Name

CREATE WORKLOAD GROUP

Description

This statement is used to create a workload group. Workload groups enable the isolation of cpu resources and memory resources on a single be.

grammar:

```
CREATE WORKLOAD GROUP [IF NOT EXISTS] "rg_name"
PROPERTIES (
    property_list
);
```

illustrate:

Properties supported by property_list:

- cpu_share: Required, used to set how much cpu time the workload group can acquire, which can achieve soft isolation of cpu resources. cpu_share is a relative value indicating the weight of cpu resources available to the running workload group. For example, if a user creates 3 workload groups rg-a, rg-b and rg-c with cpu_share of 10, 30 and 40 respectively, and at a certain moment rg-a and rg-b are running tasks while rg-c has no tasks, then rg-a can get (10 / (10 + 30)) = 25% of the cpu resources while workload group rg-b can get 75% of the cpu resources. If the system has only one workload group running, it gets all the cpu resources regardless of the value of its cpu_share.
- memory_limit: Required, set the percentage of be memory that can be used by the workload group. The absolute value of the workload group memory limit is: `physical_memory * mem_limit * memory_limit`, where mem_limit is a be configuration item. The total memory_limit of all workload groups in the system must not exceed 100%. Workload groups are guaranteed to use the memory_limit for the tasks in the group in most cases. When the workload group memory usage exceeds this limit, tasks in the group with larger memory usage may be canceled to release the excess memory, refer to enable_memory_overcommit.
- enable_memory_overcommit: Optional, enable soft memory isolation for the workload group, default is false. if set to false, the workload group is hard memory isolated and the tasks with the largest memory usage will be canceled immediately after the workload group memory usage exceeds the limit to release the excess memory. if set to true, the workload group is hard memory isolated and the tasks with the largest memory usage will be canceled immediately after the workload group memory usage exceeds the limit to release the excess memory. if set to true, the workload group is softly isolated, if the system has free memory resources, the workload group can continue to use system memory after exceeding the memory_limit limit, and when the total system memory is tight, it will cancel several tasks in the

group with the largest memory occupation, releasing part of the excess memory to relieve the system memory pressure. It is recommended that when this configuration is enabled for a workload group, the total memory_limit of all workload groups should be less than 100%, and the remaining portion should be used for workload group memory overcommit.

Example

1. Create a workload group named g1:

```
create workload group if not exists g1
properties (
    "cpu_share"="10",
    "memory_limit"="30%",
    "enable_memory_overcommit"="true"
);
```

Keywords

```
CREATE, WORKLOAD, GROUP
```

Best Practice

CREATE-SQL-BLOCK-RULE

CREATE-SQL-BLOCK-RULE

Name

CREATE SQL BLOCK RULE

Description

This statement creates a SQL blocking rule. it can restrict any kind of sql statements(no matter DDL or DML statement).

Supports configuring SQL blacklists by user:

- Refuse to specify SQL by regular matching
- Check if a sql reaches one of these limits by setting partition_num, tablet_num, cardinality
    - partition_num, tablet_num, cardinality can be set together, once a query reaches one of these limits, the query will be intercepted

grammar:

```
CREATE SQL_BLOCK_RULE rule_name
[PROPERTIES ("key"="value", ...)];
```

Parameter Description:

- sql: matching rule (based on regular matching, special characters need to be translated,for example`select *`use`select \\*`), optional, the default value is “NULL”
- sqlHash: sql hash value, used for exact matching, we will print this value in `fe.audit.log`, optional, this parameter and sql can only be selected one, the default value is “NULL”
- partition_num: the maximum number of partitions a scan node will scan, the default value is 0L
- tablet_num: The maximum number of tablets that a scanning node will scan, the default value is 0L
- cardinality: the rough scan line number of a scan node, the default value is 0L
- global: Whether to take effect globally (all users), the default is false
- enable: whether to enable blocking rules, the default is true

Example

1. Create a block rule named test_rule

    ```
    CREATE SQL_BLOCK_RULE test_rule
    PROPERTIES(
    "sql"="select \\* from order_analysis",
    "global"="false",
    "enable"="true"
    );
    ```

> Notes:
>
> That the sql statement here does not end with a semicolon

When we execute the sql we just defined in the rule, an exception error will be returned. The example is as follows:

```
select * from order_analysis;
ERROR 1064 (HY000): errCode = 2, detailMessage = sql match regex sql block rule: order_analysis_rule
```

2. Create test_rule2, limit the maximum number of scanned partitions to 30, and limit the maximum scan base to 10 billion rows. The example is as follows:

```
CREATE SQL_BLOCK_RULE test_rule2
PROPERTIES (
"partition_num" = "30",
"cardinality" = "10000000000",
"global" = "false",
"enable" = "true"
);
```

3. Create SQL BLOCK RULE with special chars

```
CREATE SQL_BLOCK_RULE test_rule3
PROPERTIES
(
"sql" = "select count\\(1\\) from db1.tbl1"
);
CREATE SQL_BLOCK_RULE test_rule4
PROPERTIES
(
"sql" = "select \\* from db1.tbl1"
);
```

4. In SQL_BLOCK_RULE, SQL matching is based on regular expressions. If want to match more patterns of SQL, need to write the corresponding regex. For example, to ignore spaces in SQL and not query tables that start with 'order_', as shown below:

```
CREATE SQL_BLOCK_RULE test_rule4
PROPERTIES(
  "sql"="\\s*select\\s*\\*\\s*from order_\\w*\\s*",
  "global"="false",
  "enable"="true"
);
```

APPENDIX

Here are some commonly used regular expressions: > . ： Matches any single character except for a newline character . > > * ： Matches the preceding element zero or more times. For example, a matches zero or more 'a'. > > + ： Matches the preceding element one or more times. For example, a+ matches one or more 'a'. > > ? ： Matches the preceding element zero or one time. For example, a? matches zero or one 'a'. > > [] ： Used to

define a character set. For example, [aeiou] matches any one vowel letter. > > [^]： In a character set, use ^ to indicate negation, matching characters that are not in the set. For example, [^0-9] matches any non-digit character. > > ()： Used for grouping expressions and applying quantifiers. For example, (ab)+ matches consecutive ‘ab’. > > |： Represents logical OR. For example, a|b matches ‘a’ or ‘b’. > > ^： Matches the beginning of a string. For example, ^abc matches a string that starts with ‘abc’. > > $： Matches the end of a string. For example, xyz$ matches a string that ends with ‘xyz’. > > ： Used to escape special characters to treat them as ordinary characters. For example, \. matches the period character ‘.’. > >： Matches any whitespace character, including spaces, tabs, newline characters, etc. > > Matches any digit character, equivalent to [0-9]. > >： Matches any word character, including letters, digits, and underscores, equivalent to [a-zA-Z0-9_].

Keywords

```
CREATE, SQL_BLCOK_RULE
```

Best Practice

ALTER-CATALOG

ALTER-CATALOG

Name

> **Tips**
>
> This feature is supported since the Apache Doris 2.0 version

ALTER CATALOG

Description

This statement is used to set properties of the specified catalog. (administrator only)

1. Rename the catalog

```
ALTER CATALOG catalog_name RENAME new_catalog_name;
```

illustrate: - The builtin catalog `internal` cannot be renamed - Only the one who has at least Alter privilege can rename a catalog - After renaming the catalog, use the REVOKE and GRANT commands to modify the appropriate user permissions

2. Modify / add properties for the catalog

```
ALTER CATALOG catalog_name SET PROPERTIES ('key1' = 'value1' [, 'key' = 'value2']);
```

Update values of specified keys. If a key does not exist in the catalog properties, it will be added.

illustrate: - property `type` cannot be modified. - properties of builtin catalog `internal` cannot be modified.

3. Modify comment for the catalog

```
ALTER CATALOG catalog_name MODIFY COMMENT "new catalog comment";
```

illustrate： - The builtin catalog `internal` cannot be modified

Example

1. rename catalog ctlg_hive to hive

```
ALTER CATALOG ctlg_hive RENAME hive;
```

3. modify property `hive.metastore.uris` of catalog hive

```
ALTER CATALOG hive SET PROPERTIES ('hive.metastore.uris'='thrift://172.21.0.1:9083');
```

4. modify comment of catalog hive

```
ALTER CATALOG hive MODIFY COMMENT "new catalog comment";
```

Keywords

ALTER,CATALOG,RENAME,PROPERTY

Best Practice

ALTER-DATABASE

ALTER-DATABASE

Name

ALTER DATABASE

Description

This statement is used to set properties of the specified database. (administrator only)

1. Set the database data quota, the unit is B/K/KB/M/MB/G/GB/T/TB/P/PB

```
ALTER DATABASE db_name SET DATA QUOTA quota;
```

2. Rename the database

```
ALTER DATABASE db_name RENAME new_db_name;
```

3. Set the quota for the number of copies of the database

```
ALTER DATABASE db_name SET REPLICA QUOTA quota;
```

illustrate: After renaming the database, use the REVOKE and GRANT commands to modify the appropriate user permissions, if necessary. The default data quota for the database is 1PB, and the default replica quota is 1073741824.

4. Modify the properties of an existing database

```
ALTER DATABASE db_name SET PROPERTIES ("key"="value", ...);
```

Example

1. Set the specified database data volume quota

```
ALTER DATABASE example_db SET DATA QUOTA 10995116277760;
The above unit is bytes, which is equivalent to
ALTER DATABASE example_db SET DATA QUOTA 10T;

ALTER DATABASE example_db SET DATA QUOTA 100G;

ALTER DATABASE example_db SET DATA QUOTA 200M;
```

2. Rename the database example_db to example_db2

```
ALTER DATABASE example_db RENAME example_db2;
```

3. Set the quota for the number of copies of the specified database

```
ALTER DATABASE example_db SET REPLICA QUOTA 102400;
```

4. Modify the default replica distribution policy for tables in db (this operation only applies to newly created tables and will not modify existing tables in db)

```
ALTER DATABASE example_db SET PROPERTIES("replication_allocation" = "tag.location.default:2");
```

5. Cancel the default replica distribution policy for tables in db (this operation only applies to newly created tables and will not modify existing tables in db)

```
ALTER DATABASE example_db SET PROPERTIES("replication_allocation" = "");
```

Keywords

```
ALTER,DATABASE,RENAME
```

Best Practice

ALTER-TABLE-COLUMN

ALTER-TABLE-COLUMN

Name

ALTER TABLE COLUMN

Description

This statement is used to perform a schema change operation on an existing table. The schema change is asynchronous, and the task is returned when the task is submitted successfully. After that, you can use the SHOW ALTER TABLE COLUMN command to view the progress.

grammar:

```
ALTER TABLE [database.]table alter_clause;
```

The alter_clause of schema change supports the following modification methods:

1. Add a column to the specified position at the specified index

grammar:

```
ADD COLUMN column_name column_type [KEY | agg_type] [DEFAULT "default_value"]
[AFTER column_name|FIRST]
[TO rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

Notice:

- If you add a value column to the aggregation model, you need to specify agg_type
- For non-aggregated models (such as DUPLICATE KEY), if you add a key column, you need to specify the KEY keyword
- You cannot add columns that already exist in the base index to the rollup index (you can recreate a rollup index if necessary)

2. Add multiple columns to the specified index

grammar:

```
ADD COLUMN (column_name1 column_type [KEY | agg_type] DEFAULT "default_value", ...)
[TO rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

Notice:

- If you add a value column to the aggregation model, you need to specify agg_type

- If you add a key column to the aggregation model, you need to specify the KEY keyword
- You cannot add columns that already exist in the base index to the rollup index (you can recreate a rollup index if necessary)

3. Delete a column from the specified index

grammar:

```
DROP COLUMN column_name
[FROM rollup_index_name]
```

Notice:

- Cannot drop partition column
- If the column is removed from the base index, it will also be removed if it is included in the rollup index

4. Modify the column type and column position of the specified index

grammar:

```
MODIFY COLUMN column_name column_type [KEY | agg_type] [NULL | NOT NULL] [DEFAULT "default_value"]
[AFTER column_name|FIRST]
[FROM rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

Notice:

- If you modify the value column in the aggregation model, you need to specify agg_type
- If you modify the key column for non-aggregate types, you need to specify the KEY keyword
- Only the type of the column can be modified, and other attributes of the column remain as they are (that is, other attributes need to be explicitly written in the statement according to the original attributes, see example 8)
- Partitioning and bucketing columns cannot be modified in any way
- The following types of conversions are currently supported (loss of precision is guaranteed by the user)
    - Conversion of TINYINT/SMALLINT/INT/BIGINT/LARGEINT/FLOAT/DOUBLE types to larger numeric types
    - Convert TINTINT/SMALLINT/INT/BIGINT/LARGEINT/FLOAT/DOUBLE/DECIMAL to VARCHAR
    - VARCHAR supports modifying the maximum length
    - VARCHAR/CHAR converted to TINTINT/SMALLINT/INT/BIGINT/LARGEINT/FLOAT/DOUBLE
    - Convert VARCHAR/CHAR to DATE (currently supports “%Y-%m-%d”, “%y-%m-%d”, “%Y%m%d”, “%y%m%d”, “%Y/%m/%d,”%y/%m/%d” six formats)
    - Convert DATETIME to DATE (only keep year-month-day information, for example: `2019-12-09 21:47:05` <–> `2019-12-09`)
    - DATE is converted to DATETIME (hours, minutes and seconds are automatically filled with zeros, for example: `2019-12-09` <–> `2019-12-09 00:00:00`)
    - Convert FLOAT to DOUBLE
    - INT is converted to DATE (if the INT type data is illegal, the conversion fails, and the original data remains unchanged)
    - All can be converted to STRING except DATE and DATETIME, but STRING cannot be converted to any other type

5. Reorder the column at the specified index

grammar:

```
ORDER BY (column_name1, column_name2, ...)
[FROM rollup_index_name]
[PROPERTIES ("key"="value", ...)]
```

Notice:

- All columns in index are written out
- the value column comes after the key column

Example

1. Add a key column new_col after col1 of example_rollup_index (non-aggregated model)

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT KEY DEFAULT "0" AFTER col1
TO example_rollup_index;
```

2. Add a value column new_col after col1 of example_rollup_index (non-aggregation model)

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT DEFAULT "0" AFTER col1
TO example_rollup_index;
```

3. Add a key column new_col (aggregation model) after col1 of example_rollup_index

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT DEFAULT "0" AFTER col1
TO example_rollup_index;
```

4. Add a value column new_col SUM aggregation type (aggregation model) after col1 of example_rollup_index

```
ALTER TABLE example_db.my_table
ADD COLUMN new_col INT SUM DEFAULT "0" AFTER col1
TO example_rollup_index;
```

5. Add multiple columns to example_rollup_index (aggregation model)

```
ALTER TABLE example_db.my_table
ADD COLUMN (col1 INT DEFAULT "1", col2 FLOAT SUM DEFAULT "2.3")
TO example_rollup_index;
```

6. Remove a column from example_rollup_index

```
ALTER TABLE example_db.my_table
DROP COLUMN col2
```

```
FROM example_rollup_index;
```

7. Modify the type of the key column col1 of the base index to BIGINT and move it to the back of the col2 column.

```
ALTER TABLE example_db.my_table
MODIFY COLUMN col1 BIGINT KEY DEFAULT "1" AFTER col2;
```

Note: Whether you modify the key column or the value column, you need to declare complete column information

8. Modify the maximum length of the val1 column of base index. The original val1 is (val1 VARCHAR(32) REPLACE DEFAULT “abc”)

```
ALTER TABLE example_db.my_table
MODIFY COLUMN val1 VARCHAR(64) REPLACE DEFAULT "abc";
```

Note: You can only modify the column’s data type; other attributes of the column must remain unchanged.

9. Reorder the columns in example_rollup_index (set the original column order as: k1,k2,k3,v1,v2)

```
ALTER TABLE example_db.my_table
ORDER BY (k3,k1,k2,v2,v1)
FROM example_rollup_index;
```

10. Do Two Actions Simultaneously

```
ALTER TABLE example_db.my_table
ADD COLUMN v2 INT MAX DEFAULT "0" AFTER k2 TO example_rollup_index,
ORDER BY (k3,k1,k2,v2,v1) FROM example_rollup_index;
```

11. Modify the length of a field in the Key column of the Duplicate key table

```
alter table example_tbl modify column k3 varchar(50) key null comment 'to 50'
```

Keywords

```
ALTER, TABLE, COLUMN, ALTER TABLE
```

Best Practice

ALTER-TABLE-PARTITION

ALTER-TABLE-PARTITION

Name

ALTER TABLE PARTITION

Description

This statement is used to modify a table with a partition.

This operation is synchronous, and the return of the command indicates the completion of the execution.

grammar:

```
ALTER TABLE [database.]table alter_clause;
```

The alter_clause of partition supports the following modification methods

1. Add partition

grammar:

```
ADD PARTITION [IF NOT EXISTS] partition_name
partition_desc ["key"="value"]
[DISTRIBUTED BY HASH (k1[,k2 ...]) [BUCKETS num]]
```

Notice:

- partition_desc supports the following two ways of writing
    - VALUES LESS THAN [MAXVALUE|(“value1”, …)]
    - VALUES [(“value1”, …), (“value1”, …))
- The partition is left closed and right open. If the user only specifies the right boundary, the system will automatically determine the left boundary
- If the bucketing method is not specified, the bucketing method and bucket number used for creating the table would be automatically used
- If the bucketing method is specified, only the number of buckets can be modified, not the bucketing method or the bucketing column. If the bucketing method is specified but the number of buckets not be specified, the default value `10` will be used for bucket number instead of the number specified when the table is created. If the number of buckets modified, the bucketing method needs to be specified simultaneously.
- The [“key”=“value”] section can set some attributes of the partition, see CREATE TABLE
- If the user does not explicitly create a partition when creating a table, adding a partition by ALTER is not supported

- If the user uses list partition then they can add default partition to the table. The default partition will store all data not satisfying prior partition key's constraints.
    - ALTER TABLE table_name ADD PARTITION partition_name

2. Delete the partition

grammar:

```
DROP PARTITION [IF EXISTS] partition_name [FORCE]
```

Notice:

- At least one partition must be reserved for tables using partitioning.
- After executing DROP PARTITION for a period of time, the deleted partition can be recovered through the RECOVER statement. For details, see SQL Manual - Database Management - RECOVER Statement
- If you execute DROP PARTITION FORCE, the system will not check whether there are unfinished transactions in the partition, the partition will be deleted directly and cannot be recovered, this operation is generally not recommended

3. Modify the partition properties

grammar:

```
MODIFY PARTITION p1|(p1[, p2, ...]) SET ("key" = "value", ...)
```

illustrate:

- Currently supports modifying the following properties of partitions:
    - storage_medium -storage_cooldown_time
    - replication_num
    - in_memory
- For single-partition tables, partition_name is the same as the table name.

Example

1. Add partition, existing partition [MIN, 2013-01-01), add partition [2013-01-01, 2014-01-01), use default bucketing method

```
ALTER TABLE example_db.my_table
ADD PARTITION p1 VALUES LESS THAN ("2014-01-01");
```

2. Increase the partition and use the new number of buckets

```
ALTER TABLE example_db.my_table
ADD PARTITION p1 VALUES LESS THAN ("2015-01-01")
DISTRIBUTED BY HASH(k1) BUCKETS 20;
```

3. Increase the partition and use the new number of replicas

```
ALTER TABLE example_db.my_table
ADD PARTITION p1 VALUES LESS THAN ("2015-01-01")
("replication_num"="1");
```

4. Modify the number of partition replicas

```
ALTER TABLE example_db.my_table
MODIFY PARTITION p1 SET("replication_num"="1");
```

5. Batch modify the specified partition

```
ALTER TABLE example_db.my_table
MODIFY PARTITION (p1, p2, p4) SET("replication_num"="1");
```

6. Batch modify all partitions

```
ALTER TABLE example_db.my_table
MODIFY PARTITION (*) SET("storage_medium"="HDD");
```

7. Delete partition

```
ALTER TABLE example_db.my_table
DROP PARTITION p1;
```

8. Add a partition specifying upper and lower bounds

```
ALTER TABLE example_db.my_table
ADD PARTITION p1 VALUES [("2014-01-01"), ("2014-02-01"));
```

Keywords

```
ALTER, TABLE, PARTITION, ALTER TABLE
```

Best Practice

ALTER-TABLE-ROLLUP

ALTER-TABLE-ROLLUP

Name

ALTER TABLE ROLLUP

Description

This statement is used to perform a rollup modification operation on an existing table. The rollup is an asynchronous operation, and the task is returned when the task is submitted successfully. After that, you can use the SHOW ALTER command to view the progress.

grammar:

```
ALTER TABLE [database.]table alter_clause;
```

The alter_clause of rollup supports the following creation methods

1. Create a rollup index

grammar:

```
ADD ROLLUP rollup_name (column_name1, column_name2, ...)
[FROM from_index_name]
[PROPERTIES ("key"="value", ...)]
```

properties: Support setting timeout time, the default timeout time is 1 day.

2. Create rollup indexes in batches

grammar:

```
ADD ROLLUP [rollup_name (column_name1, column_name2, ...)
                    [FROM from_index_name]
                    [PROPERTIES ("key"="value", ...)],...]
```

Notice:

- If from_index_name is not specified, it will be created from base index by default
- Columns in rollup table must be columns already in from_index
- In properties, the storage format can be specified. For details, see CREATE TABLE

3. Delete rollup index

grammar:

```
DROP ROLLUP rollup_name [PROPERTIES ("key"="value", ...)]
```

4. Batch delete rollup index

grammar:

```
DROP ROLLUP [rollup_name [PROPERTIES ("key"="value", ...)],...]
```

Notice:

- cannot delete base index

Example

1. Create index: example_rollup_index, based on base index (k1,k2,k3,v1,v2). Columnar storage.

```
ALTER TABLE example_db.my_table
ADD ROLLUP example_rollup_index(k1, k3, v1, v2);
```

2. Create index: example_rollup_index2, based on example_rollup_index (k1,k3,v1,v2)

```
ALTER TABLE example_db.my_table
ADD ROLLUP example_rollup_index2 (k1, v1)
FROM example_rollup_index;
```

3. Create index: example_rollup_index3, based on base index (k1,k2,k3,v1), with a custom rollup timeout of one hour.

```
ALTER TABLE example_db.my_table
ADD ROLLUP example_rollup_index(k1, k3, v1)
PROPERTIES("timeout" = "3600");
```

4. Delete index: example_rollup_index2

```
ALTER TABLE example_db.my_table
DROP ROLLUP example_rollup_index2;
```

5. Batch Delete rollup index

```
ALTER TABLE example_db.my_table
DROP ROLLUP example_rollup_index2,example_rollup_index3;
```

4. Keywords

```
ALTER, TABLE, ROLLUP, ALTER TABLE
```

Best Practice

ALTER-TABLE-RENAME

ALTER-TABLE-RENAME

Name

ALTER TABLE RENAME

Description

This statement is used to rename certain names of existing table properties. This operation is synchronous, and the return of the command indicates the completion of the execution.

grammar:

```
ALTER TABLE [database.]table alter_clause;
```

The alter_clause of rename supports modification of the following names

1. Modify the table name

grammar:

```
RENAME new_table_name;
```

2. Modify the rollup index name

grammar:

```
RENAME ROLLUP old_rollup_name new_rollup_name;
```

3. Modify the partition name

grammar:

```
RENAME PARTITION old_partition_name new_partition_name;
```

4. Modify the column name

> **Tips**
>
> This feature is supported since the Apache Doris 1/2 version

Modify the column name

grammar:

```
RENAME COLUMN old_column_name new_column_name;
```

Notice: - When creating a table, you need to set ‘light_schema_change=true’ in the property.

Example

1. Modify the table named table1 to table2

```
ALTER TABLE table1 RENAME table2;
```

2. Modify the rollup index named rollup1 in the table example_table to rollup2

```
ALTER TABLE example_table RENAME ROLLUP rollup1 rollup2;
```

3. Modify the partition named p1 in the table example_table to p2

```
ALTER TABLE example_table RENAME PARTITION p1 p2;
```

4. Modify the column named c1 in the table example_table to c2

```
ALTER TABLE example_table RENAME COLUMN c1 c2;
```

Keywords

```
ALTER, TABLE, RENAME, ALTER TABLE
```

Best Practice

ALTER-TABLE-REPLACE

ALTER-TABLE-REPLACE

Name

ALTER TABLE REPLACE

Description

Atomic substitution of two tables. This operation applies only to OLAP tables.

```
ALTER TABLE [db.]tbl1 REPLACE WITH TABLE tbl2
[PROPERTIES('swap' = 'true')];
```

Replace table tbl1 with table tbl2.

If the `swap` parameter is `true`, the data in the table named `tbl1` will be the data in the original table named `tbl2` after the replacement. The data in the table named `tbl2` is the data in the original `tbl1` table. That is, two tables of data have been swapped.

If the `swap` parameter is `false`, the data in the `tbl1` table will be the data in the `tbl2` table after the replacement. The table named `tbl2` is deleted.

Theory

The replace table function actually turns the following set of operations into an atomic operation.

If you want to replace table A with table B and `swap` is `true`, do the following:

1. Rename table B as table A.
2. Rename table A as table B.

If `swap` is `false`, do as follows:

1. Delete table A.
2. Rename table B as table A.

Notice

1. The default `swap` parameter is `true`. That is, a table replacement operation is equivalent to an exchange of data between two tables.
2. If the `swap` parameter is set to false, the replaced table (table A) will be deleted and cannot be restored.
3. The replacement operation can only occur between two OLAP tables and does not check whether the table structure of the two tables is consistent.

4. The original permission Settings are not changed. Because the permission check is based on the table name.

Example

1. Atomic swap `tbl1` with `tbl2` without dropping any tables(Note: if you delete it, you actually delete tbl1 and rename tbl2 to tbl1.)

```
ALTER TABLE tbl1 REPLACE WITH TABLE tbl2;
```

or

```
ALTER TABLE tbl1 REPLACE WITH TABLE tbl2 PROPERTIES('swap' = 'true');
```

2. Atomic swap `tbl1` with `tbl2` and deleting the `tbl2` table(Keep `tbl1` and the data of the original `tbl2`)

```
ALTER TABLE tbl1 REPLACE WITH TABLE tbl2 PROPERTIES('swap' = 'false');
```

Keywords

```
ALTER, TABLE, REPLACE, ALTER TABLE
```

Best Practice

1. Atomic overlay write operations

In some cases, the user wants to be able to rewrite the data of a certain table, but if the data is deleted first and then imported, the data cannot be viewed for a period of time in between. At this time, the user can first use the `CREATE TABLE LIKE` statement to create a new table with the same structure, import the new data into the new table, and use the replacement operation to atomically replace the old table to achieve the goal. Atomic overwrite write operations at the partition level, see temp partition documentation.

ALTER-TABLE-PROPERTY

ALTER-TABLE-PROPERTY

Name

ALTER TABLE PROPERTY

Description

This statement is used to modify the properties of an existing table. This operation is synchronous, and the return of the command indicates the completion of the execution.

Modify the properties of the table, currently supports modifying the bloom filter column, the colocate_with attribute and the dynamic_partition attribute, the replication_num and default.replication_num.

grammar:

```
ALTER TABLE [database.]table alter_clause;
```

The alter_clause of property supports the following modification methods.

Note:

Can also be merged into the above schema change operation to modify, see the example below

1. Modify the bloom filter column of the table

```
ALTER TABLE example_db.my_table SET ("bloom_filter_columns"="k1,k2,k3");
```

Can also be incorporated into the schema change operation above (note that the syntax for multiple clauses is slightly different)

```
ALTER TABLE example_db.my_table
DROP COLUMN col2
PROPERTIES ("bloom_filter_columns"="k1,k2,k3");
```

2. Modify the Colocate property of the table

```
ALTER TABLE example_db.my_table set ("colocate_with" = "t1");
```

3. Change the bucketing method of the table from Hash Distribution to Random Distribution

```
ALTER TABLE example_db.my_table set ("distribution_type" = "random");
```

4. Modify the dynamic partition attribute of the table (support adding dynamic partition attribute to the table without dynamic partition attribute)

```
ALTER TABLE example_db.my_table set ("dynamic_partition.enable" = "false");
```

If you need to add dynamic partition attributes to tables without dynamic partition attributes, you need to specify all dynamic partition attributes (Note: adding dynamic partition attributes is not supported for non-partitioned tables)

```
ALTER TABLE example_db.my_table set (
  "dynamic_partition.enable" = "true",
  "dynamic_partition.time_unit" = "DAY",
  "dynamic_partition.end" = "3",
  "dynamic_partition.prefix" = "p",
  "dynamic_partition. buckets" = "32"
);
```

5. Modify the in_memory attribute of the table, only can set value ‘false’

```
ALTER TABLE example_db.my_table set ("in_memory" = "false");
```

6. Enable batch delete function

```
ALTER TABLE example_db.my_table ENABLE FEATURE "BATCH_DELETE";
```

Note:

- Only support unique tables
- Batch deletion is supported for old tables, while new tables are already supported when they are created

7. Enable the function of ensuring the import order according to the value of the sequence column

```
ALTER TABLE example_db.my_table ENABLE FEATURE "SEQUENCE_LOAD" WITH PROPERTIES (
  "function_column.sequence_type" = "Date"
);
```

Note:

- Only support unique tables
- The sequence_type is used to specify the type of the sequence column, which can be integral and time type
- Only the orderliness of newly imported data is supported. Historical data cannot be changed

8. Change the default number of buckets for the table to 50

```
ALTER TABLE example_db.my_table MODIFY DISTRIBUTION DISTRIBUTED BY HASH(k1) BUCKETS 50;
```

Note:

- Only support non colocate table with RANGE partition and HASH distribution

9. Modify table comments

```
ALTER TABLE example_db.my_table MODIFY COMMENT "new comment";
```

10. Modify column comments

```
ALTER TABLE example_db.my_table MODIFY COLUMN k1 COMMENT "k1", MODIFY COLUMN k2 COMMENT "k2";
```

11. Modify the engine type

Only the MySQL type can be changed to the ODBC type. The value of driver is the name of the driver in the odbc.init configuration.

```
ALTER TABLE example_db.mysql_table MODIFY ENGINE TO odbc PROPERTIES("driver" = "MySQL");
```

12. Modify the number of copies

```
ALTER TABLE example_db.mysql_table SET ("replication_num" = "2");
ALTER TABLE example_db.mysql_table SET ("default.replication_num" = "2");
ALTER TABLE example_db.mysql_table SET ("replication_allocation" = "tag.location.default: 1");
ALTER TABLE example_db.mysql_table SET ("default.replication_allocation" = "tag.location.default: 1");
```

Note: 1. The property with the default prefix indicates the default replica distribution for the modified table. This modification does not modify the current actual replica distribution of the table, but only affects the replica distribution of newly created partitions on the partitioned table. 2. For non-partitioned tables, modifying the replica distribution property without the default prefix will modify both the default replica distribution and the actual replica distribution of the table. That is, after the modification, through the `show create table` and `show partitions from tbl` statements, you can see that the replica distribution has been modified. changed. 3. For partitioned tables, the actual replica distribution of the table is at the partition level, that is, each partition has its own replica distribution, which can be viewed through the `show partitions from tbl` statement. If you want to modify the actual replica distribution, see `ALTER TABLE PARTITION`.

13. **[Experimental]** turn on `light_schema_change`

For tables that were not created with light_schema_change enabled, you can enable it by using the following statement.

```
ALTER TABLE example_db.mysql_table SET ("light_schema_change" = "true");
```

Example

1. Modify the bloom filter column of the table

```
ALTER TABLE example_db.my_table SET ("bloom_filter_columns"="k1,k2,k3");
```

Can also be incorporated into the schema change operation above (note that the syntax for multiple clauses is slightly different)

```
ALTER TABLE example_db.my_table
DROP COLUMN col2
PROPERTIES ("bloom_filter_columns"="k1,k2,k3");
```

2. Modify the Colocate property of the table

```
ALTER TABLE example_db.my_table set ("colocate_with" = "t1");
```

3. Change the bucketing method of the table from Hash Distribution to Random Distribution

```
ALTER TABLE example_db.my_table set ("distribution_type" = "random");
```

4. Modify the dynamic partition attribute of the table (support adding dynamic partition attribute to the table without dynamic partition attribute)

```
ALTER TABLE example_db.my_table set ("dynamic_partition.enable" = "false");
```

If you need to add dynamic partition attributes to tables without dynamic partition attributes, you need to specify all dynamic partition attributes (Note: adding dynamic partition attributes is not supported for non-partitioned tables)

```
ALTER TABLE example_db.my_table set ("dynamic_partition.enable" = "true", "dynamic_partition.time_unit" = "DAY",
"dynamic_partition.end" = "3", "dynamic_partition.prefix" = "p", "dynamic_partition. buckets" = "32");
```

5. Modify the in_memory attribute of the table, only can set value ‘false’

```
ALTER TABLE example_db.my_table set ("in_memory" = "false");
```

6. Enable batch delete function

```
ALTER TABLE example_db.my_table ENABLE FEATURE "BATCH_DELETE";
```

7. Enable the function of ensuring the import order according to the value of the sequence column

```
ALTER TABLE example_db.my_table ENABLE FEATURE "SEQUENCE_LOAD" WITH PROPERTIES ("function_column.sequence_type" = "Date");
```

8. Change the default number of buckets for the table to 50

```
ALTER TABLE example_db.my_table MODIFY DISTRIBUTION DISTRIBUTED BY HASH(k1) BUCKETS 50;
```

9. Modify table comments

```
ALTER TABLE example_db.my_table MODIFY COMMENT "new comment";
```

10. Modify column comments

```
ALTER TABLE example_db.my_table MODIFY COLUMN k1 COMMENT "k1", MODIFY COLUMN k2 COMMENT "k2";
```

11. Modify the engine type

```
ALTER TABLE example_db.mysql_table MODIFY ENGINE TO odbc PROPERTIES("driver" = "MySQL");
```

12. Add a cold and hot separation data migration strategy to the table

```
 ALTER TABLE create_table_not_have_policy set ("storage_policy" = "created_create_table_alter_policy");
```

NOTE：The table can be successfully added only if it hasn’t been associated with a storage policy. A table just can have one storage policy.

13. Add a hot and cold data migration strategy to the partition of the table

```
ALTER TABLE create_table_partition MODIFY PARTITION (*) SET("storage_policy"="created_create_table_partition_alter_policy");
```

NOTE：The table’s partition can be successfully added only if it hasn’t been associated with a storage policy. A table just can have one storage policy.

Keywords

```
ALTER, TABLE, PROPERTY, ALTER TABLE
```

Best Practice

ALTER-TABLE-COMMENT

ALTER-TABLE-COMMENT

Name

ALTER TABLE COMMENT

Description

This statement is used to modify the comment of an existing table. The operation is synchronous, and the command returns to indicate completion.

grammar：

```
ALTER TABLE [database.]table alter_clause;
```

1. Modify table comment

grammar：

```
MODIFY COMMENT "new table comment";
```

2. Modify column comment

grammar：

```
MODIFY COLUMN col1 COMMENT "new column comment";
```

Example

1. Change the table1’s comment to table1_comment

```
ALTER TABLE table1 MODIFY COMMENT "table1_comment";
```

2. Change the table1’s col1 comment to table1_comment

```
ALTER TABLE table1 MODIFY COLUMN col1 COMMENT "table1_col1_comment";
```

Keywords

```
ALTER, TABLE, COMMENT, ALTER TABLE
```

Best Practice

CANCEL-ALTER-TABLE

CANCEL-ALTER-TABLE

Name

CANCEL ALTER TABLE

Description

This statement is used to undo an ALTER operation.

1. Undo the ALTER TABLE COLUMN operation

grammar:

```
CANCEL ALTER TABLE COLUMN
FROM db_name.table_name
```

2. Undo the ALTER TABLE ROLLUP operation

grammar:

```
CANCEL ALTER TABLE ROLLUP
FROM db_name.table_name
```

3. Batch cancel rollup operations based on job id

grammar:

```
CANCEL ALTER TABLE ROLLUP
FROM db_name.table_name (jobid,...)
```

Notice:

- This command is an asynchronous operation. You need to use `show alter table rollup` to check the task status to confirm whether the execution is successful or not.

4. Undo the ALTER CLUSTER operation

grammar:

```
(To be implemented...)
```

Example

1. Undo the ALTER COLUMN operation on my_table.

[CANCEL ALTER TABLE COLUMN]

```
CANCEL ALTER TABLE COLUMN
FROM example_db.my_table;
```

1. Undo the ADD ROLLUP operation under my_table.

   [CANCEL ALTER TABLE ROLLUP]

```
CANCEL ALTER TABLE ROLLUP
FROM example_db.my_table;
```

1. Undo the ADD ROLLUP operation under my_table according to the job id.

   [CANCEL ALTER TABLE ROLLUP]

```
CANCEL ALTER TABLE ROLLUP
FROM example_db.my_table(12801,12802);
```

Keywords

```
CANCEL, ALTER, TABLE, CANCEL ALTER
```

Best Practice

ALTER-VIEW

ALTER-VIEW

Name

ALTER VIEW

Description

This statement is used to modify the definition of a view

grammar:

```
ALTER VIEW
[db_name.]view_name
(column1[ COMMENT "col comment"][, column2, ...])
AS query_stmt
```

illustrate:

- Views are all logical, and the data in them will not be stored on physical media. When querying, the view will be used as a subquery in the statement. Therefore, modifying the definition of the view is equivalent to modifying query_stmt.
- query_stmt is any supported SQL

Example

1. Modify the view example_view on example_db

```
ALTER VIEW example_db.example_view
(
c1 COMMENT "column 1",
c2 COMMENT "column 2",
c3 COMMENT "column 3"
)
AS SELECT k1, k2, SUM(v1) FROM example_table
GROUP BY k1, k2
```

Keywords

```
ALTER, VIEW
```

Best Practice

ALTER-POLICY

ALTER-POLICY

Name

ALTER STORAGE POLICY

Description

This statement is used to modify an existing cold and hot separation migration strategy. Only root or admin users can modify resources.

```
ALTER STORAGE POLICY  'policy_name'
PROPERTIES ("key"="value", ...);
```

Example

1. Modify the name to coolown_datetime Cold and hot separation data migration time point:

```
ALTER STORAGE POLICY has_test_policy_to_alter PROPERTIES("cooldown_datetime" = "2023-06-08 00:00:00");
```

2. Modify the name to coolown_countdown of hot and cold separation data migration of ttl

```
ALTER STORAGE POLICY has_test_policy_to_alter PROPERTIES ("cooldown_ttl" = "10000");
ALTER STORAGE POLICY has_test_policy_to_alter PROPERTIES ("cooldown_ttl" = "1h");
ALTER STORAGE POLICY has_test_policy_to_alter PROPERTIES ("cooldown_ttl" = "3d");
```

Keywords

```
ALTER, STORAGE, POLICY
```

Best Practice

ALTER-RESOURCE

ALTER-RESOURCE

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Name

ALTER RESOURCE

Description

This statement is used to modify an existing resource. Only the root or admin user can modify resources. Syntax:

```
ALTER RESOURCE 'resource_name'
PROPERTIES ("key"="value", ...);
```

Note: The resource type does not support modification.

Example

1. Modify the working directory of the Spark resource named spark0:

```
ALTER RESOURCE 'spark0' PROPERTIES ("working_dir" = "hdfs://127.0.0.1:10000/tmp/doris_new");
```

2. Modify the maximum number of connections to the S3 resource named remote_s3:

```
ALTER RESOURCE 'remote_s3' PROPERTIES ("s3.connection.maximum" = "100");
```

3. Modify information related to cold and hot separation S3 resources

- Support
  - `s3.access_key` s3 ak
  - `s3.secret_key` s3 sk
  - `s3.session_token` s3 token
  - `s3.connection.maximum` default 50
  - `s3.connection.timeout` default 1000ms
  - `s3.connection.request.timeout` default 3000ms
- Not Support
  - `s3.region`

- s3.bucket"
- s3.root.path
- s3.endpoint

```
ALTER RESOURCE "showPolicy_1_resource" PROPERTIES("s3.connection.maximum" = "1111");
```

Keywords

```
ALTER, RESOURCE
```

Best Practice

ALTER-COLOCATE-GROUP

ALTER-COLOCATE-GROUP

Name

ALTER COLOCATE GROUP

Description

This statement is used to modify the colocation group.

Syntax:

```
ALTER COLOCATE GROUP [database.]group
SET (
    property_list
);
```

NOTE:

1. If the colocate group is global, that is, its name starts with `__global__`, then it does not belong to any database;
2. property_list is a colocation group attribute, currently only supports modifying `replication_num` and `replication_allocation`. After modifying these two attributes of the colocation group, at the same time, change the attribute `default.replication_allocation`, the attribute `dynamic.replication_allocation` of the table of the group, and the `replication_allocation` of the existing partition to be the same as it.

Example

1. Modify the number of copies of a global group

    ```
    # Set "colocate_with" = "__global__foo" when creating the table

    ALTER COLOCATE GROUP __global__foo
    SET (
        "replication_num"="1"
    );
    ```

2. Modify the number of copies of a non-global group

```
# Set "colocate_with" = "bar" when creating the table, and the Database is "example_db"

ALTER COLOCATE GROUP example_db.bar
SET (
    "replication_num"="1"
);
```

Keywords

```sql
ALTER, COLOCATE, GROUP
```

Best Practice

ALTER-WORKLOAD-GROUP

ALTER-WORKLOAD-GROUP

Name

ALTER WORKLOAD GROUP

Description

This statement is used to modify the workload group.

Syntax:

```
ALTER WORKLOAD GROUP "rg_name"
PROPERTIES (
    property_list
);
```

NOTE:

- Modify the memory_limit property in such a way that the sum of all memory_limit values does not exceed 100%;
- Support modifying some properties, for example, if only cpu_share is modified, just fill in cpu_share in properties.

Example

1. Modify the workload group named g1:

   ```
   alter workload group g1
   properties (
       "cpu_share"="30",
       "memory_limit"="30%"
   );
   ```

Keywords

```
ALTER, WORKLOAD, GROUP
```

Best Practice

ALTER-SQL-BLOCK-RULE

ALTER-SQL-BLOCK-RULE

Name

ALTER SQL BLOCK RULE

Description

Modify SQL blocking rules to allow modification of each item such as sql/sqlHash/partition_num/tablet_num/cardinality/global/enable.

grammar:

```
ALTER SQL_BLOCK_RULE rule_name
[PROPERTIES ("key"="value", ...)];
```

illustrate:

- sql and sqlHash cannot be set at the same time. This means that if a rule sets sql or sqlHash, the other attribute cannot be modified;
- sql/sqlHash and partition_num/tablet_num/cardinality cannot be set at the same time. For example, if a rule sets partition_num, then sql or sqlHash cannot be modified;

Example

1. Modify according to SQL properties

```
ALTER SQL_BLOCK_RULE test_rule PROPERTIES("sql"="select \\* from test_table","enable"="true")
```

2. If a rule sets partition_num, then sql or sqlHash cannot be modified

```
ALTER SQL_BLOCK_RULE test_rule2 PROPERTIES("partition_num" = "10","tablet_num"="300","enable"="true")
```

Keywords

```
ALTER,SQL_BLOCK_RULE
```

Best Practice

DROP-CATALOG

DROP-CATALOG

Name

CREATE CATALOG

Description

This statement is used to delete the external catalog.

Syntax:

```
DROP CATALOG [I EXISTS] catalog_name;
```

Example

1. Drop catalog hive

    ```
    DROP CATALOG hive;
    ```

Keywords

DROP, CATALOG

Best Practice

DROP-DATABASE

DROP-DATABASE

Name

DOPR DATABASE

Description

This statement is used to delete the database (database) grammar:

```
DROP DATABASE [IF EXISTS] db_name [FORCE];
```

illustrate:

- During the execution of DROP DATABASE, the deleted database can be recovered through the RECOVER statement. See the RECOVER statement for details
- If you execute DROP DATABASE FORCE, the system will not check the database for unfinished transactions, the database will be deleted directly and cannot be recovered, this operation is generally not recommended

Example

1. Delete the database db_test

   ```
   DROP DATABASE db_test;
   ```

Keywords

```
DROP, DATABASE
```

Best Practice

DROP-TABLE

DROP-TABLE

Name

DROP TABLE

Description

This statement is used to drop a table. grammar:

```
DROP TABLE [IF EXISTS] [db_name.]table_name [FORCE];
```

illustrate:

- After executing DROP TABLE for a period of time, the dropped table can be recovered through the RECOVER statement. See RECOVER statement for details
- If you execute DROP TABLE FORCE, the system will not check whether there are unfinished transactions in the table, the table will be deleted directly and cannot be recovered, this operation is generally not recommended

Example

1. Delete a table

   ```
   DROP TABLE my_table;
   ```

2. If it exists, delete the table of the specified database

   ```
   DROP TABLE IF EXISTS example_db.my_table;
   ```

Keywords

```
DROP, TABLE
```

Best Practice

DROP-INDEX

DROP-INDEX

Name

DROP INDEX

Description

This statement is used to delete the index of the specified name from a table. grammar:

```
DROP INDEX [IF EXISTS] index_name ON [db_name.]table_name;
```

Example

1. Delete the index

   ```
   DROP INDEX [IF NOT EXISTS] index_name ON table1 ;
   ```

Keywords

```
DROP, INDEX
```

Best Practice

DROP-MATERIALIZED-VIEW

DROP-MATERIALIZED-VIEW

Name

DROP MATERIALIZED VIEW

Description

This statement is used to drop a materialized view. Synchronous syntax

grammar:

```
DROP MATERIALIZED VIEW [IF EXISTS] mv_name ON table_name;
```

1. IF EXISTS: Do not throw an error if the materialized view does not exist. If this keyword is not declared, an error will be reported if the materialized view does not exist.
2. mv_name: The name of the materialized view to delete. Required.
3. table_name: The name of the table to which the materialized view to be deleted belongs. Required.

Example

The table structure is

```
mysql> desc all_type_table all;
+----------------+-------+----------+------+------ -+---------+-------+
| IndexName | Field | Type | Null | Key | Default | Extra |
+----------------+-------+----------+------+------ -+---------+-------+
| all_type_table | k1 | TINYINT | Yes | true | N/A | |
| | k2 | SMALLINT | Yes | false | N/A | NONE |
| | k3 | INT | Yes | false | N/A | NONE |
| | k4 | BIGINT | Yes | false | N/A | NONE |
| | k5 | LARGEINT | Yes | false | N/A | NONE |
| | k6 | FLOAT | Yes | false | N/A | NONE |
| | k7 | DOUBLE | Yes | false | N/A | NONE |
| | | | | | | | |
| k1_sumk2 | k1 | TINYINT | Yes | true | N/A | |
| | k2 | SMALLINT | Yes | false | N/A | SUM |
+----------------+-------+----------+------+------ -+---------+-------+
```

1. Drop the materialized view named k1_sumk2 of the table all_type_table

   ```
   drop materialized view k1_sumk2 on all_type_table;
   ```

   The table structure after the materialized view is deleted

```
+----------------+-------+----------+------+------ -+---------+-------+
| IndexName | Field | Type | Null | Key | Default | Extra |
+----------------+-------+----------+------+------ -+---------+-------+
| all_type_table | k1 | TINYINT | Yes | true | N/A | |
| | k2 | SMALLINT | Yes | false | N/A | NONE |
| | k3 | INT | Yes | false | N/A | NONE |
| | k4 | BIGINT | Yes | false | N/A | NONE |
| | k5 | LARGEINT | Yes | false | N/A | NONE |
| | k6 | FLOAT | Yes | false | N/A | NONE |
| | k7 | DOUBLE | Yes | false | N/A | NONE |
+----------------+-------+----------+------+------ -+---------+-------+
```

2. Drop a non-existent materialized view in the table all_type_table

```
drop materialized view k1_k2 on all_type_table;
ERROR 1064 (HY000): errCode = 2, detailMessage = Materialized view [k1_k2] does not exist in table [all_type_table]
```

   The delete request reports an error directly

3. Delete the materialized view k1_k2 in the table all_type_table, if it does not exist, no error will be reported.

```
drop materialized view if exists k1_k2 on all_type_table;
Query OK, 0 rows affected (0.00 sec)
```

   If it exists, delete it, if it does not exist, no error is reported.

Keywords

```
DROP, MATERIALIZED, VIEW
```

Best Practice

DROP-FUNCTION

Description

Delete a custom function.

Syntax

```
DROP [ GLOBAL ] <function_name> ( <arg_type> )
```

Required Parameters

1. `<function_name>`

> Specifies the name of the function to delete.
>
> The function name needs to be exactly the same as the function name when the function was created.

2. `<arg_type>`

> Specifies the argument list for the function to be deleted.
>
> Parameter List Location You need to enter the data type of the location parameter.

Optional Parameters

1.`GLOBAL`

> GLOBAL is an optional parameter.
>
> If GLOBAL is set, the function is searched for globally and deleted.
>
> If GLOABL is not entered, the function is searched for in the current database and deleted.

Access Control Requirements

The user who executes this SQL command must have at least the following permissions:

Privilege	Object	Notes
ADMIN_PRIV	Custom function	DROP is an administrative operation

Usage Notes

- A function can be deleted only when its name and parameter types are identical

Examples

```
DROP FUNCTION my_add(INT, INT)
```

```
DROP GLOBAL FUNCTION my_add(INT, INT)
```

DROP-FILE

DROP-FILE

Name

DROP FILE

Description

This statement is used to delete an uploaded file.

grammar:

```
DROP FILE "file_name" [FROM database]
[properties]
```

illustrate:

- file_name: file name.
- database: a db to which the file belongs, if not specified, the db of the current session is used.
- properties supports the following parameters:
    - `catalog`: Required. The category the file belongs to.

Example

1. Delete the file ca.pem

    ```
    DROP FILE "ca.pem" properties("catalog" = "kafka");
    ```

Keywords

```
DROP, FILE
```

Best Practice

DROP-POLICY

Name

DROP POLICY

Description

drop policy for row or storage

ROW POLICY

Grammar:

1. Drop row policy

```
DROP ROW POLICY test_row_policy_1 on table1 [FOR user| ROLE role];
```

2. Drop storage policy

```
DROP STORAGE POLICY policy_name1
```

Example

1. Drop the row policy for table1 named test_row_policy_1

```
DROP ROW POLICY test_row_policy_1 on table1
```

2. Drop the row policy for table1 using by user test

```
DROP ROW POLICY test_row_policy_1 on table1 for test
```

3. Drop the row policy for table1 using by role1

```
DROP ROW POLICY test_row_policy_1 on table1 for role role1
```

4. Drop the storage policy named policy_name1

```
DROP STORAGE POLICY policy_name1
```

Keywords

```
DROP, POLICY
```

Best Practice

DROP-ENCRYPT-KEY

DROP-ENCRYPT-KEY

Name

DROP ENCRYPTKEY

Description

grammar:

```
DROP ENCRYPTKEY key_name
```

Parameter Description:

- `key_name`: The name of the key to delete, can include the name of the database. For example: `db1.my_key`.

Delete a custom key. The name of the key is exactly the same to be deleted.

Executing this command requires the user to have `ADMIN` privileges.

Example

1. Delete a key

   ```
   DROP ENCRYPTKEY my_key;
   ```

Keywords

```
DROP, ENCRYPT, KEY
```

Best Practice

DROP-RESOURCE

DROP-RESOURCE

Name

DROP RESOURCE

Description

This statement is used to delete an existing resource. Only the root or admin user can delete resources. grammar:

```
DROP RESOURCE 'resource_name'
```

Note: ODBC/S3 resources in use cannot be deleted.

Example

1. Delete the Spark resource named spark0:

   ```
   DROP RESOURCE 'spark0';
   ```

Keywords

```
DROP, RESOURCE
```

Best Practice

DROP-WORKLOAD-GROUP

DROP-WORKLOAD-GROUP

Name

DROP WORKLOAD GROUP

Description

This statement is used to delete a workload group.

```
DROP WORKLOAD GROUP [IF EXISTS] 'rg_name'
```

Example

1. Delete the workload group named g1:

   ```
   drop workload group if exists g1;
   ```

Keywords

```
DROP, WORKLOAD, GROUP
```

Best Practice

DROP-DATABASE

DROP-SQL-BLOCK-RULE

Name

DROP SQL BLOCK RULE

Description

Delete SQL blocking rules, support multiple rules, separated by ,

grammar:

```
DROP SQL_BLOCK_RULE test_rule1,...
```

Example

1. Delete the test_rule1 and test_rule2 blocking rules

    ```
    mysql> DROP SQL_BLOCK_RULE test_rule1,test_rule2;
    Query OK, 0 rows affected (0.00 sec)
    ```

Keywords

```
DROP, SQL_BLOCK_RULE
```

Best Practice

TRUNCATE-TABLE

TRUNCATE-TABLE

Name

TRUNCATE TABLE

Description

This statement is used to clear the data of the specified table and partition grammar:

```
TRUNCATE TABLE [db.]tbl[ PARTITION(p1, p2, ...)];
```

illustrate:

- The statement clears the data, but leaves the table or partition.
- Unlike DELETE, this statement can only clear the specified table or partition as a whole, and cannot add filter conditions.
- Unlike DELETE, using this method to clear data will not affect query performance.
- The data deleted by this operation cannot be recovered.
- When using this command, the table status needs to be NORMAL, that is, operations such as SCHEMA CHANGE are not allowed.
- This command may cause the ongoing load to fail

Example

1. Clear the table tbl under example_db

   ```
   TRUNCATE TABLE example_db.tbl;
   ```

2. Empty p1 and p2 partitions of table tbl

   ```
   TRUNCATE TABLE tbl PARTITION(p1, p2);
   ```

Keywords

```
TRUNCATE, TABLE
```

Best Practice

CREATE-REPOSITORY

CREATE-REPOSITORY

Name

CREATE REPOSITORY

Description

This statement is used to create a repository. Repositories are used for backup or restore. Only root or superuser users can create repositories.

grammar:

```
CREATE [READ ONLY] REPOSITORY `repo_name`
WITH [S3|hdfs]
ON LOCATION `repo_location`
PROPERTIES ("key"="value", ...);
```

illustrate:

- Creation of repositories, accessing cloud storage directly through AWS s3 protocol, or accessing HDFS directly.
- If it is a read-only repository, restores can only be done on the repository. If not, backup and restore operations are available.
- PROPERTIES are different according to different types of S3 or hdfs, see the example for details.
- ON LOCATION : if it is S3 , here followed by the Bucket Name.

Example

1. Create a repository named s3_repo.

```
CREATE REPOSITORY `s3_repo`
WITH S3
ON LOCATION "s3://s3-repo"
PROPERTIES
(
    "s3.endpoint" = "http://s3-REGION.amazonaws.com",
    "s3.region" = "s3-REGION",
    "s3.access_key" = "AWS_ACCESS_KEY",
    "s3.secret_key"="AWS_SECRET_KEY",
    "s3.region" = "REGION"
);
```

2. Create a repository named hdfs_repo.

```
CREATE REPOSITORY `hdfs_repo`
WITH hdfs
```

```
ON LOCATION "hdfs://hadoop-name-node:54310/path/to/repo/"
PROPERTIES
(
    "fs.defaultFS"="hdfs://hadoop-name-node:54310",
    "hadoop.username"="user"
);

### Keywords
```

6. Create a repository named minio_repo to link minio storage directly through the s3 protocol.

```
CREATE REPOSITORY `minio_repo`
WITH S3
ON LOCATION "s3://minio_repo"
PROPERTIES
(
    "s3.endpoint" = "http://minio.com",
    "s3.access_key" = "MINIO_USER",
    "s3.secret_key"="MINIO_PASSWORD",
    "s3.region" = "REGION",
    "use_path_style" = "true"
);
```

7. Create a repository named minio_repo via temporary security credentials.

```
CREATE REPOSITORY `minio_repo`
WITH S3
ON LOCATION "s3://minio_repo"
PROPERTIES
(
    "s3.endpoint" = "AWS_ENDPOINT",
    "s3.access_key" = "AWS_TEMP_ACCESS_KEY",
    "s3.secret_key" = "AWS_TEMP_SECRET_KEY",
    "s3.session_token" = "AWS_TEMP_TOKEN",
    "s3.region" = "AWS_REGION"
)
```

8. Create repository using Tencent COS

```
CREATE REPOSITORY `cos_repo`
WITH S3
ON LOCATION "s3://backet1/"
PROPERTIES
(
    "s3.access_key" = "ak",
    "s3.secret_key" = "sk",
    "s3.endpoint" = "http://cos.ap-beijing.myqcloud.com",
    "s3.region" = "ap-beijing"
);
```

Keywords

CREATE, REPOSITORY

Best Practice

1. A cluster can create multiple warehouses. Only users with ADMIN privileges can create repositories.
2. Any user can view the created repositories through the SHOW REPOSITORIES command.
3. When performing data migration operations, it is necessary to create the exact same warehouse in the source cluster and the destination cluster, so that the destination cluster can view the data snapshots backed up by the source cluster through this warehouse.

DROP-REPOSITORY

DROP-REPOSITORY

Name

DROP REPOSITORY

Description

This statement is used to delete a created repository. Only root or superuser users can delete repositories.

grammar:

```
DROP REPOSITORY `repo_name`;
```

illustrate:

- Deleting a warehouse just deletes the warehouse's mapping in Doris, not the actual warehouse data. Once deleted, it can be mapped to the repository again by specifying the same LOCATION.

Example

1. Delete the repository named example_repo:

```
DROP REPOSITORY `example_repo`;
```

Keywords

```
DROP, REPOSITORY
```

Best Practice

BACKUP

BACKUP

Name

BACKUP

Description

This statement is used to back up the data under the specified database. This command is an asynchronous operation. After the submission is successful, you need to check the progress through the SHOW BACKUP command. Only backing up tables of type OLAP is supported.

Only root or superuser users can create repositories.

grammar:

```
BACKUP SNAPSHOT [db_name].{snapshot_name}
TO `repository_name`
[ON|EXCLUDE] (
    `table_name` [PARTITION (`p1`, ...)],
    ...
)
PROPERTIES ("key"="value", ...);
```

illustrate:

- There can only be one executing BACKUP or RESTORE task under the same database.
- The ON clause identifies the tables and partitions that need to be backed up. If no partition is specified, all partitions of the table are backed up by default
- Tables and partitions that do not require backup are identified in the EXCLUDE clause. Back up all partition data for all tables in this database except the specified table or partition.
- PROPERTIES currently supports the following properties:
    - “type” = “full”: indicates that this is a full update (default)
    - “timeout” = “3600”: The task timeout period, the default is one day. in seconds.

Example

1. Fully backup the table example_tbl under example_db to the warehouse example_repo:

```
BACKUP SNAPSHOT example_db.snapshot_label1
TO example_repo
ON (example_tbl)
PROPERTIES ("type" = "full");
```

2. Under the full backup example_db, the p1, p2 partitions of the table example_tbl, and the table example_tbl2 to the warehouse example_repo:

```
BACKUP SNAPSHOT example_db.snapshot_label2
TO example_repo
ON
(
    example_tbl PARTITION (p1,p2),
    example_tbl2
);
```

3. Full backup of all tables except table example_tbl under example_db to warehouse example_repo:

```
BACKUP SNAPSHOT example_db.snapshot_label3
TO example_repo
EXCLUDE (example_tbl);
```

4. Fully back up tables under example_db to the repository example_repo:

```
BACKUP SNAPSHOT example_db.snapshot_label3
TO example_repo;
```

Keywords

BACKUP

Best Practice

1. Only one backup operation can be performed under the same database.
2. The backup operation will back up the underlying table and Synchronous materialized view of the specified table or partition, and only one replica will be backed up. Asynchronous materialized view is not supported.
3. Efficiency of backup operations

   The efficiency of backup operations depends on the amount of data, the number of Compute Nodes, and the number of files. Each Compute Node where the backup data shard is located will participate in the upload phase of the backup operation. The greater the number of nodes, the higher the upload efficiency.

   The amount of file data refers only to the number of shards, and the number of files in each shard. If there are many shards, or there are many small files in the shards, the backup operation time may be increased.

CANCEL-BACKUP

CANCEL-BACKUP

Name

CANCEL BACKUP

Description

This statement is used to cancel an ongoing BACKUP task.

grammar:

```
CANCEL BACKUP FROM db_name;
```

Example

1. Cancel the BACKUP task under example_db.

```
CANCEL BACKUP FROM example_db;
```

Keywords

```
 CANCEL, BACKUP
```

Best Practice

RESTORE

RESTORE

Name

RESTORE

Description

This statement is used to restore the data backed up by the BACKUP command to the specified database. This command is an asynchronous operation. After the submission is successful, you need to check the progress through the SHOW RESTORE command. Restoring tables of type OLAP is only supported.

grammar:

```
RESTORE SNAPSHOT [db_name].{snapshot_name}
FROM `repository_name`
[ON|EXCLUDE] (
    `table_name` [PARTITION (`p1`, ...)] [AS `tbl_alias`],
    ...
)
PROPERTIES ("key"="value", ...);
```

illustrate:

- There can only be one executing BACKUP or RESTORE task under the same database.
- The tables and partitions that need to be restored are identified in the ON clause. If no partition is specified, all partitions of the table are restored by default. The specified table and partition must already exist in the warehouse backup.
- Tables and partitions that do not require recovery are identified in the EXCLUDE clause. All partitions of all other tables in the warehouse except the specified table or partition will be restored.
- The table name backed up in the warehouse can be restored to a new table through the AS statement. But the new table name cannot already exist in the database. The partition name cannot be modified.
- You can restore the backed up tables in the warehouse to replace the existing tables of the same name in the database, but you must ensure that the table structures of the two tables are exactly the same. The table structure includes: table name, column, partition, Rollup, etc.
- You can specify some partitions of the recovery table, and the system will check whether the partition Range or List can match.
- PROPERTIES currently supports the following properties:
    - “backup_timestamp” = “2018-05-04-16-45-08”: Specifies which time version of the corresponding backup to restore, required. This information can be obtained with the `SHOW SNAPSHOT ON repo;` statement.
    - “replication_num” = “3”: Specifies the number of replicas for the restored table or partition. Default is 3. If restoring an existing table or partition, the number of replicas must be the same as the number of replicas of the existing table or partition. At the same time, there must be enough hosts to accommodate multiple replicas.

- “reserve_replica” = “true”: Default is false. When this property is true, the replication_num property is ignored and the restored table or partition will have the same number of replication as before the backup. Supports multiple tables or multiple partitions within a table with different replication number.
- “reserve_dynamic_partition_enable” = “true”: Default is false. When this property is true, the restored table will have the same value of ‘dynamic_partition_enable’ as before the backup. if this property is not true, the restored table will set ‘dynamic_partition_enable=false’.
- “timeout” = “3600”: The task timeout period, the default is one day. in seconds.
- “meta_version” = 40: Use the specified meta_version to read the previously backed up metadata. Note that this parameter is used as a temporary solution and is only used to restore the data backed up by the old version of Doris. The latest version of the backup data already contains the meta version, no need to specify it.

> **Tips**
>
> This feature is supported since the Apache Doris 2.0.15 version

- “clean_tables” : Indicates whether to clean up tables that do not belong to the restore target. For example, if the target db before the restore has tables that are not present in the snapshot, specifying `clean_tables` can drop these extra tables and move them into the recycle bin during the restore.
- “clean_partitions” : Indicates whether to clean up partitions that do not belong to the restore target. For example, if the target table before the restore has partitions that are not present in the snapshot, specifying `clean_partitions` can drop these extra partitions and move them into the recycle bin during the restore.

Example

1. Restore the table backup_tbl in backup snapshot_1 from example_repo to database example_db1, the time version is “2018-05-04-16-45-08”. Revert to 1 copy:

```
RESTORE SNAPSHOT example_db1.`snapshot_1`
FROM `example_repo`
ON ( `backup_tbl` )
PROPERTIES
(
    "backup_timestamp"="2018-05-04-16-45-08",
    "replication_num" = "1"
);
```

2. Restore partitions p1, p2 of table backup_tbl in backup snapshot_2 from example_repo, and table backup_tbl2 to database example_db1, rename it to new_tbl, and the time version is “2018-05-04-17-11-01”. The default reverts to 3 replicas:

```
RESTORE SNAPSHOT example_db1.`snapshot_2`
FROM `example_repo`
ON
(
    `backup_tbl` PARTITION (`p1`, `p2`),
    `backup_tbl2` AS `new_tbl`
)
```

```
PROPERTIES
(
    "backup_timestamp"="2018-05-04-17-11-01"
);
```

3. Restore all tables except for table backup_tbl in backup snapshot_3 from example_repo to database example_db1, the time version is “2018-05-04-18-12-18”.

```
RESTORE SNAPSHOT example_db1.`snapshot_3`
FROM `example_repo`
EXCLUDE ( `backup_tbl` )
PROPERTIES
(
    "backup_timestamp"="2018-05-04-18-12-18"
);
```

Keywords

```
RESTORE
```

Best Practice

1. There can only be one ongoing recovery operation under the same database.
2. The table backed up in the warehouse can be restored and replaced with the existing table of the same name in the database, but the table structure of the two tables must be completely consistent. The table structure includes: table name, columns, partitions, materialized views, and so on.
3. When specifying a partial partition of the recovery table, the system will check whether the partition range can match.
4. Efficiency of recovery operations:

   In the case of the same cluster size, the time-consuming of the restore operation is basically the same as the time-consuming of the backup operation. If you want to speed up the recovery operation, you can first restore only one copy by setting the `replication_num` parameter, and then adjust the number of copies by ALTER TABLE PROPERTY, complete the copy.

CANCEL-RESTORE

CANCEL-RESTORE

Name

CANCEL RESTORE

Description

This statement is used to cancel an ongoing RESTORE task.

grammar:

```
CANCEL RESTORE FROM db_name;
```

Notice:

- When cancellation is around a COMMIT or later stage of recovery, the table being recovered may be rendered inaccessible. At this time, data recovery can only be performed by executing the recovery job again.

Example

1. Cancel the RESTORE task under example_db.

```
CANCEL RESTORE FROM example_db;
```

Keywords

```
CANCEL, RESTORE
```

Best Practice

STREAM-LOAD

STREAM-LOAD

Name

STREAM LOAD

Description

stream-load: load data to table in streaming

```
curl --location-trusted -u user:passwd [-H ""...] -T data.file -XPUT http://fe_host:http_port/api/{db}/{table}/_stream_load
```

This statement is used to import data into the specified table. The difference from ordinary Load is that this import method is synchronous import.

This import method can still ensure the atomicity of a batch of import tasks, either all data is imported successfully or all of them fail.

This operation will update the data of the rollup table related to this base table at the same time.

This is a synchronous operation. After the entire data import work is completed, the import result is returned to the user.

Currently, HTTP chunked and non-chunked uploads are supported. For non-chunked methods, Content-Length must be used to indicate the length of the uploaded content, which can ensure the integrity of the data.

In addition, it is best for users to set the content of the Expect Header field to 100-continue, which can avoid unnecessary data transmission in some error scenarios.

Parameter introduction: Users can pass in import parameters through the Header part of HTTP

1. label: The label imported once, the data of the same label cannot be imported multiple times. Users can avoid the problem of duplicate data import by specifying Label.

   Currently, Doris retains the most recent successful label within 30 minutes.

2. column_separator: used to specify the column separator in the import file, the default is If it is an invisible character, you need to add as a prefix and use hexadecimal to represent the separator.

   For example, the separator of the hive file needs to be specified as -H “column_separator:”.

   You can use a combination of multiple characters as column separators.

3. line_delimiter: used to specify the newline character in the imported file, the default is . Combinations of multiple characters can be used as newlines.

4. columns: used to specify the correspondence between the columns in the import file and the columns in the table. If the column in the source file corresponds exactly to the content in the table, then there is no need to specify the content of this field.

   If the source file does not correspond to the table schema, then this field is required for some data conversion. There are two forms of column, one is directly corresponding to the field in the imported file, which is directly represented by the field name;

   One is derived column, the syntax is `column_name` = expression. Give a few examples to help understand.

   Example 1: There are 3 columns “c1, c2, c3” in the table, and the three columns in the source file correspond to “c3, c2, c1” at a time; then you need to specify -H “columns: c3, c2, c1”

   Example 2: There are 3 columns “c1, c2, c3” in the table, the first three columns in the source file correspond in turn, but there is more than 1 column; then you need to specify -H “columns: c1, c2, c3, xxx”;

   The last column can be arbitrarily assigned a name and placeholder

   Example 3: There are three columns “year, month, day” in the table, and there is only one time column in the source file, which is in “2018-06-01 01:02:03” format;

   Then you can specify -H “columns: col, year = year(col), month=month(col), day=day(col)” to complete the import

5. where: used to extract part of the data. If the user needs to filter out the unnecessary data, he can achieve this by setting this option.

   Example 1: Only import data greater than k1 column equal to 20180601, then you can specify -H “where: k1 = 20180601” when importing

6. max_filter_ratio: The maximum tolerable data ratio that can be filtered (for reasons such as data irregularity). Zero tolerance by default. Data irregularities do not include rows filtered out by where conditions.

7. partitions: used to specify the partition designed for this import. If the user can determine the partition corresponding to the data, it is recommended to specify this item. Data that does not satisfy these partitions will be filtered out.

   For example, specify import to p1, p2 partition, -H “partitions: p1, p2”

8. timeout: Specify the import timeout. in seconds. The default is 600 seconds. The setting range is from 1 second to 259200 seconds.

9. strict_mode: The user specifies whether to enable strict mode for this import. The default is off. The enable mode is -H “strict_mode: true”.

10. timezone: Specifies the timezone used for this import. The default is “+08:00”. This variable replaces the session variable `time_zone` in this import transaction. See the section “Importing with timezones” in Best Practice for more information.

11. exec_mem_limit: Load memory limit. Default is 2GB. The unit is bytes.

12. format: Specify load data format, support csv, json, csv_with_names(support csv file line header filter), csv_with_names_and_types(support csv file first two lines filter), parquet, orc, default is csv.

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

13. jsonpaths: The way of importing json is divided into: simple mode and matching mode.

    Simple mode: The simple mode is not set the jsonpaths parameter. In this mode, the json data is required to be an object type, for example:

    ```
    {"k1":1, "k2":2, "k3":"hello"}, where k1, k2, k3 are column names.
    ```

    Matching mode: It is relatively complex for json data and needs to match the corresponding value through the jsonpaths parameter.

14. strip_outer_array: Boolean type, true indicates that the json data starts with an array object and flattens the array object, the default value is false. E.g:

    ```
    [
     {"k1" : 1, "v1" : 2},
     {"k1" : 3, "v1" : 4}
    ]
    ```

    When strip_outer_array is true, the final import into doris will generate two rows of data.

15. json_root: json_root is a valid jsonpath string, used to specify the root node of the json document, the default value is “”.

16. merge_type: The merge type of data, which supports three types: APPEND, DELETE, and MERGE. Among them, APPEND is the default value, which means that this batch of data needs to be appended to the existing data, and DELETE means to delete all the data with the same key as this batch of data. Line, the MERGE semantics need to be used in conjunction with the delete condition, which means that the data that meets the delete condition is processed according to the DELETE semantics and the rest is processed according to the APPEND semantics, for example: `-H "merge_type: MERGE" -H "delete: flag=1"`

17. delete: Only meaningful under MERGE, indicating the deletion condition of the data

18. function_column.sequence_col: Only applicable to UNIQUE_KEYS. Under the same key column, ensure that the value column is REPLACEed according to the source_sequence column. The source_sequence can be a column in the data source or a column in the table structure.

19. fuzzy_parse: Boolean type, true means that json will be parsed with the schema of the first row. Enabling this option can improve the efficiency of json import, but requires that the order of the keys of all json objects is the same as the first row, the default is false, only use in json format

20. num_as_string: Boolean type, true means that when parsing json data, the numeric type will be converted to a string, and then imported without losing precision.
21. read_json_by_line: Boolean type, true to support reading one json object per line, the default value is false.
22. send_batch_parallelism: Integer, used to set the parallelism of sending batch data. If the value of parallelism exceeds `max_send_batch_parallelism_per_job` in the BE configuration, the BE as a coordination point will use the value of `max_send_batch_parallelism_per_job`.
23. hidden_columns: Specify hidden column when no `columns` in Headers, multi hidden column shoud be separated by commas.

    ```
    hidden_columns: __DORIS_DELETE_SIGN__,__DORIS_SEQUENCE_COL__
    The system will use the order specified by user. in case above, data should be ended
    with __DORIS_SEQUENCE_COL__.
    ```

24. load_to_single_tablet: Boolean type, True means that one task can only load data to one tablet in the corresponding partition at a time. The default value is false. This parameter can only be set when loading data into the OLAP table with random bucketing.
25. compress_type: Specify compress type file. Only support compressed csv file now. Support gz, lzo, bz2, lz4, lzop, deflate.
26. trim_double_quotes: Boolean type, The default value is false. True means that the outermost double quotes of each field in the csv file are trimmed.
27. skip_lines: Integer type, the default value is 0. It will skip some lines in the head of csv file. It will be disabled when format is `csv_with_names` or `csv_with_names_and_types`.
28. comment: String type, the default value is ““.

> **Tips**
>
> This feature is supported since the Apache Doris 1.2.3 version

29. enclose: When the csv data field contains row delimiters or column delimiters, to prevent accidental truncation, single-byte characters can be specified as brackets for protection. For example, the column separator is “,”, the bracket is “““, and the data is”a,’b,c’”, then “b,c” will be parsed as a field. Note: when the bracket is '', trim_double_quotes must be set to true.
30. escape Used to escape characters that appear in a csv field identical to the enclosing characters. For example, if the data is “a,‘b,’c’”, enclose is “““, and you want”b,’c to be parsed as a field, you need to specify a single-byte escape character, such as “'', and then modify the data to”a,’ b,'c’”.

Example

1. Import the data in the local file ‘testData’ into the table ‘testTbl’ in the database ‘testDb’, and use Label for deduplication. Specify a timeout of 100 seconds

```
    curl --location-trusted -u root -H "label:123" -H "timeout:100" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

2. Import the data in the local file 'testData' into the table 'testTbl' in the database 'testDb', use Label for deduplication, and only import data whose k1 is equal to 20180601

```
curl --location-trusted -u root -H "label:123" -H "where: k1=20180601" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

3. Import the data in the local file 'testData' into the table 'testTbl' in the database 'testDb', allowing a 20% error rate (the user is in the defalut_cluster)

```
curl --location-trusted -u root -H "label:123" -H "max_filter_ratio:0.2" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

4. Import the data in the local file 'testData' into the table 'testTbl' in the database 'testDb', allow a 20% error rate, and specify the column name of the file (the user is in the defalut_cluster)

```
curl --location-trusted -u root -H "label:123" -H "max_filter_ratio:0.2" -H "columns: k2, k1, v1" -T testData
http://host:port/api/testDb/testTbl /_stream_load
```

5. Import the data in the local file 'testData' into the p1, p2 partitions of the table 'testTbl' in the database 'testDb', allowing a 20% error rate.

```
curl --location-trusted -u root -H "label:123" -H "max_filter_ratio:0.2" -H "partitions: p1, p2" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

6. Import using streaming (user is in defalut_cluster)

```
seq 1 10 | awk '{OFS="\t"}{print $1, $1 * 10}' | curl --location-trusted -u root -T - http://host:port/api/testDb/testTbl/
_stream_load
```

7. Import a table containing HLL columns, which can be columns in the table or columns in the data to generate HLL columns, or use hll_empty to supplement columns that are not in the data

```
curl --location-trusted -u root -H "columns: k1, k2, v1=hll_hash(k1), v2=hll_empty()" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

8. Import data for strict mode filtering and set the time zone to Africa/Abidjan

```
curl --location-trusted -u root -H "strict_mode: true" -H "timezone: Africa/Abidjan" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

9. Import a table with a BITMAP column, which can be a column in the table or a column in the data to generate a BITMAP column, or use bitmap_empty to fill an empty Bitmap

```
 curl --location-trusted -u root -H "columns: k1, k2, v1=to_bitmap(k1), v2=bitmap_empty()" -T testData
```

```
http://host:port/api/testDb/testTbl/_stream_load
```

10. Simple mode, import json data Table Structure:

```
`category` varchar(512) NULL COMMENT "",
`author` varchar(512) NULL COMMENT "",
`title` varchar(512) NULL COMMENT "",
`price` double NULL COMMENT ""
```

json data format:

```
{"category":"C++","author":"avc","title":"C++ primer","price":895}
```

Import command:

```
curl --location-trusted -u root -H "label:123" -H "format: json" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

In order to improve throughput, it supports importing multiple pieces of json data at one time, each line is a json object, and is used as a newline by default. You need to set read_json_by_line to true. The json data format is as follows:

```
{"category":"C++","author":"avc","title":"C++ primer","price":89.5}
{"category":"Java","author":"avc","title":"Effective Java","price":95}
{"category":"Linux","author":"avc","title":"Linux kernel","price":195}
```

11. Match pattern, import json data json data format:

```
[
{"category":"xuxb111","author":"1avc","title":"SayingsoftheCentury","price":895},{"category":"xuxb222","author":"2avc","
title":"SayingsoftheCentury","price":895},
{"category":"xuxb333","author":"3avc","title":"SayingsoftheCentury","price":895}
]
```

Precise import by specifying jsonpath, such as importing only three attributes of category, author, and price

```
curl --location-trusted -u root -H "columns: category, price, author" -H "label:123" -H "format: json" -H "jsonpaths:
[\"$.category\",\" $.price\",\"$.author\"]" -H "strip_outer_array: true" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

illustrate: 1) If the json data starts with an array, and each object in the array is a record, you need to set strip_outer_array to true, which means flatten the array. 2) If the json data starts with an array, and each object in the array is a record, when setting jsonpath, our ROOT node is actually an object in the array.

12. User specified json root node json data format:

```
{
 "RECORDS":[
{"category":"11","title":"SayingsoftheCentury","price":895,"timestamp":1589191587},
{"category":"22","author":"2avc","price":895,"timestamp":1589191487},
```

```
{"category":"33","author":"3avc","title":"SayingsoftheCentury","timestamp":1589191387}
]
}
```

Precise import by specifying jsonpath, such as importing only three attributes of category, author, and price

```
curl --location-trusted -u root -H "columns: category, price, author" -H "label:123" -H "format: json" -H "jsonpaths:
[\"$.category\",\" $.price\",\"$.author\"]" -H "strip_outer_array: true" -H "json_root: $.RECORDS" -T testData
http://host:port/api/testDb/testTbl/_stream_load
```

13. Delete the data with the same import key as this batch

```
curl --location-trusted -u root -H "merge_type: DELETE" -T testData http://host:port/api/testDb/testTbl/_stream_load
```

14. Delete the columns in this batch of data that match the data whose flag is listed as true, and append other rows normally

```
curl --location-trusted -u root: -H "column_separator:," -H "columns: siteid, citycode, username, pv, flag" -H "merge_type:
MERGE" -H "delete: flag=1" -T testData http://host:port/api/testDb/testTbl/_stream_load
```

15. Import data into UNIQUE_KEYS table with sequence column

```
curl --location-trusted -u root -H "columns: k1,k2,source_sequence,v1,v2" -H "function_column.sequence_col:
source_sequence" -T testData http://host:port/api/testDb/testTbl/ _stream_load
```

16. csv file line header filter import

    file data:

```
   id,name,age
   1,doris,20
   2,flink,10
```

    Filter the first line import by specifying `format=csv_with_names`

```
curl --location-trusted -u root -T test.csv  -H "label:1" -H "format:csv_with_names" -H "column_separator:,"
http://host:port/api/testDb/testTbl/_stream_load
```

17. Import data into a table whose table field contains DEFAULT CURRENT_TIMESTAMP

    Table Structure:

```
`id` bigint(30) NOT NULL,
`order_code` varchar(30) DEFAULT NULL COMMENT '',
`create_time` datetimev2(3) DEFAULT CURRENT_TIMESTAMP
```

    JSON data format:

```
{"id":1,"order_Code":"avc"}
```

Import command:

```
curl --location-trusted -u root -T test.json -H "label:1" -H "format:json" -H 'columns: id, order_code,
create_time=CURRENT_TIMESTAMP()' http://host:port/api/testDb/testTbl/_stream_load
```

Keywords

```
STREAM, LOAD
```

Best Practice

1. Check the import task status

   Stream Load is a synchronous import process. The successful execution of the statement means that the data is imported successfully. The imported execution result will be returned synchronously through the HTTP return value. And display it in Json format. An example is as follows:

```
{
    "TxnId": 17,
    "Label": "707717c0-271a-44c5-be0b-4e71bfeacaa5",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 5,
    "NumberLoadedRows": 5,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 28,
    "LoadTimeMs": 27,
    "BeginTxnTimeMs": 0,
    "StreamLoadPutTimeMs": 2,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 3,
    "CommitAndPublishTimeMs": 18
}
```

   The following main explanations are given for the Stream load import result parameters:

   - TxnId: The imported transaction ID. Users do not perceive.
   - Label: Import Label. User specified or automatically generated by the system.
   - Status: Import completion status.

     “Success”: Indicates successful import.

     “Publish Timeout”: This state also indicates that the import has been completed, except that the data may be delayed and visible without retrying.

     “Label Already Exists”: Label duplicate, need to be replaced Label.

     “Fail”: Import failed.

- ExistingJobStatus: The state of the load job corresponding to the existing Label.

  This field is displayed only when the status is “Label Already Exists”. The user can know the status of the load job corresponding to Label through this state. “RUNNING” means that the job is still executing, and “FINISHED” means that the job is successful.

- Message: Import error messages.
- NumberTotalRows: Number of rows imported for total processing.
- NumberLoadedRows: Number of rows successfully imported.
- NumberFilteredRows: Number of rows that do not qualify for data quality.
- NumberUnselectedRows: Number of rows filtered by where condition.
- LoadBytes: Number of bytes imported.
- LoadTimeMs: Import completion time. Unit milliseconds.
- BeginTxnTimeMs: The time cost for RPC to Fe to begin a transaction, Unit milliseconds.
- StreamLoadPutTimeMs: The time cost for RPC to Fe to get a stream load plan, Unit milliseconds.
- ReadDataTimeMs: Read data time, Unit milliseconds.
- WriteDataTimeMs: Write data time, Unit milliseconds.
- CommitAndPublishTimeMs: The time cost for RPC to Fe to commit and publish a transaction, Unit milliseconds.
- ErrorURL: If you have data quality problems, visit this URL to see specific error lines.

> Note: Since Stream load is a synchronous import mode, import information will not be recorded in Doris system. Users cannot see Stream load asynchronously by looking at import commands. You need to listen for the return value of the create import request to get the import result.

2. How to correctly submit the Stream Load job and process the returned results.

   Stream Load is a synchronous import operation, so the user needs to wait for the return result of the command synchronously, and decide the next processing method according to the return result.

   The user’s primary concern is the `Status` field in the returned result.

   If it is `Success`, everything is fine and you can do other operations after that.

   If the returned result shows a large number of `Publish Timeout`, it may indicate that some resources (such as IO) of the cluster are currently under strain, and the imported data cannot take effect finally. The import task in the state of `Publish Timeout` has succeeded and does not need to be retried. However, it is recommended to slow down or stop the submission of new import tasks and observe the cluster load.

If the returned result is `Fail`, the import failed, and you need to check the problem according to the specific reason. Once resolved, you can retry with the same Label.

In some cases, the user's HTTP connection may be disconnected abnormally and the final returned result cannot be obtained. At this point, you can use the same Label to resubmit the import task, and the resubmitted task may have the following results:

1. `Status` status is `Success`, `Fail` or `Publish Timeout`. At this point, it can be processed according to the normal process.
2. The `Status` status is `Label Already Exists`. At this time, you need to continue to view the `ExistingJobStatus` field. If the value of this field is `FINISHED`, it means that the import task corresponding to this Label has been successful, and there is no need to retry. If it is `RUNNING`, it means that the import task corresponding to this Label is still running. At this time, you need to use the same Label to continue to submit repeatedly at intervals (such as 10 seconds) until `Status` is not `Label Already Exists'`, or until the value of the `ExistingJobStatus` field is `FINISHED`.

3. Cancel the import task

   Import tasks that have been submitted and not yet completed can be canceled with the CANCEL LOAD command. After cancellation, the written data will also be rolled back and will not take effect.

4. Label, import transaction, multi-table atomicity

   All import tasks in Doris are atomic. And the import of multiple tables in the same import task can also guarantee atomicity. At the same time, Doris can also use the Label mechanism to ensure that the data imported is not lost or heavy. For details, see the Import Transactions and Atomicity documentation.

5. Column mapping, derived columns and filtering

   Doris can support very rich column transformation and filtering operations in import statements. Most built-in functions and UDFs are supported. For how to use this function correctly, please refer to the Column Mapping, Conversion and Filtering document.

6. Error data filtering

   Doris' import tasks can tolerate a portion of malformed data. The tolerance ratio is set via `max_filter_ratio`. The default is 0, which means that the entire import task will fail when there is an error data. If the user wants to ignore some problematic data rows, the secondary parameter can be set to a value between 0 and 1, and Doris will automatically skip the rows with incorrect data format.

   For some calculation methods of the tolerance rate, please refer to the Column Mapping, Conversion and Filtering document.

7. Strict Mode

   The `strict_mode` attribute is used to set whether the import task runs in strict mode. The format affects the results of column mapping, transformation, and filtering, and it also controls the behavior of partial updates. For a detailed description of strict mode, see the strict mode documentation.

8. Timeout

The default timeout for Stream Load is 10 minutes. from the time the task is submitted. If it does not complete within the timeout period, the task fails.

9. Limits on data volume and number of tasks

   Stream Load is suitable for importing data within a few GB. Because the data is processed by single-threaded transmission, the performance of importing excessively large data cannot be guaranteed. When a large amount of local data needs to be imported, multiple import tasks can be submitted in parallel.

   Doris also limits the number of import tasks running at the same time in the cluster, usually ranging from 10-20. Import jobs submitted after that will be rejected.

10. Importing with timezones

Since Doris currently has no built-in time types for time zones, all `DATETIME` related types only represent absolute points in time, and do not contain time zone information, which does not change due to time zone changes in the Doris system. Therefore, for importing data with a time zone, we uniformly handle it as **converting it to data in a specific target time zone**. In the Doris system, this is the time zone represented by the session variable `time_zone`.

In the import, on the other hand, our target timezone is specified by the parameter `timezone`, which will replace the session variable `time_zone` when timezone conversions occur, and when computing timezone-sensitive functions. Therefore, if there are no special circumstances, `timezone` should be set in the import transaction to match the `time_zone` of the current Doris cluster. This means that all time data with a time zone will be converted to that time zone. For example, if the Doris system timezone is “+08:00”, and the time column in the imported data contains two pieces of data, “2012-01-01 01:00:00Z” and “2015-12-12 12:12:12-08:00”, then after we specify the timezone of the imported transaction via `-H "timezone: +08:00"` during import, both pieces of data will be converted to that timezone, resulting in the results “2012-01-01 09:00:00” and “2015-12-13 04:12:12”.

For a more detailed understanding, see time-zone document.

11. The Execution Engine Choice

The Session Variable `enable_pipeline_load` determines whether to attempt to enable the Pipeline engine for Streamload tasks. See Import document for more.

MULTI-LOAD

MULTI-LOAD

Name

MULTI LOAD

Description

Users submit multiple import jobs through the HTTP protocol. Multi Load can ensure the atomic effect of multiple import jobs

```
Syntax:
    curl --location-trusted -u user:passwd -XPOST http://host:port/api/{db}/_multi_start?label=xxx
    curl --location-trusted -u user:passwd -T data.file http://host:port/api/{db}/{table1}/_load?label=xxx\&sub_label=yyy
    curl --location-trusted -u user:passwd -T data.file http://host:port/api/{db}/{table2}/_load?label=xxx\&sub_label=zzz
    curl --location-trusted -u user:passwd -XPOST http://host:port/api/{db}/_multi_commit?label=xxx
    curl --location-trusted -u user:passwd -XPOST http://host:port/api/{db}/_multi_desc?label=xxx

On the basis of 'MINI LOAD', 'MULTI LOAD' can support users to import to multiple tables at the same time. The specific commands are
shown above.
'/api/{db}/_multi_start' starts a multi-table import task
'/api/{db}/{table}/_load' adds a table to be imported to an import task. The main difference from 'MINI LOAD' is that the
'sub_label' parameter needs to be passed in
'/api/{db}/_multi_commit' submits the entire multi-table import task, and starts processing in the background
'/api/{db}/_multi_abort' Abort a multi-table import task
'/api/{db}/_multi_desc' can display the number of jobs submitted by a multi-table import task

Description of the HTTP protocol
    Authorization Authentication Currently, Doris uses HTTP Basic authorization authentication. So you need to specify the username
and password when importing
                        This method is to pass the password in clear text, since we are currently in an intranet environment. . .

    Expect Doris needs to send the http request, it needs to have 'Expect' header information, the content is '100-continue'
                        why? Because we need to redirect the request, before transmitting the data content,
                        This can avoid causing multiple transmissions of data, thereby improving efficiency.

    Content-Length Doris needs to send the request with the 'Content-Length' header. If the content sent is greater than
                        If the 'Content-Length' is less, then Palo thinks that there is a problem with the transmission, and fails
to submit the task.
                        NOTE: If more data is sent than 'Content-Length', then Doris only reads 'Content-Length'
                        length content and import

Parameter Description:
    user: If the user is in the default_cluster, the user is the user_name. Otherwise user_name@cluster_name.

    label: Used to specify the label number imported in this batch, which is used for later job status query, etc.
                        This parameter is required.

    sub_label: Used to specify the subversion number inside a multi-table import task. For loads imported from multiple tables, this
parameter must be passed in.

    columns: used to describe the corresponding column names in the import file.
                        If it is not passed in, then the order of the columns in the file is considered to be the same as the order
in which the table was created.
```

```
                      The specified method is comma-separated, for example: columns=k1,k2,k3,k4

    column_separator: used to specify the separator between columns, the default is '\t'
                      NOTE: url encoding is required, for example, '\t' needs to be specified as the delimiter,
                      Then you should pass in 'column_separator=%09'

    max_filter_ratio: used to specify the maximum ratio of non-standard data allowed to filter, the default is 0, no filtering is
allowed
                      The custom specification should be as follows: 'max_filter_ratio=0.2', which means 20% error rate is allowed
                      Passing in has effect when '_multi_start'

NOTE:
    1. This import method currently completes the import work on one machine, so it is not suitable for import work with a large
amount of data.
    It is recommended that the amount of imported data should not exceed 1GB

    2. Currently it is not possible to submit multiple files using `curl -T "{file1, file2}"`, because curl splits them into
multiple files
    The request is sent. Multiple requests cannot share a label number, so it cannot be used.

    3. Supports the use of curl to import data into Doris in a way similar to streaming, but only after the streaming ends Doris
    The real import behavior will occur, and the amount of data in this way cannot be too large.
```

Example

```
1. Import the data in the local file 'testData1' into the table 'testTbl1' in the database 'testDb', and
Import the data of 'testData2' into table 'testTbl2' in 'testDb' (user is in defalut_cluster)
    curl --location-trusted -u root -XPOST http://host:port/api/testDb/_multi_start?label=123
    curl --location-trusted -u root -T testData1 http://host:port/api/testDb/testTbl1/_load?label=123\&sub_label=1
    curl --location-trusted -u root -T testData2 http://host:port/api/testDb/testTbl2/_load?label=123\&sub_label=2
    curl --location-trusted -u root -XPOST http://host:port/api/testDb/_multi_commit?label=123

2. Abandoned in the middle of multi-table import (user is in defalut_cluster)
    curl --location-trusted -u root -XPOST http://host:port/api/testDb/_multi_start?label=123
    curl --location-trusted -u root -T testData1 http://host:port/api/testDb/testTbl1/_load?label=123\&sub_label=1
    curl --location-trusted -u root -XPOST http://host:port/api/testDb/_multi_abort?label=123

3. Multi-table import to see how much content has been submitted (the user is in the defalut_cluster)
    curl --location-trusted -u root -XPOST http://host:port/api/testDb/_multi_start?label=123
    curl --location-trusted -u root -T testData1 http://host:port/api/testDb/testTbl1/_load?label=123\&sub_label=1
    curl --location-trusted -u root -XPOST http://host:port/api/testDb/_multi_desc?label=123
```

Keywords

```
MULTI, MINI, LOAD
```

Best Practice

BROKER-LOAD

BROKER-LOAD

Name

BROKER LOAD

Description

该命令主要用于通过 Broker 服务进程读取远端存储（如 S3、HDFS）上的数据导入到 Doris 表里。

```
LOAD LABEL load_label
(
data_desc1[, data_desc2, ...]
)
WITH BROKER broker_name
[broker_properties]
[load_properties]
[COMMENT "comments"];
```

- load_label

  每个导入需要指定一个唯一的 Label。后续可以通过这个 label 来查看作业进度。

  [database.]label_name

- data_desc1

  用于描述一组需要导入的文件。

  ```
  [MERGE|APPEND|DELETE]
  DATA INFILE
  (
  "file_path1"[, file_path2, ...]
  )
  [NEGATIVE]
  INTO TABLE `table_name`
  [PARTITION (p1, p2, ...)]
  [COLUMNS TERMINATED BY "column_separator"]
  [LINES TERMINATED BY "line_delimiter"]
  [FORMAT AS "file_type"]
  [COMPRESS_TYPE AS "compress_type"]
  [(column_list)]
  [COLUMNS FROM PATH AS (c1, c2, ...)]
  [SET (column_mapping)]
  [PRECEDING FILTER predicate]
  [WHERE predicate]
  [DELETE ON expr]
  [ORDER BY source_sequence]
  [PROPERTIES ("key1"="value1", ...)]
  ```

  - [MERGE|APPEND|DELETE]

数据合并类型。默认为 APPEND，表示本次导入是普通的追加写操作。MERGE 和 DELETE 类型仅适用于 Unique Key 模型表。其中 MERGE 类型需要配合 `[DELETE ON]` 语句使用，以标注 Delete Flag 列。而 DELETE 类型则表示本次导入的所有数据皆为删除数据。

- `DATA INFILE`

  指定需要导入的文件路径。可以是多个。可以使用通配符。路径最终必须匹配到文件，如果只匹配到目录则导入会失败。

- `NEGATIVE`

  该关键词用于表示本次导入为一批"负"导入。这种方式仅针对具有整型 SUM 聚合类型的聚合数据表。该方式会将导入数据中，SUM 聚合列对应的整型数值取反。主要用于冲抵之前导入错误的数据。

- `PARTITION(p1, p2, ...)`

  可以指定仅导入表的某些分区。不在分区范围内的数据将被忽略。

- `COLUMNS TERMINATED BY`

  指定列分隔符。仅在 CSV 格式下有效。仅能指定单字节分隔符。

- `LINES TERMINATED BY`

  指定行分隔符。仅在 CSV 格式下有效。仅能指定单字节分隔符。

- `FORMAT AS`

  指定文件类型，支持 CSV、PARQUET 和 ORC 格式。默认为 CSV。

- `COMPRESS_TYPE AS` 指定文件压缩类型，支持 GZ/BZ2/LZ4FRAME。

- `column list`

  用于指定原始文件中的列顺序。关于这部分详细介绍，可以参阅 列的映射，转换与过滤 文档。

  `(k1, k2, tmpk1)`

- `COLUMNS FROM PATH AS`

  指定从导入文件路径中抽取的列。

- `SET (column_mapping)`

  指定列的转换函数。

- `PRECEDING FILTER predicate`

  前置过滤条件。数据首先根据 `column list` 和 `COLUMNS FROM PATH AS` 按顺序拼接成原始数据行。然后按照前置过滤条件进行过滤。关于这部分详细介绍，可以参阅 列的映射，转换与过滤 文档。

- `WHERE predicate`

  根据条件对导入的数据进行过滤。关于这部分详细介绍，可以参阅 列的映射，转换与过滤 文档。

- DELETE ON expr

  需配合 MEREGE 导入模式一起使用，仅针对 Unique Key 模型的表。用于指定导入数据中表示 Delete Flag 的列和计算关系。

- ORDER BY

  仅针对 Unique Key 模型的表。用于指定导入数据中表示 Sequence Col 的列。主要用于导入时保证数据顺序。

- PROPERTIES ("key1"="value1", ...)

  指定导入的 format 的一些参数。如导入的文件是json格式，则可以在这里指定json_root、jsonpaths、fuzzy_parse等参数。

  - enclose

  包围符。当 csv 数据字段中含有行分隔符或列分隔符时，为防止意外截断，可指定单字节字符作为包围符起到保护作用。例如列分隔符为",“，包围符为”“，数据为”a,'b,c'“,则”b,c”会被解析为一个字段。 注意：当 enclose 设置为"时，trim_double_quotes 一定要设置为 true。

  - escape

  转义符。用于转义在字段中出现的与包围符相同的字符。例如数据为”a,‘b,’c'“，包围符为”“，希望”b,’c 被作为一个字段解析，则需要指定单字节转义符，例如"，然后将数据修改为”a,'b,'c'“。

- WITH BROKER broker_name

  指定需要使用的 Broker 服务名称。在公有云 Doris 中。Broker 服务名称为 bos

- broker_properties

  指定 broker 所需的信息。这些信息通常被用于 Broker 能够访问远端存储系统。如 BOS 或 HDFS。关于具体信息，可参阅 Broker 文档。

  ```
  (
      "key1" = "val1",
      "key2" = "val2",
      ...
  )
  ```

  - load_properties

    指定导入的相关参数。目前支持以下参数：

    - timeout

      导入超时时间。默认为 4 小时。单位秒。

    - max_filter_ratio

      最大容忍可过滤（数据不规范等原因）的数据比例。默认零容忍。取值范围为 0 到 1。

    - exec_mem_limit

导入内存限制。默认为 2GB。单位为字节。

- `strict_mode`

  是否对数据进行严格限制。默认为 false。

- `partial_columns`

  布尔类型，为 true 表示使用部分列更新，默认值为 false，该参数只允许在表模型为 Unique 且采用 Merge on Write 时设置。

- `timezone`

  指定某些受时区影响的函数的时区，如 `strftime/alignment_timestamp/from_unixtime` 等等，具体请查阅 时区 文档。如果不指定，则使用 “Asia/Shanghai” 时区

- `load_parallelism`

  导入并发度，默认为 1。调大导入并发度会启动多个执行计划同时执行导入任务，加快导入速度。

- `send_batch_parallelism`

  用于设置发送批处理数据的并行度，如果并行度的值超过 BE 配置中的 `max_send_batch_parallelism_per_job`，那么作为协调点的 BE 将使用 `max_send_batch_parallelism_per_job` 的值。

- `load_to_single_tablet`

  布尔类型，为 true 表示支持一个任务只导入数据到对应分区的一个 tablet，默认值为 false，作业的任务数取决于整体并发度。该参数只允许在对带有 random 分桶的 olap 表导数的时候设置。

- priority

设置导入任务的优先级，可选 `HIGH/NORMAL/LOW` 三种优先级，默认为 `NORMAL`，对于处在 `PENDING` 状态的导入任务，更高优先级的任务将优先被执行进入 `LOADING` 状态。

- comment

  指定导入任务的备注信息。可选参数。

:::tip 提示 该功能自 Apache Doris 1.2 版本起支持 :::

Example

1. 从 HDFS 导入一批数据

```
LOAD LABEL example_db.label1
(
    DATA INFILE("hdfs://hdfs_host:hdfs_port/input/file.txt")
    INTO TABLE `my_table`
    COLUMNS TERMINATED BY ","
)
WITH BROKER hdfs
```

```
(
    "username"="hdfs_user",
    "password"="hdfs_password"
);
```

导入文件 file.txt，按逗号分隔，导入到表 my_table。

2. 从 HDFS 导入数据，使用通配符匹配两批文件。分别导入到两个表中。

```
LOAD LABEL example_db.label2
(
    DATA INFILE("hdfs://hdfs_host:hdfs_port/input/file-10*")
    INTO TABLE `my_table1`
    PARTITION (p1)
    COLUMNS TERMINATED BY ","
    (k1, tmp_k2, tmp_k3)
    SET (
        k2 = tmp_k2 + 1,
        k3 = tmp_k3 + 1
    )
    DATA INFILE("hdfs://hdfs_host:hdfs_port/input/file-20*")
    INTO TABLE `my_table2`
    COLUMNS TERMINATED BY ","
    (k1, k2, k3)
)
WITH BROKER hdfs
(
    "username"="hdfs_user",
    "password"="hdfs_password"
);
```

使用通配符匹配导入两批文件 file-10* 和 file-20*。分别导入到 my_table1 和 my_table2 两张表中。其中 my_table1 指定导入到分区 p1 中，并且将导入源文件中第二列和第三列的值 +1 后导入。

3. 从 HDFS 导入一批数据。

```
LOAD LABEL example_db.label3
(
    DATA INFILE("hdfs://hdfs_host:hdfs_port/user/doris/data/*/*")
    INTO TABLE `my_table`
    COLUMNS TERMINATED BY "\\x01"
)
WITH BROKER my_hdfs_broker
(
    "username" = "",
    "password" = "",
    "fs.defaultFS" = "hdfs://my_ha",
    "dfs.nameservices" = "my_ha",
    "dfs.ha.namenodes.my_ha" = "my_namenode1, my_namenode2",
    "dfs.namenode.rpc-address.my_ha.my_namenode1" = "nn1_host:rpc_port",
    "dfs.namenode.rpc-address.my_ha.my_namenode2" = "nn2_host:rpc_port",
    "dfs.client.failover.proxy.provider.my_ha" =
"org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"
);
```

指定分隔符为 Hive 的默认分隔符 \\x01，并使用通配符 * 指定 data 目录下所有目录的所有文件。使用简单认证，同时配置 namenode HA。

4. 导入 Parquet 格式数据，指定 FORMAT 为 parquet。默认是通过文件后缀判断

```
LOAD LABEL example_db.label4
(
    DATA INFILE("hdfs://hdfs_host:hdfs_port/input/file")
    INTO TABLE `my_table`
    FORMAT AS "parquet"
    (k1, k2, k3)
)
WITH BROKER hdfs
(
    "username"="hdfs_user",
    "password"="hdfs_password"
);
```

5. 导入数据，并提取文件路径中的分区字段

```
LOAD LABEL example_db.label10
(
    DATA INFILE("hdfs://hdfs_host:hdfs_port/input/city=beijing/*/*")
    INTO TABLE `my_table`
    FORMAT AS "csv"
    (k1, k2, k3)
    COLUMNS FROM PATH AS (city, utc_date)
)
WITH BROKER hdfs
(
    "username"="hdfs_user",
    "password"="hdfs_password"
);
```

my_table 表中的列为 k1，k2，k3，city，utc_date。

其中 hdfs://hdfs_host:hdfs_port/user/doris/data/input/dir/city=beijing 目录下包括如下文件：

```
hdfs://hdfs_host:hdfs_port/input/city=beijing/utc_date=2020-10-01/0000.csv
hdfs://hdfs_host:hdfs_port/input/city=beijing/utc_date=2020-10-02/0000.csv
hdfs://hdfs_host:hdfs_port/input/city=tianji/utc_date=2020-10-03/0000.csv
hdfs://hdfs_host:hdfs_port/input/city=tianji/utc_date=2020-10-04/0000.csv
```

文件中只包含 k1，k2，k3 三列数据，city，utc_date 这两列数据会从文件路径中提取。

6. 对待导入数据进行过滤。

```
LOAD LABEL example_db.label6
(
    DATA INFILE("hdfs://host:port/input/file")
    INTO TABLE `my_table`
    (k1, k2, k3)
    SET (
        k2 = k2 + 1
    )
    PRECEDING FILTER k1 = 1
    WHERE k1 > k2
)
WITH BROKER hdfs
(
    "username"="user",
    "password"="pass"
);
```

只有原始数据中，k1 = 1，并且转换后，k1 > k2 的行才会被导入。

7. 导入数据，提取文件路径中的时间分区字段，并且时间包含 %3A (在 hdfs 路径中，不允许有 ':'，所有 ':' 会由 %3A 替换)

```
LOAD LABEL example_db.label7
(
    DATA INFILE("hdfs://host:port/user/data/*/test.txt")
    INTO TABLE `tbl12`
    COLUMNS TERMINATED BY ","
    (k2,k3)
    COLUMNS FROM PATH AS (data_time)
    SET (
        data_time=str_to_date(data_time, '%Y-%m-%d %H%%3A%i%%3A%s')
    )
)
WITH BROKER hdfs
(
    "username"="user",
    "password"="pass"
);
```

路径下有如下文件：

```
/user/data/data_time=2020-02-17 00%3A00%3A00/test.txt
/user/data/data_time=2020-02-18 00%3A00%3A00/test.txt
```

表结构为：

```
data_time DATETIME,
k2        INT,
k3        INT
```

8. 从 HDFS 导入一批数据，指定超时时间和过滤比例。使用明文 my_hdfs_broker 的 broker。简单认证。并且将原有数据中与 导入数据中 v2 大于 100 的列相匹配的列删除，其他列正常导入

```
LOAD LABEL example_db.label8
(
    MERGE DATA INFILE("HDFS://test:802/input/file")
    INTO TABLE `my_table`
    (k1, k2, k3, v2, v1)
    DELETE ON v2 > 100
)
WITH HDFS
(
    "hadoop.username"="user",
    "password"="pass"
)
PROPERTIES
(
    "timeout" = "3600",
    "max_filter_ratio" = "0.1"
);
```

使用 MERGE 方式导入。`my_table` 必须是一张 Unique Key 的表。当导入数据中的 v2 列的值大于 100 时，该行会被认为是一个删除行。

导入任务的超时时间是 3600 秒，并且允许错误率在 10% 以内。

9. 导入时指定 source_sequence 列，保证 UNIQUE_KEYS 表中的替换顺序：

```
LOAD LABEL example_db.label9
(
    DATA INFILE("HDFS://test:802/input/file")
    INTO TABLE `my_table`
    COLUMNS TERMINATED BY ","
    (k1,k2,source_sequence,v1,v2)
    ORDER BY source_sequence
)
WITH HDFS
(
    "hadoop.username"="user",
    "password"="pass"
)
```

`my_table` 必须是 Unique Key 模型表，并且指定了 Sequcence Col。数据会按照源数据中 `source_sequence` 列的值来保证顺序性。

10. 从 HDFS 导入一批数据，指定文件格式为 `json` 并指定 `json_root、jsonpaths`

```
LOAD LABEL example_db.label10
(
    DATA INFILE("HDFS://test:port/input/file.json")
    INTO TABLE `my_table`
    FORMAT AS "json"
    PROPERTIES(
      "json_root" = "$.item",
      "jsonpaths" = "[$.id, $.city, $.code]"
    )
)
with HDFS (
"hadoop.username" = "user"
"password" = ""
)
PROPERTIES
(
"timeout"="1200",
"max_filter_ratio"="0.1"
);
```

`jsonpaths` 可与 `column list` 及 `SET (column_mapping)`配合：

``sql LOAD LABEL example_db.label10 (DATA INFILE("HDFS://test:port/input/file.json") INTO TABLEmy_table` FORMAT AS “json” (id, code, city) SET (id = id * 10) PROPERTIES(“json_root” = “$.item", "jsonpaths" = "[$.id, $.code, $.city]”)
) with HDFS (“hadoop.username” = “user” “password” = “”) PROPERTIES (“timeout”=“1200”, “max_filter_ratio”=“0.1”);

11. 从腾讯云 cos 中以 csv 格式导入数据。

```
LOAD LABEL example_db.label10
(
DATA INFILE("cosn://my_bucket/input/file.csv")
INTO TABLE `my_table`
(k1, k2, k3)
```

```
)
WITH BROKER "broker_name"
(
    "fs.cosn.userinfo.secretId" = "xxx",
    "fs.cosn.userinfo.secretKey" = "xxxx",
    "fs.cosn.bucket.endpoint_suffix" = "cos.xxxxxxxxx.myqcloud.com"
)
```

12. 导入 CSV 数据时去掉双引号，并跳过前 5 行。

```
LOAD LABEL example_db.label12
(
DATA INFILE("cosn://my_bucket/input/file.csv")
INTO TABLE `my_table`
(k1, k2, k3)
PROPERTIES("trim_double_quotes" = "true", "skip_lines" = "5")
)
WITH BROKER "broker_name"
(
    "fs.cosn.userinfo.secretId" = "xxx",
    "fs.cosn.userinfo.secretKey" = "xxxx",
    "fs.cosn.bucket.endpoint_suffix" = "cos.xxxxxxxxx.myqcloud.com"
)
```

Keywords

```
BROKER, LOAD
```

Best Practice

1. 查看导入任务状态

   Broker Load 是一个异步导入过程，语句执行成功仅代表导入任务提交成功，并不代表数据导入成功。导入状态需要通过 SHOW LOAD 命令查看。

2. 取消导入任务

   已提交切尚未结束的导入任务可以通过 CANCEL LOAD 命令取消。取消后，已写入的数据也会回滚，不会生效。

3. Label、导入事务、多表原子性

   Doris 中所有导入任务都是原子生效的。并且在同一个导入任务中对多张表的导入也能够保证原子性。同时，Doris 还可以通过 Label 的机制来保证数据导入的不丢不重。具体说明可以参阅 导入事务和原子性 文档。

4. 列映射、衍生列和过滤

   Doris 可以在导入语句中支持非常丰富的列转换和过滤操作。支持绝大多数内置函数和 UDF。关于如何正确的使用这个功能，可参阅 列的映射，转换与过滤 文档。

5. 错误数据过滤

Doris 的导入任务可以容忍一部分格式错误的数据。容忍了通过 `max_filter_ratio` 设置。默认为 0，即表示当有一条错误数据时，整个导入任务将会失败。如果用户希望忽略部分有问题的数据行，可以将次参数设置为 0~1 之间的数值，Doris 会自动跳过哪些数据格式不正确的行。

关于容忍率的一些计算方式，可以参阅 列的映射，转换与过滤 文档。

6. 严格模式

   `strict_mode` 属性用于设置导入任务是否运行在严格模式下。该格式会对列映射、转换和过滤的结果产生影响。关于严格模式的具体说明，可参阅 严格模式 文档。

7. 超时时间

   Broker Load 的默认超时时间为 4 小时。从任务提交开始算起。如果在超时时间内没有完成，则任务会失败。

8. 数据量和任务数限制

   Broker Load 适合在一个导入任务中导入 100GB 以内的数据。虽然理论上在一个导入任务中导入的数据量没有上限。但是提交过大的导入会导致运行时间较长，并且失败后重试的代价也会增加。

   同时受限于集群规模，我们限制了导入的最大数据量为 ComputeNode 节点数 * 3GB。以保证系统资源的合理利用。如果有大数据量需要导入，建议分成多个导入任务提交。

   Doris 同时会限制集群内同时运行的导入任务数量，通常在 3-10 个不等。之后提交的导入作业会排队等待。队列最大长度为 100。之后的提交会直接拒绝。注意排队时间也被计算到了作业总时间中。如果超时，则作业会被取消。所以建议通过监控作业运行状态来合理控制作业提交频率。 Doris also limits the number of import tasks running simultaneously in the cluster, usually ranging from 3 to 10. Import jobs submitted after that are queued. The maximum queue length is 100. Subsequent submissions will be rejected outright. Note that the queue time is also calculated into the total job time. If it times out, the job is canceled. Therefore, it is recommended to reasonably control the frequency of job submission by monitoring the running status of the job.

MYSQL-LOAD

MYSQL-LOAD

Name

```
MYSQL LOAD
```

Description

mysql-load: Import local data using the MySql client

```
LOAD DATA
[LOCAL]
INFILE 'file_name'
INTO TABLE tbl_name
[PARTITION (partition_name [, partition_name] ...)]
[COLUMNS TERMINATED BY 'string']
[LINES TERMINATED BY 'string']
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var [, col_name_or_user_var] ...)]
[SET (col_name={expr | DEFAULT} [, col_name={expr | DEFAULT}] ...)]
[PROPERTIES (key1 = value1 [, key2=value2]) ]
```

This statement is used to import data to the specified table. Unlike normal Load, this import method is a synchronous import.

This import method can still guarantee the atomicity of a batch of import tasks, either all data imports are successful or all fail.

1. MySQL Load starts with the syntax `LOAD DATA`, without specifying `LABEL`
2. Specify `LOCAL` to read client side files. Not specified to read FE server side local files. Server side load was disabled by default. It can be enabled by setting a secure path in FE configuration `mysql_load_server_secure_path`
3. The local fill path will be filled after `INFILE`, which can be a relative path or an absolute path. Currently only a single file is supported, and multiple files are not supported
4. The table name after `INTO TABLE` can specify the database name, as shown in the case. It can also be omitted, and the database where the current user is located will be used.
5. `PARTITION` syntax supports specified partition to import
6. `COLUMNS TERMINATED BY` specifies the column separator
7. `LINES TERMINATED BY` specifies the line separator
8. `IGNORE num LINES` The user skips the header of the CSV and can skip any number of lines. This syntax can also be replaced by'IGNORE num ROWS '
9. Column mapping syntax, please refer to the column mapping chapter of Imported Data Transformation
10. `PROPERTIES` parameter configuration, see below for details

PROPERTIES

1. max_filter_ratio: The maximum tolerable data ratio that can be filtered (for reasons such as data irregularity). Zero tolerance by default. Data irregularities do not include rows filtered out by where conditions.
2. timeout: Specify the import timeout. in seconds. The default is 600 seconds. The setting range is from 1 second to 259200 seconds.
3. strict_mode: The user specifies whether to enable strict mode for this import. The default is off.
4. timezone: Specify the time zone used for this import. The default is Dongba District. This parameter affects the results of all time zone-related functions involved in the import.
5. exec_mem_limit: Load memory limit. Default is 2GB. The unit is bytes.
6. trim_double_quotes: Boolean type, The default value is false. True means that the outermost double quotes of each field in the load file are trimmed.
7. enclose: When the csv data field contains row delimiters or column delimiters, to prevent accidental truncation, single-byte characters can be specified as brackets for protection. For example, the column separator is “,”, the bracket is ““, and the data is”a,’b,c’”, then “b,c” will be parsed as a field. Note: when the bracket is ", trim_double_quotes must be set to true.
8. escape: Used to escape characters that appear in a csv field identical to the enclosing characters. For example, if the data is “a,‘b,’c’”, enclose is ““, and you want”b,’c to be parsed as a field, you need to specify a single-byte escape character, such as “", and then modify the data to”a,’ b,'c’”.

Example

1. Import the data from the client side local file `testData` into the table `testTbl` in the database `testDb`. Specify a timeout of 100 seconds

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PROPERTIES ("timeout"="100")
```

2. Import the data from the server side local file `/root/testData` (set FE config `mysql_load_server_secure_path` to be `root` already) into the table `testTbl` in the database `testDb`. Specify a timeout of 100 seconds

```
LOAD DATA
INFILE '/root/testData'
INTO TABLE testDb.testTbl
PROPERTIES ("timeout"="100")
```

3. Import data from client side local file `testData` into table `testTbl` in database `testDb`, allowing 20% error rate

```
LOAD DATA LOCAL
INFILE 'testData'
```

```
INTO TABLE testDb.testTbl
PROPERTIES ("max_filter_ratio"="0.2")
```

4. Import the data from the client side local file `testData` into the table `testTbl` in the database `testDb`, allowing a 20% error rate and specifying the column names of the file

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
(k2, k1, v1)
PROPERTIES ("max_filter_ratio"="0.2")
```

5. Import the data in the local file `testData` into the p1, p2 partitions in the table of `testTbl` in the database `testDb`, allowing a 20% error rate.

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PARTITION (p1, p2)
PROPERTIES ("max_filter_ratio"="0.2")
```

6. Import the data in the CSV file `testData` with a local row delimiter of `0102` and a column delimiter of `0304` into the table `testTbl` in the database `testDb`.

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
COLUMNS TERMINATED BY '0304'
LINES TERMINATED BY '0102'
```

7. Import the data from the local file `testData` into the p1, p2 partitions in the table of `testTbl` in the database `testDb` and skip the first 3 lines.

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PARTITION (p1, p2)
IGNORE 1 LINES
```

8. Import data for strict schema filtering and set the time zone to Africa/Abidjan

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PROPERTIES ("strict_mode"="true", "timezone"="Africa/Abidjan")
```

9. Import data is limited to 10GB of load memory and timed out in 10 minutes

```
LOAD DATA LOCAL
INFILE 'testData'
INTO TABLE testDb.testTbl
PROPERTIES ("exec_mem_limit"="10737418240", "timeout"="600")
```

Keywords

```
MYSQL, LOAD
```

CREATE-ROUTINE-LOAD

CREATE-ROUTINE-LOAD

Name

CREATE ROUTINE LOAD

Description

The Routine Load function allows users to submit a resident import task, and import data into Doris by continuously reading data from a specified data source.

Currently, only data in CSV or Json format can be imported from Kakfa through unauthenticated or SSL authentication. Example of importing data in Json format

grammar:

```
CREATE ROUTINE LOAD [db.]job_name [ON tbl_name]
[merge_type]
[load_properties]
[job_properties]
FROM data_source [data_source_properties]
[COMMENT "comment"]
```

- `[db.]job_name`

  The name of the import job. Within the same database, only one job with the same name can be running.

- `tbl_name`

  Specifies the name of the table to be imported.Optional parameter, If not specified, the dynamic table method will be used, which requires the data in Kafka to contain table name information. Currently, only the table name can be obtained from the Kafka value, and it needs to conform to the format of “table_name|{”col1”: “val1”, “col2”: “val2”}” for JSON data. The “tbl_name” represents the table name, and “|” is used as the delimiter between the table name and the table data. The same format applies to CSV data, such as “table_name|val1,val2,val3”. It is important to note that the “table_name” must be consistent with the table name in Doris, otherwise it may cause import failures.

  Tips: The `columns_mapping` parameter is not supported for dynamic tables. If your table structure is consistent with the table structure in Doris and there is a large amount of table information to be imported, this method will be the best choice.

- `merge_type`

  Data merge type. The default is APPEND, which means that the imported data are ordinary append write operations. The MERGE and DELETE types are only available for Unique Key model tables. The

MERGE type needs to be used with the [DELETE ON] statement to mark the Delete Flag column. The DELETE type means that all imported data are deleted data.

Tips: When using dynamic multiple tables, please note that this parameter should be consistent with the type of each dynamic table, otherwise it will result in import failure.

- load_properties

  Used to describe imported data. The composition is as follows:

  ```
  [column_separator],
  [columns_mapping],
  [preceding_filter],
  [where_predicates],
  [partitions],
  [DELETE ON],
  [ORDER BY]
  ```

  - `column_separator`

    Specifies the column separator, defaults to `\t`

    `COLUMNS TERMINATED BY ","`

  - `columns_mapping`

    It is used to specify the mapping relationship between file columns and columns in the table, as well as various column transformations. For a detailed introduction to this part, you can refer to the [Column Mapping, Transformation and Filtering] document.

    `(k1, k2, tmpk1, k3 = tmpk1 + 1)`

    Tips: Dynamic multiple tables are not supported.

  - `preceding_filter`

    Filter raw data. For a detailed introduction to this part, you can refer to the [Column Mapping, Transformation and Filtering] document.

    Tips: Dynamic multiple tables are not supported.

  - `where_predicates`

    Filter imported data based on conditions. For a detailed introduction to this part, you can refer to the [Column Mapping, Transformation and Filtering] document.

    `WHERE k1 > 100 and k2 = 1000`

    Tips: When using dynamic multiple tables, please note that this parameter should be consistent with the type of each dynamic table, otherwise it will result in import failure.

  - `partitions`

    Specify in which partitions of the import destination table. If not specified, it will be automatically imported into the corresponding partition.

`PARTITION(p1, p2, p3)`

Tips: When using dynamic multiple tables, please note that this parameter should conform to each dynamic table, otherwise it may cause import failure.

- `DELETE ON`

  It needs to be used with the MEREGE import mode, only for the table of the Unique Key model. Used to specify the columns and calculated relationships in the imported data that represent the Delete Flag.

  `DELETE ON v3 >100`

  Tips: When using dynamic multiple tables, please note that this parameter should conform to each dynamic table, otherwise it may cause import failure.

- `ORDER BY`

  Tables only for the Unique Key model. Used to specify the column in the imported data that represents the Sequence Col. Mainly used to ensure data order when importing.

  Tips: When using dynamic multiple tables, please note that this parameter should conform to each dynamic table, otherwise it may cause import failure.

- `job_properties`

  Common parameters for specifying routine import jobs.

```
PROPERTIES (
    "key1" = "val1",
    "key2" = "val2"
)
```

  Currently we support the following parameters:

  1. `desired_concurrent_number`

     Desired concurrency. A routine import job will be divided into multiple subtasks for execution. This parameter specifies the maximum number of tasks a job can execute concurrently. Must be greater than 0. Default is 5.

     This degree of concurrency is not the actual degree of concurrency. The actual degree of concurrency will be comprehensively considered by the number of nodes in the cluster, the load situation, and the situation of the data source.

     `"desired_concurrent_number" = "3"`

  2. `max_batch_interval/max_batch_rows/max_batch_size`

     These three parameters represent:

     1. The maximum execution time of each subtask, in seconds. The range is 1 to 60. Default is 10.

2. The maximum number of lines read by each subtask. Must be greater than or equal to 200000. The default is 200000.
3. The maximum number of bytes read by each subtask. The unit is bytes and the range is 100MB to 1GB. The default is 100MB.

These three parameters are used to control the execution time and processing volume of a subtask. When either one reaches the threshold, the task ends.

```
"max_batch_interval" = "20",
"max_batch_rows" = "300000",
"max_batch_size" = "209715200"
```

3. `max_error_number`

   The maximum number of error lines allowed within the sampling window. Must be greater than or equal to 0. The default is 0, which means no error lines are allowed.

   The sampling window is `max_batch_rows * 10`. That is, if the number of error lines is greater than `max_error_number` within the sampling window, the routine operation will be suspended, requiring manual intervention to check data quality problems.

   Rows that are filtered out by where conditions are not considered error rows.

4. `strict_mode`

   Whether to enable strict mode, the default is off. If enabled, the column type conversion of non-null raw data will be filtered if the result is NULL. Specify as:

   `"strict_mode" = "true"`

   The strict mode mode means strict filtering of column type conversions during the load process. The strict filtering strategy is as follows:

   1. For column type conversion, if strict mode is true, the wrong data will be filtered. The error data here refers to the fact that the original data is not null, and the result is a null value after participating in the column type conversion.
   2. When a loaded column is generated by a function transformation, strict mode has no effect on it.
   3. For a column type loaded with a range limit, if the original data can pass the type conversion normally, but cannot pass the range limit, strict mode will not affect it. For example, if the type is decimal(1,0) and the original data is 10, it is eligible for type conversion but not for column declarations. This data strict has no effect on it.

   strict mode and load relationship of source data

   Here is an example of a column type of TinyInt.

   > Note: When a column in a table allows a null value to be loaded

source data	source data example	string to int	strict_mode	result
null		N/A	true or false	NULL
not null	aaa or 2000	NULL	true	invalid data(filtered)
not null	aaa	NULL	false	NULL
not null	1	1	true or false	correct data

Here the column type is Decimal(1,0)

> Note: When a column in a table allows a null value to be loaded

source data	source data example	string to int	strict_mode	result
null		N/A	true or false	NULL
not null	aaa	NULL	true	invalid data(filtered)
not null	aaa	NULL	false	NULL
not null	1 or 10	1	true or false	correct data

> Note: 10 Although it is a value that is out of range, because its type meets the requirements of decimal, strict mode has no effect on it. 10 will eventually be filtered in other ETL processing flows. But it will not be filtered by strict mode.

5. `timezone`

   Specifies the time zone used by the import job. The default is to use the Session's timezone parameter. This parameter affects the results of all time zone-related functions involved in the import.

6. `format`

   Specify the import data format, the default is csv, and the json format is supported.

7. `jsonpaths`

   When the imported data format is json, the fields in the Json data can be extracted by specifying jsonpaths.

   ```
   -H "jsonpaths: [\"$.k2\", \"$.k1\"]"
   ```

8. `strip_outer_array`

   When the imported data format is json, strip_outer_array is true, indicating that the Json data is displayed in the form of an array, and each element in the data will be regarded as a row of data. The default value is false.

   ```
   -H "strip_outer_array: true"
   ```

9. `json_root`

   When the import data format is json, you can specify the root node of the Json data through json_root. Doris will extract the elements of the root node through json_root for parsing. Default is empty.

   `-H "json_root: $.RECORDS"`

10. `send_batch_parallelism`

Integer, Used to set the default parallelism for sending batch, if the value for parallelism exceed `max_send_batch_parallelism_per_job` in BE config, then the coordinator BE will use the value of `max_send_batch_parallelism_per_job`.

11. `load_to_single_tablet` Boolean type, True means that one task can only load data to one tablet in the corresponding partition at a time. The default value is false. This parameter can only be set when loading data into the OLAP table with random bucketing.

12. `partial_columns` Boolean type, True means that use partial column update, the default value is false, this parameter is only allowed to be set when the table model is Unique and Merge on Write is used. Multi-table does not support this parameter.

13. `max_filter_ratio` The maximum allowed filtering rate within the sampling window. Must be between 0 and 1. The default value is 0.

    The sampling window is `max_batch_rows * 10`. That is, if the number of error lines / total lines is greater than `max_filter_ratio` within the sampling window, the routine operation will be suspended, requiring manual intervention to check data quality problems.

    Rows that are filtered out by where conditions are not considered error rows.

14. `enclose` When the csv data field contains row delimiters or column delimiters, to prevent accidental truncation, single-byte characters can be specified as brackets for protection. For example, the column separator is ",", the bracket is """, and the data is"a,'b,c'", then "b,c" will be parsed as a field. Note: when the bracket is ", trim_double_quotes must be set to true.

15. `escape` Used to escape characters that appear in a csv field identical to the enclosing characters. For example, if the data is "a,'b,'c'", enclose is """, and you want"b,'c to be parsed as a field, you need to specify a single-byte escape character, such as "", and then modify the data to"a,' b,'c'".

- `FROM data_source [data_source_properties]`

  The type of data source. Currently supports:

```
FROM KAFKA
(
    "key1" = "val1",
    "key2" = "val2"
)
```

`data_source_properties` supports the following data source properties:

1. `kafka_broker_list`

Kafka's broker connection information. The format is ip:host. Separate multiple brokers with commas.

```
"kafka_broker_list" = "broker1:9092,broker2:9092"
```

2. `kafka_topic`

   Specifies the Kafka topic to subscribe to.

   ```
   "kafka_topic" = "my_topic"
   ```

3. `kafka_partitions/kafka_offsets`

   Specify the kafka partition to be subscribed to, and the corresponding starting offset of each partition. If a time is specified, consumption will start at the nearest offset greater than or equal to the time.

   offset can specify a specific offset from 0 or greater, or:

   - `OFFSET_BEGINNING`: Start subscription from where there is data.
   - `OFFSET_END`: subscribe from the end.
   - Time format, such as: "2021-05-22 11:00:00"

   If not specified, all partitions under topic will be subscribed from `OFFSET_END` by default.

   ```
   "kafka_partitions" = "0,1,2,3",
   "kafka_offsets" = "101,0,OFFSET_BEGINNING,OFFSET_END"
   ```

   ```
   "kafka_partitions" = "0,1,2,3",
   "kafka_offsets" = "2021-05-22 11:00:00,2021-05-22 11:00:00,2021-05-22 11:00:00"
   ```

   Note that the time format cannot be mixed with the OFFSET format.

4. `property`

   Specify custom kafka parameters. The function is equivalent to the "–property" parameter in the kafka shell.

   When the value of the parameter is a file, you need to add the keyword: "FILE:" before the value.

   For how to create a file, please refer to the CREATE FILE command documentation.

   For more supported custom parameters, please refer to the configuration items on the client side in the official CONFIGURATION document of librdkafka. Such as:

   ```
   "property.client.id" = "12345",
   "property.ssl.ca.location" = "FILE:ca.pem"
   ```

   1. When connecting to Kafka using SSL, you need to specify the following parameters:

      ```
      "property.security.protocol" = "ssl",
      "property.ssl.ca.location" = "FILE:ca.pem",
      "property.ssl.certificate.location" = "FILE:client.pem",
      ```

```
"property.ssl.key.location" = "FILE:client.key",
"property.ssl.key.password" = "abcdefg"
```

in:

`property.security.protocol` and `property.ssl.ca.location` are required to indicate the connection method is SSL and the location of the CA certificate.

If client authentication is enabled on the Kafka server side, thenAlso set:

```
"property.ssl.certificate.location"
"property.ssl.key.location"
"property.ssl.key.password"
```

They are used to specify the client's public key, private key, and password for the private key, respectively.

2. Specify the default starting offset of the kafka partition

   If `kafka_partitions/kafka_offsets` is not specified, all partitions are consumed by default.

   At this point, you can specify `kafka_default_offsets` to specify the starting offset. Defaults to `OFFSET_END`, i.e. subscribes from the end.

   Example:

```
"property.kafka_default_offsets" = "OFFSET_BEGINNING"
```

- comment Comment for the routine load job.

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

Example

1. Create a Kafka routine import task named test1 for example_tbl of example_db. Specify the column separator and group.id and client.id, and automatically consume all partitions by default, and start subscribing from the location where there is data (OFFSET_BEGINNING)

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
COLUMNS TERMINATED BY ",",
COLUMNS(k1, k2, k3, v1, v2, v3 = k1 * 100)
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "false"
)
```

```
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "property.group.id" = "xxx",
    "property.client.id" = "xxx",
    "property.kafka_default_offsets" = "OFFSET_BEGINNING"
);
```

2. Create a Kafka routine dynamic multiple tables import task named “test1” for the “example_db”. Specify the column delimiter, group.id, and client.id, and automatically consume all partitions, subscribing from the position with data (OFFSET_BEGINNING).

Assuming that we need to import data from Kafka into tables “test1” and “test2” in the “example_db”, we create a routine import task named “test1”. At the same time, we write the data in “test1” and “test2” to a Kafka topic named “my_topic” so that data from Kafka can be imported into both tables through a routine import task.

```
CREATE ROUTINE LOAD example_db.test1
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "false"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "property.group.id" = "xxx",
    "property.client.id" = "xxx",
    "property.kafka_default_offsets" = "OFFSET_BEGINNING"
);
```

3. Create a Kafka routine import task named test1 for example_tbl of example_db. Import tasks are in strict mode.

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
COLUMNS(k1, k2, k3, v1, v2, v3 = k1 * 100),
PRECEDING FILTER k1 = 1,
WHERE k1 > 100 and k2 like "%doris%"
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "true"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "kafka_partitions" = "0,1,2,3",
    "kafka_offsets" = "101,0,0,200"
);
```

4. Import data from the Kafka cluster through SSL authentication. Also set the client.id parameter. The import task is in non-strict mode and the time zone is Africa/Abidjan

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
COLUMNS(k1, k2, k3, v1, v2, v3 = k1 * 100),
WHERE k1 > 100 and k2 like "%doris%"
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "false",
    "timezone" = "Africa/Abidjan"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "property.security.protocol" = "ssl",
    "property.ssl.ca.location" = "FILE:ca.pem",
    "property.ssl.certificate.location" = "FILE:client.pem",
    "property.ssl.key.location" = "FILE:client.key",
    "property.ssl.key.password" = "abcdefg",
    "property.client.id" = "my_client_id"
);
```

5. Import data in Json format. By default, the field name in Json is used as the column name mapping. Specify to import three partitions 0, 1, and 2, and the starting offsets are all 0

```
CREATE ROUTINE LOAD example_db.test_json_label_1 ON table1
COLUMNS(category,price,author)
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "false",
    "format" = "json"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "kafka_partitions" = "0,1,2",
    "kafka_offsets" = "0,0,0"
);
```

6. Import Json data, extract fields through Jsonpaths, and specify the root node of the Json document

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
COLUMNS(category, author, price, timestamp, dt=from_unixtime(timestamp, '%Y%m%d'))
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "false",
```

```
    "format" = "json",
    "jsonpaths" = "[\"$.category\",\"$.author\",\"$.price\",\"$.timestamp\"]",
    "json_root" = "$.RECORDS"
    "strip_outer_array" = "true"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "kafka_partitions" = "0,1,2",
    "kafka_offsets" = "0,0,0"
);
```

7. Create a Kafka routine import task named test1 for example_tbl of example_db. And use conditional filtering.

```
CREATE ROUTINE LOAD example_db.test1 ON example_tbl
WITH MERGE
COLUMNS(k1, k2, k3, v1, v2, v3),
WHERE k1 > 100 and k2 like "%doris%",
DELETE ON v3 >100
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
    "strict_mode" = "false"
)
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "kafka_partitions" = "0,1,2,3",
    "kafka_offsets" = "101,0,0,200"
);
```

8. Import data to Unique with sequence column Key model table

```
CREATE ROUTINE LOAD example_db.test_job ON example_tbl
COLUMNS TERMINATED BY ",",
COLUMNS(k1,k2,source_sequence,v1,v2),
ORDER BY source_sequence
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "30",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200"
) FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092,broker3:9092",
    "kafka_topic" = "my_topic",
    "kafka_partitions" = "0,1,2,3",
    "kafka_offsets" = "101,0,0,200"
);
```

9. Consume from a specified point in time

```
CREATE ROUTINE LOAD example_db.test_job ON example_tbl
PROPERTIES
(
    "desired_concurrent_number"="3",
    "max_batch_interval" = "30",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200"
) FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092",
    "kafka_topic" = "my_topic",
    "property.kafka_default_offsets" = "2021-05-21 10:00:00"
);
```

Keywords

```
CREATE, ROUTINE, LOAD, CREATE LOAD
```

Best Practice

Partition and Offset for specified consumption

Doris supports the specified Partition and Offset to start consumption, and also supports the function of consumption at a specified time point. The configuration relationship of the corresponding parameters is described here.

There are three relevant parameters:

- `kafka_partitions`: Specify a list of partitions to be consumed, such as "0, 1, 2, 3".
- `kafka_offsets`: Specify the starting offset of each partition, which must correspond to the number of `kafka_partitions` list. For example: "1000, 1000, 2000, 2000"
- `property.kafka_default_offsets`: Specifies the default starting offset of the partition.

When creating an import job, these three parameters can have the following combinations:

Composition	`kafka_partitions`	`kafka_offsets`	`property.kafka_default_offsets`	Behavior
1	No	No	No	The system will automatically find all partitions corresponding to the topic and start consumption from OFFSET_END
2	No	No	Yes	The system will automatically find all partitions corresponding to the topic and start consumption from the location specified by default offset
3	Yes	No	No	The system will start consumption from OFFSET_END of the specified partition
4	Yes	Yes	No	The system will start consumption from the specified offset of the specified partition
5	Yes	No	Yes	The system will start consumption from the specified partition, the location specified by default offset

ALTER-ROUTINE-LOAD

ALTER-ROUTINE-LOAD

Name

ALTER ROUTINE LOAD

Description

This syntax is used to modify an already created routine import job.

Only jobs in the PAUSED state can be modified.

grammar:

```
ALTER ROUTINE LOAD FOR [db.]job_name
[job_properties]
FROM data_source
[data_source_properties]
```

1. `[db.]job_name`

   Specifies the job name to modify.

2. `tbl_name`

   Specifies the name of the table to be imported.

3. `job_properties`

   Specifies the job parameters that need to be modified. Currently, only the modification of the following parameters is supported:

   1. `desired_concurrent_number`
   2. `max_error_number`
   3. `max_batch_interval`
   4. `max_batch_rows`
   5. `max_batch_size`
   6. `jsonpaths`
   7. `json_root`
   8. `strip_outer_array`
   9. `strict_mode`
   10. `timezone`
   11. `num_as_string`
   12. `fuzzy_parse`
   13. `partial_columns`
   14. `max_filter_ratio`

4. `data_source`

   The type of data source. Currently supports:

   KAFKA

5. `data_source_properties`

   Relevant properties of the data source. Currently only supports:

   1. `kafka_partitions`
   2. `kafka_offsets`
   3. `kafka_broker_list`
   4. `kafka_topic`
   5. Custom properties, such as `property.group.id`

   Note:

   1. `kafka_partitions` and `kafka_offsets` are used to modify the offset of the kafka partition to be consumed, only the currently consumed partition can be modified. Cannot add partition.

Example

1. Change `desired_concurrent_number` to 1

```
ALTER ROUTINE LOAD FOR db1.label1
PROPERTIES
(
    "desired_concurrent_number" = "1"
);
```

2. Modify `desired_concurrent_number` to 10, modify the offset of the partition, and modify the group id.

```
ALTER ROUTINE LOAD FOR db1.label1
PROPERTIES
(
    "desired_concurrent_number" = "10"
)
FROM kafka
(
    "kafka_partitions" = "0, 1, 2",
    "kafka_offsets" = "100, 200, 100",
    "property.group.id" = "new_group"
);
```

Keywords

```
ALTER, ROUTINE, LOAD
```

Best Practice

PAUSE-ROUTINE-LOAD

PAUSE-ROUTINE-LOAD

Name

PAUSE ROUTINE LOAD

Description

Used to pause a Routine Load job. A suspended job can be rerun with the RESUME command.

```
PAUSE [ALL] ROUTINE LOAD FOR job_name
```

Example

1. Pause the routine import job named test1.

   ```
   PAUSE ROUTINE LOAD FOR test1;
   ```

2. Pause all routine import jobs.

   ```
   PAUSE ALL ROUTINE LOAD;
   ```

Keywords

```
PAUSE, ROUTINE, LOAD
```

Best Practice

RESUME-ROUTINE-LOAD

RESUME-ROUTINE-LOAD

Name

RESUME ROUTINE LOAD

Description

Used to restart a suspended Routine Load job. The restarted job will continue to consume from the previously consumed offset.

```
RESUME [ALL] ROUTINE LOAD FOR job_name
```

Example

1. Restart the routine import job named test1.

   ```
   RESUME ROUTINE LOAD FOR test1;
   ```

2. Restart all routine import jobs.

   ```
   RESUME ALL ROUTINE LOAD;
   ```

Keywords

```
RESUME, ROUTINE, LOAD
```

Best Practice

STOP-ROUTINE-LOAD

STOP-ROUTINE-LOAD

Name

STOP ROUTINE LOAD

Description

User stops a Routine Load job. A stopped job cannot be rerun.

```
STOP ROUTINE LOAD FOR job_name;
```

Example

1. Stop the routine import job named test1.

   ```
   STOP ROUTINE LOAD FOR test1;
   ```

Keywords

```
STOP, ROUTINE, LOAD
```

Best Practice

CREATE-SYNC-JOB

CREATE-SYNC-JOB

Name

CREATE SYNC JOB

Description

The data synchronization (Sync Job) function supports users to submit a resident data synchronization job, and incrementally synchronizes the CDC (Change Data Capture) of the user's data update operation in the Mysql database by reading the Binlog log from the specified remote address. Features.

Currently, the data synchronization job only supports connecting to Canal, obtaining the parsed Binlog data from the Canal Server and importing it into Doris.

Users can view the data synchronization job status through SHOW SYNC JOB.

grammar:

```
CREATE SYNC [db.]job_name
 (
 channel_desc,
 channel_desc
 ...
 )
binlog_desc
```

1. `job_name`

   The synchronization job name is the unique identifier of the job in the current database. Only one job with the same `job_name` can be running.

2. `channel_desc`

   The data channel under the job is used to describe the mapping relationship between the mysql source table and the doris target table.

   grammar:

   ```
   FROM mysql_db.src_tbl INTO des_tbl
   [columns_mapping]
   ```

   1. `mysql_db.src_tbl`

      Specify the database and source table on the mysql side.

   2. `des_tbl`

Specify the target table on the doris side. Only unique tables are supported, and the batch delete function of the table needs to be enabled (see the 'batch delete function' of help alter table for how to enable it).

3. `column_mapping`

   Specifies the mapping relationship between the columns of the mysql source table and the doris target table. If not specified, FE will default the columns of the source table and the target table to one-to-one correspondence in order.

   The form col_name = expr is not supported for columns.

   Example:

   ```
   Suppose the target table column is (k1, k2, v1),

   Change the order of columns k1 and k2
   (k2, k1, v1)

   Ignore the fourth column of the source data
   (k2, k1, v1, dummy_column)
   ```

3. `binlog_desc`

   Used to describe the remote data source, currently only one canal is supported.

   grammar:

   ```
   FROM BINLOG
   (
       "key1" = "value1",
       "key2" = "value2"
   )
   ```

   1. The properties corresponding to the Canal data source, prefixed with `canal.`

      1. canal.server.ip: address of canal server
      2. canal.server.port: the port of the canal server
      3. canal.destination: the identity of the instance
      4. canal.batchSize: The maximum batch size obtained, the default is 8192
      5. canal.username: username of instance
      6. canal.password: the password of the instance
      7. canal.debug: optional, when set to true, the batch and details of each row of data will be printed out

Example

1. Simply create a data synchronization job named `job1` for `test_tbl` of `test_db`, connect to the local Canal server, corresponding to the Mysql source table `mysql_db1.tbl1`.

   ```
   CREATE SYNC `test_db`.`job1`
   (
   FROM `mysql_db1`.`tbl1` INTO `test_tbl`
   ```

```
)
FROM BINLOG
(
"type" = "canal",
"canal.server.ip" = "127.0.0.1",
"canal.server.port" = "11111",
"canal.destination" = "example",
"canal.username" = "",
"canal.password" = ""
);
```

2. Create a data synchronization job named `job1` for multiple tables of `test_db`, corresponding to multiple Mysql source tables one-to-one, and explicitly specify the column mapping.

```
CREATE SYNC `test_db`.`job1`
(
FROM `mysql_db`.`t1` INTO `test1` (k1, k2, v1) ,
FROM `mysql_db`.`t2` INTO `test2` (k3, k4, v2)
)
FROM BINLOG
(
"type" = "canal",
"canal.server.ip" = "xx.xxx.xxx.xx",
"canal.server.port" = "12111",
"canal.destination" = "example",
"canal.username" = "username",
"canal.password" = "password"
);
```

Keywords

```
CREATE, SYNC, JOB
```

Best Practice

PAUSE SYNC JOB

Description

Pause a running resident data synchronization job in a database identified by `job_name`. The suspended job will stop synchronizing data while retaining its latest consumption position until it is resumed by the user.

Syntax

```
PAUSE SYNC JOB [<db>.]<job_name>
```

Required Parameters

1. `<job_name>`

> Specifies the name of the synchronization job to be paused.

Optional Parameters

1. `<db>` > If a database is specified using the `[<db>.]` prefix, the job is located in that database; otherwise, the current database is used.

Access Control Requirements

Any user or role can perform this operation.

Example

1. Pause the data synchronization job named `job_name`.

```
PAUSE SYNC JOB `job_name`;
```

RESUME-SYNC-JOB

RESUME-SYNC-JOB

Name

RESUME SYNC JOB

Description

Resume a resident data synchronization job whose current database has been suspended by `job_name`, and the job will continue to synchronize data from the latest position before the last suspension.

grammar:

```
RESUME SYNC JOB [db.]job_name
```

Example

1. Resume the data synchronization job named `job_name`

   ```
   RESUME SYNC JOB `job_name`;
   ```

Keywords

```
RESUME, SYNC, LOAD
```

Best Practice

STOP-SYNC-JOB

STOP-SYNC-JOB

Name

STOP SYNC JOB

Description

Stop a non-stop resident data synchronization job in a database by `job_name`.

grammar:

```
STOP SYNC JOB [db.]job_name
```

Example

1. Stop the data sync job named `job_name`

   ```
   STOP SYNC JOB `job_name`;
   ```

Keywords

```
STOP, SYNC, JOB
```

Best Practice

CANCEL-LOAD

CANCEL-LOAD

Name

CANCEL LOAD

Description

This statement is used to undo an import job for the specified label. Or batch undo import jobs via fuzzy matching

```
CANCEL LOAD
[FROM db_name]
WHERE [LABEL = "load_label" | LABEL like "label_pattern" | STATE = "PENDING/ETL/LOADING"]
```

Notice: Cancel by State is supported since 1.2.0.

Example

1. Cancel the import job whose label is `example_db_test_load_label` on the database example_db

   ```
   CANCEL LOAD
   FROM example_db
   WHERE LABEL = "example_db_test_load_label";
   ```

2. Cancel all import jobs containing example* on the database example*db.

   ```
   CANCEL LOAD
   FROM example_db
   WHERE LABEL like "example_";
   ```

3. Cancel all import jobs which state are “LOADING”

   ```
   CANCEL LOAD
   FROM example_db
   WHERE STATE = "loading";
   ```

> **提示**
>
> 该功能自 Apache Doris 1.2 版本起支持

Keywords

```
CANCEL, LOAD
```

Best Practice

1. Only pending import jobs in PENDING, ETL, LOADING state can be canceled.
2. When performing batch undo, Doris does not guarantee the atomic undo of all corresponding import jobs. That is, it is possible that only some of the import jobs were successfully undone. The user can view the job status through the SHOW LOAD statement and try to execute the CANCEL LOAD statement repeatedly.

CLEAN-LABEL

CLEAN-LABEL

Name

CLEAN LABEL

Description

For manual cleanup of historical load jobs. After cleaning, the Label can be reused.

Syntax:

```
CLEAN LABEL [label] FROM db;
```

Example

1. Clean label label1 from database db1

   ```
   CLEAN LABEL label1 FROM db1;
   ```

2. Clean all labels from database db1

   ```
   CLEAN LABEL FROM db1;
   ```

Keywords

```
CLEAN, LABEL
```

Best Practice

INSERT

INSERT

Name

INSERT

Description

The change statement is to complete the data insertion operation.

```
INSERT INTO table_name
    [ PARTITION (p1, ...) ]
    [ WITH LABEL label]
    [ (column [, ...]) ]
    [ [ hint [, ...] ] ]
    { VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | query }
```

Parameters

> tablet_name: The destination table for importing data. Can be of the form `db_name.table_name`
>
> partitions: Specify the partitions to be imported, which must be partitions that exist in `table_name`. Multiple partition names are separated by commas
>
> label: specify a label for the Insert task
>
> column_name: The specified destination column, must be a column that exists in `table_name`
>
> expression: the corresponding expression that needs to be assigned to a column
>
> DEFAULT: let the corresponding column use the default value
>
> query: a common query, the result of the query will be written to the target
>
> hint: some indicator used to indicate the execution behavior of `INSERT`. You can choose one of this values: `/*+ STREAMING */`, `/*+ SHUFFLE */` or `/+ *NOSHUFFLE* /`. 1. STREAMING: At present, it has no practical effect and is only reserved for compatibility with previous versions. (In the previous version, adding this hint would return a label, but now it defaults to returning a label) 2. SHUFFLE: When the target table is a partition table, enabling this hint will do repartiiton. 3. NOSHUFFLE: Even if the target table is a partition table, repartiiton will not be performed, but some other operations will be performed to ensure that the data is correctly dropped into each partition.

For a Unique table with merge-on-write enabled, you can also perform partial columns updates using the insert statement. To perform partial column updates with the insert statement, you need to set the session variable enable_unique_key_partial_update to true (the default value for this variable is false, meaning partial columns

updates with the insert statement are not allowed by default). When performing partial columns updates, the columns being inserted must contain at least all the Key columns and specify the columns you want to update. If the Key column values for the inserted row already exist in the original table, the data in the row with the same key column values will be updated. If the Key column values for the inserted row do not exist in the original table, a new row will be inserted into the table. In this case, columns not specified in the insert statement must either have default values or be nullable. These missing columns will first attempt to be populated with default values, and if a column has no default value, it will be filled with null. If a column cannot be null, the insert operation will fail.

Please note that the default value of the session variable `enable_insert_strict`, which controls whether the insert statement operates in strict mode, is true. In other words, the insert statement is in strict mode by default, and in this mode, updating non-existing keys in partial column updates is not allowed. Therefore, when using the insert statement for partial columns update and wishing to insert non-existing keys, you need to set `enable_unique_key_partial_update` to true and simultaneously set `enable_insert_strict` to false.

Notice:

When executing the `INSERT` statement, the default behavior is to filter the data that does not conform to the target table format, such as the string is too long. However, for business scenarios that require data not to be filtered, you can set the session variable `enable_insert_strict` to `true` to ensure that `INSERT` will not be executed successfully when data is filtered out.

Example

The `test` table contains two columns `c1`, `c2`.

1. Import a row of data into the `test` table

```
INSERT INTO test VALUES (1, 2);
INSERT INTO test (c1, c2) VALUES (1, 2);
INSERT INTO test (c1, c2) VALUES (1, DEFAULT);
INSERT INTO test (c1) VALUES (1);
```

The first and second statements have the same effect. When no target column is specified, the column order in the table is used as the default target column. The third and fourth statements express the same meaning, use the default value of the `c2` column to complete the data import.

2. Import multiple rows of data into the `test` table at one time

```
INSERT INTO test VALUES (1, 2), (3, 2 + 2);
INSERT INTO test (c1, c2) VALUES (1, 2), (3, 2 * 2);
INSERT INTO test (c1) VALUES (1), (3);
INSERT INTO test (c1, c2) VALUES (1, DEFAULT), (3, DEFAULT);
```

The first and second statements have the same effect, import two pieces of data into the `test` table at one time The effect of the third and fourth statements is known, and the default value of the `c2` column is used to import two pieces of data into the `test` table

3. Import a query result into the `test` table

```
INSERT INTO test SELECT * FROM test2;
INSERT INTO test (c1, c2) SELECT * from test2;
```

4. Import a query result into the `test` table, specifying the partition and label

```
INSERT INTO test PARTITION(p1, p2) WITH LABEL `label1` SELECT * FROM test2;
INSERT INTO test WITH LABEL `label1` (c1, c2) SELECT * from test2;
```

Keywords

```
INSERT
```

Best Practice

1. View the returned results

   The INSERT operation is a synchronous operation, and the return of the result indicates the end of the operation. Users need to perform corresponding processing according to the different returned results.

   1. The execution is successful, the result set is empty

      If the result set of the insert corresponding to the select statement is empty, it will return as follows:

      ```
      mysql> insert into tbl1 select * from empty_tbl;
      Query OK, 0 rows affected (0.02 sec)
      ```

      `Query OK` indicates successful execution. `0 rows affected` means that no data was imported.

   2. The execution is successful, the result set is not empty

      In the case where the result set is not empty. The returned results are divided into the following situations:

      1. Insert executes successfully and is visible:

         ```
         mysql> insert into tbl1 select * from tbl2;
         Query OK, 4 rows affected (0.38 sec)
         {'label':'insert_8510c568-9eda-4173-9e36-6adc7d35291c', 'status':'visible', 'txnId':'4005'}

         mysql> insert into tbl1 with label my_label1 select * from tbl2;
         Query OK, 4 rows affected (0.38 sec)
         {'label':'my_label1', 'status':'visible', 'txnId':'4005'}

         mysql> insert into tbl1 select * from tbl2;
         Query OK, 2 rows affected, 2 warnings (0.31 sec)
         {'label':'insert_f0747f0e-7a35-46e2-affa-13a235f4020d', 'status':'visible', 'txnId':'4005'}

         mysql> insert into tbl1 select * from tbl2;
         Query OK, 2 rows affected, 2 warnings (0.31 sec)
         {'label':'insert_f0747f0e-7a35-46e2-affa-13a235f4020d', 'status':'committed', 'txnId':'4005'}
         ```

`Query OK` indicates successful execution. `4 rows affected` means that a total of 4 rows of data were imported. `2 warnings` indicates the number of lines to be filtered.

Also returns a json string:

```
{'label':'my_label1', 'status':'visible', 'txnId':'4005'}
{'label':'insert_f0747f0e-7a35-46e2-affa-13a235f4020d', 'status':'committed', 'txnId':'4005'}
{'label':'my_label1', 'status':'visible', 'txnId':'4005', 'err':'some other error'}
```

`label` is a user-specified label or an automatically generated label. Label is the ID of this Insert Into import job. Each import job has a unique Label within a single database.

`status` indicates whether the imported data is visible. Show `visible` if visible, `committed` if not visible.

`txnId` is the id of the import transaction corresponding to this insert.

The `err` field shows some other unexpected errors.

When you need to view the filtered rows, the user can pass the following statement

```
show load where label="xxx";
```

The URL in the returned result can be used to query the wrong data. For details, see the summary of **Viewing Error Lines** later.

Invisibility of data is a temporary state, this batch of data will eventually be visible

You can view the visible status of this batch of data with the following statement:

```
show transaction where id=4005;
```

If the `TransactionStatus` column in the returned result is `visible`, the representation data is visible.

3. Execution failed

Execution failure indicates that no data was successfully imported, and the following is returned:

```
mysql> insert into tbl1 select * from tbl2 where k1 = "a";
ERROR 1064 (HY000): all partitions have no load data. url: http://10.74.167.16:8042/api/_load_error_log?
file=__shard_2/error_log_insert_stmt_ba8bb9e158e4879-ae8de8507c0bf8a2_ba8bb9e158e4879_ae8de8507c0
```

Where `ERROR 1064 (HY000): all partitions have no load data` shows the reason for the failure. The following url can be used to query the wrong data:

```
show load warnings on "url";
```

You can view the specific error line.

2. Timeout time

The timeout for INSERT operations is controlled by session variable `insert_timeout`. The default is 4 hours. If it times out, the job will be canceled.

3. Label and atomicity

   The INSERT operation also guarantees the atomicity of imports, see the Import Transactions and Atomicity documentation.

   When using `CTE(Common Table Expressions)` as the query part in an insert operation, the `WITH LABEL` and `column` parts must be specified.

4. Filter Threshold

   Unlike other import methods, INSERT operations cannot specify a filter threshold (`max_filter_ratio`). The default filter threshold is 1, which means that rows with errors can be ignored.

   For business scenarios that require data not to be filtered, you can set session variable `enable_insert_strict` to `true` to ensure that when there is data When filtered out, `INSERT` will not be executed successfully.

5. Performance issues

   There is no single row insertion using the `VALUES` method. If you must use it this way, combine multiple rows of data into one INSERT statement for bulk commit.

SELECT

SELECT

Name

SELECT

description

Mainly introduces the use of Select syntax

grammar:

```
SELECT
    [hint_statement, ...]
    [ALL | DISTINCT | DISTINCTROW | ALL EXCEPT ( col_name1 [, col_name2, col_name3, ...] )]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
      [TABLET tabletid_list]
      [TABLESAMPLE sample_value [ROWS | PERCENT]
        [REPEATABLE pos_seek]]
    [WHERE where_condition]
    [GROUP BY [GROUPING SETS | ROLLUP | CUBE] {col_name | expr | position}]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [INTO OUTFILE 'file_name']
```

1. **Syntax Description:**

    1. select_expr, ... Columns retrieved and displayed in the result, when using an alias, as is optional.
    2. select_expr, ... Retrieved target table (one or more tables (including temporary tables generated by subqueries)
    3. where_definition retrieves the condition (expression), if there is a WHERE clause, the condition filters the row data. where_condition is an expression that evaluates to true for each row to be selected. Without the WHERE clause, the statement selects all rows. In WHERE expressions, you can use any MySQL supported functions and operators except aggregate functions
    4. `ALL | DISTINCT` : to refresh the result set, all is all, distinct/distinctrow will refresh the duplicate columns, the default is all
    5. `ALL EXCEPT`: Filter on the full (all) result set, except specifies the name of one or more columns to be excluded from the full result set. All matching column names will be ignored in the output.

This feature is supported since the Apache Doris 1.2 version

6. `INTO OUTFILE 'file_name'` : save the result to a new file (which did not exist before), the difference lies in the save format.

7. `Group by having`: Group the result set, and brush the result of group by when having appears. `Grouping Sets`, `Rollup`, `Cube` are extensions of group by, please refer to GROUPING SETS DESIGN for details.

8. `Order by`: Sort the final result, Order by sorts the result set by comparing the size of one or more columns.

   Order by is a time-consuming and resource-intensive operation, because all data needs to be sent to 1 node before it can be sorted, and the sorting operation requires more memory than the non-sorting operation.

   If you need to return the top N sorted results, you need to use the LIMIT clause; in order to limit memory usage, if the user does not specify the LIMIT clause, the first 65535 sorted results are returned by default.

9. `Limit n`: limit the number of lines in the output result, `limit m,n` means output n records starting from the mth line.You should use `order by` before you use `limit m,n`, otherwise the data may be inconsistent each time it is executed.

10. The `Having` clause does not filter the row data in the table, but filters the results produced by the aggregate function.

    Typically `having` is used with aggregate functions (eg :`COUNT(), SUM(), AVG(), MIN(), MAX()`) and `group by` clauses.

11. SELECT supports explicit partition selection using PARTITION containing a list of partitions or subpartitions (or both) following the name of the table in `table_reference`

12. `[TABLET tids] TABLESAMPLE n [ROWS | PERCENT] [REPEATABLE seek]`: Limit the number of rows read from the table in the FROM clause, select a number of Tablets pseudo-randomly from the table according to the specified number of rows or percentages, and specify the number of seeds in REPEATABLE to return the selected samples again. In addition, you can also manually specify the TableID, Note that this can only be used for OLAP tables.

13. `hint_statement`: hint in front of the selectlist indicates that hints can be used to influence the behavior of the optimizer in order to obtain the desired execution plan. Details refer to joinHint using document

Syntax constraints:

1. SELECT can also be used to retrieve calculated rows without referencing any table.
2. All clauses must be ordered strictly according to the above format, and a HAVING clause must be placed after the GROUP BY clause and before the ORDER BY clause.
3. The alias keyword AS is optional. Aliases can be used for group by, order by and having
4. Where clause: The WHERE statement is executed to determine which rows should be included in the GROUP BY section, and HAVING is used to determine which rows in the result set should be used.
5. The HAVING clause can refer to the total function, but the WHERE clause cannot refer to, such as count, sum, max, min, avg, at the same time, the where clause can refer to other functions except the total function. Column aliases cannot be used in the Where clause to define conditions.
6. Group by followed by with rollup can count the results one or more times.

Join query:

Doris supports JOIN syntax

```
JOIN
table_references:
    table_reference [, table_reference] ...
table_reference:
    table_factor
  | join_table
table_factor:
    tbl_name [[AS] alias]
        [{USE|IGNORE|FORCE} INDEX (key_list)]
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
        ON conditional_expr }
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON condition
  | table_reference LEFT [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [LEFT [OUTER]] JOIN table_factor
  | table_reference RIGHT [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [RIGHT [OUTER]] JOIN table_factor
join_condition:
    ON conditional_expr
```

UNION Grammar:

```
SELECT ...
UNION [ALL| DISTINCT] SELECT ......
[UNION [ALL| DISTINCT] SELECT ...]
```

`UNION` is used to combine the results of multiple `SELECT` statements into a single result set.

The column names in the first `SELECT` statement are used as the column names in the returned results. The selected columns listed in the corresponding position of each `SELECT` statement should have the same data type. (For example, the first column selected by the first statement should be of the same type as the first column selected by other statements.)

The default behavior of `UNION` is to remove duplicate rows from the result. The optional `DISTINCT` keyword has no effect other than the default, since it also specifies duplicate row removal. With the optional `ALL` keyword, no duplicate row removal occurs, and the result includes all matching rows in all `SELECT` statements

WITH statement:

To specify common table expressions, use the `WITH` clause with one or more comma-separated clauses. Each subclause provides a subquery that generates the result set and associates the name with the subquery. The following example defines `WITH` clauses in CTEs named `cte1` and `cte2`, and refers to the `WITH` clause below their top-level `SELECT`:

```
WITH
  cte1 AS (SELECT a, b FROM table1),
  cte2 AS (SELECT c, d FROM table2)
```

```
SELECT b, d FROM cte1 JOIN cte2
WHERE cte1.a = cte2.c;
```

In a statement containing the `WITH` clause, each CTE name can be referenced to access the corresponding CTE result set.

CTE names can be referenced in other CTEs, allowing CTEs to be defined based on other CTEs.

Recursive CTE is currently not supported.

example

1. Query the names of students whose ages are 18, 20, 25

   ```
   select Name from student where age in (18,20,25);
   ```

2. ALL EXCEPT Example

   ```
   -- Query all information except the students' age
   select * except(age) from student;
   ```

3. GROUP BY Example

   ```
   --Query the tb_book table, group by type, and find the average price of each type of book,
   select type,avg(price) from tb_book group by type;
   ```

4. DISTINCT Use

   ```
   --Query the tb_book table to remove duplicate type data
   select distinct type from tb_book;
   ```

5. ORDER BY Example

   Sort query results in ascending (default) or descending (DESC) order. Ascending NULL is first, descending NULL is last

   ```
   --Query all records in the tb_book table, sort them in descending order by id, and display three records
   select * from tb_book order by id desc limit 3;
   ```

6. LIKE fuzzy query

   Can realize fuzzy query, it has two wildcards: `%` and _, `%` can match one or more characters, _ can match one character

   ```
   --Find all books whose second character is h
   select * from tb_book where name like('_h%');
   ```

7. LIMIT limits the number of result rows

```
--1. Display 3 records in descending order
select * from tb_book order by price desc limit 3;

--2. Display 4 records from id=1
select * from tb_book where id limit 1,4;
```

8. CONCAT join multiple columns

```
--Combine name and price into a new string output
select id,concat(name,":",price) as info,type from tb_book;
```

9. Using functions and expressions

```
--Calculate the total price of various books in the tb_book table
select sum(price) as total,type from tb_book group by type;
--20% off price
select *,(price * 0.8) as "20%" from tb_book;
```

10. UNION Example

```
SELECT a FROM t1 WHERE a = 10 AND B = 1 ORDER by LIMIT 10
UNION
SELECT a FROM t2 WHERE a = 11 AND B = 2 ORDER by LIMIT 10;
```

11. WITH clause example

```
WITH cte AS
(
  SELECT 1 AS col1, 2 AS col2
  UNION ALL
  SELECT 3, 4
)
SELECT col1, col2 FROM cte;
```

12. JOIN Exampel

```
SELECT * FROM t1 LEFT JOIN (t2, t3, t4)
                ON (t2.a = t1.a AND t3.b = t1.b AND t4.c = t1.c)
```

Equivalent to

```
SELECT * FROM t1 LEFT JOIN (t2 CROSS JOIN t3 CROSS JOIN t4)
                ON (t2.a = t1.a AND t3.b = t1.b AND t4.c = t1.c)
```

13. INNER JOIN

```
SELECT t1.name, t2.salary
  FROM employee AS t1 INNER JOIN info AS t2 ON t1.name = t2.name;

SELECT t1.name, t2.salary
  FROM employee t1 INNER JOIN info t2 ON t1.name = t2.name;
```

14. LEFT JOIN

```
SELECT left_tbl.*
  FROM left_tbl LEFT JOIN right_tbl ON left_tbl.id = right_tbl.id
  WHERE right_tbl.id IS NULL;
```

15. RIGHT JOIN

```
mysql> SELECT * FROM t1 RIGHT JOIN t2 ON (t1.a = t2.a);
+------+------+------+------+
| a    | b    | a    | c    |
+------+------+------+------+
|    2 | y    |    2 | z    |
| NULL | NULL |    3 | w    |
+------+------+------+------+
```

16. TABLESAMPLE

```
--Pseudo-randomly sample 1000 rows in t1. Note that several Tablets are actually selected according to the statistics of the table, and the total number of selected Tablet rows may be greater than 1000, so if you want to explicitly return 1000 rows, you need to add Limit.
SELECT * FROM t1 TABLET(10001) TABLESAMPLE(1000 ROWS) REPEATABLE 2 limit 1000;
```

keywords

```
SELECT
```

Best Practice

1. ome additional knowledge about the SELECT clause

    - An alias can be specified for select_expr using AS alias_name. Aliases are used as column names in expressions and can be used in GROUP BY, ORDER BY or HAVING clauses. The AS keyword is a good habit to use when specifying aliases for columns.
    - table_references after FROM indicates one or more tables participating in the query. If more than one table is listed, a JOIN operation is performed. And for each specified table, you can define an alias for it
    - The selected column after SELECT can be referenced in ORDER IN and GROUP BY by column name, column alias or integer (starting from 1) representing the column position

      ```
      SELECT college, region, seed FROM tournament
        ORDER BY region, seed;

      SELECT college, region AS r, seed AS s FROM tournament
        ORDER BY r, s;

      SELECT college, region, seed FROM tournament
        ORDER BY 2, 3;
      ```

- If ORDER BY appears in a subquery and also applies to the outer query, the outermost ORDER BY takes precedence.
- If GROUP BY is used, the grouped columns are automatically sorted in ascending order (as if there was an ORDER BY statement followed by the same columns). If you want to avoid the overhead of GROUP BY due to automatic sorting, adding ORDER BY NULL can solve it:

```
SELECT a, COUNT(b) FROM test_table GROUP BY a ORDER BY NULL;
```

- When sorting columns in a SELECT using ORDER BY or GROUP BY, the server sorts values using only the initial number of bytes indicated by the max_sort_length system variable.
- Having clauses are generally applied last, just before the result set is returned to the MySQL client, and is not optimized. (while LIMIT is applied after HAVING)

  The SQL standard requires: HAVING must refer to a column in the GROUP BY list or used by an aggregate function. However, MySQL extends this by allowing HAVING to refer to columns in the Select clause list, as well as columns from outer subqueries.

  A warning is generated if the column referenced by HAVING is ambiguous. In the following statement, col2 is ambiguous:

```
SELECT COUNT(col1) AS col2 FROM t GROUP BY col2 HAVING col2 = 2;
```

- Remember not to use HAVING where WHERE should be used. HAVING is paired with GROUP BY.
- The HAVING clause can refer to aggregate functions, while WHERE cannot.

```
SELECT user, MAX(salary) FROM users
  GROUP BY user HAVING MAX(salary) > 10;
```

- The LIMIT clause can be used to constrain the number of rows returned by a SELECT statement. LIMIT can have one or two arguments, both of which must be non-negative integers.

```
/*Retrieve 6~15 rows in the result set*/
SELECT * FROM tbl LIMIT 5,10;
/*Then if you want to retrieve all rows after a certain offset is set, you can set a very large constant for the
second parameter. The following query fetches all data from row 96 onwards */
SELECT * FROM tbl LIMIT 95,18446744073709551615;
/*If LIMIT has only one parameter, the parameter specifies the number of rows that should be retrieved, and the
offset defaults to 0, that is, starting from the first row*/
```

- SELECT…INTO allows query results to be written to a file

2. Modifiers of the SELECT keyword

- deduplication

  The ALL and DISTINCT modifiers specify whether to deduplicate rows in the result set (should not be a column).

ALL is the default modifier, that is, all rows that meet the requirements are to be retrieved.

DISTINCT removes duplicate rows.

3. The main advantage of subqueries

- Subqueries allow structured queries so that each part of a statement can be isolated.
- Some operations require complex unions and associations. Subqueries provide other ways to perform these operations

4. Speed up queries

- Use Doris's partition and bucket as data filtering conditions as much as possible to reduce the scope of data scanning
- Make full use of Doris's prefix index fields as data filter conditions to speed up query speed

5. UNION

- Using only the union keyword has the same effect as using union disitnct. Since the deduplication work is more memory-intensive, the query speed using the union all operation will be faster and the memory consumption will be less. If users want to perform order by and limit operations on the returned result set, they need to put the union operation in the subquery, then select from subquery, and finally put the subquery and order by outside the subquery.

```
select * from (select age from student_01 union all select age from student_02) as t1
order by age limit 4;

+-------------+
| age |
+-------------+
| 18 |
| 19 |
| 20 |
| 21 |
+-------------+
4 rows in set (0.01 sec)
```

6. JOIN

- In the inner join condition, in addition to supporting equal-valued joins, it also supports unequal-valued joins. For performance reasons, it is recommended to use equal-valued joins.
- Other joins only support equivalent joins

DELETE

DELETE

Name

DELETE

Description

This statement is used to conditionally delete data in the specified table (base index) partition.

This operation will also delete the data of the rollup index related to this base index.

Syntax

Syntax 1: This syntax can only specify filter predicates

```
DELETE FROM table_name [PARTITION partition_name | PARTITIONS (partition_name [, partition_name])]
WHERE
column_name op { value | value_list } [ AND column_name op { value | value_list } ...];
```

Syntax 2: This syntax can only used on UNIQUE KEY model

```
DELETE FROM table_name
    [PARTITION partition_name | PARTITIONS (partition_name [, partition_name])]
    [USING additional_tables]
    WHERE condition
```

Required Parameters

- table_name: Specifies the table from which rows are removed.
- column_name: column belong to table_name
- op: Logical comparison operator, The optional types of op include: =, >, <, >=, <=, !=, in, not in
- value | value_list: value or value list used for logial comparison
- WHERE condition: Specifies a condition to use to select rows for removal

Optional Parameters

- PARTITION partition_name | PARTITIONS (partition_name [, partition_name]): Specifies the partition or partitions to select rows for removal
- table_alias: alias of table

- USING additional_tables: If you need to refer to additional tables in the WHERE clause to help identify the rows to be removed, then specify those table names in the USING clause. You can also use the USING clause to specify subqueries that identify the rows to be removed.

Note

1. Only conditions on the key column can be specified when using AGGREGATE (UNIQUE) model.
2. When the selected key column does not exist in a rollup, delete cannot be performed.
3. Wheny you use syntax 1, conditions can only have an “and” relationship. If you want to achieve an “or” relationship, you need to write the conditions in two DELETE statements.
4. In syntax 1, if it is a partitioned table, you can specify a partition. If not specified, Doris will infer partition from the given conditions. In two cases, Doris cannot infer the partition from conditions: 1) the conditions do not contain partition columns; 2) The operator of the partition column is not in. When a partition table does not specify the partition, or the partition cannot be inferred from the conditions, the session variable delete_without_partition needs to be true to make delete statement be applied to all partitions.

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

5. This statement may reduce query efficiency for a period of time after execution. The degree of impact depends on the number of delete conditions specified in the statement. The more conditions you specify, the greater the impact.

Example

1. Delete the data row whose k1 column value is 3 in my_table partition p1

```
DELETE FROM my_table PARTITION p1
    WHERE k1 = 3;
```

2. Delete the data rows where the value of column k1 is greater than or equal to 3 and the value of column k2 is “abc” in my_table partition p1

```
DELETE FROM my_table PARTITION p1
WHERE k1 >= 3 AND k2 = "abc";
```

3. Delete the data rows where the value of column k1 is greater than or equal to 3 and the value of column k2 is “abc” in my_table partition p1, p2

```
DELETE FROM my_table PARTITIONS (p1, p2)
WHERE k1 >= 3 AND k2 = "abc";
```

4. use the result of `t2` join `t3` to romve rows from `t1`,delete table only support unique key model

```
-- create t1, t2, t3 tables
CREATE TABLE t1
```

```
  (id INT, c1 BIGINT, c2 STRING, c3 DOUBLE, c4 DATE)
UNIQUE KEY (id)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1', "function_column.sequence_col" = "c4");

CREATE TABLE t2
  (id INT, c1 BIGINT, c2 STRING, c3 DOUBLE, c4 DATE)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1');

CREATE TABLE t3
  (id INT)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1');

-- insert data
INSERT INTO t1 VALUES
  (1, 1, '1', 1.0, '2000-01-01'),
  (2, 2, '2', 2.0, '2000-01-02'),
  (3, 3, '3', 3.0, '2000-01-03');

INSERT INTO t2 VALUES
  (1, 10, '10', 10.0, '2000-01-10'),
  (2, 20, '20', 20.0, '2000-01-20'),
  (3, 30, '30', 30.0, '2000-01-30'),
  (4, 4, '4', 4.0, '2000-01-04'),
  (5, 5, '5', 5.0, '2000-01-05');

INSERT INTO t3 VALUES
  (1),
  (4),
  (5);

-- remove rows from t1
DELETE FROM t1
  USING t2 INNER JOIN t3 ON t2.id = t3.id
  WHERE t1.id = t2.id;
```

the expect result is only remove the row where id = 1 in table t1

```
+----+----+----+--------+------------+
| id | c1 | c2 | c3     | c4         |
+----+----+----+--------+------------+
| 2  | 2  | 2  |    2.0 | 2000-01-02 |
| 3  | 3  | 3  |    3.0 | 2000-01-03 |
+----+----+----+--------+------------+
```

Keywords

```
DELETE
```

Best Practice

UPDATE

UPDATE

Name

UPDATE

Description

This statement is used to update the data. The UPDATE statement currently only supports the UNIQUE KEY model.

The UPDATE operation currently only supports updating the Value column. The update of the Key column can refer to Using FlinkCDC to update Key column. ##### Syntax

```
UPDATE target_table [table_alias]
    SET assignment_list
    WHERE condition

assignment_list:
    assignment [, assignment] ...

assignment:
    col_name = value

value:
    {expr | DEFAULT}
```

```
UPDATE target_table [table_alias]
    SET assignment_list
    [ FROM additional_tables]
    WHERE condition
```

Required Parameters

- target_table: The target table of the data to be updated. Can be of the form ‘db_name.table_name’
- assignment_list: The target column to be updated, in the format ‘col_name = value, col_name = value’
- where condition: the condition that is expected to be updated, an expression that returns true or false can be

Optional Parameters

- table_alias: alias of table
- FROM additional_tables: Specifies one or more tables to use for selecting rows to update or for setting new values. Note that if you want use target table here, you should give it a alias explicitly.

Note

The current UPDATE statement only supports row updates on the UNIQUE KEY model.

Example

The `test` table is a unique model table, which contains four columns: k1, k2, v1, v2. Where k1, k2 are keys, v1, v2 are values, and the aggregation method is Replace.

1. Update the v1 column in the 'test' table that satisfies the conditions k1 =1 , k2 =2 to 1

```
UPDATE test SET v1 = 1 WHERE k1=1 and k2=2;
```

2. Increment the v1 column of the k1=1 column in the 'test' table by 1

```
UPDATE test SET v1 = v1+1 WHERE k1=1;
```

3. use the result of `t2` join `t3` to update `t1`

```
-- create t1, t2, t3 tables
CREATE TABLE t1
  (id INT, c1 BIGINT, c2 STRING, c3 DOUBLE, c4 DATE)
UNIQUE KEY (id)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1', "function_column.sequence_col" = "c4");

CREATE TABLE t2
  (id INT, c1 BIGINT, c2 STRING, c3 DOUBLE, c4 DATE)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1');

CREATE TABLE t3
  (id INT)
DISTRIBUTED BY HASH (id)
PROPERTIES('replication_num'='1');

-- insert data
INSERT INTO t1 VALUES
  (1, 1, '1', 1.0, '2000-01-01'),
  (2, 2, '2', 2.0, '2000-01-02'),
  (3, 3, '3', 3.0, '2000-01-03');

INSERT INTO t2 VALUES
  (1, 10, '10', 10.0, '2000-01-10'),
  (2, 20, '20', 20.0, '2000-01-20'),
  (3, 30, '30', 30.0, '2000-01-30'),
  (4, 4, '4', 4.0, '2000-01-04'),
  (5, 5, '5', 5.0, '2000-01-05');

INSERT INTO t3 VALUES
  (1),
  (4),
  (5);

-- update t1
UPDATE t1
  SET t1.c1 = t2.c1, t1.c3 = t2.c3 * 100
```

```
  FROM t2 INNER JOIN t3 ON t2.id = t3.id
  WHERE t1.id = t2.id;
```

the expect result is only update the row where id = 1 in table t1

```
+----+----+----+--------+------------+
| id | c1 | c2 | c3     | c4         |
+----+----+----+--------+------------+
| 1  | 10 | 1  | 1000.0 | 2000-01-01 |
| 2  | 2  | 2  |    2.0 | 2000-01-02 |
| 3  | 3  | 3  |    3.0 | 2000-01-03 |
+----+----+----+--------+------------+
```

Keywords

```
UPDATE
```

Best Practice

EXPORT

EXPORT

Name

EXPORT

Description

The `EXPORT` command is used to export the data of a specified table to a designated location as a file. Currently, it supports exporting to remote storage such as HDFS, S3, BOS, and COS (Tencent Cloud) through Broker process, S3 protocol, or HDFS protocol.

`EXPORT` is an asynchronous operation, and the command submits an `EXPORT JOB` to Doris. The task will be successfully submitted and returns immediately. After execution, you can use the SHOW EXPORT to view the progress.

grammar

```
EXPORT TABLE table_name
[PARTITION (p1[,p2])]
[WHERE]
TO export_path
[opt_properties]
WITH BROKER/S3/HDFS
[broker_properties];
```

principle

The bottom layer of the `Export` statement actually executes the `select...outfile..` statement. The `Export` task will be decomposed into multiple `select...outfile..` statements to execute concurrently according to the value of the `parallelism` parameter. Each `select...outfile..` is responsible for exporting some tablets of table.

illustrate:

- `table_name`

  The table name of the table currently being exported. Only the export of Doris local table / View / External Table data is supported.

- `partition`

  It is possible to export only some specified partitions of the specified table

- `export_path`

  The exported file path can be a directory or a file directory with a file prefix, for example: `hdfs://path/to/my_file_`

- opt_properties

  Used to specify some export parameters.

  ```
  [PROPERTIES ("key"="value", ...)]
  ```

  The following parameters can be specified:

  - label: This parameter is optional, specifies the label of the export task. If this parameter is not specified, the system randomly assigns a label to the export task.
  - column_separator: Specifies the exported column separator, default is \t, mulit-bytes is supported. This parameter is only used for CSV file format.
  - line_delimiter: Specifies the line delimiter for export, the default is \n, mulit-bytes is supported. This parameter is only used for CSV file format.
  - timeout: The timeout period of the export job, the default is 2 hours, the unit is seconds.
  - columns: Specifies certain columns of the export job table
  - format: Specifies the file format, support: parquet, orc, csv, csv_with_names, csv_with_names_and_types.The default is csv format.
  - parallelism: The concurrency degree of the export job, the default is 1. The export job will be divided into select..outfile.. statements of the number of parallelism to execute concurrently. (If the value of parallelism is greater than the number of tablets in the table, the system will automatically set parallelism to the number of tablets, that is, each select..outfile.. statement is responsible for one tablet)
  - delete_existing_files: default false. If it is specified as true, you will first delete all files specified in the directory specified by the file_path, and then export the data to the directory.For example: “file_path” = “/user/tmp”, then delete all files and directory under “/user/”; “file_path” = “/user/tmp/”, then delete all files and directory under “/user/tmp/”
  - max_file_size: it is the limit for the size of a single file in the export job. If the result file exceeds this value, it will be split into multiple files. The valid range for max_file_size is [5MB, 2GB], with a default value of 1GB. (When exporting to the ORC file format, the actual size of the split files will be multiples of 64MB, for example, if max_file_size is specified as 5MB, the actual split size will be 64MB; if max_file_size is specified as 65MB, the actual split size will be 128MB.)
  - with_bom: The default is false. If it is set to true, the exported file is encoded in UTF8 with BOM (valid only for CSV-related file format).
  - timeout: This is the timeout parameter of the export job, the default timeout is 2 hours, and the unit is seconds.

  > Note that to use the delete_existing_files parameter, you also need to add the configuration enable_delete_existing_files = true to the fe.conf file and restart the FE. Only then will the

> `delete_existing_files` parameter take effect. Setting `delete_existing_files = true` is a dangerous operation and it is recommended to only use it in a testing environment.

- `WITH BROKER`

  The export function needs to write data to the remote storage through the Broker process. Here you need to define the relevant connection information for the broker to use.

```
WITH BROKER "broker_name"
("key"="value"[,...])

Broker related properties:
      username: user name
      password: password
      hadoop.security.authentication: specify the authentication method as kerberos
      kerberos_principal: specifies the principal of kerberos
      kerberos_keytab: specifies the path to the keytab file of kerberos. The file must be the absolute path to the file on
the server where the broker process is located. and can be accessed by the Broker process
```

- `WITH HDFS`

  You can directly write data to the remote HDFS.

```
WITH HDFS ("key"="value"[,...])

HDFS related properties:
      fs.defaultFS: namenode address and port
      hadoop.username: hdfs username
      dfs.nameservices: if hadoop enable HA, please set fs nameservice. See hdfs-site.xml
      dfs.ha.namenodes.[nameservice ID]: unique identifiers for each NameNode in the nameservice. See hdfs-site.xml
      dfs.namenode.rpc-address.[nameservice ID].[name node ID]: the fully-qualified RPC address for each NameNode to listen
on. See hdfs-site.xml
      dfs.client.failover.proxy.provider.[nameservice ID]: the Java class that HDFS clients use to contact the Active
NameNode, usually it is org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider

      For a kerberos-authentication enabled Hadoop cluster, additional properties need to be set:
      dfs.namenode.kerberos.principal: HDFS namenode service principal
      hadoop.security.authentication: kerberos
      hadoop.kerberos.principal: the Kerberos pincipal that Doris will use when connectiong to HDFS.
      hadoop.kerberos.keytab: HDFS client keytab location.
```

- `WITH S3`

  You can directly write data to a remote S3 object store

```
WITH S3 ("key"="value"[,...])

S3 related properties:
  AWS_ENDPOINT
  AWS_ACCESS_KEY
  AWS_SECRET_KEY
  AWS_REGION
  use_path_stype: (optional) default false . The S3 SDK uses the virtual-hosted style by default. However, some object
storage systems may not be enabled or support virtual-hosted style access. At this time, we can add the use_path_style
parameter to force the use of path style access method.
```

Example

export to local

> Export data to the local file system needs to add `enable_outfile_to_local = true` to the fe.conf and restart the Fe.

1. You can export the `test` table to a local store. Export csv format file by default.

```
EXPORT TABLE test TO "file:///home/user/tmp/";
```

2. You can export the k1 and k2 columns in `test` table to a local store, and set export label. Export csv format file by default.

```
EXPORT TABLE test TO "file:///home/user/tmp/"
PROPERTIES (
  "label" = "label1",
  "columns" = "k1,k2"
);
```

3. You can export the rows where `k1 < 50` in `test` table to a local store, and set column_separator to `,`. Export csv format file by default.

```
EXPORT TABLE test WHERE k1 < 50 TO "file:///home/user/tmp/"
PROPERTIES (
  "columns" = "k1,k2",
  "column_separator"=","
);
```

4. Export partitions p1 and p2 from the test table to local storage, with the default exported file format being csv.

```
EXPORT TABLE test PARTITION (p1,p2) TO "file:///home/user/tmp/"
PROPERTIES ("columns" = "k1,k2");
```

5. Export all data in the test table to local storage with a non-default file format.

```
// parquet file format
EXPORT TABLE test TO "file:///home/user/tmp/"
PROPERTIES (
  "columns" = "k1,k2",
  "format" = "parquet"
);

// orc file format
EXPORT TABLE test TO "file:///home/user/tmp/"
PROPERTIES (
  "columns" = "k1,k2",
  "format" = "orc"
);
```

```
// csv_with_names file format. Using 'AA' as the column delimiter and 'zz' as the line delimiter.
EXPORT TABLE test TO "file:///home/user/tmp/"
PROPERTIES (
  "format" = "csv_with_names",
  "column_separator"="AA",
  "line_delimiter" = "zz"
);

// csv_with_names_and_types file format
EXPORT TABLE test TO "file:///home/user/tmp/"
PROPERTIES (
  "format" = "csv_with_names_and_types"
);
```

6. set max_file_sizes

```
EXPORT TABLE test TO "file:///home/user/tmp/"
PROPERTIES (
  "format" = "parquet",
  "max_file_size" = "5MB"
);
```

When the exported file size is larger than 5MB, the data will be split into multiple files, with each file containing a maximum of 5MB.

7. set parallelism

```
EXPORT TABLE test TO "file:///home/user/tmp/"
PROPERTIES (
  "format" = "parquet",
  "max_file_size" = "5MB",
  "parallelism" = "5"
);
```

8. set delete_existing_files

```
EXPORT TABLE test TO "file:///home/user/tmp"
PROPERTIES (
  "format" = "parquet",
  "max_file_size" = "5MB",
  "delete_existing_files" = "true"
)
```

Before exporting data, all files and directories in the `/home/user/` directory will be deleted, and then the data will be exported to that directory.

export with S3

1. Export all data from the `testTbl` table to S3 using invisible character ‘’ as a delimiter for columns and rows.If you want to export data to minio, you also need to specify use_path_style=true.

```
EXPORT TABLE testTbl TO "s3://bucket/a/b/c"
PROPERTIES (
  "column_separator"="\\x07",
  "line_delimiter" = "\\x07"
```

```
) WITH s3 (
  "s3.endpoint" = "xxxxx",
  "s3.region" = "xxxxx",
  "s3.secret_key"="xxxx",
  "s3.access_key" = "xxxxx"
)
```

2. Export all data in the test table to HDFS in the format of parquet, limit the size of a single file to 1024MB, and reserve all files in the specified directory.

export with HDFS

1. Export all data from the `test` table to HDFS in `Parquet` format, with a limit of 512MB for the size of a single file in the export job, and retain all files under the specified directory.

```
EXPORT TABLE test TO "hdfs://hdfs_host:port/a/b/c/"
PROPERTIES(
    "format" = "parquet",
    "max_file_size" = "512MB",
    "delete_existing_files" = "false"
)
with HDFS (
"fs.defaultFS"="hdfs://hdfs_host:port",
"hadoop.username" = "hadoop"
);
```

export with Broker

You need to first start the broker process and add it to the FE before proceeding. 1. Export the `test` table to hdfs

```
EXPORT TABLE test TO "hdfs://hdfs_host:port/a/b/c"
WITH BROKER "broker_name"
(
  "username"="xxx",
  "password"="yyy"
);
```

2. Export partitions ‘p1’ and ‘p2’ from the ‘testTbl’ table to HDFS using ‘,’ as the column delimiter and specifying a label.

```
EXPORT TABLE testTbl PARTITION (p1,p2) TO "hdfs://hdfs_host:port/a/b/c"
PROPERTIES (
  "label" = "mylabel",
  "column_separator"=","
)
WITH BROKER "broker_name"
(
  "username"="xxx",
  "password"="yyy"
);
```

3. Export all data from the ‘testTbl’ table to HDFS using the non-visible character ‘’ as the column and row delimiter.

```
EXPORT TABLE testTbl TO "hdfs://hdfs_host:port/a/b/c"
PROPERTIES (
  "column_separator"="\\x07",
  "line_delimiter" = "\\x07"
)
WITH BROKER "broker_name"
(
  "username"="xxx",
  "password"="yyy"
)
```

Keywords

```
EXPORT
```

Best Practice

Concurrent Export

An Export job can be configured with the `parallelism` parameter to concurrently export data. The `parallelism` parameter specifies the number of threads to execute the `EXPORT Job`. Each thread is responsible for exporting a subset of the total tablets.

The underlying execution logic of an `Export Job`is actually the `SELECT INTO OUTFILE` statement. Each thread specified by the `parallelism` parameter executes independent `SELECT INTO OUTFILE` statements.

The specific logic for splitting an `Export Job` into multiple `SELECT INTO OUTFILE` is, to evenly distribute all the tablets of the table among all parallel threads. For example:

- If num(tablets) = 40 and parallelism = 3, then the three threads will be responsible for 14, 13, and 13 tablets, respectively.
- If num(tablets) = 2 and parallelism = 3, then Doris automatically sets the parallelism to 2, and each thread is responsible for one tablet.

When the number of tablets responsible for a thread exceeds the `maximum_tablets_of_outfile_in_export` value (default is 10, and can be modified by adding the `maximum_tablets_of_outfile_in_export` parameter in fe.conf), the thread will split the tablets which are responsibled for this thread into multiple `SELECT INTO OUTFILE` statements. For example:

- If a thread is responsible for 14 tablets and `maximum_tablets_of_outfile_in_export = 10`, then the thread will be responsible for two `SELECT INTO OUTFILE` statements. The first `SELECT INTO OUTFILE` statement exports 10 tablets, and the second `SELECT INTO OUTFILE` statement exports 4 tablets. The two `SELECT INTO OUTFILE` statements are executed serially by this thread.

memory limit

The query plan for an `Export Job` typically involves only `scanning and exporting`, and does not involve compute logic that requires a lot of memory. Therefore, the default memory limit of 2GB is usually sufficient to meet the requirements.

However, in certain scenarios, such as a query plan that requires scanning too many tablets on the same BE, or when there are too many data versions of tablets, it may result in insufficient memory. In these cases, you can adjust the session variable `exec_mem_limit` to increase the memory usage limit.

Precautions

- Exporting a large amount of data at one time is not recommended. The maximum recommended export data volume for an Export job is several tens of GB. An overly large export results in more junk files and higher retry costs. If the amount of table data is too large, it is recommended to export by partition.
- If the Export job fails, the generated files will not be deleted, and the user needs to delete it manually.
- The Export job only exports the data of the Base table / View / External table, not the data of the materialized view.
- The export job scans data and occupies IO resources, which may affect the query latency of the system.
- Currently, The `Export Job` is simply check whether the `Tablets version` is the same, it is recommended not to import data during the execution of the `Export Job`.
- The maximum number of partitions that an `Export job` allows is 2000. You can add a parameter to the fe.conf `maximum_number_of_export_partitions` and restart FE to modify the setting.

CANCEL-EXPORT

CANCEL-EXPORT

Name

> **Tips**
>
> This feature is supported since the Apache Doris 1.2.2 version

CANCEL EXPORT

Description

This statement is used to undo an export job for the specified label. Or batch undo export jobs via fuzzy matching

```
CANCEL EXPORT
[FROM db_name]
WHERE [LABEL = "export_label" | LABEL like "label_pattern" | STATE = "PENDING/IN_QUEUE/EXPORTING"]
```

Example

1. Cancel the export job whose label is `example_db_test_export_label` on the database example_db

   ```
   CANCEL EXPORT
   FROM example_db
   WHERE LABEL = "example_db_test_export_label" and STATE = "EXPORTING";
   ```

2. Cancel all export jobs containing example* on the database example*db.

   ```
   CANCEL EXPORT
   FROM example_db
   WHERE LABEL like "%example%";
   ```

3. Cancel all export jobs which state are “PENDING”

   ```
   CANCEL EXPORT
   FROM example_db
   WHERE STATE = "PENDING";
   ```

Keywords

```
CANCEL, EXPORT
```

Best Practice

1. Only pending export jobs in PENDING, IN_QUEUE,EXPORTING state can be canceled.
2. When performing batch undo, Doris does not guarantee the atomic undo of all corresponding export jobs. That is, it is possible that only some of the export jobs were successfully undone. The user can view the job status through the SHOW EXPORT statement and try to execute the CANCEL EXPORT statement repeatedly.
3. When the job of the `EXPORTING` state is revoked, part of the data may have been exported to the storage system, and the user needs to process (delete) this section to export data.

INSERT-OVERWRITE

INSERT OVERWRITE

Name

INSERT OVERWRITE

Description

The function of this statement is to overwrite a table or some partitions of a table

```
INSERT OVERWRITE table table_name
    [ PARTITION (p1, ...) ]
    [ WITH LABEL label]
    [ (column [, ...]) ]
    [ [ hint [, ...] ] ]
    { VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | query }
```

Parameters

> table_name: the destination table to overwrite. This table must exist. It can be of the form `db_name.table_name`
>
> partitions: the table partition that needs to be overwritten must be one of the existing partitions in `table_name` separated by a comma
>
> label: specify a label for the Insert task
>
> column_name: the specified destination column must be one of the existing columns in `table_name`
>
> expression: the corresponding expression that needs to be assigned to a column
>
> DEFAULT: let the column use the default value
>
> query: a common query, the result of the query will overwrite the target.
>
> hint: some indicator used to indicate the execution behavior of `INSERT`. You can choose one of this values: `/*+ STREAMING */`, `/*+ SHUFFLE */` or `/+ *NOSHUFFLE* /`.
>
> 1. STREAMING: At present, it has no practical effect and is only reserved for compatibility with previous versions. (In the previous version, adding this hint would return a label, but now it defaults to returning a label)
> 2. SHUFFLE: When the target table is a partition table, enabling this hint will do repartiiton.
> 3. NOSHUFFLE: Even if the target table is a partition table, repartiiton will not be performed, but some other operations will be performed to ensure that the data is correctly dropped into each partition.

Notice:

In the current version, the session variable `enable_insert_strict` is set to `true` by default. If some data that does not conform to the format of the target table is filtered out during the execution of the `INSERT OVERWRITE` statement, such as when overwriting a partition and not all partition conditions are satisfied, overwriting the target table will fail.

The `INSERT OVERWRITE` statement first creates a new table, inserts the data to be overwritten into the new table, and then atomically replaces the old table with the new table and modifies its name. Therefore, during the process of overwriting the table, the data in the old table can still be accessed normally until the overwriting is completed.

Example

Assuming there is a table named `test`. The table contains two columns `c1` and `c2`, and two partitions `p1` and `p2`

```
CREATE TABLE IF NOT EXISTS test (
  `c1` int NOT NULL DEFAULT "1",
  `c2` int NOT NULL DEFAULT "4"
) ENGINE=OLAP
UNIQUE KEY(`c1`)
PARTITION BY LIST (`c1`)
(
PARTITION p1 VALUES IN ("1","2","3"),
PARTITION p2 VALUES IN ("4","5","6")
)
DISTRIBUTED BY HASH(`c1`) BUCKETS 3
PROPERTIES (
  "replication_allocation" = "tag.location.default: 1",
  "in_memory" = "false",
  "storage_format" = "V2"
);
```

Overwrite Table

1. Overwrite the `test` table using the form of `VALUES`.

   ```
   // Single-row overwrite.
   INSERT OVERWRITE table test VALUES (1, 2);
   INSERT OVERWRITE table test (c1, c2) VALUES (1, 2);
   INSERT OVERWRITE table test (c1, c2) VALUES (1, DEFAULT);
   INSERT OVERWRITE table test (c1) VALUES (1);
   // Multi-row overwrite.
   INSERT OVERWRITE table test VALUES (1, 2), (3, 2 + 2);
   INSERT OVERWRITE table test (c1, c2) VALUES (1, 2), (3, 2 * 2);
   INSERT OVERWRITE table test (c1, c2) VALUES (1, DEFAULT), (3, DEFAULT);
   INSERT OVERWRITE table test (c1) VALUES (1), (3);
   ```

- The first and second statements have the same effect. If the target column is not specified during overwriting, the column order in the table will be used as the default target column. After the overwrite is successful, there is only one row of data in the `test` table.
- The third and fourth statements have the same effect. The unspecified column `c2` will be overwritten with the default value 4. After the overwrite is successful, there is only one row of data in the `test` table.

- The fifth and sixth statements have the same effect. Expressions (such as `2+2`, `2*2`) can be used in the statement. The result of the expression will be computed during the execution of the statement and then overwritten into the `test` table. After the overwrite is successful, there are two rows of data in the `test` table.
- The seventh and eighth statements have the same effect. The unspecified column `c2` will be overwritten with the default value 4. After the overwrite is successful, there are two rows of data in the `test` table.

2. Overwrite the `test` table in the form of a query statement. The data format of the `test2` table and the `test` table must be consistent. If they are not consistent, implicit data type conversion will be triggered.

```
INSERT OVERWRITE table test SELECT * FROM test2;
INSERT OVERWRITE table test (c1, c2) SELECT * from test2;
```

- The first and second statements have the same effect. The purpose of these statements is to take data from the `test2` table and overwrite the `test` table with the taken data. After the overwrite is successful, the data in the `test` table will be consistent with the data in the `test2` table.

3. Overwrite the `test` table and specify a label.

```
INSERT OVERWRITE table test WITH LABEL `label1` SELECT * FROM test2;
INSERT OVERWRITE table test WITH LABEL `label2` (c1, c2) SELECT * from test2;
```

- Users can use the `SHOW LOAD;` command to check the status of the job imported by this `label`. It should be noted that the label is unique.

Overwrite Table Partition

When using INSERT OVERWRITE to rewrite partitions, we actually encapsulate the following three steps into a single transaction and execute it. If it fails halfway through, the operations that have been performed will be rolled back: 1. Assuming that partition `p1` is specified to be rewritten, first create an empty temporary partition `pTMP` with the same structure as the target partition to be rewritten. 2. Write data to `pTMP`. 3. replace `p1` with the `pTMP` atom

The following is examples:

1. Overwrite partitions `P1` and `P2` of the `test` table using the form of `VALUES`.

```
// Single-row overwrite.
INSERT OVERWRITE table test PARTITION(p1,p2) VALUES (1, 2);
INSERT OVERWRITE table test PARTITION(p1,p2) (c1, c2) VALUES (1, 2);
INSERT OVERWRITE table test PARTITION(p1,p2) (c1, c2) VALUES (1, DEFAULT);
INSERT OVERWRITE table test PARTITION(p1,p2) (c1) VALUES (1);
// Multi-row overwrite.
INSERT OVERWRITE table test PARTITION(p1,p2) VALUES (1, 2), (4, 2 + 2);
INSERT OVERWRITE table test PARTITION(p1,p2) (c1, c2) VALUES (1, 2), (4, 2 * 2);
INSERT OVERWRITE table test PARTITION(p1,p2) (c1, c2) VALUES (1, DEFAULT), (4, DEFAULT);
INSERT OVERWRITE table test PARTITION(p1,p2) (c1) VALUES (1), (4);
```

Unlike overwriting an entire table, the above statements are overwriting partitions in the table. Partitions can be overwritten one at a time or multiple partitions can be overwritten at once. It should be noted that only data that satisfies the corresponding partition filtering condition can be overwritten successfully. If

there is data in the overwritten data that does not satisfy any of the partitions, the overwrite will fail. An example of a failure is shown below.

```
INSERT OVERWRITE table test PARTITION(p1,p2) VALUES (7, 2);
```

The data overwritten by the above statements (`c1=7`) does not satisfy the conditions of partitions `P1` and `P2`, so the overwrite will fail.

2. Overwrite partitions `P1` and `P2` of the `test` table in the form of a query statement. The data format of the `test2` table and the `test` table must be consistent. If they are not consistent, implicit data type conversion will be triggered.

```
INSERT OVERWRITE table test PARTITION(p1,p2) SELECT * FROM test2;
INSERT OVERWRITE table test PARTITION(p1,p2) (c1, c2) SELECT * from test2;
```

3. Overwrite partitions `P1` and `P2` of the `test` table and specify a label.

```
INSERT OVERWRITE table test PARTITION(p1,p2) WITH LABEL `label3` SELECT * FROM test2;
INSERT OVERWRITE table test PARTITION(p1,p2) WITH LABEL `label4` (c1, c2) SELECT * from test2;
```

Keywords

```
INSERT OVERWRITE
```

ANALYZE

ANALYZE

Name

ANALYZE

Description

This statement is used to collect statistical information for various columns.

```
ANALYZE < TABLE | DATABASE table_name | db_name >
    [ (column_name [, ...]) ]
    [ [ WITH SYNC ] [ WITH SAMPLE PERCENT | ROWS ] ];
```

- `table_name`: The specified target table. It can be in the format `db_name.table_name`.
- `column_name`: The specified target column. It must be an existing column in `table_name`. You can specify multiple column names separated by commas.
- `sync`: Collect statistics synchronously. Returns after collection. If not specified, it executes asynchronously and returns a JOB ID.
- `sample percent | rows`: Collect statistics with sampling. You can specify a sampling percentage or a number of sampling rows.

Example

Collect statistical data for a table with a 10% sampling rate:

```
ANALYZE TABLE lineitem WITH SAMPLE PERCENT 10;
```

Collect statistical data for a table with a sample of 100,000 rows:

```
ANALYZE TABLE lineitem WITH SAMPLE ROWS 100000;
```

Keywords

ANALYZE

OUTFILE

OUTFILE

Name

OURFILE

description

This statement is used to export query results to a file using the `SELECT INTO OUTFILE` command. Currently, it supports exporting to remote storage, such as HDFS, S3, BOS, COS (Tencent Cloud), through the Broker process, S3 protocol, or HDFS protocol.

grammar:

```
query_stmt
INTO OUTFILE "file_path"
[format_as]
[properties]
```

illustrate:

1. file_path

   file_path points to the path where the file is stored and the file prefix. Such as `hdfs://path/to/my_file_`.

   ```
   The final filename will consist of `my_file_`, the file number and the file format suffix. The file serial number starts
   from 0, and the number is the number of files to be divided. Such as:

   my_file_abcdefg_0.csv
   my_file_abcdefg_1.csv
   my_file_abcdegf_2.csv
   ```

   You can also omit the file prefix and specify only the file directory, such as: `hdfs://path/to/`

2. format_as

   ```
   FORMAT AS CSV
   ```

   Specifies the export format. Supported formats include CSV, PARQUET, CSV_WITH_NAMES, CSV_WITH_NAMES_AND_TYPES and ORC. Default is CSV. > Note: PARQUET, CSV_WITH_NAMES, CSV_WITH_NAMES_AND_TYPES, and ORC are supported starting in version 1.2 .

3. properties

Specify related properties. Currently exporting via the Broker process, S3 protocol, or HDFS protocol is supported.

```
grammar:
[PROPERTIES ("key"="value", ...)]
The following properties are supported:

File related properties
    column_separator: column separator,is only for CSV format. mulit-bytes is supported starting in version 1.2, such as: "\\x01", "abc".
    line_delimiter: line delimiter,is only for CSV format. mulit-bytes supported starting in version 1.2, such as: "\\x01", "abc".
    max_file_size: the size limit of a single file, if the result exceeds this value, it will be cut into multiple files, the value range of max_file_size is [5MB, 2GB] and the default is 1GB. (When specified that the file format is ORC, the size of the actual division file will be a multiples of 64MB, such as: specify max_file_size = 5MB, and actually use 64MB as the division; specify max_file_size = 65MB, and will actually use 128MB as cut division points.)
    delete_existing_files: default `false`. If it is specified as true, you will first delete all files specified in the directory specified by the file_path, and then export the data to the directory.For example: "file_path" = "/user/tmp", then delete all files and directory under "/user/"; "file_path" = "/user/tmp/", then delete all files and directory under "/user/tmp/"
    file_suffix: Specify the suffix of the export file. If this parameter is not specified, the default suffix for the file format will be used.

Broker related properties need to be prefixed with `broker.`:
    broker.name: broker name
    broker.hadoop.security.authentication: specify the authentication method as kerberos
    broker.kerberos_principal: specifies the principal of kerberos
    broker.kerberos_keytab: specifies the path to the keytab file of kerberos. The file must be the absolute path to the file on the server where the broker process is located. and can be accessed by the Broker process

HDFS related properties:
    fs.defaultFS: namenode address and port
    hadoop.username: hdfs username
    dfs.nameservices: if hadoop enable HA, please set fs nameservice. See hdfs-site.xml
    dfs.ha.namenodes.[nameservice ID]: unique identifiers for each NameNode in the nameservice. See hdfs-site.xml
    dfs.namenode.rpc-address.[nameservice ID].[name node ID]: the fully-qualified RPC address for each NameNode to listen on. See hdfs-site.xml
    dfs.client.failover.proxy.provider.[nameservice ID]: the Java class that HDFS clients use to contact the Active NameNode, usually it is org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider

    For a kerberos-authentication enabled Hadoop cluster, additional properties need to be set:
    dfs.namenode.kerberos.principal: HDFS namenode service principal
    hadoop.security.authentication: kerberos
    hadoop.kerberos.principal: the Kerberos pincipal that Doris will use when connectiong to HDFS.
    hadoop.kerberos.keytab: HDFS client keytab location.

For the S3 protocol, you can directly execute the S3 protocol configuration:
s3.endpoint
s3.access_key
s3.secret_key
s3.region
use_path_stype: (optional) default false . The S3 SDK uses the virtual-hosted style by default. However, some object storage systems may not be enabled or support virtual-hosted style access. At this time, we can add the use_path_style parameter to force the use of path style access method.
```

> Note that to use the `delete_existing_files` parameter, you also need to add the configuration `enable_delete_existing_files = true` to the fe.conf file and restart the FE. Only then will the `delete_existing_files` parameter take effect. Setting `delete_existing_files = true` is a dangerous operation and it is recommended to only use it in a testing environment.

4. Data Types for Export

   All file formats support the export of basic data types, while only csv/orc/csv_with_names/csv_with_names_and_types currently support the export of complex data types (ARRAY/MAP/STRUCT). Nested complex data types are not supported.

5. Concurrent Export

   Setting the session variable `set enable_parallel_outfile = true;` enables concurrent export using outfile. For detailed usage, see Export Query Result.

6. Export to Local

   To export to a local file, you need configure `enable_outfile_to_local=true` in fe.conf.

```
select * from tbl1 limit 10
INTO OUTFILE "file:///home/work/path/result_";
```

DataType Mapping

Parquet and ORC file formats have their own data types. The export function of Doris can automatically export the Doris data types to the corresponding data types of the Parquet/ORC file format. The following are the data type mapping relationship of the Doris data types and the Parquet/ORC file format data types:

1. The mapping relationship between the Doris data types to the ORC data types is: | Doris Type | Orc Type | | — | — | | boolean | boolean | | tinyint | tinyint | | smallint | smallint | | int | int | | bigint | bigint | | largeInt | string | | date | string | | datev2 | string | | datetime | string | | datetimev2 | timestamp | | float | float | | double | double | | char / varchar / string | string | | decimal | decimal | | struct | struct | | map | map | | array | array |

2. When Doris exports data to the Parquet file format, the Doris memory data will be converted to Arrow memory data format first, and then the paraquet file format is written by Arrow. The mapping relationship between the Doris data types to the ARROW data types is: | Doris Type | Arrow Type | | — | — | | boolean | boolean | | tinyint | int8 | | smallint | int16 | | int | int32 | | bigint | int64 | | largeInt | utf8 | | date | utf8 | | datev2 | utf8 | | datetime | utf8 | | datetimev2 | utf8 | | float | float32 | | double | float64 | | char / varchar / string | utf8 | | decimal | decimal128 | | struct | struct | | map | map | | array | list |

example

1. Use the broker method to export, and export the simple query results to the file `hdfs://path/to/result.txt`. Specifies that the export format is CSV. Use `my_broker` and set kerberos authentication information. Specify the column separator as `,` and the row separator as `\n`.

```
SELECT * FROM tbl
INTO OUTFILE "hdfs://path/to/result_"
FORMAT AS CSV
PROPERTIES
(
    "broker.name" = "my_broker",
    "broker.hadoop.security.authentication" = "kerberos",
    "broker.kerberos_principal" = "doris@YOUR.COM",
```

```
    "broker.kerberos_keytab" = "/home/doris/my.keytab",
    "column_separator" = ",",
    "line_delimiter" = "\n",
    "max_file_size" = "100MB"
);
```

If the final generated file is not larger than 100MB, it will be: `result_0.csv`. If larger than 100MB, it may be `result_0.csv, result_1.csv, ....`

2. Export the simple query results to the file `hdfs://path/to/result.parquet`. Specify the export format as PARQUET. Use `my_broker` and set kerberos authentication information.

```
SELECT c1, c2, c3 FROM tbl
INTO OUTFILE "hdfs://path/to/result_"
FORMAT AS PARQUET
PROPERTIES
(
    "broker.name" = "my_broker",
    "broker.hadoop.security.authentication" = "kerberos",
    "broker.kerberos_principal" = "doris@YOUR.COM",
    "broker.kerberos_keytab" = "/home/doris/my.keytab"
);
```

3. Export the query result of the CTE statement to the file `hdfs://path/to/result.txt`. The default export format is CSV. Use `my_broker` and set hdfs high availability information. Use the default row and column separators.

```
WITH
x1 AS
(SELECT k1, k2 FROM tbl1),
x2 AS
(SELECT k3 FROM tbl2)
SELEC k1 FROM x1 UNION SELECT k3 FROM x2
INTO OUTFILE "hdfs://path/to/result_"
PROPERTIES
(
    "broker.name" = "my_broker",
    "broker.username"="user",
    "broker.password"="passwd",
    "broker.dfs.nameservices" = "my_ha",
    "broker.dfs.ha.namenodes.my_ha" = "my_namenode1, my_namenode2",
    "broker.dfs.namenode.rpc-address.my_ha.my_namenode1" = "nn1_host:rpc_port",
    "broker.dfs.namenode.rpc-address.my_ha.my_namenode2" = "nn2_host:rpc_port",
    "broker.dfs.client.failover.proxy.provider.my_ha" =
"org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"
);
```

If the final generated file is not larger than 1GB, it will be: `result_0.csv`. If larger than 1GB, it may be `result_0.csv, result_1.csv, ....`

4. Export the query result of the UNION statement to the file `bos://bucket/result.txt`. Specify the export format as PARQUET. Use `my_broker` and set hdfs high availability information. The PARQUET format does not require a column delimiter to be specified. After the export is complete, an identity file is generated.

```
SELECT k1 FROM tbl1 UNION SELECT k2 FROM tbl1
INTO OUTFILE "bos://bucket/result_"
FORMAT AS PARQUET
PROPERTIES
(
    "broker.name" = "my_broker",
    "broker.bos_endpoint" = "http://bj.bcebos.com",
    "broker.bos_accesskey" = "xxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "broker.bos_secret_accesskey" = "yyyyyyyyyyyyyyyyyyyyyyyyy"
);
```

5. Export the query result of the select statement to the file `s3a://${bucket_name}/path/result.txt`. Specify the export format as csv. After the export is complete, an identity file is generated.

```
select k1,k2,v1 from tbl1 limit 100000
into outfile "s3a://my_bucket/export/my_file_"
FORMAT AS CSV
PROPERTIES
(
    "broker.name" = "hdfs_broker",
    "broker.fs.s3a.access.key" = "xxx",
    "broker.fs.s3a.secret.key" = "xxxx",
    "broker.fs.s3a.endpoint" = "https://cos.xxxxxx.myqcloud.com/",
    "column_separator" = ",",
    "line_delimiter" = "\n",
    "max_file_size" = "1024MB",
    "success_file_name" = "SUCCESS"
)
```

   If the final generated file is not larger than 1GB, it will be: `my_file_0.csv`. If larger than 1GB, it may be `my_file_0.csv, result_1.csv, ....` Verify on cos

```
   1. A path that does not exist will be automatically created
   2. Access.key/secret.key/endpoint needs to be confirmed with students of cos. Especially the value of endpoint does not
need to fill in bucket_name.
```

6. Use the s3 protocol to export to bos, and enable concurrent export.

```
set enable_parallel_outfile = true;
select k1 from tb1 limit 1000
into outfile "s3://my_bucket/export/my_file_"
format as csv
properties
(
     "s3.endpoint" = "http://s3.bd.bcebos.com",
     "s3.access_key" = "xxxx",
     "s3.secret_key" = "xxx",
     "s3.region" = "bd"
)
```

   The resulting file is prefixed with `my_file_{fragment_instance_id}_`.

7. Use the s3 protocol to export to bos, and enable concurrent export of session variables. Note: However, since the query statement has a top-level sorting node, even if the concurrently exported session variable is enabled for this query, it cannot be exported concurrently.

```
set enable_parallel_outfile = true;
select k1 from tb1 order by k1 limit 1000
into outfile "s3://my_bucket/export/my_file_"
format as csv
properties
(
     "s3.endpoint" = "http://s3.bd.bcebos.com",
     "s3.access_key" = "xxxx",
     "s3.secret_key" = "xxx",
     "s3.region" = "bd"
)
```

8. Use hdfs export to export simple query results to the file `hdfs://${host}:${fileSystem_port}/path/to/result.txt`. Specify the export format as CSV and the user name as work. Specify the column separator as ，and the row separator as `\n`.

```
-- the default port of fileSystem_port is 9000
SELECT * FROM tbl
INTO OUTFILE "hdfs://${host}:${fileSystem_port}/path/to/result_"
FORMAT AS CSV
PROPERTIES
(
    "fs.defaultFS" = "hdfs://ip:port",
    "hadoop.username" = "work"
);
```

If the Hadoop cluster is highly available and Kerberos authentication is enabled, you can refer to the following SQL statement:

```sql SELECT * FROM tbl INTO OUTFILE “hdfs://path/to/result_” FORMAT AS CSV PROPERTIES ( ‘fs.defaultFS’=‘hdfs://hacluster/’, ‘dfs.nameservices’=‘hacluster’, ‘dfs.ha.namenodes.hacluster’=‘n1,n2’, ‘dfs.namenode.rpc-address.hacluster.n1’=‘192.168.0.1:8020’, ‘dfs.namenode.rpc-address.hacluster.n2’=‘192.168.0.2:8020’, ‘dfs.client.failover.proxy.provider.hacluster’=‘org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider’, ‘dfs.namenode.kerberos.principal’=’hadoop/_HOST@REALM.COM’ ‘hadoop.security.authentication’=‘kerberos’, ‘hadoop.kerberos.principal’=‘doris_test@REALM.COM’, ‘hadoop.kerberos.keytab’=‘/path/to/doris_test.keytab’ );

If the final generated file is not larger than 100MB, it will be: `result_0.csv`. If larger than 100MB, it may be `result_0.csv, result_1.csv, ....`

9. Export the query result of the select statement to the file `cosn://${bucket_name}/path/result.txt` on Tencent Cloud Object Storage (COS). Specify the export format as csv. After the export is complete, an identity file is generated.

```
select k1,k2,v1 from tbl1 limit 100000
into outfile "cosn://my_bucket/export/my_file_"
FORMAT AS CSV
PROPERTIES
(
    "broker.name" = "broker_name",
    "broker.fs.cosn.userinfo.secretId" = "xxx",
    "broker.fs.cosn.userinfo.secretKey" = "xxxx",
    "broker.fs.cosn.bucket.endpoint_suffix" = "cos.xxxxxx.myqcloud.com",
    "column_separator" = ",",
```
```

```
    "line_delimiter" = "\n",
    "max_file_size" = "1024MB",
    "success_file_name" = "SUCCESS"
)
```

keywords

OUTFILE

Best Practice

1. Export data volume and export efficiency

   This function essentially executes an SQL query command. The final result is a single-threaded output. Therefore, the time-consuming of the entire export includes the time-consuming of the query itself and the time-consuming of writing the final result set. If the query is large, you need to set the session variable `query_timeout` to appropriately extend the query timeout.

2. Management of export files

   Doris does not manage exported files. Including the successful export, or the remaining files after the export fails, all need to be handled by the user.

3. Export to local file

   The ability to export to a local file is not available for public cloud users, only for private deployments. And the default user has full control over the cluster nodes. Doris will not check the validity of the export path filled in by the user. If the process user of Doris does not have write permission to the path, or the path does not exist, an error will be reported. At the same time, for security reasons, if a file with the same name already exists in this path, the export will also fail.

   Doris does not manage files exported locally, nor does it check disk space, etc. These files need to be managed by the user, such as cleaning and so on.

4. Results Integrity Guarantee

   This command is a synchronous command, so it is possible that the task connection is disconnected during the execution process, so that it is impossible to live the exported data whether it ends normally, or whether it is complete. At this point, you can use the `success_file_name` parameter to request that a successful file identifier be generated in the directory after the task is successful. Users can use this file to determine whether the export ends normally.

5. Other Points to Note

   See Export Query Result

SHOW ALTER TABLE MATERIALIZED VIEW

Description

This command is used to check the execution status of the materialized view creation job submitted via statement CREATE-MATERIALIZED-VIEW

> This statement is equivalent to `SHOW ALTER TABLE ROLLUP;`

Syntax

```
SHOW ALTER TABLE MATERIALIZED VIEW
[FROM <database>]
[<where_clause>]
[ORDER BY <order_by_key> [, ...]]
[LIMIT <limit_rows> [ OFFSET <offset_rows>]]
```

Optional Parameters

1. `FROM <database>`

> View jobs under the specified database. If not specified, the current database will be used.

2. `<where_clause>`

> You can filter the result columns, and currently, only the following columns are supported for filtering: - TableName: Only supports equality filtering. - State: Only supports equality filtering. - Createtime/FinishTime: Supports =, >=, <=, >, <, !=.

3. `ORDER BY`

> You can sort the result set by any column.

4. `LIMIT <limit_rows> [ OFFSET <offset_rows>]`

> For pagination queries.

Return Value

Column	Note
JobId	Job unique ID.
TableName	Base table name.
CreateTime	Job creation time.
FinishTime	Job finish time.
BaseIndexName	Base table name.
RollupIndexName	Materialized view name.
RollupId	The unique ID of the materialized view.
TransactionId	See the description of the State field
State	Job status.
Msg	Error msg
Progress	Progress of the assignment. Here, progress refers to the “number of completed tablets / total number of tablets”. The creation of materialized views is performed on a tablet granularity basis.
Timeout	Job timeout duration, measured in seconds.

State Description: - PENDING: The job is in preparation.

- WAITING_TXN: Before officially starting to generate materialized view data, the system will wait for the ongoing import transactions on the current table to complete. The TransactionId field indicates the ID of the transaction currently being waited for. The job will actually start once all imports with transaction IDs prior to this one have been completed.
- RUNNING: The job is currently in progress.
- FINISHED: The job has been successfully completed.
- CANCELLED: The job failed to run.

Access Control Requirements

The user who executes this SQL must have at least the following permissions:

Privilege	Object	Notes
ALTER_PRIV	table	

Examples

1. View materialized view jobs under the database example_db.

```
SHOW ALTER TABLE MATERIALIZED VIEW FROM example_db;
```

SHOW-ALTER

SHOW-ALTER

Name

SHOW ALTER

Description

This statement is used to display the execution of various modification tasks currently in progress

```
SHOW ALTER [CLUSTER | TABLE [COLUMN | ROLLUP] [FROM db_name]];
```

TABLE COLUMN: show ALTER tasks that modify columns Support syntax [WHERE TableName|CreateTime|FinishTime|State] [ORDER BY] [LIMIT] TABLE ROLLUP: Shows the task of creating or deleting a ROLLUP index If db_name is not specified, the current default db is used CLUSTER: Displays tasks related to cluster operations (only for administrators! To be implemented…)

Example

1. Display the task execution of all modified columns of the default db

   ```
   SHOW ALTER TABLE COLUMN;
   ```

2. Display the task execution status of the last modified column of a table

   ```
   SHOW ALTER TABLE COLUMN WHERE TableName = "table1" ORDER BY CreateTime DESC LIMIT 1;
   ```

3. Display the task execution of creating or deleting ROLLUP index for the specified db

   ```
   SHOW ALTER TABLE ROLLUP FROM example_db;
   ```

4. Show tasks related to cluster operations (only for administrators! To be implemented…)

   ```
   SHOW ALTER CLUSTER;
   ```

Keywords

```
SHOW, ALTER
```

Best Practice

SHOW-ANALYZE

SHOW-ANALYZE

Name

SHOW ANALYZE

Description

Use `SHOW ANALYZE` to view information about statistics collection jobs.

Syntax:

```
SHOW [AUTO] ANALYZE < table_name | job_id >
    [ WHERE [ STATE = [ "PENDING" | "RUNNING" | "FINISHED" | "FAILED" ] ] ];
```

- AUTO: Show historical information for automatic collection jobs only. Note that, by default, the status of only the last 20,000 completed automatic collection jobs is retained.
- table_name: Table name, specify to view statistics job information for that table. It can be in the format `db_name.table_name`. When not specified, it returns information for all statistics jobs.
- job_id: Job ID for statistics collection, obtained when executing `ANALYZE`. When not specified, this command returns information for all statistics jobs.

Output:

Column Name	Description
`job_id`	Job ID
`catalog_name`	Catalog Name
`db_name`	Database Name
`tbl_name`	Table Name
`col_name`	Column Name List
`job_type`	Job Type
`analysis_type`	Analysis Type
`message`	Job Information
`last_exec_time_in_ms`	Last Execution Time
`state`	Job Status
`schedule_type`	Scheduling Method

Here's an example:

```
mysql> show analyze 245073\G;
*************************** 1. row ***************************
              job_id: 245073
        catalog_name: internal
             db_name: default_cluster:tpch
            tbl_name: lineitem
            col_name:
[l_returnflag,l_receiptdate,l_tax,l_shipmode,l_suppkey,l_shipdate,l_commitdate,l_partkey,l_orderkey,l_quantity,l_linestatus,l_commen
t,l_extendedprice,l_linenumber,l_discount,l_shipinstruct]
            job_type: MANUAL
       analysis_type: FUNDAMENTALS
             message:
last_exec_time_in_ms: 2023-11-07 11:00:52
               state: FINISHED
            progress: 16 Finished  |  0 Failed  |  0 In Progress  |  16 Total
       schedule_type: ONCE
```

Each collection job can contain one or more tasks, with each task corresponding to the collection of a column. Users can use the following command to view the completion status of statistics collection for each column.

Syntax:

```
SHOW ANALYZE TASK STATUS [job_id]
```

Here's an example:

```
mysql> show analyze task status 20038 ;
+---------+----------+---------+----------------------+----------+
| task_id | col_name | message | last_exec_time_in_ms | state    |
+---------+----------+---------+----------------------+----------+
| 20039   | col4     |         | 2023-06-01 17:22:15  | FINISHED |
| 20040   | col2     |         | 2023-06-01 17:22:15  | FINISHED |
| 20041   | col3     |         | 2023-06-01 17:22:15  | FINISHED |
| 20042   | col1     |         | 2023-06-01 17:22:15  | FINISHED |
+---------+----------+---------+----------------------+----------+
```

Keywords

SHOW, ANALYZE

SHOW-BACKUP

SHOW-BACKUP

Name

SHOW BACKUP

Description

This statement is used to view BACKUP tasks

grammar:

```
SHOW BACKUP [FROM db_name]
```

illustrate:

1. Only the most recent BACKUP task is saved in Doris.
2. The meaning of each column is as follows: JobId: Unique job id SnapshotName: The name of the backup DbName: belongs to the database State: current stage PENDING: The initial state after submitting the job SNAPSHOTING: Executing snapshot UPLOAD_SNAPSHOT: Snapshot completed, ready to upload UPLOADING: Snapshot uploading SAVE_META: Save job meta information to a local file UPLOAD_INFO: Upload job meta information FINISHED: The job was successful CANCELLED: Job failed BackupObjs: Backed up tables and partitions CreateTime: task submission time SnapshotFinishedTime: Snapshot completion time UploadFinishedTime: Snapshot upload completion time FinishedTime: Job finish time UnfinishedTasks: Displays unfinished subtask ids during SNAPSHOTING and UPLOADING stages Status: If the job fails, display the failure message Timeout: Job timeout, in seconds

Example

1. View the last BACKUP task under example_db.

   ```
   SHOW BACKUP FROM example_db;
   ```

Keywords

```
SHOW, BACKUP
```

Best Practice

SHOW-BACKENDS

SHOW-BACKENDS

Name

SHOW BACKENDS

Description

This statement is used to view the BE nodes in the cluster

```
SHOW BACKENDS;
```

illustrate:

```
    1. LastStartTime indicates the last BE start time.
    2. LastHeartbeat indicates the last heartbeat.
    3. Alive indicates whether the node is alive or not.
    4. If SystemDecommissioned is true, it means that the node is being safely decommissioned.
    5. If ClusterDecommissioned is true, it means that the node is going offline in the current cluster.
    6. TabletNum represents the number of shards on the node.
    7. DataUsedCapacity Indicates the space occupied by the actual user data.
    8. AvailCapacity Indicates the available space on the disk.
    9. TotalCapacity represents the total disk space. TotalCapacity = AvailCapacity + DataUsedCapacity + other non-user data files
occupy space.
   10. UsedPct Indicates the percentage of disk used.
   11. ErrMsg is used to display the error message when the heartbeat fails.
   12. Status is used to display some status information of BE in JSON format, including the time information of the last time BE
reported its tablet.
   13. HeartbeatFailureCounter: The current number of heartbeats that have failed consecutively. If the number exceeds the
`max_backend_heartbeat_failure_tolerance_count` configuration, the isAlive will be set to false.
   14. NodeRole is used to display the role of Backend node. Now there are two roles: mix and computation. Mix node represent the
origin Backend node and computation Node represent the compute only node.
```

Example

Keywords

```
SHOW, BACKENDS
```

Best Practice

SHOW-BROKER

SHOW-BROKER

Name

SHOW BROKER

Description

This statement is used to view the currently existing broker

grammar:

```
SHOW BROKER;
```

illustrate:

1. LastStartTime indicates the last BE start time.
2. LastHeartbeat indicates the last heartbeat.
3. Alive indicates whether the node is alive or not.
4. ErrMsg is used to display the error message when the heartbeat fails.

Example

Keywords

```
SHOW, BROKER
```

Best Practice

SHOW-CATALOGS

SHOW-CATALOGS

Name

SHOW CATALOGS

Description

This statement is used for view created catalogs

Syntax:

```
SHOW CATALOGS [LIKE]
```

illustrate:

1. LIKE: Fuzzy query can be performed according to the catalog name

Return result:

- CatalogId: Unique ID of the catalog
- CatalogName: Catalog name. where “internal” is the default built-in catalog, which cannot be modified.
- Type: Catalog type.
- IsCurrent: Show yes on the line of current using catalog.

Example

1. View the data catalogs that have been created currently

```
SHOW CATALOGS;
 +-----------+-------------+----------+-----------+
 | CatalogId | CatalogName | Type     | IsCurrent |
 +-----------+-------------+----------+-----------+
 |    130100 | hive        | hms      |           |
 |         0 | internal    | internal | yes       |
 +-----------+-------------+----------+-----------+
```

2. Fuzzy query by catalog name

sql SHOW CATALOGS LIKE 'hi%'; +-----------+-------------+----------+-----------+ | CatalogId | CatalogName | Type | IsCurrent | +-----------+-------------+----------+-----------+ | 130100 | hive | hms | | +-----------+-------------+----------+-----------+

Keywords

SHOW, CATALOG, CATALOGS

Best Practice

SHOW-CREATE-TABLE

SHOW-CREATE-TABLE

Name

SHOW CREATE TABLE

Description

This statement is used to display the creation statement of the data table.

grammar:

```
SHOW [BRIEF] CREATE TABLE [DBNAME.]TABLE_NAME
```

illustrate:

1. `BRIEF` : will not show partitions info
2. `DBNAMNE` : database name
3. `TABLE_NAME` : table name

Example

1. View the table creation statement of a table

   ```
   SHOW CREATE TABLE demo.tb1
   ```

Keywords

```
SHOW, CREATE, TABLE
```

Best Practice

SHOW-CHARSET

SHOW-CHARSET

Description

Example

Keywords

```
SHOW, CHARSET
```

Best Practice

SHOW-CREATE-CATALOG

SHOW-CREATE-CATALOG

Name

> **Tips**
>
> This feature is supported since the Apache Doris 1.2 version

SHOW CREATE CATALOG

Description

This statement shows the creating statement of a doris catalog.

grammar:

```
SHOW CREATE CATALOG catalog_name;
```

illustrate: - `catalog_name`: The name of the catalog which exist in doris.

Example

1. View the creating statement of the hive catalog in doris

   ```
   SHOW CREATE CATALOG hive;
   ```

Keywords

```
SHOW, CREATE, CATALOG
```

Best Practice

SHOW-CREATE-DATABASE

SHOW-CREATE-DATABASE

Name

SHOW CREATE DATABASE

Description

This statement checks the creation of the doris database.

grammar:

```
SHOW CREATE DATABASE db_name;
```

illustrate:

- `db_name`: The name of the database where doris exists.

Example

1. View the creation of the test database in doris

```
mysql> SHOW CREATE DATABASE test;
+----------+----------------------------+
| Database | Create Database |
+----------+----------------------------+
| test | CREATE DATABASE `test` |
+----------+----------------------------+
1 row in set (0.00 sec)
```

Keywords

```
SHOW, CREATE, DATABASE
```

Best Practice

SHOW-CREATE-MATERIALIZED-VIEW

SHOW-CREATE-MATERIALIZED-VIEW

Name

SHOW CREATE MATERIALIZED VIEW

Description

This statement is used to query statements that create materialized views.

grammar：

```
SHOW CREATE MATERIALIZED VIEW mv_name ON table_name
```

1. mv_name: Materialized view name. required.
2. table_name: The table name of materialized view. required.

Example

Create materialized view

```
create materialized view id_col1 as select id,col1 from table3;
```

Return after query

```
mysql> show create materialized view id_col1 on table3;
+-----------+----------+-----------------------------------------------------------------+
| TableName | ViewName | CreateStmt                                                      |
+-----------+----------+-----------------------------------------------------------------+
| table3    | id_col1  | create materialized view id_col1 as select id,col1 from table3 |
+-----------+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

Keywords

```
SHOW, MATERIALIZED, VIEW
```

Best Practice

SHOW-CREATE-LOAD

SHOW-CREATE-LOAD

Name

SHOW CREATE LOAD

Description

This statement is used to demonstrate the creation statement of a import job.

grammar:

```
SHOW CREATE LOAD for load_name;
```

illustrate:

- `load_name`: import job name

Example

1. Show the creation statement of the specified import job under the default db

   ```
   SHOW CREATE LOAD for test_load
   ```

Keywords

```
SHOW, CREATE, LOAD
```

Best Practice

SHOW-CREATE-REPOSITORY

SHOW-CREATE-REPOSITORY

Name

SHOW CREATE REPOSITORY

Description

This statement is used to demonstrate the creation statement of a repository.

grammar:

```
SHOW CREATE REPOSITORY for repository_name;
```

illustrate: - `repository_name`: repository name

Example

1. Show the creation statement of the specified repository

   ```
   SHOW CREATE REPOSITORY for test_repository
   ```

Keywords

```
SHOW, CREATE, REPOSITORY
```

Best Practice

SHOW-CREATE-ROUTINE-LOAD

SHOW-CREATE-ROUTINE-LOAD

Name

SHOW CREATE ROUTINE LOAD

Description

This statement is used to demonstrate the creation statement of a routine import job.

The kafka partition and offset in the result show the currently consumed partition and the corresponding offset to be consumed.

grammar:

```
SHOW [ALL] CREATE ROUTINE LOAD for load_name;
```

illustrate:

1. `ALL`: optional parameter, which means to get all jobs, including historical jobs
2. `load_name`: routine import job name

Example

1. Show the creation statement of the specified routine import job under the default db

   ```
   SHOW CREATE ROUTINE LOAD for test_load
   ```

Keywords

```
SHOW, CREATE, ROUTINE, LOAD
```

Best Practice

SHOW-CREATE-FUNCTION

Description

This statement is used to show the creation statement of a user-defined function

Syntax

```
SHOW CREATE [ GLOBAL ] FUNCTION <function_name>( <arg_type> ) [ FROM <db_name> ];
```

Required Parameters

1. `<function_name>`

> The name of the custom function that you want to query for the creation statement.

2. `<arg_type>`

> The parameter list of the custom function that needs to be queried for the creation statement.
>
> Parameter list location you need to enter the data type of the location parameter

Optional Parameters

1.`GLOBAL`

> GLOBAL is an optional parameter.
>
> If GLOBAL is set, the function is searched for globally and deleted.
>
> If GLOABL is not entered, the function is searched for in the current database and deleted.

2.`<db_name>`

> FROM db_name indicates that the custom function is queried from the specified database

Return Value

Column	Description
SYMBOL	Function package name
FILE	jar package path
ALWAYS_NULLABLE	Whether the result can be NULL
TYPE	Function type

Access Control Requirements

The user who executes this SQL command must have at least the following permissions:

Privilege	Object	Notes
SHOW_PRIV	Function	You need to have the show permission on this function

Examples

```
SHOW CREATE FUNCTION add_one(INT)
```

```
| Function Signature | Create Function
+--------------------+-------------------------------------------------------
| add_one(INT)       | CREATE FUNCTION add_one(INT) RETURNS INT PROPERTIES (
  "SYMBOL"="org.apache.doris.udf.AddOne",
  "FILE"="file:///xxx.jar",
  "ALWAYS_NULLABLE"="true",
  "TYPE"="JAVA_UDF"
  ); |
+--------------------+-------------------------------------------------------
```

SHOW-COLUMNS

SHOW-COLUMNS

Name

SHOW FULL COLUMNS

Description

This statement is used to specify the column information of the table

grammar:

```
SHOW [FULL] COLUMNS FROM tbl;
```

Example

1. View the column information of the specified table

   ```
   SHOW FULL COLUMNS FROM tbl;
   ```

Keywords

```
SHOW, COLUMNS
```

Best Practice

SHOW-COLUMN-STATS

SHOW-COLUMN-STATS

Name

SHOW COLUMN STATS

Description

Use `SHOW COLUMN STATS` to view various statistics data for columns.

Syntax:

```
SHOW COLUMN [cached] STATS table_name [ (column_name [, ...]) ];
```

Where:

- cached: Show statistics information in the current FE memory cache.
- table_name: The target table for collecting statistics. It can be in the format `db_name.table_name`.
- column_name: Specifies the target column, which must be an existing column in `table_name`. You can specify multiple column names separated by commas.

Here's an example:

```
mysql> show column stats lineitem(l_tax)\G;
*************************** 1. row ***************************
  column_name: l_tax
        count: 6001215.0
          ndv: 9.0
     num_null: 0.0
    data_size: 4.800972E7
avg_size_byte: 8.0
          min: 0.00
          max: 0.08
       method: FULL
         type: FUNDAMENTALS
      trigger: MANUAL
  query_times: 0
 updated_time: 2023-11-07 11:00:46
```

Keywords

SHOW, TABLE, STATS

SHOW-COLLATION

Description

In Doris, the `SHOW COLLATION` command is used to display the character set collations available in the database. A collation is a set of rules that determine how data is sorted and compared. These rules affect the storage and retrieval of character data. The content returned by this command is only for compatibility with MySQL's behavior. It does not represent the list of character set collations that Doris actually supports. Doris currently mainly supports the proofreading method utf8mb4_0900_bin.

Syntax

```
SHOW COLLATION
```

Result Fields

The `SHOW COLLATION` command returns the following fields:

- Collation: The collation name
- Charset: The character set
- Id: The collation's ID
- Default: Whether this is the default collation for the character set
- Compiled: Whether the collation is compiled
- Sortlen: Sort length

Example

```
show collation;
```

```
+-----------------+---------+------+---------+----------+---------+
| Collation       | Charset | Id   | Default | Compiled | Sortlen |
+-----------------+---------+------+---------+----------+---------+
| utf8_general_ci | utf8    |   33 | Yes     | Yes      |       1 |
+-----------------+---------+------+---------+----------+---------+
```

Notice

In Doris, although it is compatible with MySQL's commands for setting the collation, the setting actually does not take effect. When executed, Doris will always use utf8mb4_0900_bin as the comparison rule.

SHOW-DATABASES

SHOW-DATABASES

Name

SHOW DATABASES

Description

This statement is used to display the currently visible db

grammar:

```
SHOW DATABASES [FROM catalog] [filter expr];
```

illustrate: 1. `SHOW DATABASES` will get all database names from current catalog. 2. `SHOW DATABASES FROM catalog` will all database names from the catalog named 'catalog'. 3. `SHOW DATABASES filter_expr` will get filtered database names from current catalog. 4. `SHOW DATABASES FROM catalog filter_expr` is not support yet.

Example

1. Display all the database names from current catalog.

   ```
   SHOW DATABASES;
   ```

   ```
   +--------------------+
   | Database           |
   +--------------------+
   | test               |
   | information_schema |
   +--------------------+
   ```

2. Display all database names from the catalog named 'hms_catalog'.

   ```
   SHOW DATABASES from hms_catalog;
   ```

   ```
   +---------------+
   | Database      |
   +---------------+
   | default       |
   | tpch          |
   +---------------+
   ```

3. Display the filtered database names from current catalog with the expr 'like'.

```
SHOW DATABASES like 'infor%';
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
```

Keywords

```
SHOW, DATABASES
```

Best Practice

SHOW DATA SKEW

SHOW-DATA-SKEW

Name

SHOW DATA SKEW

Description

```
This statement is used to view the data skew of a table or a partition.

grammar:

    SHOW DATA SKEW FROM [db_name.]tbl_name [PARTITION (p1)];

Description:

    1. Only one partition must be specified. For non-partitioned tables, the partition name is the same as the table name.
    2. The result will show row count and data volume of each bucket under the specified partition, and the proportion of the data
volume of each bucket in the total data volume.
```

Example

```
1. View the data skew of the table

    SHOW DATA SKEW FROM db1.test PARTITION(p1);
```

Keywords

```
SHOW, DATA, SKEW
```

Best Practice

SHOW-DATABASE-ID

SHOW-DATABASE-ID

Name

SHOW DATABASE ID

Description

This statement is used to find the corresponding database name according to the database id (only for administrators)

grammar:

```
SHOW DATABASE [database_id]
```

Example

1. Find the corresponding database name according to the database id

   ```
   SHOW DATABASE 1001;
   ```

Keywords

```
SHOW, DATABASE, ID
```

Best Practice

SHOW-DYNAMIC-PARTITION

SHOW-DYNAMIC-PARTITION

Name

SHOW DYNAMIC

Description

This statement is used to display the status of all dynamic partition tables under the current db

grammar:

```
SHOW DYNAMIC PARTITION TABLES [FROM db_name];
```

Example

1. Display all dynamic partition table status of database database

    ```
    SHOW DYNAMIC PARTITION TABLES FROM database;
    ```

Keywords

```
SHOW, DYNAMIC, PARTITION
```

Best Practice

SHOW-DELETE

SHOW-DELETE

Name

SHOW DELETE

Description

This statement is used to display the historical delete tasks that have been successfully executed

grammar:

```
SHOW DELETE [FROM db_name]
```

Example

1. Display all historical delete tasks of database database

   ```
   SHOW DELETE FROM database;
   ```

Keywords

```
SHOW, DELETE
```

Best Practice

SHOW-DATA

SHOW-DATA

Name

SHOW DATA

Description

This statement is used to display the amount of data, the number of replicas, and the number of statistical rows.

grammar:

```
SHOW DATA [FROM [db_name.]table_name] [ORDER BY ...];
```

illustrate:

1. If the FROM clause is not specified, the data volume and number of replicas subdivided into each table under the current db will be displayed. The data volume is the total data volume of all replicas. The number of replicas is the number of replicas for all partitions of the table and all materialized views.
2. If the FROM clause is specified, the data volume, number of copies and number of statistical rows subdivided into each materialized view under the table will be displayed. The data volume is the total data volume of all replicas. The number of replicas is the number of replicas for all partitions of the corresponding materialized view. The number of statistical rows is the number of statistical rows for all partitions of the corresponding materialized view.
3. When counting the number of rows, the one with the largest number of rows among the multiple copies shall prevail.
4. The `Total` row in the result set represents the total row. The `Quota` line represents the quota set by the current database. The `Left` line indicates the remaining quota.
5. If you want to see the size of each Partition, see `help show partitions`.
6. You can use ORDER BY to sort on any combination of columns.

Example

1. Display the data volume, replica number, aggregate data volume and aggregate replica number of each table in the default db.

   ```
   SHOW DATA;
   ```

```
+-----------+-------------+--------------+
| TableName | Size        | ReplicaCount |
+-----------+-------------+--------------+
| tbl1      | 900.000 B   | 6            |
| tbl2      | 500.000 B   | 3            |
| Total     | 1.400 KB    | 9            |
| Quota     | 1024.000 GB | 1073741824   |
| Left      | 1021.921 GB | 1073741815   |
+-----------+-------------+--------------+
```

2. Display the subdivided data volume, the number of replicas and the number of statistical rows of the specified table under the specified db

```
SHOW DATA FROM example_db.test;
```

```
+-----------+-----------+-----------+--------------+----------+
| TableName | IndexName | Size      | ReplicaCount | RowCount |
+-----------+-----------+-----------+--------------+----------+
| test      | r1        | 10.000MB  | 30           | 10000    |
|           | r2        | 20.000MB  | 30           | 20000    |
|           | test2     | 50.000MB  | 30           | 50000    |
|           | Total     | 80.000    | 90           |          |
+-----------+-----------+-----------+--------------+----------+
```

3. It can be combined and sorted according to the amount of data, the number of copies, the number of statistical rows, etc.

```
SHOW DATA ORDER BY ReplicaCount desc,Size asc;
```

```
+-----------+-------------+--------------+
| TableName | Size        | ReplicaCount |
+-----------+-------------+--------------+
| table_c   | 3.102 KB    | 40           |
| table_d   | .000        | 20           |
| table_b   | 324.000 B   | 20           |
| table_a   | 1.266 KB    | 10           |
| Total     | 4.684 KB    | 90           |
| Quota     | 1024.000 GB | 1073741824   |
| Left      | 1024.000 GB | 1073741734   |
+-----------+-------------+--------------+
```

Keywords

```
SHOW, DATA
```

Best Practice

SHOW-ENGINES

SHOW-ENGINES

Description

Example

Keywords

```
SHOW, ENGINES
```

Best Practice

SHOW-EVENTS

SHOW-EVENTS

Description

Example

Keywords

```
SHOW, EVENTS
```

Best Practice

SHOW-EXPORT

SHOW-EXPORT

Name

SHOW EXPORT

Description

This statement is used to display the execution of the specified export task

grammar:

```
SHOW EXPORT
[FROM db_name]
  [
    WHERE
      [ID=your_job_id]
      [STATE = ["PENDING"|"EXPORTING"|"FINISHED"|"CANCELLED"]]
      [LABEL=your_label]
   ]
[ORDER BY...]
[LIMIT limit];
```

illustrate: 1. If db_name is not specified, the current default db is used 2. If STATE is specified, matches EXPORT state 3. You can use ORDER BY to sort any combination of columns 4. If LIMIT is specified, limit matching records are displayed. Otherwise show all

Example

1. Show all export tasks of default db

   ```
   SHOW EXPORT;
   ```

2. Display the export tasks of the specified db, sorted by StartTime in descending order

   ```
    SHOW EXPORT FROM example_db ORDER BY StartTime DESC;
   ```

3. Display the export tasks of the specified db, the state is “exporting”, and sort by StartTime in descending order

   ```
   SHOW EXPORT FROM example_db WHERE STATE = "exporting" ORDER BY StartTime DESC;
   ```

4. Display the export task of the specified db and specified job_id

```
SHOW EXPORT FROM example_db WHERE ID = job_id;
```

5. Display the specified db and specify the export task of the label

```
SHOW EXPORT FROM example_db WHERE LABEL = "mylabel";
```

Keywords

```
SHOW, EXPORT
```

Best Practice

SHOW-ENCRYPT-KEY

SHOW-ENCRYPT-KEY

Name

SHOW ENCRYPTKEYS

Description

View all custom keys under the database. If the user specifies a database, check the corresponding database, otherwise directly query the database where the current session is located.

Requires `ADMIN` privilege on this database

grammar:

```
SHOW ENCRYPTKEYS [IN|FROM db] [LIKE 'key_pattern']
```

parameter

> `db`: database name to query `key_pattern`: parameter used to filter key names

Example

```
mysql> SHOW ENCRYPTKEYS;
+------------------+------------------+
| EncryptKey Name   | EncryptKey String |
+------------------+------------------+
| example_db.my_key | ABCD123456789     |
+------------------+------------------+
1 row in set (0.00 sec)

mysql> SHOW ENCRYPTKEYS FROM example_db LIKE "%my%";
+------------------+------------------+
| EncryptKey Name   | EncryptKey String |
+------------------+------------------+
| example_db.my_key | ABCD123456789     |
+------------------+------------------+
1 row in set (0.00 sec)
```

Keywords

```
SHOW, ENCRYPT, KEY
```

Best Practice

SHOW-FUNCTIONS

Description

View all custom and system provided functions under the database.

Syntax

```
SHOW [ FULL ] [ BUILTIN ] FUNCTIONS [ { IN | FROM } <db> ]  [ LIKE '<function_pattern>' ]
```

Varaint Syntax

```
SHOW GLOBAL [ FULL ] FUNCTIONS [ LIKE '<function_pattern>' ]
```

Required Parameters

1. `<function_pattern>`

> Matching pattern rules used to filter function names

Optional Parameters

1. `FULL`

> FULL is an optional parameter.
>
> This parameter indicates the detailed information about the function.

2. `BUILTIN`

> BUILTIN is an optional parameter.
>
> This parameter indicates that the functions provided by the system need to be displayed

3. `<db>`

> db is an optional parameter.

> This parameter indicates the query under the specified database

Return Value

Column	Description
Signature	Function name and parameter type
Return Type	The data type of the value returned by the function
Function Type	Type of function
Intermediate Type	Intermediate result type
Properties	Detailed properties of a function

Access Control Requirements

The user who executes this SQL command must have at least the following permissions:

Privilege	Object	Notes
SHOW_PRIV	Function	You need to have the show permission on this function

Examples

```
show full functions in testDb
```

```
*************************** 1. row ***************************
Signature: my_add(INT,INT)
Return Type: INT
Function Type: Scalar
Intermediate Type: NULL
Properties:
{"symbol":"_ZN9doris_udf6AddUdfEPNS_15FunctionContextERKNS_6IntValES4_","object_file":"http://host:port/libudfsample.so","md5":"cfe7
a362d10f3aaf6c49974ee0f1f878"}
*************************** 2. row ***************************
Signature: my_count(BIGINT)
Return Type: BIGINT
Function Type: Aggregate
Intermediate Type: NULL
Properties:
{"object_file":"http://host:port/libudasample.so","finalize_fn":"_ZN9doris_udf13CountFinalizeEPNS_15FunctionContextERKNS_9BigIntValE
","init_fn":"_ZN9doris_udf9CountInitEPNS_15FunctionContextEPNS_9BigIntValE","merge_fn":"_ZN9doris_udf10CountMergeEPNS_15FunctionCont
extERKNS_9BigIntValEPS2_","md5":"37d185f80f95569e2676da3d5b5b9d2f","update_fn":"_ZN9doris_udf11CountUpdateEPNS_15FunctionContextERKN
S_6IntValEPNS_9BigIntValE"}
*************************** 3. row ***************************
Signature: id_masking(BIGINT)
Return Type: VARCHAR
Function Type: Alias
Intermediate Type: NULL
Properties: {"parameter":"id","origin_function":"concat(left(`id`, 3), `****`, right(`id`, 4))"}
```

```
show builtin functions in testDb like 'year%';
```

```
+---------------+
| Function Name |
+---------------+
| year          |
| years_add     |
| years_diff    |
| years_sub     |
+---------------+
```

```
show global full functions
```

```
*************************** 1. row ***************************
        Signature: decimal(ALL, INT, INT)
      Return Type: VARCHAR
    Function Type: Alias
Intermediate Type: NULL
       Properties: {"parameter":"col, precision, scale","origin_function":"CAST(`col` AS decimal(`precision`, `scale`))"}
*************************** 2. row ***************************
        Signature: id_masking(BIGINT)
      Return Type: VARCHAR
    Function Type: Alias
Intermediate Type: NULL
       Properties: {"parameter":"id","origin_function":"concat(left(`id`, 3), `****`, right(`id`, 4))"}
```

```
show global functions
```

```
+---------------+
| Function Name |
+---------------+
| decimal       |
| id_masking    |
+---------------+
```

SHOW-TYPECAST

SHOW-TYPECAST

Name

SHOW TYPECAST

Description

View all type cast under the database. If the user specifies a database, then view the corresponding database, otherwise directly query the database where the current session is located

Requires `SHOW` permission on this database

grammar

```
SHOW TYPE_CAST [IN|FROM db]
```

Parameters

> `db`: database name to query

Example

```
mysql> show type_cast in testDb\G
*************************** 1. row ******************** ******
Origin Type: TIMEV2
  Cast Type: TIMEV2
*************************** 2. row ******************** ******
Origin Type: TIMEV2
  Cast Type: TIMEV2
*************************** 3. row ******************** ******
Origin Type: TIMEV2
  Cast Type: TIMEV2

3 rows in set (0.00 sec)
```

Keywords

```
SHOW, TYPECAST
```

Best Practice

SHOW-FILE

SHOW-FILE

Name

SHOW FILE

Description

This statement is used to display a file created in a database

grammar:

```
SHOW FILE [FROM database];
```

illustrate:

```
FileId: file ID, globally unique
DbName: the name of the database to which it belongs
Catalog: Custom Category
FileName: file name
FileSize: file size, in bytes
MD5: MD5 of the file
```

Example

1. View the uploaded files in the database my_database

   ```
   SHOW FILE FROM my_database;
   ```

Keywords

```
SHOW, FILE
```

Best Practice

SHOW-GRANTS

SHOW-GRANTS

Name

SHOW GRANTS

Description

This statement is used to view user permissions.

grammar:

```
SHOW [ALL] GRANTS [FOR user_identity];
```

illustrate:

1. SHOW ALL GRANTS can view the permissions of all users.
2. If user_identity is specified, view the permissions of the specified user. And the user_identity must be created by the CREATE USER command.
3. If user_identity is not specified, view the permissions of the current user.

Example

1. View all user permission information

   ```
   SHOW ALL GRANTS;
   ```

2. View the permissions of the specified user

   ```
   SHOW GRANTS FOR jack@'%';
   ```

3. View the permissions of the current user

   ```
   SHOW GRANTS;
   ```

Keywords

```
SHOW, GRANTS
```

Best Practice

SHOW-LAST-INSERT

SHOW-LAST-INSERT

Name

SHOW LAST INSERT

Description

This syntax is used to view the result of the latest insert operation in the current session connection

grammar:

```
SHOW LAST INSERT
```

Example of returned result:

```
     TransactionId: 64067
             Label: insert_ba8f33aea9544866-8ed77e2844d0cc9b
          Database: default_cluster:db1
             Table: t1
TransactionStatus: VISIBLE
        LoadedRows: 2
      FilteredRows: 0
```

illustrate:

- TransactionId: transaction id
- Label: the label corresponding to the insert task
- Database: the database corresponding to insert
- Table: the table corresponding to insert
- TransactionStatus: transaction status
    - PREPARE: preparation stage
    - PRECOMMITTED: Pre-commit stage
    - COMMITTED: The transaction succeeded, but the data was not visible
    - VISIBLE: The transaction succeeded and the data is visible
    - ABORTED: Transaction failed
- LoadedRows: Number of imported rows
- FilteredRows: The number of rows being filtered

Example

Keywords

```
SHOW, LASR ,INSERT
```

Best Practice

SHOW-LOAD-PROFILE

SHOW-LOAD-PROFILE

Name

SHOW LOAD PROFILE

Description

This statement is used to view the Profile information of the import operation. This function requires the user to open the Profile settings. The versions before 0.15 perform the following settings:

```
SET is_report_success=true;
```

Versions 0.15 and later perform the following settings:

```
SET [GLOBAL] enable_profile=true;
```

grammar:

```
show load profile "/";

show load profile "/[queryId]"

show load profile "/[queryId]/[TaskId]"

show load profile "/[queryId]/[TaskId]/[FragmentId]/"

show load profile "/[queryId]/[TaskId]/[FragmentId]/[InstanceId]"
```

This command will list all currently saved import profiles. Each line corresponds to one import. where the QueryId column is the ID of the import job. This ID can also be viewed through the SHOW LOAD statement. We can select the QueryId corresponding to the Profile we want to see to see the specific situation

Example

1. List all Load Profiles

```
mysql> show load profile "/"\G
*************************** 1. row ***************************
             JobId: 20010
           QueryId: 980014623046410a-af5d36f23381017f
              User: root
         DefaultDb: default_cluster:test
               SQL: LOAD LABEL xxx
         QueryType: Load
         StartTime: 2023-03-07 19:48:24
           EndTime: 2023-03-07 19:50:45
```

```
          TotalTime: 2m21s
         QueryState: N/A
            TraceId:
       AnalysisTime: NULL
           PlanTime: NULL
       ScheduleTime: NULL
    FetchResultTime: NULL
    WriteResultTime: NULL
WaitAndFetchResultTime: NULL
*************************** 2. row ***************************
              JobId: N/A
            QueryId: 7cc2d0282a7a4391-8dd75030185134d8
               User: root
          DefaultDb: default_cluster:test
                SQL: insert into xxx
          QueryType: Load
          StartTime: 2023-03-07 19:49:15
            EndTime: 2023-03-07 19:49:15
          TotalTime: 102ms
         QueryState: OK
            TraceId:
       AnalysisTime: 825.277us
           PlanTime: 4.126ms
       ScheduleTime: N/A
    FetchResultTime: 0ns
    WriteResultTime: 0ns
WaitAndFetchResultTime: N/A
```

2. View an overview of the subtasks with imported jobs:

```
mysql> show load profile "/980014623046410a-af5d36f23381017f";
+----------------------------------+------------+
| TaskId                           | ActiveTime |
+----------------------------------+------------+
| 980014623046410a-af5d36f23381017f | 3m14s      |
+----------------------------------+------------+
```

3. View the plan tree of the specified subtask

   ```sql show load profile “/980014623046410a-af5d36f23381017f/980014623046410a-af5d36f23381017f”;

```
                  ┌---┐
                  |[-1: OlapTableSink] |
                  |Fragment: 0 |
                  |MaxActiveTime: 86.541ms|
                  └---┘
                            |
                            |
                    ┌-----------------------------------┐
                    |[1: VEXCHANGE_NODE]|
                    |Fragment: 0 |
                    └-----------------------------------┘
    ┌---┘   └-----------┐
    | |
```

┌-----------------------┐ ┌-----------------------------------┐ |[MemoryUsage]| |[1: VDataStreamSender]| |Fragment: 0 | |Fragment: 1 | └-------------┘ |MaxActiveTime: 34.882ms| └-----------------------------------┘ | | ┌-----------------------------------------┐
```

```
│[0: VNewOlapScanNode(tbl1)]│ │Fragment: 1 │ └──────────────────────────────────────┘
┌──────────────────────────┴──────────┐ │ │ ┌──────────────────┐ ┌──────────────┐
│[MemoryUsage]│ │[VScanner] │ │Fragment: 1 │ │Fragment: 1│ └──────────────┘
└─────────────────┘ ┌──────────┴──────────┐ │ │ ┌──────────────────────┐
┌──────────────────┐ │[SegmentIterator]│ │[MemoryUsage]│ │Fragment: 1 │ │Fragment: 1 │
└──────────────────────┘ └─────────────────┘ └──────────────────────┘
```

```sql

This will show the plan tree and fragment id on it

4. View the Instance overview of the specified subtask

```
mysql> show load profile "/980014623046410a-af5d36f23381017f/980014623046410a-af5d36f23381017f/1"\G
+----------------------------------+-----------------+------------+
| Instances | Host | ActiveTime |
+----------------------------------+-----------------+------------+
980014623046410a-88e260f0c43031f2	10.81.85.89:9067	3m7s
980014623046410a-88e260f0c43031f3	10.81.85.89:9067	3m6s
980014623046410a-88e260f0c43031f4	10.81.85.89:9067	3m10s
980014623046410a-88e260f0c43031f5	10.81.85.89:9067	3m14s
+----------------------------------+-----------------+------------+
```

5. Continue to view the detailed Profile of each operator on a specific Instance

```
mysql> show load profile "/980014623046410a-af5d36f23381017f/980014623046410a-af5d36f23381017f/1/980014623046410a-
88e260f0c43031f5"\G

*************************** 1. row ***************************

Instance:

      ┌--┐
      |[-1: OlapTableSink] |
      |(Active: 2m17s, non-child: 70.91) |
      | - Counters: |
      | - CloseWaitTime: 1m53s |
      | - ConvertBatchTime: 0ns |
      | - MaxAddBatchExecTime: 1m46s |
      | - NonBlockingSendTime: 3m11s |
      | - NumberBatchAdded: 782 |
      | - NumberNodeChannels: 1 |
      | - OpenTime: 743.822us |
      | - RowsFiltered: 0 |
      | - RowsRead: 1.599729M (1599729) |
      | - RowsReturned: 1.599729M (1599729)|
      | - SendDataTime: 11s761ms |
      | - TotalAddBatchExecTime: 1m46s |
      | - ValidateDataTime: 9s802ms |
      └--┘
                          |
┌---┐
|[0: BROKER_SCAN_NODE] |
|(Active: 56s537ms, non-child: 29.06) |
| - Counters: |
| - BytesDecompressed: 0.00 |
| - BytesRead: 5.77 GB |
| - DecompressTime: 0ns |
| - FileReadTime: 34s263ms |
| - MaterializeTupleTime(*): 45s54ms |
```
```

```
|       - NumDiskAccess: 0                                |
|       - PeakMemoryUsage: 33.03 MB                       |
|       - RowsRead: 1.599729M (1599729)                   |
|       - RowsReturned: 1.599729M (1599729)               |
|       - RowsReturnedRate: 28.295K sec                   |
|       - TotalRawReadTime(*): 1m20s                      |
|       - TotalReadThroughput: 30.39858627319336 MB/sec   |
|       - WaitScannerTime: 56s528ms                       |
└---------------------------------------------------------┘
```

Keywords

```
SHOW, LOAD, PROFILE
```

Best Practice

SHOW-LOAD-WARNINGS

SHOW-LOAD-WARNINGS

Name

SHOW LOAD WARNINGS

Description

If the import task fails and the error message is `ETL_QUALITY_UNSATISFIED`, it means that there is an import quality problem. If you want to see these import tasks with quality problems, change the statement to complete this operation.

grammar:

```
SHOW LOAD WARNINGS
[FROM db_name]
[
    WHERE
    [LABEL[="your_label"]]
    [LOAD_JOB_ID = ["job id"]]
]
```

1. If db_name is not specified, the current default db is used
2. If LABEL = is used, it matches the specified label exactly
3. If LOAD_JOB_ID is specified, match the specified JOB ID exactly

Example

1. Display the data with quality problems in the import task of the specified db, and specify the label as “load_demo_20210112”

   ```
   SHOW LOAD WARNINGS FROM demo WHERE LABEL = "load_demo_20210112"
   ```

Keywords

```
SHOW, LOAD, WARNINGS
```

Best Practice

SHOW-INDEX

SHOW-INDEX

Name

SHOW INDEX

Description

This statement is used to display information about indexes in a table.

grammar:

```
SHOW INDEX[ES] FROM [db_name.]table_name [FROM database];
or
SHOW KEY[S] FROM [db_name.]table_name [FROM database];
```

Example

1. Display the lower index of the specified table_name

   ```
   SHOW INDEX FROM example_db.table_name;
   ```

Keywords

```
SHOW, INDEX
```

Best Practice

SHOW PARTITION ID

Description

This statement is used to find the corresponding database name, table name, and partition name based on the partition ID.

Syntax

```
SHOW PARTITION <partition_id>
```

Required Parameters

1. `<partition_id>`

> partition id

Access Control Requirements

The user who executes this SQL must have at least the following permissions:

Privilege	Object	Notes
ADMIN_PRIV		

Examples

1. To find the corresponding database name, table name, and partition name based on the partition ID.

    ```
    SHOW PARTITION 10002;
    ```

SHOW-SNAPSHOT

SHOW-SNAPSHOT

Name

SHOW SNAPSHOT

Description

This statement is used to view backups that already exist in the repository.

grammar:

```
SHOW SNAPSHOT ON `repo_name`
[WHERE SNAPSHOT = "snapshot" [AND TIMESTAMP = "backup_timestamp"]];
```

illustrate:

1. The meanings of the columns are as follows: Snapshot: The name of the backup Timestamp: corresponds to the time version of the backup Status: If the backup is normal, it will display OK, otherwise it will display an error message
2. If TIMESTAMP is specified, the following additional information is displayed: Database: The name of the database to which the backup data originally belonged Details: In the form of Json, the data directory and file structure of the entire backup are displayed

Example

1. View the existing backups in the repository example_repo

   ```
   SHOW SNAPSHOT ON example_repo;
   ```

2. View only the backup named backup1 in the repository example_repo:

   ```
   SHOW SNAPSHOT ON example_repo WHERE SNAPSHOT = "backup1";
   ```

3. View the details of the backup named backup1 in the warehouse example_repo with the time version “2018-05-05-15-34-26”:

   ```
   SHOW SNAPSHOT ON example_repo
   WHERE SNAPSHOT = "backup1" AND TIMESTAMP = "2018-05-05-15-34-26";
   ```

Keywords

```
SHOW, SNAPSHOT
```

Best Practice

SHOW-SQL-BLOCK-RULE

SQL-BLOCK-RULE

Name

SHOW SQL BLOCK RULE

Description

View the configured SQL blocking rules. If you do not specify a rule name, you will view all rules.

grammar:

```
SHOW SQL_BLOCK_RULE [FOR RULE_NAME];
```

Example

1. View all rules.

```
mysql> SHOW SQL_BLOCK_RULE;
+------------+----------------------------+---------+- -------------+------------+-------------+--------+- -------+
| Name | Sql | SqlHash | PartitionNum | TabletNum | Cardinality | Global | Enable |
+------------+----------------------------+---------+- -------------+------------+-------------+--------+- -------+
| test_rule | select * from order_analysis | NULL | 0 | 0 | 0 | true | true |
| test_rule2 | NULL | NULL | 30 | 0 | 10000000000 | false | true |
+------------+----------------------------+---------+- -------------+------------+-------------+--------+- -------+
2 rows in set (0.01 sec)
```

2. Make a rule name query

```
mysql> SHOW SQL_BLOCK_RULE FOR test_rule2;
+------------+------+---------+---------------+---- -------+-------------+--------+--------+
| Name | Sql | SqlHash | PartitionNum | TabletNum | Cardinality | Global | Enable |
+------------+------+---------+---------------+---- -------+-------------+--------+--------+
| test_rule2 | NULL | NULL | 30 | 0 | 10000000000 | false | true |
+------------+------+---------+---------------+---- -------+-------------+--------+--------+
1 row in set (0.00 sec)
```

Keywords

```
SHOW, SQL_BLOCK_RULE
```

Best Practice

SHOW-ROUTINE-LOAD

SHOW-ROUTINE-LOAD

Name

SHOW ROUTINE LOAD

Description

This statement is used to display the running status of the Routine Load job

grammar:

```
SHOW [ALL] ROUTINE LOAD [FOR jobName];
```

Result description:

```
                 Id: job ID
               Name: job name
         CreateTime: job creation time
          PauseTime: The last job pause time
            EndTime: Job end time
             DbName: corresponding database name
          TableName: The name of the corresponding table (In the case of multiple tables, since it is a dynamic table, the specific
table name is not displayed, and we uniformly display it as "multi-table").
          IsMultiTbl: Indicates whether it is a multi-table
              State: job running state
     DataSourceType: Data source type: KAFKA
     CurrentTaskNum: The current number of subtasks
      JobProperties: Job configuration details
DataSourceProperties: Data source configuration details
    CustomProperties: custom configuration
           Statistic: Job running status statistics
            Progress: job running progress
                 Lag: job delay status
ReasonOfStateChanged: The reason for the job state change
        ErrorLogUrls: The viewing address of the filtered unqualified data
            OtherMsg: other error messages
```

- State

  ```
  There are the following 5 states:
  * NEED_SCHEDULE: The job is waiting to be scheduled
  * RUNNING: The job is running
  * PAUSED: The job is paused
  * STOPPED: The job has ended
  * CANCELLED: The job was canceled
  ```

- Progress

```
For Kafka data sources, displays the currently consumed offset for each partition. For example, {"0":"2"} indicates that
the consumption progress of Kafka partition 0 is 2.
```

*Lag

```
  For Kafka data sources, shows the consumption latency of each partition. For example, {"0":10} means that the consumption delay of
Kafka partition 0 is 10.
```

Example

1. Show all routine import jobs named test1 (including stopped or canceled jobs). The result is one or more lines.

   ```
   SHOW ALL ROUTINE LOAD FOR test1;
   ```

2. Show the currently running routine import job named test1

   ```
   SHOW ROUTINE LOAD FOR test1;
   ```

3. Display all routine import jobs (including stopped or canceled jobs) under example_db. The result is one or more lines.

   ```
   use example_db;
   SHOW ALL ROUTINE LOAD;
   ```

4. Display all running routine import jobs under example_db

   ```
   use example_db;
   SHOW ROUTINE LOAD;
   ```

5. Display the currently running routine import job named test1 under example_db

   ```
   SHOW ROUTINE LOAD FOR example_db.test1;
   ```

6. Displays all routine import jobs named test1 under example_db (including stopped or canceled jobs). The result is one or more lines.

   ```
   SHOW ALL ROUTINE LOAD FOR example_db.test1;
   ```

Keywords

```
SHOW, ROUTINE, LOAD
```

Best Practice

SHOW-SYNC-JOB

SHOW-SYNC-JOB

Name

SHOW SYNC JOB

Description

This command is used to currently display the status of resident data synchronization jobs in all databases.

grammar:

```
SHOW SYNC JOB [FROM db_name]
```

Example

1. Display the status of all data synchronization jobs in the current database.

   ```
   SHOW SYNC JOB;
   ```

2. Display the status of all data synchronization jobs under the database `test_db`.

   ```
   SHOW SYNC JOB FROM `test_db`;
   ```

Keywords

```
SHOW, SYNC, JOB
```

Best Practice

SHOW-WHITE-LIST

SHOW-WHITE-LIST

Description

Example

Keywords

```
SHOW, WHITE, LIST
```

Best Practice

SHOW-WARNING

SHOW-WARNING

Description

Example

Keywords

```
SHOW, WARNING
```

Best Practice

SHOW TABLET

Description

This statement is used to show details of a specific tablet (only for administrators).

Syntax

```
SHOW TABLET <tablet_id>;
```

Required Parameters

1. `<tablet_id>`

> The ID of the specific tablet to display information for.

Return Value

When using `SHOW TABLET <tablet_id>`, the following columns are returned:

Column	DataType	Note
DbName	String	The name of the database that contains the tablet.
TableName	String	The name of the table that contains the tablet.
PartitionName	String	The name of the partition that contains the tablet.
IndexName	String	The name of the index that contains the tablet.
DbId	Int	The ID of the database.
TableId	Int	The ID of the table.
PartitionId	Int	The ID of the partition.
IndexId	Int	The ID of the index.
IsSync	Boolean	Whether the tablet is in sync with its replicas.
Order	Int	The order of the tablet.
QueryHits	Int	The number of query hits on this tablet.
DetailCmd	String	The command to get more detailed information about the tablet.

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

Show details of a specific tablet:

```
SHOW TABLET 10145;
```

```
+--------+-----------+---------------+-----------+-------+---------+-------------+---------+--------+-------+-----------+-----------
-----------------------------------------------+
| DbName | TableName | PartitionName | IndexName | DbId  | TableId | PartitionId | IndexId | IsSync | Order | QueryHits | DetailCmd
|
+--------+-----------+---------------+-----------+-------+---------+-------------+---------+--------+-------+-----------+-----------
-----------------------------------------------+
| test   | sell_user | sell_user     | sell_user | 10103 | 10143   | 10142       | 10144   | true   | 0     | 0         | SHOW PROC
'/dbs/10103/10143/partitions/10142/10144/10145'; |
+--------+-----------+---------------+-----------+-------+---------+-------------+---------+--------+-------+-----------+-----------
-----------------------------------------------+
```

SHOW TABLETS BELONG

Description

This statement is used to show tablets and information of their belonging table.

Syntax

```
SHOW TABLETS BELONG <tablet_id> [, <tablet_id>]...;
```

Required Parameters

1. `<tablet_id>`

> One or more tablet IDs, separated by commas. Duplicate IDs will be deduplicated in the result.

Return Value

When using `SHOW TABLETS BELONG <tablet_id> [, <tablet_id>]...`, the following columns are returned:

Column	DataType	Note
DbName	String	The name of the database that contains the tablet.
TableName	String	The name of the table that contains the tablet.
TableSize	String	The size of the table (e.g., “8.649 KB”).
PartitionNum	Int	The number of partitions in the table.
BucketNum	Int	The number of buckets in the table.
ReplicaCount	Int	The number of replicas in the table.
TabletIds	Array	The list of tablet IDs that belong to the table.

Access Control Requirements

The user executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
Admin_priv	Database	Required to execute administrative operations on the database, including managing tables, partitions, and system-level commands.

Examples

Show information about a specific tablet:

```
SHOW TABLETS BELONG 10145;
```

```
+--------+-----------+-----------+--------------+-----------+--------------+-----------+
| DbName | TableName | TableSize | PartitionNum | BucketNum | ReplicaCount | TabletIds |
+--------+-----------+-----------+--------------+-----------+--------------+-----------+
| test   | sell_user | 8.649 KB  | 1            | 4         | 4            | [10145]   |
+--------+-----------+-----------+--------------+-----------+--------------+-----------+
```

Show information about multiple tablets:

```
SHOW TABLETS BELONG 27028,78880,78382,27028;
```

```
+---------------------+-----------+-----------+--------------+-----------+--------------+----------------+
| DbName              | TableName | TableSize | PartitionNum | BucketNum | ReplicaCount | TabletIds      |
+---------------------+-----------+-----------+--------------+-----------+--------------+----------------+
| default_cluster:db1 | kec       | 613.000 B | 379          | 604       | 604          | [78880, 78382] |
| default_cluster:db1 | test      | 1.874 KB  | 1            | 1         | 1            | [27028]        |
+---------------------+-----------+-----------+--------------+-----------+--------------+----------------+
```

SHOW-VARIABLES

SHOW-VARIABLES

Name

SHOW VARIABLES

Description

This statement is used to display Doris system variables, which can be queried by conditions

grammar:

```
SHOW [GLOBAL | SESSION] VARIABLES
     [LIKE 'pattern' | WHERE expr]
```

illustrate:

- show variables is mainly used to view the values of system variables.
- Executing the SHOW VARIABLES command does not require any privileges, it only requires being able to connect to the server.
- Use the like statement to match with variable_name.
- The % percent wildcard can be used anywhere in the matching pattern

Example

1. The default here is to match the Variable_name, here is the exact match

   ```
   show variables like 'max_connections';
   ```

2. Matching through the percent sign (%) wildcard can match multiple items

   ```
   show variables like '%connec%';
   ```

3. Use the Where clause for matching queries

   ```
   show variables where variable_name = 'version';
   ```

Keywords

```
SHOW, VARIABLES
```

Best Practice

SHOW-PLUGINS

SHOW-PLUGINS

Name

SHOW PLUGINS

Description

This statement is used to display installed plugins

grammar:

```
SHOW PLUGINS
```

This command will display all user-installed and system built-in plugins

Example

1. Show installed plugins:

   ```
   SHOW PLUGINS;
   ```

Keywords

```
SHOW, PLUGINS
```

Best Practice

SHOW-ROLES

SHOW-ROLES

Name

SHOW ROLES

Description

This statement is used to display all created role information, including role name, included users and permissions.

grammar:

```
SHOW ROLES
```

Example

1. View created roles

   ```
   SHOW ROLES
   ```

Keywords

```
SHOW, ROLES
```

Best Practice

SHOW-PROCEDURE

SHOW-PROCEDURE

Description

Example

Keywords

```
SHOW, PROCEDURE
```

Best Practice

SHOW-ROUTINE-LOAD-TASK

SHOW-ROUTINE-LOAD-TASK

Name

SHOW ROUTINE LOAD TASK

Description

View the currently running subtasks of a specified Routine Load job.

```
SHOW ROUTINE LOAD TASK
WHERE JobName = "job_name";
```

The returned results are as follows:

```
              TaskId: d67ce537f1be4b86-abf47530b79ab8e6
               TxnId: 4
           TxnStatus: UNKNOWN
               JobId: 10280
          CreateTime: 2020-12-12 20:29:48
    ExecuteStartTime: 2020-12-12 20:29:48
             Timeout: 20
                BeId: 10002
DataSourceProperties: {"0":19}
```

- `TaskId`: The unique ID of the subtask.
- `TxnId`: The import transaction ID corresponding to the subtask.
- `TxnStatus`: The import transaction status corresponding to the subtask. When TxnStatus is null, it means that the subtask has not yet started scheduling.
- `JobId`: The job ID corresponding to the subtask.
- `CreateTime`: The creation time of the subtask.
- `ExecuteStartTime`: The time when the subtask is scheduled to be executed, usually later than the creation time.
- `Timeout`: Subtask timeout, usually twice the `max_batch_interval` set by the job.
- `BeId`: The ID of the BE node executing this subtask.
- `DataSourceProperties`: The starting offset of the Kafka Partition that the subtask is ready to consume. is a Json format string. Key is Partition Id. Value is the starting offset of consumption.

Example

1. Display the subtask information of the routine import task named test1.

   ```
   SHOW ROUTINE LOAD TASK WHERE JobName = "test1";
   ```

Keywords

```
SHOW, ROUTINE, LOAD, TASK
```

Best Practice

With this command, you can view how many subtasks are currently running in a Routine Load job, and which BE node is running on.

SHOW-PROC

SHOW-PROC

Name

SHOW PROC

Description

The Proc system is a unique feature of Doris. Students who have used Linux may understand this concept better. In Linux systems, proc is a virtual file system, usually mounted in the /proc directory. Users can view the internal data structure of the system through this file system. For example, you can view the details of the specified pid process through /proc/pid.

Similar to the proc system in Linux, the proc system in Doris is also organized into a directory-like structure to view different system information according to the “directory path (proc path)” specified by the user.

The proc system is designed mainly for system administrators, so that it is convenient for them to view some running states inside the system. Such as the tablet status of the table, the cluster balance status, the status of various jobs, and so on. is a very useful function

There are two ways to view the proc system in Doris.

1. View through the WEB UI interface provided by Doris, visit the address: `http://FE_IP:FE_HTTP_PORT`
2. Another way is by command

All commands supported by Doris PROC can be seen through `SHOW PROC "/";`

After connecting to Doris through the MySQL client, you can execute the SHOW PROC statement to view the information of the specified proc directory. The proc directory is an absolute path starting with “/”.

The results of the show proc statement are presented in a two-dimensional table. And usually the first column of the result table is the next subdirectory of proc.

```
mysql> show proc "/";
+---------------------------+
| name                      |
+---------------------------+
| auth                      |
| backends                  |
| bdbje                     |
| brokers                   |
| catalogs                  |
| cluster_balance           |
| cluster_health            |
| colocation_group          |
| current_backend_instances |
| current_queries           |
| current_query_stmts       |
| dbs                       |
```

```
| frontends                 |
| jobs                      |
| load_error_hub            |
| monitor                   |
| resources                 |
| routine_loads             |
| statistic                 |
| stream_loads              |
| tasks                     |
| transactions              |
| trash                     |
+---------------------------+
23 rows in set (0.00 sec)
```

illustrate:

1. auth: User name and corresponding permission information
2. backends: Displays the node list of BE in the cluster, equivalent to SHOW BACKENDS
3. bdbje: To view the bdbje database list, you need to modify the `fe.conf` file to add `enable_bdbje_debug_mode=true`, and then start `FE` through `sh start_fe.sh --daemon` to enter the `debug` mode. After entering `debug` mode, only `http server` and `MySQLServer` will be started and the `BDBJE` instance will be opened, but no metadata loading and subsequent startup processes will be entered.
4. brokers : View cluster broker node information, equivalent to SHOW BROKER
5. catalogs : View the data catalogs that have been created currently, equivalent to SHOW CATALOGS
6. cluster_balance : To check the balance of the cluster, please refer to Data Copy Management
7. cluster_health: Run `SHOW PROC '/cluster_health/tablet_health';` statement to view the replica status of the entire cluster.
8. colocation_group : This command can view the existing Group information in the cluster. For details, please refer to the Colocation Join chapter
9. current_backend_instances : Displays a list of be nodes that are currently executing jobs
10. current_queries : View the list of queries being executed, the SQL statement currently running.
11. current_query_stmts: Returns the currently executing query.
12. dbs: Mainly used to view the metadata information of each database and the tables in the Doris cluster. This information includes table structure, partitions, materialized views, data shards and replicas, and more. Through this directory and its subdirectories, you can clearly display the table metadata in the cluster, and locate some problems such as data skew, replica failure, etc.
13. frontends: Display all FE node information in the cluster, including IP address, role, status, whether it is a mater, etc., equivalent to SHOW FRONTENDS
14. jobs: show statistics of all kind of jobs. If a specific `dbId` is given, will return statistics data of the database. If `dbId` is -1, will return total statistics data of all databases
15. load_error_hub: Doris supports centralized storage of error information generated by load jobs in an error hub. Then view the error message directly through the `SHOW LOAD WARNINGS;` statement. Shown here is the configuration information of the error hub.
16. monitor : shows the resource usage of FE JVM
17. resources : View system resources, ordinary accounts can only see resources that they have USAGE_PRIV permission to use. Only the root and admin accounts can see all resources. Equivalent to SHOW RESOURCES
18. routine_loads: Display all routine load job information, including job name, status, etc.
19. statistics: It is mainly used to summarize and view the number of databases, tables, partitions, shards, and replicas in the Doris cluster. and the number of unhealthy copies. This information helps us to control the size of the cluster meta-information in general. It helps us view the cluster sharding situation

from an overall perspective, and can quickly check the health of the cluster sharding. This further locates problematic data shards.

20. stream_loads: Returns the stream load task being executed.
21. tasks : Displays the total number of tasks of various jobs and the number of failures.
22. transactions : used to view the transaction details of the specified transaction id, equivalent to SHOW TRANSACTION
23. trash: This statement is used to view the space occupied by garbage data in the backend. Equivalent to SHOW TRASH

Example

1. For example, “/dbs” displays all databases, and “/dbs/10002” displays all tables under the database with id 10002

```
mysql> show proc "/dbs/10002";
+---------+----------------------+----------+--------------------+--------------+--------+------+-------------------------
-+--------------+
| TableId | TableName            | IndexNum | PartitionColumnName | PartitionNum | State  | Type | LastConsistencyCheckTime
| ReplicaCount |
+---------+----------------------+----------+--------------------+--------------+--------+------+-------------------------
-+--------------+
| 10065   | dwd_product_live     | 1        | dt                 | 9            | NORMAL | OLAP | NULL
| 18           |
| 10109   | ODS_MR_BILL_COSTS_DO | 1        | NULL               | 1            | NORMAL | OLAP | NULL
| 1            |
| 10119   | test                 | 1        | NULL               | 1            | NORMAL | OLAP | NULL
| 1            |
| 10124   | test_parquet_import  | 1        | NULL               | 1            | NORMAL | OLAP | NULL
| 1            |
+---------+----------------------+----------+--------------------+--------------+--------+------+-------------------------
-+--------------+
4 rows in set (0.00 sec)
```

2. Display information about the number of all database tables in the cluster.

```
mysql> show proc '/statistic';
+-------+----------------------+----------+--------------+----------+-----------+------------+
| DbId  | DbName               | TableNum | PartitionNum | IndexNum | TabletNum | ReplicaNum |
+-------+----------------------+----------+--------------+----------+-----------+------------+
| 10002 | default_cluster:test | 4        | 12           | 12       | 21        | 21         |
| Total | 1                    | 4        | 12           | 12       | 21        | 21         |
+-------+----------------------+----------+--------------+----------+-----------+------------+
2 rows in set (0.00 sec)
```

3. The following command can view the existing Group information in the cluster.

```
SHOW PROC '/colocation_group';

+-------------+--------------+--------------+------------+----------------+----------+----------+
| GroupId     | GroupName    | TableIds     | BucketsNum | ReplicationNum | DistCols | IsStable |
+-------------+--------------+--------------+------------+----------------+----------+----------+
| 10005.10008 | 10005_group1 | 10007, 10040 | 10         | 3              | int(11)  | true     |
+-------------+--------------+--------------+------------+----------------+----------+----------+
```

- GroupId: The cluster-wide unique identifier of a group, the first half is the db id, and the second half is the group id.
- GroupName: The full name of the Group.
- TabletIds: The id list of Tables contained in this Group.
- BucketsNum: The number of buckets.
- ReplicationNum: The number of replicas.
- DistCols: Distribution columns, that is, the bucket column type.
- IsStable: Whether the Group is stable (for the definition of stability, see the `Colocation replica balance and repair` section).

4. Use the following commands to further view the data distribution of a Group:

```
SHOW PROC '/colocation_group/10005.10008';

+-------------+---------------------+
| BucketIndex | BackendIds          |
+-------------+---------------------+
| 0           | 10004, 10002, 10001 |
| 1           | 10003, 10002, 10004 |
| 2           | 10002, 10004, 10001 |
| 3           | 10003, 10002, 10004 |
| 4           | 10002, 10004, 10003 |
| 5           | 10003, 10002, 10001 |
| 6           | 10003, 10004, 10001 |
| 7           | 10003, 10004, 10002 |
+-------------+---------------------+
```

- BucketIndex: The index of the bucket sequence.
- BackendIds: The list of BE node IDs where the data shards in the bucket are located.

5. Display the total number of tasks of various jobs and the number of failures.

```
mysql> show proc '/tasks';
+--------------------------+-----------+----------+
| TaskType                 | FailedNum | TotalNum |
+--------------------------+-----------+----------+
| CREATE                   | 0         | 0        |
| DROP                     | 0         | 0        |
| PUSH                     | 0         | 0        |
| CLONE                    | 0         | 0        |
| STORAGE_MEDIUM_MIGRATE   | 0         | 0        |
| ROLLUP                   | 0         | 0        |
| SCHEMA_CHANGE            | 0         | 0        |
| CANCEL_DELETE            | 0         | 0        |
| MAKE_SNAPSHOT            | 0         | 0        |
| RELEASE_SNAPSHOT         | 0         | 0        |
| CHECK_CONSISTENCY        | 0         | 0        |
| UPLOAD                   | 0         | 0        |
| DOWNLOAD                 | 0         | 0        |
| CLEAR_REMOTE_FILE        | 0         | 0        |
| MOVE                     | 0         | 0        |
| REALTIME_PUSH            | 0         | 0        |
| PUBLISH_VERSION          | 0         | 0        |
| CLEAR_ALTER_TASK         | 0         | 0        |
| CLEAR_TRANSACTION_TASK   | 0         | 0        |
| RECOVER_TABLET           | 0         | 0        |
| STREAM_LOAD              | 0         | 0        |
| UPDATE_TABLET_META_INFO  | 0         | 0        |
```

```
| ALTER                  | 0         | 0         |
| INSTALL_PLUGIN         | 0         | 0         |
| UNINSTALL_PLUGIN       | 0         | 0         |
| Total                  | 0         | 0         |
+------------------------+-----------+-----------+
26 rows in set (0.01 sec)
```

6. Display the replica status of the entire cluster.

```
mysql> show proc '/cluster_health/tablet_health';
+----------+---------------------------+-----------+------------+-------------------+----------------------+---------------
-------+--------------+---------------------------+------------------------+------------------+--------------------+---
------------------+----------------------+-----------------+----------------------------+-----------------+-------------
+------------+
| DbId     | DbName                    | TabletNum | HealthyNum | ReplicaMissingNum | VersionIncompleteNum |
ReplicaRelocatingNum | RedundantNum | ReplicaMissingInClusterNum | ReplicaMissingForTagNum | ForceRedundantNum |
ColocateMismatchNum | ColocateRedundantNum | NeedFurtherRepairNum | UnrecoverableNum | ReplicaCompactionTooSlowNum |
InconsistentNum | OversizeNum | CloningNum |
+----------+---------------------------+-----------+------------+-------------------+----------------------+---------------
-------+--------------+---------------------------+------------------------+------------------+--------------------+---
------------------+----------------------+-----------------+----------------------------+-----------------+-------------
+------------+
| 25852112 | default_cluster:bowen     | 1920      | 1920       | 0                 | 0                    | 0
| 0            | 0                         | 0                       | 0                | 0                   | 0
| 0                    | 0               | 0                         | 0              | 0           | 0          |
| 25342914 | default_cluster:bw        | 128       | 128        | 0                 | 0                    | 0
| 0            | 0                         | 0                       | 0                | 0                   | 0
| 0                    | 0               | 0                         | 0              | 0           | 0          |
| 2575532  | default_cluster:cps       | 1440      | 1440       | 0                 | 0                    | 0
| 0            | 0                         | 0                       | 0                | 0                   | 0
| 0                    | 0               | 0                         | 0              | 16          | 0          |
| 26150325 | default_cluster:db        | 38374     | 38374      | 0                 | 0                    | 0
| 0            | 0                         | 0                       | 0                | 0                   | 0
| 0                    | 0               | 0                         | 0              | 453         | 0          |
+----------+---------------------------+-----------+------------+-------------------+----------------------+---------------
-------+--------------+---------------------------+------------------------+------------------+--------------------+---
------------------+----------------------+-----------------+----------------------------+-----------------+-------------
+------------+
4 rows in set (0.01 sec)
```

View the replica status under a database, such as a database with a DbId of 25852112.

```
mysql> show proc '/cluster_health/tablet_health/25852112';
```

Keywords

SHOW, PROC

Best Practice

SHOW-TABLE-STATS

SHOW-TABLE-STATS

Name

SHOW TABLE STATS

Description

Use `SHOW TABLE STATS` to view an overview of statistics collection for a table.

Syntax:

```
SHOW TABLE STATS table_name;
```

Where:

- table_name: The target table name. It can be in the format `db_name.table_name`.

Output:

Column Name	Description
`updated_rows`	Updated rows since the last ANALYZE
`query_times`	Reserved column for recording the number of times the table was queried in future versions
`row_count`	Number of rows (does not reflect the exact number of rows at the time of command execution)
`updated_time`	Last update time
`columns`	Columns for which statistics information has been collected

Here's an example:

```
mysql> show table stats lineitem \G;
*************************** 1. row ***************************
updated_rows: 0
 query_times: 0
   row_count: 6001215
updated_time: 2023-11-07
     columns: [l_returnflag, l_receiptdate, l_tax, l_shipmode, l_suppkey, l_shipdate, l_commitdate, l_partkey, l_orderkey,
l_quantity, l_linestatus, l_comment, l_extendedprice, l_linenumber, l_discount, l_shipinstruct]
     trigger: MANUAL
```

Keywords

SHOW, TABLE, STATS

SHOW-TABLE-STATUS

SHOW-TABLE-STATUS

Name

SHOW TABLE STATUS

Description

This statement is used to view some information about the Table.

grammar:

```
SHOW TABLE STATUS
[FROM db] [LIKE "pattern"]
```

illustrate:

1. This statement is mainly used to be compatible with MySQL syntax, currently only a small amount of information such as Comment is displayed

Example

1. View the information of all tables under the current database

   ```
   SHOW TABLE STATUS;
   ```

2. View the information of the table whose name contains example under the specified database

   ```
   SHOW TABLE STATUS FROM db LIKE "%example%";
   ```

Keywords

```
SHOW, TABLE, STATUS
```

Best Practice

SHOW-REPOSITORIES

SHOW-REPOSITORIES

Name

SHOW REPOSITORIES

Description

This statement is used to view the currently created warehouse

grammar:

```
SHOW REPOSITORIES;
```

illustrate:

1. The meanings of the columns are as follows: RepoId: Unique repository ID RepoName: repository name CreateTime: The time when the repository was first created IsReadOnly: Whether it is a read-only repository Location: The root directory in the warehouse for backing up data Broker: Dependent Broker ErrMsg: Doris will regularly check the connectivity of the warehouse, if there is a problem, an error message will be displayed here

Example

1. View the created repository:

```
  SHOW REPOSITORIES;
```

Keywords

```
SHOW, REPOSITORIES
```

Best Practice

SHOW-QUERY-PROFILE

SHOW-QUERY-PROFILE

Name

SHOW QUERY PROFILE

Description

This statement is used to view the tree profile information of the query operation,this function requires the user to open profile settings. Before versions 0.15, perform the following settings:

```
SET is_report_success=true;
```

For versions 0.15 and later, perform the following settings:

```
SET [GLOBAL] enable_profile=true;
```

grammar:

```
show query profile "/";
```

This command will list the profiles of all currently saved query operations.

```
show query profile "/queryId"\G
show query profile "/queryId/fragment_id/instance_id"\G
```

Get the tree profile information of the specified query ID,Return to profile simple tree.Specify fragment_ ID and instance_ ID returns the corresponding detailed profile tree.

Example

1. List all query Profile

```
mysql> show query profile "/";
+----------------------------------+------+-------------------------+-------------------+-----------+---------------------
-+---------------------+-----------+------------+
| QueryId                          | User | DefaultDb               | SQL               | QueryType | StartTime
| EndTime             | TotalTime | QueryState |
+----------------------------------+------+-------------------------+-------------------+-----------+---------------------
-+---------------------+-----------+------------+
| 327167e0db4749a9-adce3b3d770b2bb1 | root | default_cluster:test_db | select * from test | Query     | 2022-08-09 10:50:09
| 2022-08-09 10:50:09 | 19ms      | EOF        |
+----------------------------------+------+-------------------------+-------------------+-----------+---------------------
-+---------------------+-----------+------------+
1 row in set (0.00 sec)
```

2. List the query profile of the specified queryid

```
mysql> show query profile "/327167e0db4749a9-adce3b3d770b2bb1"\G
*************************** 1. row ***************************
Fragments: ┌──────────────────────────────────────────────┐
           │[-1: VDataBufferSender]  │
           │Fragment: 0              │
           │MaxActiveTime: 783.263us │
           └──────────────────────────────────────────────┘
                          ┌─┘
                          │
             ┌────────────────────────────┐
             │[1: VEXCHANGE_NODE]│
             │Fragment: 0        │
             └────────────────────────────┘
                          └─┐
                            │
           ┌──────────────────────────────────────────────┐
           │[1: VDataStreamSender]   │
           │Fragment: 1              │
           │MaxActiveTime: 847.612us │
           └──────────────────────────────────────────────┘
                            │
                            │
             ┌────────────────────────────┐
             │[0: VOLAP_SCAN_NODE]│
             │Fragment: 1         │
             └────────────────────────────┘
                          ┌─┘
                          │
                ┌──────────────────┐
                │[OlapScanner]│
                │Fragment: 1   │
                └──────────────────┘
                          │
                          │
           ┌──────────────────────────────┐
           │[SegmentIterator]│
           │Fragment: 1      │
           └──────────────────────────────┘
1 row in set (0.00 sec)
```

3. Lists the instance profile of the specified fragment:

```
mysql> show query profile "/327167e0db4749a9-adce3b3d770b2bb1/1/"\G
*************************** 1. row ***************************
 Instances: 327167e0db4749a9-adce3b3d770b2bb2
      Host: 172.26.0.1:9111
ActiveTime: 847.612us
1 row in set (0.01 sec)
```

4. Continue to view the detailed profile of each operator on a specific instance:

```
mysql> show query profile "/327167e0db4749a9-adce3b3d770b2bb1/1/327167e0db4749a9-adce3b3d770b2bb2"\G
*************************** 1. row ***************************
Instance: ┌──────────────────────────────────────────────────────────────────────┐
          │[1: VDataStreamSender]                 │
          │(Active: 36.944us, non-child: 0.20)    │
          │   - Counters:                         │
          │       - BytesSent: 0.00               │
          │       - IgnoreRows: 0                 │
```

```
|       - LocalBytesSent: 20.00 B         |
|       - OverallThroughput: 0.0 /sec     |
|       - PeakMemoryUsage: 0.00           |
|       - SerializeBatchTime: 0ns         |
|       - UncompressedRowBatchSize: 0.00  |
└─────────────────────────────────────────┘
                    |
                    |
┌─────────────────────────────────────────┐
| [0: VOLAP_SCAN_NODE]                    |
| (Active: 563.241us, non-child: 3.00)    |
|   - Counters:                           |
|       - BatchQueueWaitTime: 444.714us   |
|       - BytesRead: 37.00 B              |
|       - NumDiskAccess: 1                |
|       - NumScanners: 2                  |
|       - PeakMemoryUsage: 320.00 KB      |
|       - RowsRead: 4                     |
|       - RowsReturned: 4                 |
|       - RowsReturnedRate: 7.101K /sec   |
|       - ScannerBatchWaitTime: 206.40us  |
|       - ScannerSchedCount : 2           |
|       - ScannerWorkerWaitTime: 34.640us |
|       - TabletCount : 2                 |
|       - TotalReadThroughput: 0.0 /sec   |
└─────────────────────────────────────────┘
                    |
                    |
    ┌──────────────────────────────────┐
    | [OlapScanner]                    |
    | (Active: 0ns, non-child: 0.00)   |
    |   - Counters:                    |
    |       - BlockConvertTime: 0ns    |
    |       - BlockFetchTime: 183.741us|
    |       - ReaderInitTime: 180.741us|
    |       - RowsDelFiltered: 0       |
    |       - RowsPushedCondFiltered: 0|
    |       - ScanCpuTime: 388.576us   |
    |       - ScanTime: 0ns            |
    |       - ShowHintsTime_V1: 0ns    |
    └──────────────────────────────────┘
                    |
                    |
  ┌──────────────────────────────────────┐
  | [SegmentIterator]                    |
  | (Active: 0ns, non-child: 0.00)       |
  |   - Counters:                        |
  |       - BitmapIndexFilterTimer: 124ns|
  |       - BlockLoadTime: 179.202us     |
  |       - BlockSeekCount: 5            |
  |       - BlockSeekTime: 18.792us      |
  |       - BlocksLoad: 4                |
  |       - CachedPagesNum: 2            |
  |       - CompressedBytesRead: 0.00    |
  |       - DecompressorTimer: 0ns       |
  |       - IOTimer: 0ns                 |
  |       - IndexLoadTime_V1: 0ns        |
  |       - NumSegmentFiltered: 0        |
  |       - NumSegmentTotal: 2           |
  |       - RawRowsRead: 4               |
  |       - RowsBitmapIndexFiltered: 0   |
  |       - RowsBloomFilterFiltered: 0   |
  |       - RowsConditionsFiltered: 0    |
  |       - RowsKeyRangeFiltered: 0      |
  |       - RowsStatsFiltered: 0         |
```

```
  |       - RowsVectorPredFiltered: 0     |
  |       - TotalPagesNum: 2              |
  |       - UncompressedBytesRead: 0.00   |
  |       - VectorPredEvalTime: 0ns       |
  └────────────────────────────────────────┘

1 row in set (0.01 sec)
```

Keywords

```
SHOW, QUERY, PROFILE
```

Best Practice

SHOW-OPEN-TABLES

SHOW-OPEN-TABLES

Description

Example

Keywords

```
SHOW, OPEN, TABLES
```

Best Practice

SHOW-TABLETS

SHOW-TABLETS

Name

SHOW TABLETS

Description

This statement is used to list tablets of the specified table (only for administrators)

grammar:

```
SHOW TABLETS FROM [database.]table [PARTITIONS(p1,p2)]
[WHERE where_condition]
[ORDER BY col_name]
[LIMIT [offset,] row_count]
```

1. **Syntax Description:**

where_condition could be one of:

```
Version = version
state = "NORMAL|ROLLUP|CLONE|DECOMMISSION"
BackendId = backend_id
IndexName = rollup_name
```

or compound them with operator `AND`.

Example

1. list all tablets of the specified table

   ```
   SHOW TABLETS FROM example_db.table_name;
   ```

2. list all tablets of the specified partitions

   ```
   SHOW TABLETS FROM example_db.table_name PARTITIONS(p1, p2);
   ```

3. list all DECOMMISSION tablets on the specified backend

   ```
   SHOW TABLETS FROM example_db.table_name WHERE state="DECOMMISSION" AND BackendId=11003;
   ```

Keywords

```
SHOW, TABLETS
```

Best Practice

SHOW-LOAD

SHOW-LOAD

Name

SHOW LOAD

Description

This statement is used to display the execution of the specified import task

grammar:

```
SHOW LOAD
[FROM db_name]
[
   WHERE
   [LABEL [ = "your_label" | LIKE "label_matcher"]]
   [STATE = ["PENDING"|"ETL"|"LOADING"|"FINISHED"|"CANCELLED"|]]
]
[ORDER BY...]
[LIMIT limit][OFFSET offset];
```

illustrate:

1. If db_name is not specified, the current default db is used
2. If LABEL LIKE is used, it will match import tasks whose label contains label_matcher
3. If LABEL = is used, it will match the specified label exactly
4. If STATE is specified, matches the LOAD state
5. You can use ORDER BY to sort on any combination of columns
6. If LIMIT is specified, limit matching records are displayed. Otherwise show all
7. If OFFSET is specified, the query results are displayed starting at offset offset. By default the offset is 0.
8. If you are using broker/mini load, the connections in the URL column can be viewed using the following command:

   ```
   SHOW LOAD WARNINGS ON 'url'
   ```

Example

1. Show all import tasks for default db

```
SHOW LOAD;
```

2. Display the import tasks of the specified db, the label contains the string “2014_01_02”, and display the oldest 10

```
SHOW LOAD FROM example_db WHERE LABEL LIKE "2014_01_02" LIMIT 10;
```

3. Display the import tasks of the specified db, specify the label as “load_example_db_20140102” and sort by LoadStartTime in descending order

```
SHOW LOAD FROM example_db WHERE LABEL = "load_example_db_20140102" ORDER BY LoadStartTime DESC;
```

4. Display the import task of the specified db, specify the label as “load_example_db_20140102”, the state as “loading”, and sort by LoadStartTime in descending order

```
SHOW LOAD FROM example_db WHERE LABEL = "load_example_db_20140102" AND STATE = "loading" ORDER BY LoadStartTime DESC;
```

5. Display the import tasks of the specified db and sort them in descending order by LoadStartTime, and display 10 query results starting from offset 5

```
SHOW LOAD FROM example_db ORDER BY LoadStartTime DESC limit 5,10;
SHOW LOAD FROM example_db ORDER BY LoadStartTime DESC limit 10 offset 5;
```

6. Small batch import is a command to check the import status

```
curl --location-trusted -u {user}:{passwd} http://{hostname}:{port}/api/{database}/_load_info?label={labelname}
```

Keywords

```
SHOW, LOAD
```

Best Practice

SHOW-TABLES

SHOW-TABLES

Name

SHOW TABLES

Description

This statement is used to display all tables under the current db

grammar:

```
SHOW [FULL] TABLES [LIKE]
```

illustrate:

1. LIKE: Fuzzy query can be performed according to the table name

Example

1. View all tables under DB

```
mysql> show tables;
+---------------------------------+
| Tables_in_demo                  |
+---------------------------------+
| ads_client_biz_aggr_di_20220419 |
| cmy1                            |
| cmy2                            |
| intern_theme                    |
| left_table                      |
+---------------------------------+
5 rows in set (0.00 sec)
```

2. Fuzzy query by table name

```
mysql> show tables like '%cm%';
+----------------+
| Tables_in_demo |
+----------------+
| cmy1           |
| cmy2           |
+----------------+
2 rows in set (0.00 sec)
```

Keywords

```
SHOW, TABLES
```

Best Practice

SHOW-RESOURCES

SHOW-RESOURCES

Name

SHOW RESOURCES

Description

This statement is used to display resources that the user has permission to use. Ordinary users can only display resources with permission, and root or admin users will display all resources.

grammar:

```
SHOW RESOURCES
[
  WHERE
  [NAME [ = "your_resource_name" | LIKE "name_matcher"]]
  [RESOURCETYPE = ["SPARK"]]
]
[ORDER BY...]
[LIMIT limit][OFFSET offset];
```

illustrate:

1. If NAME LIKE is used, it will match Resource whose Name contains name_matcher in RESOURCES
2. If NAME = is used, it will match the specified Name exactly
3. If RESOURCETYPE is specified, match the corresponding Resrouce type
4. You can use ORDER BY to sort on any combination of columns
5. If LIMIT is specified, limit matching records are displayed. Otherwise show all
6. If OFFSET is specified, the query results are displayed starting at offset offset. By default the offset is 0.

Example

1. Display all resources that the current user has permissions to

   ```
   SHOW RESOURCES;
   ```

2. Display the specified Resource, the name contains the string “20140102”, and display 10 attributes

   ```
   SHOW RESOURCES WHERE NAME LIKE "2014_01_02" LIMIT 10;
   ```

3. Display the specified Resource, specify the name as “20140102” and sort by KEY in descending order

   ```
   SHOW RESOURCES WHERE NAME = "20140102" ORDER BY `KEY` DESC;
   ```

Keywords

```
SHOW, RESOURCES
```

Best Practice

SHOW-workload-GROUPS

SHOW-workload-GROUPS

Name

SHOW workload GROUPS

Description

This statement is used to display the resource groups for which the current user has usage_priv privileges.

grammar:

```
SHOW workload GROUPS;
```

Description:

This statement only does a simple display of workload groups, for a more complex display refer to tvf workload_groups().

Example

1. Show all workload groups:

```
mysql> show workload groups;
+----------+--------+--------------------------+---------+
| Id       | Name   | Item                     | Value   |
+----------+--------+--------------------------+---------+
| 10343386 | normal | cpu_share                | 10      |
| 10343386 | normal | memory_limit             | 30%     |
| 10343386 | normal | enable_memory_overcommit | true    |
| 10352416 | g1     | memory_limit             | 20%     |
| 10352416 | g1     | cpu_share                | 10      |
+----------+--------+--------------------------+---------+
```

Keywords

```
SHOW, workload, GROUPS, GROUP
```

Best Practice

SHOW-PARTITIONS

SHOW-PARTITIONS

Name

SHOW PARTITIONS

Description

This statement is used to display partition information for tables in Internal catalog or Hive Catalog

grammar:

```
SHOW [TEMPORARY] PARTITIONS FROM [db_name.]table_name [WHERE] [ORDER BY] [LIMIT];
```

illustrate:

When used in Internal catalog: 1. Support the filtering of PartitionId, PartitionName, State, Buckets, ReplicationNum, LastConsistencyCheckTime and other columns 2. TEMPORARY specifies to list temporary partitions

Please note that: 1. Currently, the where clause only supports the `=`, `!=`, `like` operator for string type PartitionName, State. For the other columns only support operators such as `=`, `!=`, `>`, `<`, `>=`, `<=`. 2. When using the above operators in the where clause, the column name needs to be on the left side. 3. Where clause can contains `AND`.

when used in Hive Catalog: Will return all partitions' name. Support multilevel partition table

Example

1. Display all non-temporary partition information of the specified table under the specified db

```
SHOW PARTITIONS FROM example_db.table_name;
```

2. Display all temporary partition information of the specified table under the specified db

   ```
   SHOW TEMPORARY PARTITIONS FROM example_db.table_name;
   ```

3. Display the information of the specified non-temporary partition of the specified table under the specified db

   ```
   SHOW PARTITIONS FROM example_db.table_name WHERE PartitionName = "p1";
   ```

4. Display the latest non-temporary partition information of the specified table under the specified db

```
SHOW PARTITIONS FROM example_db.table_name ORDER BY PartitionId DESC LIMIT 1;
```

Keywords

```
SHOW, PARTITIONS
```

Best Practice

SHOW-FRONTENDS

SHOW-FRONTENDS

Name

SHOW FRONTENDS

Description

This statement is used to view FE nodes

grammar:

```
SHOW FRONTENDS;
```

illustrate:

1. name indicates the name of the FE node in bdbje.
2. If Join is true, it means that the node has joined the cluster before. But it does not mean that it is still in the cluster (may have lost contact)
3. Alive indicates whether the node is alive or not.
4. ReplayedJournalId indicates the maximum metadata journal id that has been replayed by the node.
5. LastHeartbeat is the last heartbeat.
6. IsHelper indicates whether the node is a helper node in bdbje.
7. ErrMsg is used to display the error message when the heartbeat fails.
8. CurrentConnected indicates whether the FE node is currently connected

Example

Keywords

```
SHOW, FRONTENDS
```

Best Practice

SHOW FRONTENDS DISKS

Description

This statement is used to view the disk information of important directories on the FE node, such as metadata, logs, audit logs, and temporary directories.

Syntax

```
SHOW FRONTENDS DISKS;
```

Return Value

Column	Note
Name	The name of the FE node in bdbje
Host	The IP address of the FE node.
DirType	The types of directories to be displayed include the following four categories: meta, log, audit-log, temp, and deploy.
Dir	The directories for the types of directories to be displayed.
FileSystem	The file system in the Linux system where the types of directories to be displayed are located.
Capacity	The capacity of the file system.
Used	The used size of the file system.
Available	The remaining capacity of the file system.
UseRate	The usage percentage of the file system capacity.
MountOn	The mount directory of the file system.

Access Control Requirements

The user who executes this SQL must have at least the following permissions:

Privilege	Object	Notes
ADMIN_PRIV or NODE_PRIV		

Usage Notes

If further filtering of the query results is required, the table-valued function frontends_disks() can be used. SHOW BACKENDS is equivalent to the following statement:

```
SELECT * FROM FRONTENDS_DISKS();
```

Examples

```
SHOW FRONTENDS DISKS;
```

```
+----------------------------------------+-------------+-----------+--------------------------------+------------+----------+-----
-+-----------+---------+------------+
| Name                                   | Host        | DirType   | Dir                            | Filesystem | Capacity | Used
| Available | UseRate | MountOn    |
+----------------------------------------+-------------+-----------+--------------------------------+------------+----------+-----
-+-----------+---------+------------+
| fe_a1daac68_5ec0_477c_b5e8_f90a33cdc1bb | 10.xx.xx.90 | meta      | /home/disk/output/fe/doris-meta | /dev/sdf1  | 7T       | 2T
| 4T        | 36%     | /home/disk |
| fe_a1daac68_5ec0_477c_b5e8_f90a33cdc1bb | 10.xx.xx.90 | log       | /home/disk/output/fe/log        | /dev/sdf1  | 7T       | 2T
| 4T        | 36%     | /home/disk |
| fe_a1daac68_5ec0_477c_b5e8_f90a33cdc1bb | 10.xx.xx.90 | audit-log | /home/disk/output/fe/log        | /dev/sdf1  | 7T       | 2T
| 4T        | 36%     | /home/disk |
| fe_a1daac68_5ec0_477c_b5e8_f90a33cdc1bb | 10.xx.xx.90 | temp      | /home/disk/output/fe/temp_dir   | /dev/sdf1  | 7T       | 2T
| 4T        | 36%     | /home/disk |
| fe_a1daac68_5ec0_477c_b5e8_f90a33cdc1bb | 10.xx.xx.90 | deploy    | /home/disk/output/fe            | /dev/sdf1  | 7T       | 2T
| 4T        | 36%     | /home/disk |
+----------------------------------------+-------------+-----------+--------------------------------+------------+----------+-----
-+-----------+---------+------------+
```

SHOW-RESTORE

SHOW-RESTORE

Name

SHOW RESTORE

Description

This statement is used to view RESTORE tasks

grammar:

```
SHOW [BRIEF] RESTORE [FROM DB_NAME]
```

illustrate: 1. Only the most recent RESTORE task is saved in Doris. 2. The meaning of each column is as follows: JobId: Unique job id Label: The name of the backup to restore Timestamp: The time version of the backup to restore DbName: belongs to the database State: current stage PENDING: The initial state after submitting the job SNAPSHOTING: Executing snapshot DOWNLOAD: The snapshot is complete, ready to download the snapshot in the repository DOWNLOADING: Snapshot downloading COMMIT: Snapshot download is complete, ready to take effect COMMITING: in effect FINISHED: The job was successful CANCELLED: Job failed AllowLoad: Whether to allow import when restoring (currently not supported) ReplicationNum: Specifies the number of replicas to restore RestoreJobs: Tables and partitions to restore CreateTime: task submission time MetaPreparedTime: Metadata preparation completion time SnapshotFinishedTime: Snapshot completion time DownloadFinishedTime: Snapshot download completion time FinishedTime: Job finish time UnfinishedTasks: Displays unfinished subtask ids during SNAPSHOTING, DOWNLOADING and COMMITING stages Status: If the job fails, display the failure message Timeout: Job timeout, in seconds

```
    2. brief: only show key information of RESTORE tasks, columns RestoreObjs, Progress, TaskErrMsg will not show
```

Example

1. View the latest RESTORE task under example_db.

    ```
    SHOW RESTORE FROM example_db;
    ```

Keywords

```
SHOW, RESTORE
```

Best Practice

SHOW-PROPERTY

SHOW-PROPERTY

Description

This statement is used to view the attributes of the user

```
SHOW PROPERTY [FOR user] [LIKE key]
SHOW ALL PROPERTIES [LIKE key]
```

- `user`

  View the attributes of the specified user. If not specified, check the current user's.

- `LIKE`

  Fuzzy matching can be done by attribute name.

- `ALL`

  View the properties of all users (supported since version 2.0.3)

Return result description:

```
mysql> show property like'%connection%';
+----------------------+-------+
| Key                  | Value |
+----------------------+-------+
| max_user_connections | 100   |
+----------------------+-------+
1 row in set (0.01 sec)
```

- `Key`

  Property name.

- `Value`

  Attribute value.

```
mysql> show all properties like "%connection%";
+-------------------+--------------------------------------+
| User              | Properties                           |
+-------------------+--------------------------------------+
| root              | {"max_user_connections": "100"}      |
| admin             | {"max_user_connections": "100"}      |
| default_cluster:a | {"max_user_connections": "1000"}     |
+-------------------+--------------------------------------+
```

- `User`

username.

- `Properties`

  Key: value corresponding to each property of the user.

Example

1. View the attributes of the jack user

   ```
   SHOW PROPERTY FOR'jack';
   ```

2. View the attribute of jack user connection limit

   ```
   SHOW PROPERTY FOR'jack' LIKE'%connection%';
   ```

3. View all users importing cluster related properties

   ```
   SHOW ALL PROPERTIES LIKE '%load_cluster%'
   ```

Keywords

```
SHOW, PROPERTY, ALL
```

Best Practice

SHOW-TRIGGERS

SHOW-TRIGGERS

Description

Example

Keywords

```
SHOW, TRIGGERS
```

Best Practice

SHOW-PROCESSLIST

SHOW-PROCESSLIST

Name

SHOW PROCESSLIST

Description

Display the running threads of the user. It should be noted that except the root user who can see all running threads, other users can only see their own running threads, and cannot see the running threads of other users.

grammar:

```
SHOW [FULL] PROCESSLIST
```

illustrate:

- CurrentConnected: Indicates whether the connection is currently connected
- Id: It is the unique identifier of this thread. When we find that there is a problem with this thread, we can use the kill command to add this Id value to kill this thread. Earlier we said that the information displayed by show processlist comes from the information_schema.processlist table, so this Id is the primary key of this table.
- User: refers to the user who started this thread.
- Host: Records the IP and port number of the client sending the request. Through this information, when troubleshooting the problem, we can locate which client and which process sent the request.
- LoginTime: Timestamp when the connection is make.
- Catalog: Catalog name.
- Db: which database the currently executed command is on. If no database is specified, the value is NULL .
- Command: refers to the command that the thread is executing at the moment.
- Time: Indicates the time the thread is in the current state.
- State: The state of the thread, corresponding to Command.
- QueryId: The ID of the current query statement.
- Info: Generally recorded is the statement executed by the thread. By default, only the first 100 characters are displayed, that is, the statement you see may be truncated. To see all the information, you need to use show full processlist.

Common Command types are as follows：

- Query: The thread is executing a statement
- Sleep: is waiting for a client to send it an execute statement
- Quit: the thread is exiting
- Kill : The kill statement is being executed to kill the specified thread

Other types can refer to MySQL official website for explanation

Example

1. View the threads running by the current user

```
SHOW PROCESSLIST
```

return

```
MySQL [test]> show full processlist;
+------------------+------+------+-----------------+---------------------+----------+------+---------+------+-------+------
----------------------------+-----------------------+
| CurrentConnected | Id   | User | Host            | LoginTime           | Catalog  | Db   | Command | Time | State |
QueryId                          | Info                  |
+------------------+------+------+-----------------+---------------------+----------+------+---------+------+-------+------
----------------------------+-----------------------+
| Yes              |    0 | root | 127.0.0.1:34650 | 2023-09-06 12:01:02 | internal | test | Query   |    0 | OK    |
c84e397193a54fe7-bbe9bc219318b75e | select 1              |
|                  |    1 | root | 127.0.0.1:34776 | 2023-09-06 12:01:07 | internal |      | Sleep   |   29 | EOF   |
886ffe2894314f50-8dd73a6ca06699e4 | show full processlist |
+------------------+------+------+-----------------+---------------------+----------+------+---------+------+-------+------
----------------------------+-----------------------+
```

Keywords

```
SHOW, PROCESSLIST
```

Best Practice

SHOW TRASH

Description

This statement is used to view the space occupied by garbage data in backend.

Syntax

```
SHOW TRASH [ON ("<be_host>:<be_heartbeat_port>" [, ...])];
```

Optional Parameters

1. `[ON ("<be_host>:<be_heartbeat_port>" [, ...])]`

Specify the backend that you want to view. If you do not add ON, all backend is displayed by default.

Access Control Requirements

Users executing this SQL command must have at least the following privileges:

Privilege	Object	Notes
ADMIN_PRIV or NODE_PRIV	User or Role	Only users or roles with the ADMIN_PRIV or NODE_PRIV privilege can perform the SHOW TRASH operation.

Examples

1. View the garbage data usage of all be nodes.

```
SHOW TRASH;
```

2. View the junk data usage of '192.168.0.1:9050' (disk information is displayed).

```
SHOW TRASH ON "192.168.0.1:9050";
```

SHOW-VIEW

SHOW-VIEW

Name

SHOW VIEW

Description

This statement is used to display all views based on the given table

grammar:

```
SHOW VIEW { FROM | IN } table [ FROM db ]
```

Example

1. Show all views created based on table testTbl

   ```
   SHOW VIEW FROM testTbl;
   ```

Keywords

```
SHOW, VIEW
```

Best Practice

SHOW-TRANSACTION

SHOW-TRANSACTION

Name

SHOW TRANSACTION

Description

This syntax is used to view transaction details for the specified transaction id or label.

grammar:

```
SHOW TRANSACTION
[FROM db_name]
WHERE
[id=transaction_id]
[label = label_name];
```

Example of returned result:

```
     TransactionId: 4005
             Label: insert_8d807d5d-bcdd-46eb-be6d-3fa87aa4952d
       Coordinator: FE: 10.74.167.16
 TransactionStatus: VISIBLE
 LoadJobSourceType: INSERT_STREAMING
       PrepareTime: 2020-01-09 14:59:07
        CommitTime: 2020-01-09 14:59:09
        FinishTime: 2020-01-09 14:59:09
            Reason:
ErrorReplicasCount: 0
        ListenerId: -1
         TimeoutMs: 300000
```

- TransactionId: transaction id
- Label: the label corresponding to the import task
- Coordinator: The node responsible for transaction coordination
- TransactionStatus: transaction status
    - PREPARE: preparation stage
    - COMMITTED: The transaction succeeded, but the data was not visible
    - VISIBLE: The transaction succeeded and the data is visible
    - ABORTED: Transaction failed
- LoadJobSourceType: Type of import job.
- PrepareTime: transaction start time
- CommitTime: The time when the transaction was successfully committed
- FinishTime: The time when the data is visible
- Reason: error message
- ErrorReplicasCount: The number of replicas with errors

- ListenerId: The id of the related import job
- TimeoutMs: Transaction timeout, in milliseconds

Example

1. View the transaction with id 4005:

```
SHOW TRANSACTION WHERE ID=4005;
```

2. In the specified db, view the transaction with id 4005:

```
SHOW TRANSACTION FROM db WHERE ID=4005;
```

3. View the transaction whose label is label_name:

```
SHOW TRANSACTION WHERE LABEL = 'label_name';
```

Keywords

```
SHOW, TRANSACTION
```

Best Practice

SHOW-STREAM-LOAD

SHOW-STREAM-LOAD

Name

SHOW STREAM LOAD

Description

This statement is used to display the execution of the specified Stream Load task

grammar:

```
SHOW STREAM LOAD
[FROM db_name]
[
  WHERE
  [LABEL [ = "your_label" | LIKE "label_matcher"]]
  [STATUS = ["SUCCESS"|"FAIL"]]
]
[ORDER BY...]
[LIMIT limit][OFFSET offset];
```

illustrate:

1. By default, BE does not record Stream Load records. If you want to view records that need to be enabled on BE, the configuration parameter is: `enable_stream_load_record=true`. For details, please refer to BE Configuration Items
2. If db_name is not specified, the current default db is used
3. If LABEL LIKE is used, it will match the tasks whose label of the Stream Load task contains label_matcher
4. If LABEL = is used, it will match the specified label exactly
5. If STATUS is specified, matches STREAM LOAD status
6. You can use ORDER BY to sort on any combination of columns
7. If LIMIT is specified, limit matching records are displayed. Otherwise show all
8. If OFFSET is specified, the query results are displayed starting at offset offset. By default the offset is 0.

Example

1. Show all Stream Load tasks of the default db

   ```
   SHOW STREAM LOAD;
   ```

2. Display the Stream Load task of the specified db, the label contains the string “2014_01_02”, and display the oldest 10

```
SHOW STREAM LOAD FROM example_db WHERE LABEL LIKE "2014_01_02" LIMIT 10;
```

3. Display the Stream Load task of the specified db and specify the label as “load_example_db_20140102”

```
SHOW STREAM LOAD FROM example_db WHERE LABEL = "load_example_db_20140102";
```

4. Display the Stream Load task of the specified db, specify the status as “success”, and sort by StartTime in descending order

```
SHOW STREAM LOAD FROM example_db WHERE STATUS = "success" ORDER BY StartTime DESC;
```

5. Display the import tasks of the specified db and sort them in descending order of StartTime, and display 10 query results starting from offset 5

```
SHOW STREAM LOAD FROM example_db ORDER BY StartTime DESC limit 5,10;
SHOW STREAM LOAD FROM example_db ORDER BY StartTime DESC limit 10 offset 5;
```

Keywords

```
SHOW, STREAM, LOAD
```

Best Practice

SHOW-STATUS

SHOW-STATUS

Name

SHOW STATUS

Description

This command is used to view the execution of the Create Materialized View job submitted through the CREATE-MATERIALIZED-VIEW statement.

> This statement is equivalent to `SHOW ALTER TABLE ROLLUP;`

```
SHOW ALTER TABLE MATERIALIZED VIEW
[FROM database]
[WHERE]
[ORDER BY]
[LIMIT OFFSET]
```

- database: View jobs under the specified database. If not specified, the current database is used.
- WHERE: You can filter the result column, currently only the following columns are supported:
    - TableName: Only equal value filtering is supported.
    - State: Only supports equivalent filtering.
    - Createtime/FinishTime: Support =, >=, <=, >, <, !=
- ORDER BY: The result set can be sorted by any column.
- LIMIT: Use ORDER BY to perform page-turning query.

Return result description:

```
mysql> show alter table materialized view\G
*************************** 1. row ******************** ******
          JobId: 11001
      TableName: tbl1
     CreateTime: 2020-12-23 10:41:00
     FinishTime: NULL
  BaseIndexName: tbl1
RollupIndexName: r1
       RollupId: 11002
  TransactionId: 5070
          State: WAITING_TXN
            Msg:
       Progress: NULL
        Timeout: 86400
1 row in set (0.00 sec)
```

- `JobId`: Job unique ID.

- `TableName`: base table name
- `CreateTime/FinishTime`: Job creation time and end time.
- `BaseIndexName/RollupIndexName`: Base table name and materialized view name.
- `RollupId`: The unique ID of the materialized view.
- `TransactionId`: See the description of the State field.
- `State`: job status.
    - PENDING: The job is in preparation.
    - WAITING_TXN:

      Before officially starting to generate materialized view data, it will wait for the current running import transaction on this table to complete. And the `TransactionId` field is the current waiting transaction ID. When all previous imports for this ID are complete, the job will actually start.
    - RUNNING: The job is running.
    - FINISHED: The job ran successfully.
    - CANCELLED: The job failed to run.
- `Msg`: error message
- `Progress`: job progress. The progress here means `completed tablets/total tablets`. Materialized views are created at tablet granularity.
- `Timeout`: Job timeout, in seconds.

Example

Keywords

```
SHOW, STATUS
```

Best Practice

SHOW-TABLE-ID

SHOW-TABLE-ID

Name

SHOW TABLE ID

Description

This statement is used to find the corresponding database name, table name according to the table id (only for administrators)

grammar:

```
SHOW TABLE [table_id]
```

Example

1. Find the corresponding database name, table name according to the table id

   ```
   SHOW TABLE 10001;
   ```

Keywords

```
SHOW, TABLE, ID
```

Best Practice

SHOW-SMALL-FILES

SHOW-SMALL-FILES

Name

SHOW FILE

Description

This statement is used to display files created by the CREATE FILE command within a database.

```
SHOW FILE [FROM database];
```

Return result description:

- FileId: file ID, globally unique
- DbName: the name of the database to which it belongs
- Catalog: Custom Category
- FileName: file name
- FileSize: file size in bytes
- MD5: MD5 of the file

Example

1. View the uploaded files in the database my_database

   ```
   SHOW FILE FROM my_database;
   ```

Keywords

```
SHOW, SMALL, FILES
```

Best Practice

SHOW-POLICY

SHOW-POLICY

Name

SHOW ROW POLICY

Description

View the row security policy under the current DB

```
SHOW ROW POLICY [FOR user| ROLE role]
```

Example

1. view all security policies.

```
mysql> SHOW ROW POLICY;
+------------------+----------------------+-----------+------+------------+------------------+------+------------------
-------------------------------------------------------------------------------------------------------------------------+
| PolicyName       | DbName               | TableName | Type | FilterType | WherePredicate   | User | OriginStmt
|
+------------------+----------------------+-----------+------+------------+------------------+------+------------------
-------------------------------------------------------------------------------------------------------------------------+
| test_row_policy_1 | default_cluster:test | table1    | ROW  | RESTRICTIVE | `id` IN (1, 2)    | root | /*
ApplicationName=DataGrip 2021.3.4 */ CREATE ROW POLICY test_row_policy_1 ON test.table1 AS RESTRICTIVE TO root USING (id in
(1, 2));
|
| test_row_policy_2 | default_cluster:test | table1    | ROW  | RESTRICTIVE | `col1` = 'col1_1' | root | /*
ApplicationName=DataGrip 2021.3.4 */ CREATE ROW POLICY test_row_policy_2 ON test.table1 AS RESTRICTIVE TO root USING
(col1='col1_1');
|
+------------------+----------------------+-----------+------+------------+------------------+------+------------------
-------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

2. specify user name query

```
mysql> SHOW ROW POLICY FOR test;
+------------------+----------------------+-----------+------+------------+------------------+----------------------+----
-------------------------------------------------------------------------------------------------------------------------
-----------+
| PolicyName       | DbName               | TableName | Type | FilterType | WherePredicate   | User                 |
OriginStmt
|
+------------------+----------------------+-----------+------+------------+------------------+----------------------+----
-------------------------------------------------------------------------------------------------------------------------
-----------+
| test_row_policy_3 | default_cluster:test | table1    | ROW  | PERMISSIVE | `col1` = 'col1_2' | default_cluster:test | /*
ApplicationName=DataGrip 2021.3.4 */ CREATE ROW POLICY test_row_policy_3 ON test.table1 AS PERMISSIVE TO test USING
(col1='col1_2');
```

```
|
+------------------+---------------------+----------+------+-----------+-----------------+--------------------+----
----------------------------------------------------------------------------------------------------------------------
----------+
1 row in set (0.01 sec)
```

3. specify role name query

```
mysql> SHOW ROW POLICY for role role1;
+------------+--------+-----------+------+-------------+---------------+------+-------+----------------------------------
-----------------------------------------------+
| PolicyName | DbName | TableName | Type | FilterType  | WherePredicate | User | Role  | OriginStmt
|
+------------+--------+-----------+------+-------------+---------------+------+-------+----------------------------------
-----------------------------------------------+
| zdtest1    | zd     | user      | ROW  | RESTRICTIVE | `user_id` = 1  | NULL | role1 | create row policy zdtest1 on user
as restrictive to role role1 using (user_id=1) |
+------------+--------+-----------+------+-------------+---------------+------+-------+----------------------------------
-----------------------------------------------+
1 row in set (0.01 sec)
```

4. demonstrate data migration strategies `sql mysql> SHOW STORAGE POLICY; +----------------------+---------
+-----------------------+---------------------+-------------+--
----

---+ | PolicyName | Type |
StorageResource | CooldownDatetime | CooldownTtl | properties | +----------------------+---------+-----
------------------+---------------------+-------------+--

---+ | showPolicy_1_policy | STORAGE |
showPolicy_1_resource | 2022-06-08 00:00:00 | -1 | { "type": "s3", "s3.endpoint": "bj.s3.comaaaa",
"s3.region": "bj", "s3.access_key": "bbba", "s3.secret_key": "******", "s3.root.path":
"path/to/rootaaaa", "s3.bucket": "test-bucket", "s3.connection.request.timeout": "3000"
"3.connection.maximum": "50", "s3.connection.timeout": "1000", } | +----------------------+---------+--
----------------------+---------------------+-------------+--
----

---+ 1 row in set (0.00 sec)`

Keywords

```
SHOW, POLICY
```

Best Practice

SHOW-STORAGE-POLICY-USING

SHOW-STORAGE-POLICY-USING

Name

SHOW STORAGE POLICY USING

Description

This command is used to show tables and partitions which is using storage policy

```
SHOW STORAGE POLICY USING [FOR some_policy]
```

Example

1. get all objects which are using storage policy

```
mysql> show storage policy using;
+----------------------+----------------------------------------+---------------------------------------+------------+
| PolicyName           | Database                               | Table                                 | Partitions |
+----------------------+----------------------------------------+---------------------------------------+------------+
| test_storage_policy   | regression_test_cold_heat_separation_p2 | table_with_storage_policy_1           | ALL        |
| test_storage_policy   | regression_test_cold_heat_separation_p2 | partition_with_multiple_storage_policy | p201701    |
| test_storage_policy_2 | regression_test_cold_heat_separation_p2 | partition_with_multiple_storage_policy | p201702    |
| test_storage_policy_2 | regression_test_cold_heat_separation_p2 | table_with_storage_policy_2           | ALL        |
| test_policy          | db2                                    | db2_test_1                            | ALL        |
+----------------------+----------------------------------------+---------------------------------------+------------+
```

2. get objects which are using the storage policy named test_storage_policy

```
mysql> show storage policy using for test_storage_policy;
+--------------------+-----------+-------------------------------+------------+
| PolicyName         | Database  | Table                         | Partitions |
+--------------------+-----------+-------------------------------+------------+
| test_storage_policy | db_1      | partition_with_storage_policy_1 | p201701    |
| test_storage_policy | db_1      | table_with_storage_policy_1     | ALL        |
+--------------------+-----------+-------------------------------+------------+
```

Keywords

```
SHOW, STORAGE, POLICY, USING
```

Best Practice

SHOW-CATALOG-RECYCLE-BIN

SHOW-CATALOG-RECYCLE-BIN

Name

SHOW CATALOG RECYCLE BIN

Description

This statement is used to display the dropped meta informations that can be recovered

grammar:

```
SHOW CATALOG RECYCLE BIN [ WHERE NAME [ = "name" | LIKE "name_matcher"] ]
```

grammar:

```
The meaning of each column is as follows:
        Type:                type of meta information:Database、Table、Partition
        Name:                name of meta information
        DbId:                id of database
        TableId:             id of table
        PartitionId:         id of partition
        DropTime:            drop time of meta information
```

Example

1. Display all meta informations that can be recovered

   ```
   SHOW CATALOG RECYCLE BIN;
   ```

2. Display meta informations with name ‘test’

   ```
   SHOW CATALOG RECYCLE BIN WHERE NAME = 'test';
   ```

Keywords

```
SHOW, CATALOG RECYCLE BIN
```

Best Practice

SHOW-QUERY-STATS

SHOW-QUERY-STATS

Name

SHOW QUERY STATS

Description

This statement is used to show the query hit statistics of the database and table

grammar:

```
SHOW QUERY STATS [[FOR db_name]|[FROM table_name]] [ALL] [VERBOSE];
```

Remarks：

1. Support query database and table history query hit statistics, restart fe after data will be reset, each fe separately statistics
2. Use FOR DATABASE and FROM TABLE to specify the query database or table hit statistics, respectively followed by the database name or table name
3. ALL can specify whether to display all index query hit statistics, VERBOSE can display more detailed hit statistics, these two parameters can be used separately, but must be placed at the end and can only be used on table queries
4. If no database is used, execute `SHOW QUERY STATS` directly to display the hit statistics of all databases
5. The result may have two columns: QueryCount: The number of times the column was queried FilterCount: The number of times the column was queried as a where condition #### Example
6. Show the query hit statistics for `baseall`

```
MySQL [test_query_db]> show query stats from baseall;
+-------+------------+-------------+
| Field | QueryCount | FilterCount |
+-------+------------+-------------+
| k0    | 0          | 0           |
| k1    | 0          | 0           |
| k2    | 0          | 0           |
| k3    | 0          | 0           |
| k4    | 0          | 0           |
| k5    | 0          | 0           |
| k6    | 0          | 0           |
| k10   | 0          | 0           |
| k11   | 0          | 0           |
| k7    | 0          | 0           |
| k8    | 0          | 0           |
| k9    | 0          | 0           |
```

```
| k12   | 0          | 0           |
| k13   | 0          | 0           |
+-------+------------+-------------+
14 rows in set (0.002 sec)

MySQL [test_query_db]> select k0, k1,k2, sum(k3) from baseall  where k9 > 1 group by k0,k1,k2;
+------+------+--------+-------------+
| k0   | k1   | k2     | sum(`k3`)   |
+------+------+--------+-------------+
|    0 |    6 |  32767 |        3021 |
|    1 |   12 |  32767 | -2147483647 |
|    0 |    3 |   1989 |        1002 |
|    0 |    7 | -32767 |        1002 |
|    1 |    8 |    255 |  2147483647 |
|    1 |    9 |   1991 | -2147483647 |
|    1 |   11 |   1989 |       25699 |
|    1 |   13 | -32767 |  2147483647 |
|    1 |   14 |    255 |         103 |
|    0 |    1 |   1989 |        1001 |
|    0 |    2 |   1986 |        1001 |
|    1 |   15 |   1992 |        3021 |
+------+------+--------+-------------+
12 rows in set (0.050 sec)

MySQL [test_query_db]> show query stats from baseall;
+-------+------------+-------------+
| Field | QueryCount | FilterCount |
+-------+------------+-------------+
| k0    | 1          | 0           |
| k1    | 1          | 0           |
| k2    | 1          | 0           |
| k3    | 1          | 0           |
| k4    | 0          | 0           |
| k5    | 0          | 0           |
| k6    | 0          | 0           |
| k10   | 0          | 0           |
| k11   | 0          | 0           |
| k7    | 0          | 0           |
| k8    | 0          | 0           |
| k9    | 1          | 1           |
| k12   | 0          | 0           |
| k13   | 0          | 0           |
+-------+------------+-------------+
14 rows in set (0.001 sec)
```

7. Show the query hit statistics summary for all the mv in a table

```
MySQL [test_query_db]> show query stats from baseall all;
+-----------+------------+
| IndexName | QueryCount |
+-----------+------------+
| baseall   | 1          |
+-----------+------------+
1 row in set (0.005 sec)
```

8. Show the query hit statistics detail info for all the mv in a table

```
MySQL [test_query_db]> show query stats from baseall all verbose;
+-----------+-------+------------+-------------+
| IndexName | Field | QueryCount | FilterCount |
+-----------+-------+------------+-------------+
| baseall   | k0    | 1          | 0           |
```

```
|           | k1    | 1          | 0           |
|           | k2    | 1          | 0           |
|           | k3    | 1          | 0           |
|           | k4    | 0          | 0           |
|           | k5    | 0          | 0           |
|           | k6    | 0          | 0           |
|           | k10   | 0          | 0           |
|           | k11   | 0          | 0           |
|           | k7    | 0          | 0           |
|           | k8    | 0          | 0           |
|           | k9    | 1          | 1           |
|           | k12   | 0          | 0           |
|           | k13   | 0          | 0           |
+-----------+-------+------------+-------------+
14 rows in set (0.017 sec)
```

9. Show the query hit for a database

```
MySQL [test_query_db]> show query stats for test_query_db;
+---------------------------+------------+
| TableName                 | QueryCount |
+---------------------------+------------+
| compaction_tbl            | 0          |
| bigtable                  | 0          |
| empty                     | 0          |
| tempbaseall               | 0          |
| test                      | 0          |
| test_data_type            | 0          |
| test_string_function_field | 0          |
| baseall                   | 1          |
| nullable                  | 0          |
+---------------------------+------------+
9 rows in set (0.005 sec)
```

10. Show query hit statistics for all the databases

```
MySQL [(none)]> show query stats;
+-----------------+------------+
| Database        | QueryCount |
+-----------------+------------+
| test_query_db   | 1          |
+-----------------+------------+
1 rows in set (0.005 sec)
```

SHOW QUERY STATS; ```

Keywords

```
SHOW,  QUERY, STATS;
```

Best Practice

IN

IN

IN

description

Syntax

```
expr IN (value, ...)
```

```
expr IN (subquery)
```

If expr is equal to any value in the IN list, return true; otherwise, return false.

Subquery can only return one column, and the column types returned by subquery must be compatible with expr types.

If subquery returns a bitmap data type column, expr must be an integer.

example

```
mysql> select id from cost where id in (1, 2);
+------+
| id   |
+------+
|    2 |
|    1 |
+------+
```

```
mysql> select id from tbl1 where id in (select id from tbl2);
+------+
| id   |
+------+
|    1 |
|    4 |
|    5 |
+------+
```

```
mysql> select id from tbl1 where id in (select bitmap_col from tbl3);
+------+
| id   |
+------+
|    1 |
|    3 |
+------+
```

keywords

```
IN
```

HELP

HELP

Name

HELP

Description

The directory of help can be queried by changing the command

grammar:

```
HELP <item>
```

All Doris commands can be listed with `help`

```
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
? (\?) Synonym for `help'.
clear (\c) Clear the current input statement.
connect (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
edit (\e) Edit command with $EDITOR.
ego (\G) Send command to mysql server, display result vertically.
exit (\q) Exit mysql. Same as quit.
go (\g) Send command to mysql server.
help (\h) Display this help.
nopager (\n) Disable pager, print to stdout.
note (\t) Don't write into outfile.
pager (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print (\p) Print current command.
prompt (\R) Change your mysql prompt.
quit (\q) Quit mysql.
rehash (\#) Rebuild completion hash.
source (\.) Execute an SQL script file. Takes a file name as an argument.
status (\s) Get status information from the server.
system (\!) Execute a system shell command.
tee (\T) Set outfile [to_outfile]. Append everything into given outfile.
use (\u) Use another database. Takes database name as argument.
charset (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.

For server side help, type 'help contents'
```

Get the Doris SQL help contents via `help contents`

```
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following
```

```
categories:
   sql-functions
   sql-statements
```

Example

1. List all SQL help contents of Doris

   ```
   help contents
   ```

2. The command to list all function directories of the Doris cluster

   ```
   help sql-functions
   ```

3. List all functions under the date function

   ```
   help date-time-functions
   ```

Keywords

```
HELP
```

Best Practice

USE

USE

Name

USE

Description

The USE command allows us to use the database

grammar:

```
USE <[CATALOG_NAME].DATABASE_NAME>
```

illustrate: 1. `USE CATALOG_NAME.DATABASE_NAME` will switch the current catalog into `CATALOG_NAME` and then change the current database into `DATABASE_NAME`

Example

1. If the demo database exists in current catalog, try accessing it:

   ```
   mysql> use demo;
   Database changed
   ```

2. If the demo database exists in catalog hms_catalog, try switching the catalog and accessing it:

   ```
   mysql> use hms_catalog.demo;
   Database changed
   ```

Keywords

```
USE
```

Best Practice

DESCRIBE

DESCRIBE

Name

DESCRIBE

Description

This statement is used to display the schema information of the specified table

grammar:

```
DESC[RIBE] [db_name.]table_name [ALL];
```

illustrate:

1. If ALL is specified, the schemas of all indexes (rollup) of the table will be displayed

Example

1. Display the Base table schema

   ```
   DESC table_name;
   ```

2. Display the schema of all indexes of the table

   ```
   DESC db1.table_name ALL;
   ```

Keywords

```
DESCRIBE
```

Best Practice

SWITCH

SWITCH

Name

SWITCH

Description

This statement is used to switch catalog.

Syntax:

```
SWITCH catalog_name
```

Example

1. Switch to hive

    ```
    SWITCH hive;
    ```

Keywords

SWITCH, CATALOG

Best Practice

REFRESH

REFRESH

Name

REFRESH

Description

This statement refreshes the metadata of the specified Catalog/Database/Table.

syntax:

```
REFRESH CATALOG catalog_name;
REFRESH DATABASE [catalog_name.]database_name;
REFRESH TABLE [catalog_name.][database_name.]table_name;
```

When the Catalog is refreshed, the object-related Cache is forced to be invalidated.

Including Partition Cache, Schema Cache, File Cache, etc.

Example

1. Refresh hive catalog

   ```
   REFRESH CATALOG hive;
   ```

2. Refresh database1

   ```
   REFRESH DATABASE ctl.database1;
   REFRESH DATABASE database1;
   ```

3. Refresh table1

   ```
   REFRESH TABLE ctl.db.table1;
   REFRESH TABLE db.table1;
   REFRESH TABLE table1;
   ```

Keywords

REFRESH, CATALOG, DATABASE, TABLE

Best Practice

SYNC

Description

This statement is used to synchronize metadata for non-master Frontend (FE) nodes. In Apache Doris, only the master FE node can write metadata, while other FE nodes forward metadata write operations to the master. After the master completes the metadata writing operation, non-master nodes may experience a short delay in replaying the metadata. You can use this statement to force synchronization of metadata.

Syntax

```
SYNC;
```

Access Control Requirements

Any user or role can perform this operation.

Examples

Synchronize metadata:

````
```sql
SYNC;
```
````

CLEAN-QUERY-STATS

CLEAN-QUERY-STATS

Name

CLEAN QUERY STATS

Description

This statement is used to clear query statistics

grammar:

```
CLEAN [ALL| DATABASE | TABLE] QUERY STATS [[FOR db_name]|[FROM|IN] table_name]];
```

Remarks:

1. If ALL is specified, all query statistics are cleared, including database and table, admin privilege is needed
2. If DATABASE is specified, the query statistics of the specified database are cleared, alter privilege for this database is needed
3. If TABLE is specified, the query statistics of the specified table are cleared, alter privilege for this table is needed

Example

1. Clear all statistics
2. ```
   clean all query stats;
   ```
3. Clear the specified database statistics

   ```
   clean database query stats for test_query_db;
   ```
4. Clear the specified table statistics

   ```
   clean table query stats from test_query_db.baseall;
   ```

Keywords

```
CLEAN, QUERY, STATS
```

Best Practice
```

FE Configuration

FE Configuration

This document mainly introduces the relevant configuration items of FE.

The FE configuration file `fe.conf` is usually stored in the `conf/` directory of the FE deployment path. In version 0.14, another configuration file `fe_custom.conf` will be introduced. The configuration file is used to record the configuration items that are dynamically configured and persisted by the user during operation.

After the FE process is started, it will read the configuration items in `fe.conf` first, and then read the configuration items in `fe_custom.conf`. The configuration items in `fe_custom.conf` will overwrite the same configuration items in `fe.conf`.

The location of the `fe_custom.conf` file can be configured in `fe.conf` through the `custom_config_dir` configuration item.

Precautions

1. For the purpose of simplifying the architecture, modifying the configuration through the mysql protocol will only modify the data in the local FE memory, and will not synchronize the changes to all FEs. For Config items that only take effect on the Master FE, the modification request will be automatically forwarded to the Master FE.

2. Note that the option `forward_to_master` will affect the display results of `admin show frontend config`, if `forward_to_master=true`, `admin show frontend config` shows the Config of Master FE (Even if you are connecting to a Follower FE currently), this may cause you to be unable to see the modification of the local FE configuration; if you expect show config of the FE you're connecting, then execute the command `set forward_to_master=false`.

View configuration items

There are two ways to view the configuration items of FE:

1. FE web page

   Open the FE web page `http://fe_host:fe_http_port/Configure` in the browser. You can see the currently effective FE configuration items in `Configure Info`.

2. View by command

   After the FE is started, you can view the configuration items of the FE in the MySQL client with the following command:

   ```
   ADMIN SHOW FRONTEND CONFIG;
   ```

   The meanings of the columns in the results are as follows:

- Key: the name of the configuration item.
- Value: The value of the current configuration item.
- Type: The configuration item value type, such as integer or string.
- IsMutable: whether it can be dynamically configured. If true, the configuration item can be dynamically configured at runtime. If false, it means that the configuration item can only be configured in `fe.conf` and takes effect after restarting FE.
- MasterOnly: Whether it is a unique configuration item of Master FE node. If it is true, it means that the configuration item is meaningful only at the Master FE node, and is meaningless to other types of FE nodes. If false, it means that the configuration item is meaningful in all types of FE nodes.
- Comment: The description of the configuration item.

Set configuration items

There are two ways to configure FE configuration items:

1. Static configuration

   Add and set configuration items in the `conf/fe.conf` file. The configuration items in `fe.conf` will be read when the FE process starts. Configuration items not in `fe.conf` will use default values.

2. Dynamic configuration via MySQL protocol

   After the FE starts, you can set the configuration items dynamically through the following commands. This command requires administrator privilege.

   ```
   ADMIN SET FRONTEND CONFIG (" fe_config_name "=" fe_config_value ");
   ```

   Not all configuration items support dynamic configuration. You can check whether the dynamic configuration is supported by the `IsMutable` column in the`ADMIN SHOW FRONTEND CONFIG;`command result.

   If the configuration item of `MasterOnly` is modified, the command will be directly forwarded to the Master FE and only the corresponding configuration item in the Master FE will be modified.

   Configuration items modified in this way will become invalid after the FE process restarts.

   For more help on this command, you can view it through the `HELP ADMIN SET CONFIG;` command.

3. Dynamic configuration via HTTP protocol

   For details, please refer to Set Config Action

   This method can also persist the modified configuration items. The configuration items will be persisted in the `fe_custom.conf` file and will still take effect after FE is restarted.

Examples

1. Modify `async_pending_load_task_pool_size`

   Through `ADMIN SHOW FRONTEND CONFIG;` you can see that this configuration item cannot be dynamically configured (`IsMutable` is false). You need to add in `fe.conf`:

```
async_pending_load_task_pool_size = 20
```

Then restart the FE process to take effect the configuration.

2. Modify `dynamic_partition_enable`

   Through `ADMIN SHOW FRONTEND CONFIG;` you can see that the configuration item can be dynamically configured (`IsMutable` is true). And it is the unique configuration of Master FE. Then first we can connect to any FE and execute the following command to modify the configuration:

   ```
   ADMIN SET FRONTEND CONFIG ("dynamic_partition_enable" = "true"); `
   ```

   Afterwards, you can view the modified value with the following command:

   ```
   set forward_to_master = true;
   ADMIN SHOW FRONTEND CONFIG;
   ```

   After modification in the above manner, if the Master FE restarts or a Master election is performed, the configuration will be invalid. You can add the configuration item directly in `fe.conf` and restart the FE to make the configuration item permanent.

3. Modify `max_distribution_pruner_recursion_depth`

   Through `ADMIN SHOW FRONTEND CONFIG;` you can see that the configuration item can be dynamically configured (`IsMutable` is true). It is not unique to Master FE.

   Similarly, we can modify the configuration by dynamically modifying the configuration command. Because this configuration is not unique to the Master FE, user need to connect to different FEs separately to modify the configuration dynamically, so that all FEs use the modified configuration values.

Configurations

> Note:
>
> The following content is automatically generated by `docs/generate-config-and-variable-doc.sh`.
>
> If you need to modify, please modify the description information in `fe/fe-common/src/main/java/org/apache/doris/common/Config.java`.

<–DOC_PLACEHOLDER–>

CDC Load

Ecosystem Overview

Deploy Doris on Helm Chart

Helm is the best way to find, share, and use software built for Kubernetes. Helm Charts help to define, install, and upgrade even the most complex Kubernetes application. Helm Chart makes it easy to deploy Doris clusters and skip difficult configuration steps.

Add Helm repository of Doris-Operator

This Doris repository have resources about RBAC , deployment …etc for doris-operator running. 1. Add the Doris repository

```
$ helm repo add doris-repo https://charts.selectdb.com
```

2. Update the Chart to the latest version

```
$ helm repo update doris-repo
```

3. Check the Helm Chart Repo is the latest version

```
$ helm search repo doris-repo
NAME                          CHART VERSION    APP VERSION   DESCRIPTION
doris-repo/doris-operator     1.3.1            1.3.1         Doris-operator for doris creat ...
doris-repo/doris              1.3.1            2.0.3         Apache Doris is an easy-to-use ...
```

Install doris-operator

1. Install

- Install doris-operator， with default config in a namespace named `doris`

```
$ helm install operator doris-repo/doris-operator
```

- The repo defines the basic function for running doris-operator, Please use next command to deploy operator, when you have completed customization of values.yaml

```
$ helm install -f values.yaml operator doris-repo/doris-operator
```

2. Validate installation Status

Check the deployment status of Pods through the `kubectl get pods` command. Observe that the Pod of doris-operator is in the Running state and all containers in the Pod are ready, that means, the deployment is successful.

```
$ kubectl get pod --namespace doris
NAME                              READY   STATUS    RESTARTS   AGE
doris-operator-866bd449bb-zl5mr   1/1     Running   0          18m
```

Install doriscluster

1. Install

- Use default config for deploying doriscluster. This only deploys 3 FE and 3 BE components and using default `storageClass` for providing persistent volume.

```
$ helm install doriscluster doris-repo/doris
```

- Custom Doris deploying, specify resources or different deployment type, please customize the resource configuration according to the annotations of each resource configuration in values.yaml and use next command for deploying.

```
$ helm install -f values.yaml doriscluster doris-repo/doris
```

2. Validate installation Status

After executing the installation command, deployment and distribution, service deployment scheduling and startup will take a certain amount of time. Check the deployment status of Pods through the `kubectl get pods` command.

Observe that the Pod of `doriscluster` is in the `Running` state and all containers in the Pod are ready, that means, the deployment is successful.

```
$  kubectl get pod --namespace doris
NAME                     READY   STATUS    RESTARTS   AGE
doriscluster-helm-fe-0   1/1     Running   0          1m39s
doriscluster-helm-fe-1   1/1     Running   0          1m39s
doriscluster-helm-fe-2   1/1     Running   0          1m39s
doriscluster-helm-be-0   1/1     Running   0          16s
doriscluster-helm-be-1   1/1     Running   0          16s
doriscluster-helm-be-2   1/1     Running   0          16s
```

Uninstall Doris by Helm

Uninstall doriscluster

Please confirm the Doris is not used, when using next command to uninstall `doriscluster`.

```
$ helm uninstall doriscluster
```

Uninstall doris-operator

Please confirm that Doris is not running in Kubernetes, use next command to uninstall `doris-operator`.

```
$ helm uninstall operator
```

Access the Doris cluster

In Kubernetes, a Service is a resource that defines a set of pods and provides stable access to this set of pods, so it can be associated with a set of pods. Deploying a cluster through Doris-Operator will automatically generate corresponding Service resources according to the `spec.*Spec.service` configuration. Currently, ClusterIP, LoadBalancer and NodePort modes are supported. Support users' access needs in different scenarios.

Access within the Kubernetes cluster

ClusterIP mode

Doris provides `ClusterIP` access within the kubernetes cluster by default on kubernetes. For FE and BE components, we provide corresponding Service resources for users to use on demand on kubernetes. Use the following command to view the Service of the corresponding component. The Service naming rule provided by Doris-Operator is `{clusterName}-{fe/be}-service`.

```
$ kubectl -n {namespace} get service
```

During use, please replace {namespace} with the namespace specified during deployment. Take our default sample deployed Doris cluster as an example:

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  labels:
    app.kubernetes.io/name: doriscluster
    app.kubernetes.io/instance: doriscluster-sample
    app.kubernetes.io/part-of: doris-operator
  name: doriscluster-sample
spec:
  feSpec:
    replicas: 3
    limits:
      cpu: 6
      memory: 12Gi
    requests:
      cpu: 6
      memory: 12Gi
    image: selectdb/doris.fe-ubuntu:2.0.2
  beSpec:
    replicas: 3
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    image: selectdb/doris.be-ubuntu:2.0.2
```

We view `kubectl get service` through the command as follows:

```
$ kubectl get service
NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                               AGE
doriscluster-sample-be-internal   ClusterIP   None             <none>        9050/TCP                              12h
doriscluster-sample-be-service    ClusterIP   172.20.217.234   <none>        9060/TCP,8040/TCP,9050/TCP,8060/TCP   12h
doriscluster-sample-fe-internal   ClusterIP   None             <none>        9030/TCP                              12h
doriscluster-sample-fe-service    ClusterIP   172.20.183.136   <none>        8030/TCP,9020/TCP,9030/TCP,9010/TCP   12h
```

There are two types of services, FE and BE, displayed through the command. The service with the suffix `internal` is the service used by Doris for internal communication and is not available externally. The suffix `-service` is a Service for users to use. In this example, the `CLUSTER-IP` corresponding to `doriscluster-sample-fe-service` and the corresponding `PORT` can be used on the K8s cluster to access different port services of FE. Using `doriscluster-sample-be-service` Service And the corresponding `PORT` port to access BE's services.

Access outside the Kubernetes cluster

LoadBalancer Mode

If the cluster is created on a relevant cloud platform, it is recommended to use the `LoadBalancer` mode to access the FE and BE services within the cluster. The `ClusterIP` mode is used by default. If you want to use the `LoadBalancer` mode, please configure the following configuration in the spec of each component:

```
service:
  type: LoadBalancer
```

Taking the default configuration as a modification blueprint for example, we use `LoadBalancer` as the access mode of FE and BE on the cloud platform. The deployment configuration is as follows:

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  labels:
    app.kubernetes.io/name: doriscluster
    app.kubernetes.io/instance: doriscluster-sample
    app.kubernetes.io/part-of: doris-operator
  name: doriscluster-sample
spec:
  feSpec:
    replicas: 3
    service:
      type: LoadBalancer
    limits:
      cpu: 6
      memory: 12Gi
    requests:
      cpu: 6
      memory: 12Gi
    image: selectdb/doris.fe-ubuntu:2.0.2
  beSpec:
    replicas: 3
    service:
      type: LoadBalancer
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
```

```
      memory: 16Gi
    image: selectdb/doris.be-ubuntu:2.0.2
```

By viewing the `kubectl get service` command, view the corresponding Service display as follows:

```
$ kubectl get service
NAME                              TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)                                                  AGE
doriscluster-sample-be-internal   ClusterIP      None             <none>
9050/TCP                                                 14h
doriscluster-sample-be-service    LoadBalancer   172.20.217.234   a46bbcd6998c7436bab8ee8fba9f5e81-808549982.us-east-
1.elb.amazonaws.com    9060:32060/TCP,8040:30615/TCP,9050:31742/TCP,8060:31127/TCP   14h
doriscluster-sample-fe-internal   ClusterIP      None             <none>
9030/TCP                                                 14h
doriscluster-sample-fe-service    LoadBalancer   172.20.183.136   ac48284932b044251bfac389b453118f-1412731848.us-east-
1.elb.amazonaws.com   8030:32213/TCP,9020:31080/TCP,9030:31433/TCP,9010:30585/TCP   14h
```

External ports corresponding to `EXTERNAL-IP` and `PORT` can be used outside K8s to access the services of various components within K8s. For example, to access the `mysql` client service corresponding to FE's 9030, you can use the following command to connect:

```
mysql -h ac48284932b044251bfac389b453118f-1412731848.us-east-1.elb.amazonaws.com -P 9030 -uroot
```

NodePort Mode

In a private network environment, to access internal services outside K8s, it is recommended to use the `NodePort` mode of K8s. Use the default configuration as a blueprint to configure the `NodePort` access mode in the private network as follows:

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  labels:
    app.kubernetes.io/name: doriscluster
    app.kubernetes.io/instance: doriscluster-sample
    app.kubernetes.io/part-of: doris-operator
  name: doriscluster-sample
spec:
  feSpec:
    replicas: 3
    service:
      type: NodePort
    limits:
      cpu: 6
      memory: 12Gi
    requests:
      cpu: 6
      memory: 12Gi
    image: selectdb/doris.fe-ubuntu:2.0.2
  beSpec:
    replicas: 3
    service:
      type: NodePort
    limits:
      cpu: 8
      memory: 16Gi
    requests:
```

```
      cpu: 8
      memory: 16Gi
    image: selectdb/doris.be-ubuntu:2.0.2
```

After deployment, view the corresponding Service by viewing the `kubectl get service` command:

```
$ kubectl get service
NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes                        ClusterIP   10.152.183.1     <none>        443/TCP
169d
doriscluster-sample-fe-internal   ClusterIP   None             <none>        9030/TCP
2d
doriscluster-sample-fe-service    NodePort    10.152.183.58    <none>
8030:31041/TCP,9020:30783/TCP,9030:31545/TCP,9010:31610/TCP   2d
doriscluster-sample-be-internal   ClusterIP   None             <none>        9050/TCP
2d
doriscluster-sample-be-service    NodePort    10.152.183.244   <none>
9060:30940/TCP,8040:32713/TCP,9050:30621/TCP,8060:30926/TCP   2d
```

The above command obtains the port that can be used outside K8s, and obtains the host managed by K8s through the following command:

```
$ kubectl get nodes -owide
NAME             STATUS   ROLES    AGE    VERSION                    INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION       CONTAINER-RUNTIME
vm-10-7-centos   Ready    <none>   88d    v1.23.17-2+40cc20cc310518   10.16.10.7    <none>        TencentOS Server 3.1 (Final)
5.4.119-19.0009.25   containerd://1.5.13
vm-10-8-centos   Ready    <none>   169d   v1.23.17-2+40cc20cc310518   10.16.10.8    <none>        TencentOS Server 3.1 (Final)
5.4.119-19-0009.3    containerd://1.5.13
```

In a private network environment, use the K8s host and mapped ports to access K8s internal services. For example, we use the host's IP and FE's 9030 mapped port (31545) to connect to `mysql`:

```
$ mysql -h 10.16.10.8 -P 31545 -uroot
```

In addition, you can specify the nodePort you need according to your own platform needs. The Kubernetes master will allocate a port from the given configuration range (general default: 30000-32767) and each Node will proxy to the Service from that port (the same port on each Node). Like the example above, a random port will be automatically generated if not specified.

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  labels:
    app.kubernetes.io/name: doriscluster
    app.kubernetes.io/instance: doriscluster-sample
    app.kubernetes.io/part-of: doris-operator
  name: doriscluster-sample
spec:
  feSpec:
    replicas: 3
    service:
      type: NodePort
      servicePorts:
        - nodePort: 31001
```

```
          targetPort: 8030
        - nodePort: 31002
          targetPort: 9020
        - nodePort: 31003
          targetPort: 9030
        - nodePort: 31004
          targetPort: 9010
    limits:
      cpu: 6
      memory: 12Gi
    requests:
      cpu: 6
      memory: 12Gi
    image: selectdb/doris.fe-ubuntu:2.0.2
  beSpec:
    replicas: 3
    service:
      type: NodePort
      servicePorts:
        - nodePort: 31005
          targetPort: 9060
        - nodePort: 31006
          targetPort: 8040
        - nodePort: 31007
          targetPort: 9050
        - nodePort: 31008
          targetPort: 8060
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    image: selectdb/doris.be-ubuntu:2.0.2
```

After deployment, check the corresponding Service by viewing the `kubectl get service` command. For access methods, please refer to the above:

```
$ kubectl get service
NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes                        ClusterIP   10.152.183.1     <none>        443/TCP
169d
doriscluster-sample-fe-internal   ClusterIP   None             <none>        9030/TCP
2d
doriscluster-sample-fe-service    NodePort    10.152.183.67    <none>
8030:31001/TCP,9020:31002/TCP,9030:31003/TCP,9010:31004/TCP   2d
doriscluster-sample-be-internal   ClusterIP   None             <none>        9050/TCP
2d
doriscluster-sample-be-service    NodePort    10.152.183.24    <none>
9060:31005/TCP,8040:31006/TCP,9050:31007/TCP,8060:31008/TCP   2d
```

Doris data exchange

Stream load

Stream load is a synchronous import method. Users send requests to import local files or data streams into Doris by sending HTTP protocol. In a regular deployment, users submit import commands via the HTTP protocol. Generally, users will submit the request to FE, and FE will forward the request to a certain BE through

the HTTP redirect command. However, in a Kubernetes-based deployment scenario, it is recommended that users **directly submit the import command to BE's Srevice**, and then the Service will be load balanced to a certain BE pod based on Kubernetes rules. The actual effects of these two operations are the same. When Flink or Spark uses the official connector to submit, the write request can also be submitted to the BE Service.

ErrorURL

When import methods such as Stream load and Routine load These import methods will print `errorURL` or `tracking_url` in the return structure or log when encountering errors such as incorrect data format. You can locate the cause of the import error by visiting this link. However, this URL is only accessible within the internal environment of a specific BE node container in a Kubernetes deployed cluster.

The following scenario takes the errorURL returned by Doris as an example: `http://doriscluster-sample-be-2.doriscluster-sample-be-internal.doris.svc.cluster.local:8040/api/_load_error_log?file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e`

1. Kubernetes cluster internal access

You need to obtain the access method of BE's Service or pod through the `kubectl get service` or `kubectl get pod -o wide` command, replace the domain name and port of the original URL, and then access again.

for example:

```
$ kubectl get pod -o wide
NAME                       READY   STATUS    RESTARTS     AGE   IP           NODE                     NOMINATED NODE   READINESS
GATES
doriscluster-sample-be-0   1/1     Running   0            9h    10.0.2.105   10-0-2-47.ec2.internal   <none>           <none>
doriscluster-sample-be-1   1/1     Running   0            9h    10.0.2.104   10-0-2-5.ec2.internal    <none>           <none>
doriscluster-sample-be-2   1/1     Running   0            9h    10.0.2.103   10-0-2-6.ec2.internal    <none>           <none>
doriscluster-sample-fe-0   1/1     Running   0            9h    10.0.2.102   10-0-2-47.ec2.internal   <none>           <none>
doriscluster-sample-fe-1   1/1     Running   0            9h    10.0.2.101   10-0-2-5.ec2.internal    <none>           <none>
doriscluster-sample-fe-2   1/1     Running   0            9h    10.0.2.100   10-0-2-6.ec2.internal    <none>           <none>
```

The above errorURL is changed to: `http://10.0.2.103:8040/api/_load_error_log?file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e`

2. NodePort mode for external access to Kubernetes cluster

Obtaining error report details from outside Kubernetes requires additional bridging means. The following are the processing steps for using NodePort mode Service when deploying Doris. Obtain error report details by creating a new Service: Handle Service template be_streamload_errror_service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.doris.service/role: debug
    app.kubernetes.io/component: be
  name: doriscluster-detail-error
spec:
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
```

```
  ports:
    - name: webserver-port
      port: 8040
      protocol: TCP
      targetPort: 8040
  selector:
    app.kubernetes.io/component: be
    statefulset.kubernetes.io/pod-name: ${podName}
  sessionAffinity: None
  type: NodePort
```

Use the following command to view the mapping of Service 8040 port on the host machine

```
$ kubectl get service -n doris doriscluster-detail-error
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
doriscluster-detail-error NodePort 10.152.183.35 <none> 8040:31201/TCP 32s
```

Use the IP of any host and the NodePort (31201) port obtained above to access and replace errorURL to obtain a detailed error report. The above errorURL is changed to: `http://10.152.183.35:31201/api/_load_error_log?file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968e_af474190276a2e9c_49bb9d175b8e968e`

3. Access LoadBalancer mode from outside the Kubernetes cluster

Obtaining error report details from outside Kubernetes requires additional bridging means. The following are the processing steps for using the LoadBalancer mode Service when deploying Doris. Obtain error report details by creating a new Service: Handle Service template be_streamload_errror_service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.doris.service/role: debug
    app.kubernetes.io/component: be
  name: doriscluster-detail-error
spec:
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: webserver-port
      port: 8040
      protocol: TCP
      targetPort: 8040
  selector:
    app.kubernetes.io/component: be
    statefulset.kubernetes.io/pod-name: ${podName}
  sessionAffinity: None
  type: LoadBalancer
```

`podName` is replaced with the highest-level domain name of errorURL: `doriscluster-sample-be-2`.

In the `namespace` deployed by Doris (generally the default is `doris`, use `doris` for the following operations), use the following command to deploy the service processed above:

```
$ kubectl apply -n doris -f be_streamload_errror_service.yaml
```

Use the following command to view the mapping of Service 8040 port on the host machine

```
$ kubectl get service -n doris doriscluster-detail-error
NAME                          TYPE          CLUSTER-IP       EXTERNAL-IP
PORT(S)           AGE
doriscluster-detail-error    LoadBalancer  172.20.183.136   ac4828493dgrftb884g67wg4tb68gyut-1137856348.us-east-1.elb.amazonaws.com
8040:32003/TCP    14s
```

Use `EXTERNAL-IP` and Port (8040) port access to replace errorURL to obtain a detailed error report. The above errorURL is changed to: `http://ac4828493dgrftb884g67wg4tb68gyut-1137856348.us-east-1.elb.amazonaws.com:8040/api/_load_error_log?file=__shard_1/error_log_insert_stmt_af474190276a2e9c-49bb9d175b8e968 e_af474190276a2e9c_49bb9d175b8e968e`

Note: For the above method of setting up access outside the cluster, it is recommended to clear the current Service after use. The reference command is as follows:

```
$ kubectl delete service -n doris doriscluster-detail-error
```

Deploy Doris on Kubernetes

Doris-Operator is software extension to Kubernetes that make use of custom resource to manage Doris and it components. It provides DorisCluster a Kubernetes CustomResourceDefinition for user to custom resource.
Deploy Doris on Kubernetes

Start Kubernetes

Having a Kubernetes environment is the premise to deploy Doris on Kubernetes. If you already have it, please ignore this step. Hosted Kubernetes on cloud platform or set-up by yourself are all good choice.

Hosted EKS
1. Check that the following command-line tools are installed in your environment:
- Install and configure AWS command-line tool AWS CLI. - Install EKS cluster command-line tool eksctl. - Install Kubernetes cluster command-line tool kubectl.

2. Use one of the following methods to create an EKS cluster:

- Use eksctl to quickly create an EKS cluster.
- Manually create an EKS cluster with the AWS console and AWS CLI.

Hosted GKE
Complete all the prerequisites when create a GKE cluster.

Create as Kubernetes recommend
Kubernetes official documents recommends some ways to set up Kubernetes, as minikube,kOps.

Deploy Doris-Operator on Kubernetes

1. Apply the Custom Resource Definition(CRD)

```
kubectl apply -f https://raw.githubusercontent.com/apache/doris-
operator/master/config/crd/bases/doris.selectdb.com_dorisclusters.yaml
```

2. Install Doris-Operator
If you want to use the defaults operator resource:

```
kubectl apply -f https://raw.githubusercontent.com/apache/doris-operator/master/config/operator/operator.yaml
```

The user defined deployment in github repo are simply:
Instead of using the command below, apply your local version of the Operator manifest to the cluster when you custom operator resource.

```
kubectl apply -f operator.yaml
```

3. Validate The Operator is Running

Using the command `kubectl -n {namespace} get pods` get the status of deployed operator.

```
kubectl -n doris get pods
NAME                              READY   STATUS    RESTARTS        AGE
doris-operator-5b9f7f57bf-tsvjz   1/1     Running   66 (164m ago)   6d22h
```

Expected result, the Pod `STATUS` is `Running` and all containers in Pod are all `READY`.

Start Doris on Kubernetes

1. Initialize Doris Cluster

User can directly deploy Doris by examples provided by Doris-Operator. Below is the command:

```
kubectl apply -f https://raw.githubusercontent.com/apache/doris-operator/master/doc/examples/doriscluster-sample.yaml
```

Or download doriscluster-sample a custom resource that tells the Operator how to configure the Kubernetes cluster, and custom resource as api.md and how_to_use docs. Instead of using the command below, apply the customized resource.

```
kubeectl apply -f doriscluster-sample.yaml
```

2. Validate Doris Cluster Status

Using the command `kubectl -n {namespace} get pods` check pods status.

```
kubectl get pods
  NAME                        READY   STATUS    RESTARTS   AGE
  doriscluster-sample-fe-0    1/1     Running   0          20m
  doriscluster-sample-be-0    1/1     Running   0          19m
```

All Pods created by DorisCluster resource should be in `Running` STATUS, and each pod's containers should be `RREADY`. #### Use Doris Cluster
On kubernetes Doris-Operator provide `Service` a resource build in kubernetes for access to Doris.

The command `kubectl -n {namespace} get svc -l "app.doris.ownerreference/name={dorisCluster.Name}"` used to get `service` created by Doris-Operator. `dorisCluster.Nmae` is the name of DorisCluster resource deployed by step 1.

```
kubectl -n default get svc -l "app.doris.ownerreference/name=doriscluster-sample"
NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP                                          PORT(S)
AGE
doriscluster-sample-fe-internal   ClusterIP   None             <none>                                               9030/TCP
30m
doriscluster-sample-fe-service    ClusterIP   10.152.183.37    a7509284bf3784983a596c6eec7fc212-618xxxxxx.com
8030/TCP,9020/TCP,9030/TCP,9010/TCP   30m
doriscluster-sample-be-internal   ClusterIP   None             <none>                                               9050/TCP
29m
doriscluster-sample-be-service    ClusterIP   10.152.183.141   <none>
9060/TCP,8040/TCP,9050/TCP,8060/TCP   29m
```

Use SQL Client for Access

Service created by Doris-Operator have two types, suffix is `-internal` or `-service`. Service have the `-internal` suffix for communicating in Doris components, Service have `-service` suffix for user to access.

- In Kubernetes
  In kubernetes, Using `CLUSTER-IP` is recommended. For example, the fe service's `CLUSTER-IP` is `10.152.183.37` that displayed by above command. Using below command to access fe service.

  ```
  mysql -h 10.152.183.37 -uroot -P9030
  ```

- Out Kubernetes
  Using `EXTERNAL-IP` to access fe from Kubernetes external. In default, Doris-Operator not provided `EXTERNAL-IP` mode. If you want to use `EXTERNAL-IP`, should custom resource `Service` field, reference the doc api.md to deploy.

> **If the doc not cover your requirements, Pleaser reference the docs how_to_use.md and api.md to custom `DorisCluster` resource to deploy.**

Persistent Volume and ConfigMap

Doris-Operator supports mounting PV (Persistent Volume) on pods of various Doris components.

PV is generally created by the kubernetes system administrator. Doris-Operator does not use PV directly when deploying Doris services. Instead, it declares a set of resources through PVC to apply for PV from the kubernetes cluster. When a PVC is created, Kubernetes will attempt to bind it to an available PV that meets the requirements. StorageClass shields administrators from the process of manually creating PVs. When there are no ready-made PVs that meet PVC requirements, PVs can be dynamically allocated based on StorageClass. PV provides a variety of storage types, mainly divided into two categories: network storage and local storage. Based on their respective principles and implementations, the two provide users with different performance and usage experiences. Users can choose according to their own containerized service types and their own needs.

If PVC is not configured during deployment, Doris-Operator uses the `emptyDir` mode by default to store metadata data files and run logs. When the pod is restarted, related data will be lost.

Recommended node directory type for persistent storage: * FE: doris-meta, log * BE: storage, log * CN: storage, log * BROKER: log

Doris-Operator outputs logs to the console and the specified directory at the same time. If the user's Kubernetes system has complete log collection capabilities, log information at the Doris INFO level (default) can be collected through console output. However, it is still recommended to configure PVC to persist log files, because in addition to INFO level logs, there are also logs such as fe.out, be.out, audit.log and garbage collection logs, which facilitates quick problem location and audit log backtracking.

ConfigMap is a resource object used to store configuration files in Kubernetes. It allows dynamically mounting configuration files and decouples configuration files from applications, making configuration management more flexible and maintainable. Like PVCs, ConfigMap can be referenced by Pods in order to use configuration data in the application.

StorageClass

Doris-Operator provides Kubernetes default `StorageClass` mode to support FE and BE data storage, where the storage path (mountPath) uses the default configuration in the image. If users need to specify the StorageClass themselves, they need to modify `persistentVolumeClaimSpec.storageClassName` in `spec.feSpec.persistentVolumes`, as shown below:

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  labels:
    app.kubernetes.io/name: doriscluster
  name: doriscluster-sample-storageclass1
spec:
  feSpec:
    replicas: 3
    image: selectdb/doris.fe-ubuntu:2.0.2
    limits:
      cpu: 8
      memory: 16Gi
```

```
    requests:
      cpu: 8
      memory: 16Gi
    persistentVolumes:
    - mountPath: /opt/apache-doris/fe/doris-meta
      name: storage0
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        storageClassName: ${your_storageclass}
        accessModes:
        - ReadWriteOnce
        resources:
          # notice: if the storage size less 5G, fe will not start normal.
          requests:
            storage: 100Gi
    - mountPath: /opt/apache-doris/fe/log
      name: storage1
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        storageClassName: ${your_storageclass}
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
  beSpec:
    replicas: 3
    image: selectdb/doris.be-ubuntu:2.0.2
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    persistentVolumes:
    - mountPath: /opt/apache-doris/be/storage
      name: storage2
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        storageClassName: ${your_storageclass}
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
    - mountPath: /opt/apache-doris/be/log
      name: storage3
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        storageClassName: ${your_storageclass}
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
```

Customized ConfigMap

Doris uses `ConfigMap` in Kubernetes to decouple configuration files and services. Before deploying `doriscluster`, you need to deploy the `ConfigMap` you want to use under the same `namespace` in advance. The following example shows that FE uses `ConfigMap` named fe-configmap and BE uses `ConfigMap` named be-configmap. Cluster related yaml:

ConfigMap sample for FE

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: fe-configmap
  labels:
    app.kubernetes.io/component: fe
data:
  fe.conf: |
    CUR_DATE=`date +%Y%m%d-%H%M%S`

    # the output dir of stderr and stdout
    LOG_DIR = ${DORIS_HOME}/log

    JAVA_OPTS="-Djavax.security.auth.useSubjectCredsOnly=false -Xss4m -Xmx8192m -XX:+UseMembar -XX:SurvivorRatio=8 -
XX:MaxTenuringThreshold=7 -XX:+PrintGCDateStamps -XX:+PrintGCDetails -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -
XX:+CMSClassUnloadingEnabled -XX:-CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=80 -XX:SoftRefLRUPolicyMSPerMB=0 -
Xloggc:$DORIS_HOME/log/fe.gc.log.$CUR_DATE"

    # For jdk 9+, this JAVA_OPTS will be used as default JVM options
    JAVA_OPTS_FOR_JDK_9="-Djavax.security.auth.useSubjectCredsOnly=false -Xss4m -Xmx8192m -XX:SurvivorRatio=8 -
XX:MaxTenuringThreshold=7 -XX:+CMSClassUnloadingEnabled -XX:-CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=80 -
XX:SoftRefLRUPolicyMSPerMB=0 -Xlog:gc*:$DORIS_HOME/log/fe.gc.log.$CUR_DATE:time"

    # INFO, WARN, ERROR, FATAL
    sys_log_level = INFO

    # NORMAL, BRIEF, ASYNC
    sys_log_mode = NORMAL

    # Default dirs to put jdbc drivers,default value is ${DORIS_HOME}/jdbc_drivers
    # jdbc_drivers_dir = ${DORIS_HOME}/jdbc_drivers

    http_port = 8030
    rpc_port = 9020
    query_port = 9030
    edit_log_port = 9010

    enable_fqdn_mode = true
```

Note that when using FE's ConfigMap, you must add `enable_fqdn_mode = true` to `fe.conf`. For specific reasons, please refer to document here

BE's ConfigMap sample

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: be-configmap
  labels:
    app.kubernetes.io/component: be
data:
  be.conf: |
    CUR_DATE=`date +%Y%m%d-%H%M%S`

    PPROF_TMPDIR="$DORIS_HOME/log/"

    JAVA_OPTS="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xloggc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command=DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"
```

```
    # For jdk 9+, this JAVA_OPTS will be used as default JVM options
    JAVA_OPTS_FOR_JDK_9="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xlog:gc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command=DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"

    # since 1.2, the JAVA_HOME need to be set to run BE process.
    # JAVA_HOME=/path/to/jdk/

    # https://github.com/apache/doris/blob/master/docs/zh-CN/community/developer-guide/debug-tool.md#jemalloc-heap-profile
    # https://jemalloc.net/jemalloc.3.html

JEMALLOC_CONF="percpu_arena:percpu,background_thread:true,metadata_thp:auto,muzzy_decay_ms:15000,dirty_decay_ms:15000,oversize_thres
hold:0,lg_tcache_max:20,prof:false,lg_prof_interval:32,lg_prof_sample:19,prof_gdump:false,prof_accum:false,prof_leak:false,prof_fina
l:false"
    JEMALLOC_PROF_PRFIX=""

    # INFO, WARNING, ERROR, FATAL
    sys_log_level = INFO

    # ports for admin, web, heartbeat service
    be_port = 9060
    webserver_port = 8040
    heartbeat_service_port = 9050
    brpc_port = 8060
```

`doriscluster` deployment example using the above two `ConfigMap`:

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  labels:
    app.kubernetes.io/name: doriscluster
  name: doriscluster-sample-configmap
spec:
  feSpec:
    replicas: 3
    image: selectdb/doris.fe-ubuntu:2.0.2
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    configMapInfo:
      # use kubectl create configmap fe-configmap --from-file=fe.conf
      configMapName: fe-configmap
      resolveKey: fe.conf
  beSpec:
    replicas: 3
    image: selectdb/doris.be-ubuntu:2.0.2
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    configMapInfo:
      # use kubectl create configmap be-configmap --from-file=be.conf
      configMapName: be-configmap
      resolveKey: be.conf
  brokerSpec:
    replicas: 3
    image: selectdb/doris.broker-ubuntu:2.0.2
```

```
    limits:
      cpu: 2
      memory: 4Gi
    requests:
      cpu: 2
      memory: 4Gi
    configMapInfo:
      # use kubectl create configmap broker-configmap --from-file=apache_hdfs_broker.conf
      configMapName: broker-configmap
      resolveKey: apache_hdfs_broker.conf
```

The `resolveKey` here is the name of the incoming configuration file (must be `fe.conf`, `be.conf` or `apache_hdfs_broker.conf`, the cn node is also `be.conf`) used to parse the incoming Doris cluster configuration file, doris-operator will parse the file to guide the customized deployment of doriscluster.

Add special configuration files to the conf directory

This paragraph is for reference. Containerized deployment solutions that configure other files need to be placed in the conf directory of the Doris node. For example, the common HDFS/Hive configuration file mapping of Data Lake Multi-catalog.

Here we take BE's ConfigMap and the core-site.xml file that needs to be added as an example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: be-configmap
  labels:
    app.kubernetes.io/component: be
data:
  be.conf: |
    be_port = 9060
    webserver_port = 8040
    heartbeat_service_port = 9050
    brpc_port = 8060
  core-site.xml: |
    <?xml version="1.0" encoding="UTF-8"?>
    <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
    <configuration>
      <property>
      <name>hadoop.security.authentication</name>
        <value>kerberos</value>
      </property>
    </configuration>
    ...
```

Note that the data structure in `data` is as follows: key-value pair mapping:

```
data:
  file_name_1:
    file_content_1
  file_name_2:
    file_content_2
  file_name_3:
    file_content_3
```

BE multi-disk configuration

Doris' BE service supports multi-disk mounting, which can well solve the problem of mismatch between computing resources and storage resources in the server era. At the same time, using multiple disks can also greatly improve the storage efficiency of doris. On Kubernetes, Doris can also mount multiple disks to maximize storage efficiency. Using multiple disks on Kubernetes requires using configuration files. In order to achieve decoupling of service and configuration, doris uses `ConfigMap` as the bearer of configuration to dynamically mount configuration files for service use. The following is the doriscluster configuration in which the BE service uses `ConfigMap` to host the configuration file and mount two disks for BE use:

```
apiVersion: doris.selectdb.com/v1
kind: DorisCluster
metadata:
  labels:
    app.kubernetes.io/name: doriscluster
  name: doriscluster-sample-storageclass1
spec:
  feSpec:
    replicas: 3
    image: selectdb/doris.fe-ubuntu:2.0.2
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    persistentVolumes:
    - mountPath: /opt/apache-doris/fe/doris-meta
      name: storage0
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          # notice: if the storage size less 5G, fe will not start normal.
          requests:
            storage: 100Gi
    - mountPath: /opt/apache-doris/fe/log
      name: storage1
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
  beSpec:
    replicas: 3
    image: selectdb/doris.be-ubuntu:2.0.2
    limits:
      cpu: 8
      memory: 16Gi
    requests:
      cpu: 8
      memory: 16Gi
    configMapInfo:
      configMapName: be-configmap
      resolveKey: be.conf
    persistentVolumes:
```

```
    - mountPath: /opt/apache-doris/be/storage
      name: storage2
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
    - mountPath: /opt/apache-doris/be/storage1
      name: storage3
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
    - mountPath: /opt/apache-doris/be/log
      name: storage4
      persistentVolumeClaimSpec:
        # when use specific storageclass, the storageClassName should reConfig, example as annotation.
        #storageClassName: openebs-jiva-csi-default
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
```

Compared with the default example, the configuration of `configMapInfo` is added, and a configuration of `persistentVolumeClaimSpec` is also added, `persistentVolumeClaimSpec` fully follows the definition format of the Kubernetes native resource PVC spec. In the example, `configMapInfo` identifies which ConfigMap under the same `namespace` and which key corresponding content will be used as the configuration file after BE is deployed, where the key must be be.conf. The following is an example of the above `doriscluster` ConfigMap that needs to be pre-deployed:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: be-configmap
  labels:
    app.kubernetes.io/component: be
data:
  be.conf: |
    CUR_DATE=`date +%Y%m%d-%H%M%S`

    PPROF_TMPDIR="$DORIS_HOME/log/"

    JAVA_OPTS="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xloggc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command=DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"

    # For jdk 9+, this JAVA_OPTS will be used as default JVM options
    JAVA_OPTS_FOR_JDK_9="-Xmx1024m -DlogPath=$DORIS_HOME/log/jni.log -Xlog:gc:$DORIS_HOME/log/be.gc.log.$CUR_DATE -
Djavax.security.auth.useSubjectCredsOnly=false -Dsun.java.command=DorisBE -XX:-CriticalJNINatives -DJDBC_MIN_POOL=1 -
DJDBC_MAX_POOL=100 -DJDBC_MAX_IDLE_TIME=300000 -DJDBC_MAX_WAIT_TIME=5000"

    # since 1.2, the JAVA_HOME need to be set to run BE process.
    # JAVA_HOME=/path/to/jdk/
```

```
    # https://github.com/apache/doris/blob/master/docs/zh-CN/community/developer-guide/debug-tool.md#jemalloc-heap-profile
    # https://jemalloc.net/jemalloc.3.html

JEMALLOC_CONF="percpu_arena:percpu,background_thread:true,metadata_thp:auto,muzzy_decay_ms:15000,dirty_decay_ms:15000,oversize_thres
hold:0,lg_tcache_max:20,prof:false,lg_prof_interval:32,lg_prof_sample:19,prof_gdump:false,prof_accum:false,prof_leak:false,prof_fina
l:false"
    JEMALLOC_PROF_PRFIX=""

    # INFO, WARNING, ERROR, FATAL
    sys_log_level = INFO

    # ports for admin, web, heartbeat service
    be_port = 9060
    webserver_port = 8040
    heartbeat_service_port = 9050
    brpc_port = 8060

    storage_root_path = /opt/apache-doris/be/storage,medium:ssd;/opt/apache-doris/be/storage1,medium:ssd
```

When using multiple disks, the path in the corresponding value of `storage_root_path` in `ConfigMap` should correspond to each mounting path of `persistentVolume` in `doriscluster`. storage root path For the corresponding writing rules, please refer to the document in the link. When using cloud disks, the media is uniformly `SSD`.

Building log analysis platform

Logs record key events in the system and contain crucial information such as the events' subject, time, location, and content. To meet the diverse needs of observability in operations, network security monitoring, and business analysis, enterprises might need to collect scattered logs for centralized storage, querying, and analysis to extract valuable content from the log data further.

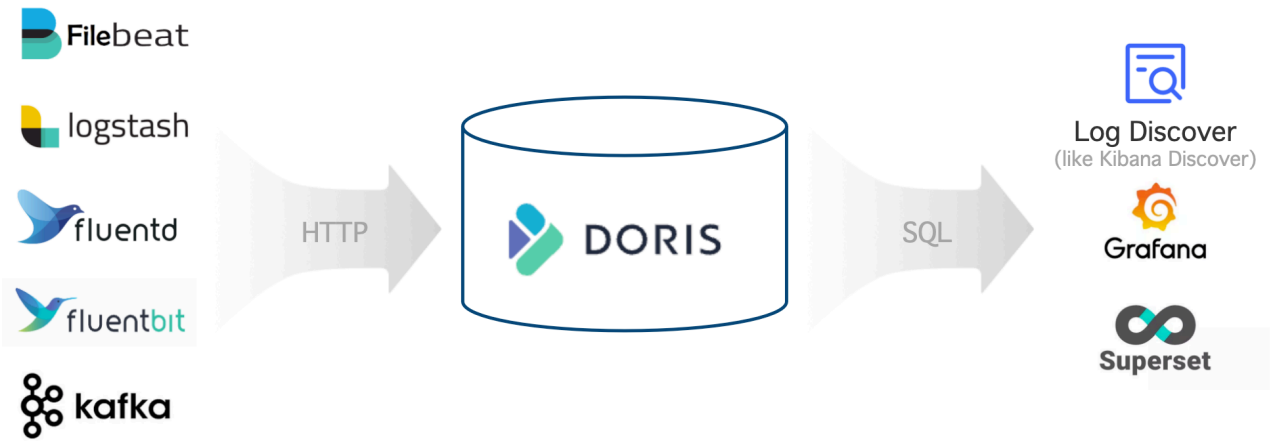
In this scenario, Apache Doris provides a corresponding solution. With the characteristics of log scenarios in mind, Apache Doris added inverted-index and ultra-fast full-text search capabilities, optimizing write performance and storage space to the extreme. This allows users to build an open, high-performance, cost-effective, and unified log storage and analysis platform based on Apache Doris.

Focused on this solution, this chapter contains the following 3 sections:

- **Overall architecture**: This section explains the core components and architecture of the log storage and analysis platform built on Apache Doris.
- **Features and advantages**: This section explains the features and advantages of the log storage and analysis platform built on Apache Doris.
- **Operational guide**: This section explains how to build a log storage and analysis platform based on Apache Doris.

Overall architecture

The following figure illustrates the architecture of the log storage and analysis platform built on Apache Doris:



Overall architecture

The architecture contains the following 3 parts:

- **Log collection and preprocessing**: Various log collection tools can write log data into Apache Doris through HTTP APIs.
- **Log storage and analysis engine**: Apache Doris provides high-performance, low-cost unified log storage, offering rich retrieval and analysis capabilities through SQL interfaces.

- **Log analysis and alert interface**: Various log retrieval and analysis tools can query Apache Doris through standard SQL interfaces, providing users with a simple and user-friendly interface.

Features and advantages

The following figure illustrates the architecture of the log storage and analysis platform built on Apache Doris::

- **High throughput, low latency log writing**: Supports stable writing of log data at the level of hundreds of TBs and GB/s per day, while maintaining latency within 1 second.
- **Cost-effective storage of massive log data**: Supports petabyte-scale storage, saving 60% to 80% in storage costs compared to Elasticsearch, and further reducing storage costs by 50% by storing cold data in S3/HDFS.
- **High-performance log full-text search and analysis**: Supports inverted indexes and full-text search, providing second-level response times for common log queries (keyword searches, trend analysis, etc.).
- **Open and user-friendly upstream and downstream ecosystem**: Upstream integration with common log collection systems and data sources such as Logstash, Filebeat, Fluentbit, and Kafka through Stream Load's universal HTTP APIs, and downstream integration with various visual analytics UIs using standard MySQL protocol and syntax, such as observability platform Grafana, BI analytics Superset, and log retrieval Doris WebUI similar to Kibana.

Cost-effective performance

After Benchmark testing and production validation, the log storage and analysis platform built on Apache Doris has a 5 to 10 times cost-performance advantage over Elasticsearch. Apache Doris's performance benefits are mainly due to its globally leading high-performance storage and query engine, as well as specialized optimizations for log scenarios:

- **Improved write throughput**: Elasticsearch's write performance bottleneck lies in CPU consumption for parsing data and building inverted indexes. In comparison, Apache Doris has optimized writes in two aspects: using SIMD and other CPU vector instructions to improve JSON data parsing speed and index-building performance and simplifying the inverted index structure for log scenarios by removing unnecessary data structures like forward indexes, effectively reducing index build complexity. With the same resources, Apache Doris's write performance is 3 to 5 times higher than Elasticsearch.
- **Reduced Storage Costs**: The storage bottleneck in Elasticsearch lies in the multiple storage formats for forward indexes, inverted indexes, and Docvalue columns, as well as the relatively low compression ratio of its general compression algorithms. In contrast, Apache Doris has made the following optimizations in storage: it removes the forward index, reducing the index data size by 30%; it uses columnar storage and the Zstandard compression algorithm, achieving a compression ratio of 5 to 10 times, which is significantly higher than Elasticsearch's 1.5 times; in log data, cold data is accessed very infrequently, and Apache Doris's hot and cold data tiering feature can automatically store logs that exceed a defined time period into lower-cost object storage, reducing the storage cost of cold data by more than 70%. For the same raw data, Doris's storage costs are only about 20% of those of Elasticsearch.

Strong analytical capabilities

Apache Doris supports standard SQL and is compatible with MySQL protocol and syntax. Therefore, log systems built on Apache Doris can use SQL for log analysis, giving the following advantages to log systems:

- **Easy to use**: Engineers and data analysts are very familiar with SQL, their expertise can be reused, no need to learn new technology stacks to quickly get started.
- **Rich ecosystem**: The MySQL ecosystem is the most widely used language in the database field, seamlessly integrating with and applying to the MySQL ecosystem. Doris can leverage MySQL command line and various GUI tools, BI tools, and other big data ecosystem tools for more complex and diverse data processing and analysis needs.
- **Strong analytical capabilities**: SQL has become the de facto standard for database and big data analysis, with powerful expressive capabilities and functions supporting retrieval, aggregation, multi-table JOIN, subqueries, UDFs, logical views, materialized views, and various data analysis capabilities.

Flexible Schema

Here is a typical example of a semi-structured log in JSON format. The top-level fields are some fixed fields, such as timestamp, source, node, component, level, clientRequestID, message, and properties, which are present in every log entry. The nested fields of the properties , such as properties.size and properties.format are more dynamic, and the fields of each log may vary.

```
{
  "timestamp": "2014-03-08T00:50:03.8432810Z",
  "source": "ADOPTIONCUSTOMERS81",
  "node": "Engine000000000405",
  "level": "Information",
  "component": "DOWNLOADER",
  "clientRequestId": "671db15d-abad-94f6-dd93-b3a2e6000672",
  "message": "Downloading file path: benchmark/2014/ADOPTIONCUSTOMERS81_94_0.parquet.gz",
  "properties": {
    "size": 1495636750,
    "format": "parquet",
    "rowCount": 855138,
    "downloadDuration": "00:01:58.3520561"
  }
}
```

Apache Doris provides several aspects of support for Flexible Schema log data:

- For changes to top-level fields, Light Schema Change can be used to add or remove columns and to add or remove indexes, enabling schema changes to be completed in seconds. When planning a log platform, users only need to consider which fields need to be indexed.

Operational guide

Step 1: Estimate resources

Before deploying the cluster, you need to estimate the hardware resources required for the servers. Follow the steps below:

1. Estimate the resources for data writing by the following calculation formulas:

- `Average write throughput = Daily data increment / 86400 s`
- `Peak write throughput = Average write throughput \* Ratio of the peak write throughput to the average write throughput`
- `Number of CPU cores for the peak write throughput = Peak write throughput / Write throughput of a single-core CPU`

1. Estimate the resources for data storage by the calculation formula: `Storage space = Daily data increment / Data compression ratio * Number of data copies * Data storage duration`.
2. Estimate the resources for data querying. The resources for data querying depend on the query volume and complexity. It is recommended to reserve 50% of CPU resources for data query initially and then adjust according to the actual test results.
3. Integrate the calculation results as follows:
    1. Divide the number of CPU cores calculated in Step 1 and Step 3 by the number of CPU cores of a BE server, and you can get the number of BE servers.
    2. Based on the number of BE servers and the calculation result of Step 2, estimate the storage space required for each BE server.
    3. Allocate the storage space required for each BE server to 4 to 12 data disks, and you can get the storage capacity required for a single data disk.

For example, suppose that the daily data increment is 100 TB, the data compression ratio is 5, the number of data copies is 1, the storage duration of hot data is 3 days, the storage duration of cold data is 30 days, the ratio of the peak write throughput to the average write throughput is 200%, the write throughput of a single-core CUP is 10 MB/s, and 50% of CPU resources are reserved for data querying, one can estimate that:

- 3 FE servers are required, each configured with a 16-core CPU, 64 GB memory, and an 1 100 GB SSD disk.
- 15 BE servers are required, each configured with a 32-core CPU, 256 GB memory, and 10 600 GB SSD disks.
- S3 object storage space 600 TB

Refer to the following table to learn about the values of indicators in the example above and how they are calculated.

Indicator (Unit)	Value	Description
Daily data increment (TB)	100	Specify the value according to your actual needs.
Data compression ratio	5	Specify the value according to your actual needs, which is often between 3 to 10. Note that the data contains index data.

Indicator (Unit)	Value	Description
Number of data copies	1	Specify the value according to your actual needs, which can be 1, 2, or 3. The default value is 1.
Storage duration of hot data (day)	3	Specify the value according to your actual needs.
Storage duration of cold data (day)	30	Specify the value according to your actual needs.
Data storage duration	33	Calculation formula: `Storage duration of hot data + Storage duration of cold data`
Estimated storage space for hot data (TB)	60	Calculation formula: `Daily data increment / Data compression ratios * Number of data copies * Storage duration of hot data`
Estimated storage space for cold data (TB)	600	Calculation formula: `Daily data increment / Data compression ratios * Number of data copies * Storage duration of cold data`
Ratio of the peak write throughput to the average write throughput	200%	Specify the value according to your actual needs. The default value is 200%.
Number of CPU cores of a BE server	32	Specify the value according to your actual needs. The default value is 32.
Average write throughput (MB/s)	1214	Calculation formula: `Daily data increment / 86400 s`
Peak write throughput (MB/s)	2427	Calculation formula: `Average write throughput * Ratio of the peak write throughput to the average write throughput`
Number of CPU cores for the peak write throughput	242.7	Calculation formula: `Peak write throughput / Write throughput of a single-core CPU`
Percent of CPU resources reserved for data querying	50%	Specify the value according to your actual needs. The default value is 50%.
Estimated number of BE servers	15.2	Calculation formula: `Number of CPU cores for the peak write throughput / Number of CPU cores of a BE server /(1 - Percent of CPU resources reserved for data querying)`
Rounded number of BE servers	15	Calculation formula: `MAX (Number of data copies, Estimated number of BE servers)`
Estimated data storage space for each BE server (TB)	5.7	Calculation formula: `Estimated storage space for hot data / Estimated number of BE servers /(1 - 30%)`, where 30% represents the percent of reserved storage space. It is recommended to mount 4 to 12 data disks on each BE server to enhance I/O capabilities.

Step 2: Deploy the cluster

After estimating the resources, you need to deploy the cluster. It is recommended to deploy in both physical and virtual environments manually. For manual deployment, refer to Manual Deployment.

Step 3: Optimize FE and BE configurations

After completing the cluster deployment, it is necessary to optimize the configuration parameters for both the front-end and back-end separately, so as to better suit the scenario of log storage and analysis.

Optimize FE configurations

You can find FE configuration fields in `fe/conf/fe.conf`. Refer to the following table to optimize FE configurations.

Configuration fields to be optimized	Description
`max_running_txn_num_per_db = 10000`	Increase the parameter value to adapt to high-concurrency import transactions.
`streaming_label_keep_max_second = 3600 label_keep_max_second = 7200`	Increase the retention time to handle high-frequency import transactions with high memory usage.
`enable_round_robin_create_tablet = true`	When creating Tablets, use a Round Robin strategy to distribute evenly.
`tablet_rebalancer_type = partition`	When balancing Tablets, use a strategy to evenly distribute within each partition.
`autobucket_min_buckets = 10`	Increase the minimum number of automatically bucketed buckets from 1 to 10 to avoid insufficient buckets when the log volume increases.
`max_backend_heartbeat_failure_tolerance_count = 10`	In log scenarios, the BE server may experience high pressure, leading to short-term timeouts, so increase the tolerance count from 1 to 10.

For more information, refer to FE Configuration.

Optimize BE configurations

You can find BE configuration fields in `be/conf/be.conf`. Refer to the following table to optimize BE configurations.

Module	Configuration fields to be optimized	Description
Storage	`storage_root_path = /path/to/dir1;/path/to/dir2;...;/path/to/dir12`	Configure the storage path for hot data on disk directories.
-	`enable_file_cache = true`	Enable file caching.

Module	Configuration fields to be optimized	Description
-	`file_cache_path = [{"path": "/mnt/datadisk0/file_cache", "total_size":53687091200, "query_limit": "10737418240"}, {"path": "/mnt/datadisk1/file_cache", "total_size":53687091200,"query_limit": "10737418240"}]`	Configure the cache path and related settings for cold data with the following specific configurations: `path`: cache path `total_size`: total size of the cache path in bytes, where 53687091200 bytes equals 50 GB `query_limit`: maximum amount of data that can be queried from the cache path in one query in bytes, where 10737418240 bytes equals 10 GB
Write	`write_buffer_size = 1073741824`	Increase the file size of the write buffer to reduce small files and random I/O operations, improving performance.
-	`max_tablet_version_num = 20000`	In coordination with the time_series compaction strategy for table creation, allow more versions to remain temporarily unmerged
Compaction	`max_cumu_compaction_threads = 8`	Set to CPU core count / 4, indicating that 1/4 of CPU resources are used for writing, 1/4 for background compaction, and 2/1 for queries and other operations.
-	`inverted_index_compaction_enable = true`	Enable inverted index compaction to reduce CPU consumption during compaction.
-	`enable_segcompaction = false enable_ordered_data_compaction = false`	Disable two compaction features that are unnecessary for log scenarios.
-	`enable_compaction_priority_scheduling = false`	Low-priority compaction is limited to 2 tasks on a single disk, which can affect the speed of compaction.
-	`total_permits_for_compaction_score = 200000`	The parameter is used to control memory, under the memory time series strategy, the parameter itself can control memory.
Cache	`disable_storage_page_cache = true inverted_index_searcher_cache_limit = 30%`	Due to the large volume of log data and limited caching effect, switch from data caching to index caching.
-	`inverted_index_cache_stale_sweep_time_sec = 3600 index_cache_entry_stay_time_after_lookup_s = 3600`	Maintain index caching in memory for up to 1 hour.
-	`enable_inverted_index_cache_on_cooldown = true enable_write_index_searcher_cache = false`	Enable automatic caching of cold data storage during index uploading.
-	`tablet_schema_cache_recycle_interval = 3600 segment_cache_capacity = 20000`	Reduce memory usage by other caches.
-	`inverted_index_ram_dir_enable = true`	Reduce the IO overhead caused by writing to index files temporarily.
Thread	`pipeline_executor_size = 24 doris_scanner_thread_pool_thread_num = 48`	Configure computing threads and I/O threads for a 32-core CPU in proportion to core count.
-	`scan_thread_nice_value = 5`	Lower the priority of query I/O threads to ensure writing performance and timeliness.
Other	`string_type_length_soft_limit_bytes = 10485760`	Increase the length limit of string-type data to 10 MB.
-	`trash_file_expire_time_sec = 300 path_gc_check_interval_second = 900 path_scan_interval_second = 900`	Accelerate the recycling of trash files.

For more information, refer to BE Configuration.

Step 4: Create tables

Due to the distinct characteristics of both writing and querying log data, it is recommended to configure tables with targeted settings to enhance performance.

Configure data partitioning and bucketing

- For data partitioning:
    - Enable range partitioning (`PARTITION BY RANGE(ts)`) with dynamic partitions (`"dynamic_partition.enable" = "true"`) managed automatically by day.
    - Use a field in the DATETIME type as the key (`DUPLICATE KEY(ts)`) for accelerated retrieval of the latest N log entries.
- For data bucketing:
    - Configure the number of buckets to be roughly three times the total number of disks in the cluster, with each bucket containing approximately 5GB of data after compression.
    - Use the Random strategy (`DISTRIBUTED BY RANDOM BUCKETS 60`) to optimize batch writing efficiency when paired with single tablet imports.

For more information, refer to Data Partitioning.

Configure compression parameters

Use the zstd compression algorithm (“compression” = “zstd”) to improve data compression efficiency.

Configure compaction parameters

Configure compaction fields as follows:

- Use the time_series strategy (`"compaction_policy" = "time_series"`) to reduce write amplification, which is crucial for high-throughput log writes.

Configure index parameters

Configuring index fields as follows:

- Create indexes for fields that are frequently queried (`USING INVERTED`).
- For fields that require full-text search, specify the parser field as unicode, which satisfies most requirements. If there is a need to support phrase queries, set the support_phrase field to true; if not needed, set it to false to reduce storage space.

Configure storage parameters

Configure storage policies as follows:

- For storage of hot data, if using cloud storage, configure the number of data copies as 1; if using physical disks, configure the number of data copies as at least 2 (`"replication_num" = "2"`).

- Configure the storage location for log_s3 (`CREATE RESOURCE "log_s3"`) and set the log_policy_3day policy (`CREATE STORAGE POLICY log_policy_3day`), where the data is cooled and moved to the specified storage location of log_s3 after 3 days. Refer to the code below.

```
CREATE DATABASE log_db;
USE log_db;

CREATE RESOURCE "log_s3"
PROPERTIES
(
    "type" = "s3",
    "s3.endpoint" = "your_endpoint_url",
    "s3.region" = "your_region",
    "s3.bucket" = "your_bucket",
    "s3.root.path" = "your_path",
    "s3.access_key" = "your_ak",
    "s3.secret_key" = "your_sk"
);

CREATE STORAGE POLICY log_policy_3day
PROPERTIES(
    "storage_resource" = "log_s3",
    "cooldown_ttl" = "259200"
);

CREATE TABLE log_table
(
  `ts` DATETIME,
  `host` TEXT,
  `path` TEXT,
  `message` TEXT,
  INDEX idx_host (`host`) USING INVERTED,
  INDEX idx_path (`path`) USING INVERTED,
  INDEX idx_message (`message`) USING INVERTED PROPERTIES("parser" = "unicode", "support_phrase" = "true")
)
ENGINE = OLAP
DUPLICATE KEY(`ts`)
PARTITION BY RANGE(`ts`) ()
DISTRIBUTED BY RANDOM BUCKETS 60
PROPERTIES (
  "compression" = "zstd",
  "compaction_policy" = "time_series",
  "dynamic_partition.enable" = "true",
  "dynamic_partition.create_history_partition" = "true",
  "dynamic_partition.time_unit" = "DAY",
  "dynamic_partition.start" = "-30",
  "dynamic_partition.end" = "1",
  "dynamic_partition.prefix" = "p",
  "dynamic_partition.buckets" = "60",
  "dynamic_partition.replication_num" = "2", -- unneccessary for the compute-storage coupled mode
  "replication_num" = "2", -- unneccessary for the compute-storage coupled mode
  "storage_policy" = "log_policy_3day" -- unneccessary for the compute-storage coupled mode
);
```

Step 5: Collect logs

After completing table creation, you can proceed with log collection.

Apache Doris provides open and versatile Stream HTTP APIs, through which you can connect with popular log collectors such as Logstash, Filebeat, Kafka, and others to carry out log collection work. This section explains

how to integrate these log collectors using the Stream HTTP APIs.

Integrating Logstash

Follow these steps:

1. Download and install the Logstash Doris Output plugin. You can choose one of the following two methods:
    - Click to download and install.
    - Compile from the source code and run the following command to install:

```
./bin/logstash-plugin install logstash-output-doris-1.2.0.gem
```

2. Configure Logstash. Specify the following fields:

- `logstash.yml`: Used to configure Logstash batch processing log sizes and timings for improved data writing performance.

```
Plain Text pipeline.batch.size: 1000000 pipeline.batch.delay: 10000
```

- `logstash_demo.conf`: Used to configure the specific input path of the collected logs and the settings for output to Apache Doris.

```
input {
    file {
    path => "/path/to/your/log"
  }
}

output {
  doris {
    http_hosts => [ "<http://fehost1:http_port>", "<http://fehost2:http_port>", "<http://fehost3:http_port">]
    user => "your_username"
    password => "your_password"
    db => "your_db"
    table => "your_table"

    # doris stream load http headers
    headers => {
    "format" => "json"
    "read_json_by_line" => "true"
    "load_to_single_tablet" => "true"
    }

    # field mapping: doris fileld name => logstash field name
    # %{} to get a logstash field, [] for nested field such as [host][name] for host.name
    mapping => {
    "ts" => "%{@timestamp}"
    "host" => "%{[host][name]}"
    "path" => "%{[log][file][path]}"
    "message" => "%{message}"
    }
    log_request => true
    log_speed_interval => 10
  }
}
    ```
```
```

```
3. Run Logstash according to the command below, collect logs, and output to Apache Doris.

```shell
./bin/logstash -f logstash_demo.conf
```

For more information about the Logstash Doris Output plugin, see Logstash Doris Output Plugin.

Integrating Filebeat

Follow these steps:

1. Obtain the Filebeat binary file that supports output to Apache Doris. You can click to download or compile it from the Apache Doris source code.

2. Configure Filebeat. Specify the filebeat_demo.yml field that is used to configure the specific input path of the collected logs and the settings for output to Apache Doris.

```
# input
filebeat.inputs:
- type: log
enabled: true
paths:
    - /path/to/your/log
multiline:
    type: pattern
    pattern: '^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}'
    negate: true
    match: after
    skip_newline: true

processors:
- script:
    lang: javascript
    source: >
        function process(event) {
            var msg = event.Get("message");
            msg = msg.replace(/\t/g, "  ");
            event.Put("message", msg);
        }
- dissect:
    # 2024-06-08 18:26:25,481 INFO (report-thread|199) [ReportHandler.cpuReport():617] begin to handle
    tokenizer: "%{day} %{time} %{log_level} (%{thread}) [%{position}] %{content}"
    target_prefix: ""
    ignore_failure: true
    overwrite_keys: true

# queue and batch
queue.mem:
events: 1000000
flush.min_events: 100000
flush.timeout: 10s

# output
output.doris:
fenodes: [ "http://fehost1:http_port", "http://fehost2:http_port", "http://fehost3:http_port" ]
user: "your_username"
password: "your_password"
database: "your_db"
table: "your_table"
# output string format
```
```

```
codec_format_string: '{"ts": "%{[day]} %{[time]}", "host": "%{[agent][hostname]}", "path": "%{[log][file][path]}", "message": "%
{[message]}"}'
headers:
    format: "json"
    read_json_by_line: "true"
    load_to_single_tablet: "true"
```

3. Run Filebeat according to the command below, collect logs, and output to Apache Doris.

```
chmod +x filebeat-doris-2.1.1
./filebeat-doris-2.1.1 -c filebeat_demo.yml
```

For more information about Filebeat, refer to Beats Doris Output Plugin.

Integrating Kafka

Write JSON formatted logs to Kafka's message queue, create a Kafka Routine Load, and allow Apache Doris to actively pull data from Kafka.

You can refer to the example below, where `property.*` represents Librdkafka client-related configurations and needs to be adjusted according to the actual Kafka cluster situation.

```
CREATE ROUTINE LOAD load_log_kafka ON log_db.log_table
COLUMNS(ts, clientip, request, status, size)
PROPERTIES (
"max_batch_interval" = "60",
"max_batch_rows" = "20000000",
"max_batch_size" = "1073741824",
"load_to_single_tablet" = "true",
"format" = "json"
)
FROM KAFKA (
"kafka_broker_list" = "host:port",
"kafka_topic" = "log__topic_",
"property.group.id" = "your_group_id",
"property.security.protocol"="SASL_PLAINTEXT",
"property.sasl.mechanism"="GSSAPI",
"property.sasl.kerberos.service.name"="kafka",
"property.sasl.kerberos.keytab"="/path/to/xxx.keytab",
"property.sasl.kerberos.principal"="<xxx@yyy.com>"
);
<br />SHOW ROUTINE LOAD;
```

For more information about Kafka, see Routine Load.

Using customized programs to collect logs

In addition to integrating common log collectors, you can also customize programs to import log data into Apache Doris using the Stream Load HTTP API. Refer to the following code:

```
curl
--location-trusted
-u username:password
-H "format:json"
-H "read_json_by_line:true"
-H "load_to_single_tablet:true"
-H "timeout:600"
```

```
-T logfile.json
http://fe_host:fe_http_port/api/log_db/log_table/_stream_load
```

When using custom programs, pay attention to the following key points:

- Use Basic Auth for HTTP authentication, calculate using the command echo -n 'username:password' | base64.
- Set HTTP header "format:json" to specify the data format as JSON.
- Set HTTP header "read_json_by_line:true" to specify one JSON per line.
- Set HTTP header "load_to_single_tablet:true" to import data into one bucket at a time to reduce small file imports.
- It is recommended to write batches whose sizes are between 100MB to 1GB on the client side. For Apache Doris version 2.1 and higher, you need to reduce batch sizes on the client side through the Group Commit function.

Step 6: Query and analyze logs

Query logs

Apache Doris supports standard SQL, so you can connect to the cluster through MySQL client or JDBC to execute SQL for log queries.

Plain Text `mysql -h fe_host -P fe_mysql_port -u your_username -Dyour_db_name`

Here are 5 common SQL query commands for reference:

- View the latest 10 log entries

```
SELECT * FROM your_table_name ORDER BY ts DESC LIMIT 10;
```

- Query the latest 10 log entries with the host as 8.8.8.8

```
SELECT * FROM your_table_name WHERE host = '8.8.8.8' ORDER BY ts DESC LIMIT 10;
```

- Retrieve the latest 10 log entries with error or 404 in the request field. In the command below, MATCH_ANY is a full-text search SQL syntax used by Apache Doris for matching any keyword in the fields.

```
SELECT * FROM your_table_name WHERE message **MATCH_ANY** 'error 404'
ORDER BY ts DESC LIMIT 10;
```

- Retrieve the latest 10 log entries with image and faq in the request field. In the command below, MATCH_ALL is a full-text search SQL syntax used by Apache Doris for matching all keywords in the fields.

```
SELECT * FROM your_table_name WHERE message **MATCH_ALL** 'image faq'
```

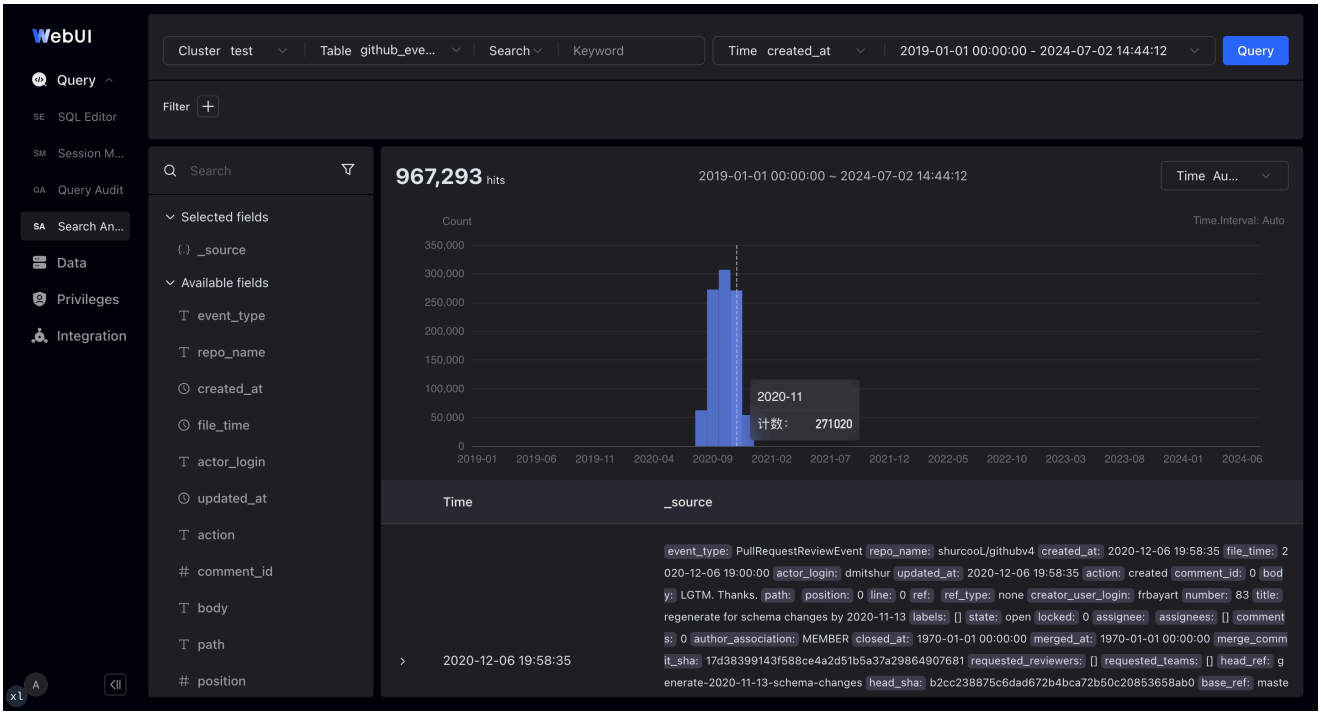
```
ORDER BY ts DESC LIMIT 10;
```

- Retrieve the latest 10 entries with image and faq in the request field. In the following command, MATCH_PHRASE is a full-text search SQL syntax used by Apache Doris for matching all keywords in the fields and requiring consistent order. In the example below, a image faq b can match, but a faq image b cannot match because the order of image and faq does not match the syntax.

```
SELECT * FROM your_table_name WHERE message **MATCH_PHRASE** 'image faq'
ORDER BY ts DESC LIMIT 10;
```

Analyze logs visually

Some third-party vendors offer visual log analysis development platforms based on Apache Doris, which include a log search and analysis interface similar to Kibana Discover. These platforms provide an intuitive and user-friendly exploratory log analysis interaction.



WebUI

- Support for full-text search and SQL modes
- Support for selecting query log timeframes with time boxes and histograms
- Display of detailed log information, expandable into JSON or tables
- Interactive clicking to add and remove filter conditions in the log data context
- Display of top field values in search results for finding anomalies and further drilling down for analysis

Please contact dev@doris.apache.org to find more.

Get Profile

Background

We often encounter situations where the execution time of the corresponding SQL is less than expected. In order to optimize the SQL to achieve the expected Query delay, through the Profile we can see what optimizations can be done. Now we will explain how to get the Profile corresponding to the Query in different environments.

Request

`HTTP://FE_IP:HTTP_PORT GET /API/Profile`

The Doris cluster can normally access the external network

1. Enable profile reporting parameters enable_profile

   This parameter turns on the session variable. It is not recommended to turn this variable on globally.

```
--open variable
mysql> set enable_profile=true;
Query OK, 0 rows affected (0.00 sec)

--Confirm whether the variable is opened normally
mysql> show variables like '%profile%';
+----------------+-------+---------------+---------+
| Variable_name  | Value | Default_Value | Changed |
+----------------+-------+---------------+---------+
| enable_profile | true  | false         | 1       |
+----------------+-------+---------------+---------+
1 row in set (0.00 sec)
```

2. Execute the corresponding query

   In the case of a cluster with multiple fes, you need to execute the corresponding query on the fes that enable profile reporting parameters. The parameters do not take effect globally.

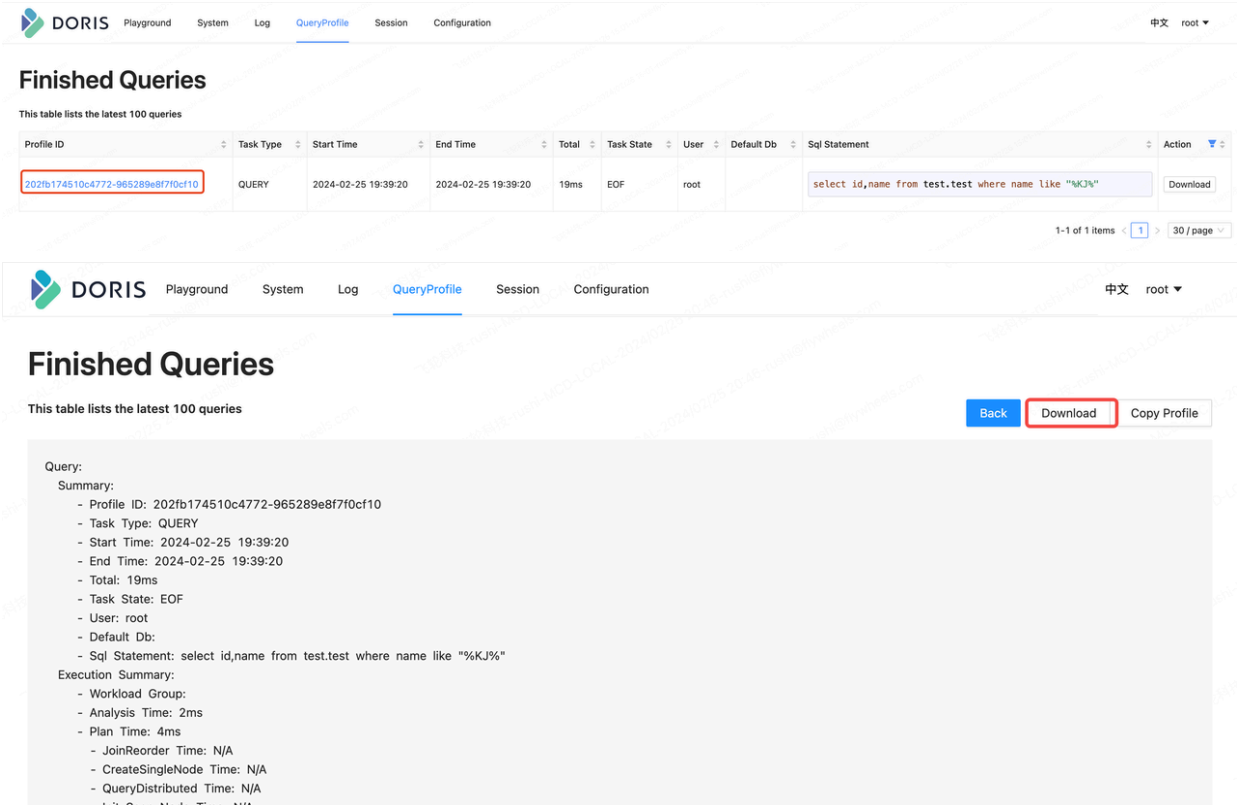
```
--open variable
mysql> set enable_profile=true;
Query OK, 0 rows affected (0.00 sec)
--Confirm whether the variable is opened normally
mysql> show variables like '%profile%';
+----------------+-------+---------------+---------+
| Variable_name  | Value | Default_Value | Changed |
+----------------+-------+---------------+---------+
| enable_profile | true  | false         | 1       |
+----------------+-------+---------------+---------+
1 row in set (0.00 sec)

--Execute the corresponding Query
mysql> select id,name from test.test where name like "%RuO%";
+--------------------------+-------------------------------------------------------------------------------------------------------------
```

```
------------------------------+
| id                          | name
|
+---------------------------+-------------------------------------------------------------------------------------------------------
------------------------------+
| 1ZWXYGbb8nr5Pi29J4cEMyEMb |
ZN1nqzBRSl1rTrr99rnX1aplxhRuOUTLw6so7rzjlRQ317gTPxhOdHljmrARDJjH7FjRkJW9c7YuUBmWikq7eNgmFKJPreWirDrGrFzUYH4eP6kDtSA3UTnNIIj
|
+---------------------------+-------------------------------------------------------------------------------------------------------
------------------------------+
1 row in set (0.01 sec)
```

3. Get Profile

   When the cluster has multiple fes, you need to access the QueryProfile page of the FE HTTP interface (HTTP://FE_IP:HTTP_PORT) that executes the corresponding query. Click the corresponding Profile ID to view the corresponding Profile. You can also download the corresponding Profile in the Profile interface.



Doris cluster’s access to the external network is restricted

When the cluster cannot access the external network normally, it needs to obtain the corresponding profile through the API (HTTP://FE_IP:HTTP_PORT/API/Profile?Query_ID=). The IP and PORT refer to the IP and PORT corresponding to FE that executes the corresponding Query. At this time, the first two steps of obtaining the Profile corresponding to the Query are the same as when accessing the external network normally. There will be a difference in the third step of obtaining the Profile.

Get Profile

- Find the corresponding Profile ID

```
--Find the Profile ID according to the corresponding query
mysql> show query profile "/";
+----------------------------------+-----------+---------------------+---------------------+-------+------------+------+--
----------+-----------------------------------------------------------+
| Profile ID                       | Task Type | Start Time          | End Time            | Total | Task State | User |
Default Db | Sql Statement                                             |
+----------------------------------+-----------+---------------------+---------------------+-------+------------+------+--
----------+-----------------------------------------------------------+
| 1b0bb22689734d30-bbe56e17c2ff21dc | QUERY     | 2024-02-28 11:00:17 | 2024-02-28 11:00:17 | 7ms   | EOF        | root |
| select id,name from test.test where name like "%RuO%" |
| 202fb174510c4772-965289e8f7f0cf10 | QUERY     | 2024-02-25 19:39:20 | 2024-02-25 19:39:20 | 19ms  | EOF        | root |
| select id,name from test.test where name like "%KJ%"  |
+----------------------------------+-----------+---------------------+---------------------+-------+------------+------+--
----------+-----------------------------------------------------------+
2 rows in set (0.00 sec)
```

- Query Profile and redirect Profile to a text

```
template: curl -X GET -u user:password http://fe_ip:http_port/api/profile?query_id=1b0bb22689734d30-bbe56e17c2ff21dc >
test.profile

[user@VM-10-6-centos profile]$ curl -X GET -u root:root http://127.0.0.1:8030/api/profile?query_id=1b0bb22689734d30-
bbe56e17c2ff21dc > test.profile
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1211    0  1211    0     0   168k      0 --:--:-- --:--:-- --:--:--  168k
```

- The returned Profile line break is , which is inconvenient to analyze. You can replace with in a text editing tool.

```
[user@VM-10-6-centos profile]$ cat test.profile
{"msg":"success","code":0,"data":{"profile":"Query:\n  Summary:\n
- Profile ID: 1b0bb22689734d30-bbe56e17c2ff21dc\n     - Task Type: QUERY\n
- Start Time: 2024-02-28 11:00:17\n     - End Time: 2024-02-28 11:00:17\n
- Total: 7ms\n     - Task State: EOF\n     - User: root\n     - Default Db: \n
- Sql Statement: select id,name from test.test where name like \"%RuO%\"\n  Execution Summary:\n
- Workload Group: \n     - Analysis Time: 1ms\n
- Plan Time: 2ms\n       - JoinReorder Time: N/A\n
- CreateSingleNode Time: N/A\n       - QueryDistributed Time: N/A\n
- Init Scan Node Time: N/A\n       - Finalize Scan Node Time: N/A\n
- Get Splits Time: N/A\n           - Get PARTITIONS Time: N/A\n
- Get PARTITION FILES Time: N/A\n         - Create Scan Range Time: N/A\n
- Schedule Time: N/A\n     - Fetch Result Time: 0ms\n     - Write Result Time: 0ms\n
- Wait and Fetch Result Time: N/A\n     - Doris Version: doris-2.0.4-rc06-003a815b63\n
- Is Nereids: Yes\n     - Is Pipeline: Yes\n     - Is Cached: Yes\n
- Total Instances Num: 0\n     - Instances Num Per BE: \n
- Parallel Fragment Exec Instance Num: 48\n     - Trace ID: \n"},"count":0}
```

- The effect after replacement is as follows

```
{"msg":"success","code":0,"data":{"profile":"Query:
  Summary:
     - Profile ID: 1b0bb22689734d30-bbe56e17c2ff21dc
     - Task Type: QUERY
     - Start Time: 2024-02-28 11:00:17
     - End Time: 2024-02-28 11:00:17
     - Total: 7ms
     - Task State: EOF
     - User: root
```

```
      - Default Db:
      - Sql Statement: select id,name from test.test where name like \"%RuO%\"
    Execution Summary:
      - Workload Group:
      - Analysis Time: 1ms
      - Plan Time: 2ms
        - JoinReorder Time: N/A
        - CreateSingleNode Time: N/A
        - QueryDistributed Time: N/A
        - Init Scan Node Time: N/A
        - Finalize Scan Node Time: N/A
          - Get Splits Time: N/A
            - Get PARTITIONS Time: N/A
            - Get PARTITION FILES Time: N/A
          - Create Scan Range Time: N/A
      - Schedule Time: N/A
      - Fetch Result Time: Oms
      - Write Result Time: Oms
      - Wait and Fetch Result Time: N/A
      - Doris Version: doris-2.0.4-rc06-003a815b63
      - Is Nereids: Yes
      - Is Pipeline: Yes
      - Is Cached: Yes
      - Total Instances Num: 0
      - Instances Num Per BE:
      - Parallel Fragment Exec Instance Num: 48
      - Trace ID:
"},"count":0}
```

Overview

SECOND_TIMESTAMP

Description

The function converts a `DATETIME` value into a Unix timestamp (in seconds) starting from `1970-01-01 00:00:00 UTC`.

Syntax

```
SECOND_TIMESTAMP(<datetime>)
```

Parameters

Parameter	Description
`<datetime>`	Required. The DATETIME value to be converted into a Unix timestamp.

Return Value

- Returns an integer representing the Unix timestamp (in seconds) corresponding to the input datetime value.
- If `<datetime>` is NULL, the function returns NULL.
- If `<datetime>` is out of valid range, the function may return an error or unexpected value.

Example

```
SELECT SECOND_TIMESTAMP('2025-01-23 12:34:56');
```

```
+------------------------------------------------------------+
| second_timestamp(cast('2025-01-23 12:34:56' as DATETIMEV2(0))) |
+------------------------------------------------------------+
|                                                 1737606896 |
+------------------------------------------------------------+
```

CLEAN-PROFILE

CLEAN-PROFILE

Name

CLEAN PROFILE

Description

For manual cleanup all of historical query or load profile.

Syntax:

```
CLEAN ALL PROFILE;
```

Example

1. Clean all profile

    ```
    CLEAN ALL PROFILE;
    ```

Keywords

```
CLEAN, PROFILE
```

Best Practice

SHOW-CONVERT-LIGHT-SCHEMA-CHANGE-PROCESS

SHOW-CONVERT-LIGHT-SCHEMA-CHANGE-PROCESS

Name

SHOW CONVERT LIGHT SCHEMA CHANGE PROCESS

Description

This statement is used to show the process of converting light schema change process. should enable config `enable_convert_light_weight_schema_change`.

grammar:

```
SHOW CONVERT_LIGHT_SCHEMA_CHANGE_PROCESS [FROM db]
```

Example

1. View the converting process in db named test

   ```
   SHOW CONVERT_LIGHT_SCHEMA_CHANGE_PROCESS FROM test;
   ```

2. View the converting process globally

   ```
   SHOW CONVERT_LIGHT_SCHEMA_CHANGE_PROCESS;
   ```

Keywords

```
SHOW, CONVERT_LIGHT_SCHEMA_CHANGE_PROCESS
```

Best Practice

SHOW-DATA-TYPES

SHOW-DATA-TYPES

Name

SHOW DATA TYPES

Description

````
This statement is used to view all supported data types.

grammar:
    ```sql
    SHOW DATA TYPES;
    ```
````

Example

```
1. Display all supported data types

    SHOW DATA TYPES;
```

Keywords

```
SHOW,DATA,TYPES
```

Best Practice

SHOW-JOB-TASK

SHOW-JOB-TASK

Name

SHOW JOB TASK

Description

This statement is used to display the list of execution results of JOB subtasks, and the latest 20 records will be kept by default.

grammar:

```
SHOW JOB TASKS FOR job_name;
```

Result description:

```
                        JobId: JobId
                        TaskId: TaskId
                    StartTime: start execution time
                      EndTime: end time
                        Status: status
                        Result: execution result
                        ErrMsg: error message
```

- State

  ```
  There are the following 2 states:
  * SUCCESS
  * FAIL
  ```

Example

1. Display the task execution list of the JOB named test1

   ```
   SHOW JOB TASKS FOR test1;
   ```

###Keywords

```
SHOW, JOB, TASK
```

Best Practice

SHOW-MIGRATIONS

SHOW-MIGRATIONS

Name

SHOW MIGRATIONS

Description

This statement is used to view the progress of database migration

grammar:

```
SHOW MIGRATIONS
```

Example

Keywords

```
SHOW, MIGRATIONS
```

Best Practice

REFRESH-LDAP

REFRESH-LDAP

Name

REFRESH-LDAP

Description

This statement is used to refresh the cached information of LDAP in Doris. The default timeout for LDAP information cache in Doris is 12 hours, which can be viewed by `ADMIN SHOW FRONTEND CONFIG LIKE 'ldap_user_cache_timeout_s';`.

syntax:

```
REFRESH LDAP ALL;
REFRESH LDAP [for user_name];
```

Example

1. refresh all LDAP user cache information

   ```
   REFRESH LDAP ALL;
   ```

2. refresh the cache information for the current LDAP user

   ```
   REFRESH LDAP;
   ```

3. refresh the cache information of the specified LDAP user user1

   ```
   REFRESH LDAP for user1;
   ```

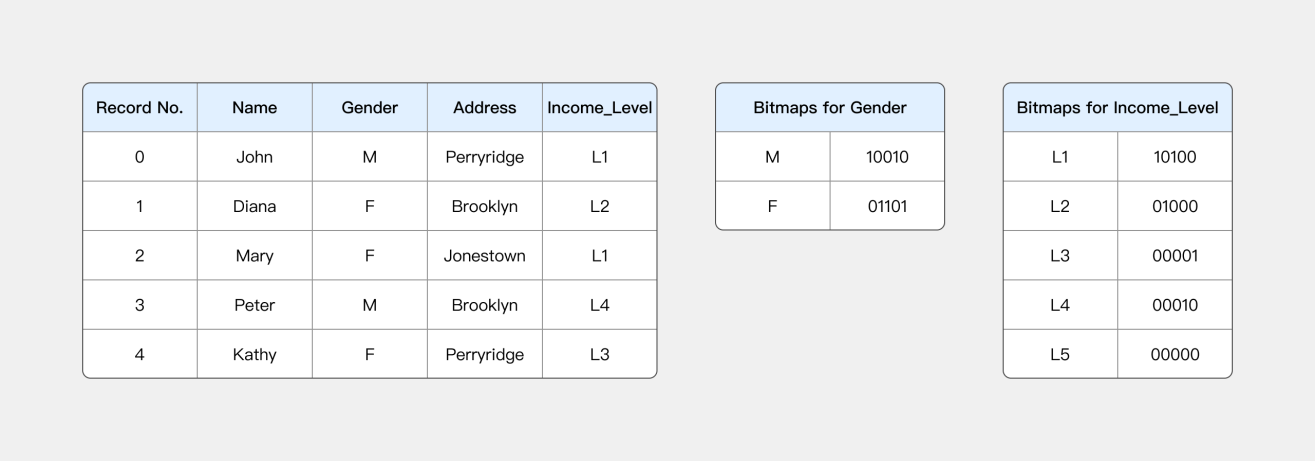
Keywords

REFRESH, LDAP

Best Practice

Bitmap Index

Bitmap Index is an index represented by bitmaps, where a bitmap is created for each key value in a column. Compared to other indexes, it occupies very little storage space and is very fast to create and use. However, it has a disadvantage of having a large lock granularity for modification operations, making it unsuitable for frequent updates.



Record No.	Name	Gender	Address	Income_Level
0	John	M	Perryridge	L1
1	Diana	F	Brooklyn	L2
2	Mary	F	Jonestown	L1
3	Peter	M	Brooklyn	L4
4	Kathy	F	Perryridge	L3

Bitmaps for Gender	
M	10010
F	01101

Bitmaps for Income_Level	
L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

bitmap index

Applicable scenarios

- Columns with high value repetition, recommended to be between 100 and 100,000, such as occupation, city, etc. If the repetition is too high, there is no significant advantage compared to other types of indexes; if the repetition is too low, space efficiency and performance will be greatly reduced.
- Specific types of queries such as `count`, `or`, and `and` logical operations that only require bitwise operations. For example: querying with multiple conditions combined, `select count(*) from table where city = 'Nanjing' and job = 'Doctor' and phonetype = 'iphone' and gender = 'Male'`. If bitmap indexes are established on each query condition column, the database can perform efficient bit operations, accurately locate the required data, reduce disk IO, and the smaller the filtered result set, the more obvious the advantage of bitmap indexes.
- Suitable for ad-hoc queries, multi-dimensional analysis, and other analytical scenarios. If a table has 100 columns and users use 20 of them as query conditions (arbitrarily using several columns from these 20 columns), creating 20 bitmap indexes on these columns will allow all queries to utilize the indexes.

Inapplicable scenarios

- Columns with low value repetition, such as ID cards, phone numbers, etc.
- Columns with excessively high repetition, such as gender. Although bitmap indexes can be established, it is not recommended to use them as query conditions alone. It is recommended to filter them together with other conditions.
- Columns that often need to be updated.

Creating bitmap index

Creating a bitmap index named index_name on the column siteid in the table named table_name:

```
CREATE INDEX [IF NOT EXISTS] index_name ON table1 (siteid) USING BITMAP;
```

Viewing bitmap index

Displaying indexes under the specified table_name:

```
SHOW INDEX FROM table_name;
```

Deleting index

Deleting the index named index_name under the specified table_name:

```
DROP INDEX [IF EXISTS] index_name ON table_name;
```

Notes:

- Bitmap indexes are only created on single columns.
- Bitmap indexes can be applied to all columns in Duplicate and Uniq data models and key columns in the Aggregate model.
- The data types supported by bitmap indexes are as follows:
    - TINYINT
    - SMALLINT
    - INT
    - BIGINT
    - CHAR
    - VARCHAR
    - DATE
    - DATETIME
    - LARGEINT
    - DECIMAL
    - BOOL