Apache Doris V1.2 Documentation

20250117

Content

1 Getting Started

1.1 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 4000 companies worldwide, including giants such as TikTok, Baidu, Cisco, Tencent, and NetEase. It is also widely used across industries from finance, retailing, and telecommunications to energy, manufacturing, medical care, etc.

1.1.1 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:

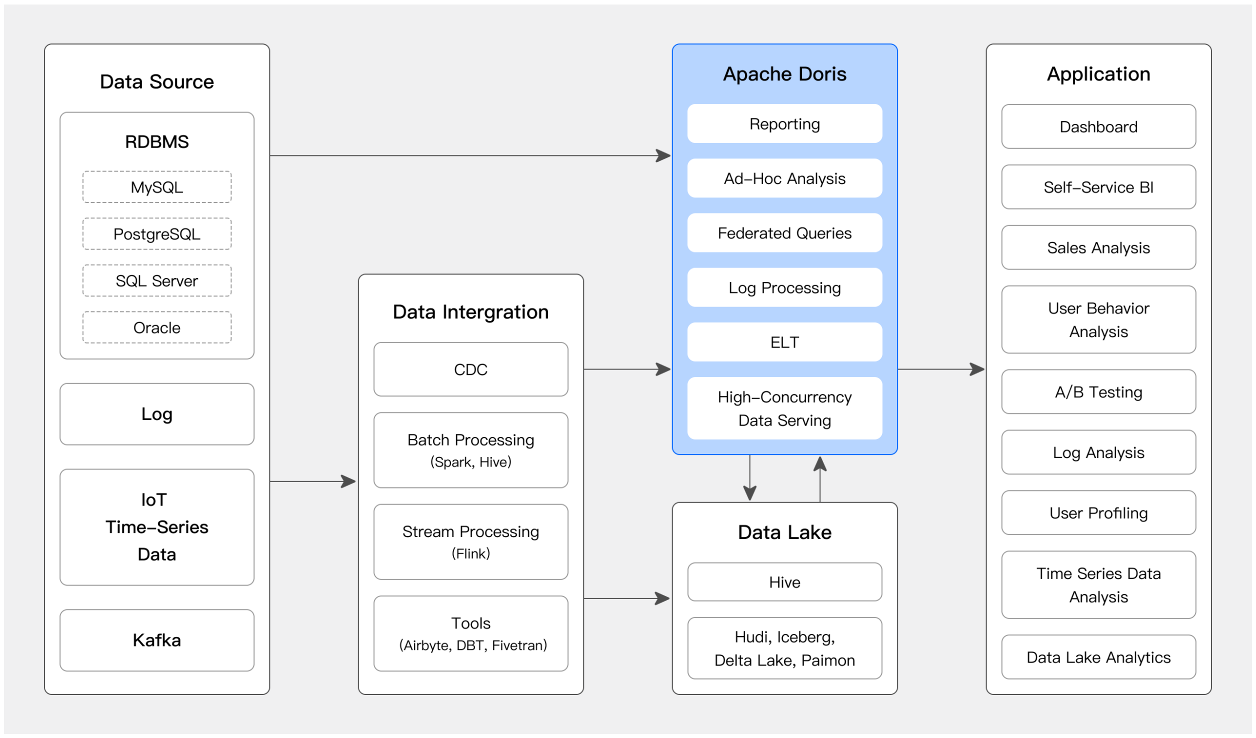


Figure 1: Apache Doris usage scenarios

- 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.

1.1.2 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.

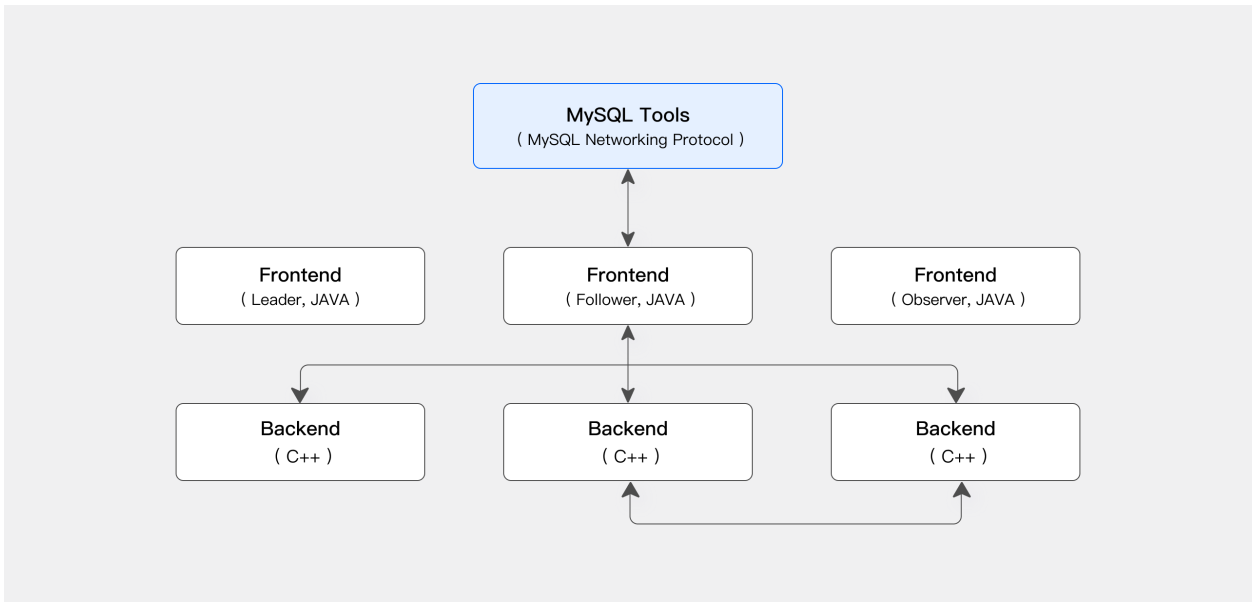


Figure 2: Technical overview

1.1.3 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.

1.1.4 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.

1.1.5 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.

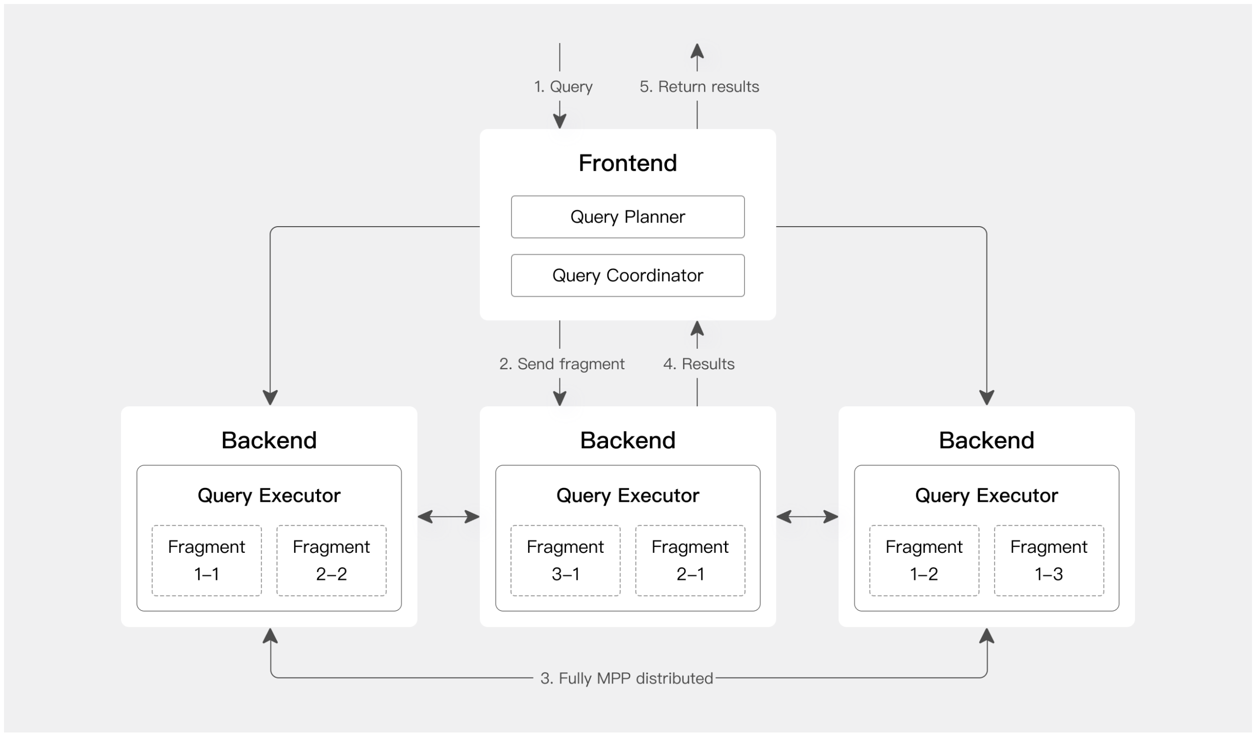


Figure 3: Query engine

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.

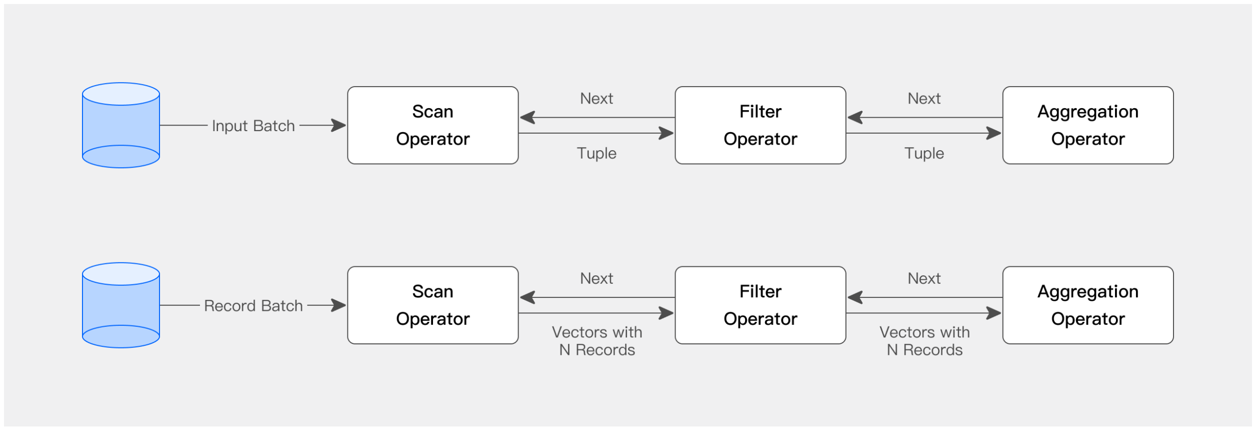


Figure 4: Query engine

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.

1.2 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.

1.2.1 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).

1.2.2 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://apache-doris-releases.oss-accelerate.aliyuncs.com/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
```

1.2.3 Install Apache Doris

1.2.3.1 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
```

1.2.3.2 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
```

1.2.3.3 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
```

1.2.3.4 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
```

1.2.3.5 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.

1.2.3.6 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.

1.2.3.7 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.

1.2.4 Create database and table

1.2.4.1 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.

1.2.4.2 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');
```

1.2.4.3 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.

1.2.4.4 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)
```

1.2.5 Stop Apache Doris

1.2.5.1 Stop FE

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

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

1.2.5.2 Stop BE

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

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

2 Guides

2.1 Doris Introduction

2.1.1 Introduction to Apache Doris

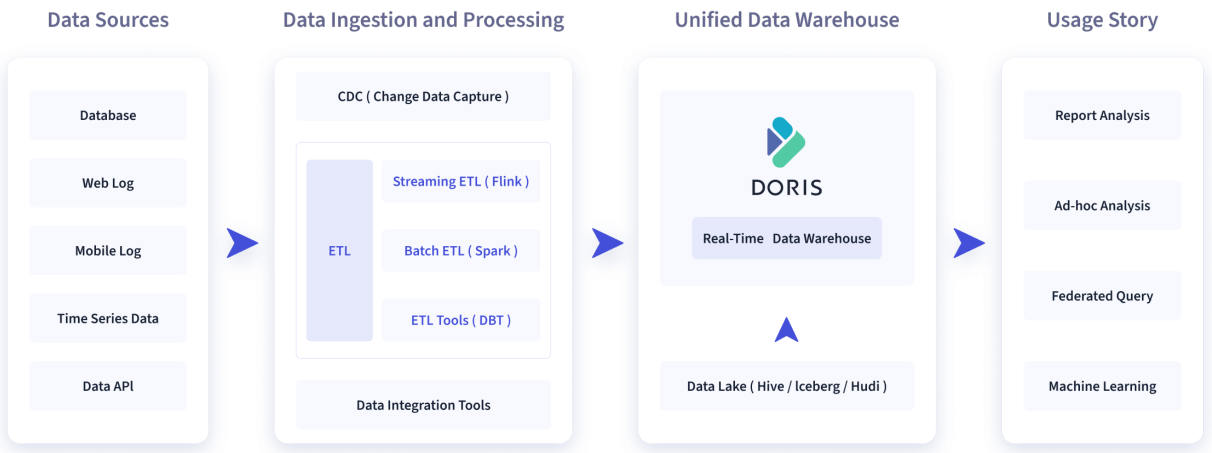
Apache Doris is a high-performance, real-time analytical database based on MPP architecture, known for its extreme speed and ease of use. It only requires a sub-second response time to return query results under massive data and can support not only high-concurrent point query scenarios but also high-throughput complex analysis scenarios. All this makes Apache Doris an ideal tool for scenarios including report analysis, ad-hoc query, unified data warehouse, and data lake query acceleration. On Apache Doris, users can build various applications, such as user behavior analysis, AB test platform, log retrieval analysis, user portrait analysis, and 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 Foundation for incubation in July 2018, where it was operated by members of the incubator project management committee under the guidance of Apache mentors. Currently, the Apache Doris community has gathered more than 400 contributors from hundreds companies in different industries, and the number of active contributors is more than 100 per month. In June 2022, Apache Doris graduated from Apache incubator as a Top-Level Project.

Apache Doris now has a wide user base in China and around the world, and as of today, Apache Doris is used in production environments in over 2000 companies worldwide. Of the top 50 Chinese Internet companies by market capitalization (or valuation), more than 80% are long-term users of Apache Doris, including Baidu, Meituan, Xiaomi, Jingdong, Bytedance, Tencent, NetEase, Kwai, Weibo, and Ke Holdings. It is also widely used in some traditional industries such as finance, energy, manufacturing, and telecommunications.

2.1.2 Usage Scenarios

As shown in the figure below, after various data integration and processing, the data sources are usually stored in the real-time data warehouse Doris and the offline data lake or data warehouse (in Apache Hive, Apache Iceberg or Apache Hudi).



Apache Doris is widely used in the following scenarios:

- Reporting Analysis
  - Real-time dashboards
  - Reports for in-house analysts and managers
  - Highly concurrent user-oriented or customer-oriented report analysis: such as website analysis and ad reporting that usually require thousands of QPS and quick response times measured in miliseconds. A successful user case is that Doris has been used by the Chinese e-commerce giant JD.com in ad reporting, where it receives 10 billion rows of data per day, handles over 10,000 QPS, and delivers a 99 percentile query latency of 150 ms.
- Ad-Hoc Query. Analyst-oriented self-service analytics with irregular query patterns and high throughput requirements. XiaoMi has built a growth analytics platform (Growth Analytics, GA) based on Doris, using user behavior data for business growth analysis, with an average query latency of 10 seconds and a 95th percentile query latency of 30 seconds or less, and tens of thousands of SQL queries per day.
- Unified Data Warehouse Construction. Apache Doris allows users to build a unified data warehouse via one single platform and save the trouble of handling complicated software stacks. Chinese hot pot chain Haidilao has built a unified data warehouse with Doris to replace its old complex architecture consisting of Apache Spark, Apache Hive, Apache Kudu, Apache HBase, and Apache Phoenix.

- Data Lake Query. Apache Doris avoids data copying by federating the data in Apache Hive, Apache Iceberg, and Apache Hudi using external tables, and thus achieves outstanding query performance.

2.1.3 Technical Overview

As shown in the figure below, the Apache Doris architecture is simple and neat, with only two types of processes.

- Frontend (FE): user request access, query parsing and planning, metadata management, node management, etc.
- Backend (BE): data storage and query plan execution

Both types of processes are horizontally scalable, and a single cluster can support up to hundreds of machines and tens of petabytes of storage capacity. And these two types of processes guarantee high availability of services and high reliability of data through consistency protocols. This highly integrated architecture design greatly reduces the operation and maintenance cost of a distributed system.

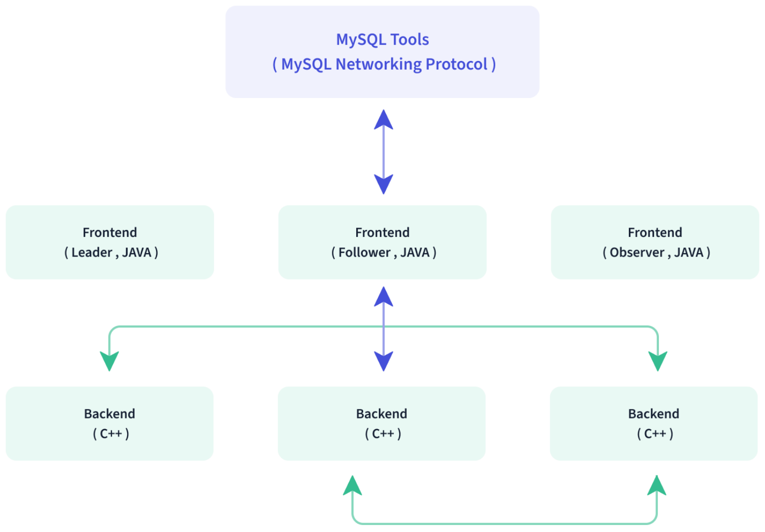


Figure 5: Image description

In terms of interfaces, Apache Doris adopts MySQL protocol, supports standard SQL, and is highly compatible with MySQL dialect. Users can access Doris through various client tools and it supports seamless connection with BI tools.

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

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.
- Z-order Index: This allows users to efficiently run range queries on any combination of fields in their schema.

- MIN/MAX Indexing: This enables effective filtering of equivalence and range queries for numeric types.
- Bloom Filter: very effective in equivalence filtering and pruning of high cardinality columns
- Invert Index: This enables fast search for any field.

Doris supports a variety of storage models and has optimized them for different scenarios:

- Aggregate Key Model: able to merge the value columns with the same keys and significantly improve performance
- Unique Key Model: Keys are unique in this model and data with the same key will be overwritten to achieve row-level data updates.
- Duplicate Key Model: This is a detailed data model capable of detailed storage of fact tables.

Doris also supports strongly consistent materialized views. Materialized views are automatically selected and updated, which greatly reduces maintenance costs for users.

Doris adopts the MPP model in its query engine to realize parallel execution between and within nodes. It also supports distributed shuffle join for multiple large tables so as to handle complex queries.

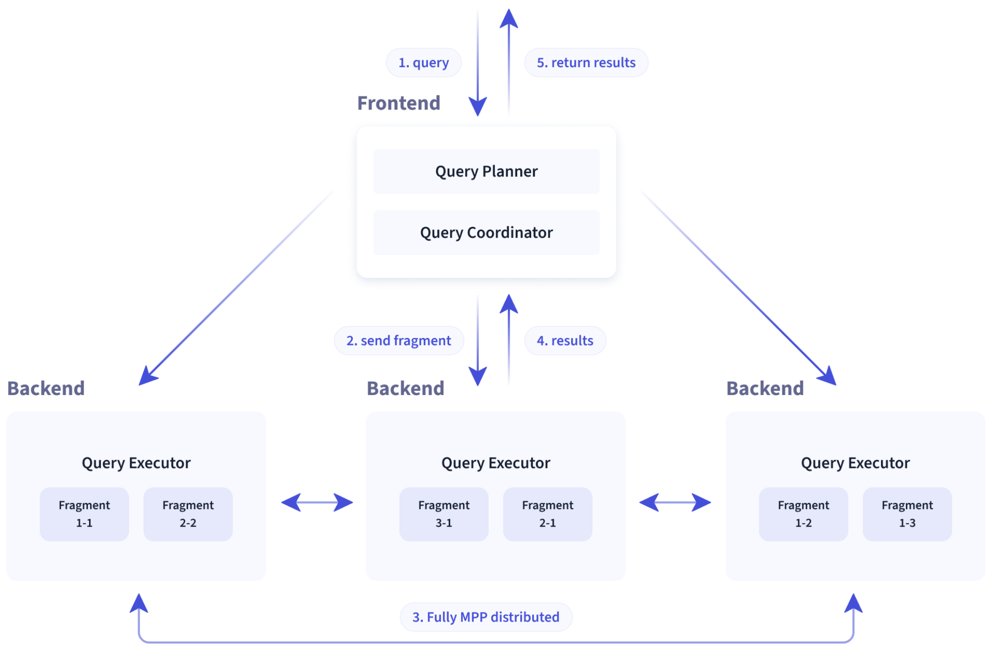


Figure 6: Image description

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

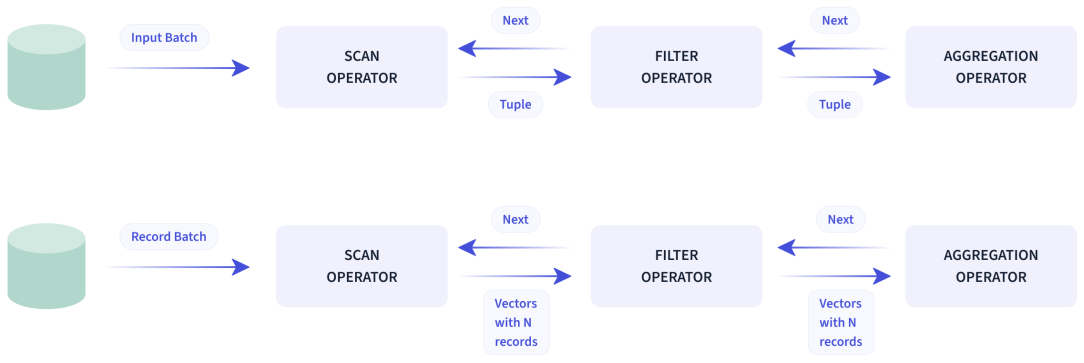


Figure 7: Image description

Apache Doris uses Adaptive Query Execution technology to dynamically adjust the execution plan based on runtime statistics. For example, it can generate runtime filter, push it to the probe side, and automatically penetrate it to the Scan node at the bottom, which drastically reduces the amount of data in the probe and increases join performance. The runtime filter in Doris supports In/Min/Max/Bloom filter.

In terms of optimizers, Doris uses a combination of CBO and RBO. RBO supports constant folding, subquery rewriting, predicate pushdown and CBO supports Join Reorder. The Doris CBO is under continuous optimization for more accurate statistical information collection and derivation, and more accurate cost model prediction.

2.2 Install And Deploy

2.2.1 Standard deployment

This topic is about the hardware and software environment needed to deploy Doris, the recommended deployment mode, cluster scaling, and common problems occur in creating and running clusters.

Before continue reading, you might want to compile Doris following the instructions in the General Compile topic.

2.2.1.1 Software and Hardware Requirements

2.2.1.1.1 Overview

Doris, as an open source OLAP database with an MPP architecture, can run on most mainstream commercial servers. For you to take full advantage of the high concurrency and high availability of Doris, we recommend that your computer meet the following requirements:

Linux Operating System Version Requirements

Linux System	Version
Centos	7.1 and above

Linux System	Version
Ubuntu	16.04 and above

Software Requirements

Soft	Version
Java	1.8 and above
GCC	4.8.2 and above

OS Installation Requirements

Set the maximum number of open file descriptors in the system

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

Clock synchronization

The metadata in Doris requires a time precision of less than 5000ms, so all machines in all clusters need to synchronize their clocks to avoid service exceptions caused by inconsistencies in metadata caused by clock problems.

Close the swap partition

The Linux swap partition can cause serious performance problems for Doris, so you need to disable the swap partition before installation.

Linux file system

Both ext4 and xfs file systems are supported.

Development Test Environment

Module	CPU	Memory	Disk	Network	Number of Instances
Frontend	8 core +	8GB +	SSD or SATA, 10GB + *	Gigabit Network Card	1
Backend	8 core +	16GB +	SSD or SATA, 50GB + *	Gigabit Network Card	1-3*

Production Environment

Module	CPU	Memory	Disk	Network	Number of Instances (Minimum Requirements)
Frontend	16 core +	64GB +	SSD or RAID card, 100GB + *	10,000 Mbp network card	1-3*
Backend	16 core +	64GB +	SSD or SATA, 100G + *	10-100 Mbp network card	3 *

Note 1:

1. The disk space of FE is mainly used to store metadata, including logs and images. It usually ranges from several hundred MB to several GB.
2. The disk space of BE is mainly used to store user data. The total disk space taken up is 3 times the total user data (3 copies). Then an additional 40% of the space is reserved for background compaction and intermediate data storage.
3. On one single machine, you can deploy multiple BE instances but only one FE instance. If you need 3 copies of the data, you need to deploy 3 BE instances on 3 machines (1 instance per machine) instead of 3 BE instances on one machine). Clocks of the FE servers must be consistent (allowing a maximum clock skew of 5 seconds).
4. The test environment can also be tested with only 1 BE instance. In the actual production environment, the number of BE instances directly determines the overall query latency.
5. Disable swap for all deployment nodes.

Note 2: Number of FE nodes

1. FE nodes are divided into Followers and Observers based on their roles. (Leader is an elected role in the Follower group, hereinafter referred to as Follower, too.)
2. The number of FE nodes should be at least 1 (1 Follower). If you deploy 1 Follower and 1 Observer, you can achieve high read availability; if you deploy 3 Followers, you can achieve high read-write availability (HA).
3. The number of Followers must be odd, and there is no limit on the number of Observers.
4. According to past experience, for business that requires high cluster availability (e.g. online service providers), we recommend that you deploy 3 Followers and 1-3 Observers; for offline business, we recommend that you deploy 1 Follower and 1-3 Observers.

- Usually we recommend 10 to 100 machines to give full play to Doris' performance (deploy FE on 3 of them (HA) and BE on the rest).
- The performance of Doris is positively correlated with the number of nodes and their configuration. With a minimum of four machines (one FE, three BEs; hybrid deployment of one BE and one Observer FE to provide metadata backup) and relatively low configuration, Doris can still run smoothly.
- In hybrid deployment of FE and BE, you might need to be watchful for resource competition and ensure that the metadata catalogue and data catalogue belong to different disks.

Broker Deployment

Broker is a process for accessing external data sources, such as hdfs. Usually, deploying one broker instance on each machine should be enough.

Network Requirements

Doris instances communicate directly over the network. The following table shows all required ports.

Instance Name	Port Name	Default Port	Communication Direction	Description
BE	be_port	9060	FE –> BE	Thrift server port on BE for receiving requests from FE
BE	webserver_port	8040	BE <–> BE	HTTP server port on BE
BE	heartbeat_service_port	9050	FE –> BE	Heart beat service port (thrift) on BE, used to receive heartbeat from FE
BE	brpc_port	8060	FE <–> BE, BE <–> BE	BRPC port on BE for communication between BEs
FE	http_port	8030	FE <–> FE, user <–> FE	HTTP server port on FE
FE	rpc_port	9020	BE –> FE, FE <–> FE	Thrift server port on FE; The configurations of each FE should be consistent.
FE	query_port	9030	user <–> 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 port on Broker for receiving requests

Note:

1. When deploying multiple FE instances, make sure that the http_port configuration of each FE is consistent.
2. Make sure that each port has access in its proper direction before deployment.

IP Binding

Because of the existence of multiple network cards, or the existence of virtual network cards caused by the installation of docker and other environments, the same host may have multiple different IPs. Currently Doris does not automatically identify available IPs. So when you encounter multiple IPs on the deployment host, you must specify the correct IP via the `priority_networks` configuration item.

`priority_networks` is a configuration item that both FE and BE have. It needs to be written in fe.conf and be.conf. It is used to tell the process which IP should be bound when FE or BE starts. Examples are as follows:

`priority_networks=10.1.3.0/24`

This is a representation of CIDR. FE or BE will find the matching IP based on this configuration item as their own local IP.

Note: Configuring `priority_networks` and starting FE or BE only ensure the correct IP binding of FE or BE. You also need to specify the same IP in ADD BACKEND or ADD FRONTEND statements, otherwise the cluster cannot be created. For example:

BE is configured as `priority_networks = 10.1.3.0/24'.`.

If you use the following IP in the ADD BACKEND statement: `ALTER SYSTEM ADD BACKEND "192.168.0.1:9050";`

Then FE and BE will not be able to communicate properly.

At this point, you must DROP the wrong BE configuration and use the correct IP to perform ADD BACKEND.

The same works for FE.

Broker currently does not have the `priority_networks` configuration item, nor does it need. Broker's services are bound to 0.0.0.0 by default. You can simply execute the correct accessible BROKER IP when using ADD BROKER.

Table Name Case Sensitivity

By default, table names in Doris are case-sensitive. If you need to change that, you may do it before cluster initialization. The table name case sensitivity cannot be changed after cluster initialization is completed.

See the `lower_case_table_names` section in Variables for details.

2.2.1.2 Cluster Deployment

2.2.1.2.1 Manual Deployment

Deploy FE

- Copy the FE deployment file into the specified node

  Find the Fe folder under the output generated by source code compilation, copy it into the specified deployment path of FE nodes and put it the corresponding directory.

- Configure FE

1. The configuration file is conf/fe.conf. Note: `meta_dir` indicates the metadata storage location. The default value is `${DORIS` ↪ `_HOME}/doris-meta`. The directory needs to be created manually.

   Note: For production environments, it is better not to put the directory under the Doris installation directory but in a separate disk (SSD would be the best); for test and development environments, you may use the default configuration.

2. The default maximum Java heap memory of JAVA_OPTS in fe.conf is 4GB. For production environments, we recommend that it be adjusted to more than 8G.

- Start FE

  `bin/start_fe.sh --daemon`

  The FE process starts and enters the background for execution. Logs are stored in the log/ directory by default. If startup fails, you can view error messages by checking out log/fe.log or log/fe.out.

- For details about deployment of multiple FEs, see the FE scaling section.

Deploy BE

- Copy the BE deployment file into all nodes that are to deploy BE on

  Find the BE folder under the output generated by source code compilation, copy it into to the specified deployment paths of the BE nodes.

- Modify all BE configurations

Modify be/conf/be.conf, which mainly involves configuring `storage_root_path`: data storage directory. By default, under be/storage, the directory needs to be created manually. Use `;` to separate multiple paths (do not add `;` after the last directory).

You may specify the directory storage medium in the path: HDD or SSD. You may also add capacity limit to the end of every path and use `,` for separation. Unless you use a mix of SSD and HDD disks, you do not need to follow the configuration methods in Example 1 and Example 2 below, but only need to specify the storage directory; you do not need to modify the default storage medium configuration of FE, either.

Example 1:

Note: For SSD disks, add `.SSD` to the end of the directory; for HDD disks, add `.HDD`.

```
`storage_root_path=/home/disk1/doris.HDD;/home/disk2/doris.SSD;/home/disk2/doris`
```

Description

```
* 1./home/disk1/doris.HDD: The storage medium is HDD;
* 2./home/disk2/doris.SSD: The storage medium is SSD;
* 3./home/disk2/doris: The storage medium is HDD (default).
```

Example 2:

Note: You do not need to add the `.SSD` or `.HDD` suffix, but to specify the medium in the `storage_root_path` parameter

```
`storage_root_path=/home/disk1/doris,medium:HDD;/home/disk2/doris,medium:SSD`
```

Description

```
* 1./home/disk1/doris,medium:HDD  :  The storage medium is HDD;
* 2./home/disk2/doris,medium:SSD  :  The storage medium is SSD.
```

- BE webserver_port configuration

  If the BE component is installed in hadoop cluster, you need to change configuration `webserver_port=8040` to avoid port used.

- Set JAVA_HOME environment variable

Java UDF is supported since version 1.2, so BEs are dependent on the Java environment. It is necessary to set the JAVA_HOME environment variable before starting. You can also do this by adding `export JAVA_HOME=your_java_home_path` to the first line of the `start_be.sh` startup script.

- Install Java UDF

Because Java UDF is supported since version 1.2, you need to download the JAR package of Java UDF from the official website and put them under the lib directory of BE, otherwise it may fail to start.

- Add all BE nodes to FE

  BE nodes need to be added in FE before they can join the cluster. You can use mysql-client (Download MySQL 5.7) to connect to FE:

`./mysql-client -h fe_host -P query_port -uroot`

`fe_host` is the node IP where FE is located; `query_port` is in fe/conf/fe.conf; the root account is used by default and no password is required in login.

After login, execute the following command to add all the BE host and heartbeat service port:

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

`be_host` is the node IP where BE is located; `heartbeat_service_port` is in be/conf/be.conf.

- Start BE

  `bin/start_be.sh --daemon`

  The BE process will start and go into the background for execution. Logs are stored in be/log/directory by default. If startup fails, you can view error messages by checking out be/log/be.log or be/log/be.out.

- View BE status

  Connect to FE using mysql-client and execute `SHOW PROC '/backends';` to view BE operation status. If everything goes well, the `isAlive`column should be `true`.

(Optional) FS_Broker Deployment

Broker is deployed as a plug-in, which is independent of Doris. If you need to import data from a third-party storage system, you need to deploy the corresponding Broker. By default, Doris provides fs_broker for HDFS reading and object storage (supporting S3 protocol). fs_broker is stateless and we recommend that you deploy a Broker for each FE and BE node.

- Copy the corresponding Broker directory in the output directory of the source fs_broker to all the nodes that need to be deployed. It is recommended to keep the Broker directory on the same level as the BE or FE directories.

- Modify the corresponding Broker configuration

  You can modify the configuration in the corresponding broker/conf/directory configuration file.

- Start Broker

  `bin/start_broker.sh --daemon`

- Add Broker

  To let Doris FE and BE know which nodes Broker is on, add a list of Broker nodes by SQL command.

  Use mysql-client to connect the FE started, and execute the following commands:

  ```
  ALTER SYSTEM ADD BROKER broker_name "broker_host1:broker_ipc_port1","broker_host2:broker_ipc
  ↪ _port2",...;
  ```

  `broker\_host` is the Broker node ip; `broker_ipc_port` is in conf/apache_hdfs_broker.conf in the Broker configuration file.

- View Broker status

  Connect any started FE using mysql-client and execute the following command to view Broker status: `SHOW PROC '/`
  `↪ brokers';`

Note: In production environments, daemons should be used to start all instances to ensure that processes are automatically pulled up after they exit, such as Supervisor. For daemon startup, in Doris 0.9.0 and previous versions, you need to remove the last & symbol in the start_xx.sh scripts. In Doris 0.10.0 and the subsequent versions, you may just call `sh start_xx.sh` directly to start.

2.2.1.3 FAQ

2.2.1.3.1 Process-Related Questions

1. How can we know whether the FE process startup succeeds?

   After the FE process starts, metadata is loaded first. Based on the role of FE, you can see `transfer from UNKNOWN to ↪ MASTER/FOLLOWER/OBSERVER` in the log. Eventually, you will see the `thrift server started` log and can connect to FE through MySQL client, which indicates that FE started successfully.

   You can also check whether the startup was successful by connecting as follows:

   `http://fe_host:fe_http_port/api/bootstrap`

   If it returns:

   `{"status":"OK","msg":"Success"}`

   The startup is successful; otherwise, there may be problems.

> Note: If you can't see the information of boot failure in fe. log, you may check in fe. out.

2. How can we know whether the BE process startup succeeds?

   After the BE process starts, if there have been data there before, it might need several minutes for data index loading.

   If BE is started for the first time or the BE has not joined any cluster, the BE log will periodically scroll the words `waiting ↪ to receive first heartbeat from frontend`, meaning that BE has not received the Master's address through FE's heartbeat and is waiting passively. Such error log will disappear after sending the heartbeat by ADD BACKEND in FE. If the word `master client', get client from cache failed. host:, port: 0, code: 7` appears after receiving the heartbeat, it indicates that FE has successfully connected BE, but BE cannot actively connect FE. You may need to check the connectivity of rpc_port from BE to FE.

   If BE has been added to the cluster, the heartbeat log from FE will be scrolled every five seconds: `get heartbeat, host ↪ :xx. xx.xx.xx, port:9020, cluster id:xxxxxxx`, indicating that the heartbeat is normal.

   Secondly, if the word `finish report task success. return code: 0` is scrolled every 10 seconds in the log, that indicates that the BE-to-FE communication is normal.

   Meanwhile, if there is a data query, you will see the rolling logs and the `execute time is xxx` logs, indicating that BE is started successfully, and the query is normal.

   You can also check whether the startup was successful by connecting as follows:

   `http://be_host:be_http_port/api/health`

   If it returns:

   `{"status": "OK","msg": "To Be Added"}`

   That means the startup is successful; otherwise, there may be problems.

> Note: If you can't see the information of boot failure in be.INFO, you may see it in be.out.

3. How can we confirm that the connectivity of FE and BE is normal after building the system?

   Firstly, you need to confirm that FE and BE processes have been started separately and worked normally. Then, you need to confirm that all nodes have been added through `ADD BACKEND` or `ADD FOLLOWER/OBSERVER` statements.

   If the heartbeat is normal, BE logs will show `get heartbeat, host:xx.xx.xx.xx, port:9020, cluster id: ↪ xxxxx`; if the heartbeat fails, you will see `backend [10001] got Exception: org.apache.thrift.transport ↪ .TTransportException` in FE's log, or other thrift communication abnormal log, indicating that the heartbeat from FE to 10001 BE fails. Here you need to check the connectivity of the FE to BE host heartbeat port.

   If the BE-to-FE communication is normal, the BE log will display the words `finish report task success. return ↪ code: 0`. Otherwise, the words `master client, get client from cache failed` will appear. In this case, you need to check the connectivity of BE to the rpc_port of FE.

4. What is the Doris node authentication mechanism?

   In addition to Master FE, the other role nodes (Follower FE, Observer FE, Backend) need to register to the cluster through the `ALTER SYSTEM ADD` statement before joining the cluster.

   When Master FE is started for the first time, a cluster_id is generated in the doris-meta/image/VERSION file.

   When FE joins the cluster for the first time, it first retrieves the file from Master FE. Each subsequent reconnection between FEs (FE reboot) checks whether its cluster ID is the same as that of other existing FEs. If it is not the same, the FE will exit automatically.

   When BE first receives the heartbeat of Master FE, it gets the cluster ID from the heartbeat and records it in the `cluster_id` file of the data directory. Each heartbeat after that compares that to the cluster ID sent by FE. If the cluster IDs are not matched, BE will refuse to respond to FE's heartbeat.

   The heartbeat also contains Master FE's IP. If the Master FE changes, the new Master FE will send the heartbeat to BE together with its own IP, and BE will update the Master FE IP it saved.

> priority_network
>
> priority_network is a configuration item that both FE and BE have. It is used to help FE or BE identify their own IP addresses in the cases of multi-network cards. priority_network uses CIDR notation: RFC 4632
>
> If the connectivity of FE and BE is confirmed to be normal, but timeout still occurs in creating tables, and the FE log shows the error message `backend does not find. host:xxxx.xxx.XXXX`, this means that there is a problem with the IP address automatically identified by Doris and that the priority_network parameter needs to be set manually.
>
> The explanation to this error is as follows. When the user adds BE through the ADD BACKEND statement, FE recognizes whether the statement specifies hostname or IP. If it is a hostname, FE automatically converts it to an IP address and stores it in the metadata. When BE reports on the completion of the task, it carries its own IP address. If FE finds that the IP address reported by BE is different from that in the metadata, the above error message will show up.
>
> Solutions to this error: 1) Set priority_network in FE and BE separately. Usually FE and BE are in one network segment, so their priority_network can be set to be the same. 2) Put the correct IP address instead of the hostname in the ADD BACKEND statement to prevent FE from getting the wrong IP address.

5. What is the number of file descriptors of a BE process?

The number of file descriptor of a BE process is determined by two parameters: `min_file_descriptor_number`/`max_file_`
↪ `descriptor_number`.

If it is not in the range [`min_file_descriptor_number`, `max_file_descriptor_number`], error will occurs when starting a BE process. You may use the ulimit command to reset the parameters.

The default value of `min_file_descriptor_number` is 65536.

The default value of `max_file_descriptor_number` is 131072.

For example, the command `ulimit -n 65536;` means to set the number of file descriptors to 65536.

After starting a BE process, you can use cat /proc/$pid/limits to check the actual number of file descriptors of the process.

If you have used `supervisord` and encountered a file descriptor error, you can fix it by modifying `minfds` in supervisord.conf.

```
vim /etc/supervisord.conf

minfds=65535                 ; (min. avail startup file descriptors;default 1024)
```

2.2.2 Docker Deployment

2.2.2.1 Build Docker Image

This topic is about how to build a running image of Apache Doris through Dockerfile, so that an Apache Doris image can be quickly pulled in a container orchestration tool or during a quick test to complete the cluster creation.

2.2.2.1.1 Software and Hardware Requirements

Overview

Prepare the production machine before building a Docker image. The platform architecture of the Docker image will be the same as that of the machine. For example, if you use an X86_64 machine to build a Docker image, you need to download the Doris binary program of X86_64, and the Docker image built can only run on the X86_64 platform. The ARM platform (or M1), likewise.

Hardware Requirements

Minimum configuration: 2C 4G

Recommended configuration: 4C 16G

Software Requirements

Docker Version: 20.10 or newer

2.2.2.1.2 Build Docker Image

During Dockerfile scripting, please note that: > 1. Use the official OpenJDK image certified by Docker-Hub as the base parent image (Version: JDK 1.8). > 2. Use the official binary package for download; do not use binary packages from unknown sources. > 3. Use embedded scripts for tasks such as FE startup, multi-FE registration, FE status check, BE startup, registration of BE to FE, and BE status check. > 4. Do not use `--daemon` to start applications in Docker. Otherwise there will be exceptions during the deployment of orchestration tools such as K8S.

Apache Doris 1.2 and the subsequent versions support JavaUDF, so you also need a JDK environment for BE. The recommended images are as follows:

Doris Program	Recommended Base Parent Image
Frontend	openjdk:8u342-jdk
Backend	openjdk:8u342-jdk
Broker	openjdk:8u342-jdk

Script Preparation

In the Dockerfile script for compiling the Docker Image, there are two methods to load the binary package of the Apache Doris program:

1. Execute the download command via wget / curl when compiling, and then start the docker build process.
2. Download the binary package to the compilation directory in advance, and then load it into the docker build process through the ADD or COPY command.

Method 1 can produce a smaller Docker image, but if the docker build process fails, the download operation might be repeated and result in longer build time; Method 2 is more suitable for less-than-ideal network environments.

The examples below are based on Method 2. If you prefer to go for Method 1, you may modify the steps accordingly.

Prepare Binary Package

Please noted that if you have a need for custom development, you need to modify the source code, compile and package it, and then place it in the build directory.

If you have no such needs, you can just download the binary package from the official website. ###### Steps

Build FE

The build environment directory is as follows:

```
└── docker-build                                   // build root directory
    └── fe                                         // FE build directory
        ├── dockerfile                             // dockerfile script
        └── resource                               // resource directory
            ├── init_fe.sh                         // startup and registration script
            └── apache-doris-x.x.x-bin-fe.tar.gz   // binary package
```

1. Create a build environment directory

```
mkdir -p ./docker-build/fe/resource
```

2. Download official binary package/compiled binary package

Copy the binary package to the `./docker-build/fe/resource` directory

3. Write the Dockerfile script for FE

```
# Select the base image
FROM openjdk:8u342-jdk

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

# Download the software into the image (you can modify based on your own needs)
ADD ./resource/apache-doris-fe-${x.x.x}-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-fe-${x.x.x}-bin /opt/apache-doris/fe

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"]
```

After writing, name it `Dockerfile` and save it to the `./docker-build/fe` directory.

4. Write the execution script of FE

You can refer to init_fe.sh.

After writing, name it `init_fe.sh` and save it to the `./docker-build/fe/resouce` directory.

5. Execute the build

Please note that `${tagName}` needs to be replaced with the tag name you want to package and name, such as: `apache-doris` ↪ `:1.1.3-fe`

Build FE:

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

Build BE

1. Create a build environment directory

```
mkdir -p ./docker-build/be/resource
```

2. The build environment directory is as follows:

```
└── docker-build                                              // build root directory
    └── be                                                    // BE build directory
        ├── dockerfile                                        // dockerfile script
        └── resource                                          // resource directory
            ├── init_be.sh                                    // startup and registration
              ↪ script
            └── apache-doris-x.x.x-bin-x86_64/arm-be.tar.gz // binary package
```

3. Write the Dockerfile script for BE

```
# Select the base image
FROM openjdk:8u342-jdk

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

# Download the software into the image (you can modify based on your own needs)
ADD ./resource/apache-doris-be-${x.x.x}-bin-x86_64.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-be-${x.x.x}-bin-x86_64 /opt/apache-doris/be

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"]
```

After writing, name it `Dockerfile` and save it to the `./docker-build/be` directory

4. Write the execution script of BE

You can refer to init_be.sh.

After writing, name it `init_be.sh` and save it to the `./docker-build/be/resouce` directory.

5. Execute the build

Please note that `${tagName}` needs to be replaced with the tag name you want to package and name, such as: `apache-doris` ↪ `:1.1.3-be`

Build BE:

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

After the build process is completed, you will see the prompt `Success`. Then, you can check the built image using the following command.

```
docker images
```

Build Broker

1. Create a build environment directory

```
mkdir -p ./docker-build/broker/resource
```

2. The build environment directory is as follows:

```
└── docker-build                                          // build root directory
    └── broker                                            // BROKER build directory
        ├── dockerfile                                    // dockerfile script
        └── resource                                      // resource directory
            ├── init_broker.sh                            // startup and registration script
            └── apache-doris-x.x.x-bin-broker.tar.gz // binary package
```

3. Write the Dockerfile script for Broker

```
# Select the base image
FROM openjdk:8u342-jdk

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

# Download the software into the image, where the broker directory is synchronously compressed
    ↪  to the binary package of FE, which needs to be decompressed and repackaged (you can
    ↪ modify based on your own needs)
ADD ./resource/apache_hdfs_broker.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_hdfs_broker /opt/apache-doris/broker

ADD ./resource/init_broker.sh /opt/apache-doris/broker/bin
RUN chmod 755 /opt/apache-doris/broker/bin/init_broker.sh

ENTRYPOINT ["/opt/apache-doris/broker/bin/init_broker.sh"]
```

After writing, name it `Dockerfile` and save it to the `./docker-build/broker` directory

4. Write the execution script of BE

You can refer to init_broker.sh.

After writing, name it `init_broker.sh` and save it to the `./docker-build/broker/resouce` directory.

5. Execute the build

Please note that `${tagName}` needs to be replaced with the tag name you want to package and name, such as: `apache-doris` ↪ `:1.1.3-broker`

Build Broker:

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

After the build process is completed, you will see the prompt `Success`. Then, you can check the built image using the following command.

```
docker images
```

2.2.2.1.3 Push Image to DockerHub or Private Warehouse

Log into your DockerHub account

```
docker login
```

If the login succeeds, you will see the prompt `Success` , and then you can push the Docker image to the warehouse.

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

2.2.2.2 Deploy Docker cluster

2.2.2.2.1 Background description

This article will briefly describe how to quickly build a complete Doris test cluster through `docker run` or `docker-compose up` commands.

2.2.2.2.2 Applicable scene

It is recommended to use Doris Docker in SIT or DEV environment to simplify the deployment process.

If you want to test a certain function point in the new version, you can use Doris Docker to deploy a Playground environment. Or when you want to reproduce a certain problem during debugging, you can also use the docker environment to simulate.

In the production environment, currently try to avoid using containerized solutions for Doris deployment.

2.2.2.2.3 Software Environment

Software	Version
Docker	20.0 and above
docker-compose	2.10 and above

2.2.2.2.4 Hardware environment

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

2.2.2.2.5 Pre-environment preparation

The following command needs to be executed on the host machine

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

2.2.2.2.6 Docker Compose

Different platforms need to use different Image images. This article takes the X86_64 platform as an example.

Network Mode Description

There are two network modes applicable to Doris Docker.

1. HOST mode suitable for deployment across multiple nodes, this mode is suitable for deploying 1FE 1BE on each node.
2. The subnet bridge mode is suitable for deploying multiple Doris processes on a single node. This mode is suitable for single-node deployment (recommended). If you want to deploy multiple nodes, you need to deploy more components (not recommended).

For the sake of presentation, this chapter only demonstrates scripts written in subnet bridge mode.

Interface Description

From the version of `Apache Doris 1.2.1 Docker Image`, the interface list of each process image is as follows:

process name	interface name	interface definition	interface example
FE\BE\ ↪ BROKER	FE_SERVERS	FE node main information	fe1:172.20.80.2:9010,fe2:172.20.80.3:9010,fe3:172.20.80.4:9010
FE	FE_ID	FE node ID	1
BE	BE_ADDR	BE node main information	172.20.80.5:9050
BE	NODE_ROLE	BE node type	computation
BROKER	BROKER_ADDR	Main information of BROKER node	172.20.80.6:8000

Note that the above interface must fill in the information, otherwise the process cannot be started.

> FE_SERVERS interface rules are: `FE_NAME:FE_HOST:FE_EDIT_LOG_PORT[,FE_NAME:FE_HOST:FE_EDIT_↪ LOG_PORT]`
>
> The FE_ID interface rule is: an integer of `1-9`, where the FE number `1` is the Master node.
>
> BE_ADDR interface rule is: `BE_HOST:BE_HEARTBEAT_SERVICE_PORT`
>
> The NODE_ROLE interface rule is: `computation` or empty, where empty or other values indicate that the node type is `mix` type
>
> BROKER_ADDR interface rule is: `BROKER_HOST:BROKER_IPC_PORT`

Script Template

Docker Run command

Create a subnet bridge

```
docker network create --driver bridge --subnet=172.20.80.0/24 doris-network
```

1FE & 1BE Command Templates

```
docker run -itd \
--name=fe \
--env FE_SERVERS="fe1:172.20.80.2:9010" \
--env FE_ID=1 \
-p 8030:8030 \
-p 9030:9030 \
-v /data/fe/doris-meta:/opt/apache-doris/fe/doris-meta \
-v /data/fe/conf:/opt/apache-doris/fe/conf \
-v /data/fe/log:/opt/apache-doris/fe/log \
--network=doris-network \
--ip=172.20.80.2 \
apache/doris:1.2.1-fe-x86_64
```

```
docker run -itd \
--name=be \
--env FE_SERVERS="fe1:172.20.80.2:9010" \
--env BE_ADDR="172.20.80.3:9050" \
-p 8040:8040 \
-v /data/be/storage:/opt/apache-doris/be/storage \
-v /data/be/conf:/opt/apache-doris/be/conf \
-v /data/be/log:/opt/apache-doris/be/log \
--network=doris-network \
--ip=172.20.80.3 \
apache/doris:1.2.1-be-x86_64  # if CPU does not support AVX2, use
                              # apache/doris:1.2.1-be-x86_64-noavx2
```

Note: if you CPU does not support AVX2, the backend will fail to start. If this is the case, use the `apache/doris:X.X.X-be-x86_64-noavx2` backend image. Use `docker logs -f be` to check the backend for error messages.

3FE & 3BE run command template can be downloaded here.

Docker Compose script

1FE & 1BE template

```
version: '3'
services:
   docker-fe:
     image: "apache/doris:1.2.1-fe-x86_64"
     container_name: "doris-fe"
     hostname: "fe"
     environment:
       - FE_SERVERS=fe1:172.20.80.2:9010
       - FE_ID=1
     ports:
       - 8030:8030
       - 9030:9030
     volumes:
       - /data/fe/doris-meta:/opt/apache-doris/fe/doris-meta
       - /data/fe/conf:/opt/apache-doris/fe/conf
       - /data/fe/log:/opt/apache-doris/fe/log
     networks:
       doris_net:
         ipv4_address: 172.20.80.2
   docker-be:
     image: "apache/doris:1.2.1-be-x86_64"  # use apache/doris:1.2.1-be-x86_64-noavx2, if CPU
          ↪ does not support AVX2
     container_name: "doris-be"
     hostname: "be"
     depends_on:
       - docker-fe
     environment:
       - FE_SERVERS=fe1:172.20.80.2:9010
       - BE_ADDR=172.20.80.3:9050
     ports:
       - 8040:8040
     volumes:
       - /data/be/storage:/opt/apache-doris/be/storage
       - /data/be/conf:/opt/apache-doris/be/conf
       - /data/be/script:/docker-entrypoint-initdb.d
       - /data/be/log:/opt/apache-doris/be/log
     networks:
       doris_net:
         ipv4_address: 172.20.80.3
networks:
   doris_net:
     ipam:
       config:
         - subnet: 172.20.80.0/16
```

3FE & 3BE Docker Compose file can be downloaded here.

2.2.2.2.7 Deploy Doris Docker

You can choose one of the two deployment methods:

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

Special case description

Due to the different ways of implementing containers internally on MacOS, it may not be possible to directly modify the value of `max_map_count` on the host during deployment. You need to create the following containers first:

```
docker run -it --privileged --pid=host --name=change_count debian nsenter -t 1 -m -u -n -i sh
```

The container was created successfully executing the following command:

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

Then `exit` exits and creates the Doris Docker cluster.

2.2.2.2.8 Unfinished business

1. Compose Demo List

2.2.3 Kubernetes Deployment

2.2.3.1 Environmental Preparation

- Installation k8s
- Build or download a Doris image
  - Building an image Build Docker Image
  - Download Image https://hub.docker.com/r/apache/doris/tags
- Create or download the yml file for Doris on k8s
  - https://github.com/apache/doris/blob/master/docker/runtime/k8s/doris_follower.yml
  - https://github.com/apache/doris/blob/master/docker/runtime/k8s/doris_be.yml
  - https://github.com/apache/doris/blob/master/docker/runtime/k8s/doris_cn.yml

2.2.3.2 Starting a cluster

Start FE (role type is Follower):`kubectl create -f doris_follower.yml`

Start BE:`kubectl create -f doris_be.yml`

Start the BE (role type is Compute Node):`kubectl create -f doris_cn.yml`

2.2.3.3 Expansion and contraction capacity

- FE
- Currently, scaling is not supported. It is recommended to initialize 1 or 3 nodes as needed
- BE
- Command:`kubectl scale statefulset doris-be-cluster1 --replicas=4`
- BE (role type is Compute Node)
- Command:`kubectl scale statefulset doris-cn-cluster1 --replicas=4`

2.2.3.4 test and verify

Connect to the FE using mysql-client and perform operations such as' show backends' and 'show frontends' to view the status of each node

2.2.3.5 K8s simple operation command

- Executing the yml file for the first time `kubectl create -f xxx.yml`
- Execute after modifying the yml file `kubectl apply -f xxx.yml`
- Delete all resources defined by yml `kubectl delete -f xxx.yml`

- View the pod list `kubectl get pods`
- Entering the container `kubectl exec -it xxx`（podName） `-- /bin/sh`
- view log `kubectl logs xxx`（podName）
- View IP and port information `kubectl get ep`
- More knowledge of k8s

2.2.3.6 common problem

- How is data persistent?

Users need to mount PVC on their own to persist metadata information, data information, or log information - How to safely shrink the BE node?

BE:User manual execution is required before current resizingALTER-SYSTEM-DECOMMISSION-BACKEND

BE(The role type is Compute Node): Do not store data files and can directly shrink，About Computing Nodes - FE startup error “failed to init statefulSetName”

doris_ The environment variables statefulSetName and serviceName for follower. yml must appear in pairs, such as CN configured_ SERVICE, CN must be configured_ STATEFULSET

2.3 Table Design

2.3.1 Data Model

This topic introduces the data models in Doris from a logical perspective so you can make better use of Doris in different business scenarios.

2.3.1.1 Basic concepts

In Doris, data is logically described in the form of tables. A table consists of rows and columns. Row is a row of user data. Column is used to describe different fields in a row of data.

Columns can be divided into two categories: Key and Value. From a business perspective, Key and Value correspond to dimension columns and indicator columns, respectively.

Data models in Doris fall into three types:

- Aggregate
- Unique
- Duplicate

The following is the detailed introduction to each of them.

2.3.1.2 Aggregate Model

We illustrate what aggregation model is and how to use it correctly with practical examples.

2.3.1.2.1 Example 1: Importing data aggregation

Assume that the business has the following data table schema:

ColumnName	Type	AggregationType	Comment
userid	LARGEINT		user id
date	DATE		date of data filling
City	VARCHAR (20)		User City
age	SMALLINT		User age
sex	TINYINT		User gender
Last_visit_date	DATETIME	REPLACE	Last user access time
Cost	BIGINT	SUM	Total User Consumption
max dwell time	INT	MAX	Maximum user residence time
min dwell time	INT	MIN	User minimum residence time

The corresponding CREATE TABLE statement would be as follows (omitting the Partition and Distribution information):

```
CREATE TABLE IF NOT EXISTS example_db.example_tbl
(
    `user_id` LARGEINT NOT NULL COMMENT "user id",
    `date` DATE NOT NULL COMMENT "data import time",
    `city` VARCHAR(20) COMMENT "city",
    `age` SMALLINT COMMENT "age",
    `sex` TINYINT COMMENT "gender",
    `last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT "last visit date
        ↪ time",
    `cost` BIGINT SUM DEFAULT "0" COMMENT "user total cost",
    `max_dwell_time` INT MAX DEFAULT "0" COMMENT "user max dwell time",
    `min_dwell_time` INT MIN DEFAULT "99999" COMMENT "user min dwell time"
)
AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `sex`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
```

As you can see, this is a typical fact table of user information and visit behaviors. In star models, user information and visit behaviors are usually stored in dimension tables and fact tables, respectively. Here, for the convenience of explanation, we store the two types of information in one single table.

The columns in the table are divided into Key (dimension) columns and Value (indicator columns) based on whether they are set with an `AggregationType`. Key columns are not set with an `AggregationType`, such as `user_id`, `date`, and age, while Value columns are.

When data are imported, rows with the same contents in the Key columns will be aggregated into one row, and their values in the Value columns will be aggregated as their `AggregationType` specify. Currently, their are four aggregation types:

1. SUM: Accumulate the values in multiple rows.
2. REPLACE: The newly imported value will replace the previous value.
3. MAX: Keep the maximum value.
4. MIN: Keep the minimum value.

Suppose that you have the following import data (raw data):

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 06:00	20	10	10
10000	2017-10-01	Beijing	20	0	2017-10-01 07:00	15	2	2
10001	2017-10-01	Beijing	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	Shanghai	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	Guangzhou	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	Shenzhen	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	Shenzhen	35	0	2017-10-03 10:20:22	11	6	6

Assume that this is a table recording the user behaviors when visiting a certain commodity page. The first row of data, for example, is explained as follows:

Data	Description
10000	User id, each user uniquely identifies id
2017-10-01	Data storage time, accurate to date
Beijing	User City
20	User Age
0	Gender male (1 for female)
2017-10-01 06:00	User’s time to visit this page, accurate to seconds
20	Consumption generated by the user’s current visit
10	User’s visit, time to stay on the page
10	User’s current visit, time spent on the page (redundancy)

After this batch of data is imported into Doris correctly, it will be stored in Doris as follows:

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	35	10	2
10001	2017-10-01	Beijing	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	Shanghai	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	Guangzhou	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	Shenzhen	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	Shenzhen	35	0	2017-10-03 10:20:22	11	6	6

As you can see, the data of User 10000 have been aggregated to one row, while those of other users remain the same. The explanation for the aggregated data of User 10000 is as follows (the first 5 columns remain unchanged, so it starts with Column 6 `last_visit_date`):

*`2017-10-01 07:00`: The `last_visit_date` column is aggregated by REPLACE, so `2017-10-01 07:00` has replaced `2017-10-01 06:00`.

Note: When using REPLACE to aggregate data from the same import batch, the order of replacement is uncertain. That means, in this case, the data eventually saved in Doris could be `2017-10-01 06:00`. However, for different import batches, it is certain that data from the new batch will replace those from the old batch.

*35: The `cost`column is aggregated by SUM, so the update value 35 is the result of 20 + 15.

*10: The `max_dwell_time` column is aggregated by MAX, so 10 is saved as it is the maximum between 10 and 2.

*2: The `min_dwell_time` column is aggregated by MIN, so 2 is saved as it is the minimum between 10 and 2.

After aggregation, Doris only stores the aggregated data. In other words, the detailed raw data will no longer be available.

2.3.1.2.2 Example 2: keep detailed data

Here is a modified version of the table schema in Example 1:

ColumnName	Type	AggregationType	Comment
user_id	LARGEINT		User ID
date	DATE		Date when the data are imported
time stamp	DATETIME		Date and time when the data are imported (with second-level accuracy)
city	VARCHAR (20)		User location city
age	SMALLINT		User age
sex	TINYINT		User gender
last visit date	DATETIME	REPLACE	Last visit time of the user
cost	BIGINT	SUM	Total consumption of the user
max_dwell_time	INT	MAX	Maximum user dwell time
min_dwell_time	INT	MIN	Minimum user dwell time

A new column `timestamp` has been added to record the date and time when the data are imported (with second-level accuracy).

Suppose that the import data are as follows:

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	2017-10-01 08:00:05	Beijing	20	0	2017-10-01 06:00	20	10	10
10000	2017-10-01	2017-10-01 09:00:05	Beijing	20	0	2017-10-01 07:00	15	2	2
10001	2017-10-01	2017-10-01 18:12:10	Beijing	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	2017-10-02 13:10:00	Shanghai	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	2017-10-02 13:15:00	Guangzhou	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	2017-10-01 12:12:48	Shenzhen	35	0	2017-10-01 10:00:15	100	3	3

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10004	2017-10-03	2017-10-03 12:38:20	Shenzhen	35	0	2017-10-03 10:20:22	11	6	6

After importing, this batch of data will be stored in Doris as follows:

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	2017-10-01 08:00:05	Beijing	20	0	2017-10-01 06:00	20	10	10
10000	2017-10-01	2017-10-01 09:00:05	Beijing	20	0	2017-10-01 07:00	15	2	2
10001	2017-10-01	2017-10-01 18:12:10	Beijing	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	2017-10-02 13:10:00	Shanghai	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	2017-10-02 13:15:00	Guangzhou	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	2017-10-01 12:12:48	Shenzhen	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	2017-10-03 12:38:20	Shenzhen	35	0	2017-10-03 10:20:22	11	6	6

As you can see, the stored data are exactly the same as the import data. No aggregation has ever happened. This is because, the newly added `timestamp` column results in difference of Keys among the rows. That is to say, as long as the Keys of the rows are not identical in the import data, Doris can save the complete detailed data even in the Aggregate Model.

2.3.1.2.3 Example 3: aggregate import data and existing data

Based on Example 1, suppose that you have the following data stored in Doris:

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	35	10	2
10001	2017-10-01	Beijing	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	Shanghai	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	Guangzhou	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	Shenzhen	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	Shenzhen	35	0	2017-10-03 10:20:22	11	6	6

Now you need to import a new batch of data:

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10004	2017-10-03	Shenzhen	35	0	2017-10-03 11:22:00	44	19	19
10005	2017-10-03	Changsha	29	1	2017-10-03 18:11:02	3	1	1

After importing, the data stored in Doris will be updated as follows:

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	35	10	2
10001	2017-10-01	Beijing	30	1	2017-10-01 17:05:45	2	22	22

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10002	2017-10-02	Shanghai	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	Guangzhou	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	Shenzhen	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	Shenzhen	35	0	2017-10-03 11:22:00	55	19	6
10005	2017-10-03	Changsha	29	1	2017-10-03 18:11:02	3	1	1

As you can see, the existing data and the newly imported data of User 10004 have been aggregated. Meanwhile, the new data of User 10005 have been added.

In Doris, data aggregation happens in the following 3 stages:

1. The ETL stage of each batch of import data. At this stage, the batch of import data will be aggregated internally.
2. The data compaction stage of the underlying BE. At this stage, BE will aggregate data from different batches that have been imported.
3. The data query stage. The data involved in the query will be aggregated accordingly.

At different stages, data will be aggregated to varying degrees. For example, when a batch of data is just imported, it may not be aggregated with the existing data. But for users, they can only query aggregated data. That is, what users see are the aggregated data, and they should not assume that what they have seen are not or partly aggregated. (See the Limitations of Aggregate Model section for more details.)

2.3.1.3 Unique Model

In some multi-dimensional analysis scenarios, users are highly concerned about how to ensure the uniqueness of the Key, that is, how to create uniqueness constraints for the Primary Key. Therefore, we introduce the Unique Model. Prior to Doris 1.2, the Unique Model was essentially a special case of the Aggregate Model and a simplified representation of table schema. The Aggregate Model is implemented by Merge on Read, so it might not deliver high performance in some aggregation queries (see the Limitations of Aggregate Model (#Limitations of Aggregate Model) section). In Doris 1.2, we have introduced a new implementation for the Unique Model–Merge on Write, which can help achieve optimal query performance. For now, Merge on Read and Merge on Write will coexist in the Unique Model for a while, but in the future, we plan to make Merge on Write the default implementation of the Unique Model. The following will illustrate the two implementations with examples.

2.3.1.3.1 Merge on Read (Same Implementation as Aggregate Model)

ColumnName	Type	IsKey	Comment
user_id	BIGINT	Yes	User ID
username	VARCHAR (50)	Yes	Username
city	VARCHAR (20)	No	User location city
age	SMALLINT	No	User age
sex	TINYINT	No	User gender
phone	LARGEINT	No	User phone number
address	VARCHAR (500)	No	User address
register_time	DATETIME	No	User registration time

This is a typical user basic information table. There is no aggregation requirement for such data. The only concern is to ensure the uniqueness of the primary key. (The primary key here is user_id + username). The CREATE TABLE statement for the above table is as follows:

```
CREATE TABLE IF NOT EXISTS example_db.example_tbl
(
`user_id` LARGEINT NOT NULL COMMENT "User ID",
`username` VARCHAR (50) NOT NULL COMMENT "Username",
`city` VARCHAR (20) COMMENT "User location city",
`age` SMALLINT COMMENT "User age",
`sex` TINYINT COMMENT "User sex",
`phone` LARGEINT COMMENT "User phone number",
`address` VARCHAR (500) COMMENT "User address",
`register_time` DATETIME COMMENT "User registration time"
)
UNIQUE KEY (`user_id`, `username`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
```

This is the same table schema and the CREATE TABLE statement as those of the Aggregate Model:

ColumnName	Type	AggregationType	Comment
user_id	BIGINT		User ID
username	VARCHAR (50)		Username
city	VARCHAR (20)	REPLACE	User location city
age	SMALLINT	REPLACE	User age
sex	TINYINT	REPLACE	User gender
phone	LARGEINT	REPLACE	User phone number
address	VARCHAR (500)	REPLACE	User address
register_time	DATETIME	REPLACE	User registration time

```
CREATE TABLE IF NOT EXISTS example_db.example_tbl
(
`user_id` LARGEINT NOT NULL COMMENT "User ID",
`username` VARCHAR (50) NOT NULL COMMENT "Username",
`city` VARCHAR (20) REPLACE COMMENT "User location city",
`sex` TINYINT REPLACE COMMENT "User gender",
`phone` LARGEINT REPLACE COMMENT "User phone number",
`address` VARCHAR(500) REPLACE COMMENT "User address",
`register_time` DATETIME REPLACE COMMENT "User registration time"
)
AGGREGATE KEY(`user_id`, `username`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
```

```
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
```

That is to say, the Merge on Read implementation of the Unique Model is equivalent to the REPLACE aggregation type in the Aggregate Model. The internal implementation and data storage are exactly the same.

2.3.1.3.2 Merge on Write (Since Doris 1.2)

The Merge on Write implementation of the Unique Model is completely different from that of the Aggregate Model. It can deliver better performance in aggregation queries with primary key limitations.

In Doris 1.2.0, as a new feature, Merge on Write is disabled by default, and users can enable it by adding the following property:

```
"enable_unique_key_merge_on_write" = "true"
```

Take the previous table as an example, the corresponding CREATE TABLE statement should be:

```
CREATE TABLE IF NOT EXISTS example_db.example_tbl
(
`user_id` LARGEINT NOT NULL COMMENT "User ID",
`username` VARCHAR (50) NOT NULL COMMENT "Username",
`city` VARCHAR (20) COMMENT "User location city",
`age` SMALLINT COMMENT "Userage",
`sex` TINYINT COMMENT "User gender",
`phone` LARGEINT COMMENT "User phone number",
`address` VARCHAR (500) COMMENT "User address",
`register_time` DATETIME COMMENT "User registration time"
)
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"
);
```

The table schema produced by the above statement will be different from that of the Aggregate Model.

ColumnName	Type	AggregationType	Comment
user_id	BIGINT		User ID
username	VARCHAR (50)		Username
city	VARCHAR (20)	NONE	User location city
age	SMALLINT	NONE	User age
sex	TINYINT	NONE	User gender
phone	LARGEINT	NONE	User phone number
address	VARCHAR (500)	NONE	User address
register_time	DATETIME	NONE	User registration time

On a Unique table with the Merge on Write option enabled, during the import stage, the data that are to be overwritten and updated will be marked for deletion, and new data will be written in. When querying, all data marked for deletion will be filtered out at the file level, and only the latest data would be readed. This eliminates the data aggregation cost while reading, and supports many types of predicate pushdown now. Therefore, it can largely improve performance in many scenarios, especially in aggregation queries.

NOTE

1. The new Merge on Write implementation is disabled by default, and can only be enabled by specifying a property when creating a new table.
2. The old Merge on Read cannot be seamlessly upgraded to the new implementation (since they have completely different data organization). If you want to switch to the Merge on Write implementation, you need to manually execute `insert ↪ into unique-mow- table select * from source table` to load data to new table.
3. The two unique features `delete sign` and `sequence col` of the Unique Model can be used as normal in the new implementation, and their usage remains unchanged.

2.3.1.4 Duplicate Model

In some multi-dimensional analysis scenarios, there is no need for primary keys or data aggregation. For these cases, we introduce the Duplicate Model to. Here is an example:

ColumnName	Type	SortKey	Comment
timstamp	DATETIME	Yes	Log time
type	INT	Yes	Log type
error_code	INT	Yes	Error code
Error_msg	VARCHAR (1024)	No	Error details
op_id	BIGINT	No	Operator ID
op_time	DATETIME	No	Operation time

The corresponding CREATE TABLE statement is as follows:

```
CREATE TABLE IF NOT EXISTS example_db.example_tbl
(
    `timestamp` DATETIME NOT NULL COMMENT "Log time",
    `type` INT NOT NULL COMMENT "Log type",
    `error_code` INT COMMENT "Error code",
    `error_msg` VARCHAR(1024) COMMENT "Error details",
    `op_id` BIGINT COMMENT "Operator ID",
    `op_time` DATETIME COMMENT "Operation time"
)
DUPLICATE KEY(`timestamp`, `type`, `error_code`)
DISTRIBUTED BY HASH(`type`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
```

Different from the Aggregate and Unique Models, the Duplicate Model stores the data as they are and executes no aggregation. Even if there are two identical rows of data, they will both be retained. The DUPLICATE KEY in the CREATE TABLE statement is only used to specify based on which columns the data are sorted. (A more appropriate name than DUPLICATE KEY would be SORTING COLUMN, but it is named as such to specify the data model used. For more information, see Prefix Index.) For the choice of DUPLICATE KEY, we recommend the first 2-4 columns.

The Duplicate Mode l is suitable for storing raw data without aggregation requirements or primary key uniqueness constraints. For more usage scenarios, see the Limitations of Aggregate Model section.

2.3.1.5 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

user_id	date	cost
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
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.

2.3.1.5.1 Merge on Write of Unique 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.

2.3.1.5.2 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.

2.3.1.6 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".

2.3.1.7 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 unique primary key. The downside is that it cannot exploit the advantage brought by pre-aggregation such as ROLLUP in queries.
3. Users who have high performance requirements for aggregate queries are recommended to use the newly added Merge on Write implementation since version 1.2.
4. The Unique Model only supports entire-row updates. If you require primary key uniqueness as well as partial updates of certain columns (such as loading multiple source tables into one Doris table), you can consider using the Aggregate Model, while setting the aggregate type of the non-primary key columns to REPLACE_IF_NOT_NULL. See CREATE TABLE Manual for more details.
5. 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).

2.3.2 Data Partition

This topic is about table creation and data partitioning in Doris, including the common problems in table creation and their solutions.

2.3.2.1 Basic Concepts

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

2.3.2.1.1 Row & Column

A table contains rows and columns.

Row refers to a row of data about the user. Column is used to describe different fields in a row of data.

Columns can be divided into two categories: Key and Value. From a business perspective, Key and Value correspond to dimension columns and metric columns, respectively. In the Aggregate Model, rows with the same values in Key columns will be aggregated into one row. The way how Value columns are aggregated is specified by the user when the table is built. For more information about the Aggregate Model, please see the Data Model.

2.3.2.1.2 Tablet & Partition

In the Doris storage engine, user data are horizontally divided into data tablets (also known as data buckets). Each tablet contains several rows of data. The data between the individual tablets do not intersect and is physically stored independently.

Tablets are logically attributed to different Partitions. One Tablet belongs to only one Partition, and one Partition contains several Tablets. Since the tablets are physically stored independently, the partitions can be seen as physically independent, too. Tablet is the smallest physical storage unit for data operations such as movement and replication.

A Table is formed of multiple Partitions. Partition can be thought of as the smallest logical unit of management. Data import and deletion can be performed on only one Partition.

2.3.2.2 Data Partitioning

The following illustrates on data partitioning in Doris using the example of a CREATE TABLE operation.

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 on the syntax, please refer to CREATE TABLE, or input the `HELP CREATE TABLE;` command.

This section introduces how to create tables in Doris.

```sql
-- Range Partition

CREATE TABLE IF NOT EXISTS example_db.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 LESS THAN ("2017-02-01"),
    PARTITION `p201702` VALUES LESS THAN ("2017-03-01"),
    PARTITION `p201703` VALUES LESS THAN ("2017-04-01"),
    PARTITION `p2018` VALUES [("2018-01-01"), ("2019-01-01"))
)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 16
PROPERTIES
(
    "replication_num" = "3",
    "storage_medium" = "SSD",
    "storage_cooldown_time" = "2018-01-01 12:00:00"
);


-- List Partition

CREATE TABLE IF NOT EXISTS example_db.example_list_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 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")
)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 16
PROPERTIES
(
    "replication_num" = "3",
    "storage_medium" = "SSD",
    "storage_cooldown_time" = "2018-01-01 12:00:00"
);
```

2.3.2.2.1 Definition of Column

See Doris Data Model for more information.

You can view the basic types of columns by executing `HELP CREATE TABLE;` in MySQL Client.

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.

A few suggested rules for defining columns include:

1. The Key columns must precede all Value columns.
2. Try to choose the INT type as much as possible. Because calculations and lookups on INT types are much more efficient than those on strings.
3. For the lengths of the INT types, follow the good enough principle.
4. For the lengths of the VARCHAR and STRING types, also follow the good enough principle.

2.3.2.2.2 Partitioning and Bucketing

Doris supports two layers of data partitioning. The first level is Partition, including range partitioning and list partitioning. The second is Bucket (Tablet), which only supports hash partitioning.

It is also possible to use one layer of data partitioning. In this case, it only supports data bucketing.

1. Partition

- You can specify one or more columns as the partitioning columns, but they have to be KEY columns. The usage of multi-column partitions is described further below.
- Regardless of the type of the partitioning columns, double quotes are required for partition values.
- There is no theoretical limit on the number of partitions.
- If users create a table without specifying the partitions, the system will automatically generate a Partition with the same name as the table. This Partition contains all data in the table and is neither visible to users nor modifiable.
- Partitions should not have overlapping ranges.

#### Range Partitioning

- Partitioning columns are usually time columns for easy management of old and new data.
- Range partitioning supports specifying only the upper bound by `VALUES LESS THAN (...)`. The system will use the upper bound of the previous partition as the lower bound of the next partition, and generate a left-closed right-open interval. It also supports specifying both the upper and lower bounds by `VALUES [...)`, and generate a left-closed right-open interval.
- The following takes the `VALUES [...)` method as an example since it is more comprehensible. It shows how the partition ranges change as we use the `VALUES LESS THAN (...)` statement to add or delete partitions:
  - As in the `example_range_tbl` example above, when the table is created, the following 3 partitions are automatically generated:

```
    P201701: [MIN_VALUE, 2017-02-01)
    P201702: [2017-02-01, 2017-03-01)
    P201703: [2017-03-01, 2017-04-01)
```

```
* If we add Partition p201705 VALUES LESS THAN ("2017-06-01"), the results will be as follows:
```

```
    P201701: [MIN_VALUE, 2017-02-01)
    P201702: [2017-02-01, 2017-03-01)
    P201703: [2017-03-01, 2017-04-01)
    P201705: [2017-04-01, 2017-06-01)
```

```
* Then we delete Partition p201703, the results will be as follows:
```

```
    p201701: [MIN_VALUE, 2017-02-01)
    p201702: [2017-02-01, 2017-03-01)
    p201705: [2017-04-01, 2017-06-01)
```

```
  > Note that the partition range of p201702 and p201705 has not changed, and there is a gap
    ↪ between the two partitions: [2017-03-01, 2017-04-01). That means, if the imported data
    ↪  is within this gap range, the import would fail.

* Now we go on and delete Partition p201702, the results will be as follows:
```

```
    p201701: [MIN_VALUE, 2017-02-01)
    p201705: [2017-04-01, 2017-06-01)
```

```
  > The gap range expands to: [2017-02-01, 2017-04-01)

* Then we add Partition p201702new VALUES LESS THAN ("2017-03-01"), the results will be as
  ↪ follows:
```

```
    p201701: [MIN_VALUE, 2017-02-01)
    p201702new: [2017-02-01, 2017-03-01)
    p201705: [2017-04-01, 2017-06-01)
```

```
  > The gap range shrinks to: [2017-03-01, 2017-04-01)

* Now we delete Partition p201701 and add Partition p201612 VALUES LESS THAN ("2017-01-01"), the
    ↪  partition result is as follows:
```

```
      p201612: [MIN_VALUE, 2017-01-01)
      p201702new: [2017-02-01, 2017-03-01)
      p201705: [2017-04-01, 2017-06-01)
```

```
  > This results in a new gap range: [2017-01-01, 2017-02-01)
```

In summary, the deletion of a partition does not change the range of the existing partitions, but might result in gaps. When a partition is added via the VALUES LESS THAN statement, the lower bound of one partition is the upper bound of its previous partition.

In addition to the single-column partitioning mentioned above, Range Partitioning also supports multi-column partitioning. Examples are as follows:

```
    PARTITION BY RANGE(`date`, `id`)
    (
        PARTITION `p201701_1000` VALUES LESS THAN ("2017-02-01", "1000"),
        PARTITION `p201702_2000` VALUES LESS THAN ("2017-03-01", "2000"),
        PARTITION `p201703_all` VALUES LESS THAN ("2017-04-01")
    )
```

```
In the above example, we specify `date` (DATE type) and `id` (INT type) as the partitioning
    ↪ columns, so the resulting partitions will be as follows:
```

```
  *p201701_1000: [(MIN_VALUE, MIN_VALUE), ("2017-02-01", "1000") )
  *p201702_2000: [("2017-02-01", "1000"), ("2017-03-01", "2000") )
  *p201703_all: [("2017-03-01", "2000"), ("2017-04-01", MIN_VALUE))
```

Note that in the last partition, the user only specifies the partition value of the date column, so the system fills in MIN_VALUE as the partition value of the id column by default. When data are imported, the system will compare them with the partition values in order, and put the data in their corresponding partitions. Examples are as follows:

```
  * Data --> Partition
  * 2017-01-01, 200   --> p201701_1000
  * 2017-01-01, 2000  --> p201701_1000
  * 2017-02-01, 100   --> p201701_1000
  * 2017-02-01, 2000  --> p201702_2000
  * 2017-02-15, 5000  --> p201702_2000
  * 2017-03-01, 2000  --> p201703_all
  * 2017-03-10, 1     --> p201703_all
  * 2017-04-01, 1000  --> Unable to import
  * 2017-05-01, 1000  --> Unable to import
```

Range partitioning also supports batch partitioning. For example, you can create multiple partitions that are divided by day at a time using the `FROM ("2022-01-03")TO ("2022-01-06")INTERVAL 1 DAY`: 2022-01-03 to 2022-01-06 (not including 2022-01-06), the results will be as follows:

```
p20220103:    [2022-01-03,  2022-01-04)
p20220104:    [2022-01-04,  2022-01-05)
p20220105:    [2022-01-05,  2022-01-06)
```

List Partitioning

- The partitioning columns support the `BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, LARGEINT, DATE,` ↪ `DATETIME, CHAR, VARCHAR` data types, and the partition values are enumeration values. Partitions can be only hit if the data is one of the enumeration values in the target partition.
- List partitioning supports using `VALUES IN (...)` to specify the enumeration values contained in each partition.
- The following example illustrates how partitions change when adding or deleting a partition.
- As in the `example_list_tbl` example above, when the table is created, the following three partitions are automatically created.

```
    p_cn: ("Beijing", "Shanghai", "Hong Kong")
    p_usa: ("New York", "San Francisco")
    p_jp: ("Tokyo")
```

- If we add Partition p_uk VALUES IN (“London”), the results will be as follows:

```
    p_cn: ("Beijing", "Shanghai", "Hong Kong")
    p_usa: ("New York", "San Francisco")
    p_jp: ("Tokyo")
    p_uk: ("London")
```

- Now we delete Partition p_jp, the results will be as follows:

```
    p_cn: ("Beijing", "Shanghai", "Hong Kong")
    p_usa: ("New York", "San Francisco")
    p_jp: ("Tokyo")
    p_uk: ("London")
```

List partitioning also supports multi-column partitioning. Examples are as follows:

```
  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"))
  )
```

In the above example, we specify `id` (INT type) and `city` (VARCHAR type) as the partitioning columns, so the resulting partitions will be as follows:

```
* p1_city: [("1", "Beijing"), ("1", "Shanghai")]
* p2_city: [("2", "Beijing"), ("2", "Shanghai")]
* p3_city: [("3", "Beijing"), ("3", "Shanghai")]
```

When data are imported, the system will compare them with the partition values in order, and put the data in their corresponding partitions. Examples are as follows:

```
Data ---> Partition
1, Beijing  ---> p1_city
1, Shanghai ---> p1_city
2, Shanghai ---> p2_city
3, Beijing  ---> p3_city
1, Tianjin  ---> Unable to import
4, Beijing  ---> Unable to import
```

2. Bucketing

- If you use the Partition method, the `DISTRIBUTED ...` statement will describe how data are divided among partitions. If you do not use the Partition method, that statement will describe how data of the whole table are divided.
- You can specify multiple columns as the bucketing columns. In Aggregate and Unique Models, bucketing columns must be Key columns; in the Duplicate Model, bucketing columns can be Key columns and Value columns. Bucketing columns can either be partitioning columns or not.
- The choice of bucketing columns is a trade-off between query throughput and query concurrency:
    1. If you choose to specify multiple bucketing columns, the data will be more evenly distributed. However, if the query condition does not include the equivalent conditions for all bucketing columns, the system will scan all buckets, largely increasing the query throughput and decreasing the latency of a single query. This method is suitable for high-throughput, low-concurrency query scenarios.
    2. If you choose to specify only one or a few bucketing columns, point queries might scan only one bucket. Thus, when multiple point queries are preformed concurrently, they might scan various buckets, with no interaction between the IO operations (especially when the buckets are stored on various disks). This approach is suitable for high-concurrency point query scenarios.
- AutoBucket: Calculates the number of partition buckets based on the amount of data. For partitioned tables, you can determine a bucket based on the amount of data, the number of machines, and the number of disks in the historical partition.
- There is no theoretical limit on the number of buckets.

3. Recommendations on the number and data volume for Partitions and Buckets.

- The total number of tablets in a table is equal to (Partition num * Bucket num).
- The recommended number of tablets in a table, regardless of capacity expansion, is slightly more than the number of disks in the entire cluster.

- The data volume of a single tablet does not have an upper or lower limit theoretically, but is recommended to be in the range of 1G - 10G. Overly small data volume of a single tablet can impose a stress on data aggregation and metadata management; while overly large data volume can cause trouble in data migration and completion, and increase the cost of Schema Change or Rollup operation failures (These operations are performed on the Tablet level).
- For the tablets, if you cannot have the ideal data volume and the ideal quantity at the same time, it is recommended to prioritize the ideal data volume.
- Upon table creation, you specify the same number of Buckets for each Partition. However, when dynamically increasing partitions (`ADD PARTITION`), you can specify the number of Buckets for the new partitions separately. This feature can help you cope with data reduction or expansion.
- Once you have specified the number of Buckets for a Partition, you may not change it afterwards. Therefore, when determining the number of Buckets, you need to consider the need of cluster expansion in advance. For example, if there are only 3 hosts, and each host has only 1 disk, and you have set the number of Buckets is only set to 3 or less, then no amount of newly added machines can increase concurrency.
- For example, suppose that there are 10 BEs and each BE has one disk, if the total size of a table is 500MB, you can consider dividing it into 4-8 tablets; 5GB: 8-16 tablets; 50GB: 32 tablets; 500GB: you may consider dividing it into partitions, with each partition about 50GB in size, and 16-32 tablets per partition; 5TB: divided into partitions of around 50GB and 16-32 tablets per partition.

> Note: You can check the data volume of the table using the show data command. Divide the returned result by the number of copies, and you will know the data volume of the table.

4. About the settings and usage scenarios of Random Distribution:

- If the OLAP table does not have columns of REPLACE type, set the data bucketing mode of the table to RANDOM. This can avoid severe data skew. (When loading data into the partition corresponding to the table, each batch of data in a single load task will be written into a randomly selected tablet).
- When the bucketing mode of the table is set to RANDOM, since there are no specified bucketing columns, it is impossible to query only a few buckets, so all buckets in the hit partition will be scanned when querying the table. Thus, this setting is only suitable for aggregate query analysis of the table data as a whole, but not for highly concurrent point queries.
- If the data distribution of the OLAP table is Random Distribution, you can set `load to single tablet` to true when importing data. In this way, when importing large amounts of data, in one task, data will be only written in one tablet of the corresponding partition. This can improve both the concurrency and throughput of data import and reduce write amplification caused by data import and compaction, and thus, ensure cluster stability.

Compound Partitioning vs Single Partitioning

Compound Partitioning

- The first layer of data partitioning is called Partition. Users can specify a dimension column as the partitioning column (currently only supports columns of INT and TIME types), and specify the value range of each partition.
- The second layer is called Distribution, which means bucketing. Users can perform HASH distribution on data by specifying the number of buckets and one or more dimension columns as the bucketing columns, or perform random distribution on data by setting the mode to Random Distribution.

Compound partitioning is recommended for the following scenarios:

- Scenarios with time dimensions or similar dimensions with ordered values, which can be used as partitioning columns. The partitioning granularity can be evaluated based on data import frequency, data volume, etc.
- Scenarios with a need to delete historical data: If, for example, you only need to keep the data of the last N days), you can use compound partitioning so you can delete historical partitions. To remove historical data, you can also send a DELETE statement within the specified partition.
- Scenarios with a need to avoid data skew: you can specify the number of buckets individually for each partition. For example, if you choose to partition the data by day, and the data volume per day varies greatly, you can customize the number of buckets for each partition. For the choice of bucketing column, it is recommended to select the column(s) with variety in values.

Users can also choose for single partitioning, which is about HASH distribution.

2.3.2.2.3 PROPERTIES

In the `PROPERTIES` section at the last of the CREATE TABLE statement, you can set the relevant parameters. Please see CREATE TABLE for a detailed introduction.

2.3.2.2.4 ENGINE

In this example, the ENGINE is of OLAP type, which is the default ENGINE type. In Doris, only the OALP ENGINE type is managed and stored by Doris. Other ENGINE types, such as MySQL, Broker, ES, are essentially mappings to tables in other external databases or systems to ensure that Doris can read the data. And Doris itself does not create, manage, or store any tables and data of non-OLAP ENGINE type.

2.3.2.2.5 Other

`IF NOT EXISTS` means to create the table if it is non-existent. Note that the system only checks the existence of table based on the table name, but not compare the schema of the newly created table with the existing ones. So if there exists a table of the same name but different schema, the command will also return, but it does not mean that a new table of a new schema has been created.

2.3.2.3 FAQ

2.3.2.3.1 Table Creation

1. If a syntax error occurs in a long CREATE TABLE statement, the error message may be incomplete. Here is a list of possible syntax errors for your reference in manual touble shooting:

- Incorrect syntax. Please use `HELP CREATE TABLE;` to check the relevant syntax.
- Reserved words. Reserved words in user-defined names should be enclosed in backquotes “. It is recommended that all user-defined names be enclosed in backquotes.
- 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 that you check for these characters using a text editor that can display non-printable characters.

2. `Failed to create partition [xxx] . Timeout`

In Doris, tables are created in the order of the partitioning granularity. This error prompt may appear when a partition creation task fails, but it could also appear in table creation tasks with no partitioning operations, because, as mentioned earlier, Doris will create an unmodifiable default partition for tables with no partitions specified.

This error usually pops up because the tablet creation goes wrong in BE. You can follow the steps below for troubleshooting:

1. In fe.log, find the `Failed to create partition` log of the corresponding time point. In that log, find a number pair that looks like `{10001-10010}` . The first number of the pair is the Backend ID and the second number is the Tablet ID. As for `{10001-10010}`, it means that on Backend ID 10001, the creation of Tablet ID 10010 failed.
2. After finding the target Backend, go to the corresponding be.INFO log and find the log of the target tablet, and then check the error message.
3. A few common tablet creation failures include but not limited to:
   - The task is not received by BE. In this case, the tablet ID related information will be found in be.INFO, or the creation is successful in BE but it still reports a failure. To solve the above problems, see Installation and Deployment about how to check the connectivity of FE and BE.
   - Pre-allocated memory failure. It may be that the length of a row in the table exceeds 100KB.
   - `Too many open files`. The number of open file descriptors exceeds the Linux system limit. In this case, you need to change the open file descriptor limit of the Linux system.

If it is a timeout error, you can set `tablet_create_timeout_second=xxx` and `max_create_table_timeout_second=xxx` ↪ in fe.conf. The default value of `tablet_create_timeout_second=xxx` is 1 second, and that of `max_create_table_` ↪ `timeout_second=xxx` is 60 seconds. The overall timeout would be min(tablet_create_timeout_second * replication_num, max_create_table_timeout_second). For detailed parameter settings, please check FE Configuration.

3. The build table command does not return results for a long time.

Doris’s table creation command is a synchronous command. The timeout of this command is currently set to be relatively simple, ie (tablet num * replication num) seconds. If you create more data fragments and have fragment creation failed, it may cause an error to be returned after waiting for a long timeout.

Under normal circumstances, the statement will return in a few seconds or ten seconds. If it is more than one minute, it is recommended to cancel this operation directly and go to the FE or BE log to view the related errors.

2.3.2.4 More Help

For more detailed instructions on data partitioning, please refer to the CREATE TABLE command manual, or enter `HELP CREATE` ↪ `TABLE;` in MySQL Client.

2.3.3 Guidelines for Basic Use

Doris uses MySQL protocol for communication. Users can connect to Doris clusters through MySQL Client or MySQL JDBC. MySQL Client 5.1 or newer versions are recommended because they support user names of more than 16 characters. This topic walks you through how to use Doris with the example of MySQL Client.

2.3.3.1 Create Users

2.3.3.1.1 Root User Login and Change Password

Doris has its built-in root, and the default password is empty.

> Note:
>
> Doris provides a default root.
>
> The root user has all the privileges about the clusters by default. Users who have both Grant_priv and Node_priv can grant these privileges to other users. Node changing privileges include adding, deleting, and offlining FE, BE, and BROKER nodes.
>
> For more instructions on privileges, please refer to Privilege Management

After starting the Doris program, root or admin users can connect to Doris clusters. You can use the following command to log in to Doris. After login, you will enter the corresponding MySQL command line interface.

```
[root@doris ~]# mysql  -h FE_HOST -P9030 -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.1.0 Doris version 1.0.0-preview2-b48ee2734

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>
```

> 1. FE_HOST is the IP address of any FE node. 9030 is the query_port configuration in fe.conf.

After login, you can change the root password by the following command:

```
mysql> SET PASSWORD FOR 'root' = PASSWORD('your_password');
Query OK, 0 rows affected (0.00 sec)
```

your_password is a new password for the root user, which can be set at will. A strong password is recommended for security. The new password is required in the next login.

2.3.3.1.2 Create New Users

You can create a regular user named test with the following command:

```
mysql> CREATE USER 'test' IDENTIFIED BY 'test_passwd';
Query OK, 0 rows affected (0.00 sec)
```

Follow-up logins can be performed with the following connection commands.

```
[root@doris ~]# mysql -h FE_HOST -P9030 -utest -ptest_passwd
```

By default, the newly created regular users do not have any privileges. Privileges can be granted to these users.

2.3.3.2 Create Data Table and Import Data

2.3.3.2.1 Create a database

Initially, root or admin users can create a database by the following command:

```
CREATE DATABASE example_db;
```

You can use the HELP command to check the syntax of all commands. For example, HELP CREATE DATABASE;. Or you can refer to the SHOW CREATE DATABASE command manual.

If you don't know the full name of the command, you can use "HELP + a field of the command" for fuzzy query. For example, if you type in HELP CREATE, you can find commands including CREATE DATABASE, CREATE
↪ TABLE, and CREATE USER.

```
mysql> HELP CREATE;
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following
topics:
   CREATE CLUSTER
   CREATE DATABASE
   CREATE ENCRYPTKEY
   CREATE FILE
   CREATE FUNCTION
   CREATE INDEX
   CREATE MATERIALIZED VIEW
   CREATE REPOSITORY
   CREATE RESOURCE
   CREATE ROLE
   CREATE SYNC JOB
   CREATE TABLE
   CREATE USER
   CREATE VIEW
   ROUTINE LOAD
   SHOW CREATE FUNCTION
   SHOW CREATE ROUTINE LOAD
```

After the database is created, you can view the information about the database via the SHOW DATABASES command.

```
MySQL> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| example_db         |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)
```

`information_schema` exists for compatibility with MySQL protocol, so the information might not be 100% accurate in practice. Therefore, for information about the specific databases, please query the corresponding databases directly.

2.3.3.2.2 Authorize an Account

After `example_db` is created, root/admin users can grant read/write privileges of `example_db` to regular users, such as `test`, using the GRANT command. After authorization, user `test` can perform operations on `example_db`.

```
mysql> GRANT ALL ON example_db TO test;
Query OK, 0 rows affected (0.01 sec)
```

2.3.3.2.3 Create a Table

You can create a table using the CREATE TABLE command. For detailed parameters, you can send a `HELP CREATE TABLE;` command.

Firstly, you need to switch to the target database using the USE command:

```
mysql> USE example_db;
Database changed
```

Doris supports two table creation methods: compound partitioning and single partitioning. The following takes the Aggregate Model as an example to demonstrate how to create tables with these two methods, respectively.

Single Partitioning

Create a logical table named `table1`. The bucketing column is the `siteid` column, and the number of buckets is 10.

The table schema is as follows:

- `siteid`: INT (4 bytes); default value: 10
- `citycode`: SMALLINT (2 bytes)
- `username`: VARCHAR, with a maximum length of 32; default value: empty string
- pv: BIGINT (8 bytes); default value: 0; This is a metric column, and Doris will aggregate the metric columns internally. The pv column is aggregated by SUM.

The corresponding CREATE TABLE statement is as follows:

```
CREATE TABLE table1
(
    siteid INT DEFAULT '10',
    citycode SMALLINT,
    username VARCHAR(32) DEFAULT '',
    pv BIGINT SUM DEFAULT '0'
)
AGGREGATE KEY(siteid, citycode, username)
DISTRIBUTED BY HASH(siteid) BUCKETS 10
PROPERTIES("replication_num" = "1");
```

Compound Partitioning

Create a logical table named `table2`.

The table schema is as follows:

- `event_day`: DATE; no default value
- `siteid`: INT (4 bytes); default value: 10
- `citycode`: SMALLINT (2 bytes)
- `username`: VARCHAR, with a maximum length of 32; default value: empty string
- pv: BIGINT (8 bytes); default value: 0; This is a metric column, and Doris will aggregate the metric columns internally. The pv column is aggregated by SUM.

Use the `event_day` column as the partitioning column and create 3 partitions: p201706, p201707, and p201708.

- p201706: Range [Minimum, 2017-07-01)
- p201707: Range [2017-07-01, 2017-08-01)
- p201708: Range [2017-08-01, 2017-09-01)

Note that the intervals are left-closed and right-open.

HASH bucket each partition based on `siteid`. The number of buckets per partition is 10.

The corresponding CREATE TABLE statement is as follows:

```
CREATE TABLE table2
(
    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)
PARTITION BY RANGE(event_day)
(
    PARTITION p201706 VALUES LESS THAN ('2017-07-01'),
    PARTITION p201707 VALUES LESS THAN ('2017-08-01'),
    PARTITION p201708 VALUES LESS THAN ('2017-09-01')
)
DISTRIBUTED BY HASH(siteid) BUCKETS 10
PROPERTIES("replication_num" = "1");
```

After the table is created, you can view the information of the table in `example_db`:

```
MySQL> SHOW TABLES;
+----------------------+
| Tables_in_example_db |
+----------------------+
| table1               |
| table2               |
+----------------------+
2 rows in set (0.01 sec)

MySQL> DESC table1;
+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | Yes  | true  | 10      |       |
| citycode | smallint(6) | Yes  | true  | N/A     |       |
| username | varchar(32) | Yes  | true  |         |       |
| pv       | bigint(20)  | Yes  | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
4 rows in set (0.00 sec)

MySQL> DESC table2;
+-----------+-------------+------+-------+---------+-------+
| Field     | Type        | Null | Key   | Default | Extra |
+-----------+-------------+------+-------+---------+-------+
| event_day | date        | Yes  | true  | N/A     |       |
| siteid    | int(11)     | Yes  | true  | 10      |       |
| citycode  | smallint(6) | Yes  | true  | N/A     |       |
| username  | varchar(32) | Yes  | true  |         |       |
| pv        | bigint(20)  | Yes  | false | 0       | SUM   |
+-----------+-------------+------+-------+---------+-------+
5 rows in set (0.00 sec)
```

Note:

1. As `replication_num` is set to 1, the above tables are created with only one copy. We recommend that you adopt the default three-copy settings to ensure high availability.
2. You can dynamically add or delete partitions of compoundly partitioned tables. See `HELP ALTER TABLE`.
3. You can import data into the specified Partition. See `HELP LOAD;`.
4. You can dynamically change the table schema. See `HELP ALTER TABLE;`.
5. You can add Rollups to Tables to improve query performance. See the Rollup-related section in "Advanced Usage".
6. The default value in the Null column is true, which may affect query performance.

2.3.3.2.4 Load data

Doris supports a variety of data loading methods. You can refer to Data Loading for more details. The following uses Stream Load and Broker Load as examples.

Stream Load

The Stream Load method transfers data to Doris via HTTP protocol. It can import local data directly without relying on any other systems or components. For the detailed syntax, please see `HELP STREAM LOAD;`.

Example 1: Use "table1_20170707" as the Label, import the local file `table1_data` into `table1`.

```
curl --location-trusted -u test:test_passwd -H "label:table1_20170707" -H "column_separator:," -T
    ↪  table1_data http://FE_HOST:8030/api/example_db/table1/_stream_load
```

1. FE_HOST is the IP address of any FE node and 8030 is the http_port in fe.conf.
2. You can use the IP address of any BE and the webserver_port in be.conf for import. For example: BE_HOST
   ↪ :8040.

The local file `table1_data` uses `,` as the separator between data. The details are as follows:

```
1,1,Jim,2
2,1,grace,2
3,2,tom,2
4,3,bush,3
5,3,helen,3
```

Example 2: Use "table2_20170707" as the Label, import the local file `table2_data` into `table2`.

```
curl --location-trusted -u test:test -H "label:table2_20170707" -H "column_separator:|" -T table2
    ↪ _data http://127.0.0.1:8030/api/example_db/table2/_stream_load
```

The local file `table2_data` uses `|` as the separator between data. The details are as follows:

```
2017-07-03|1|1|jim|2
2017-07-05|2|1|grace|2
2017-07-12|3|2|tom|2
2017-07-15|4|3|bush|3
2017-07-12|5|3|helen|3
```

Note:

1. The recommended file size for Stream Load is less than 10GB. Excessive file size will result in higher retry cost.
2. Each batch of import data should have a Label. Label serves as the unique identifier of the load task, and guarantees that the same batch of data will only be successfully loaded into a database once. For more details, please see Data Loading and Atomicity.
3. Stream Load is a synchronous command. The return of the command indicates that the data has been loaded successfully; otherwise the data has not been loaded.

Broker Load

The Broker Load method imports externally stored data via deployed Broker processes. For more details, please see `HELP BROKER`
↪ `LOAD;`

Example: Use "table1_20170708" as the Label, import files on HDFS into `table1`.

```
LOAD LABEL table1_20170708
(
    DATA INFILE("hdfs://your.namenode.host:port/dir/table1_data")
    INTO TABLE table1
)
WITH BROKER hdfs
(
    "username"="hdfs_user",
    "password"="hdfs_password"
)
PROPERTIES
(
    "timeout"="3600",
    "max_filter_ratio"="0.1"
);
```

The Broker Load is an asynchronous command. Successful execution of it only indicates successful submission of the task. You can check if the import task has been completed by `SHOW LOAD;` . For example:

```
SHOW LOAD WHERE LABEL = "table1_20170708";
```

In the return result, if you find `FINISHED` in the `State` field, that means the import is successful.

For more instructions on `SHOW LOAD`, see`HELP SHOW LOAD;`.

Asynchronous import tasks can be cancelled before it is completed:

```
CANCEL LOAD WHERE LABEL = "table1_20170708";
```

2.3.3.3 Query the Data

2.3.3.3.1 Simple Query

Query example::

```
MySQL> SELECT * FROM table1 LIMIT 3;
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      2 |        1 | 'grace'  |    2 |
|      5 |        3 | 'helen'  |    3 |
|      3 |        2 | 'tom'    |    2 |
+--------+----------+----------+------+
3 rows in set (0.01 sec)

MySQL> SELECT * FROM table1 ORDER BY citycode;
```

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      2 |        1 | 'grace'  |    2 |
|      1 |        1 | 'jim'    |    2 |
|      3 |        2 | 'tom'    |    2 |
|      4 |        3 | 'bush'   |    3 |
|      5 |        3 | 'helen'  |    3 |
+--------+----------+----------+------+
5 rows in set (0.01 sec)
```

2.3.3.3.2 SELECT * EXCEPT

The SELECT * EXCEPT statement is used to exclude one or more columns from the result. The output will not include any of the specified columns.

```
MySQL> SELECT * except (username, citycode) FROM table1;
+--------+------+
| siteid | pv   |
+--------+------+
|      2 |    2 |
|      5 |    3 |
|      3 |    2 |
+--------+------+
3 rows in set (0.01 sec)
```

Note: SELECT * EXCEPT does not exclude columns that do not have a name.

2.3.3.3.3 Join Query

Query example::

```
MySQL> SELECT SUM(table1.pv) FROM table1 JOIN table2 WHERE table1.siteid = table2.siteid;
+--------------------+
| sum(`table1`.`pv`) |
+--------------------+
|                 12 |
+--------------------+
1 row in set (0.20 sec)
```

2.3.3.3.4 Subquery

Query example::

```
MySQL> SELECT SUM(pv) FROM table2 WHERE siteid IN (SELECT siteid FROM table1 WHERE siteid > 2);
+-----------+
| sum(`pv`) |
+-----------+
|         8 |
+-----------+
1 row in set (0.13 sec)
```

2.3.3.4 Change Table Schema

Use the ALTER TABLE COLUMN command to modify the table Schema, including the following changes.

- Adding columns
- Deleting columns
- Modify column types
- Changing the order of columns

The following table structure changes are illustrated by using the following example.

The Schema for the original table1 is as follows:

```
+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | No   | true  | 10      |       |
| citycode | smallint(6) | No   | true  | N/A     |       |
| username | varchar(32) | No   | true  |         |       |
| pv       | bigint(20)  | No   | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
```

We add a new column uv, type BIGINT, aggregation type SUM, default value 0:

```
ALTER TABLE table1 ADD COLUMN uv BIGINT SUM DEFAULT '0' after pv;
```

After successful submission, you can check the progress of the job with the following command:

```
SHOW ALTER TABLE COLUMN;
```

When the job status is `FINISHED`, the job is complete. The new Schema has taken effect.

After ALTER TABLE is completed, you can view the latest Schema via `DESC TABLE`.

```
mysql> DESC table1;
+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | No   | true  | 10      |       |
| citycode | smallint(6) | No   | true  | N/A     |       |
| username | varchar(32) | No   | true  |         |       |
| pv       | bigint(20)  | No   | false | 0       | SUM   |
| uv       | bigint(20)  | No   | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
5 rows in set (0.00 sec)
```

You can cancel the currently executing job with the following command:

```
CANCEL ALTER TABLE COLUMN FROM table1;
```

For more help, see `HELP ALTER TABLE`.

2.3.3.5 Rollup

Rollup can be seen as a materialized index structure for a Table, materialized in the sense that its data is physically independent in storage, and indexed in the sense that Rollup can reorder columns to increase the hit rate of prefix indexes as well as reduce Key columns to increase the aggregation level of data.

You can perform various changes to Rollup using ALTER TABLE ROLLUP.

The following is an exemplified illustration.

The original schema of `table1` is as follows:

```
+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | No   | true  | 10      |       |
| citycode | smallint(6) | No   | true  | N/A     |       |
| username | varchar(32) | No   | true  |         |       |
| pv       | bigint(20)  | No   | false | 0       | SUM   |
| uv       | bigint(20)  | No   | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
```

For `table1`, `siteid`, `citycode`, and `username` constitute a set of Key, based on which the pv fields are aggregated; if you have a frequent need to view the total city pv, you can create a Rollup consisting of only `citycode` and pv:

```
ALTER TABLE table1 ADD ROLLUP rollup_city(citycode, pv);
```

After successful submission, you can check the progress of the task with the following command:

```
SHOW ALTER TABLE ROLLUP;
```

If the task status is `FINISHED`, the job is completed.

After the Rollup is created, you can use `DESC table1 ALL` to check the information of the Rollup.

```
mysql> desc table1 all;
+-------------+----------+-------------+------+-------+--------+-------+
| IndexName   | Field    | Type        | Null | Key   | Default | Extra |
+-------------+----------+-------------+------+-------+---------+-------+
| table1      | siteid   | int(11)     | No   | true  | 10      |       |
|             | citycode | smallint(6) | No   | true  | N/A     |       |
|             | username | varchar(32) | No   | true  |         |       |
|             | pv       | bigint(20)  | No   | false | 0       | SUM   |
|             | uv       | bigint(20)  | No   | false | 0       | SUM   |
|             |          |             |      |       |         |       |
| rollup_city | citycode | smallint(6) | No   | true  | N/A     |       |
|             | pv       | bigint(20)  | No   | false | 0       | SUM   |
+-------------+----------+-------------+------+-------+---------+-------+
8 rows in set (0.01 sec)
```

You can cancel the currently ongoing task using the following command:

```
CANCEL ALTER TABLE ROLLUP FROM table1;
```

With created Rollups, you do not need to specify the Rollup in queries, but only specify the original table for the query. The program will automatically determine if Rollup should be used. You can check whether Rollup is hit or not using the `EXPLAIN your_sql;` command.

For more help, see `HELP ALTER TABLE`.

2.3.3.6 Materialized Views

Materialized views are a space-for-time data analysis acceleration technique. Doris supports building materialized views on top of base tables. For example, a partial column-based aggregated view can be built on top of a table with a granular data model, allowing for fast querying of both granular and aggregated data.

Doris can automatically ensure data consistency between materialized views and base tables, and automatically match the appropriate materialized view at query time, greatly reducing the cost of data maintenance for users and providing a consistent and transparent query acceleration experience.

For more information about materialized views, see Materialized Views

2.3.3.7 Data Table Queries

2.3.3.7.1 Memory Limit

To prevent excessive memory usage of one single query, Doris imposes memory limit on queries. By default, one query task should consume no more than 2GB of memory on one single BE node.

If you find a `Memory limit exceeded` error, that means the program is trying to allocate more memory than the memory limit. You can solve this by optimizing your SQL statements.

You can change the 2GB memory limit by modifying the memory parameter settings.

Show the memory limit for one query:

```
mysql> SHOW VARIABLES LIKE "%mem_limit%";
+---------------+------------+
| Variable_name | Value      |
+---------------+------------+
| exec_mem_limit| 2147483648 |
+----------------+------------+
1 row in set (0.00 sec)
```

exec_mem_limit is measured in byte. You can change the value of exec_mem_limit using the SET command. For example, you can change it to 8GB as follows:

```
mysql> SET exec_mem_limit = 8589934592;
Query OK, 0 rows affected (0.00 sec)
mysql> SHOW VARIABLES LIKE "%mem_limit%";
+----------------+------------+
| Variable_name | Value      |
+----------------+------------+
| exec_mem_limit| 8589934592 |
+----------------+------------+
1 row in set (0.00 sec)
```

- The above change is executed at the session level and is only valid for the currently connected session. The default memory limit will restore after reconnection.
- If you need to change the global variable, you can set: `SET GLOBAL exec_mem_limit = 8589934592;` ↪ . After this, you disconnect and log back in, and then the new parameter will take effect permanently.

2.3.3.7.2 Query Timeout

The default query timeout is set to 300 seconds. If a query is not completed within 300 seconds, it will be cancelled by the Doris system. You change this parameter and customize the timeout for your application to achieve a blocking method similar to wait(timeout).

View the current timeout settings:

```
mysql> SHOW VARIABLES LIKE "%query_timeout%";
+----------------+-------+
| Variable_name | Value |
+----------------+-------+
| QUERY_TIMEOUT | 300   |
+----------------+-------+
1 row in set (0.00 sec)
```

Change query timeout to 1 minute:

```
mysql>  SET query_timeout = 60;
Query OK, 0 rows affected (0.00 sec)
```

- The current timeout check interval is 5 seconds, so if you set the query timeout to less than 5 seconds, it might not be executed accurately.
- The above changes are also performed at the session level. You can change the global variable by `SET` ↪ `GLOBAL`.

2.3.3.7.3 Broadcast/Shuffle Join

The default way to implement Join is to filter the sub table conditionally, broadcast it to each node of the overall table to form a memory Hash table, and then stream read the data of the overall table for Hash Join, but if the filtered data of the sub table cannot be put into memory, the Join will not be completed, and then usually a memory overrun error will occur.

In this case, it is recommended to explicitly specify Shuffle Join, also known as Partitioned Join, where both the sub table and overall table are Hashed according to the Key of the Join and then perform a distributed Join, with the memory consumption being spread across all compute nodes in the cluster.

Doris will automatically attempt a Broadcast Join and switch to a Shuffle Join if the sub table is estimated to be too large; note that if a Broadcast Join is explicitly specified at this point, it will enforce Broadcast Join.

Use Broadcast Join (default):

```
mysql> select sum(table1.pv) from table1 join table2 where table1.siteid = 2;
+--------------------+
| sum(`table1`.`pv`) |
+--------------------+
|                 10 |
+--------------------+
1 row in set (0.20 sec)
```

Use Broadcast Join (explicitly specified):

```
mysql> select sum(table1.pv) from table1 join [broadcast] table2 where table1.siteid = 2;
+--------------------+
| sum(`table1`.`pv`) |
+--------------------+
|                 10 |
+--------------------+
1 row in set (0.20 sec)
```

Use Shuffle Join:

```
mysql> select sum(table1.pv) from table1 join [shuffle] table2 where table1.siteid = 2;
+--------------------+
| sum(`table1`.`pv`) |
+--------------------+
|                 10 |
+--------------------+
1 row in set (0.15 sec)
```

2.3.3.7.4 Query Retry and High Availability

When deploying multiple FE nodes, you can deploy a load balancing layer on top of multiple FEs to achieve high availability of Doris.

Please refer to Load Balancing for details on installation, deployment, and usage.

2.3.3.8 Update and Delete Data

Doris supports two methods to delete imported data. One is to use the `DELETE FROM` statement and specify the target data by the `WHERE` condition. This method is widely applicable and suitable for less frequent scheduled deletion tasks.

The other method is only used in the Unique Models with a unique primary key. It imports the the primary key rows that are to be deleted, and the final physical deletion of the data is performed internally by Doris using the deletion mark. This method is suitable for real-time deletion of data.

For specific instructions on deleting and updating data, see Data Update.

2.3.4 Rollup and query

In multidimensional analysis, Rollups refers to data tables that are aggregated at a specified granular level.

2.3.4.0.1 Basic Concepts

In Doris, tables that are created by users via the CREATE TABLE statement are called “Base Tables”. Base Tables contains the basic data stored in the way specified by the user in the the CREATE TABLE statement.

On top of Base Tables, you can create any number of Rollups. Data in Rollups are generated based on the Base Tables and are physically independent in storage.

Rollups are created to aggregated versions of Base Tables.

Let’s illustrate the ROLLUP tables and their roles in different data models with examples.

ROLLUP in Aggregate Model and Uniq Model

Because Uniq is only a special case of the Aggregate model, we do not distinguish it here.

Example 1: Get the total consumption per user

Following Data Model Aggregate Model in the Aggregate Model section, the Base table structure is as follows:

ColumnName	Type	AggregationType	Comment
user_id	LARGEINT		user id
date	DATE		date of data filling
timestamp	DATETIME		Data filling time, accurate to seconds
city	VARCHAR (20)		User City
age	SMALLINT		User age

ColumnName	Type	AggregationType	Comment
sex	TINYINT		User gender
last_visit_date	DATETIME	REPLACE	Last user access time
cost	BIGINT	SUM	Total User Consumption
max_dwell_time	INT	MAX	Maximum user residence time
min_dwell_time	INT	MIN	User minimum residence time

The data stored are as follows:

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	2017-10-01 08:00:05	Beijing	20	0	2017-10-01 06:00	20	10	10
10000	2017-10-01	2017-10-01 09:00:05	Beijing	20	0	2017-10-01 07:00	15	2	2
10001	2017-10-01	2017-10-01 18:12:10	Beijing	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	2017-10-02 13:10:00	Shanghai	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	2017-10-02 13:15:00	Guangzhou	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	2017-10-01 12:12:48	Shenzhen	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	2017-10-03 12:38:20	Shenzhen	35	0	2017-10-03 10:20:22	11	6	6

On this basis, we create a ROLLUP:

ColumnName
user_id
cost

The ROLLUP contains only two columns: user_id and cost. After the creation, the data stored in the ROLLUP is as follows:

user_id	cost
10000	35
10001	2
10002	200
10003	30
10004	111

As you can see, ROLLUP retains only the results of SUM on the cost column for each user_id. So when we do the following query:

```
SELECT user_id, sum(cost)FROM table GROUP BY user_id;
```

Doris automatically hits the ROLLUP table, thus completing the aggregated query by scanning only a very small amount of data.

2. Example 2: Get the total consumption, the longest and shortest page residence time of users of different ages in different cities

Follow example 1. Based on the Base table, we create a ROLLUP:

ColumnName	Type	AggregationType	Comment
city	VARCHAR (20)		User City
age	SMALLINT		User age
cost	BIGINT	SUM	Total User Consumption
max_dwell_time	INT	MAX	Maximum user residence time
min_dwell_time	INT	MIN	User minimum residence time

After the creation, the data stored in the ROLLUP is as follows:

city	age	cost	max_dwell_time	min_dwell_time
Beijing	20	35	10	2
Beijing	30	2	22	22
Shanghai	20	200	5	5
Guangzhou	32	30	11	11
Shenzhen	35	111	6	3

When we do the following queries:

```
mysql> SELECT city, age, sum(cost), max(max_dwell_time), min(min_dwell_time) FROM table GROUP BY
    ↪ city, age;
mysql> SELECT city, sum(cost), max(max_dwell_time), min(min_dwell_time) FROM table GROUP BY city;
mysql> SELECT city, age, sum(cost), min(min_dwell_time) FROM table GROUP BY city, age;
```

Doris automatically hits the ROLLUP table.

ROLLUP in Duplicate Model

Because the Duplicate model has no aggregate semantics. So the ROLLLUP in this model has lost the meaning of “scroll up”. It’s just to adjust the column order to hit the prefix index. In the next section, we will introduce prefix index in data model prefix index, and how to use ROLLUP to change prefix index in order to achieve better query efficiency.

2.3.4.1 ROLLUP adjusts prefix index

Because column order is specified when a table is built, there is only one prefix index for a table. This may be inefficient for queries that use other columns that cannot hit prefix indexes as conditions. Therefore, we can manually adjust the order of columns by creating ROLLUP. Examples are given.

The structure of the Base table is as follows:

ColumnName	Type
user_id	BIGINT
age	INT
message	VARCHAR(100)
max_dwell_time	DATETIME
min_dwell_time	DATETIME

On this basis, we can create a ROLLUP table:

ColumnName	Type
age	INT
user_id	BIGINT
message	VARCHAR(100)
max_dwell_time	DATETIME
min_dwell_time	DATETIME

As you can see, the columns of ROLLUP and Base tables are exactly the same, just changing the order of user_id and age. So when we do the following query:

```
mysql> SELECT * FROM table where age=20 and message LIKE "%error%";
```

The ROLLUP table is preferred because the prefix index of ROLLUP matches better.

2.3.4.2 Some Explanations of ROLLUP

- The fundamental role of ROLLUP is to improve the query efficiency of some queries (whether by aggregating to reduce the amount of data or by modifying column order to match prefix indexes). Therefore, the meaning of ROLLUP has gone beyond the scope of “roll-up”. That’s why we named it Materialized Index in the source code.
- ROLLUP is attached to the Base table and can be seen as an auxiliary data structure of the Base table. Users can create or delete ROLLUP based on the Base table, but cannot explicitly specify a query for a ROLLUP in the query. Whether ROLLUP is hit or not is entirely determined by the Doris system.
- ROLLUP data is stored in separate physical storage. Therefore, the more ROLLUP you create, the more disk space you occupy. It also has an impact on the speed of import (the ETL phase of import automatically generates all ROLLUP data), but it does not reduce query efficiency (only better).
- Data updates for ROLLUP are fully synchronized with Base representations. Users need not care about this problem.
- Columns in ROLLUP are aggregated in exactly the same way as Base tables. There is no need to specify or modify ROLLUP when creating it.
- A necessary (inadequate) condition for a query to hit ROLLUP is that all columns ** (including the query condition columns in select list and where) involved in the query exist in the column of the ROLLUP. Otherwise, the query can only hit the Base table.
- Certain types of queries (such as count (*)) cannot hit ROLLUP under any conditions. See the next section Limitations of the aggregation model.
- The query execution plan can be obtained by `EXPLAIN your_sql;` command, and in the execution plan, whether ROLLUP has been hit or not can be checked.
- Base tables and all created ROLLUP can be displayed by `DESC tbl_name ALL;` statement.

2.3.4.3 Query

As a polymer view in Doris, Rollup can play two roles in queries:

- Index
- Aggregate data (only for aggregate models, aggregate key)

However, in order to hit Rollup, certain conditions need to be met, and the value of PreAggregation of ScanNdo node in the execution plan can be used to determine whether Rollup can be hit or not, and the Rollup field can be used to determine which Rollup table is hit.

2.3.4.3.1 Index

Doris’s prefix index has been introduced in the previous query practice, that is, Doris will generate the first 36 bytes in the Base/Rollup table separately in the underlying storage engine (with varchar type, the prefix index may be less than 36 bytes, varchar will truncate the prefix index, and use up to 20 bytes of varchar). A sorted sparse index data (data is also sorted, positioned by index, and then searched by dichotomy in the data), and then matched each Base/Rollup prefix index according to the conditions in the query, and selected a Base/Rollup that matched the longest prefix index.

```
        ---> matching from left to right
+----+----+----+----+----+----+
| c1 | c2 | c3 | c4 | c5 |... |
```

As shown in the figure above, the conditions of where and on in the query are pushed up and down to ScanNode and matched from the first column of the prefix index. Check if there are any of these columns in the condition, and then accumulate the matching length until the matching cannot match or the end of 36 bytes (columns of varchar type can only match 20 bytes and match less than 36 words). Section truncates prefix index, and then chooses a Base/Rollup with the longest matching length. The following example shows how to create a Base table and four rollups:

```
+---------------+-------+--------------+------+-------+---------+-------+
| IndexName     | Field | Type         | Null | Key   | Default | Extra |
+---------------+-------+--------------+------+-------+---------+-------+
| test          | 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(9,3) | Yes  | true  | N/A     |       |
|               | k6    | CHAR(5)      | Yes  | true  | N/A     |       |
|               | k7    | DATE         | Yes  | true  | N/A     |       |
|               | k8    | DATETIME     | Yes  | true  | N/A     |       |
|               | k9    | VARCHAR(20)  | Yes  | true  | N/A     |       |
|               | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|               | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
|               |       |              |      |       |         |       |
| rollup_index1 | k9    | VARCHAR(20)  | Yes  | true  | N/A     |       |
|               | 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(9,3) | Yes  | true  | N/A     |       |
|               | k6    | CHAR(5)      | Yes  | true  | N/A     |       |
|               | k7    | DATE         | Yes  | true  | N/A     |       |
|               | k8    | DATETIME     | Yes  | true  | N/A     |       |
|               | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|               | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
|               |       |              |      |       |         |       |
| rollup_index2 | k9    | VARCHAR(20)  | Yes  | true  | N/A     |       |
|               | k2    | SMALLINT     | Yes  | true  | N/A     |       |
|               | k1    | TINYINT      | Yes  | true  | N/A     |       |
|               | k3    | INT          | Yes  | true  | N/A     |       |
|               | k4    | BIGINT       | Yes  | true  | N/A     |       |
|               | k5    | DECIMAL(9,3) | Yes  | true  | N/A     |       |
|               | k6    | CHAR(5)      | Yes  | true  | N/A     |       |
|               | k7    | DATE         | Yes  | true  | N/A     |       |
|               | k8    | DATETIME     | Yes  | true  | N/A     |       |
|               | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|               | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
|               |       |              |      |       |         |       |
| rollup_index3 | k4    | BIGINT       | Yes  | true  | N/A     |       |
|               | k5    | DECIMAL(9,3) | Yes  | true  | N/A     |       |
|               | k6    | CHAR(5)      | Yes  | true  | N/A     |       |
|               | k1    | TINYINT      | Yes  | true  | N/A     |       |
|               | k2    | SMALLINT     | Yes  | true  | N/A     |       |
|               | k3    | INT          | Yes  | true  | N/A     |       |
|               | k7    | DATE         | Yes  | true  | N/A     |       |
|               | k8    | DATETIME     | Yes  | true  | N/A     |       |
|               | k9    | VARCHAR(20)  | Yes  | true  | N/A     |       |
|               | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|               | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
|               |       |              |      |       |         |       |
| rollup_index4 | k4    | BIGINT       | Yes  | true  | N/A     |       |
|               | k6    | CHAR(5)      | Yes  | true  | N/A     |       |
|               | k5    | DECIMAL(9,3) | Yes  | true  | N/A     |       |
|               | k1    | TINYINT      | Yes  | true  | N/A     |       |
|               | k2    | SMALLINT     | Yes  | true  | N/A     |       |
|               | k3    | INT          | Yes  | true  | N/A     |       |
|               | k7    | DATE         | Yes  | true  | N/A     |       |
|               | k8    | DATETIME     | Yes  | true  | N/A     |       |
|               | k9    | VARCHAR(20)  | Yes  | true  | N/A     |       |
|               | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|               | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
+---------------+-------+--------------+------+-------+---------+-------+
```

The prefix indexes of the five tables are

```
Base(k1 ,k2, k3, k4, k5, k6, k7)


rollup_index1(k9)


rollup_index2(k9)
```

```
rollup_index3(k4, k5, k6, k1, k2, k3, k7)

rollup_index4(k4, k6, k5, k1, k2, k3, k7)
```

Conditions on columns that can be indexed with the prefix need to be = < > <= >= in between, and these conditions are side-by-side and the relationship uses and connections', which cannot be hit for or、 != and so on. Then look at the following query:

```
SELECT * FROM test WHERE k1 = 1 AND k2 > 3;
```

With the conditions on K1 and k2, check that only the first column of Base contains K1 in the condition, so match the longest prefix index, test, explain:

```
|   0:OlapScanNode
|      TABLE: test
|      PREAGGREGATION: OFF. Reason: No AggregateInfo
|      PREDICATES: `k1` = 1, `k2` > 3
|      partitions=1/1
|      rollup: test
|      buckets=1/10
|      cardinality=-1
|      avgRowSize=0.0
|      numNodes=0
|      tuple ids: 0
```

Look again at the following queries:

```
SELECT * FROM test WHERE k4 = 1 AND k5 > 3;
```

With K4 and K5 conditions, check that the first column of rollup_index3 and rollup_index4 contains k4, but the second column of rollup_index3 contains k5, so the matching prefix index is the longest.

```
|   0:OlapScanNode
|      TABLE: test
|      PREAGGREGATION: OFF. Reason: No AggregateInfo
|      PREDICATES: `k4` = 1, `k5` > 3
|      partitions=1/1
|      rollup: rollup_index3
|      buckets=10/10
|      cardinality=-1
|      avgRowSize=0.0
|      numNodes=0
|      tuple ids: 0
```

Now we try to match the conditions on the column containing varchar, as follows:

```
SELECT * FROM test WHERE k9 IN ("xxx", "yyyy") AND k1 = 10;
```

There are K9 and K1 conditions. The first column of rollup_index1 and rollup_index2 contains k9. It is reasonable to choose either rollup here to hit the prefix index and randomly select the same one (because there are just 20 bytes in varchar, and the prefix index is truncated in less than 36 bytes). The current strategy here will continue to match k1, because the second rollup_index1 is listed as k1, so rollup_index1 is chosen, in fact, the latter K1 condition will not play an accelerating role. (If the condition outside the prefix index needs to accelerate the query, it can be accelerated by establishing a Bloom Filter filter. Typically for string types, because Doris has a Block level for columns, a Min/Max index for shaping and dates.) The following is the result of explain.

```
|   0:OlapScanNode
|      TABLE: test
|      PREAGGREGATION: OFF. Reason: No AggregateInfo
|      PREDICATES: `k9` IN ('xxx', 'yyyy'), `k1` = 10
|      partitions=1/1
|      rollup: rollup_index1
|      buckets=1/10
|      cardinality=-1
|      avgRowSize=0.0
|      numNodes=0
|      tuple ids: 0
```

Finally, look at a query that can be hit by more than one Rollup:

```
SELECT * FROM test WHERE K4 < 1000 AND K5 = 80 AND K6 = 10000;
```

There are three conditions: k4, K5 and k6. The first three columns of rollup_index3 and rollup_index4 contain these three columns respectively. So the prefix index length matched by them is the same. Both can be selected. The current default strategy is to select a rollup created earlier. Here is rollup_index3.

```
|   0:OlapScanNode
|      TABLE: test
|      PREAGGREGATION: OFF. Reason: No AggregateInfo
|      PREDICATES: `k4` < 1000, `k5` = 80, `k6` >= 10000.0
|      partitions=1/1
|      rollup: rollup_index3
|      buckets=10/10
|      cardinality=-1
|      avgRowSize=0.0
|      numNodes=0
|      tuple ids: 0
```

If you modify the above query slightly as follows:

```
SELECT * FROM test WHERE k4 < 1000 AND k5 = 80 OR k6 >= 10000;
```

The query here cannot hit the prefix index. (Even any Min/Max in the Doris storage engine, the BloomFilter index doesn’t work.)

2.3.4.3.2 Aggregate data

Of course, the function of aggregated data is indispensable for general polymer views. Such materialized views are very helpful for aggregated queries or report queries. To hit the polymer views, the following prerequisites are needed:

1. There is a separate Rollup for all columns involved in a query or subquery.
2. If there is Join in a query or sub-query, the type of Join needs to be Inner join.

The following are some types of aggregated queries that can hit Rollup.

Column type Query type	Sum	Distinct/Count Distinct	Min	Max	APPROX_COUNT_DISTINCT
Key	false	true	true	true	true
Value(Sum)	true	false	false	false	false
Value(Replace)	false	false	false	false	false
Value(Min)	false	false	true	false	false
Value(Max)	false	false	false	true	false

If the above conditions are met, there will be two stages in judging the hit of Rollup for the aggregation model:

1. Firstly, the Rollup table with the longest index hit by prefix index is matched by conditions. See the index strategy above.
2. Then compare the rows of Rollup and select the smallest Rollup.

The following Base table and Rollup:

```
+-------------+-------+--------------+------+-------+---------+-------+
| IndexName   | Field | Type         | Null | Key   | Default | Extra |
+-------------+-------+--------------+------+-------+---------+-------+
| test_rollup | 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(9,3) | Yes  | true  | N/A     |       |
|             | k6    | CHAR(5)      | Yes  | true  | N/A     |       |
|             | k7    | DATE         | Yes  | true  | N/A     |       |
|             | k8    | DATETIME     | Yes  | true  | N/A     |       |
|             | k9    | VARCHAR(20)  | Yes  | true  | N/A     |       |
|             | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|             | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
|             |       |              |      |       |         |       |
| rollup2     | k1    | TINYINT      | Yes  | true  | N/A     |       |
|             | k2    | SMALLINT     | Yes  | true  | N/A     |       |
|             | k3    | INT          | Yes  | true  | N/A     |       |
|             | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|             | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
|             |       |              |      |       |         |       |
| rollup1     | 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(9,3) | Yes  | true  | N/A     |       |
|              | k10   | DOUBLE       | Yes  | false | N/A     | MAX   |
|              | k11   | FLOAT        | Yes  | false | N/A     | SUM   |
+-------------+-------+--------------+------+-------+---------+-------+
```

See the following queries:

```
SELECT SUM(k11) FROM test_rollup WHERE k1 = 10 AND k2 > 200 AND k3 in (1,2,3);
```

Firstly, it judges whether the query can hit the aggregated Rollup table. After checking the graph above, it is possible. Then the condition contains three conditions: k1, K2 and k3. The first three columns of test_rollup, rollup1 and rollup2 contain all the three conditions. So the prefix index length is the same. Then, it is obvious that the aggregation degree of rollup2 is the highest when comparing the number of rows. Row 2 is selected because of the minimum number of rows.

```
|   0:OlapScanNode                                           |
|      TABLE: test_rollup                                    |
|      PREAGGREGATION: ON                                    |
|      PREDICATES: `k1` = 10, `k2` > 200, `k3` IN (1, 2, 3) |
|      partitions=1/1                                        |
|      rollup: rollup2                                       |
|      buckets=1/10                                          |
|      cardinality=-1                                        |
|      avgRowSize=0.0                                        |
|      numNodes=0                                            |
|      tuple ids: 0                                          |
```

2.3.5 Best Practices

2.3.5.1 1 tabulation

2.3.5.1.1 1.1 Data Model Selection

Doris data model is currently divided into three categories: AGGREGATE KEY, UNIQUE KEY, DUPLICATE KEY. Data in all three models are sorted by KEY.

1.1.1. AGGREGATE KEY

When AGGREGATE KEY is the same, old and new records are aggregated. The aggregation functions currently supported are SUM, MIN, MAX, REPLACE.

AGGREGATE KEY model can aggregate data in advance and is suitable for reporting and multi-dimensional analysis business.

```
CREATE TABLE site_visit
(
siteid      INT,
City: SMALLINT,
username VARCHAR (32),
pv BIGINT   SUM DEFAULT '0'
)
AGGREGATE KEY(siteid, city, username)
DISTRIBUTED BY HASH(siteid) BUCKETS 10;
```

1.1.2. UNIQUE KEY

When UNIQUE KEY is the same, the new record covers the old record. Before version 1.2, UNIQUE KEY implements the same REPLACE aggregation method as AGGREGATE KEY, and they are essentially the same. We introduced a new merge-on-write implementation for UNIQUE KEY since version 1.2, which have better performance on many scenarios. Suitable for analytical business with updated requirements.

```
CREATE TABLE sales_order
(
orderid     BIGINT,
status      TINYINT,
username VARCHAR (32),
amount      BIGINT DEFAULT '0'
)
KEY (orderid) UNIT
DISTRIBUTED BY HASH(orderid) BUCKETS 10;
```

1.1.3. DUPLICATE KEY

Only sort columns are specified, and the same rows are not merged. It is suitable for the analysis business where data need not be aggregated in advance.

```
CREATE TABLE session_data
(
visitorid SMALLINT,
sessionid   BIGINT,
visit time DATETIME,
City CHAR (20),
province    CHAR(20),
ip. varchar (32),
brower      CHAR(20),
url: VARCHAR (1024)
)
DUPLICATE KEY (visitor time, session time)
DISTRIBUTED BY HASH(sessionid, visitorid) BUCKETS 10;
```

2.3.5.1.2 1.2 Wide Table vs. Star Schema

When the business side builds tables, in order to adapt to the front-end business, they often do not distinguish between dimension information and indicator information, and define the Schema as a large wide table, this operation is actually not so friendly to the database, we recommend users to use the star model.

- There are many fields in Schema, and there may be more key columns in the aggregation model. The number of columns that need to be sorted in the import process will increase.
- Dimensional information updates are reflected in the whole table, and the frequency of updates directly affects the efficiency of queries.

In the process of using Star Schema, users are advised to use Star Schema to distinguish dimension tables from indicator tables as much as possible. Frequently updated dimension tables can also be placed in MySQL external tables. If there are only a few updates, they can be placed directly in Doris. When storing dimension tables in Doris, more copies of dimension tables can be set up to improve Join’s performance.

2.3.5.1.3 1.3 Partitioning and Bucketing

Doris supports two-level partitioned storage. The first level is partition, which currently supports both RANGE and LIST partition types, and the second layer is HASH bucket.

1.3.1. Partitioning

Partition is used to divide data into different intervals, which can be logically understood as dividing the original table into multiple sub-tables. Data can be easily managed by partition, for example, to delete data more quickly.

1.3.1.1. Range Partitioning

In business, most users will choose to partition on time, which has the following advantages:

- Differentiable heat and cold data
- Availability of Doris Hierarchical Storage (SSD + SATA)

1.3.1.2. List Partitioning

In business„ users can select cities or other enumeration values for partition.

1.3.2. Hash Bucketing

The data is divided into different buckets according to the hash value.

- It is suggested that columns with large differentiation should be used as buckets to avoid data skew.
- In order to facilitate data recovery, it is suggested that the size of a single bucket should not be too large and should be kept within 10GB. Therefore, the number of buckets should be considered reasonably when building tables or increasing partitions, among which different partitions can specify different buckets.

2.3.5.1.4 1.4 Sparse Index and Bloom Filter

Doris stores the data in an orderly manner, and builds a sparse index for Doris on the basis of ordered data. The index granularity is block (1024 rows).

Sparse index chooses fixed length prefix in schema as index content, and Doris currently chooses 36 bytes prefix as index.

- When building tables, it is suggested that the common filter fields in queries should be placed in front of Schema. The more distinguishable the query fields are, the more frequent the query fields are.

- One particular feature of this is the varchar type field. The varchar type field can only be used as the last field of the sparse index. The index is truncated at varchar, so if varchar appears in front, the length of the index may be less than 36 bytes. Specifically, you can refer to data model, ROLLUP and query.
- In addition to sparse index, Doris also provides bloomfilter index. Bloomfilter index has obvious filtering effect on columns with high discrimination. If you consider that varchar cannot be placed in a sparse index, you can create a bloomfilter index.

2.3.5.1.5 1.5 Rollup

Rollup can essentially be understood as a physical index of the original table. When creating Rollup, only some columns in Base Table can be selected as Schema. The order of fields in Schema can also be different from that in Base Table.

Rollup can be considered in the following cases:

1.5.1. Low ratio of data aggregation in the Base Table

This is usually due to the fact that Base Table has more differentiated fields. At this point, you can consider selecting some columns and establishing Rollup.

For the ‘site_visit’table:

```
site -u visit (siteid, city, username, pv)
```

Siteid may lead to a low degree of data aggregation. If business parties often base their PV needs on city statistics, they can build a city-only, PV-based rollup:

```
ALTER TABLE site_visit ADD ROLLUP rollup_city(city, pv);
```

1.5.2. The prefix index in Base Table cannot be hit

Generally, the way Base Table is constructed cannot cover all query modes. At this point, you can consider adjusting the column order and establishing Rollup.

Database Session

```
session -u data (visitorid, sessionid, visittime, city, province, ip, browser, url)
```

In addition to visitorid analysis, there are Brower and province analysis cases, Rollup can be established separately.

```
ALTER TABLE session_data ADD ROLLUP rollup_brower(brower,province,ip,url) DUPLICATE KEY(brower,
    ↪ province);
```

2.3.5.2 Schema Change

Users can modify the Schema of an existing table through the Schema Change operation, currently Doris supports the following modifications:

- Adding and deleting columns
- Modify column types
- Reorder columns
- Adding or modifying Bloom Filter
- Adding or removing bitmap index

For details, please refer to Schema Change

2.3.6 Index

2.3.6.1 Index Overview

Indexes are used to help quickly filter or find data.

Doris currently supports two main types of indexes: 1. built-in smart indexes, including prefix indexes and ZoneMap indexes. 2. User-created secondary indexes, including the bloomfilter index and bitmap index.

The ZoneMap index is the index information automatically maintained for each column in the column storage format, including Min/Max, the number of Null values, and so on. This index is transparent to the user.

2.3.6.1.1 Prefix Index

Unlike traditional database designs, Doris does not support creating indexes on arbitrary columns. an OLAP database with an MPP architecture like Doris is typically designed to handle large amounts of data by increasing concurrency.

Essentially, Doris data is stored in a data structure similar to an SSTable (Sorted String Table). This structure is an ordered data structure that can be stored sorted by specified columns. On this data structure, it will be very efficient to perform lookups with sorted columns as a condition.

In the Aggregate, Unique and Duplicate data models. The underlying data storage is sorted and stored according to the columns specified in the respective table building statements, AGGREGATE KEY, UNIQUE KEY and DUPLICATE KEY.

The prefix index, which is based on sorting, is an indexing method to query data quickly based on a given prefix column.

2.3.6.1.2 Example

We use the first 36 bytes of a row of data as the prefix index of this row of data. Prefix indexes are simply truncated when a VARCHAR type is encountered. We give an example:

1. The prefix index of the following table structure is 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

2. The prefix index of the following table structure is user_name(20 Bytes). Even if it does not reach 36 bytes, because VARCHAR is encountered, it is directly truncated and will not continue further.

ColumnName	Type
user_name	VARCHAR(20)
age	INT
message	VARCHAR(100)

ColumnName	Type
max_dwell_time	DATETIME
min_dwell_time	DATETIME

When our query condition is the prefix of the prefix index, the query speed can be greatly accelerated. For example, in the first example, we execute 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, when building a table, choosing the correct column order can greatly improve query efficiency.

2.3.6.1.3 Adjust prefix index by ROLLUP

Because the column order has been specified when the table is created, there is only one prefix index for a table. This may not be efficient for queries that use other columns that cannot hit the prefix index as conditions. Therefore, we can artificially adjust the column order by creating a ROLLUP. For details, please refer to ROLLUP.

2.3.6.2 BloomFilter index

BloomFilter is a fast search algorithm for multi-hash function mapping proposed by Bloom in 1970. Usually used in some occasions where it is necessary to quickly determine whether an element belongs to a set, but is not strictly required to be 100% correct, BloomFilter has the following characteristics:

- A highly space-efficient probabilistic data structure used to check whether an element is in a set.
- For a call to detect whether an element exists, BloomFilter will tell the caller one of two results: it may exist or it must not exist.
- The disadvantage is that there is a misjudgment, telling you that it may exist, not necessarily true.

Bloom filter is actually composed of an extremely long binary bit array and a series of hash functions. The binary bit array is all 0 initially. When an element to be queried is given, this element will be calculated by a series of hash functions to map out a series of values, and all values are treated as 1 in the offset of the bit array.

Figure below shows an example of Bloom Filter with m=18, k=3 (m is the size of the Bit array, and k is the number of Hash functions). The three elements of x, y, and z in the set are hashed into the bit array through three different hash functions. When querying the element w, after calculating by the Hash function, because one bit is 0, w is not in the set.

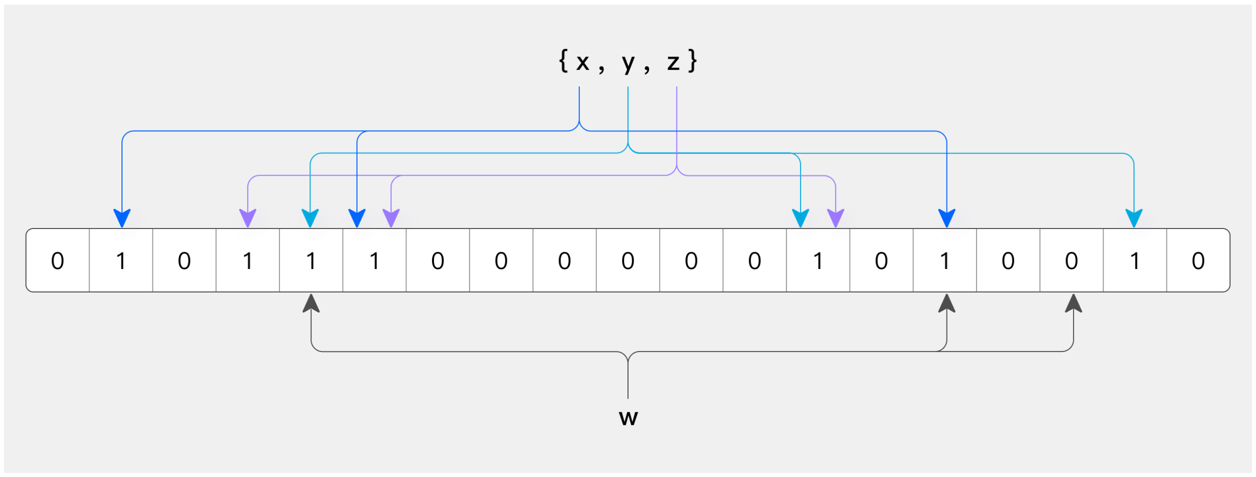


Figure 8: Bloom_filter.svg

So how to judge whether the plot and the elements are in the set? Similarly, all the offset positions of this element are obtained after hash function calculation. If these positions are all 1, then it is judged that this element is in this set, if one is not 1, then it is judged that this element is not in this set. It' s that simple!

2.3.6.2.1 Doris BloomFilter index and usage scenarios

When we use HBase, we know that the Hbase data block index provides an effective method to find the data block of the HFile that should be read when accessing a specific row. But its utility is limited. The default size of the HFile data block is 64KB, and this size cannot be adjusted too much.

If you are looking for a short row, only building an index on the starting row key of the entire data block cannot give you fine-grained index information. For example, if your row occupies 100 bytes of storage space, a 64KB data block contains (64 * 1024)/100 = 655.53 = ~700 rows, and you can only put the starting row on the index bit. The row you are looking for may fall in the row interval on a particular data block, but it is not necessarily stored in that data block. There are many possibilities for this, or the row does not exist in the table, or it is stored in another HFile, or even in MemStore. In these cases, reading data blocks from the hard disk will bring IO overhead and will abuse the data block cache. This can affect performance, especially when you are facing a huge data set and there are many concurrent users.

So HBase provides Bloom filters that allow you to do a reverse test on the data stored in each data block. When a row is requested, first check the Bloom filter to see if the row is not in this data block. The Bloom filter must either confirm that the line is not in the answer, or answer that it doesn' t know. This is why we call it a reverse test. Bloom filters can also be applied to the cells in the row. Use the same reverse test first when accessing a column of identifiers.

Bloom filters are not without cost. Storing this additional index level takes up additional space. Bloom filters grow as their index object data grows, so row-level bloom filters take up less space than column identifier-level bloom filters. When space is not an issue, they can help you squeeze the performance potential of the system.

The BloomFilter index of Doris is specified when the table is built, or completed by the ALTER operation of the table. Bloom Filter is essentially a bitmap structure, used to quickly determine whether a given value is in a set. This judgment will produce a small probability of misjudgment. That is, if it returns false, it must not be in this set. And if the range is true, it may be in this set.

The BloomFilter index is also created with Block as the granularity. In each Block, the value of the specified column is used as a set to generate a BloomFilter index entry, which is used to quickly filter the data that does not meet the conditions in the query.

Let’ s take a look at how Doris creates BloomFilter indexes through examples.

Create BloomFilter Index

The Doris BloomFilter index is created by adding “bloom_filter_columns” = “k1,k2,k3” to the PROPERTIES of the table building statement, this attribute, k1,k2,k3 is the Key column name of the BloomFilter index you want to create, for example, we Create a BloomFilter index for the saler_id and category_id in the table.

```
CREATE TABLE IF NOT EXISTS sale_detail_bloom (
    sale_date date NOT NULL COMMENT "Sales time",
    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)
PARTITION BY RANGE(sale_date)
(
PARTITION P_202111 VALUES [('2021-11-01'), ('2021-12-01'))
)
DISTRIBUTED BY HASH(saler_id) BUCKETS 10
PROPERTIES (
"replication_num" = "3",
"bloom_filter_columns"="saler_id,category_id",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "MONTH",
"dynamic_partition.time_zone" = "Asia/Shanghai",
"dynamic_partition.start" = "-2147483648",
"dynamic_partition.end" = "2",
"dynamic_partition.prefix" = "P_",
"dynamic_partition.replication_num" = "3",
"dynamic_partition.buckets" = "3"
);
```

View BloomFilter Index

Check that the BloomFilter index we built on the table is to use:

```
SHOW CREATE TABLE <table_name>;
```

Delete BloomFilter index

Deleting the index is to remove the index column from the bloom_filter_columns attribute:

```
ALTER TABLE <db.table_name> SET ("bloom_filter_columns" = "");
```

Modify BloomFilter index

Modifying the index is to modify the bloom_filter_columns attribute of the table:

```
ALTER TABLE <db.table_name> SET ("bloom_filter_columns" = "k1,k3");
```

Doris BloomFilter usage scenarios

You can consider establishing a Bloom Filter index for a column when the following conditions are met:

1. First, BloomFilter is suitable for non-prefix filtering.
2. The query will be filtered according to the high frequency of the column, and most of the query conditions are in and = filtering.
3. Unlike Bitmap, BloomFilter is suitable for high cardinality columns. Such as UserID. Because if it is created on a low-cardinality column, such as a “gender” column, each Block will almost contain all values, causing the BloomFilter index to lose its meaning.

Doris BloomFilter use precautions

1. It does not support the creation of Bloom Filter indexes for Tinyint, Float, and Double columns.
2. The Bloom Filter index only has an acceleration effect on in and = filtering queries.
3. If you want to check whether a query hits the Bloom Filter index, you can check the profile information of the query.

2.3.6.3 Bitmap Index

Users can speed up queries by creating a bitmap index This document focuses on how to create an index job, as well as some considerations and frequently asked questions when creating an index.

2.3.6.3.1 Glossary

- bitmap index: a fast data structure that speeds up queries

2.3.6.3.2 Basic Principles

Creating and dropping index is essentially a schema change job. For details, please refer to Schema Change.

2.3.6.3.3 Syntax

Create index

Create a bitmap index for siteid on table1

```
CREATE INDEX [IF NOT EXISTS] index_name ON table1 (siteid) USING BITMAP COMMENT 'balabala';
```

View index

Display the lower index of the specified table_name

```
SHOW INDEX FROM example_db.table_name;
```

Delete index

Delete the lower index of the specified table_name

```
DROP INDEX [IF EXISTS] index_name ON [db_name.]table_name;
```

2.3.6.3.4 Notice

- Currently only index of bitmap type is supported.
- The bitmap index is only created on a single column.
- Bitmap indexes can be applied to all columns of the `Duplicate` , `Uniq` data model and key columns of the `Aggregate` models.
- The data types supported by bitmap indexes are as follows:
  - `TINYINT`
  - `SMALLINT`
  - `INT`
  - `BIGINT`
  - `CHAR`
  - `VARCHAR`
  - `DATE`
  - `DATETIME`
  - `LARGEINT`
  - `DECIMAL`
  - `BOOL`
- The bitmap index takes effect only in segmentV2. The table' s storage format will be converted to V2 automatically when creating index.

More Help

For more detailed syntax and best practices for using bitmap indexes, please refer to the CREARE INDEX / SHOW INDEX / DROP INDEX command manual. You can also enter HELP CREATE INDEX / HELP SHOW INDEX / HELP DROP INDEX on the MySql client command line.

2.4 Data Operation

2.4.1 Import

2.4.1.1 Import Overview

2.4.1.1.1 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	Import Method
Object Storage (s3), HDFS	Import data using Broker
Local file	Import local data
Kafka	Subscribe to Kafka data
Mysql, PostgreSQL, Oracle, SQLServer	Sync data via external table
Import via JDBC	Sync data using JDBC
Import JSON format data	JSON format data import
MySQL Binlog	Binlog Load

Divided by import method

Import method name	Use method
Spark Load	Import external data via Spark
Broker Load	Import external storage data via Broker
Stream Load	Stream import data (local file and memory data)
Routine Load	Import Kafka data
Binlog Load	collect Mysql Binlog import data
Insert Into	External table imports data through INSERT
S3 Load	Object storage data import of S3 protocol

2.4.1.1.2 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

2.4.1.1.3 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

2.4.1.1.4 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.

2.4.1.1.5 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.

2.4.1.1.6 Import the data of array type

The array function can only be supported in vectorization scenarios, but non-vectorization scenarios are not supported. if you want to apply the array function to import data, you should enable vectorization engine. Then you need to cast the input parameter column into the array type according to the parameter of the array function. Finally, you can continue to use the array function.

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" );
```

2.4.1.2 Import Scenes

2.4.1.2.1 Import local data

The following mainly introduces how to import local data in client.

Now Doris support using Stream Load to load data from client local file. Read the following section to learn about how to use Stream Load. 2

Use Stream Load to import local data

Stream Load is used to import local files into Doris.

Unlike the submission methods of other commands, Stream Load communicates with Doris through the HTTP protocol.

The HOST:PORT involved in this method should be the HTTP protocol port.

- BE’s HTTP protocol port, the default is 8040.
- FE’s HTTP protocol port, the default is 8030. However, it must be ensured that the network of the machine where the client is located can connect to the machine where the BE is located.

In this document, we use the curl command as an example to demonstrate how to import data.

At the end of the document, we give a code example of importing data using Java

Import Data

The request body of Stream Load is as follows:

```
PUT /api/{db}/{table}/_stream_load
```

1. Create a table

Use the CREATE TABLE command to create a table in the demo to store the data to be imported. For the specific import method, please refer to the CREATE TABLE command manual. An example is as follows:

```
CREATE TABLE IF NOT EXISTS load_local_file_test
(
    id INT,
    age TINYINT,
    name VARCHAR(50)
)
unique key(id)
DISTRIBUTED BY HASH(id) BUCKETS 3;
```

2. Import data

Execute the following curl command to import the local file:

```
curl -u user:passwd -H "label:load_local_file_test" -T /path/to/local/demo.txt http://host:
    ↪ port/api/demo/load_local_file_test/_stream_load
```

- user:passwd is the user created in Doris. The initial user is admin/root, and the password is blank in the initial state.
- host:port is the HTTP protocol port of BE, the default is 8040, which can be viewed on the Doris cluster WEB UI page.
- label: Label can be specified in the Header to uniquely identify this import task.

For more advanced operations of the Stream Load command, see Stream Load Command documentation.

3. Wait for the import result

The Stream Load command is a synchronous command, and a successful return indicates that the import is successful. If the imported data is large, a longer waiting time may be required. Examples are as follows:

```
{
    "TxnId": 1003,
    "Label": "load_local_file_test",
    "Status": "Success",
    "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 status of the `Status` field is `Success`, which means the import is successful.
- For details of other fields, please refer to the Stream Load command documentation.

Import suggestion

- Stream Load can only import local files.
- It is recommended to limit the amount of data for an import request to 1 - 2 GB. If you have a large number of local files, you can submit them concurrently in batches.

Java code example

Here is a simple JAVA example to execute Stream Load:

```java
package demo.doris;

import org.apache.commons.codec.binary.Base64;
import org.apache.http.HttpHeaders;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.entity.FileEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.DefaultRedirectStrategy;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;

/*
This is an example of Doris Stream Load, which requires dependencies
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
</dependency>
 */
public class DorisStreamLoader {
    //You can choose to fill in the FE address and the http_port of the FE, but the connectivity
        ↪ between the client and the BE node must be guaranteed.
    private final static String HOST = "your_host";
    private final static int PORT = 8040;
    private final static String DATABASE = "db1"; // database to import
    private final static String TABLE = "tbl1"; // table to import
    private final static String USER = "root"; // Doris username
    private final static String PASSWD = ""; // Doris password
    private final static String LOAD_FILE_NAME = "/path/to/1.txt"; // local file path to import

    private final static String loadUrl = String.format("http://%s:%s/api/%s/%s/_stream_load",
            HOST, PORT, DATABASE, TABLE);

    private final static HttpClientBuilder httpClientBuilder = HttpClients
            .custom()
            .setRedirectStrategy(new DefaultRedirectStrategy() {
                @Override
                protected boolean isRedirectable(String method) {
                    // If the connection target is FE, you need to handle 307 redirect.
                    return true;
                }
            });

    public void load(File file) throws Exception {
        try (CloseableHttpClient client = httpClientBuilder.build()) {
            HttpPut put = new HttpPut(loadUrl);
            put.setHeader(HttpHeaders.EXPECT, "100-continue");
            put.setHeader(HttpHeaders.AUTHORIZATION, basicAuthHeader(USER, PASSWD));

            // You can set stream load related properties in Header, here we set label and column
                ↪ _separator.
            put.setHeader("label","label1");
            put.setHeader("column_separator",",");

            // Set the import file.
            // StringEntity can also be used here to transfer arbitrary data.
            FileEntity entity = new FileEntity(file);
            put.setEntity(entity);

            try (CloseableHttpResponse response = client.execute(put)) {
                String loadResult = "";
                if (response.getEntity() != null) {
                    loadResult = EntityUtils.toString(response.getEntity());
                }

                final int statusCode = response.getStatusLine().getStatusCode();
                if (statusCode != 200) {
                    throw new IOException(
                            String.format("Stream load failed. status: %s load result: %s",
                                ↪ statusCode, loadResult));
                }

                System.out.println("Get load result: " + loadResult);
            }
        }
    }

    private String basicAuthHeader(String username, String password) {
        final String tobeEncode = username + ":" + password;
        byte[] encoded = Base64.encodeBase64(tobeEncode.getBytes(StandardCharsets.UTF_8));
        return "Basic " + new String(encoded);
    }

    public static void main(String[] args) throws Exception{
        DorisStreamLoader loader = new DorisStreamLoader();
        File file = new File(LOAD_FILE_NAME);
        loader.load(file);
    }
}
```

Note: The version of http client here is 4.5.13

```
<dependency>
<groupId>org.apache.httpcomponents</groupId>
```

```
<artifactId>httpclient</artifactId>
<version>4.5.13</version>
</dependency>
```

2.4.1.2.2 External storage data import

The following mainly introduces how to import data stored in an external system. For example (HDFS, All object stores that support the S3 protocol) ###### HDFS LOAD

Ready to work

Upload the files to be imported to HDFS. For specific commands, please refer to HDFS upload command

start import

Hdfs load creates an import statement. The import method is basically the same as Broker Load, only need to `WITH BROKER ↪ broker_name ()` statement with the following

```
LOAD LABEL db_name.label_name
(data_desc, ...)
WITH HDFS
[PROPERTIES (key1=value1, ... )]
```

1. create a table

```
CREATE TABLE IF NOT EXISTS load_hdfs_file_test
(
    id INT,
    age TINYINT,
    name VARCHAR(50)
)
unique key(id)
DISTRIBUTED BY HASH(id) BUCKETS 3;
```

2. Import data Execute the following command to import HDFS files:

```
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://testFs",
    "hdfs_user"="user"
    )
    PROPERTIES
    (
    "timeout"="1200",
    "max_filter_ratio"="0.1"
    );
```

For parameter introduction, please refer to Broker Load, HA cluster creation syntax, view through HELP BROKER LOAD

3. Check import status

Broker load is an asynchronous import method. The specific import results can be accessed through SHOW LOAD command to view

```
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; dpp.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)
```

S3 LOAD

Starting from version 0.14, Doris supports the direct import of data from online storage systems that support the S3 protocol through the S3 protocol.

This document mainly introduces how to import data stored in AWS S3. It also supports the import of other object storage systems that support the S3 protocol. ####### Applicable scenarios

- Source data in S3 protocol accessible storage systems, such as S3.
- Data volumes range from tens to hundreds of GB.

Preparing

1. Standard AK and SK First, you need to find or regenerate AWS `Access keys`, you can find the generation method in `My ↪ Security Credentials` of AWS console, as shown in the following figure: AK_SK Select `Create New Access Key` and pay attention to save and generate AK and SK.
2. Prepare REGION and ENDPOINT REGION can be selected when creating the bucket or can be viewed in the bucket list. ENDPOINT can be found through REGION on the following page AWS Documentation

Other cloud storage systems can find relevant information compatible with S3 in corresponding documents

Start Loading

Like Broker Load just replace `WITH BROKER broker_name ()` with

```
WITH S3
(
    "AWS_ENDPOINT" = "AWS_ENDPOINT",
    "AWS_ACCESS_KEY" = "AWS_ACCESS_KEY",
    "AWS_SECRET_KEY"="AWS_SECRET_KEY",
    "AWS_REGION" = "AWS_REGION"
)
```

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"
);
```

FAQ

1. S3 SDK uses 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:

```
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"
)
```

2. Support using temporary security credentials to access object stores that support the S3 protocol:

```
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"
)
```

2.4.1.2.3 Kafka Data Subscription

Users can directly subscribe to message data in Kafka by submitting routine import jobs to synchronize data in near real-time.

Doris itself can ensure that messages in Kafka are subscribed without loss or weight, that is, `Exactly-Once` consumption semantics.

Subscribe to Kafka messages

Subscribing to Kafka messages uses the Routine Load feature in Doris.

The user first needs to create a routine import job. The job will send a series of tasks continuously through routine scheduling, and each task will consume a certain number of messages in Kafka.

Please note the following usage restrictions:

1. Support unauthenticated Kafka access and SSL-authenticated Kafka clusters.
2. The supported message formats are as follows:

- csv text format. Each message is a line, and the end of the line does not contain a newline.
- Json format, see Import Json Format Data.

3. Only supports Kafka 0.10.0.0 (inclusive) and above.

Accessing SSL-authenticated Kafka clusters

The routine import feature supports unauthenticated Kafka clusters, as well as SSL-authenticated Kafka clusters.

Accessing an SSL-authenticated Kafka cluster requires the user to provide a certificate file (ca.pem) for authenticating the Kafka Broker public key. If client authentication is also enabled in the Kafka cluster, the client's public key (client.pem), key file (client.key), and key password must also be provided. The files required here need to be uploaded to Doris through the `CREAE FILE` command, and the catalog name is `kafka`. The specific help of the `CREATE FILE` command can be found in the CREATE FILE command manual . Here is an example:

- upload files

```
CREATE FILE "ca.pem" PROPERTIES("url" = "https://example_url/kafka-key/ca.pem", "catalog" = "
    ↪ kafka");
CREATE FILE "client.key" PROPERTIES("url" = "https://example_urlkafka-key/client.key", "catalog
    ↪ " = "kafka");
CREATE FILE "client.pem" PROPERTIES("url" = "https://example_url/kafka-key/client.pem", "
    ↪ catalog" = "kafka");
```

After the upload is complete, you can view the uploaded files through the SHOW FILES command.

Create a routine import job

For specific commands to create routine import tasks, see ROUTINE LOAD command manual. Here is an example:

1. Access the Kafka cluster without authentication

```
CREATE ROUTINE LOAD demo.my_first_routine_load_job ON test_1
COLUMNS TERMINATED BY ","
PROPERTIES
(
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
)
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"
);
```

- `max_batch_interval/max_batch_rows/max_batch_size` is used to control the running period of a subtask. The running period of a subtask is determined by the longest running time, the maximum number of rows consumed, and the maximum amount of data consumed.

2. Access an SSL-authenticated Kafka cluster

```
CREATE ROUTINE LOAD demo.my_first_routine_load_job ON test_1
COLUMNS TERMINATED BY ",",
PROPERTIES
(
    "max_batch_interval" = "20",
    "max_batch_rows" = "300000",
    "max_batch_size" = "209715200",
)
FROM KAFKA
(
   "kafka_broker_list"= "broker1:9091,broker2:9091",
   "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"
);
```

View import job status

See SHOW ROUTINE LOAD for specific commands and examples for checking the status of a job) command documentation.

See SHOW ROUTINE LOAD TASK command documentation.

Only the currently running tasks can be viewed, and the completed and unstarted tasks cannot be viewed.

Modify job properties

The user can modify some properties of the job that has been created. For details, please refer to the ALTER ROUTINE LOAD command manual.

Job Control

The user can control the stop, pause and restart of the job through the `STOP/PAUSE/RESUME` three commands.

For specific commands, please refer to STOP ROUTINE LOAD , PAUSE ROUTINE LOAD, RESUME ROUTINE LOAD command documentation.

more help

For more detailed syntax and best practices for ROUTINE LOAD, see ROUTINE LOAD command manual.

2.4.1.2.4 Synchronize data through external table

Doris can create external tables. Once created, you can query the data of the external table directly with the SELECT statement or import the data of the external table with the `INSERT INTO SELECT` method.

Doris external tables currently support the following data sources.

- MySQL

- Oracle
- PostgreSQL
- SQLServer
- Hive
- Iceberg
- ElasticSearch

This document describes how to create external tables accessible through the ODBC protocol and how to import data from these external tables.

Create external table

For a detailed introduction to creating ODBC external tables, please refer to the CREATE ODBC TABLE syntax help manual.

Here is just an example of how to use it.

1. Create an ODBC Resource

The purpose of ODBC Resource is to manage the connection information of external tables uniformly.

```
   CREATE EXTERNAL RESOURCE `oracle_test_odbc`
   PROPERTIES (
       "type" = "odbc_catalog",
       "host" = "192.168.0.1",
       "port" = "8086",
       "user" = "oracle",
       "password" = "oracle",
       "database" = "oracle",
       "odbc_type" = "oracle",
       "driver" = "Oracle"
   );
```

Here we have created a Resource named `oracle_test_odbc`, whose type is `odbc_catalog`, indicating that this is a Resource used to store ODBC information. `odbc_type` is `oracle`, indicating that this OBDC Resource is used to connect to the Oracle database. For other types of resources, see the resource management documentation for details.

2. Create an external table

```
CREATE EXTERNAL TABLE `ext_oracle_demo` (
  `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 ""
) ENGINE=ODBC
COMMENT "ODBC"
PROPERTIES (
    "odbc_catalog_resource" = "oracle_test_odbc",
    "database" = "oracle",
    "table" = "baseall"
);
```

Here we create an `ext_oracle_demo` external table and reference the `oracle_test_odbc` Resource created earlier

Import Data

1. Create the Doris table

Here we create a Doris table with the same column information as the external table `ext_oracle_demo` created in the previous step:

```
  CREATE TABLE `doris_oralce_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"
  );
```

For detailed instructions on creating Doris tables, see CREATE-TABLE syntax help.

2. Import data (from `ext_oracle_demo` table to `doris_oralce_tbl` table)

```
  INSERT INTO doris_oralce_tbl SELECT k1,k2,k3 FROM ext_oracle_demo limit 200
```

The INSERT command is a synchronous command, and a successful return indicates that the import was successful.

Precautions

- It must be ensured that the external data source and the Doris cluster can communicate with each other, including the network between the BE node and the external data source.
- ODBC external tables essentially access the data source through a single ODBC client, so it is not suitable to import a large amount of data at one time. It is recommended to import multiple times in batches.

2.4.1.2.5 Synchronize data using Insert method

Users can use INSERT statement to import data through MySQL protocol.

The INSERT statement is used in a similar way to the INSERT statement used in databases such as MySQL. The INSERT statement supports the following two syntaxes:

```
* INSERT INTO table SELECT ...
* INSERT INTO table VALUES(...)
```

Here we only introduce the second way. For a detailed description of the INSERT command, see the INSERT command documentation.

Single write

Single write means that the user directly executes an INSERT command. An example is as follows:

```
INSERT INTO example_tbl (col1, col2, col3) VALUES (1000, "test", 3.25);
```

For Doris, an INSERT command is a complete import transaction.

Therefore, whether it is importing one piece of data or multiple pieces of data, we do not recommend using this method for data import in the production environment. The INSERT operation of high-frequency words will result in a large number of small files in the storage layer, which will seriously affect the system performance.

This method is only used for simple offline tests or low-frequency operations.

Or you can use the following methods for batch insert operations:

```
INSERT INTO example_tbl VALUES
(1000, "baidu1", 3.25)
(2000, "baidu2", 4.25)
(3000, "baidu3", 5.25);
```

We recommend that the number of inserts in a batch be as large as possible, such as thousands or even 10,000 at a time. Or you can use PreparedStatement to perform batch inserts through the following procedure.

JDBC example

Here we give a simple JDBC batch INSERT code example:

```
package demo.doris;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class DorisJDBCDemo {

    private static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String DB_URL_PATTERN = "jdbc:mysql://%s:%d/%s?rewriteBatchedStatements=
        ↪ true";
    private static final String HOST = "127.0.0.1"; // Leader Node host
    private static final int PORT = 9030; // query port of Leader Node
    private static final String DB = "demo";
    private static final String TBL = "test_1";
    private static final String USER = "admin";
    private static final String PASSWD = "my_pass";

    private static final int INSERT_BATCH_SIZE = 10000;

    public static void main(String[] args) {
        insert();
    }

    private static void insert() {
        // Be careful not to add a semicolon ";" at the end
        String query = "insert into " + TBL + " values(?, ?)";
        // Set Label to be idempotent.
        // String query = "insert into " + TBL + " WITH LABEL my_label values(?, ?)";

        Connection conn = null;
        PreparedStatement stmt = null;
        String dbUrl = String.format(DB_URL_PATTERN, HOST, PORT, DB);
        try {
            Class.forName(JDBC_DRIVER);
            conn = DriverManager.getConnection(dbUrl, USER, PASSWD);
            stmt = conn.prepareStatement(query);

            for (int i =0; i < INSERT_BATCH_SIZE; i++) {
                stmt.setInt(1, i);
                stmt.setInt(2, i * 100);
                stmt.addBatch();
            }

            int[] res = stmt.executeBatch();
            System.out.println(res);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (stmt != null) {
                    stmt.close();
                }
            } catch (SQLException se2) {
                se2.printStackTrace();
            }
            try {
                if (conn != null) conn.close();
            } catch (SQLException se) {
                se.printStackTrace();
            }
        }
    }
}
```

Please note the following:

1. The JDBC connection string needs to add the `rewriteBatchedStatements=true` parameter and use the `PreparedStatement` ↪ method.

Currently, Doris does not support PrepareStatemnt on the server side, so the JDBC Driver will perform batch Prepare on the client side.

`rewriteBatchedStatements=true` will ensure that the Driver executes batches. And finally form an INSERT statement of the following form and send it to Doris:

```
INSERT INTO example_tbl VALUES
(1000, "baidu1", 3.25)
(2000, "baidu2", 4.25)
(3000, "baidu3", 5.25);
```

2. Batch size

Because batch processing is performed on the client, if a batch is too large, it will occupy the memory resources of the client, so you need to pay attention.

Doris will support PrepareStatemnt on the server in the future, so stay tuned.

3. Import atomicity

Like other import methods, the INSERT operation itself supports atomicity. Each INSERT operation is an import transaction, which guarantees atomic writing of all data in an INSERT.

As mentioned earlier, we recommend that when using INSERT to import data, use the “batch” method to import, rather than a single insert.

At the same time, we can set a Label for each INSERT operation. Through the Label mechanism, the idempotency and atomicity of operations can be guaranteed, and the data will not be lost or heavy in the end. For the specific usage of Label in INSERT, you can refer to the INSERT document.

2.4.1.2.6 Data import things and atomicity

All import operations in Doris have atomicity guarantees, that is, the data in an import job either all succeed or all fail. It will not happen that only part of the data is imported successfully.

In BROKER LOAD we can also implement atomic import of multiple tables .

For the materialized view attached to the table, atomicity and consistency with the base table are also guaranteed.

Label mechanism

Doris' s import job can set a Label. This Label is usually a user-defined string with certain business logic attributes.

The main function of Label is to uniquely identify an import task, and to ensure that the same Label will only be successfully imported once.

The Label mechanism can ensure that the imported data is not lost or heavy. If the upstream data source can guarantee the At-Least-Once semantics, with the Doris Label mechanism, the Exactly-Once semantics can be guaranteed.

Label is unique under a database. The retention period for labels is 3 days by default. That is, after 3 days, the completed Label will be automatically cleaned up, and then the Label can be reused.

Best Practices

Labels are usually formatted as `business logic + time`. Such as `my_business1_20220330_125000`.

This Label is usually used to represent: a batch of data generated by the business `my_business1` at `2022-03-30 12:50:00`. Through this Label setting, the business can query the import task status through the Label to clearly know whether the batch data has been imported successfully at this point in time. If unsuccessful, you can continue to retry the import using this Label

2.4.1.2.7 Data transformation, column mapping and filtering

Supported import methods

- 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
...
```

The above import methods all support column mapping, transformation and filtering operations on the source data:

- Pre-filtering: filter the read raw data once.

```
PRECEDING FILTER k1 = 1
```

- Mapping: Define the columns in the source data. If the defined column name is the same as the column in the table, it is directly mapped to the column in the table. If different, the defined column can be used for subsequent transformation operations. As in the example above:

```
(k1, k2, tmpk3)
```

- Conversion: Convert the mapped columns in the first step, you can use built-in expressions, functions, and custom functions for conversion, and remap them to the corresponding columns in the table. As in the example above:

```
k3 = tmpk3 + 1
```

- Post filtering: Filter the mapped and transformed columns by expressions. Filtered data rows are not imported into the system. As in the example above:

```
WHERE k1 > k2
```

column mapping

The purpose of column mapping is mainly to describe the information of each column in the import file, which is equivalent to defining the name of the column in the source data. By describing the column mapping relationship, we can import source files with different column order and different number of columns into Doris. Below we illustrate with an example:

Assuming that the source file has 4 columns, the contents are as follows (the header column names are only for convenience, and there is no header actually):

Column 1	Column 2	Column 3	Column 4
1	100	beijing	1.1
2	200	shanghai	1.2
3	300	guangzhou	1.3
4	`\N`	chongqing	1.4

Note: `\N` means null in the source file.

1. Adjust the mapping order

Suppose there are 4 columns `k1`,`k2`,`k3`,`k4` in the table. The import mapping relationship we want is as follows:

```
column 1 -> k1
column 2 -> k3
column 3 -> k2
column 4 -> k4
```

Then the column mapping should be written in the following order:

```
(k1, k3, k2, k4)
```

2. There are more columns in the source file than in the table

Suppose there are 3 columns `k1`,`k2`,`k3` in the table. The import mapping relationship we want is as follows:

```
column 1 -> k1
column 2 -> k3
column 3 -> k2
```

Then the column mapping should be written in the following order:

```
(k1, k3, k2, tmpk4)
```

where `tmpk4` is a custom column name that does not exist in the table. Doris ignores this non-existing column name.

3. The number of columns in the source file is less than the number of columns in the table, fill with default values

Suppose there are 5 columns `k1`,`k2`,`k3`,`k4`,`k5` in the table. The import mapping relationship we want is as follows:

```
column 1 -> k1
column 2 -> k3
column 3 -> k2
```

Here we only use the first 3 columns from the source file. The two columns `k4,k5` want to be filled with default values.

Then the column mapping should be written in the following order:

```
(k1, k3, k2)
```

If the `k4,k5` columns have default values, the default values will be populated. Otherwise, if it is a `nullable` column, it will be populated with a `null` value. Otherwise, the import job will report an error.

Column pre-filtering

Pre-filtering is to filter the read raw data once. Currently only BROKER LOAD and ROUTINE LOAD are supported.

Pre-filtering has the following application scenarios:

1. Filter before conversion

Scenarios where you want to filter before column mapping and transformation. It can filter out some unwanted data first.

2. The filter column does not exist in the table, it is only used as a filter identifier

For example, the source data stores the data of multiple tables (or the data of multiple tables is written to the same Kafka message queue). Each row in the data has a column name to identify which table the row of data belongs to. Users can filter the corresponding table data for import by pre-filtering conditions.

Column conversion

The column transformation function allows users to transform column values in the source file. Currently, Doris supports most of the built-in functions and user-defined functions for conversion.

> Note: The user-defined function belongs to a certain database. When using the user-defined function for conversion, the user needs to have read permission to this database.

Transformation operations are usually defined along with column mappings. That is, the columns are first mapped and then converted. Below we illustrate with an example:

Assuming that the source file has 4 columns, the contents are as follows (the header column names are only for convenience, and there is no header actually):

Column 1	Column 2	Column 3	Column 4
1	100	beijing	1.1
2	200	shanghai	1.2
3	300	guangzhou	1.3
`\N`	400	chongqing	1.4

1. Convert the column values in the source file and import them into the table

Suppose there are 4 columns k1 , k2 , k3 , k4 in the table. Our desired import mapping and transformation relationship is as follows:

```
column 1 -> k1
column 2 * 100 -> k3
column 3 -> k2
column 4 -> k4
```

Then the column mapping should be written in the following order:

```
(k1, tmpk3, k2, k4, k3 = tmpk3 * 100)
```

This is equivalent to us naming the second column in the source file `tmpk3`, and specifying that the value of the `k3` column in the table is `tmpk3 * 100`. The data in the final table is as follows:

k1	k2	k3	k4
1	beijing	10000	1.1
2	shanghai	20000	1.2
3	guangzhou	30000	1.3
null	chongqing	40000	1.4

2. Through the case when function, column conversion is performed conditionally.

Suppose there are 4 columns k1 , k2 , k3 , k4 in the table. We hope that `beijing, shanghai, guangzhou, chongqing` in the source data are converted to the corresponding region ids and imported:

```
column 1 -> k1
column 2 -> k2
Column 3 after region id conversion -> k3
column 4 -> k4
```

Then the column mapping should be written in the following order:

```
(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)
```

The data in the final table is 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

3. Convert the null value in the source file to 0 and import it. At the same time, the region id conversion in Example 2 is also performed.

Suppose there are 4 columns k1,k2,k3,k4 in the table. While converting the region id, we also want to convert the null value of the k1 column in the source data to 0 and import:

```
Column 1 is converted to 0 if it is null -> k1
column 2 -> k2
column 3 -> k3
column 4 -> k4
```

Then the column mapping should be written in the following order:

```
(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)
```

The data in the final table is as follows:

k1	k2	k3	k4
1	100	1	1.1
2	200	2	1.2
3	300	3	1.3
0	400	4	1.4

List filter

After column mapping and transformation, we can filter the data that we do not want to import into Doris through filter conditions. Below we illustrate with an example:

Assuming that the source file has 4 columns, the contents are as follows (the header column names are only for convenience, and there is no header actually):

Column 1	Column 2	Column 3	Column 4
1	100	beijing	1.1
2	200	shanghai	1.2
3	300	guangzhou	1.3
\N	400	chongqing	1.4

1. In the default case of column mapping and transformation, filter directly

Suppose there are 4 columns k1,k2,k3,k4 in the table. We can define filter conditions directly with default column mapping and transformation. If we want to import only the data rows whose fourth column in the source file is greater than 1.2, the filter conditions are as follows:

```
where k4 > 1.2
```

The data in the final table is as follows:

k1	k2	k3	k4
3	300	guangzhou	1.3
null	400	chongqing	1.4

By default, Doris maps columns sequentially, so column 4 in the source file is automatically mapped to column k4 in the table.

2. Filter the column-transformed data

Suppose there are 4 columns `k1,k2,k3,k4` in the table. In the column conversion example, we converted province names to ids. Here we want to filter out the data with id 3. Then the conversion and filter conditions are 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
```

The data in the final table is as follows:

k1	k2	k3	k4
1	100	1	1.1
2	200	2	1.2
null	400	4	1.4

Here we see that the column value when performing the filter is the final column value after mapping and transformation, not the original data.

3. Multi-condition filtering

Suppose there are 4 columns `k1,k2,k3,k4` in the table. We want to filter out the data whose k1 column is `null`, and at the same time filter out the data whose k4 column is less than 1.2, the filter conditions are as follows:

```
where k1 is not null and k4 >= 1.2
```

The data in the final table is as follows:

k1	k2	k3	k4
2	200	2	1.2
3	300	3	1.3

Data Quality Issues and Filtering Thresholds

The rows of data processed in an import job can be divided into the following three types:

1. Filtered Rows

Data that was filtered out due to poor data quality. Unqualified data quality includes data format problems such as type error, precision error, long string length, mismatched file column number, and data rows that are filtered out because there is no corresponding partition.

2. Unselected Rows

This part is the row of data that was filtered out due to `preceding filter` or `where` column filter conditions.

3. Loaded Rows

Rows of data being imported correctly.

Doris' s import task allows the user to set a maximum error rate (`max_filter_ratio`). If the error rate of the imported data is below the threshold, those erroneous rows will be ignored and other correct data will be imported.

The error rate is calculated as:

```
#####Filtered Rows / (#Filtered Rows + #Loaded Rows)
```

That is to say, `Unselected` Rows will not participate in the calculation of the error rate.

2.4.1.2.8 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" = "xxxxxxxxxxxxxxxxxxxxxxxxxx",
    "bos_secret_accesskey"="yyyyyyyyyyyyyyyyyyyyyyyyy"
)
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

Strict mode means strict 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	`\N`	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	`\N`	null	On or Off	NULL
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.

2.4.1.3 Import Way

2.4.1.3.1 Binlog Load

The Binlog Load feature enables Doris to incrementally synchronize update operations in MySQL, so as to CDC(Change Data Capture) of data on Mysql.

Scenarios

- Support insert / update / delete operations
- Filter query
- Temporarily incompatible with DDL statements

Principle

In the design of phase one, Binlog Load needs to rely on canal as an intermediate medium, so that canal can be pretended to be a slave node to get and parse the binlog on the MySQL master node, and then Doris can get the parsed data on the canal. This process mainly involves mysql, canal and Doris. The overall data flow is as follows:

```
+---------------------------------------------+
|                    Mysql                    |
+----------------------+----------------------+
                       | Binlog
+----------------------v----------------------+
|                 Canal Server                |
+-------------------+-----^-------------------+
               Get  |     |  Ack
+-------------------|-----|-------------------+
| FE                |     |                   |
| +-----------------|-----|----------------+  |
| | Sync Job        |     |                |  |
| |    +------------v-----+-----------+    |  |
| |    | Canal Client                 |    |  |
| |    |   +-----------------------+  |    |  |
| |    |   |       Receiver        |  |    |  |
| |    |   +-----------------------+  |    |  |
| |    |   +-----------------------+  |    |  |
| |    |   |       Consumer        |  |    |  |
| |    |   +-----------------------+  |    |  |
| |    +------------------------------+    |  |
| +----+---------------+--------------+----+  |
|      |               |              |       |
| +----v-----+   +-----v----+   +-----v----+  |
| | Channel1 |   | Channel2 |   | Channel3 |  |
| | [Table1] |   | [Table2] |   | [Table3] |  |
| +----+-----+   +-----+----+   +-----+----+  |
|      |               |              |       |
|   +--|-------+   +---|------+   +---|------+|
|  +---v------+|  +----v-----+|  +----v-----+||
| +----------+|+ +----------+|+ +----------+|+|
| |   Task   |+  |   Task   |+  |   Task   |+ |
| +----------+   +----------+   +----------+  |
+----------------------+----------------------+
     |                 |                  |
+----v-----------------v------------------v---+
|                 Coordinator                 |
|                          BE                          |
+----+-----------------+------------------+---+
     |                 |                  |
+----v---+         +---v----+         +----v---+
|   BE   |         |   BE   |         |   BE   |
+--------+         +--------+         +--------+
```

As shown in the figure above, the user first submits a SyncJob to the Fe.

Then, Fe will start a Canal Client for each SyncJob to subscribe to and get data from the Canal Server.

The Receiver in the Canal Client will receives data by the GET request. Every time a Batch is received, it will be distributed by the Consumer to different Channels according to the corresponding target table. Once a channel received data distributed by Consumer, it will submit a send task for sending data.

A Send task is a request from Channel to Be, which contains the data of the same Batch distributed to the current channel.

Channel controls the begin, commit and abort of transaction of single table. In a transaction, the consumer may distribute multiple Batches of data to a channel, so multiple send tasks may be generated. These tasks will not actually take effect until the transaction is committed successfully.

When certain conditions are met (for example, a certain period of time was passed, reach the maximun data size of commit), the Consumer will block and notify each channel to try commit the transaction.

If and only if all channels are committed successfully, Canal Server will be notified by the ACK request and Canal Client continue to get and consume data.

If there are any Channel fails to commit, it will retrieve data from the location where the last consumption was successful and commit again (the Channel that has successfully commited before will not commmit again to ensure the idempotency of commit).

In the whole cycle of a SyncJob, Canal Client continuously received data from Canal Server and send it to Be through the above process to complete data synchronization.

Configure MySQL Server

In the master-slave synchronization of MySQL Cluster mode, the binary log file (binlog) records all data changes on the master node. Data synchronization and backup among multiple nodes of the cluster should be carried out through binlog logs, so as to improve the availability of the cluster.

The architecture of master-slave synchronization is usually composed of a master node (responsible for writing) and one or more slave nodes (responsible for reading). All data changes on the master node will be copied to the slave node.

Note that: Currently, you must use MySQL version 5.7 or above to support Binlog Load

To enable the binlog of MySQL, you need to edit the my.cnf file and set it like:

```
[mysqld]
log-bin = mysql-bin # Enable binlog
binlog-format=ROW # Select ROW mode
```

Principle Description

On MySQL, the binlog files usually name as mysql-bin.000001, mysql-bin.000002⋯ And MySQL will automatically segment the binlog file when certain conditions are met:

1. MySQL is restarted
2. The user enters the `flush logs` command
3. The binlog file size exceeds 1G

To locate the latest consumption location of binlog, the binlog file name and position (offset) must be needed.

For instance, the binlog location of the current consumption so far will be saved on each slave node to prepare for disconnection, reconnection and continued consumption at any time.

```
---------------------                                 ---------------------
|       Slave       |               read              |       Master      |
| FileName/Position | <<<------------------------------ |    Binlog Files   |
---------------------                                 ---------------------
```

For the master node, it is only responsible for writing to the binlog. Multiple slave nodes can be connected to a master node at the same time to consume different parts of the binlog log without affecting each other.

Binlog log supports two main formats (in addition to mixed based mode):

- Statement-based format:

  Binlog only records the SQL statements executed on the master node, and the slave node copies them to the local node for re-execution.

- Row-based format:

  Binlog will record the data change information of each row and all columns of the master node, and the slave node will copy and execute the change of each row to the local node.

The first format only writes the executed SQL statements. Although the log volume will be small, it has the following disadvantages:

1. The actual data of each row is not recorded
2. The UDF, random and time functions executed on the master node will have inconsistent results on the slave node
3. The execution order of limit statements may be inconsistent

Therefore, we need to choose the second format which parses each row of data from the binlog log.

In the row-based format, binlog will record the timestamp, server ID, offset and other information of each binlog event. For instance, the following transaction with two insert statements:

```
begin;
insert into canal_test.test_tbl values (3, 300);
insert into canal_test.test_tbl values (4, 400);
commit;
```

There will be four binlog events, including one begin event, two insert events and one commit event:

```
SET TIMESTAMP=1538238301/*!*/;
BEGIN
/*!*/.
```

```
##### at 211935643
##### at 211935698
#####180930 0:25:01 server id 1 end_log_pos 211935698 Table_map: 'canal_test'.'test_tbl' mapped
    ↪ to number 25
#####180930 0:25:01 server id 1 end_log_pos 211935744 Write_rows: table-id 25 flags: STMT_END_F
...
'/*!*/;
####### INSERT INTO canal_test.test_tbl
####### SET
####### @1=1
####### @2=100
##### at 211935744
#####180930 0:25:01 server id 1 end_log_pos 211935771 Xid = 2681726641
...
'/*!*/;
####### INSERT INTO canal_test.test_tbl
####### SET
####### @1=2
####### @2=200
##### at 211935771
#####180930 0:25:01 server id 1 end_log_pos 211939510 Xid = 2681726641
COMMIT/*!*/;
```

As shown above, each insert event contains modified data. During delete/update, an event can also contain multiple rows of data, making the binlog more compact.

Open GTID mode (Optional)

A global transaction ID (global transaction identifier) identifies a transaction that has been committed on the master node, which is unique and valid in global. After binlog is enabled, the gtid will be written to the binlog file.

To open the gtid mode of MySQL, you need to edit the my.cnf configuration file and set it like:

```
gtid-mode=on // Open gtid mode
enforce-gtid-consistency=1 // Enforce consistency between gtid and transaction
```

In gtid mode, the master server can easily track transactions, recover data and replicas without binlog file name and offset.

In gtid mode, due to the global validity of gtid, the slave node will no longer need to locate the binlog location on the master node by saving the file name and offset, but can be located by the data itself. During SyncJob, the slave node will skip the execution of any gtid transaction already executed before.

Gtid is expressed as a pair of coordinates, `source_ID` identifies the master node, `transaction_ID` indicates the order in which this transaction is executed on the master node (max 263-1).

```
GTID = source_id:transaction_id
```

For example, the gtid of the 23rd transaction executed on the same master node is:

```
3E11FA47-71CA-11E1-9E33-C80AA9429562:23
```

Configure Canal Server

Canal is a sub project of Alibaba Otter project. Its main purpose is to provide incremental data subscription and consumption based on MySQL database binlog analysis, which is originally used to solve the scenario of cross machine-room synchronization.

Canal version 1.1.5 and above is recommended. download link

After downloading, please follow the steps below to complete the deployment.

1. Unzip the canal deployer
2. Create a new directory under the conf folder and rename it as the root directory of instance. The directory name is the destination mentioned later.
3. Modify the instance configuration file (you can copy from `conf/example/instance.properties`)

```
vim conf/{your destination}/instance.properties
```

```
## canal instance serverId
canal.instance.mysql.slaveId = 1234
## mysql adress
canal.instance.master.address = 127.0.0.1:3306
## mysql username/password
canal.instance.dbUsername = canal
canal.instance.dbPassword = canal
```

4. start up canal server

```
sh bin/startup.sh
```

5. Validation start up successfully

```
cat logs/{your destination}/{your destination}.log
```

```
2013-02-05 22:50:45.636 [main] INFO  c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer -
    ↪  Loading properties file from class path resource [canal.properties]
2013-02-05 22:50:45.641 [main] INFO  c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer -
    ↪  Loading properties file from class path resource [xxx/instance.properties]
2013-02-05 22:50:45.803 [main] INFO  c.a.otter.canal.instance.spring.CanalInstanceWithSpring
    ↪ - start CannalInstance for 1-xxx
2013-02-05 22:50:45.810 [main] INFO  c.a.otter.canal.instance.spring.CanalInstanceWithSpring
    ↪ - start successful....
```

Canal End Description

By faking its own MySQL dump protocol, canal disguises itself as a slave node, get and parses the binlog of the master node.

Multiple instances can be started on the canal server. An instance can be regarded as a slave node. Each instance consists of the following parts:

```
-------------------------------------------------
|  Server                                        |
|  -------------------------------------------- |
|  |  Instance 1                              | |
|  |  -----------   -----------  -----------  | |
|  |  |  Parser |   |  Sink   |  | Store   |  | |
|  |  -----------   -----------  -----------  | |
|  |  -----------------------------------     | |
|  |  |             MetaManager         |     | |
|  |  -----------------------------------     | |
|  -------------------------------------------- |
-------------------------------------------------
```

- Parser: Access the data source, simulate the dump protocol, interact with the master, and analyze the protocol
- Sink: Linker between parser and store, for data filtering, processing and distribution
- Store: Data store
- Meta Manager: Metadata management module

Each instance has its own unique ID in the cluster, that is, server ID.

In the canal server, the instance is identified by a unique string named destination. The canal client needs destination to connect to the corresponding instance.

Note that: canal client and canal instance should correspond to each other one by one

Binlog load has forbidded multiple SyncJobs to connect to the same destination.

The data flow direction in instance is binlog -> Parser -> sink -> store.

Instance parses binlog logs through the parser module, and the parsed data is cached in the store. When the user submits a SyncJob to Fe, it will start a Canal Client to subscripe and get the data in the store in the corresponding instance.

The store is actually a ring queue. Users can configure its length and storage space by themselves.

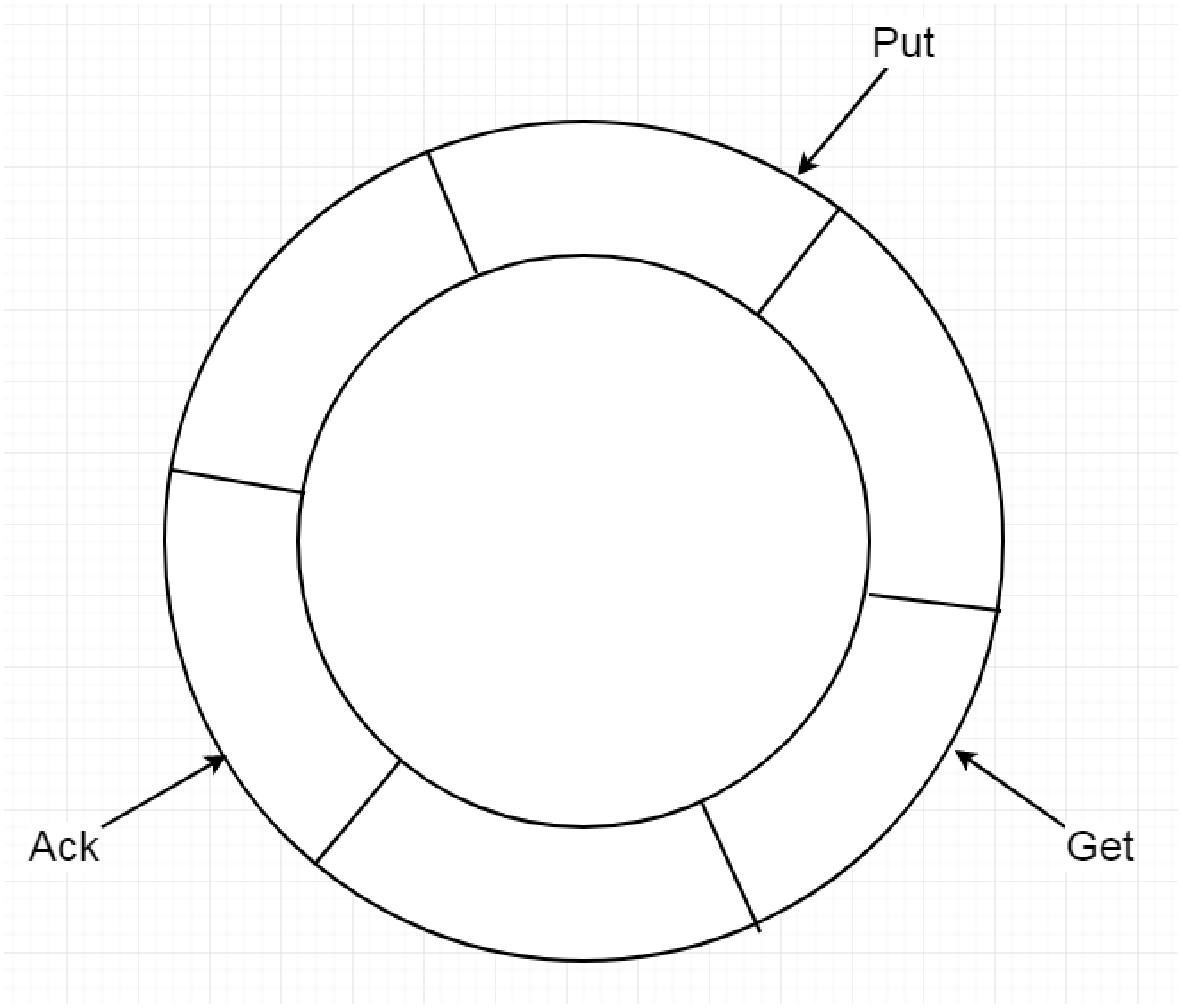


Figure 9: store

Store manages the data in the queue through three pointers:

1. Get pointer: the GET pointer points to the last location get by the Canal Client.
2. Ack pointer: the ACK pointer points to the location of the last successful consumption.
3. Put pointer: the PUT pointer points to the location where the sink module successfully wrote to the store at last.

```
canal client asynchronously get data in the store

      get 0       get 1      get 2                 put
        |           |          |        ......      |
        v           v          v                    v
---------------------------------------------------------------- store ring queue
        ^           ^
```

```
          |              |
        ack 0          ack 1
```

When the Canal Client calls the Get command, the Canal Server will generate data batches and send them to the Canal Client, and move the Get pointer to the right. The Canal Client can get multiple batches until the Get pointer catches up with the Put pointer.

When the consumption of data is successful, the Canal Client will return Ack + Batch ID, notify that the consumption has been successful, and move the Ack pointer to the right. The store will delete the data of this batch from the ring queue, make room to get data from the upstream sink module, and then move the Put pointer to the right.

When the data consumption fails, the client will return a rollback notification of the consumption failure, and the store will reset the Get pointer to the left to the Ack pointer' s position, so that the next data get by the Canal Client can start from the Ack pointer again.

Like the slave node in mysql, Canal Server also needs to save the latest consumption location of the client. All metadata in Canal Server (such as gtid and binlog location) is managed by the metamanager. At present, these metadata is persisted in the meta.dat file in the instance' s root directory in JSON format by default.

Basic Operation

Configure Target Table Properties

User needs to first create the target table which is corresponding to the MySQL side

Binlog Load can only support unique target tables from now, and the batch delete feature of the target table must be activated.

For the method of enabling Batch Delete, please refer to the batch delete function in ALTER TABLE PROPERTY.

Example:

```
--create Mysql table
CREATE TABLE `demo.source_test` (
  `id` int(11) NOT NULL COMMENT "",
  `name` int(11) NOT NULL COMMENT ""
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


-- create Doris table
CREATE TABLE `target_test` (
  `id` int(11) NOT NULL COMMENT "",
  `name` int(11) NOT NULL COMMENT ""
) ENGINE=OLAP
UNIQUE KEY(`id`)
COMMENT "OLAP"
DISTRIBUTED BY HASH(`id`) BUCKETS 8;

-- enable batch delete
ALTER TABLE target_test ENABLE FEATURE "BATCH_DELETE";
```

! ! Doris table structure and Mysql table structure field order must be consistent ! !

Create SyncJob

```
CREATE SYNC `demo`.`job`
(
FROM `demo`.`source_test1` INTO `target_test`
(id,name)
)
FROM BINLOG
(
"type" = "canal",
"canal.server.ip" = "127.0.0.1",
"canal.server.port" = "11111",
"canal.destination" = "xxx",
"canal.username" = "canal",
"canal.password" = "canal"
);
```

The detailed syntax for creating a data synchronization job can be connected to Doris and CREATE SYNC JOB to view the syntax help. Here is a detailed introduction to the precautions when creating a job.

grammar:

```
CREATE SYNC [db.]job_name
 (
        channel_desc,
        channel_desc
        ...
 )
binlog_desc
```

- job_name

  `job_Name` is the unique identifier of the SyncJob in the current database. With a specified job name, only one SyncJob can be running at the same time.

- channel_desc

  `column_Mapping` mainly refers to the mapping relationship between the columns of the MySQL source table and the Doris target table.

  If it is not specified, the columns of the source table and the target table will consider correspond one by one in order.

  However, we still recommend explicitly specifying the mapping relationship of columns, so that when the schema-change of the target table (such as adding a nullable column), data synchronization can still be carried out.

  Otherwise, when the schema-change occur, because the column mapping relationship is no longer one-to-one, the SyncJob will report an error.

- binlog_desc

  `binlog_desc` defines some necessary information for docking the remote binlog address.

  At present, the only supported docking type is the canal type. In canal type, all configuration items need to be prefixed with the canal prefix.

1. canal.server.ip: the address of the canal server
2. canal.server.port: the port of canal server
3. canal.destination: the identifier of the instance
4. canal.batchSize: the maximum batch size get from the canal server for each batch. Default 8192
5. canal.username: the username of instance
6. canal.password: the password of instance
7. canal.debug: when set to true, the details message of each batch and each row will be printed, which may affect the performance.

Show Job Status

Specific commands and examples for viewing job status can be viewed through the SHOW SYNC JOB command.

The parameters in the result set have the following meanings:

- State

  The current stage of the job. The transition between job states is shown in the following figure:

```
                       +-------------+
           create job  |  PENDING    |    resume job
           +-----------+             <-------------+
           |           +-------------+             |
      +----v-------+                       +-------+----+
      |  RUNNING   |     pause job         |  PAUSED    |
      |            +----------------------->            |
      +----+-------+     run error         +-------+----+
           |           +-------------+             |
           |           | CANCELLED   |             |
           +----------->             <-------------+
          stop job     +-------------+    stop job
          system error
```

```
After the SyncJob is submitted, the status is pending.

After the Fe scheduler starts the canal client, the status becomes running.

User can control the status of the job by three commands: `stop/pause/resume`. After the
    ↪ operation, the job status is `cancelled/paused/running` respectively.

There is only one final stage of the job, Cancelled. When the job status changes to Canceled, it
    ↪ cannot be resumed again.

When an error occurs during SyncJob is running, if the error is unrecoverable, the status will
    ↪ change to cancelled, otherwise it will change to paused.
```

- Channel

  The mapping relationship between all source tables and target tables of the job.

- Status

  The latest consumption location of the current binlog (if the gtid mode is on, the gtid will be displayed), and the delay time of the Doris side compared with the MySQL side.

- JobConfig

  The remote server information of the docking, such as the address of the Canal Server and the destination of the connected instance.

Control Operation

Users can control the status of jobs through `stop/pause/resume` commands.

You can use STOP SYNC JOB ; PAUSE SYNC JOB; And RESUME SYNC JOB; commands to view help and examples.

Related Parameters

Canal configuration

- `canal.ip`

  canal server's ip address

- `canal.port`

  canal server's port

- `canal.instance.memory.buffer.size`

  The queue length of the store ring queue, must be set to the power of 2, the default length is 16384. This value is equal to the maximum number of events that can be cached on the canal side and directly determines the maximum number of events that can be accommodated in a transaction on the Doris side. It is recommended to make it large enough to prevent the upper limit of the amount of data that can be accommodated in a transaction on the Doris side from being too small, resulting in too frequent transaction submission and data version accumulation.

- `canal.instance.memory.buffer.memunit`

  The default space occupied by an event at the canal end, default value is 1024 bytes. This value multiplied by `canal.`
  `↪ instance.memory.buffer.size` is equal to the maximum space of the store. For example, if the queue length of the store is 16384, the space of the store is 16MB. However, the actual size of an event is not actually equal to this value, but is determined by the number of rows of data in the event and the length of each row of data. For example, the insert event of a table with only two columns is only 30 bytes, but the delete event may reach thousands of bytes. This is because the number of rows of delete event is usually more than that of insert event.

Fe configuration

The following configuration belongs to the system level configuration of SyncJob. The configuration value can be modified in configuration file fe.conf.

- `sync_commit_interval_second`

  Maximum interval time between commit transactions. If there is still data in the channel that has not been committed after this time, the consumer will notify the channel to commit the transaction.

- `min_sync_commit_size`

  The minimum number of events required to commit a transaction. If the number of events received by Fe is less than it, Fe 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 ensure that this value is less than the `canal.instance` ↪ `.memory.buffer.size` configuration on the canal side (16384 by default). Otherwise, Fe will try to get more events than the length of the store queue without ack, causing the store queue to block until timeout.

- `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 ensure that this value is less than the product `canal.instance.memory.` ↪ `buffer.size` and `canal.instance.memory.buffer.memunit` on the canal side (16MB by default). Otherwise, Fe will try to get data from canal larger than the store space without ack, causing the store queue to block until timeout.

- `max_bytes_sync_commit`

  The maximum size of the data when the transaction is committed. If the data size received by Fe is larger than it, it will immediately commit the transaction and send the accumulated data. The default value is 64MB. If you want to modify this configuration, please ensure that this value is greater than the product of `canal.instance.memory.buffer.size` and `canal.instance.memory.buffer.mmemunit` on the canal side (16MB by default) and `min_bytes_sync_commit`。

- `max_sync_task_threads_num`

  The maximum number of threads in the SyncJobs' thread pool. There is only one thread pool in the whole Fe for synchronization, which is used to process the tasks created by all SyncJobs in the Fe.

FAQ

1. Will modifying the table structure affect data synchronization?

   Yes. The SyncJob cannot prohibit `alter table` operation. When the table's schema changes, if the column mapping cannot match, the job may be suspended incorrectly. It is recommended to reduce such problems by explicitly specifying the column mapping relationship in the data synchronization job, or by adding nullable columns or columns with default values.

2. Will the SyncJob continue to run after the database is deleted?

   No. In this case, the SyncJob will be checked by the Fe's scheduler thread and be stopped.

3. Can multiple SyncJobs be configured with the same `IP:Port + destination`?

   No. When creating a SyncJob, FE will check whether the `IP:Port + destination` is duplicate with the existing job to prevent multiple jobs from connecting to the same instance.

4. Why is the precision of floating-point type different between MySQL and Doris during data synchronization?

   The precision of Doris floating-point type is different from that of MySQL. You can choose to use decimal type instead.

More Help

For more detailed syntax and best practices used by Binlog Load, see Binlog Load command manual, you can also enter HELP
↪ BINLOG in the MySql client command line for more help information.

2.4.1.3.2 Broker Load

Broker load is an asynchronous import method, and the supported data sources depend on the data sources supported by the Broker process.

Because the data in the Doris table is ordered, Broker load uses the doris cluster resources to sort the data when importing data. Complete massive historical data migration relative to Spark load, the Doris cluster resource usage is relatively large. , this method is used when the user does not have Spark computing resources. If there are Spark computing resources, it is recommended to use Spark load.

Users need to create Broker load import through MySQL protocol and import by viewing command to check the import result.

Applicable scene

- The source data is in a storage system that the broker can access, such as HDFS.
- The amount of data is at the level of tens to hundreds of GB.

Fundamental

After the user submits the import task, FE will generate the corresponding Plan and distribute the Plan to multiple BEs for execution according to the current number of BEs and file size, and each BE executes a part of the imported data.

BE pulls data from the broker during execution, and imports the data into the system after transforming the data. All BEs are imported, and FE ultimately decides whether the import is successful.

```
                 +
                 | 1. user create broker load
                 v
            +----+----+
            |         |
            |   FE    |
            |         |
            +----+----+
                 |
                 | 2. BE etl and load the data
    +--------------------------+
    |            |             |
+---v---+     +--v----+    +---v---+
|       |     |       |    |       |
|  BE   |     |  BE   |    |   BE  |
|       |     |       |    |       |
+---+-^-+     +---+-^-+    +--+-^--+
    | |           | |         | |
    | |           | |         | | 3. pull data from broker
```

```
+---v-+-+     +---v-+-+    +--v-+--+
|       |     |       |    |       |
|Broker |     |Broker |    |Broker |
|       |     |       |    |       |
+---+-^-+     +---+-^-+    +---+-^-+
    | |           | |          | |
+---v-+-----------v-+----------v-+-+
|       HDFS/BOS/AFS cluster       |
|                                  |
+----------------------------------+
```

start import

Let’ s look at Broker Load through several actual scenario examples. use

Data import of Hive partition table

1. Create Hive table

```
######Data format is: default, partition field is: day
CREATE TABLE `ods_demo_detail`(
  `id` string,
  `store_id` string,
  `company_id` string,
  `tower_id` string,
  `commodity_id` string,
  `commodity_name` string,
  `commodity_price` double,
  `member_price` double,
  `cost_price` double,
  `unit` string,
  `quantity` double,
  `actual_price` double
)
PARTITIONED BY (day string)
row format delimited fields terminated by ','
lines terminated by '\n'
```

Then use Hive’ s Load command to import your data into the Hive table

```
load data local inpath '/opt/custorm' into table ods_demo_detail;
```

2. Create a Doris table, refer to the specific table syntax: CREATE TABLE

```
CREATE TABLE `doris_ods_test_detail` (
  `rq` date NULL,
  `id` varchar(32) NOT NULL,
  `store_id` varchar(32) NULL,
  `company_id` varchar(32) NULL,
  `tower_id` varchar(32) NULL,
  `commodity_id` varchar(32) NULL,
  `commodity_name` varchar(500) NULL,
  `commodity_price` decimal(10, 2) NULL,
  `member_price` decimal(10, 2) NULL,
  `cost_price` decimal(10, 2) NULL,
  `unit` varchar(50) NULL,
  `quantity` int(11) NULL,
  `actual_price` decimal(10, 2) NULL
) ENGINE=OLAP
UNIQUE KEY(`rq`, `id`, `store_id`)
PARTITION BY RANGE(`rq`)
(
PARTITION P_202204 VALUES [('2022-04-01'), ('2022-05-01')))
DISTRIBUTED BY HASH(`store_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3",
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "MONTH",
"dynamic_partition.start" = "-2147483648",
"dynamic_partition.end" = "2",
"dynamic_partition.prefix" = "P_",
"dynamic_partition.buckets" = "1",
"in_memory" = "false",
"storage_format" = "V2"
);
```

3. Start importing data

Specific syntax reference: Broker Load

```
LOAD LABEL broker_load_2022_03_23
(
    DATA INFILE("hdfs://192.168.20.123:8020/user/hive/warehouse/ods.db/ods_demo_detail/*/*")
    INTO TABLE doris_ods_test_detail
    COLUMNS TERMINATED BY ","
  (id,store_id,company_id,tower_id,commodity_id,commodity_name,commodity_price,member_price,cost_
      ↪ price,unit,quantity,actual_price)
    COLUMNS FROM PATH AS (`day`)
   SET
   (rq = str_to_date(`day`,'%Y-%m-%d'),id=id,store_id=store_id,company_id=company_id,tower_id=
       ↪ tower_id,commodity_id=commodity_id,commodity_name=commodity_name,commodity_price=
        ↪ commodity_price,member_price =member_price,cost_price=cost_price,unit=unit,quantity=
        ↪ quantity,actual_price=actual_price)
)
WITH BROKER "broker_name_1"
(
"username" = "hdfs",
"password" = ""
)
PROPERTIES
(
    "timeout"="1200",
    "max_filter_ratio"="0.1"
);
```

Hive partition table import (ORC format)

1. Create Hive partition table, ORC format

```
#####Data format: ORC partition: day
CREATE TABLE `ods_demo_orc_detail`(
  `id` string,
  `store_id` string,
  `company_id` string,
  `tower_id` string,
  `commodity_id` string,
  `commodity_name` string,
  `commodity_price` double,
  `member_price` double,
  `cost_price` double,
  `unit` string,
  `quantity` double,
  `actual_price` double
)
PARTITIONED BY (day string)
row format delimited fields terminated by ','
lines terminated by '\n'
STORED AS ORC
```

2. Create a Doris table. The table creation statement here is the same as the Doris table creation statement above. Please refer to the above .

3. Import data using Broker Load

```
   LOAD LABEL dish_2022_03_23
   (
       DATA INFILE("hdfs://10.220.147.151:8020/user/hive/warehouse/ods.db/ods_demo_orc_detail/*/*
           ↪ ")
       INTO TABLE doris_ods_test_detail
       COLUMNS TERMINATED BY ","
       FORMAT AS "orc"
   (id,store_id,company_id,tower_id,commodity_id,commodity_name,commodity_price,member_price,cost
        ↪ _price,unit,quantity,actual_price)
       COLUMNS FROM PATH AS (`day`)
      SET
      (rq = str_to_date(`day`,'%Y-%m-%d'),id=id,store_id=store_id,company_id=company_id,tower_id=
           ↪ tower_id,commodity_id=commodity_id,commodity_name=commodity_name,commodity_price=
           ↪ commodity_price,member_price =member_price,cost_price=cost_price,unit=unit,quantity
           ↪ =quantity,actual_price=actual_price)
   )
   WITH BROKER "broker_name_1"
   (
   "username" = "hdfs",
   "password" = ""
   )
   PROPERTIES
   (
       "timeout"="1200",
       "max_filter_ratio"="0.1"
   );
```

Notice:

- `FORMAT AS "orc"` : here we specify the data format to import
- SET : Here we define the field mapping relationship between the Hive table and the Doris table and some operations for field conversion

HDFS file system data import

Let’ s continue to take the Doris table created above as an example to demonstrate importing data from HDFS through Broker Load.

The statement to import the job is as follows:

```
LOAD LABEL demo.label_20220402
        (
            DATA INFILE("hdfs://10.220.147.151:8020/tmp/test_hdfs.txt")
            INTO TABLE `ods_dish_detail_test`
            COLUMNS TERMINATED BY "\t" (id,store_id,company_id,tower_id,commodity_id,commodity_
                ↪ name,commodity_price,member_price,cost_price,unit,quantity,actual_price)
        )
        with HDFS (
            "fs.defaultFS"="hdfs://10.220.147.151:8020",
            "hadoop.username"="root"
        )
        PROPERTIES
        (
            "timeout"="1200",
            "max_filter_ratio"="0.1"
        );
```

The specific parameters here can refer to: Broker and Broker Load documentation

View import status

We can view the status information of the above import task through the following command,

The specific syntax reference for viewing the import status SHOW LOAD

```
mysql> show load order by createtime desc limit 1\G;
*************************** 1. row ******************** ******
         JobId: 41326624
         Label: broker_load_2022_03_23
         State: FINISHED
      Progress: ETL: 100%; LOAD: 100%
          Type: BROKER
       EtlInfo: unselected.rows=0; dpp.abnorm.ALL=0; dpp.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)
```

Cancel import

When the broker load job status is not CANCELLED or FINISHED, it can be manually canceled by the user. When canceling, you need to specify the Label of the import task to be canceled. Cancel the import command syntax to execute CANCEL LOAD view.

For example: cancel the import job with the label broker_load_2022_03_23 on the database demo

```
CANCEL LOAD FROM demo WHERE LABEL = "broker_load_2022_03_23";
```

Relevant system configuration

Broker parameters

Broker Load needs to use the Broker process to access remote storage. Different brokers need to provide different parameters. For details, please refer to Broker documentation.

FE configuration

The following configurations belong to the system-level configuration of Broker load, that is, the configurations that apply to all Broker load import tasks. The configuration values are adjusted mainly by modifying `fe.conf`.

- min_bytes_per_broker_scanner/max_bytes_per_broker_scanner/max_broker_concurrency

The first two configurations limit the minimum and maximum amount of data processed by a single BE. The third configuration limits the maximum number of concurrent imports for a job. The minimum amount of data processed, the maximum number of concurrency, the size of the source file and the number of BEs in the current cluster ** together determine the number of concurrent imports**.

```
The number of concurrent imports this time = Math.min (source file size/minimum processing
    ↪ capacity, maximum concurrent number, current number of BE nodes)
The processing volume of a single BE imported this time = the size of the source file / the
    ↪ number of concurrent imports this time
```

Usually the maximum amount of data supported by an import job is `max_bytes_per_broker_scanner * number of BE ↪ nodes`. If you need to import a larger amount of data, you need to adjust the size of the `max_bytes_per_broker_scanner` parameter appropriately.

default allocation:

```
Parameter name: min_bytes_per_broker_scanner, the default is 64MB, the unit is bytes.
Parameter name: max_broker_concurrency, default 10.
Parameter name: max_bytes_per_broker_scanner, the default is 3G, the unit is bytes.
```

Best Practices

Application scenarios

The most suitable scenario for using Broker load is the scenario where the original data is in the file system (HDFS, BOS, AFS). Secondly, since Broker load is the only way of asynchronous import in a single import, if users need to use asynchronous access when importing large files, they can also consider using Broker load.

The amount of data

Only the case of a single BE is discussed here. If the user cluster has multiple BEs, the amount of data in the title below should be multiplied by the number of BEs. For example: if the user has 3 BEs, the value below 3G (inclusive) should be multiplied by 3, that is, below 9G (inclusive).

- Below 3G (included)

Users can directly submit Broker load to create import requests.

- Above 3G

Since the maximum processing capacity of a single import BE is 3G, the import of files exceeding 3G needs to be adjusted by adjusting the import parameters of Broker load to realize the import of large files.

1. Modify the maximum scan amount and maximum concurrent number of a single BE according to the current number of BEs and the size of the original file.

```
    Modify the configuration in fe.conf
    max_broker_concurrency = number of BEs
    The amount of data processed by a single BE of the current import task = original file size
        ↪ / max_broker_concurrency
    max_bytes_per_broker_scanner >= the amount of data processed by a single BE of the current
        ↪ import task

    For example, for a 100G file, the number of BEs in the cluster is 10
    max_broker_concurrency = 10
    max_bytes_per_broker_scanner >= 10G = 100G / 10
```

```
After modification, all BEs will process the import task concurrently, each BE processing part
    ↪ of the original file.

*Note: The configurations in the above two FEs are all system configurations, that is to say,
    ↪ their modifications are applied to all Broker load tasks. *
```

2. Customize the timeout time of the current import task when creating an import

```
    The amount of data processed by a single BE of the current import task / the slowest import
        ↪ speed of the user Doris cluster (MB/s) >= the timeout time of the current import
        ↪ task >= the amount of data processed by a single BE of the current import task / 10M
        ↪ /s

    For example, for a 100G file, the number of BEs in the cluster is 10
    timeout >= 1000s = 10G / 10M/s
```

3. When the user finds that the timeout time calculated in the second step exceeds the default import timeout time of 4 hours

   At this time, it is not recommended for users to directly increase the maximum import timeout to solve the problem. If the single import time exceeds the default import maximum timeout time of 4 hours, it is best to divide the files to be imported and import them in multiple times to solve the problem. The main reason is: if a single import exceeds 4 hours, the time cost of retrying after the import fails is very high.

   The expected maximum import file data volume of the Doris cluster can be calculated by the following formula:

```
    Expected maximum import file data volume = 14400s * 10M/s * number of BEs
    For example: the number of BEs in the cluster is 10
    Expected maximum import file data volume = 14400s * 10M/s * 10 = 1440000M ≈ 1440G
    Note: The average user's environment may not reach the speed of 10M/s, so it is recommended
        ↪ that files over 500G be divided and imported.
```

Job scheduling

The system limits the number of running Broker Load jobs in a cluster to prevent too many Load jobs from running at the same time.

First, the configuration parameter of FE: `desired_max_waiting_jobs` will limit the number of Broker Load jobs that have not started or are running (job status is PENDING or LOADING) in a cluster. Default is 100. If this threshold is exceeded, newly submitted jobs will be rejected outright.

A Broker Load job is divided into pending task and loading task phases. Among them, the pending task is responsible for obtaining the information of the imported file, and the loading task will be sent to the BE to execute the specific import task.

The FE configuration parameter `async_pending_load_task_pool_size` is used to limit the number of pending tasks running at the same time. It is also equivalent to controlling the number of import tasks that are actually running. This parameter defaults to 10. That is to say, assuming that the user submits 100 Load jobs, at the same time only 10 jobs will enter the LOADING state and start execution, while other jobs are in the PENDING waiting state.

The configuration parameter `async_loading_load_task_pool_size` of FE is used to limit the number of tasks of loading tasks running at the same time. A Broker Load job will have one pending task and multiple loading tasks (equal to the number of DATA INFILE clauses in the LOAD statement). So `async_loading_load_task_pool_size` should be greater than or equal to `async_`
↪ `pending_load_task_pool_size`.

Performance Analysis

Session variables can be enabled by executing `set enable_profile=true` before submitting the LOAD job. Then submit the import job. After the import job is completed, you can view the profile of the import job in the `Queris` tab of the FE web page.

You can check the SHOW LOAD PROFILE help document for more usage help information.

This Profile can help analyze the running status of import jobs.

Currently the Profile can only be viewed after the job has been successfully executed

common problem

- Import error: `Scan bytes per broker scanner exceed limit:xxx`

Please refer to the Best Practices section in the document to modify the FE configuration items `max_bytes_per_broker_scanner` and `max_broker_concurrency`

- Import error: `failed to send batch` or `TabletWriter add batch with unknown id`

Modify `query_timeout` and `streaming_load_rpc_max_alive_time_sec` appropriately.

streaming_load_rpc_max_alive_time_sec:

During the import process, Doris will open a Writer for each Tablet to receive data and write. This parameter specifies the Writer' s wait timeout. If the Writer does not receive any data within this time, the Writer will be automatically destroyed. When the system processing speed is slow, the Writer may not receive the next batch of data for a long time, resulting in an import error:

`TabletWriter add batch with unknown id`. At this time, this configuration can be appropriately increased. Default is 600 seconds

- Import error: `LOAD_RUN_FAIL; msg:Invalid Column Name:xxx`

If it is data in PARQUET or ORC format, the column name of the file header needs to be consistent with the column name in the doris table, such as:

```
(tmp_c1,tmp_c2)
SET
(
    id=tmp_c2,
    name=tmp_c1
)
```

Represents getting the column with (tmp_c1, tmp_c2) as the column name in parquet or orc, which is mapped to the (id, name) column in the doris table. If set is not set, the column in column is used as the map.

Note: If you use the orc file directly generated by some hive versions, the header in the orc file is not hive meta data, but (_col0, _col1, _col2, …), which may cause Invalid Column Name error, then you need to use set to map

more help

For more detailed syntax and best practices used by Broker Load, see Broker Load command manual, you can also enter HELP ↪ BROKER LOAD in the MySql client command line for more help information.

2.4.1.3.3 Routine Load

The Routine Load feature provides users with a way to automatically load data from a specified data source.

This document describes the implementation principles, usage, and best practices of this feature.

Glossary

- RoutineLoadJob: A routine load job submitted by the user.
- JobScheduler: A routine load job scheduler for scheduling and dividing a RoutineLoadJob into multiple Tasks.
- Task: RoutineLoadJob is divided by JobScheduler according to the rules.
- TaskScheduler: Task Scheduler. Used to schedule the execution of a Task.

Principle

```
         +---------+
         |  Client |
         +----+----+
              |
+-----------------------------+
| FE          |               |
| +-----------v------------+  |
| |                        |  |
| |   Routine Load Job     |  |
| |                        |  |
| +---+--------+--------+--+  |
|     |        |        |     |
| +---v--+ +---v--+ +---v--+  |
| | task | | task | | task |  |
| +--+---+ +---+--+ +---+--+  |
|    |         |        |     |
+-----------------------------+
     |         |        |
     v         v        v
 +---+--+   +--+---+   ++-----+
 |  BE  |   |  BE  |   |  BE  |
 +------+   +------+   +------+
```

As shown above, the client submits a routine load job to FE.

FE splits an load job into several Tasks via JobScheduler. Each Task is responsible for loading a specified portion of the data. The Task is assigned by the TaskScheduler to the specified BE.

On the BE, a Task is treated as a normal load task and loaded via the Stream Load load mechanism. After the load is complete, report to FE.

The JobScheduler in the FE continues to generate subsequent new Tasks based on the reported results, or retry the failed Task.

The entire routine load job completes the uninterrupted load of data by continuously generating new Tasks.

Kafka Routine load

Currently we only support routine load from the Kafka system. This section details Kafka’s routine use and best practices.

Usage restrictions

1. Support unauthenticated Kafka access and Kafka clusters certified by SSL.
2. The supported message format is csv text or json format. Each message is a line in csv format, and the end of the line does not contain a ** line break.
3. Kafka 0.10.0.0 (inclusive) or above is supported by default. If you want to use Kafka versions below 0.10.0.0 (0.9.0, 0.8.2, 0.8.1, 0.8.0), you need to modify the configuration of be, set the value of kafka_broker_version_fallback to be the older version, or directly set the value of property.broker.version.fallback to the old version when creating routine load. The cost of the old version is that some of the new features of routine load may not be available, such as setting the offset of the kafka partition by time.

Create a routine load task

The detailed syntax for creating a routine import task can be found in CREATE ROUTINE LOAD after connecting to Doris command manual, or execute `HELP ROUTINE LOAD;` for syntax help.

Here we illustrate how to create Routine Load tasks with a few examples.

1. Create a Kafka example 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 subscribe from the location where data is available (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 example import task named test1 for example_tbl of example_db in strict mode.

```
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" = "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"
        );
```

Notes：

“strict_mode” = “true”

3. Example of importing data in Json format

Routine Load only supports the following two types of json formats

The first one has only one record and is a json object.

```
{"category":"a9jadhx","author":"test","price":895}
```

The second one is a json array, which can contain multiple records

```
[
    {
        "category":"11",
        "author":"4avc",
        "price":895,
        "timestamp":1589191587
    },
    {
        "category":"22",
        "author":"2avc",
        "price":895,
        "timestamp":1589191487
    },
    {
        "category":"33",
        "author":"3avc",
        "price":342,
        "timestamp":1589191387
    }
]
```

Create the Doris data table to be imported

```
CREATE TABLE `example_tbl` (
   `category` varchar(24) NULL COMMENT "",
   `author` varchar(24) NULL COMMENT "",
   `timestamp` bigint(20) NULL COMMENT "",
   `dt` int(11) NULL COMMENT "",
   `price` double REPLACE
) ENGINE=OLAP
AGGREGATE KEY(`category`,`author`,`timestamp`,`dt`)
```

```
COMMENT "OLAP"
PARTITION BY RANGE(`dt`)
(
  PARTITION p0 VALUES [("-2147483648"), ("20200509")),
    PARTITION p20200509 VALUES [("20200509"), ("20200510")),
    PARTITION p20200510 VALUES [("20200510"), ("20200511")),
    PARTITION p20200511 VALUES [("20200511"), ("20200512"))
)
DISTRIBUTED BY HASH(`category`,`author`,`timestamp`) BUCKETS 4
PROPERTIES (
    "replication_num" = "1"
);
```

Import json data in simple mode

```
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"
 );
```

Accurate import of data in json format

```
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\"]",
    "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"
);
```

Notes:

The partition field `dt` in the table is not in our data, but is converted in our Routine load statement by `dt=from_unixtime(timestamp, '%Y%m%d')`

strict mode import relationship with source data

Here is an example with a column type of TinyInt

Notes: When a column in the table allows importing null values

source data	source data example	string to int	strict_mode	result
Null value	\N	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 is an example with the column type Decimal(1,0)

Notes:

When the columns in the table allow importing null values

source data	source data example	string to int	strict_mode	result
Null value	\N	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

Notes:

Although 10 is an out-of-range value, it is not affected by strict mode because its type meets the decimal requirement. 10 will eventually be filtered in other ETL processing processes. But it will not be filtered by strict mode.

Accessing an SSL-certified Kafka cluster

Accessing an SSL-certified Kafka cluster requires the user to provide the certificate file (ca.pem) used to authenticate the Kafka Broker’s public key. If the Kafka cluster also has client authentication enabled, the client’s public key (client.pem), the key file (client.key), and the key password are also required. The required files need to be uploaded to Doris first via the `CREAE FILE` command, and the catalog name is `kafka`. See `HELP CREATE FILE;` for help with the `CREATE FILE` command. Here are some examples.

1. uploading a file

```
CREATE FILE "ca.pem" PROPERTIES("url" = "https://example_url/kafka-key/ca.pem", "catalog" = "
    ↪ kafka");
CREATE FILE "client.key" PROPERTIES("url" = "https://example_urlkafka-key/client.key", "catalog"
    ↪ = "kafka");
CREATE FILE "client.pem" PROPERTIES("url" = "https://example_url/kafka-key/client.pem", "catalog"
    ↪  = "kafka");
```

2. Create routine import jobs

```
CREATE ROUTINE LOAD db1.job1 on tbl1
PROPERTIES
(
    "desired_concurrent_number"="1"
)
FROM KAFKA
(
    "kafka_broker_list"= "broker1:9091,broker2:9091",
    "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"
);
```

Doris accesses Kafka clusters through Kafka' s C++ API `librdkafka`. The parameters supported by `librdkafka` can be found in

https://github.com/edenhill/librdkafka/blob/master/CONFIGURATION.md

Accessing a Kerberos-certified Kafka cluster

Accessing a Kerberos-certified Kafka cluster. The following configurations need to be added:

- security.protocol=SASL_PLAINTEXT : Use SASL plaintext
- sasl.kerberos.service.name=$SERVICENAME : Broker service name
- sasl.kerberos.keytab=/etc/security/keytabs/${CLIENT_NAME}.keytab : Client keytab location
- sasl.kerberos.principal=$CLIENT_NAME/${CLIENT_HOST} : sasl.kerberos.principal

1. Create routine import jobs

```
CREATE ROUTINE LOAD db1.job1 on tbl1
PROPERTIES (
"desired_concurrent_number"="1",
 )
FROM KAFKA
(
    "kafka_broker_list" = "broker1:9092,broker2:9092",
    "kafka_topic" = "my_topic",
    "property.security.protocol" = "SASL_PLAINTEXT",
    "property.sasl.kerberos.service.name" = "kafka",
    "property.sasl.kerberos.keytab" = "/etc/krb5.keytab",
    "property.sasl.kerberos.principal" = "doris@YOUR.COM"
);
```

Note: - To enable Doris to access the Kafka cluster with Kerberos authentication enabled, you need to deploy the Kerberos client kinit on all running nodes of the Doris cluster, configure krb5.conf, and fill in KDC service information. - Configure property.sasl.kerberos The value of keytab needs to specify the absolute path of the keytab local file and allow Doris processes to access the local file.

Viewing Job Status

Specific commands and examples to view the status of jobs can be viewed with the `HELP SHOW ROUTINE LOAD;` command.

Specific commands and examples to view the status of tasks running can be viewed with the `HELP SHOW ROUTINE LOAD TASK;` command.

Only currently running tasks can be viewed; closed and unstarted tasks cannot be viewed.

Modify job properties

Users can modify jobs that have already been created. The details can be viewed with the `HELP ALTER ROUTINE LOAD;` command or see ALTER ROUTINE LOAD.

Job Control

The user can control the stop, pause and restart of jobs with the `STOP/PAUSE/RESUME` commands. Help and examples can be viewed with the `HELP STOP ROUTINE LOAD;` `HELP PAUSE ROUTINE LOAD;` and `HELP RESUME ROUTINE LOAD;` commands.

Other notes

1. The relationship between a routine import job and an ALTER TABLE operation

- Example import does not block SCHEMA CHANGE and ROLLUP operations. However, note that if the column mapping relationships do not match after the SCHEMA CHANGE completes, it can cause a spike in error data for the job and eventually cause the job to pause. It is recommended that you reduce this problem by explicitly specifying column mapping relationships in routine import jobs and by adding Nullable columns or columns with Default values.
- Deleting a Partition of a table may cause the imported data to fail to find the corresponding Partition and the job to enter a pause. 2.

2. Relationship between routine import jobs and other import jobs (LOAD, DELETE, INSERT)

- There is no conflict between the routine import and other LOAD operations and INSERT operations.
- When the DELETE operation is executed, the corresponding table partition cannot have any ongoing import jobs. Therefore, before executing DELETE operation, you may need to suspend the routine import job and wait until all the issued tasks are completed before executing DELETE. 3.

3. The relationship between routine import and DROP DATABASE/TABLE operations

When the database or table corresponding to the routine import is deleted, the job will automatically CANCEL.

4. The relationship between kafka type routine import jobs and kafka topic

When the `kafka_topic` declared by the user in the create routine import statement does not exist in the kafka cluster.

- If the broker of the user’s kafka cluster has `auto.create.topics.enable = true` set, then `kafka_topic` will be created automatically first, and the number of partitions created automatically is determined by the configuration of the broker in the user’s kafka cluster with `num. partitions`. The routine job will keep reading data from the topic as normal.
- If the broker in the user’s kafka cluster has `auto.create.topics.enable = false` set, the topic will not be created automatically and the routine job will be suspended with a status of PAUSED before any data is read.

So, if you want the kafka topic to be automatically created by the routine when it does not exist, just set `auto.create.topics.` ↪ `enable = true` for the broker in the user’s kafka cluster.

5. Problems that may arise in network isolated environments In some environments there are isolation measures for network segments and domain name resolution, so care needs to be taken

6. the Broker list specified in the Create Routine load task must be accessible by the Doris service

7. If `advertised.listeners` is configured in Kafka, the addresses in `advertised.listeners` must be accessible to the Doris service

8. Specify the Partition and Offset for consumption

Doris supports specifying a Partition and Offset to start consumption. The new version also supports the ability to specify time points for consumption. The configuration of the corresponding parameters is explained here.

There are three relevant parameters.

- `kafka_partitions`: Specify the list of partitions to be consumed, e.g., "0, 1, 2, 3".
- `kafka_offsets`: specifies 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_offset`: specifies the default starting offset of the partitions.

When creating an import job, these three parameters can have the following combinations.

combinations	`kafka_` ↪ `partitions` ↪	`kafka_` ↪ `offsets` ↪	`property.kafka_` ↪ `default_offset`	behavior
1	No	No	No	The system will automatically find all partitions corresponding to the topic and start consuming them from OFFSET_END
2	No	No	Yes	The system will automatically find all the partitions corresponding to the topic and start consuming them from the default offset location.
3	Yes	No	No	The system will start consuming from the OFFSET_END of the specified partition.
4	Yes	Yes	No	The system will start consuming at the specified offset of the specified partition.
5	Yes	No	Yes	The system will start consuming at the default offset of the specified partition

7. The difference between STOP and PAUSE

FE will automatically clean up the ROUTINE LOAD in STOP status periodically, while the PAUSE status can be restored to enable again.

Related Parameters

Some system configuration parameters can affect the use of routine import.

1. max_routine_load_task_concurrent_num

FE configuration item, defaults to 5 and can be modified at runtime. This parameter limits the maximum number of concurrent subtasks for a routine import job. It is recommended to keep the default value. Setting it too large may result in too many concurrent tasks and consume cluster resources.

2. max_routine_load_task_num_per_be

FE configuration item, default is 5, can be modified at runtime. This parameter limits the maximum number of concurrently executed subtasks per BE node. It is recommended to keep the default value. If set too large, it may lead to too many concurrent tasks and consume cluster resources.

3. max_routine_load_job_num

FE configuration item, default is 100, can be modified at runtime. This parameter limits the total number of routine import jobs, including the states NEED_SCHEDULED, RUNNING, PAUSE. After this, no new jobs can be submitted.

4. max_consumer_num_per_group

BE configuration item, default is 3. This parameter indicates the maximum number of consumers that can be generated for data consumption in a subtask. For a Kafka data source, a consumer may consume one or more kafka partitions. If there are only 2 partitions, only 2 consumers are generated, each consuming 1 partition. 5. push_write_mby

5. max_tolerable_backend_down_num

FE configuration item, the default value is 0. Doris can PAUSED job rescheduling to RUNNING if certain conditions are met. 0 means rescheduling is allowed only if all BE nodes are ALIVE.

6. period_of_auto_resume_min

FE configuration item, the default is 5 minutes, Doris rescheduling will only be attempted up to 3 times within the 5 minute period. If all 3 attempts fail, the current task is locked and no further scheduling is performed. However, manual recovery can be done through human intervention.

More help

For more detailed syntax on the use of Routine Load, you can type `HELP ROUTINE LOAD` at the Mysql client command line for more help.

2.4.1.3.4 Spark Load

Spark load realizes the preprocessing of load data by spark, improves the performance of loading large amount of Doris data and saves the computing resources of Doris cluster. It is mainly used for the scene of initial migration and large amount of data imported into Doris.

Spark load uses the resources of the spark cluster to sort the data to be imported, and Doris be writes files directly, which can greatly reduce the resource usage of the Doris cluster, and is very good for historical mass data migration to reduce the resource usage and load of the Doris cluster. Effect.

If users do not have the resources of Spark cluster and want to complete the migration of external storage historical data conveniently and quickly, they can use Broker load . Compared with Spark load, importing Broker load will consume more resources on the Doris cluster.

Spark load is an asynchronous load method. Users need to create spark type load job by MySQL protocol and view the load results by `show load`.

Applicable scenarios

- The source data is in a file storage system that spark can access, such as HDFS.
- The data volume ranges from tens of GB to TB.

Explanation of terms

1. Spark ETL: in the load process, it is mainly responsible for ETL of data, including global dictionary construction (bitmap type), partition, sorting, aggregation, etc.
2. Broker: broker is an independent stateless process. It encapsulates the file system interface and provides the ability of Doris to read the files in the remote storage system.
3. Global dictionary: it stores the data structure from the original value to the coded value. The original value can be any data type, while the encoded value is an integer. The global dictionary is mainly used in the scene of precise de duplication precomputation.

Basic principles

Basic process

The user submits spark type load job by MySQL client, Fe records metadata and returns that the user submitted successfully.

The implementation of spark load task is mainly divided into the following five stages.

1. Fe schedules and submits ETL tasks to spark cluster for execution.
2. Spark cluster executes ETL to complete the preprocessing of load data. It includes global dictionary building (bitmap type), partitioning, sorting, aggregation, etc.
3. After the ETL task is completed, Fe obtains the data path of each partition that has been preprocessed, and schedules the related be to execute the push task.
4. Be reads data through broker and converts it into Doris underlying storage format.
5. Fe schedule the effective version and complete the load job.

```
                 +
                 | 0. User create spark load job
            +----v----+
            |   FE    |---------------------------------+
            +----+----+                                 |
                 | 3. FE send push tasks                |
                 | 5. FE publish version                |
    +------------+------------+                         |
    |            |            |                         |
+---v---+    +---v---+    +---v---+                     |
|  BE   |    |  BE   |    |  BE   |                     |1. FE submit Spark ETL job
+---^---+    +---^---+    +---^---+                     |
    |4. BE push with broker   |                         |
+---+---+    +---+---+    +---+---+                     |
|Broker |    |Broker |    |Broker |                     |
+---^---+    +---^---+    +---^---+                     |
    |            |            |                         |
+---+------------+------------+---+ 2.ETL +-------------v---------------+
|               HDFS              +------->       Spark cluster         |
|                                 <-------+                             |
+---------------------------------+       +-----------------------------+
```

Global dictionary

Applicable scenarios

At present, the bitmap column in Doris is implemented using the class library ‘`roaingbitmap`’, while the input data type of ‘`roaringbitmap`’ can only be integer. Therefore, if you want to pre calculate the bitmap column in the import process, you need to convert the type of input data to integer.

In the existing Doris import process, the data structure of global dictionary is implemented based on hive table, which stores the mapping from original value to encoded value.

Build process

1. Read the data from the upstream data source and generate a hive temporary table, which is recorded as `hive_table`.
2. Extract the de duplicated values of the fields to be de duplicated from the `hive_table`, and generate a new hive table, which is marked as `distinct_value_table`.
3. Create a new global dictionary table named `dict_table`; one column is the original value, and the other is the encoded value.
4. Left join the `distinct_value_table` and `dict_table`, calculate the new de duplication value set, and then code this set with window function. At this time, the original value of the de duplication column will have one more column of encoded value. Finally, the data of these two columns will be written back to `dict_table`.
5. Join the `dict_table` with the `hive_table` to replace the original value in the `hive_table` with the integer encoded value.

6. `hive_table` will be read by the next data preprocessing process and imported into Doris after calculation.

Data preprocessing (DPP)

Basic process

1. Read data from the data source. The upstream data source can be HDFS file or hive table.
2. Map the read data, calculate the expression, and generate the bucket field `bucket_id` according to the partition information.
3. Generate rolluptree according to rollup metadata of Doris table.
4. Traverse rolluptree to perform hierarchical aggregation. The rollup of the next level can be calculated from the rollup of the previous level.
5. After each aggregation calculation, the data will be calculated according to the `bucket_id`is divided into buckets and then written into HDFS.
6. Subsequent brokers will pull the files in HDFS and import them into Doris be.

Hive Bitmap UDF

Spark supports loading hive-generated bitmap data directly into Doris, see hive-bitmap-udf documentation

Basic operation

Configure ETL cluster

As an external computing resource, spark is used to complete ETL work in Doris. In the future, there may be other external resources that will be used in Doris, such as spark / GPU for query, HDFS / S3 for external storage, MapReduce for ETL, etc. Therefore, we introduce resource management to manage these external resources used by Doris.

Before submitting the spark import task, you need to configure the spark cluster that performs the ETL task.

Grammar:

```
-- create spark resource
CREATE EXTERNAL RESOURCE resource_name
PROPERTIES
(
  type = spark,
  spark_conf_key = spark_conf_value,
  working_dir = path,
  broker = broker_name,
  broker.property_key = property_value,
  broker.hadoop.security.authentication = kerberos,
  broker.kerberos_principal = doris@YOUR.COM,
  broker.kerberos_keytab = /home/doris/my.keytab
  broker.kerberos_keytab_content = ASDOWHDLAWIDJHWLDKSALDJSDIWALD
)

-- drop spark resource
```

```
DROP RESOURCE resource_name

-- show resources
SHOW RESOURCES
SHOW PROC "/resources"

-- privileges
GRANT USAGE_PRIV ON RESOURCE resource_name TO user_identity
GRANT USAGE_PRIV ON RESOURCE resource_name TO ROLE role_name

REVOKE USAGE_PRIV ON RESOURCE resource_name FROM user_identity
REVOKE USAGE_PRIV ON RESOURCE resource_name FROM ROLE role_name
```

Create resource

`resource_name` is the name of the spark resource configured in Doris.

`Properties` are the parameters related to spark resources, as follows:

- `type`: resource type, required. Currently, only spark is supported.
- Spark related parameters are as follows:
- `spark.master`: required, yarn is supported at present, `spark://host:port`.
- `spark.submit.deployMode`: the deployment mode of Spark Program. It is required and supports cluster and client.
- `spark.hadoop.fs.defaultfs`: required when master is yarn.
- Other parameters are optional, refer to `http://spark.apache.org/docs/latest/configuration.html`
- YARN RM related parameters are as follows:
  - If Spark is a single-point RM, you need to configure `spark.hadoop.yarn.resourcemanager.address`, address of the single point resource manager.
  - If Spark is RM-HA, it needs to be configured (where hostname and address are optional):
    * `spark.hadoop.yarn.resourcemanager.ha.enabled`: ResourceManager enables HA, set to true.
    * `spark.hadoop.yarn.resourcemanager.ha.rm-ids`: List of ResourceManager logical IDs.
    * `spark.hadoop.yarn.resourcemanager.hostname.rm-id`: For each rm-id, specify the hostname of the ResourceManager.
    * `spark.hadoop.yarn.resourcemanager.address.rm-id`: For each rm-id, specify the host:port for clients to submit jobs.
- HDFS HA related parameters are as follows:
  - `spark.hadoop.fs.defaultFS`, hdfs client default path prefix.
  - `spark.hadoop.dfs.nameservices`, hdfs cluster logical name.
  - `spark.hadoop.dfs.ha.namenodes.nameservices01` , unique identifier for each NameNode in the nameservice.
  - `spark.hadoop.dfs.namenode.rpc-address.nameservices01.mynamenode1`, fully qualified RPC address for each NameNode.
  - `spark.hadoop.dfs.namenode.rpc-address.nameservices01.mynamenode2`, fully qualified RPC address for each NameNode.
  - `spark.hadoop.dfs.client.failover.proxy.provider = org.apache.hadoop.hdfs.server.namenode.`
    `↪ ha.ConfiguredFailoverProxyProvider`, set the implementation class. -`working_dir`: directory used by ETL. Spark is required when used as an ETL resource. For example: `hdfs://host :port/tmp/doris`.

- `broker.hadoop.security.authentication`: Specify the authentication method as kerberos.
- `broker.kerberos_principal`: Specify the principal of kerberos.
- `broker.kerberos_keytab`: Specify the path to the keytab file for kerberos. The file must be an absolute path to a file on the server where the broker process is located. And can be accessed by the Broker process.
- `broker.kerberos_keytab_content`: Specify the content of the keytab file in kerberos after base64 encoding. You can choose one of these with `broker.kerberos_keytab` configuration.
- `broker`: the name of the broker. Spark is required when used as an ETL resource. You need to use the ‘alter system add broker’ command to complete the configuration in advance.
- `broker.property_key`: the authentication information that the broker needs to specify when reading the intermediate file generated by ETL.
- env: Specify the spark environment variable and support dynamic setting. For example, when the authentication mode of Hadoop is simple, set the Hadoop user name and password
- env.HADOOP_USER_NAME: user name
- env.HADOOP_USER_PASSWORD: user password

Example:

```
-- 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 standalone client mode
CREATE EXTERNAL RESOURCE "spark1"
PROPERTIES
(
  "type" = "spark",
  "spark.master" = "spark://127.0.0.1:7777",
  "spark.submit.deployMode" = "client",
  "working_dir" = "hdfs://127.0.0.1:10000/tmp/doris",
  "broker" = "broker1"
);
```

```
-- yarn HA mode
CREATE EXTERNAL RESOURCE sparkHA
PROPERTIES
(
  "type" = "spark",
  "spark.master" = "yarn",
  "spark.submit.deployMode" = "cluster",
  "spark.executor.memory" = "1g",
  "spark.yarn.queue" = "default",
  "spark.hadoop.yarn.resourcemanager.ha.enabled" = "true",
  "spark.hadoop.yarn.resourcemanager.ha.rm-ids" = "rm1,rm2",
  "spark.hadoop.yarn.resourcemanager.address.rm1" = "xxxx:8032",
  "spark.hadoop.yarn.resourcemanager.address.rm2" = "xxxx:8032",
  "spark.hadoop.fs.defaultFS" = "hdfs://nameservices01",
  "spark.hadoop.dfs.nameservices" = "nameservices01",
  "spark.hadoop.dfs.ha.namenodes.nameservices01" = "mynamenode1,mynamenode2",
  "spark.hadoop.dfs.namenode.rpc-address.nameservices01.mynamenode1" = "xxxx:8020",
  "spark.hadoop.dfs.namenode.rpc-address.nameservices01.mynamenode2" = "xxxx:8020",
  "spark.hadoop.dfs.client.failover.proxy.provider" = "org.apache.hadoop.hdfs.server.namenode.ha.
      ↪ ConfiguredFailoverProxyProvider",
  "working_dir" = "hdfs://nameservices01/doris_prd_data/sinan/spark_load/",
  "broker" = "broker_name",
  "broker.username" = "username",
  "broker.password" = "",
  "broker.dfs.nameservices" = "nameservices01",
  "broker.dfs.ha.namenodes.HDFS4001273" = "mynamenode1, mynamenode2",
  "broker.dfs.namenode.rpc-address.nameservices01.mynamenode1" = "xxxx:8020",
  "broker.dfs.namenode.rpc-address.nameservices01.mynamenode2" = "xxxx:8020",
  "broker.dfs.client.failover.proxy.provider.nameservices01" = "org.apache.hadoop.hdfs.server.
      ↪ namenode.ha.ConfiguredFailoverProxyProvider"
);
```

Spark Load supports Kerberos authentication

If Spark load accesses Hadoop cluster resources with Kerberos authentication, we only need to specify the following parameters when creating Spark resources:

- `broker.hadoop.security.authentication`: Specify the authentication method as kerberos.
- `broker.kerberos_principal`: Specify the principal of kerberos.
- `broker.kerberos_keytab`: Specify the path to the keytab file for kerberos. The file must be an absolute path to a file on the server where the broker process is located. And can be accessed by the Broker process.
- `broker.kerberos_keytab_content`: Specify the content of the keytab file in kerberos after base64 encoding. You can choose one of these with `kerberos_keytab` configuration.

Example:

```
CREATE EXTERNAL RESOURCE "spark_on_kerberos"
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.hadoop.security.authentication" = "kerberos",
  "broker.kerberos_principal" = "doris@YOUR.COM",
  "broker.kerberos_keytab" = "/home/doris/my.keytab"
);
```

Show resources

Ordinary accounts can only see the resources that they have USAGE_PRIV to use.

The root and admin accounts can see all the resources.

Resource privilege

Resource permissions are managed by grant revoke. Currently, only USAGE_PRIV permission is supported.

You can use the USAGE_PRIV permission is given to a user or a role, and the role is used the same as before.

```
-- Grant permission to the spark0 resource to user user0

GRANT USAGE_PRIV ON RESOURCE "spark0" TO "user0"@"%";

-- Grant permission to the spark0 resource to role ROLE0

GRANT USAGE_PRIV ON RESOURCE "spark0" TO ROLE "role0";

-- Grant permission to all resources to user user0

GRANT USAGE_PRIV ON RESOURCE * TO "user0"@"%";

-- Grant permission to all resources to role ROLE0

GRANT USAGE_PRIV ON RESOURCE * TO ROLE "role0";

-- Revoke the spark0 resource permission of user user0
```

```
REVOKE USAGE_PRIV ON RESOURCE "spark0" FROM "user0"@"%";
```

Configure spark client

The Fe submits the spark task by executing the spark submit command. Therefore, it is necessary to configure the spark client for Fe. It is recommended to use the official version of spark 2 above 2.4.3, download spark here. After downloading, please follow the steps to complete the following configuration.

Configure SPARK_HOME environment variable

Place the spark client on the same machine as Fe and configure `spark_home_default_dir` in the `fe.conf`. This configuration item defaults to the `fe/lib/spark2x` path. This config cannot be empty.

Configure spark dependencies

Package all jar packages in jars folder under spark client root path into a zip file, and configure `spark_resource_patj` in `fe.conf` as this zip file's path.

When the spark load task is submitted, this zip file will be uploaded to the remote repository, and the default repository path will be hung in `working_dir/{cluster_ID}` directory named as `__spark_repository__{resource_name}`, which indicates that a resource in the cluster corresponds to a remote warehouse. The directory structure of the remote warehouse is as follows:

```
__spark_repository__spark0/
    |-__archive_1.0.0/
    |        |-__lib_990325d2c0d1d5e45bf675e54e44fb16_spark-dpp-1.0.0-jar-with-dependencies.jar
    |        |-__lib_7670c29daf535efe3c9b923f778f61fc_spark-2x.zip
    |-__archive_1.1.0/
    |        |-__lib_64d5696f99c379af2bee28c1c84271d5_spark-dpp-1.1.0-jar-with-dependencies.jar
    |        |-__lib_1bbb74bb6b264a270bc7fca3e964160f_spark-2x.zip
    |-__archive_1.2.0/
    |        |-...
```

In addition to spark dependency (named by `spark-2x.zip` by default), Fe will also upload DPP's dependency package to the remote repository. If all the dependency files submitted by spark load already exist in the remote repository, then there is no need to upload dependency, saving the time of repeatedly uploading a large number of files each time.

Configure yarn client

The Fe obtains the running application status and kills the application by executing the yarn command. Therefore, you need to configure the yarn client for Fe. It is recommended to use the official version of Hadoop above 2.5.2, download hadoop. After downloading, please follow the steps to complete the following configuration.

Configure the yarn client path

Place the downloaded yarn client in the same machine as Fe, and configure `yarn_client_path` in the `fe.conf` as the executable file of yarn, which is set as the `fe/lib/yarn-client/hadoop/bin/yarn` by default.

(optional) when Fe obtains the application status or kills the application through the yarn client, the configuration files required for executing the yarn command will be generated by default in the `lib/yarn-config` path in the Fe root directory. This path can be configured by configuring `yarn-config-dir` in the `fe.conf`. The currently generated configuration yarn config files include `core-site.xml` and `yarn-site.xml`.

Create Load

Grammar:

```
LOAD LABEL load_label
    (data_desc, ...)
    WITH RESOURCE resource_name resource_properties
    [PROPERTIES (key1=value1, ... )]

* load_label:
    db_name.label_name

* data_desc:
    DATA INFILE ('file_path', ...)
    [NEGATIVE]
    INTO TABLE tbl_name
    [PARTITION (p1, p2)]
    [COLUMNS TERMINATED BY separator ]
    [(col1, ...)]
    [SET (k1=f1(xx), k2=f2(xx))]
    [WHERE predicate]

* resource_properties:
    (key2=value2, ...)
```

Example 1: when the upstream data source is HDFS file

```
LOAD LABEL db1.label1
(
    DATA INFILE("hdfs://abc.com:8888/user/palo/test/ml/file1")
    INTO TABLE tbl1
    COLUMNS TERMINATED BY ","
    (tmp_c1,tmp_c2)
    SET
    (
        id=tmp_c2,
        name=tmp_c1
    ),
    DATA INFILE("hdfs://abc.com:8888/user/palo/test/ml/file2")
    INTO TABLE tbl2
    COLUMNS TERMINATED BY ","
    (col1, col2)
    where col1 > 1
)
WITH RESOURCE 'spark0'
(
    "spark.executor.memory" = "2g",
    "spark.shuffle.compress" = "true"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

Example 2: when the upstream data source is hive table

```
step 1:新建hive外部表
CREATE EXTERNAL TABLE hive_t1
(
    k1 INT,
    K2 SMALLINT,
    k3 varchar(50),
    uuid varchar(100)
)
ENGINE=hive
properties
(
"database" = "tmp",
"table" = "t1",
"hive.metastore.uris" = "thrift://0.0.0.0:8080"
);

step 2: 提交load命令
LOAD LABEL db1.label1
(
    DATA FROM TABLE hive_t1
    INTO TABLE tbl1
    (k1,k2,k3)
    SET
    (
        uuid=bitmap_dict(uuid)
    )
)
WITH RESOURCE 'spark0'
(
    "spark.executor.memory" = "2g",
    "spark.shuffle.compress" = "true"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

Example 3: when the upstream data source is hive binary type table

```
step 1: create hive external table
CREATE EXTERNAL TABLE hive_t1
(
    k1 INT,
    K2 SMALLINT,
    k3 varchar(50),
    uuid varchar(100)
)
ENGINE=hive
properties
(
"database" = "tmp",
"table" = "t1",
"hive.metastore.uris" = "thrift://0.0.0.0:8080"
);

step 2: submit load command
LOAD LABEL db1.label1
(
    DATA FROM TABLE hive_t1
    INTO TABLE tbl1
    (k1,k2,k3)
    SET
    (
        uuid=binary_bitmap(uuid)
    )
)
WITH RESOURCE 'spark0'
(
    "spark.executor.memory" = "2g",
    "spark.shuffle.compress" = "true"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

Example 4: Import data from hive partitioned table

```
-- hive create table statement
create table test_partition(
id int,
name string,
age int
```

```
)
partitioned by (dt string)
row format delimited fields terminated by ','
stored as textfile;

-- doris create table statement
CREATE TABLE IF NOT EXISTS test_partition_04
(
dt date,
id int,
name string,
age int
)
UNIQUE KEY(`dt`, `id`)
DISTRIBUTED BY HASH(`id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 1"
);
-- spark load
CREATE EXTERNAL RESOURCE "spark_resource"
PROPERTIES
(
"type" = "spark",
"spark.master" = "yarn",
"spark.submit.deployMode" = "cluster",
"spark.executor.memory" = "1g",
"spark.yarn.queue" = "default",
"spark.hadoop.yarn.resourcemanager.address" = "localhost:50056",
"spark.hadoop.fs.defaultFS" = "hdfs://localhost:9000",
"working_dir" = "hdfs://localhost:9000/tmp/doris",
"broker" = "broker_01"
);
LOAD LABEL demo.test_hive_partition_table_18
(
    DATA INFILE("hdfs://localhost:9000/user/hive/warehouse/demo.db/test/dt=2022-08-01/*")
    INTO TABLE test_partition_04
    COLUMNS TERMINATED BY ","
    FORMAT AS "csv"
    (id,name,age)
    COLUMNS FROM PATH AS (`dt`)
    SET
    (
        dt=dt,
        id=id,
        name=name,
        age=age
    )
)
WITH RESOURCE 'spark_resource'
(
    "spark.executor.memory" = "1g",
    "spark.shuffle.compress" = "true"
)
PROPERTIES
(
    "timeout" = "3600"
);
```

You can view the details syntax about creating load by input `help spark load`. This paper mainly introduces the parameter meaning and precautions in the creation and load syntax of spark load.

Label

Identification of the import task. Each import task has a unique label within a single database. The specific rules are consistent with `broker load`.

Data description parameters

Currently, the supported data sources are CSV and hive table. Other rules are consistent with `broker load`.

Load job parameters

Load job parameters mainly refer to the `opt_properties` in the spark load. Load job parameters are applied to the entire load job. The rules are consistent with `broker load`.

Spark resource parameters

Spark resources need to be configured into the Doris system in advance, and users should be given USAGE_PRIV. Spark load can only be used after priv permission.

When users have temporary requirements, such as adding resources for tasks and modifying spark configs, you can set them here. The settings only take effect for this task and do not affect the existing configuration in the Doris cluster.

```
WITH RESOURCE 'spark0'
(
  "spark.driver.memory" = "1g",
  "spark.executor.memory" = "3g"
)
```

Load when data source is hive table

At present, if you want to use hive table as a data source in the import process, you need to create an external table of type hive,

Then you can specify the table name of the external table when submitting the Load command.

Load process to build global dictionary

The data type applicable to the aggregate columns of the Doris table is of type bitmap.

In the load command, you can specify the field to build a global dictionary. The format is: ' `doris field name=bitmap_dict` ↪ `(hive_table field name)`

It should be noted that the construction of global dictionary is supported only when the upstream data source is hive table.

Load when data source is hive binary type table

The data type applicable to the aggregate column of the doris table is bitmap type, and the data type of the corresponding column in the hive table of the data source is binary (through the org.apache.doris.load.loadv2.dpp.BitmapValue (FE spark-dpp) class serialized) type.

There is no need to build a global dictionary, just specify the corresponding field in the load command, the format is: `doris field` ↪ `name=binary_bitmap (hive table field name)`

Similarly, the binary (bitmap) type of data import is currently only supported when the upstream data source is a hive table,You can refer to the use of hive bitmap hive-bitmap-udf

Show Load

Spark load is asynchronous just like broker load, so the user must create the load label record and use label in the show load command to view the load result. The show load command is common in all load types. The specific syntax can be viewed by executing help show load.

Example:

```
mysql> show load order by createtime desc limit 1\G
*************************** 1. row ***************************
         JobId: 76391
         Label: label1
         State: FINISHED
      Progress: ETL:100%; LOAD:100%
          Type: SPARK
       EtlInfo: unselected.rows=4; dpp.abnorm.ALL=15; dpp.norm.ALL=28133376
      TaskInfo: cluster:cluster0; timeout(s):10800; max_filter_ratio:5.0E-5
      ErrorMsg: N/A
    CreateTime: 2019-07-27 11:46:42
  EtlStartTime: 2019-07-27 11:46:44
 EtlFinishTime: 2019-07-27 11:49:44
 LoadStartTime: 2019-07-27 11:49:44
LoadFinishTime: 2019-07-27 11:50:16
           URL: http://1.1.1.1:8089/proxy/application_1586619723848_0035/
    JobDetails: {"ScannedRows":28133395,"TaskNumber":1,"FileNumber":1,"FileSize":200000}
```

Refer to broker load for the meaning of parameters in the returned result set. The differences are as follows:

- State

The current phase of the load job. After the job is submitted, the status is pending. After the spark ETL is submitted, the status changes to ETL. After ETL is completed, Fe schedules be to execute push operation, and the status changes to finished after the push is completed and the version takes effect.

There are two final stages of the load job: cancelled and finished. When the load job is in these two stages, the load is completed. Among them, cancelled is load failure, finished is load success.

- Progress

Progress description of the load job. There are two kinds of progress: ETL and load, corresponding to the two stages of the load process, ETL and loading.

The progress range of load is 0 ~ 100%.

```
Load progress = the number of tables that have completed all replica imports / the total number
↪ of tables in this import task * 100%
```

If all load tables are loaded, the progress of load is 99%, the load enters the final effective stage. After the whole load is completed, the load progress will be changed to 100%.

The load progress is not linear. Therefore, if the progress does not change over a period of time, it does not mean that the load is not in execution.

- Type

Type of load job. Spark load is spark.

- CreateTime/EtlStartTime/EtlFinishTime/LoadStartTime/LoadFinishTime

These values represent the creation time of the load, the start time of the ETL phase, the completion time of the ETL phase, the start time of the loading phase, and the completion time of the entire load job.

- JobDetails

Display the detailed running status of some jobs, which will be updated when ETL ends. It includes the number of loaded files, the total size (bytes), the number of subtasks, the number of processed original lines, etc.

```
{"ScannedRows":139264,"TaskNumber":1,"FileNumber":1,"FileSize":940754064}
```

- URL

Copy this url to the browser and jump to the web interface of the corresponding application.

View spark launcher commit log

Sometimes users need to view the detailed logs generated during the spark submission process. The logs are saved in the `log/`
↪ `spark_launcher_log` under the Fe root directory named as `spark_launcher_{load_job_id}_{label}.log`. The log will be saved in this directory for a period of time. When the load information in Fe metadata is cleaned up, the corresponding log will also be cleaned. The default saving log time is 3 days.

cancel load

When the spark load job status is not cancelled or finished, it can be manually cancelled by the user. When canceling, you need to specify the label to cancel the load job. The syntax of the cancel load command can be viewed by executing `help cancel load`.

Related system configuration

FE configuration

The following configuration belongs to the system level configuration of spark load, that is, the configuration for all spark load import tasks. Mainly through modification`fe.conf` to modify the configuration value.

- `enable_spark_load`

Open spark load and create resource. The default value is false. This feature is turned off.

- `spark_load_default_timeout_second`

The default timeout for tasks is 259200 seconds (3 days).

- `spark_home_default_dir`

Spark client path (`Fe/lib/spark2x`).

- `spark_resource_path`

The path of the packaged spark dependent file (empty by default).

- `spark_launcher_log_dir`

The directory where the spark client’s commit log is stored (`Fe/log/spark)_launcher_log`）.

- `yarn_client_path`

The path of the yarn binary executable file (‘Fe/lib/yarn-client/Hadoop/bin/yarn’).

- `yarn_config_dir`

The path to generate the yarn configuration file (`Fe/lib/yarn-config`).

Best practices

Application scenarios

The most suitable scenario to use spark load is that the raw data is in the file system (HDFS), and the amount of data is tens of GB to TB. Stream load or broker load is recommended for small amount of data.

FAQ

- Spark load does not yet support the import of Doris table fields that are of type String. If your table fields are of type String, please change them to type varchar, otherwise the import will fail, prompting `type:ETL_QUALITY_UNSATISFIED; msg: ↪ quality not good enough to cancel`

- When using spark load, the `HADOOP_CONF_DIR` environment variable is no set in the `spark-env.sh`.

If the `HADOOP_CONF_DIR` environment variable is not set, the error `When running with master 'yarn' either HADOOP_CONF_DIR or YARN_CONF_DIR must be set in the environment` will be reported.

- When using spark load, the `spark_home_default_dir` does not specify correctly.

The spark submit command is used when submitting a spark job. If `spark_home_default_dir` is set incorrectly, an error `Cannot run program 'xxx/bin/spark_submit', error = 2, no such file or directory` will be reported.

- When using spark load, `spark_resource_path` does not point to the packaged zip file.

If `spark_resource_path` is not set correctly. An error `file XXX/jars/spark-2x.zip` does not exist will be reported.

- When using spark load `yarn_client_path` does not point to a executable file of yarn.

If `yarn_client_path` is not set correctly. An error `yarn client does not exist in path: XXX/yarn-client/hadoop/bin/yarn` will be reported.

- When using spark load, the `JAVA_HOME` environment variable is no set in the `hadoop-config.sh` on the yarn clinet.

If the `JAVA_HOME` environment variable is not set, the error `yarn application kill failed. app id: xxx, load job id: xxx, msg: which: no xxx/lib/yarn-client/hadoop/bin/yarn in ((null))Error: JAVA_HOME is not set and could not be found` will be reported.

More Help

For more detailed syntax used by Spark Load, you can enter `HELP SPARK LOAD` on the Mysql client command line for more help.

2.4.1.3.5 Stream load

Stream load is a synchronous way of importing. Users import local files or data streams into Doris by sending HTTP protocol requests. Stream load synchronously executes the import and returns the import result. Users can directly determine whether the import is successful by the return body of the request.

Stream load is mainly suitable for importing local files or data from data streams through procedures.

Basic Principles

The following figure shows the main flow of Stream load, omitting some import details.

```
                         ^      +
                         |      |
                         |      | 1A. User submit load to FE
                         |      |
                         |   +--v-----------+
                         |   | FE           |
5. Return result to user |   +--+-----------+
                         |      |
                         |      | 2. Redirect to BE
                         |      |
                         |   +--v-----------+
                         +---+Coordinator BE| 1B. User submit load to BE
                             +-+-----+----+-+
                               |     |    |
                         +-----+     |    +-----+
                         |           |          | 3. Distrbute data
                         |           |          |
                       +-v-+       +-v-+      +-v-+
                       |BE |       |BE |      |BE |
                       +---+       +---+      +---+
```

In Stream load, Doris selects a node as the Coordinator node. This node is responsible for receiving data and distributing data to other data nodes.

Users submit import commands through HTTP protocol. If submitted to FE, FE forwards the request to a BE via the HTTP redirect instruction. Users can also submit import commands directly to a specified BE.

The final result of the import is returned to the user by Coordinator BE.

Support data format

Stream Load currently supports data formats: CSV (text), JSON

PARQUET and ORC 1.2+ support PARQUET and ORC

Basic operations

Create Load

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

format 1.2 supports csv_with_names (support csv file line header filter), csv_with_names_and_types (support 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.

- enable_profile

When `enable_profile` is true, the Stream Load profile will be printed to the log. Otherwise it won't print.

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
```

```
  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."
  }
```

```
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
```

```
  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."
  }
```

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.

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
```

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.

2.4.1.3.6 S3 Load

Starting from version 0.14, Doris supports the direct import of data from online storage systems that support the S3 protocol through the S3 protocol.

This document mainly introduces how to import data stored in AWS S3. It also supports the import of other object storage systems that support the S3 protocol, such as Baidu Cloud’s BOS, Alibaba Cloud’s OSS and Tencent Cloud’s COS, etc.

Applicable scenarios

- Source data in S3 protocol accessible storage systems, such as S3, BOS.
- Data volumes range from tens to hundreds of GB.

Preparing

1. Standard AK and SK First, you need to find or regenerate AWS `Access keys`, you can find the generation method in `My` ↪ `Security Credentials` of AWS console, as shown in the following figure: AK_SK Select `Create New Access Key` and pay attention to save and generate AK and SK.
2. Prepare REGION and ENDPOINT REGION can be selected when creating the bucket or can be viewed in the bucket list. ENDPOINT can be found through REGION on the following page AWS Documentation

Other cloud storage systems can find relevant information compatible with S3 in corresponding documents

Start Loading

Like Broker Load just replace `WITH BROKER broker_name ()` with

```
    WITH S3
    (
        "AWS_ENDPOINT" = "AWS_ENDPOINT",
        "AWS_ACCESS_KEY" = "AWS_ACCESS_KEY",
        "AWS_SECRET_KEY"="AWS_SECRET_KEY",
        "AWS_REGION" = "AWS_REGION"
    )
```

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"
    );
```

FAQ

1. S3 SDK uses 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:

```
  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"
 )
```

2. Support using temporary security credentials to access object stores that support the S3 protocol:

```
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"
)
```

2.4.1.3.7 Insert Into

The use of Insert Into statements is similar to that of Insert Into statements in databases such as MySQL. But in Doris, all data writing is a separate import job. So Insert Into is also introduced here as an import method.

The main Insert Into command contains the following two kinds;

- INSERT INTO tbl SELECT …
- INSERT INTO tbl (col1, col2, …) VALUES (1, 2, …), (1,3, …);

The second command is for Demo only, not in a test or production environment.

Import operations and load results

The Insert Into command needs to be submitted through MySQL protocol. Creating an import request returns the import result synchronously.

The following are examples of the use of two Insert Intos:

```
INSERT INTO tbl2 WITH LABEL label1 SELECT * FROM tbl3;
INSERT INTO tbl1 VALUES ("qweasdzxcqweasdzxc"), ("a");
```

Note: When you need to use `CTE(Common Table Expressions)` as the query part in an insert operation, you must specify the `WITH LABEL` and column list parts or wrap CTE. Example:

```
INSERT INTO tbl1 WITH LABEL label1
WITH cte1 AS (SELECT * FROM tbl1), cte2 AS (SELECT * FROM tbl2)
SELECT k1 FROM cte1 JOIN cte2 WHERE cte1.k1 = 1;


INSERT INTO tbl1 (k1)
WITH cte1 AS (SELECT * FROM tbl1), cte2 AS (SELECT * FROM tbl2)
SELECT k1 FROM cte1 JOIN cte2 WHERE cte1.k1 = 1;

INSERT INTO tbl1 (k1)
select * from (
WITH cte1 AS (SELECT * FROM tbl1), cte2 AS (SELECT * FROM tbl2)
SELECT k1 FROM cte1 JOIN cte2 WHERE cte1.k1 = 1) as ret
```

For specific parameter description, you can refer to INSERT INTO command or execute `HELP INSERT` to see its help documentation for better use of this import method.

Insert Into itself is a SQL command, and the return result is divided into the following types according to the different execution results:

1. Result set is empty

If the result set of the insert corresponding SELECT statement is empty, it is returned 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 loaded.

2. 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 is successful and data 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 load job. Each load job has a label that is unique within a single database
    ↪ .

`status` indicates whether the loaded data is visible. If visible, show `visible`, if not, show
    ↪ `committed`.

`txnId` is the id of the load transaction corresponding to this insert.

The `err` field displays some other unexpected errors.

When user need to view the filtered rows, the user can use 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
    ↪ following **View Error Lines** Summary.    **"Data is not visible" is a temporary
    ↪ status, this batch of data must be visible eventually**

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 data is visible.
```

2. Insert fails

   Execution failure indicates that no data was successfully loaded, and returns as follows:

```
    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_ae8de850e8de850
```

```
Where `ERROR 1064 (HY000): all partitions have no load data` shows the reason for the failure.
    ↪ The latter url can be used to query the wrong data. For details, see the following **
    ↪ View Error Lines** Summary.
```

In summary, the correct processing logic for the results returned by the insert operation should be:

1. If the returned result is `ERROR 1064 (HY000)`, it means that the import failed.
2. If the returned result is `Query OK`, it means the execution was successful.
3. If `rows affected` is 0, the result set is empty and no data is loaded.
4. If`rows affected` is greater than 0:
   1. If `status` is`committed`, the data is not yet visible. You need to check the status through the `show transaction` statement until `visible`.
   2. If `status` is`visible`, the data is loaded successfully.
5. If `warnings` is greater than 0, it means that some data is filtered. You can get the url through the `show load` statement to see the filtered rows.

In the previous section, we described how to follow up on the results of insert operations. However, it is difficult to get the json string of the returned result in some mysql libraries. Therefore, Doris also provides the `SHOW LAST INSERT` command to explicitly retrieve the results of the last insert operation.

After executing an insert operation, you can execute `SHOW LAST INSERT` on the same session connection. This command returns the result of the most recent insert operation, e.g.

```
mysql> show last insert\G
*************************** 1. row ***************************
    TransactionId: 64067
            Label: insert_ba8f33aea9544866-8ed77e2844d0cc9b
         Database: default_cluster:db1
            Table: t1
TransactionStatus: VISIBLE
       LoadedRows: 2
     FilteredRows: 0
```

This command returns the insert results and the details of the corresponding transaction. Therefore, you can continue to execute the `show last insert` command after each insert operation to get the insert results.

> Note: This command will only return the results of the last insert operation within the same session connection. If the connection is broken or replaced with a new one, the empty set will be returned.

Relevant System Configuration

FE configuration

- time out

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, Insert Into does not support custom import timeout time. All Insert Into imports have a uniform timeout time. The default timeout time is 1 hour. If the imported source file cannot complete the import within the specified time, the parameter `insert_load_default_timeout_second` of FE needs to be adjusted.

At the same time, the Insert Into statement receives the restriction of the Session variable `insert_timeout`. You can increase the timeout time by `SET insert_timeout = xxx;` in seconds.

Session Variables

- enable_insert_strict

The Insert Into import itself cannot control the tolerable error rate of the import. Users can only use the Session parameter `enable ↪ _insert_strict`. When this parameter is set to false, it indicates that at least one data has been imported correctly, and then it returns successfully. When this parameter is set to true, the import fails if there is a data error. The default is false. It can be set by `SET enable_insert_strict = true;`.

- query_timeout

Insert Into itself is also an SQL command, and the Insert Into statement is restricted by the Session variable `insert_timeout`. You can increase the timeout time by `SET insert_timeout = xxx;` in seconds.

Best Practices

Application scenarios

1. Users want to import only a few false data to verify the functionality of Doris system. The grammar of INSERT INTO VALUES is suitable at this time.
2. Users want to convert the data already in the Doris table into ETL and import it into a new Doris table, which is suitable for using INSERT INTO SELECT grammar.
3. Users can create an external table, such as MySQL external table mapping a table in MySQL system. Or create Broker external tables to map data files on HDFS. Then the data from the external table is imported into the Doris table for storage through the INSERT INTO SELECT grammar.

Data volume

Insert Into has no limitation on the amount of data, and large data imports can also be supported. However, Insert Into has a default timeout time, and the amount of imported data estimated by users is too large, so it is necessary to modify the system's Insert Into import timeout time.

```
Import data volume = 36G or less than 3600s*10M/s
Among them, 10M/s is the maximum import speed limit. Users need to calculate the average import
    ↪ speed according to the current cluster situation to replace 10M/s in the formula.
```

Complete examples

Users have a table store sales in the database sales. Users create a table called bj store sales in the database sales. Users want to import the data recorded in the store sales into the new table bj store sales. The amount of data imported is about 10G.

```
large sales scheme
(id, total, user_id, sale_timestamp, region)

Order large sales schedule:
(id, total, user_id, sale_timestamp)
```

Cluster situation: The average import speed of current user cluster is about 5M/s

- Step1: Determine whether you want to modify the default timeout of Insert Into

```
  Calculate the approximate time of import
  10G / 5M /s = 2000s

  Modify FE configuration
  insert_load_default_timeout_second = 2000
```

- Step2: Create Import Tasks

Since users want to ETL data from a table and import it into the target table, they should use the Insert in query_stmt mode to import it.

```
  INSERT INTO bj_store_sales SELECT id, total, user_id, sale_timestamp FROM store_sales where
      ↪ region = "bj";
```

Common Questions

- View the wrong line

Because Insert Into can't control the error rate, it can only tolerate or ignore the error data completely by `enable_insert_ ↪ strict`. So if `enable_insert_strict` is set to true, Insert Into may fail. If `enable_insert_strict` is set to false, then only some qualified data may be imported. However, in either case, Doris is currently unable to provide the ability to view substandard data rows. Therefore, the user cannot view the specific import error through the Insert Into statement.

The causes of errors are usually: source data column length exceeds destination data column length, column type mismatch, partition mismatch, column order mismatch, etc. When it' s still impossible to check for problems. At present, it is only recommended that the SELECT command in the Insert Into statement be run to export the data to a file, and then import the file through Stream load to see the specific errors.

more help

For more detailed syntax and best practices used by insert into, see insert command manual, you can also enter `HELP INSERT` in the MySql client command line for more help information.

2.4.1.3.8 Load JSON Format Data

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": ["76959836937749932879763573681792701709",
    ↪ "26017042825937891692910431521038521227"]}
```

```
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 | [76959836937749932879763573681792701709, 26017042825937891692910431521038521227]   |
+------+------------------------------------------------------------------------------------+
```

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)

2.4.2 Export

2.4.2.1 Data export

Export is a function provided by Doris to export data. This function can export user-specified table or partition data in text format to remote storage through Broker process, such as HDFS / Object storage (supports S3 protocol) etc.

This document mainly introduces the basic principles, usage, best practices and precautions of Export.

2.4.2.1.1 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.
- Broker: Doris can manipulate files for remote storage through the Broker process.
- Tablet: Data fragmentation. A table is divided into multiple data fragments.

2.4.2.1.2 Principle

After the user submits an Export job. Doris counts all Tablets involved in this job. These tablets are then grouped to generate a special query plan for each group. The query plan reads the data on the included tablet and then writes the data to the specified path of the remote storage through Broker. It can also be directly exported to the remote storage that supports S3 protocol through S3 protocol.

The overall mode of dispatch is as follows:

```
+--------+
| Client |
+---+----+
    |  1. Submit Job
    |
+---v--------------------+
| FE                     |
|                        |
| +-------------------+  |
| | ExportPendingTask |  |
| +-------------------+  |
|                        | 2. Generate Tasks
| +--------------------+ |
| | ExportExporingTask | |
| +--------------------+ |
|                        |
| +-----------+          |     +----+   +------+   +---------+
| | QueryPlan +----------------> BE +--->Broker+--->         |
| +-----------+          |     +----+   +------+   | Remote  |
| +-----------+          |     +----+   +------+   | Storage |
| | QueryPlan +----------------> BE +--->Broker+--->         |
| +-----------+          |     +----+   +------+   +---------+
+------------------------+         3. Execute Tasks
```

1. The user submits an Export job to FE.
2. FE's Export scheduler performs an Export job in two stages:
   1. PENDING: FE generates Export Pending Task, sends snapshot command to BE, and takes a snapshot of all Tablets involved. And generate multiple query plans.
   2. EXPORTING: FE generates Export ExportingTask and starts executing the query plan.

query plan splitting

The Export job generates multiple query plans, each of which scans a portion of the Tablet. The number of Tablets scanned by each query plan is specified by the FE configuration parameter `export_tablet_num_per_task`, which defaults to 5. That is, assuming a total of 100 Tablets, 20 query plans will be generated. Users can also specify this number by the job attribute `tablet_num_per_task`, when submitting a job.

Multiple query plans for a job are executed sequentially.

Query Plan Execution

A query plan scans multiple fragments, organizes read data in rows, batches every 1024 actions, and writes Broker to remote storage.

The query plan will automatically retry three times if it encounters errors. If a query plan fails three retries, the entire job fails.

Doris will first create a temporary directory named `doris_export_tmp_12345` (where 12345 is the job id) in the specified remote storage path. The exported data is first written to this temporary directory. Each query plan generates a file with an example file name:

`export-data-c69fcf2b6db5420f-a96b94c1ff8bccef-1561453713822`

Among them, `c69fcf2b6db5420f-a96b94c1ff8bccef` is the query ID of the query plan. 1561453713822 Timestamp generated for the file.

When all data is exported, Doris will rename these files to the user-specified path.

Broker parameter

Export needs to use the Broker process to access remote storage. Different brokers need to provide different parameters. For details, please refer to Broker documentation

2.4.2.1.3 Start Export

For detailed usage of Export, please refer to SHOW EXPORT.

Export's detailed commands can be passed through `HELP EXPORT;` 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",
    "exec_mem_limit"="2147483648",
    "timeout" = "3600"
)
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 `'\x07'`.
- `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 `'\x07'`.
- `exec_mem_limit`: Represents the memory usage limitation of a query plan on a single BE in an Export job. Default 2GB. Unit bytes.
- `timeout`: homework timeout. Default 2 hours. Unit seconds.
- `tablet_num_per_task`: The maximum number of fragments allocated per query plan. The default is 5.

Export to object storage (supports S3 protocol)

```
EXPORT TABLE test TO "s3://bucket/path/to/export/dir/" WITH S3  (
        "AWS_ENDPOINT" = "http://host",
        "AWS_ACCESS_KEY" = "AK",
        "AWS_SECRET_KEY"="SK",
        "AWS_REGION" = "region"
    );
```

- `AWS_ACCESS_KEY/AWS_SECRET_KEY:` Is your key to access the object storage API.
- `AWS_ENDPOINT:` Endpoint indicates the access domain name of object storage external services.
- `AWS_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":["*"],"exec mem limit":2147483648,"column separator":",","line
    ↪ delimiter":"\n","tablet num":1,"broker":"hdfs","coord num":1,"db":"default_cluster:db1"
    ↪ ,"tbl":"tbl3"}
      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
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 a total of 10 query plans 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. * Represents all partitions.
  - exec MEM limit: query plan memory usage limit. Unit bytes.
  - 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.
  - Coord num: Number of query plans.
- 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.

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%";
```

2.4.2.1.4 Best Practices

Splitting Query Plans

How many query plans need to be executed for an Export job depends on the total number of Tablets and how many Tablets can be allocated for a query plan at most. Since multiple query plans are executed serially, the execution time of jobs can be reduced if more fragments are processed by one query plan. However, if the query plan fails (e.g., the RPC fails to call Broker, the remote storage jitters, etc.), too many tablets can lead to a higher retry cost of a query plan. Therefore, it is necessary to arrange the number of query plans and the number of fragments to be scanned for each query plan in order to balance the execution time and the success rate of execution. It is generally recommended that the amount of data scanned by a query plan be within 3-5 GB (the size and number of tables in a table can be viewed by `SHOW TABLETS FROM tbl_name;` statement.

exec_mem_limit

Usually, a query plan for an Export job has only two parts `scan`- `export`, and does not involve computing logic that requires too much memory. So usually the default memory limit of 2GB can satisfy the requirement. But in some scenarios, such as a query plan, too many Tablets need to be scanned on the same BE, or too many data versions of Tablets, may lead to insufficient memory. At this point, larger memory needs to be set through this parameter, such as 4 GB, 8 GB, etc.

2.4.2.1.5 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 `__doris_export_tmp_xxx` temporary directory generated in the remote storage and the generated files will not be deleted, requiring the user to delete them manually.
- If the Export job runs successfully, the `__doris_export_tmp_xxx` directory generated in the remote storage may be retained or cleared according to the file system semantics of the remote storage. For example, in object storage (supporting the S3 protocol), after removing the last file in a directory through rename operation, the directory will also be deleted. If the directory is not cleared, the user can clear it manually.
- When the Export runs successfully or fails, the FE reboots or cuts, then some information of the jobs displayed by `SHOW`
  `↪ EXPORT` will be lost and cannot be viewed.
- Export jobs only export data from Base tables, not Rollup Index.
- Export jobs scan data and occupy IO resources, which may affect the query latency of the system.

2.4.2.1.6 Relevant configuration

FE

- `expo_checker_interval_second`: Scheduling interval of Export job scheduler, default is 5 seconds. Setting this parameter requires restarting FE.

- `export_running_job_num_limit`: Limit on the number of Export jobs running. If exceeded, the job will wait and be in PENDING state. The default is 5, which can be adjusted at run time.
- `Export_task_default_timeout_second`: Export job default timeout time. The default is 2 hours. It can be adjusted at run time.
- `export_tablet_num_per_task`: The maximum number of fragments that a query plan is responsible for. The default is 5.
- `label`: The label of this Export job. Doris will generate a label for an Export job if this param is not set.

2.4.2.1.7 More Help

For more detailed syntax and best practices used by Export, please refer to the Export command manual, you can also You can enter HELP EXPORT at the command line of the MySql client for more help.

2.4.2.2 Export Query Result

This document describes how to use the SELECT INTO OUTFILE command to export query results.

2.4.2.2.1 Example

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 file

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.

2.4.2.2.2 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                                                    |
+-----------------------------------------------------------------------------+
```

2.4.2.2.3 Usage example

For details, please refer to OUTFILE Document.

2.4.2.2.4 Return result

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 ...
```

2.4.2.2.5 Notice

- 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 spliting 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.

2.4.2.2.6 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.

2.4.2.3 Using MySQL Dump

Doris has supported exporting data or table structures through the `mysqldump` tool after version 0.15

2.4.2.3.1 Example

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
5. Since there is no concept of tablespace in mysql in Doris, add the `--no-tablespaces` parameter when using `mysqldump`
6. 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.

2.4.3 Update and Delete

2.4.3.1 Batch Delete

Currently, Doris supports multiple import methods such as broker load, routine load, stream load, etc. The data can only be deleted through the delete statement at present. When the delete statement is used to delete, a new data version will be generated every time delete is executed. Frequent deletion will seriously affect the query performance, and when using the delete method to delete, it is achieved by generating an empty rowset to record the deletion conditions. Each time you read, you must filter the deletion jump conditions. Also when there are many conditions, Performance affects. Compared with other systems, the implementation of greenplum is more like a traditional database product. Snowflake is implemented through the merge syntax.

For scenarios similar to the import of cdc data, insert and delete in the data data generally appear interspersed. In this scenario, our current import method is not enough, even if we can separate insert and delete, it can solve the import problem , But still cannot solve the problem of deletion. Use the batch delete function to solve the needs of these scenarios. There are three ways to merge data import: 1. APPEND: All data are appended to existing data 2. DELETE: delete all rows with the same key column value as the imported data(When there is a `sequence` column in the table, the same primary key and the logic of the size of the sequence column must be satisfied at the same time to delete it correctly, see use case 4 below for details.) 3. MERGE: APPEND or DELETE according to DELETE ON decision

2.4.3.1.1 Fundamental

This is achieved by adding a hidden column `__DORIS_DELETE_SIGN__`, because we are only doing batch deletion on the unique model, so we only need to add a hidden column whose type is bool and the aggregate function is replace. In be, the various aggregation write processes are the same as normal columns, and there are two read schemes:

Remove `__DORIS_DELETE_SIGN__` when fe encounters extensions such as *, and add the condition of `__DORIS_DELETE_SIGN`
`↪ __!= true` by default When be reads, a column is added for judgment, and the condition is used to determine whether to delete.

Import

When importing, set the value of the hidden column to the value of the `DELETE ON` expression during fe parsing. The other aggregation behaviors are the same as the replace aggregation column.

Read

When reading, add the condition of `__DORIS_DELETE_SIGN__!= true` to all olapScanNodes with hidden columns, be does not perceive 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

In Base Compaction, delete the rows marked for deletion to reduce the space occupied by data.

2.4.3.1.2 Enable bulk delete support

There are two ways of enabling batch delete support:

1. By adding `enable_batch_delete_by_default=true` in the fe configuration file, all newly created tables after restarting fe support batch deletion, this option defaults to false
2. For tables that have not changed the above fe configuration or for existing tables that do not support the bulk delete function, you can use the following statement: `ALTER TABLE tablename ENABLE FEATURE "BATCH_DELETE"` to enable the batch delete.

If you want to determine whether a table supports batch delete, you can set a session variable to display the hidden columns `SET show_hidden_columns=true`, and then use `desc tablename`, if there is a `__DORIS_DELETE_SIGN__` column in the output, it is supported, if not, 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"
    );
```

2.4.3.1.3 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.
3. 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 select statement or open a new session to run the select statement.

2.4.3.1.4 Usage example

Check if bulk delete support is enabled

```
mysql> SET show_hidden_columns=true;
Query OK, 0 rows affected (0.00 sec)

mysql> DESC test;
+-----------------------+--------------+------+-------+---------+---------+
| 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 |
+-----------------------+--------------+------+-------+---------+---------+
4 rows in set (0.00 sec)
```

Stream Load usage example

1. Import data normally:

```
curl --location-trusted -u root: -H "column_separator:," -H "columns: siteid, citycode,
    ↪ username, pv" -H "merge_type: APPEND" -T ~/table1_data http://127.0.0.1: 8130/api/
    ↪ test/table1/_stream_load
```

The APPEND condition can be omitted, which has the same effect as the following statement:

```
curl --location-trusted -u root: -H "column_separator:," -H "columns: siteid, citycode,
    ↪ username, pv" -T ~/table1_data http://127.0.0.1:8130/api/test/table1 /_stream_load
```

2. Delete all data with the same key as the imported data

```
curl --location-trusted -u root: -H "column_separator:," -H "columns: siteid, citycode,
    ↪ username, pv" -H "merge_type: DELETE" -T ~/table1_data http://127.0.0.1: 8130/api/
    ↪ test/table1/_stream_load
```

Before load:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      3 |        2 | tom      |    2 |
|      4 |        3 | bush     |    3 |
|      5 |        3 | helen    |    3 |
+--------+----------+----------+------+
```

Load data:

```
3,2,tom,0
```

After load:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      4 |        3 | bush     |    3 |
|      5 |        3 | helen    |    3 |
+--------+----------+----------+------+
```

3. Import the same row as the key column of the row with `site_id=1`

```
curl --location-trusted -u root: -H "column_separator:," -H "columns: siteid, citycode,
    ↪ username, pv" -H "merge_type: MERGE" -H "delete: siteid=1" -T ~/ table1_data http
    ↪ ://127.0.0.1:8130/api/test/table1/_stream_load
```

Before load:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      4 |        3 | bush     |    3 |
|      5 |        3 | helen    |    3 |
|      1 |        1 | jim      |    2 |
+--------+----------+----------+------+
```

Load data:

```
2,1,grace,2
3,2,tom,2
1,1,jim,2
```

After load:

```
+--------+----------+----------+------+
| siteid | citycode | username | pv   |
+--------+----------+----------+------+
|      4 |        3 | bush     |    3 |
|      2 |        1 | grace    |    2 |
|      3 |        2 | tom      |    2 |
|      5 |        3 | helen    |    3 |
+--------+----------+----------+------+
```

4. When the table has the sequence column, delete all data with the same key as the imported data

```
curl --location-trusted -u root: -H "column_separator:," -H "columns: name, gender, age" -H "
    ↪ function_column.sequence_col: age" -H "merge_type: DELETE"  -T ~/table1_data http
    ↪ ://127.0.0.1:8130/api/test/table1/_stream_load
```

When the unique table has the sequence column, sequence column is used as the basis for the replacement order of the REPLACE aggregate function under the same key column, and the larger value can replace the smaller value. If you want delete some data, the imported data must have the same key and the sequence column must be larger or equal than before.

for example, one table like this:

```
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)
```

Before load:

```
+-------+--------+------+
| name  | gender | age  |
+-------+--------+------+
| li    | male   |   10 |
| wang  | male   |   14 |
| zhang | male   |   12 |
+-------+--------+------+
```

If you load data like this:

```
li,male,10
```

After load:

```
+-------+--------+------+
| name  | gender | age  |
+-------+--------+------+
| wang  | male   |   14 |
| zhang | male   |   12 |
+-------+--------+------+
```

You will find that the data is deleted.

```
li,male,10
```

But if you load data like this:

```
li,male,9
```

After load:

```
+-------+--------+------+
| name  | gender | age  |
+-------+--------+------+
| li    | male   |   10 |
| wang  | male   |   14 |
| zhang | male   |   12 |
+-------+--------+------+
```

You will find that the data is not deleted.

```
li,male,10
```

This is because in the underlying dependencies, it will first judge the case of the same key, display the row data with a large value in the sequence column, and then check whether the `__DORIS_DELETE_SIGN__` value of the row is 1. If it is 1, it will not be displayed. If it is 0, it will still be read out.

When data is written and deleted at the same time in the imported data (such as in the Flink CDC scenario), using the sequence column can effectively ensure the consistency when the data arrives out of order, avoiding the deletion operation of an old version that arrives later, and accidentally deleting The new version of the data that arrives first.

2.4.3.2 update

This article mainly describes how to use the UPDATE command to operate if we need to modify or update the data in Doris. The data update is limited to the version of Doris and can only be used in Doris Version 0.15.x +.

2.4.3.2.1 Applicable scenarios

- Modify its value for rows that meet certain conditions;
- Point update, small range update, the row to be updated is preferably a very small part of the entire table;
- The update command can only be executed on a table with a Unique data model.

2.4.3.2.2 Fundamentals

Use the query engine's own where filtering logic to filter the rows that need to be updated from the table to be updated. Then use the Unique model's own Value column replacement logic to change the rows to be updated and reinsert them into the table. This enables row-level updates.

Synchronization

The Update syntax is a synchronization syntax in Doris. If the Update statement succeeds, the update succeeds and the data is visible.

Performance

The performance of the Update statement is closely related to the number of rows to be updated and the retrieval efficiency of the condition.

- Number of rows to be updated: The more rows to be updated, the slower the Update statement will be. This is consistent with the principle of importing. Doris updates are more suitable for occasional update scenarios, such as changing the values of individual rows. Doris is not suitable for large batches of data changes. Large modifications can make Update statements take a long time to run.

- Condition retrieval efficiency: Doris Update implements the principle of reading the rows that satisfy the condition first, so if the condition retrieval efficiency is high, the Update will be faster. The condition column should ideally be hit, indexed, or bucket clipped. This way Doris does not need to scan the entire table and can quickly locate the rows that need to be updated. This improves update efficiency. It is strongly discouraged to include the UNIQUE model value column in the condition column.

Concurrency Control

By default, multiple concurrent Update operations on the same table are not allowed at the same time.

The main reason for this is that Doris currently supports row updates, which means that even if the user declares SET `v2 = 1`, virtually all other Value columns will be overwritten (even though the values are not changed).

This presents a problem in that if two Update operations update the same row at the same time, the behavior may be indeterminate. That is, there may be dirty data.

However, in practice, the concurrency limit can be turned on manually if the user himself can guarantee that even if concurrent updates are performed, they will not operate on the same row at the same time. This is done by modifying the FE configuration `enable_concurrent_update`. When the configuration value is true, there is no limit on concurrent updates. > Note: After enabling the configuration, there will be certain performance risks. You can refer to the performance section above to improve update efficiency.

2.4.3.2.3 Risks of use

Since Doris currently supports row updates and uses a two-step read-and-write operation, there is uncertainty about the outcome of an Update statement if it modifies the same row as another Import or Delete statement.

Therefore, when using Doris, you must be careful to control the concurrency of Update statements and other DML statements on the user side itself.

2.4.3.2.4 Usage example

Suppose there is an order table in Doris, where the order id is the Key column, the order status and the order amount are the Value column. The data status is as follows:

order id	order amount	order status
1	100	Pending Payment

```
+----------+--------------+--------------+
| order_id | order_amount | order_status |
+----------+--------------+--------------+
| 1        |          100 | 待付款       |
+----------+--------------+--------------+
1 row in set (0.01 sec)
```

At this time, after the user clicks to pay, the Doris system needs to change the status of the order with the order id ‘1’ to ‘Pending Shipping’, and the Update function needs to be used.

```
mysql> UPDATE test_order SET order_status = 'Pending Shipping' WHERE order_id = 1;
Query OK, 1 row affected (0.11 sec)
{'label':'update_20ae22daf0354fe0-b5aceeaaddc666c5', 'status':'VISIBLE', 'txnId':'33', 'queryId':
    ↪ '20ae22daf0354fe0-b5aceeaaddc666c5'}
```

The result after the update is as follows

```
+----------+--------------+------------------+
| order_id | order_amount | order_status     |
+----------+--------------+------------------+
| 1        |          100 | Pending Shipping |
+----------+--------------+------------------+
1 row in set (0.01 sec)
```

After the user executes the UPDATE command, the system performs the following three steps.

Step 1: Read the rows that satisfy WHERE order id=1 (1, 100, ‘pending payment’) Step 2: Change the order status of the row from ‘Pending Payment’ to ‘Pending Shipping’ (1, 100, ‘Pending shipment’) Step 3: Insert the updated row back into the table to achieve the updated effect.

order id	order amount	order status
1	100	Pending Payment
1	100	Pending shipments

Since the table order is a UNIQUE model, the rows with the same Key, after which the latter will take effect, so the final effect is as follows.

order id	order amount	order status
1	100	Pending shipments

2.4.3.2.5 Update primary key column

Currently, the Update operation only supports updating the Value column, and the update of the Key column can refer to Using FlinkCDC to update key column

2.4.3.2.6 More Help

For more detailed syntax used by data update, please refer to the update command manual , you can also enter HELP UPDATE in the Mysql client command line to get more help information.

2.4.3.3 Delete

Delete is different from other import methods. It is a synchronization process, similar to Insert into. All Delete operations are an independent import job in Doris. Generally, the Delete statement needs to specify the table and partition and delete conditions to filter the data to be deleted. , and will delete the data of the base table and the rollup table at the same time.

2.4.3.3.1 Syntax

Please refer to the official website for the DELETE syntax of the delete operation.

2.4.3.3.2 Delete Result

The delete command is an SQL command, and the returned results are synchronous. It can be divided into the following types:

1. Successful visible

If delete completes successfully and is visible, the following results will be returned, `query OK` indicates success.

```
mysql> delete from test_tbl PARTITION p1 where k1 = 1;
 Query OK, 0 rows affected (0.04 sec)
 {'label':'delete_e7830c72-eb14-4cb9-bbb6-eebd4511d251', 'status':'VISIBLE', 'txnId':'4005'}
```

2. Submitted successfully, but not visible

The transaction submission of Doris is divided into two steps: submission and publish version. Only after the publish version step is completed, the result will be visible to the user. If it has been submitted successfully, then it can be considered that the publish version step will eventually success. Doris will try to wait for publishing for a period of time after submitting. If it has timed out, even if the publishing version has not been completed, it will return to the user in priority and prompt the user that the submission has been completed but not visible. If delete has been committed and executed, but has not been published and visible, the following results will be returned.

```
mysql> delete from test_tbl PARTITION p1 where k1 = 1;
Query OK, 0 rows affected (0.04 sec)
{'label':'delete_e7830c72-eb14-4cb9-bbb6-eebd4511d251', 'status':'COMMITTED', 'txnId':'4005',
    ↪ 'err':'delete job is committed but may be taking effect later' }
```

```
The result will return a JSON string at the same time:
```

`affected rows`: Indicates the row affected by this deletion. Since the deletion of Doris is currently a logical deletion, the value is always 0.

`label`: The label generated automatically to be the signature of the delete jobs. Each job has a unique label within a single database.

`status`: Indicates whether the data deletion is visible. If it is visible, `visible` will be displayed. If it is not visible, `committed` will be displayed.

`txnId`: The transaction ID corresponding to the delete job

`err`: Field will display some details of this deletion

3. Commit failed, transaction cancelled

If the delete statement is not submitted successfully, it will be automatically aborted by Doris and the following results will be returned

```
mysql> delete from test_tbl partition p1 where k1 > 80;
ERROR 1064 (HY000): errCode = 2, detailMessage = {错误原因}
```

example:

A timeout deletion will return the timeout and unfinished replicas displayed as `(tablet = replica)`

```
mysql> delete from test_tbl partition p1 where k1 > 80;
ERROR 1064 (HY000): errCode = 2, detailMessage = failed to delete replicas from job: 4005,
    ↪ Unfinished replicas:10000=60000, 10001=60000, 10002=60000
```

The correct processing logic for the returned results of the delete operation is as follows:

1. If `Error 1064 (HY000)` is returned, deletion fails
2. If the returned result is `Query OK`, the deletion is successful
    1. If `status` is `committed`, the data deletion is committed and will be eventually invisible. Users can wait for a while and then use the `show delete` command to view the results.
    2. If `status` is `visible`, the data have been deleted successfully.

2.4.3.3.3 Delete operation related FE configuration

TIMEOUT configuration

In general, Doris’s deletion timeout is limited from 30 seconds to 5 minutes. The specific time can be adjusted through the following configuration items

- `tablet_delete_timeout_second`

The timeout of delete itself can be elastically changed by the number of tablets in the specified partition. This configuration represents the average timeout contributed by a tablet. The default value is 2.

Assuming that there are 5 tablets under the specified partition for this deletion, the timeout time available for the deletion is 10 seconds. Because the minimum timeout is 30 seconds which is higher than former timeout time, the final timeout is 30 seconds.

- `load_straggler_wait_second`

If the user estimates a large amount of data, so that the upper limit of 5 minutes is insufficient, the user can adjust the upper limit of timeout through this item, and the default value is 300.

The specific calculation rule of timeout(seconds)

`TIMEOUT = MIN(load_straggler_wait_second, MAX(30, tablet_delete_timeout_second * tablet_num))`

- `query_timeout`

Because delete itself is an SQL command, the deletion statement is also limited by the session variables, and the timeout is also affected by the session value `query'timeout`. You can increase the value by `set query'timeout = xxx`.

InPredicate configuration

- `max_allowed_in_element_num_of_delete`

If the user needs to take a lot of elements when using the in predicate, the user can adjust the upper limit of the allowed in elements number, and the default value is 1024.

2.4.3.3.4 Show delete history

The user can view the deletion completed in history through 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)
```

Note

Unlike the Insert into command, delete cannot specify `label` manually. For the concept of label, see the Insert Into documentation.

2.4.3.3.5 More Help

For more detailed syntax used by delete, see the delete command manual, You can also enter `HELP DELETE` in the Mysql client command line to get more help information

2.4.3.4 Sequence Column

The sequence column currently only supports the Uniq model. The Uniq model is mainly aimed at scenarios that require a unique primary key, which can guarantee the uniqueness constraint of the primary key. However, due to the REPLACE aggregation method, the replacement order of data imported in the same batch is not guaranteed. See Data Model. If the replacement order cannot be guaranteed, the specific data finally imported into the table cannot be determined, and there is uncertainty.

In order to solve this problem, Doris supports the sequence column. The user specifies the sequence column when importing. Under the same key column, the REPLACE aggregation type column will be replaced according to the value of the sequence column. The larger value can replace the smaller value, and vice versa. Cannot be replaced. This method leaves the determination of the order to the user, who controls the replacement order.

2.4.3.4.1 Applicable scene

Sequence columns can only be used under the Uniq data model.

2.4.3.4.2 Fundamental

By adding a hidden column `__DORIS_SEQUENCE_COL__`, the type of the column is specified by the user when creating the table, the specific value of the column is determined during import, and the REPLACE column is replaced according to this value.

Create Table

When creating a Uniq table, a hidden column `__DORIS_SEQUENCE_COL__` will be automatically added according to the user-specified type.

Import

When importing, fe sets the value of the hidden column to the value of the `order by` expression (broker load and routine load), or the value of the `function_column.sequence_col` expression (stream load) during the parsing process, the value column will be Replace with this value. The value of the hidden column `__DORIS_SEQUENCE_COL__` can be set to either a column in the data source or a column in the table structure.

Read

When the request contains the value column, the `__DORIS_SEQUENCE_COL__` column needs to be additionally read. This column is used as the basis for the replacement order of the REPLACE aggregate function under the same key column. The larger value can replace the smaller value, otherwise it cannot be replaced.

Cumulative Compaction

The principle is the same as that of the reading process during Cumulative Compaction.

Base Compaction

The principle is the same as the reading process during Base Compaction.

Syntax

There are two ways to create a table with sequence column. 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.

Set `sequence_col`(recommend)

When you create the Uniq table, you can specify the mapping of sequence column to other columns

```
PROPERTIES (
    "function_column.sequence_col" = 'column_name',
);
```

The 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.

The import method is the same as that without the sequence column. It is relatively simple and recommended.

Set `sequence_type`

When you create the Uniq table, you can specify the sequence column type

```
PROPERTIES (
    "function_column.sequence_type" = 'Date',
);
```

The sequence_type is used to specify the type of the sequence column, which can be integral and time (DATE / DATETIME).

The mapping column needs to be specified when importing.

Stream Load

The syntax of the stream load is to add the mapping of hidden columns corresponding to source_sequence in the `function_`
↪ `column.sequence_col` field in the header, for 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
```

Broker Load

Set the source_sequence field for the hidden column map at ORDER BY

```
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"
);
```

Routine Load

The mapping method is the same as above, as shown below

```
   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"
    );
```

2.4.3.4.3 Enable 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 column. For a table that does not support sequence column, if you want to use this function, 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.

If you are not sure whether a table supports sequence column, you can display hidden columns by setting a session variable SET ↪ show_hidden_columns=true, then use `desc tablename`, if there is a __DORIS_SEQUENCE_COL__ column in the output, it is supported, if not, it is not supported .

2.4.3.4.4 Usage example

Let’s take the stream Load as an example to show how to use it 1. Create a table that supports sequence column.

Create the test_table data table of the unique model and specify that the sequence column maps 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"
);
```

The table structure is shown below

```
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. Import data normally:

Import 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
```

Take the Stream Load as an example here and map the sequence column to the modify_date column

```
curl --location-trusted -u root: -T testData http://host:port/api/test/test_table/_stream_load
```

The results is

```
MySQL > select * from test_table;
+---------+------------+----------+-------------+---------+
| user_id | date       | group_id | modify_date | keyword |
+---------+------------+----------+-------------+---------+
|       1 | 2020-02-22 |        1 | 2020-03-05  | c       |
+---------+------------+----------+-------------+---------+
```

In this import, the c is eventually retained in the keyword column because the value of the sequence column (the value in modify_date) is the maximum value: ‘2020-03-05’.

3. Guarantee of substitution order

After the above steps are completed, import the following data

```
1       2020-02-22      1       2020-02-22      a
1       2020-02-22      1       2020-02-23      b
```

Query 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 this import, the c is eventually retained in the keyword column because the value of the sequence column (the value in modify_date) in all imports is the maximum value: ‘2020-03-05’. Try importing the following data again

```
1       2020-02-22      1       2020-02-22      a
1       2020-02-22      1       2020-03-23      w
```

Query data

```
MySQL [test]> select * from test_table;
+---------+------------+----------+-------------+---------+
| user_id | date       | group_id | modify_date | keyword |
+---------+------------+----------+-------------+---------+
|       1 | 2020-02-22 |        1 | 2020-03-23  | w       |
+---------+------------+----------+-------------+---------+
```

At this point, you can replace the original data in the table. To sum up, the sequence column will be compared among all the batches, the largest value of the same key will be imported into Doris table.

2.5 Advanced Usage

2.5.1 Alter Table

2.5.1.1 Schema Change

Users can modify the schema of existing tables through the Schema Change operation. Doris currently supports the following modifications:

- Add and delete columns
- Modify column type
- Adjust column order
- Add and modify Bloom Filter
- Add and delete bitmap index

This document mainly describes how to create a Schema Change job, as well as some considerations and frequently asked questions about Schema Change. ##### Glossary

- Base Table: When each table is created, it corresponds to a base table. The base table stores the complete data of this table. Rollups are usually created based on the data in the base table (and can also be created from other rollups).
- 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 incrementing Transaction ID.
- Rollup: Roll-up tables based on base tables or other rollups.

2.5.1.1.1 Basic Principles

The basic process of executing a Schema Change is to generate a copy of the index data of the new schema from the data of the original index. Among them, two parts of data conversion are required. 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.

```
+----------+
| Load Job |
+----+-----+
     |
     | Load job generates both origin and new index data
     |
     |      +------------------+ +---------------+
     |      | Origin Index     | | Origin Index  |
     +------> New Incoming Data| | History Data  |
     |      +------------------+ +------+--------+
     |                                  |
     |                                  | Convert history data
     |                                  |
     |      +------------------+ +------v--------+
     |      | New Index        | | New Index     |
     +------> New Incoming Data| | History Data  |
            +------------------+ +---------------+
```

Before starting the conversion of 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 guarantees that all import tasks after the watershed will generate data for both the original Index and the new Index. In this way, when the historical data conversion is completed, the data in the new Index can be guaranteed to be complete. ##### Create Job

The specific syntax for creating a Schema Change can be found in the help ALTER TABLE COLUMN for the description of the Schema Change section .

The creation of Schema Change is an asynchronous process. After the job is submitted successfully, the user needs to view the job progress through the `SHOW ALTER TABLE COLUMN` command. ##### View Job

`SHOW ALTER TABLE COLUMN` You can view the Schema Change jobs that are currently executing or completed. When multiple indexes are involved in a Schema Change job, the command displays multiple lines, each 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: A unique ID for each Schema Change job.
- TableName: The table name of the base table corresponding to Schema Change.
- CreateTime: Job creation time.
- FinishedTime: The end time of the job. If it is not finished, "N/A" is displayed.
- IndexName: The name of an Index involved in this modification.
- IndexId: The unique ID of the new Index.
- OriginIndexId: The unique ID of the old Index.
- SchemaVersion: Displayed in M: N format. M is the version of this Schema Change, and N is the corresponding hash value. With each Schema Change, the version is incremented.
- TransactionId: the watershed transaction ID of the conversion history data.
- State: The phase of the operation.
- PENDING: The job is waiting in the queue to be scheduled.
- WAITING_TXN: Wait for the import task before the watershed transaction ID to complete.
- RUNNING: Historical data conversion.
- FINISHED: The operation was successful.
- CANCELLED: The job failed.
- Msg: If the job fails, a failure message is displayed here.
- Progress: operation progress. Progress is displayed only in the RUNNING state. Progress is displayed in M/N. Where N is the total number of copies involved in the Schema Change. M is the number of copies of historical data conversion completed.
- Timeout: Job timeout time. Unit of second.

2.5.1.1.2 Cancel Job

In the case that the job status is not FINISHED or CANCELLED, you can cancel the Schema Change job with the following command: `CANCEL ALTER TABLE COLUMN FROM tbl_name;`

2.5.1.1.3 Best Practice

Schema Change can make multiple changes to multiple indexes in one job. For example: Source Schema:

```
+-----------+-------+------+------+------+---------+-------+
| 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     |       |
|           |       |      |      |      |         |       |
| rollup2   | k2    | INT  | No   | true | N/A     |       |
|           |       |      |      |      |         |       |
| rollup1   | k1    | INT  | No   | true | N/A     |       |
|           | k2    | INT  | No   | true | N/A     |       |
+-----------+-------+------+------+------+---------+-------+
```

You can add a row k4 to both rollup1 and rollup2 by adding the following 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;
```

When 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 you can see, the base table tbl1 also automatically added k4, k5 columns. That is, columns added to any rollup are automatically added to the Base table.

At the same time, columns that already exist in the Base table are not allowed to be added to Rollup. If you need to do this, you can re-create a Rollup with the new columns and then delete the original Rollup.

Modify Key column

Modifying the Key column of a table is done through the key keyword. Let’s take a look at an example below.

This usage is only for the key column of the duplicate key table

Source Schema :

```
+-----------+-------+-------------+------+------+---------+-------+
| IndexName | Field | Type        | Null | Key  | Default | Extra |
+-----------+-------+-------------+------+------+---------+-------+
| tbl1      | k1    | INT         | No   | true | N/A     |       |
|           | k2    | INT         | No   | true | N/A     |       |
|           | k3    | varchar(20) | No   | true | N/A     |       |
|           | k4    | INT         | No   | false| N/A     |       |
+-----------+-------+-------------+------+------+---------+-------+
```

The modification statement is as follows, we will change the degree of the k3 column to 50

```
alter table example_tbl modify column k3 varchar(50) key null comment 'to 50'
```

When done, the Schema becomes:

```
+-----------+-------+-------------+------+------+---------+-------+
| IndexName | Field | Type        | Null | Key  | Default | Extra |
+-----------+-------+-------------+------+------+---------+-------+
| tbl1      | k1    | INT         | No   | true | N/A     |       |
|           | k2    | INT         | No   | true | N/A     |       |
|           | k3    | varchar(50) | No   | true | N/A     |       |
|           | k4    | INT         | No   | false| N/A     |       |
+-----------+-------+-------------+------+------+---------+-------+
```

Because the Schema Chanage job is an asynchronous operation, only one Schema chanage job can be performed on the same table at the same time. To check the operation status of the job, you can use the following command

```
SHOW ALTER TABLE COLUMN\G;
```

2.5.1.1.4 Notice

- Only one Schema Change job can be running on a table at a time.
- Schema Change operation does not block import and query operations.
- The partition column and bucket column cannot be modified.
- If there is a value column aggregated by REPLACE in the schema, the Key column is not allowed to be deleted.

If the Key column is deleted, Doris cannot determine the value of the REPLACE column. All non-Key columns of the Unique data model table are REPLACE aggregated. * When adding a value column whose aggregation type is SUM or REPLACE, the default value of this column has no meaning to historical data.

Because the historical data has lost the detailed information, the default value cannot actually reflect the aggregated value. * When modifying the column type, fields other than Type need to be completed according to the information on the original column.

If you modify the column `k1 INT SUM NULL DEFAULT" 1 "` as type BIGINT, you need to execute the following command:

`ALTER TABLE tbl1 MODIFY COLUMN `k1` BIGINT SUM NULL DEFAULT "1";`

Note that in addition to the new column types, such as the aggregation mode, Nullable attributes, and default values must be completed according to the original information.

- Modifying column names, aggregation types, nullable attributes, default values, and column comments is not supported.

2.5.1.1.5 FAQ

- the execution speed of Schema Change

  At present, the execution speed of Schema Change is estimated to be about 10MB / s according to the worst efficiency. To be conservative, users can set the timeout for jobs based on this rate.

- Submit job error `Table xxx is not stable. ...`

  Schema Change can only be started when the table data is complete and unbalanced. If some data shard copies of the table are incomplete, or if some copies are undergoing an equalization operation, the submission is rejected. Whether the data shard copy is complete can be checked with the following command: `ADMIN SHOW REPLICA STATUS FROM tbl WHERE ↪ STATUS != "OK";`

  If a result is returned, there is a problem with the copy. These problems are usually fixed automatically by the system. You can also use the following commands to repair this table first:
  `ADMIN REPAIR TABLE tbl1;`

  You can check if there are running balancing tasks with the following command:

```
SHOW PROC "/cluster_balance/pending_tablets";
```

```
You can wait for the balancing task to complete, or temporarily disable the balancing operation
    ↪ with the following command:
```

```
ADMIN SET FRONTEND CONFIG ("disable_balance" = "true");
```

2.5.1.1.6 Configurations

FE Configurations

- `alter_table_timeout_second`: The default timeout for the job is 86400 seconds.

BE Configurations

- `alter_tablet_worker_count`: Number of threads used to perform historical data conversion on the BE side. The default is 3. If you want to speed up the Schema Change job, you can increase this parameter appropriately and restart the BE. But too many conversion threads can cause increased IO pressure and affect other operations. This thread is shared with the Rollup job.

2.5.1.1.7 More Help

For more detailed syntax and best practices used by Schema Change, see ALTER TABLE COLUMN command manual, you can also enter `HELP ALTER TABLE COLUMN` in the MySql client command line for more help information.

2.5.1.2 Replace Table

In version 0.14, Doris supports atomic replacement of two tables. This operation only applies to OLAP tables.

For partition level replacement operations, please refer to Temporary Partition Document

2.5.1.2.1 Syntax

```
ALTER TABLE [db.]tbl1 REPLACE WITH tbl2
[PROPERTIES('swap' = 'true')];
```

Replace table `tbl1` with table `tbl2`.

If the swap parameter is `true`, after replacement, the data in the table named `tbl1` is the data in the original `tbl2` table. The data in the table named `tbl2` is the data in the original table `tbl1`. That is, the data of the two tables are interchanged.

If the swap parameter is `false`, after replacement, the data in the table named `tbl1` is the data in the original `tbl2` table. The table named `tbl2` is dropped.

2.5.1.2.2 Principle

The replacement table function actually turns the following set of operations into an atomic operation.

Suppose you want to replace table A with table B, and swap is `true`, the operation is as follows:

1. Rename table B to table A.
2. Rename table A to table B.

If swap is `false`, the operation is as follows:

1. Drop table A.
2. Rename table B to table A.

2.5.1.2.3 Notice

1. The swap parameter defaults to `true`. That is, the replacement table operation is equivalent to the exchange of two table data.
2. If the swap parameter is set to `false`, the replaced table (table A) will be dropped and cannot be recovered.
3. The replacement operation can only occur between two OLAP tables, and the table structure of the two tables is not checked for consistency.
4. The replacement operation will not change the original permission settings. Because the permission check is based on the table name.

2.5.1.2.4 Best Practices

1. Atomic Overwrite Operation

   In some cases, the user wants to be able to rewrite the data of a certain table, but if it is dropped and then imported, there will be a period of time in which the data cannot be viewed. 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 replace the old table atomically through the replacement operation to achieve the goal. For partition level atomic overwrite operation, please refer to Temporary partition document

2.5.2 Doris Partition

2.5.2.1 Dynamic Partition

Dynamic partition is a new feature introduced in Doris version 0.12. It’s designed to manage partition’s Time-to-Life (TTL), reducing the burden on users.

At present, the functions of dynamically adding partitions and dynamically deleting partitions are realized.

Dynamic partitioning is only supported for Range partitions.

2.5.2.1.1 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.

2.5.2.1.2 Principle

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.

Through the dynamic partitioning feature, users can set the rules of dynamic partitioning when building tables. FE will start a background thread to create or delete partitions according to the rules specified by the user. Users can also change existing rules at runtime.

2.5.2.1.3 Usage

Establishment of tables

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",
    ...
)
```

Dynamic partition 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`

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

- `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 `https://en.wikipedia.org/wiki/` ↪ `List_of_tz_database_time_zones`.

- `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`

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`

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`.

  Note: 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, and MONTH". 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", end=3, start=-3, reserved_history_periods="[2020-06-01,2020-06-20],[2020-10-31,2020-`
`↪ .`

```
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`

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.

1. `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`, `-histoty_partition_num`);

2. `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:

1. 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
```

2. `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
```

3. `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
```

Notice

If some partitions between `dynamic_partition.start` and `dynamic_partition.end` are lost due to some unexpected circumstances when using dynamic partition, the lost partitions between the current time and `dynamic_partition.end` will be recreated, but the lost partitions between `dynamic_partition.start` and the current time will not be recreated.

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.

* Note: 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

2.5.2.1.4 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.

NOTICE: 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`.

2.5.2.1.5 Common problem

1. After creating the dynamic partition table, it prompts `Could not create table with dynamic partition when ↪ fe config dynamic_partition_enable is false`

   Because the main switch of dynamic partition, that is, the configuration of FE `dynamic_partition_enable` is false, the dynamic partition table cannot be created.

   At this time, please modify the FE configuration file, add a line `dynamic_partition_enable=true`, and restart FE. Or execute the command ADMIN SET FRONTEND CONFIG (“dynamic_partition_enable” = “true”) to turn on the dynamic partition switch.

2. Replica settings for dynamic partitions

   Dynamic partitions are automatically created by scheduling logic inside the system. When creating a partition automatically, the partition properties (including the number of replicas of the partition, etc.) are all prefixed with `dynamic_partition`, rather than the default properties of the table. for example:

```
CREATE TABLE tbl1 (
`k1` int,
`k2` date
)
PARTITION BY RANGE(k2)()
DISTRIBUTED BY HASH(k1) BUCKETS 3
PROPERTIES
(
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.end" = "3",
"dynamic_partition.prefix" = "p",
"dynamic_partition.buckets" = "32",
"dynamic_partition.replication_num" = "1",
"dynamic_partition.start" = "-3",
"replication_num" = "3"
);
```

```
In this example, no initial partition is created (partition definition in PARTITION BY clause is
    ↪ empty), and `DISTRIBUTED BY HASH(k1) BUCKETS 3`, `"replication_num" = "3"`, `"dynamic_
    ↪ partition is set. replication_num" = "1` and `"dynamic_partition.buckets" = "32"`.

We make the first two parameters the default parameters for the table, and the last two
    ↪ parameters the dynamic partition-specific parameters.

When the system automatically creates a partition, it will use the two configurations of bucket
    ↪ number 32 and replica number 1 (that is, parameters dedicated to dynamic partitions).
    ↪ Instead of the two configurations of bucket number 3 and replica number 3.

When a user manually adds a partition through the `ALTER TABLE tbl1 ADD PARTITION` statement, the
    ↪  two configurations of bucket number 3 and replica number 3 (that is, the default
    ↪ parameters of the table) will be used.

That is, dynamic partitioning uses a separate set of parameter settings. The table's default
    ↪ parameters are used only if no dynamic partition-specific parameters are set. as follows:
```

```
    CREATE TABLE tbl2 (
    `k1` int,
    `k2` date
    )
    PARTITION BY RANGE(k2)()
    DISTRIBUTED BY HASH(k1) BUCKETS 3
    PROPERTIES
    (
    "dynamic_partition.enable" = "true",
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.start" = "-3",
    "dynamic_partition.buckets" = "32",
    "replication_num" = "3"
    );
```

```
In this example, if `dynamic_partition.replication_num` is not specified separately, the default
    ↪ parameter of the table is used, which is `"replication_num" = "3"`.

And the following example:
```

```
CREATE TABLE tbl3 (
`k1` int,
`k2` date
)
PARTITION BY RANGE(k2)(
    PARTITION p1 VALUES LESS THAN ("2019-10-10")
)
DISTRIBUTED BY HASH(k1) BUCKETS 3
PROPERTIES
(
"dynamic_partition.enable" = "true",
"dynamic_partition.time_unit" = "DAY",
"dynamic_partition.end" = "3",
"dynamic_partition.prefix" = "p",
"dynamic_partition.start" = "-3",
"dynamic_partition.buckets" = "32",
"dynamic_partition.replication_num" = "1",
"replication_num" = "3"
);
```

```
In this example, there is a manually created partition p1. This partition will use the default
    ↪ settings for the table, which are 3 buckets and 3 replicas. The dynamic partitions
    ↪ automatically created by the subsequent system will still use the special parameters for
    ↪  dynamic partitions, that is, the number of buckets is 32 and the number of replicas is
    ↪ 1.
```

2.5.2.1.6 More Help

For more detailed syntax and best practices for using dynamic partitions, see SHOW DYNAMIC PARTITION Command manual, you can also enter `HELP ALTER TABLE` in the MySql client command line for more help information.

2.5.2.2 Temporary partition

Since version 0.12, Doris supports temporary partitioning.

A temporary partition belongs to a partitioned table. Only partitioned tables can create temporary partitions.

2.5.2.2.1 Rules

- The partition columns of the temporary partition is 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.

2.5.2.2.2 Supported operations

The temporary partition supports add, delete, and replace operations.

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 operation:

- 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.

   Here are some examples:

   - 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"), ("3", "shanghai")) "), ("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.
```

Some instructions for the replacement operation:

- After the partition is replaced successfully, the replaced partition will be deleted and cannot be recovered.

2.5.2.2.3 Load and query of temporary partitions

Users can load data into temporary partitions or specify temporary partitions for querying.

1. Load temporary partition

   The syntax for specifying a temporary partition is slightly different depending on the load method. Here is a simple illustration through an example:

```
    INSERT INTO tbl TEMPORARY PARTITION (tp1, tp2, ...) SELECT ....
```

```
    curl --location-trusted -u root: -H "label: 123" -H "temporary_partition: 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
    (...);
```

2. Query the temporary partition

```
SELECT ... FROM
tbl1 TEMPORARY PARTITION (tp1, tp2, ...)
JOIN
tbl2 TEMPORARY PARTITION (tp1, tp2, ...)
ON ...
WHERE ...;
```

2.5.2.2.4 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.

2.5.2.2.5 Best Practices

1. Atomic overwrite

In some cases, the user wants to be able to rewrite the data of a certain partition, but if it is dropped first and then loaded, there will be a period of time when the data cannot be seen. At this moment, the user can first create a corresponding temporary partition, load new data into the temporary partition, and then replace the original partition atomically through the `REPLACE` operation to achieve the purpose. For atomic overwrite operations of non-partitioned tables, please refer to Replace Table Document

2. Modify the number of buckets

   In some cases, the user used an inappropriate number of buckets when creating a partition. The user can first create a temporary partition corresponding to the partition range and specify a new number of buckets. Then use the `INSERT INTO` command to load the data of the formal partition into the temporary partition. Through the replacement operation, the original partition is replaced atomically to achieve the purpose.

3. Merge or split partitions

   In some cases, users want to modify the range of partitions, such as merging two partitions, or splitting a large partition into multiple smaller partitions. Then the user can first create temporary partitions corresponding to the merged or divided range, and then load the data of the formal partition into the temporary partition through the `INSERT INTO` command. Through the replacement operation, the original partition is replaced atomically to achieve the purpose.

2.5.3 Data Cache

2.5.3.1 Partition Cache

2.5.3.1.1 Demand scenario

In most data analysis scenarios, write less and read more. Data is written once and read frequently. For example, the dimensions and indicators involved in a report are calculated at one time in the early morning, but there are hundreds or even thousands of times every 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
- Kanban for complex charts, complex Dashboard or large-screen applications, the data comes from multiple tables, each page has dozens of queries, although each query is only tens of milliseconds, but the overall query time will be in a few seconds
- Trend analysis, the query for a given date range, the indicators are displayed by day, 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 and a wide range of queries, and the query time often takes tens of seconds
- User repeated query, if the product does not have an anti-reload mechanism, the user repeatedly refreshes the page due to hand error or other reasons, resulting in a large number of repeated SQL submissions

In the above four scenarios, the solution at the application layer is to put the query results in Redis, update the cache periodically or manually refresh the cache by the user, but this solution has the following problems:

- Data inconsistency, unable to perceive the update of data, causing users to often see old data
- Low hit rate, cache the entire query result, if the data is written in real time, the cache is frequently invalidated, the hit rate is low and the system load is heavy
- Additional cost, the introduction of external cache components will bring system complexity and increase additional costs

2.5.3.1.2 Solution

This partitioned caching strategy can solve the above problems, giving priority to ensuring data consistency. On this basis, the cache granularity is refined and the hit rate is improved. Therefore, it has the following characteristics:

- Users do not need to worry about data consistency, cache invalidation is controlled by version, and the cached data is consistent with the data queried from BE
- No additional components and costs, cached results are stored in BE' s memory, 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

2.5.3.1.3 SQLCache

SQLCache stores and retrieves the cache according to 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 changes, such as SQL changes, such as 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, use the latest updated partition ID and the latest version number. If one of the tables is updated, the partition ID or version number will be different, and the cache will also not be hit.

SQLCache is more suitable for T+1 update scenarios. Data is updated in the early morning. The results obtained from the BE for the first query are put into the cache, and subsequent identical queries are obtained from the cache. Real-time update data can also be used, but there may be a problem of low hit rate. You can refer to the following PartitionCache.

2.5.3.1.4 PartitionCache

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. Split into read-only partitions and updatable partitions, read-only partitions are cached, and update partitions are not cached

As above, query the number of users per day in the last 7 days, such as partitioning by date, the data is only written to the partition of the day, and the data of other partitions other than the day is fixed. Under the same query SQL, query a certain part that does not update Partition indicators are fixed. As follows, the number of users in the first 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 is from the partition because it is constantly being written that day.

Therefore, when querying N days of data, the data is updated on the most recent D days. Every day is only a query with a different date range and a similar query. Only D partitions need to be queried, and the other parts are from the cache, which can effectively reduce the cluster load and reduce query time.

```
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 | //First from partition, subsequent from cache
| 2020-03-09 |               25 | //from partition
+------------+-----------------+
7 rows in set (0.02 sec)
```

In PartitionCache, the first-level key of the cache is the 128-bit MD5 signature of the SQL after the partition condition is removed. 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, and the auxiliary 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 |
+------------+-----------------+
```

1. SQL and data to get data from BE 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 |
+------------+-----------------+
```

1. 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 |
+------------+-----------------+
```

1. 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.

Partition fields can also be other fields, but need to ensure that only a small number of partition updates.

Some restrictions

- Only OlapTable is supported, other tables such as MySQL have no version information and cannot sense whether the data is updated
- Only supports grouping by partition field, does not support grouping by other fields, grouping by other fields, the grouped data may be updated, which will cause the cache to be invalid
- Only the first half of the result set, the second half of the result set and all cache hits are supported, and the result set is not supported to be divided into several parts by the cached data

2.5.3.1.5 How to use

Enable SQLCache

Make sure cache_enable_sql_mode=true in fe.conf (default is true)

```
vim fe/conf/fe.conf
cache_enable_sql_mode=true
```

Setting variables in MySQL command line

```
MySQL [(none)]> set [global] enable_sql_cache=true;
```

Note: global is a global variable, not referring to the current session variable

Enable PartitionCache

Make sure cache_enable_partition_mode=true in fe.conf (default is true)

```
vim fe/conf/fe.conf
cache_enable_partition_mode=true
```

Setting variables in MySQL command line

```
MySQL [(none)]> set [global] enable_partition_cache=true;
```

If two caching strategies are enabled at the same time, the following parameters need to be paid attention to:

```
cache_last_version_interval_second=900
```

If the interval between the latest version of the partition 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, press PartitionCache data.

Monitoring

FE monitoring items:

```
query_table          //Number of tables in Query
query_olap_table     //Number of Olap tables in Query
cache_mode_sql       //Identify the number of queries whose cache mode is sql
cache_hit_sql        //The number of Cache hits by Query with mode sql
query_mode_partition //Identify the number of queries whose cache mode is Partition
cache_hit_partition  //Number of queries hit by Partition
partition_all        //All partitions scanned in Query
partition_hit        //Number of partitions hit by Cache

Cache hit ratio = (cache_hit_sql + cache_hit_partition) / query_olap_table
Partition hit ratio = partition_hit / partition_all
```

BE’ s monitoring items:

```
query_cache_memory_total_byte     //Cache memory size
query_query_cache_sql_total_count //Number of SQL in Cache
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, the Query Percentile and other indicators in the Query statistics from Grafana, and adjust the Cache parameters to achieve business goals.

Optimization parameters

The configuration item cache_result_max_row_count of FE, the maximum number of rows in the cache for the query result set, can be adjusted according to the actual situation, but it is recommended not to set it too large to avoid taking up too much memory, and the result set exceeding this size will not be cached.

```
vim fe/conf/fe.conf
cache_result_max_row_count=3000
```

The maximum number of partitions in BE cache_max_partition_count 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 to a larger value and modify it at the same time. Parameter of cache_result_max_row_count.

```
vim be/conf/be.conf
cache_max_partition_count=1024
```

The cache memory setting in BE consists of two parameters, query_cache_max_size and query_cache_elasticity_size (in MB). If the memory exceeds query_cache_max_size + cache_elasticity_size, it will start to clean up and control the memory to 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.

```
query_cache_max_size_mb=256
query_cache_elasticity_size_mb=128
```

Calculation method:

If 10000 queries are cached, each query caches 1000 rows, each row is 128 bytes, distributed on 10 BEs, then each BE requires about 128M memory (10000 * 1000 * 128/10).

2.5.3.1.6 Unfinished Matters

- Can the data of T+1 also be cached by Partition? Currently not supported
- Similar SQL, 2 indicators were queried before, but now 3 indicators are queried. Can the cache of 2 indicators be used? Not currently supported
- Partition by date, but need to aggregate data by week dimension, is PartitionCache available? Not currently supported

2.5.4 AutoBucket

DISTRIBUTED BY ··· BUCKETS auto

Users often set inappropriate buckets, leading to various problems. For now, it only works for olap tables

2.5.5 Implementation

In the past, when creating buckets, you had to set the number of buckets manually, but the automatic bucket projection 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. First, 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 Starting from the principle, it becomes easy to understand the detailed logic of the automatic bucket imputation function. First look at the initial bucketing

First, use the value of estimate_partition_size divided by 5 (calculated as a 5-to-1 data compression ratio in text format into Doris) to obtain the following result.

- (, 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 each 50G of disk capacity counts as 1. Then the rule for calculating M is M = number of BE nodes (one disk block size / 50GB) number of disk blocks For example, if there are 3 BEs, each with 4 500GB disks, then M = 3 (500GB / 50GB) 4 = 120
3. Calculation logic to get the final number of buckets. First 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.

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
```

As you can see, the detailed logic matches the principle. 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.

1. 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)

2. 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.

2.5.6 Description

When autobucket is enabled, the schema you see in `show create table` is also BUCKETS AUTO. If you want to see the exact number of buckets, you can do so by `show partitions from ${table};`.

2.5.7 Broker

Broker is an optional process in the Doris cluster. It is mainly used to support Doris to read and write files or directories on remote storage. Now support:

- Apache HDFS
- Aliyun OSS
- Tencent Cloud CHDFS
- Tencent Cloud GFS (since 1.2.0)
- Huawei Cloud OBS (since 1.2.0)
- Amazon S3
- JuiceFS (since 2.0.0)

Broker provides services through an RPC service port. It is a stateless JVM process that is responsible for encapsulating some POSIX-like file operations for read and write operations on remote storage, such as open, pred, pwrite, and so on. In addition, the Broker does not record any other information, so the connection information, file information, permission information, and so on stored remotely need to be passed to the Broker process in the RPC call through parameters in order for the Broker to read and write files correctly .

Broker only acts as a data channel and does not participate in any calculations, so it takes up less memory. Usually one or more Broker processes are deployed in a Doris system. And the same type of Broker will form a group and set a ** Broker name **.

Broker’s position in the Doris system architecture is as follows:

```
+----+   +----+
| FE |   | BE |
+-^--+   +--^-+
  |         |
  |         |
+-v---------v-+
|   Broker    |
+------^------+
       |
       |
+------v------+
|HDFS/BOS/AFS |
+-------------+
```

This document mainly introduces the parameters that Broker needs when accessing different remote storages, such as connection information, authorization information, and so on.

2.5.7.1 Supported Storage System

Different types of brokers support different storage systems.

1. Community HDFS

    - Support simple authentication access

- Support kerberos authentication access
- Support HDFS HA mode access

2. Object storage

- All object stores that support the S3 protocol

2.5.7.2 Function provided by Broker

1. Broker Load
2. Export
3. Backup

2.5.7.3 Broker Information

Broker information includes two parts: ** Broker name ** and ** Certification information **. The general syntax is as follows:

```
WITH BROKER "broker_name"
(
    "username" = "xxx",
    "password" = "yyy",
    "other_prop" = "prop_value",
    ...
);
```

2.5.7.3.1 Broker Name

Usually the user needs to specify an existing Broker Name through the `WITH BROKER" broker_name "` clause in the operation command. Broker Name is a name that the user specifies when adding a Broker process through the ALTER SYSTEM ADD BROKER command. A name usually corresponds to one or more broker processes. Doris selects available broker processes based on the name. You can use the `SHOW BROKER` command to view the Brokers that currently exist in the cluster.

Note: Broker Name is just a user-defined name and does not represent the type of Broker.

2.5.7.3.2 Certification Information

Different broker types and different access methods need to provide different authentication information. Authentication information is usually provided as a Key-Value in the Property Map after `WITH BROKER" broker_name "`.

Community HDFS

1. Simple Authentication

   Simple authentication means that Hadoop configures `hadoop.security.authentication` to`simple`.

   Use system users to access HDFS. Or add in the environment variable started by Broker: HADOOP_USER_NAME.

```
    (
        "username" = "user",
        "password" = ""
    );
```

```
Just leave the password blank.
```

2. Kerberos Authentication

   The authentication method needs to provide the following information::

   - `hadoop.security.authentication`: Specify the authentication method as kerberos.
   - `kerberos_principal`: Specify the principal of kerberos.
   - `kerberos_keytab`: Specify the path to the keytab file for kerberos. The file must be an absolute path to a file on the server where the broker process is located. And can be accessed by the Broker process.
   - `kerberos_keytab_content`: Specify the content of the keytab file in kerberos after base64 encoding. You can choose one of these with `kerberos_keytab` configuration.

   Examples are as follows:

```
    (
        "hadoop.security.authentication" = "kerberos",
        "kerberos_principal" = "doris@YOUR.COM",
        "kerberos_keytab" = "/home/doris/my.keytab"
    )
```

```
    (
        "hadoop.security.authentication" = "kerberos",
        "kerberos_principal" = "doris@YOUR.COM",
        "kerberos_keytab_content" = "ASDOWHDLAWIDJHWLDKSALDJSDIWALD"
    )
```

```
If Kerberos authentication is used, the [krb5.conf](https://web.mit.edu/kerberos/krb5-1.12/doc/
    ↪ admin/conf_files/krb5_conf.html) file is required when deploying the Broker process.
The krb5.conf file contains Kerberos configuration information, Normally, you should install your
    ↪  krb5.conf file in the directory /etc. You can override the default location by setting
    ↪ the environment variable KRB5_CONFIG.
An example of the contents of the krb5.conf file 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
    }
```

3. HDFS HA Mode

   This configuration is used to access HDFS clusters deployed in HA mode.

   - `dfs.nameservices`: Specify the name of the hdfs service, custom, such as "dfs.nameservices" = "my_ha".
   - `dfs.ha.namenodes.xxx`: Custom namenode names. Multiple names are separated by commas, where xxx is the custom name in `dfs.nameservices`, such as" dfs.ha.namenodes.my_ha "=" my_nn ".
   - `dfs.namenode.rpc-address.xxx.nn`: Specify the rpc address information of namenode, Where nn represents the name of the namenode configured in `dfs.ha.namenodes.xxx`, such as: "dfs.namenode.rpc-address.my_ha.my_nn" = "host:port".
   - `dfs.client.failover.proxy.provider.[nameservice ID]`: Specify the provider for the client to connect to the namenode. The default is: org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider.

   Examples are as follows:

```
(
    "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"
)
```

```
The HA mode can be combined with the previous two authentication methods for cluster access. If
    ↪ you access HA HDFS with simple authentication:
```

```
(
    "username"="user",
    "password"="passwd",
    "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"
)
```

The configuration for accessing the HDFS cluster can be written to the hdfs-site.xml file. When users use the Broker process to read data from the HDFS cluster, they only need to fill in the cluster file path and authentication information.

Tencent Cloud CHDFS

Same as Apache HDFS

Aliyun OSS

```
(
    "fs.oss.accessKeyId" = "",
    "fs.oss.accessKeySecret" = "",
    "fs.oss.endpoint" = ""
)
```

Huawei Cloud OBS

```
(
    "fs.obs.access.key" = "xx",
    "fs.obs.secret.key" = "xx",
    "fs.obs.endpoint" = "xx"
)
```

Amazon S3

```
(
    "fs.s3a.access.key" = "xx",
    "fs.s3a.secret.key" = "xx",
    "fs.s3a.endpoint" = "xx"
)
```

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"
)
```

2.5.8 Best Practice

2.5.8.1 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.

2.5.8.1.1 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.

```
     ┌────┐
     │Sort│
     └────┘
        │
  ┌──────────────┐
  │Aggregation   │
  └──────────────┘
        │
     ┌────┐
     │Join│
     └────┘
    ┌───┴───┐
┌──────┐ ┌──────┐
│Scan-1│ │Scan-2│
└──────┘ └──────┘
```

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.

```
        ┌────┐
        │Sort│
        │F1  │
        └────┘
           │
     ┌──────────────┐
     │Aggregation   │
     │F1            │
     └──────────────┘
           │
        ┌────┐
        │Join│
        │F1  │
        └────┘
    ┌─────┴─────┐
┌──────┐ ┌────────────┐
│Scan-1│ │ExchangeNode│
│F1    │ │F1          │
└──────┘ └────────────┘
                │
```

```
          ┌───────────────┐
          │DataStreamDink  │
          │F2              │
          └───────────────┘
                │
            ┌──────┐
            │Scan-2│
            │F2    │
            └──────┘
```

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.

2.5.8.1.2 View query plan

You can view the execution plan of a SQL through the following three commands.

- `EXPLAIN GRAPH select ...;` OR `DESC GRAPH select ...;`
- `EXPLAIN select ...;`
- `EXPLAIN VERBOSE select ...;`

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]|       |[6: EXCHANGE]|         |
|    |[Fragment: 2]|       |[Fragment: 2]|         |
|    └-------------┘       └-------------┘         |
|           |                      |                |
| ┌-------------------┐ ┌-------------------┐       |
| │[5: DataStreamSink]│ │[6: DataStreamSink]│       |
| │[Fragment: 0]      │ │[Fragment: 1]      │       |
| │STREAM DATA SINK   │ │STREAM DATA SINK   │       |
| │  EXCHANGE ID: 05  │ │  EXCHANGE ID: 06  │       |
| │  HASH_PARTITIONED │ │  HASH_PARTITIONED │       |
| └-------------------┘ └-------------------┘       |
|           │                      │                |
|  ┌-----------------┐    ┌-----------------┐       |
|  │[0: OlapScanNode]│    │[1: OlapScanNode]│       |
|  │[Fragment: 0]    │    │[Fragment: 1]    │       |
|  │TABLE: tbl1      │    │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.

2.5.8.1.3 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   │
                    └──────────────┘
                           ┌┘
                ┌─────────────────────┐
                │[8: AGGREGATION_NODE]│
                │Fragment: 1          │
                └─────────────────────┘
                           └┐
                  ┌──────────────────┐
                  │[7: EXCHANGE_NODE]│
```

```
               │Fragment: 1       │
               └------------------┘
                         │
             ┌----------------------┐
             │[7: DataStreamSender] │
             │Fragment: 2           │
             │MaxActiveTime: 3.505ms│
             └----------------------┘
                        ┌┘
             ┌---------------------┐
             │[3: AGGREGATION_NODE]│
             │Fragment: 2          │
             └---------------------┘
                        │
              ┌-------------------┐
              │[2: HASH_JOIN_NODE]│
              │Fragment: 2        │
              └-------------------┘
           ┌------------┴------------┐
 ┌------------------┐      ┌------------------┐
 │[5: EXCHANGE_NODE]│      │[6: EXCHANGE_NODE]│
 │Fragment: 2       │      │Fragment: 2       │
 └------------------┘      └------------------┘
           │                         │
┌---------------------┐ ┌------------------------┐
│[5: DataStreamSender]│ │[6: DataStreamSender]   │
│Fragment: 4          │ │Fragment: 3             │
│MaxActiveTime: 1.87ms│ │MaxActiveTime: 636.767us│
└---------------------┘ └------------------------┘
           │                        ┌┘
 ┌-------------------┐    ┌-------------------┐
 │[0: OLAP_SCAN_NODE]│    │[1: OLAP_SCAN_NODE]│
 │Fragment: 4        │    │Fragment: 3        │
 └-------------------┘    └-------------------┘
           │                        │
    ┌-------------┐          ┌-------------┐
    │[OlapScanner]│          │[OlapScanner]│
    │Fragment: 4  │          │Fragment: 3  │
    └-------------┘          └-------------┘
           │                        │
  ┌-----------------┐      ┌-----------------┐
  │[SegmentIterator]│      │[SegmentIterator]│
  │Fragment: 4      │      │Fragment: 3      │
  └-----------------┘      └-----------------┘
```

```
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.

1. 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.

2.5.8.2 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.

2.5.8.2.1 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.

2.5.8.2.2 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.

2.5.8.3 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.

2.5.8.3.1 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

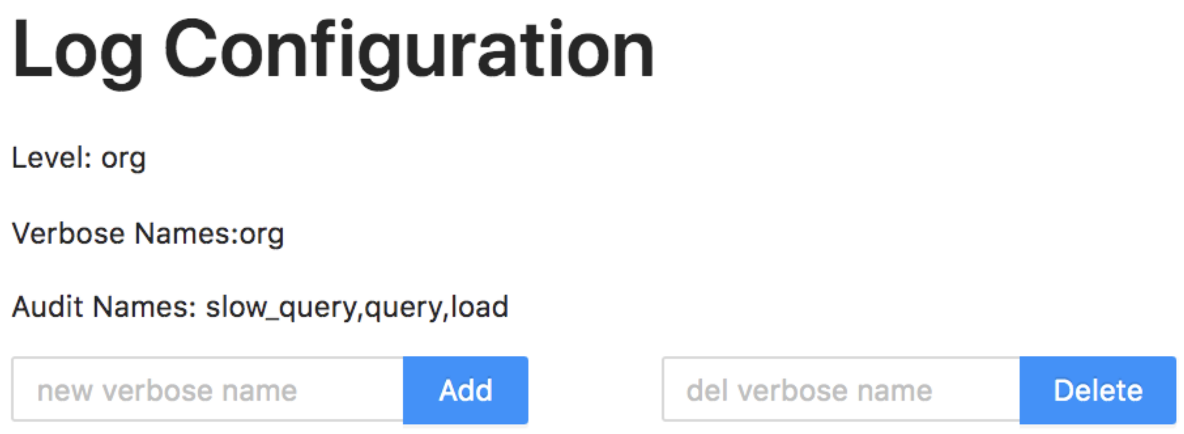
Add the configuration item `sys_log_verbose_modules` to fe.conf. An example is as follows:

```
# 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.


Log Configuration

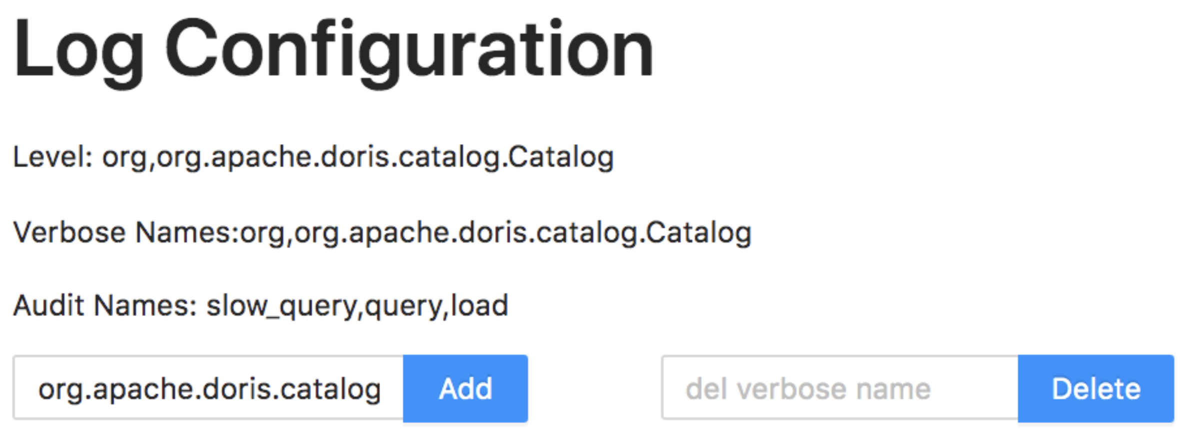
Level: org

Verbose Names:org

Audit Names: slow_query,query,load

Figure 10: 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:


Log Configuration

Level: org,org.apache.doris.catalog.Catalog

Verbose Names:org,org.apache.doris.catalog.Catalog

Audit Names: slow_query,query,load

Figure 11: 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.

2.5.8.3.2 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.

2.5.8.4 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 two compaction strategies for tuning:

2.5.8.4.1 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)

2.5.8.4.2 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_threshold_segment_num` 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.

2.5.9 Resource management

In order to save the compute and storage resources in the Doris cluster, Doris needs to reference to some other external resources to do the related work. such as spark/GPU for query, HDFS/S3 for external storage, spark/MapReduce for ETL, connect to external storage by ODBC driver. Therefore, Doris need a resource management mechanism to manage these external resources.

2.5.9.1 Fundamental Concept

A resource contains basic information such as name and type. The name is globally unique. Different types of resources contain different attributes. Please refer to the introduction of each resource for details.

The creation and deletion of resources can only be performed by users own `admin` permission. One resource belongs to the entire Doris cluster. Users with `admin` permission can assign permission of resource to other users. Please refer to `HELP GRANT` or doris document.

2.5.9.2 Operation Of Resource

There are three main commands for resource management: `create resource`, `drop resource` and `show resources`. They are to create, delete and check resources. The specific syntax of these three commands can be viewed by executing `help CMD` after MySQL client connects to Doris.

1. CREATE RESOURCE

   This statement is used to create a resource. For details, please refer to CREATE RESOURCE.

2. DROP RESOURCE

   This command can delete an existing resource. For details, see DROP RESOURCE.

3. SHOW RESOURCES

   This command can view the resources that the user has permission to use. For details, see SHOW RESOURCES.

2.5.9.3 Resources Supported

Currently, Doris can support

- Spark resource: do ETL work
- ODBC resource: query and import data from external tables

The following shows how the two resources are used.

2.5.9.3.1 Spark

Parameter

Spark Parameters:

`spark.master`: required, currently supported yarn, spark://host:port.

`spark.submit.deployMode`: The deployment mode of spark. required. It supports 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: http://spark.apache.org/docs/latest/configuration.html

If spark is used for ETL, also need to specify the following parameters:

`working_dir`: Directory used by ETL. Spark is required when used as an ETL resource. For example: hdfs://host:port/tmp/doris.

`broker`: The name of broker. Is required when spark be used as ETL resource. You need to use the `ALTER SYSTEM ADD BROKER` command to complete the configuration in advance.

- `broker.property_key`: When the broker reads the intermediate file generated by ETL, it needs the specified authentication information.

Example

Create a spark resource named `spark0`in the 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"
);
```

2.5.9.3.2 ODBC

Parameter

ODBC Parameters:

`type`: Required, must be `odbc_catalog`. As the type identifier of resource.

`user`: The user name of the external table, required.

`password`: The user password of the external table, required.

`host`: The ip address of the external table, required.

`port`: The port of the external table, required.

`odbc_type`: Indicates the type of external table. Currently, Doris supports `MySQL` and `Oracle`. In the future, it may support more databases. The ODBC external table referring to the resource is required. The old MySQL external table referring to the resource is optional.

`driver`: Indicates the driver dynamic library used by the ODBC external table. The ODBC external table referring to the resource is required. The old MySQL external table referring to the resource is optional.

For the usage of ODBC resource, please refer to ODBC of Doris

Example

Create the ODBC resource of Oracle, named `oracle_odbc`.

```
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"
);
```

2.5.10 Orthogonal BITMAP calculation

2.5.10.1 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.

2.5.10.2 User guide

1. Create a table and add hid column to represent bitmap column value ID range as hash bucket column
2. Usage scenarios

2.5.10.2.1 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

2.5.10.2.2 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

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

2.5.10.2.3 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);
```

2.5.11 Approximate deduplication using HLL

2.5.11.1 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.

2.5.11.2 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 = 2ˆk_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 = 2ˆ12

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 ≠ 2ˆ6 . 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.

2.5.11.3 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

2.5.11.4 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

2.5.11.4.1 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"
);
```

2.5.11.4.2 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)
    )
 );
```

2.5.11.4.3 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)
```

2.5.12 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.

2.5.12.1 Variable setting and viewing

2.5.12.1.1 View

All or specified variables can be viewed via `SHOW VARIABLES [LIKE 'xxx'];`. Such as:

```
SHOW VARIABLES;
SHOW VARIABLES LIKE '%time_zone%';
```

2.5.12.1.2 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');
```

2.5.12.1.3 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.

2.5.12.2 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.

- `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.

- `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.

Translated with www.DeepL.com/Translator (free version)

- `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 5 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 for compatibility with MySQL clients. No practical effect.

- `system_time_zone`

  Displays the current system time zone. Cannot be changed.

- `time_zone`

  Used to set the time zone of the current session. The time zone has an effect on the results of certain time functions. For the time zone, see here.

- `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 sucess-fully.

- `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. 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.

- `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`

  Use a fixed replica to query. If use_fix_replica is 1, the smallest one is used, if use_fix_replica is 2, the second smallest one is used, and so on. The default value is -1, which means it is not enabled.

- `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. 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_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.

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.

2.5.13 Time zone

Doris supports multiple time zone settings

2.5.13.1 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.

2.5.13.2 Basic concepts

There are multiple time zone related parameters in Doris

- `system_time_zone`:

When the server starts, it will be set automatically according to the time zone set by the machine, which cannot be modified after setting.

- `time_zone`:

Server current time zone, set it at session level or global level.

2.5.13.3 Specific operations

1. `SHOW VARIABLES LIKE '% time_zone%'`

   View the current time zone related configuration

2. `SET time_zone = 'Asia/Shanghai'`

   This command can set the session level time zone, which will fail after disconnection.

3. `SET global time_zone = 'Asia/Shanghai'`

   This command can set time zone parameters at the global level. The FE will persist the parameters and will not fail when the connection is disconnected.

2.5.13.3.1 Impact of time zone

Time zone setting affects the display and storage of time zone sensitive values.

It includes the values displayed by time functions such as `NOW()` or `CURTIME()`, as well as the time values in `SHOW LOAD` and `SHOW BACKENDS` statements.

However, it does not affect the `LESS THAN VALUE` of the time-type partition column in the `CREATE TABLE` statement, nor does it affect the display of values stored as `DATE/DATETIME` type.

Functions affected by time zone:

- `FROM_UNIXTIME`: Given a UTC timestamp, return the date and time of the specified time zone, such as `FROM_UNIXTIME(0)`, return the CST time zone: `1970-01-08:00`.
- `UNIX_TIMESTAMP`: Given a specified time zone date and time, return UTC timestamp, such as CST time zone `UNIX_ ↪ TIMESTAMP('1970-01 08:00:00')`, return 0.
- `CURTIME`: Returns the datetime of specified time zone.
- `NOW`: Returns the specified date and time of specified time zone.
- `CONVERT_TZ`: Converts a date and time from one specified time zone to another.

2.5.13.4 Restrictions

Time zone values can be given in several formats, case-insensitive:

- A string representing UTC offset, such as ‘+10:00’ or ‘-6:00’.
- Standard time zone formats, such as “Asia/Shanghai”, “America/Los_Angeles”
- Abbreviated time zone formats such as MET and CTT are not supported. Because the abbreviated time zone is ambiguous in different scenarios, it is not recommended to use it.
- In order to be compatible with Doris and support CST abbreviated time zone, CST will be internally transferred to “Asia/Shanghai”, which is Chinese standard time zone.

2.5.13.5 Time zone format list

List of TZ database time zones

2.5.14 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.

2.5.14.1 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.

2.5.14.2 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.

2.5.14.3 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;`.

2.5.14.3.1 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"
    );
```

2.5.14.3.2 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;
```

2.5.14.3.3 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");
```

2.5.14.4 Implementation details

2.5.14.4.1 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.

2.5.14.4.2 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.

2.5.14.5 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.

2.5.14.6 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.

2.5.14.7 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.

2.5.15 Cold hot separation

2.5.15.1 Demand scenario

A big usage scenario in the future is similar to the es log storage. In the log scenario, the data will be cut by date. Many data are cold data, with few queries. Therefore, the storage cost of such data needs to be reduced. From the perspective of saving storage costs 1. The price of ordinary cloud disks of cloud manufacturers is higher than that of object storage 2. In the actual online use of the doris cluster, the utilization rate of ordinary cloud disks cannot reach 100% 3. Cloud disk is not paid on demand, but object storage can be paid on demand 4. High availability based on ordinary cloud disks requires multiple replicas, and a replica migration is required for a replica exception. This problem does not exist when data is placed on the object store, because the object store is shared。

2.5.15.2 Solution

Set the freeze time on the partition level to indicate how long the partition will be frozen, and define the location of remote storage stored after the freeze. On the be, the daemon thread will periodically determine whether the table needs to be frozen. If it does, it will upload the data to s3.

The cold and hot separation supports all doris functions, but only places some data on object storage to save costs without sacrificing functions. Therefore, it has the following characteristics:

- When cold data is stored on object storage, users need not worry about data consistency and data security
- Flexible freeze policy, cooling remote storage property can be applied to table and partition levels
- Users query data without paying attention to the data distribution location. If the data is not local, they will pull the data on the object and cache it to be local
- Optimization of replica clone. If the stored data is on the object, the replica clone does not need to pull the stored data locally
- Remote object space recycling recycler. If the table and partition are deleted, or the space is wasted due to abnormal conditions in the cold and hot separation process, the recycler thread will periodically recycle, saving storage resources
- Cache optimization, which caches the accessed cold data to be local, achieving the query performance of non cold and hot separation
- Be thread pool optimization, distinguish whether the data source is local or object storage, and prevent the delay of reading objects from affecting query performance
- newly created materialized view would inherit storage policy from it’s base table’s correspoding partition

2.5.15.3 Storage policy

The storage policy is the entry to use the cold and hot separation function. Users only need to associate a storage policy with a table or partition during table creation or doris use. that is, they can use the cold and hot separation function.

When creating an S3 RESOURCE, the S3 remote link verification will be performed to ensure that the RESOURCE is created correctly.

In addition, fe configuration needs to be added: `enable_storage_policy=true`

For example:

```
CREATE RESOURCE "remote_s3"
PROPERTIES
(
    "type" = "s3",
    "AWS_ENDPOINT" = "bj.s3.com",
    "AWS_REGION" = "bj",
    "AWS_BUCKET" = "test-bucket",
    "AWS_ROOT_PATH" = "path/to/root",
    "AWS_ACCESS_KEY" = "bbb",
    "AWS_SECRET_KEY" = "aaaa",
    "AWS_MAX_CONNECTIONS" = "50",
    "AWS_REQUEST_TIMEOUT_MS" = "3000",
    "AWS_CONNECTION_TIMEOUT_MS" = "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(
    "storage_policy" = "test_policy"
);
```

Or for an existing table, associate the storage policy

```
ALTER TABLE create_table_not_have_policy set ("storage_policy" = "test_policy");
```

Or associate a storage policy with an existing partition

```
ALTER TABLE create_table_partition MODIFY PARTITION (*) SET("storage_policy"="test_policy");
```

For details, please refer to the resource, policy, create table, alter and other documents in the docs directory

2.5.15.3.1 Some restrictions

- A single table or a single partition can only be associated with one storage policy. After association, the storage policy cannot be dropped
- The object information associated with the storage policy does not support modifying the data storage path information, such as bucket, endpoint, and root_Path and other information
- Currently, the storage policy only supports creation and modification, not deletion

2.5.15.4 Show size of objects occupied by cold data

1. Through show proc '/backends', you can view the size of each object being uploaded to, and the RemoteUsedCapacity item.

2. Through show tables from tableName, you can view the object size occupied by each table, and the RemoteDataSize item.

2.5.15.5 cold data cache

As above, cold data introduces the cache in order to optimize query performance. After the first hit after cooling, Doris will reload the cooled data to be's local disk. The cache has the following characteristics: - The cache is actually stored on the be local disk

and does not occupy memory. - the cache can limit expansion and clean up data through LRU - The be parameter `file_cache_`
↪ `alive_time_sec` can set the maximum storage time of the cache data after it has not been accessed. The default is 604800, which is one week. - The be parameter `file_cache_max_size_per_disk` can set the disk size occupied by the cache. Once this setting is exceeded, the cache that has not been accessed for the longest time will be deleted. The default is 0, means no limit to the size, unit: byte. - The be parameter `file_cache_type` is optional `sub_file_cache` (segment the remote file for local caching) and `whole_file_cache` (the entire remote file for local caching), the default is "", means no file is cached, please set it when caching is required this parameter.

2.5.15.6 Unfinished Matters

- After the data is frozen, there are new data updates or imports, etc. The compression has not been processed at present.
- The schema change operation after the data is frozen is not supported at present.

2.5.16 Compute Node

<! – 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 expressed or implied. See the License for the Specific language governing permissions and limitations Under the License. – >

2.5.16.1 Scenario

At present, Doris is a typical Share-Nothing architecture, which achieves very high performance by binding data and computing resources in the same node. With the continuous improvement of the performance for the Doris computing engine, more and more users have begun to use Doris to directly query data on data lake. This is a Share-Disk scenario that data is often stored on the remote HDFS/S3, and calculated in Doris. Doris will get the data through the network, and then completes the computation in memory. For these two mixed loads in one cluster, current Doris architecture will appear some disadvantages: 1. Poor resource isolation, the response requirements of these two loads are different, and the hybrid deployment will have mutual effects. 2. Poor disk usage, the data lake query only needs the computing resources, while doris binding the storage and computing and we have to expand them together, and cause a low utilization rate for disk. 3. Poor expansion efficiency, when the cluster is expanded, Doris will start the migration of Tablet data, and this process will take a lot of time. And the data lake query load has obvious peaks and valleys, it need hourly flexibility.

2.5.16.2 solution

Implement a BE node role specially used for federated computing named `Compute node`. `Compute node` is used to handle remote federated queries such as this query of data lake. The original BE node type is called `hybrid node`, and this type of node can not only execute SQL query, but also handle tablet data storage. And the `Compute node` only can execute SQL query, it have no data on node.

With the computing node, the cluster deployment topology will also change: - the `hybrid` node is used for the data calculation of the OLAP type table, the node is expanded according to the storage demand - the `computing` node is used for the external computing, and this node is expanded according to the query load.

Since the compute node has no storage, the compute node can be deployed on an HDD disk machine with other workload or on a container.

2.5.16.3 Usage of ComputeNode

2.5.16.3.1 Configure

Add configuration items to BE's configuration file `be.conf`:

```
be_node_role = computation
```

This defualt value of this is `mix`, and this is original BE node type. After setting to `computation`, the node is a computing node.

You can see the value of the' NodeRole 'field through the `show backend\G` command. If it is' mix', it is a mixed node, and if it is' computation', it is a computing node

```
*************************** 1. row ***************************
              BackendId: 10010
                Cluster: default_cluster
                     IP: 10.248.181.219
          HeartbeatPort: 9050
                 BePort: 9060
               HttpPort: 8040
               BrpcPort: 8060
          LastStartTime: 2022-11-30 23:01:40
          LastHeartbeat: 2022-12-05 15:01:18
                  Alive: true
   SystemDecommissioned: false
  ClusterDecommissioned: false
              TabletNum: 753
       DataUsedCapacity: 1.955 GB
          AvailCapacity: 202.987 GB
          TotalCapacity: 491.153 GB
                UsedPct: 58.67 %
         MaxDiskUsedPct: 58.67 %
     RemoteUsedCapacity: 0.000
                    Tag: {"location" : "default"}
                 ErrMsg:
                Version: doris-0.0.0-trunk-80baca264
                 Status: {"lastSuccessReportTabletsTime":"2022-12-05 15:00:38","
                    ↪ lastStreamLoadTime":-1,"isQueryDisabled":false,"isLoadDisabled":false}
HeartbeatFailureCounter: 0
               NodeRole: computation
```

2.5.16.3.2 Usage

Add configuration items in fe.conf

```
prefer_compute_node_for_external_table=true
min_backend_num_for_external_table=3
```

For parameter description, please refer to: FE configuration item

When using the MultiCatalog function when querying, the query will be dispatched to the computing node first.

2.5.16.3.3 some restrictions

- Compute nodes are controlled by configuration items, so do not configure mixed type nodes, modify the configuration to compute nodes.

2.5.16.4 TODO

- Computational spillover: Doris inner table query, when the cluster load is high, the upper layer (outside TableScan) operator can be scheduled to the compute node.
- Graceful offline:
- When the compute node goes offline, the new task of the task is automatically scheduled to online nodes
- the node go offline after all the old tasks on the node are completed
- when the old task cannot be completed on time, the task can kill by itself

2.5.17 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.

2.5.17.1 Grammar

```
LATERAL VIEW  generator_function ( expression [, ...] ) table_identifier AS column_identifier [,
    ↪ ...]
```

2.5.17.2 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.

2.5.17.3 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)
```

2.6 Query Acceleration

2.6.1 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.

2.6.1.1 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.

2.6.1.2 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.

2.6.1.3 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`.

2.6.1.4 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.

2.6.1.4.1 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
```

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, as this may cause conflicts between multiple materialized views and cause query hit failures. (Of course, these possible problems can be verified in the test environment)

2.6.1.4.2 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)

2.6.1.4.3 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.

2.6.1.4.4 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.

2.6.1.4.5 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.

2.6.1.4.6 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

2.6.1.4.7 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

2.6.1.4.8 Cancel Create materialized view

```
CANCEL ALTER TABLE MATERIALIZED VIEW FROM db_name.table_name
```

2.6.1.5 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.

2.6.1.6 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.
```

2.6.1.7 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.
```

2.6.1.8 Best Practice 4

In `Doris 2.0`, we have made some enhancements to the expressions supported by the materialized view. This example will mainly reflect the support for various expressions of the new version of the materialized view.

1. Create a base table and insert some data.

```
create table d_table (
   k1 int null,
   k2 int not null,
   k3 bigint null,
   k4 varchar(100) 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.

```
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),day(k4) from d_table;
```

3. Use some queries to test if the materialized view was successfully hit.

```
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; // hit kymd
select year(k4)+month(k4)+day(k4) from d_table where year(k4) = 2020; // hit kymd
```

2.6.1.9 Limitations

1. The parameter of the aggregate function of the materialized view does not support the expression only supports a single column, for example: sum(a+b) does not support. (Supported after 2.0)

2. If the conditional column of the delete statement exists in the materialized view, the delete operation cannot be performed. If you must delete data, you need to delete the materialized view before deleting the data.
3. Too many materialized views on a single table will affect the efficiency of importing: When importing data, the materialized view and base table data are updated synchronously. If a table has more than 10 materialized view tables, it may cause the import speed to be very high. slow. This is the same as a single import needs to import 10 tables at the same time.
4. The same column with different aggregate functions cannot appear in a materialized view at the same time. For example, select sum(a), min(a) from table are not supported. (Supported after 2.0)
5. For the Unique Key data model, the materialized view can only change the column order and cannot play the role of aggregation. Therefore, in the Unique Key model, it is not possible to perform coarse-grained aggregation operations on the data by creating a materialized view.

2.6.1.10 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

2.6.1.11 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.

2.6.2 Join Optimization

2.6.2.1 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。

2.6.2.1.1 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.

2.6.2.1.2 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.

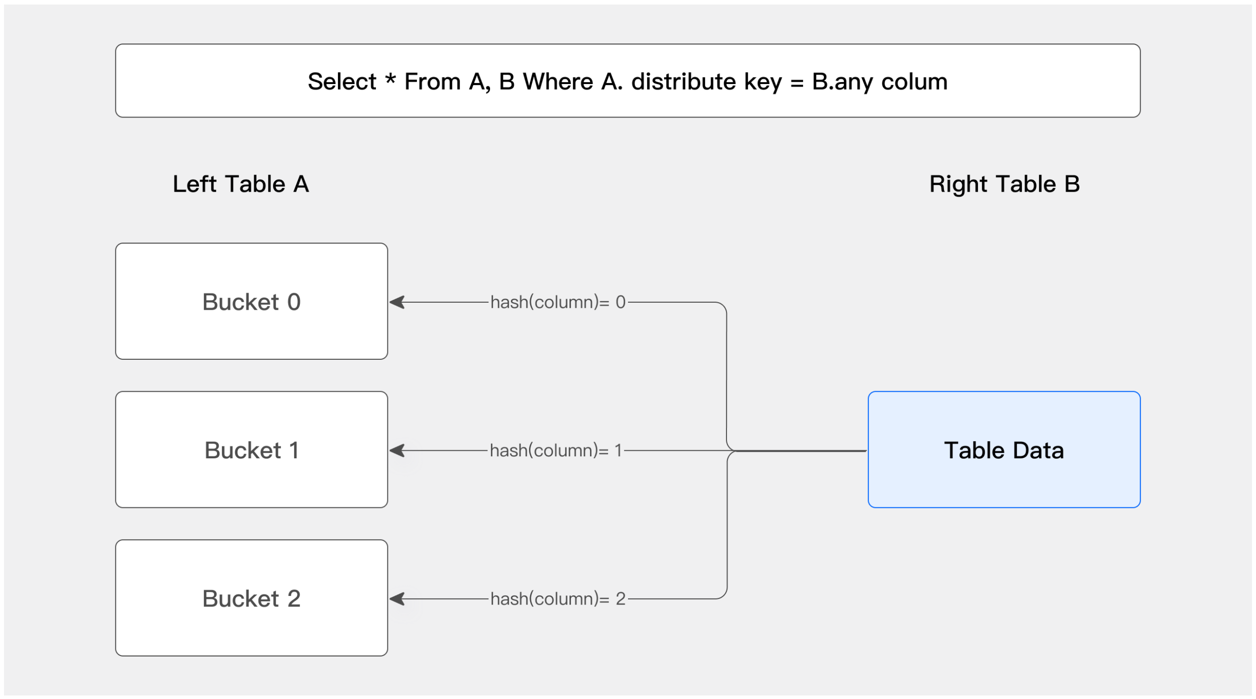


Figure 12: Shuffle Join.png

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 Colorate 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.

2.6.2.1.3 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 Colorate 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`。

2.6.2.1.4 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 Colorate 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 colorate table, the data distribution rules of each partition are determined. So the bucket shuffle join can perform better on the colorate table.

2.6.2.2 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.

2.6.2.2.1 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.

2.6.2.2.2 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.

2.6.2.2.3 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.

2.6.2.2.4 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)

2.6.2.2.5 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.

2.6.2.2.6 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.

2.6.2.3 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.

2.6.2.3.1 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.

2.6.2.3.2 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.

2.6.2.3.3 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 102400

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.
    - Currently, bitmap filter is only supported in vectorization engine.

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
```

2.6.2.3.4 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.

2.6.2.4 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.

2.6.2.4.1 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).

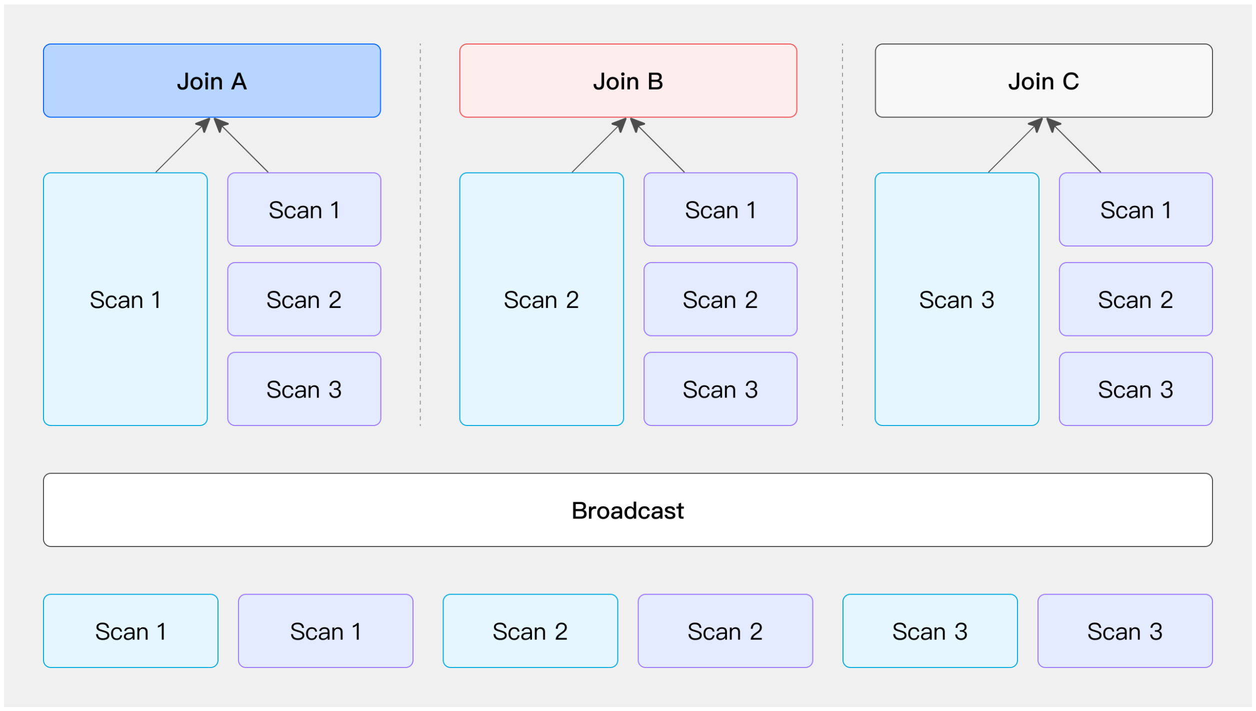


Figure 13: BroadCast Join

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(R) + T(N), but it can only support Hash Join, because it also calculates buckets according to the conditions of Join.

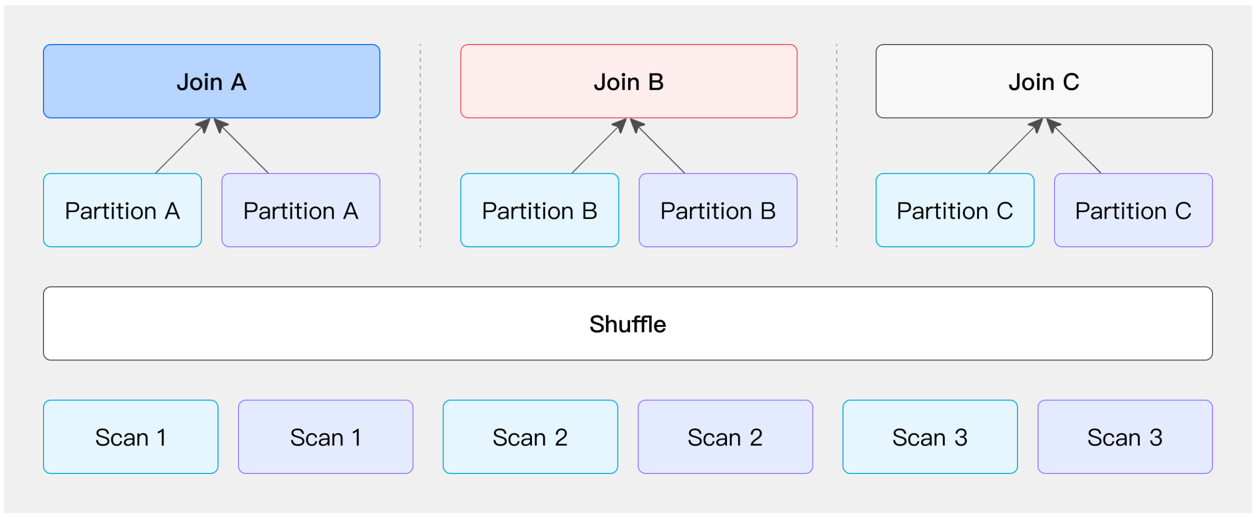


Figure 14: Shuffle Join

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.

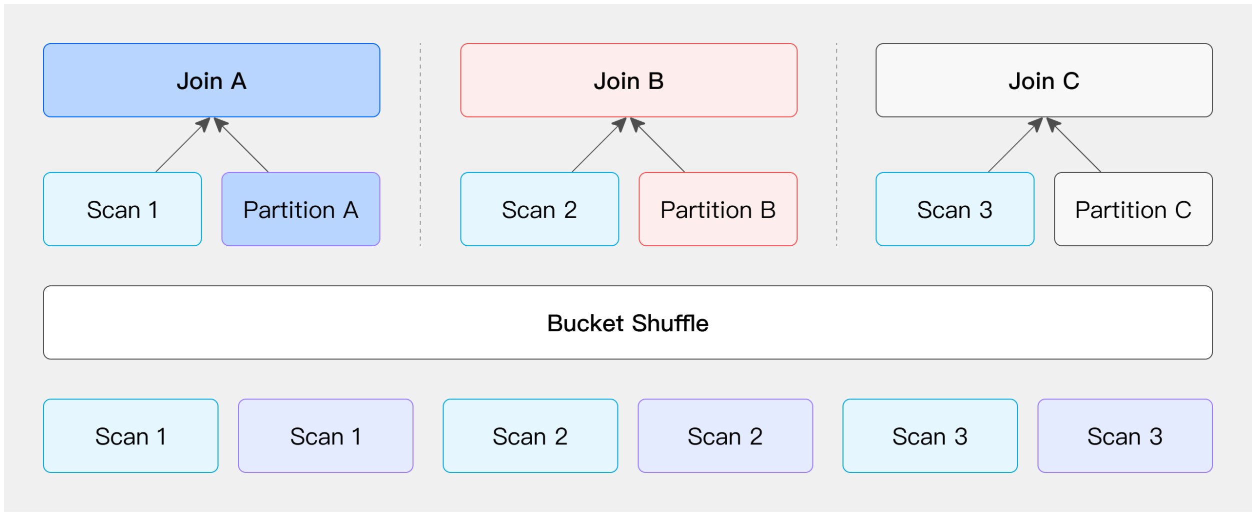


Figure 15: Bucket Shuffle Join

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.

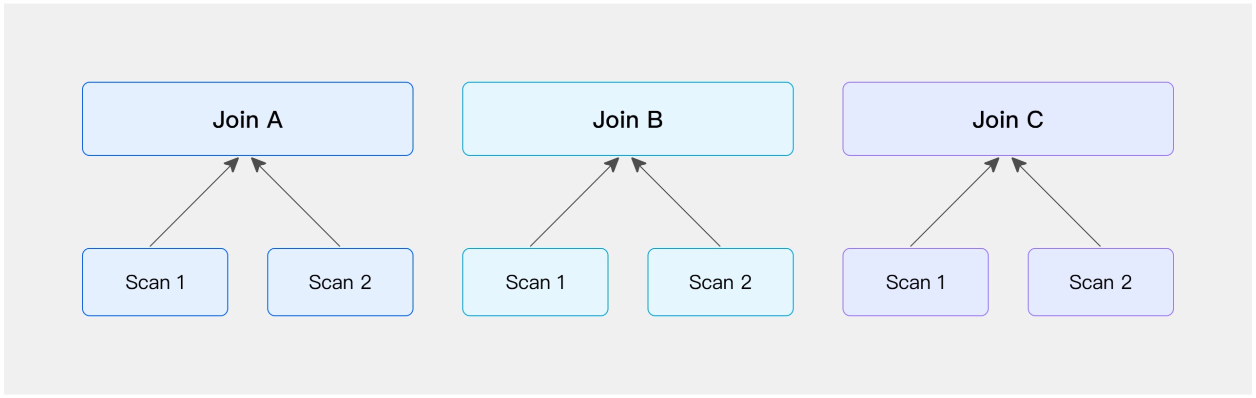


Figure 16: Colocation Join

   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
BroadCast	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.

2.6.2.4.2 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 Broad-Cast. 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.

2.6.2.4.3 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.

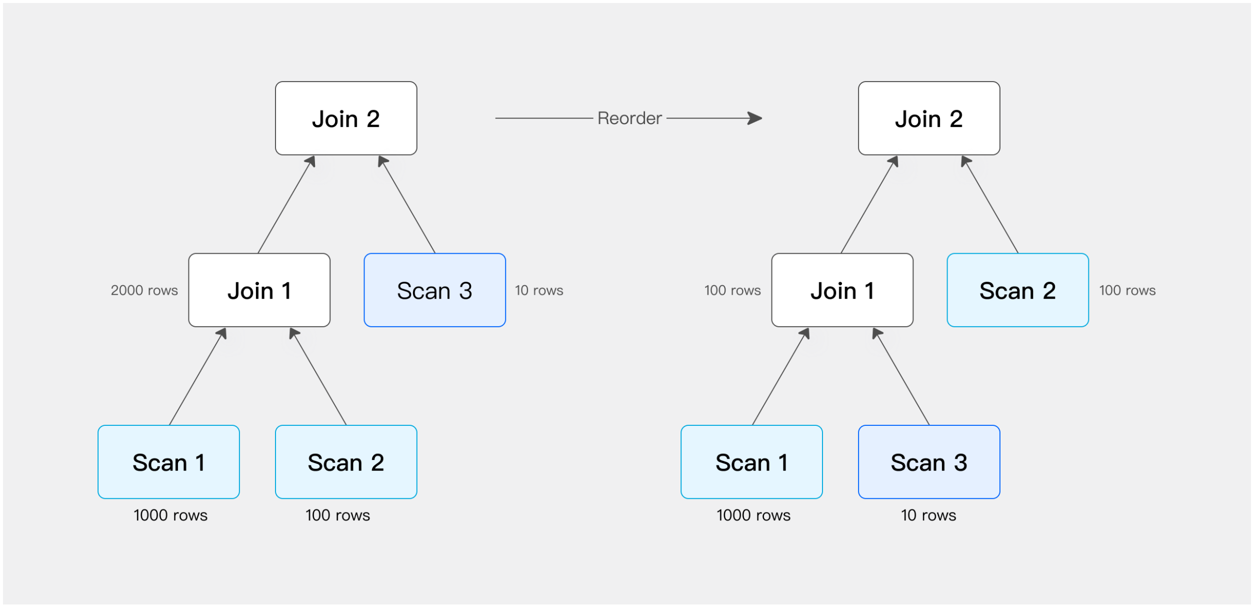


Figure 17: Join Reorder

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.

2.6.2.4.4 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.

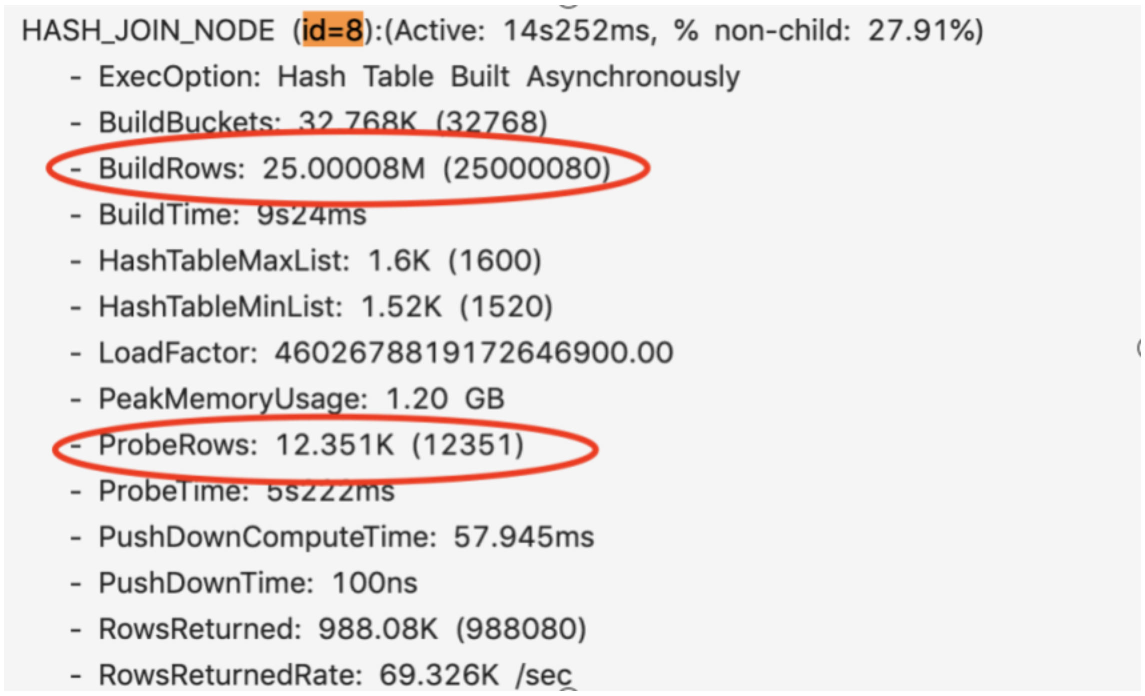
The above 4 steps basically complete a standard Join tuning process, and then it is to actually query and verify it to see what the effect is.

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.

2.6.2.4.5 Optimization case practice

Case 1

A four-table join query, through Profile, found that the second join took a long time, taking 14 seconds.



```
HASH_JOIN_NODE (id=8):(Active: 14s252ms, % non-child: 27.91%)
   - ExecOption: Hash Table Built Asynchronously
   - BuildBuckets: 32.768K (32768)
   - BuildRows: 25.00008M (25000080)
   - BuildTime: 9s24ms
   - HashTableMaxList: 1.6K (1600)
   - HashTableMinList: 1.52K (1520)
   - LoadFactor: 4602678819172646900.00
   - PeakMemoryUsage: 1.20 GB
   - ProbeRows: 12.351K (12351)
   - ProbeTime: 5s222ms
   - PushDownComputeTime: 57.945ms
   - PushDownTime: 100ns
   - RowsReturned: 988.08K (988080)
   - RowsReturnedRate: 69.326K /sec
```

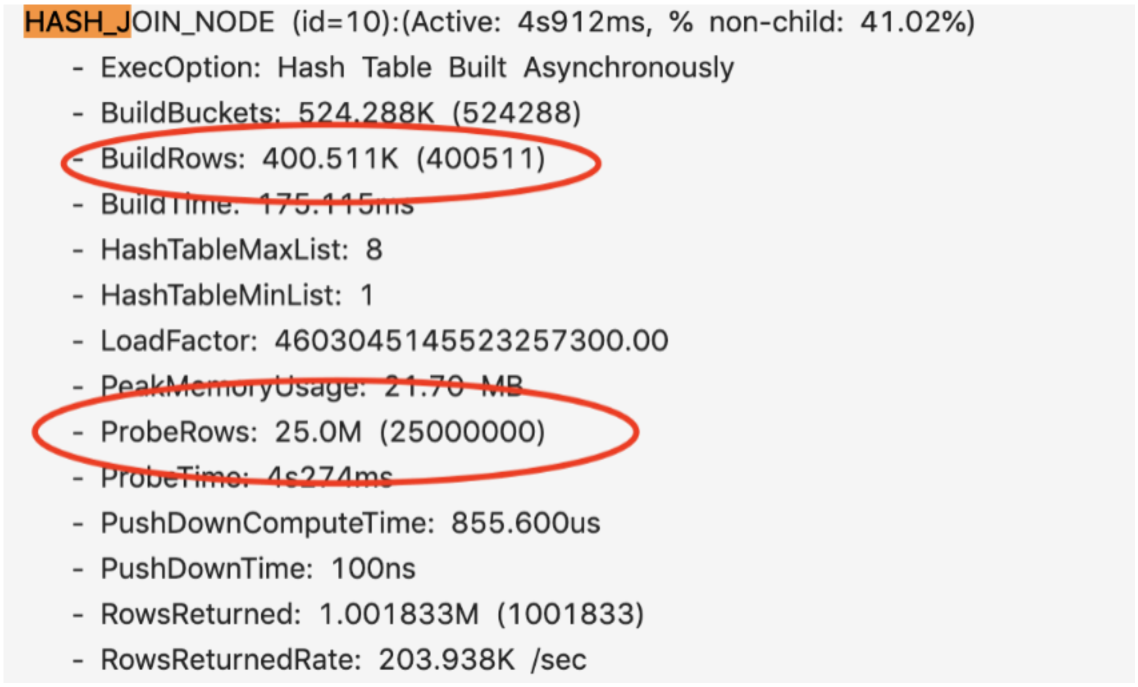
Figure 18: image-20220523153600797

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.

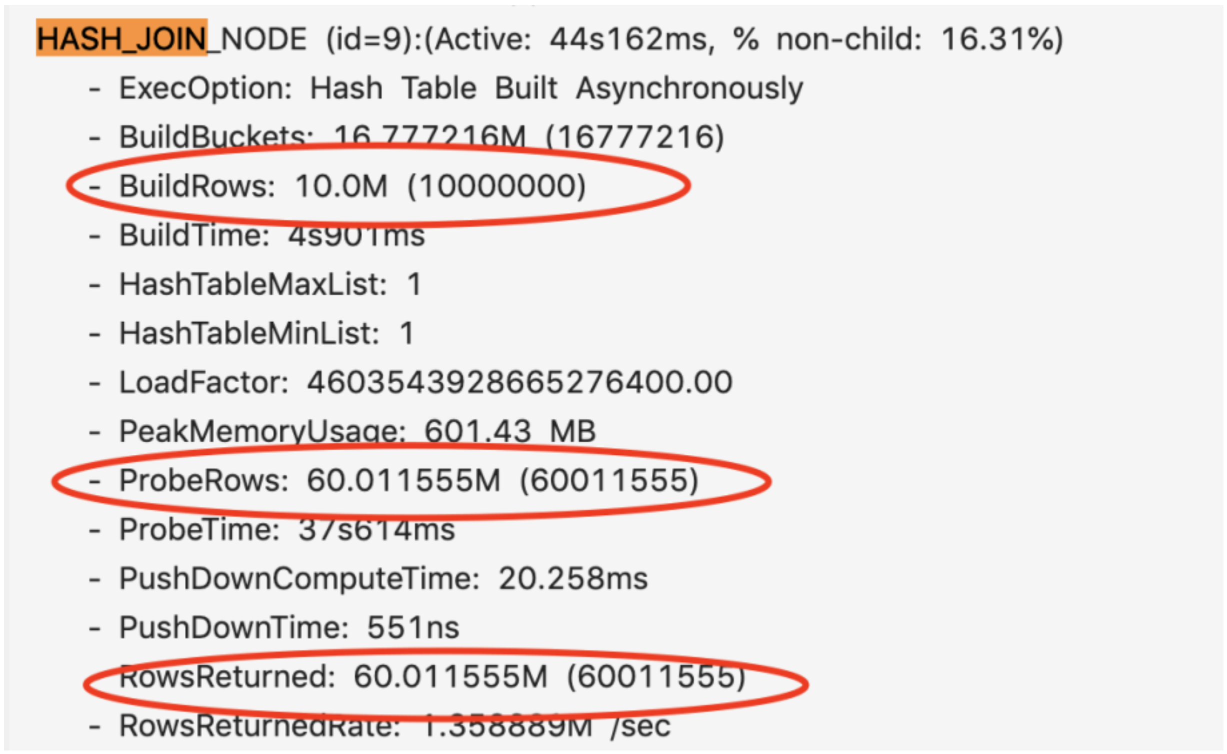


```
HASH_JOIN_NODE (id=10):(Active: 4s912ms, % non-child: 41.02%)
   - ExecOption: Hash Table Built Asynchronously
   - BuildBuckets: 524.288K (524288)
   - BuildRows: 400.511K (400511)
   - BuildTime: 175.115ms
   - HashTableMaxList: 8
   - HashTableMinList: 1
   - LoadFactor: 4603045145523257300.00
   - PeakMemoryUsage: 21.70 MB
   - ProbeRows: 25.0M (25000000)
   - ProbeTime: 4s274ms
   - PushDownComputeTime: 855.600us
   - PushDownTime: 100ns
   - RowsReturned: 1.001833M (1001833)
   - RowsReturnedRate: 203.938K /sec
```

Figure 19: image-20220523153757607

Case 2

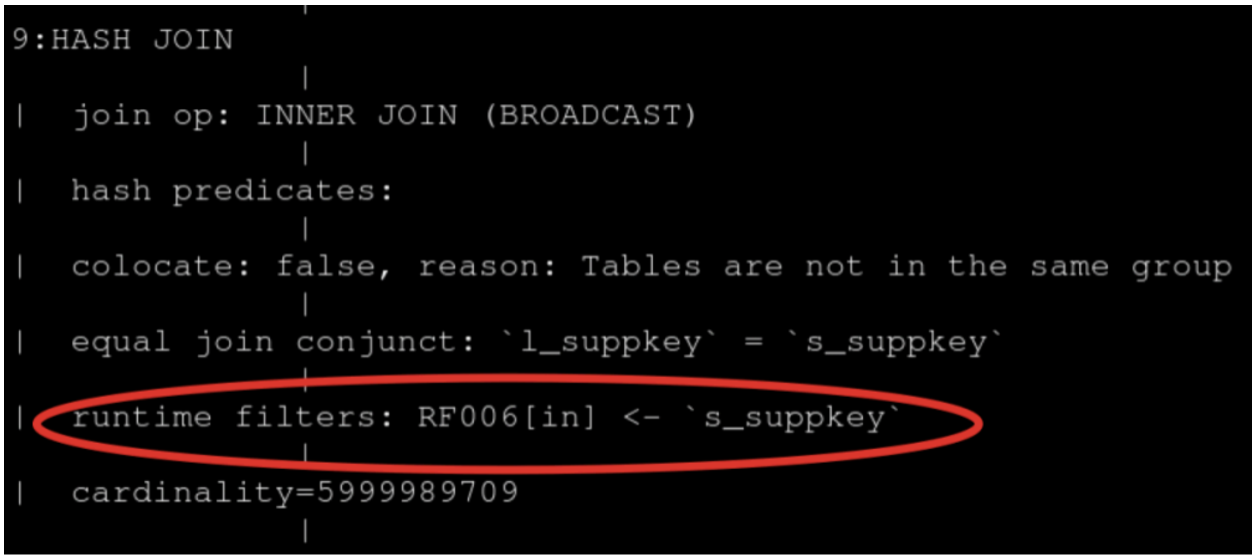
There is a slow query. After viewing the Profile, the entire Join node takes about 44 seconds. Its right table has 10 million, the left table has 60 million, and the final returned result is only 60 million.



```
HASH_JOIN_NODE (id=9):(Active: 44s162ms, % non-child: 16.31%)
   - ExecOption: Hash Table Built Asynchronously
   - BuildBuckets: 16.777216M (16777216)
   - BuildRows: 10.0M (10000000)
   - BuildTime: 4s901ms
   - HashTableMaxList: 1
   - HashTableMinList: 1
   - LoadFactor: 4603543928665276400.00
   - PeakMemoryUsage: 601.43 MB
   - ProbeRows: 60.011555M (60011555)
   - ProbeTime: 37s614ms
   - PushDownComputeTime: 20.258ms
   - PushDownTime: 551ns
   - RowsReturned: 60.011555M (60011555)
   - RowsReturnedRate: 1.358889M /sec
```

Figure 20: 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.



```
9:HASH JOIN
|  join op: INNER JOIN (BROADCAST)
|  hash predicates:
|  colocate: false, reason: Tables are not in the same group
|  equal join conjunct: `l_suppkey` = `s_suppkey`
|  runtime filters: RF006[in] <- `s_suppkey`
|  cardinality=5999989709
```

Figure 21: 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.

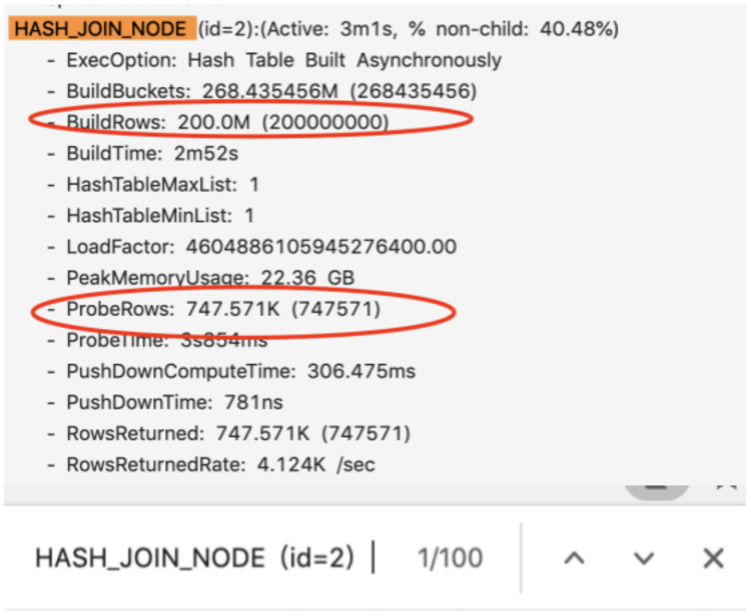
This is changed to BloomFilter, and the 60 million pieces of data in the left table have filtered 59 million pieces. Basically, 99% of the data is filtered out, and this effect is very significant. The query has also dropped from the original 44 seconds to 13 seconds, and the performance has been improved 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
    1_partkey = p_partkey
    and 1_shipdate >= date '1997-06-01'
    and 1 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.

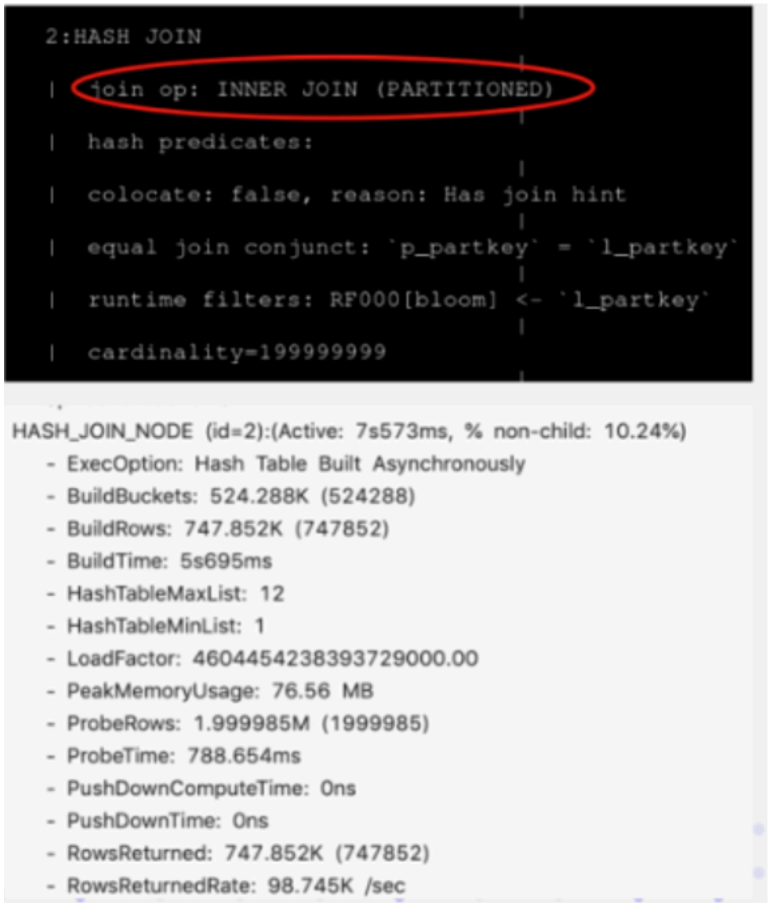


```
HASH_JOIN_NODE (id=2):(Active: 3m1s, % non-child: 40.48%)
   - ExecOption: Hash Table Built Asynchronously
   - BuildBuckets: 268.435456M (268435456)
   - BuildRows: 200.0M (200000000)
   - BuildTime: 2m52s
   - HashTableMaxList: 1
   - HashTableMinList: 1
   - LoadFactor: 4604886105945276400.00
   - PeakMemoryUsage: 22.36 GB
   - ProbeRows: 747.571K (747571)
   - ProbeTime: 3s854ms
   - PushDownComputeTime: 306.475ms
   - PushDownTime: 781ns
   - RowsReturned: 747.571K (747571)
   - RowsReturnedRate: 4.124K /sec
```

Figure 22: 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.



```
2:HASH JOIN
|  join op: INNER JOIN (PARTITIONED)
|  hash predicates:
|  colocate: false, reason: Has join hint
|  equal join conjunct: `p_partkey` = `l_partkey`
|  runtime filters: RF000[bloom] <- `l_partkey`
|  cardinality=199999999
```

```
HASH_JOIN_NODE (id=2):(Active: 7s573ms, % non-child: 10.24%)
   - ExecOption: Hash Table Built Asynchronously
   - BuildBuckets: 524.288K (524288)
   - BuildRows: 747.852K (747852)
   - BuildTime: 5s695ms
   - HashTableMaxList: 12
   - HashTableMinList: 1
   - LoadFactor: 4604454238393729000.00
   - PeakMemoryUsage: 76.56 MB
   - ProbeRows: 1.999985M (1999985)
   - ProbeTime: 788.654ms
   - PushDownComputeTime: 0ns
   - PushDownTime: 0ns
   - RowsReturned: 747.852K (747852)
   - RowsReturnedRate: 98.745K /sec
```

Figure 23: image-20220523160915229

2.6.2.4.6 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.

2.6.3 Slow Query Analysis

2.6.3.1 Get Profile

2.6.3.1.1 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.

2.6.3.1.2 Request

`HTTP://FE_IP:HTTP_PORT` GET `/API/Profile`

2.6.3.1.3 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)
```

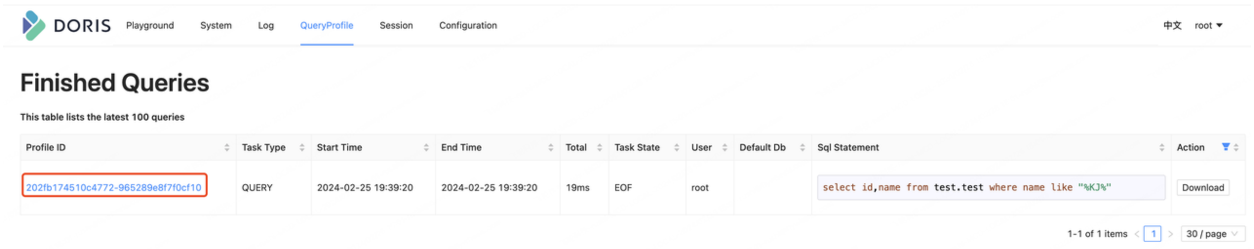
2. Execute the corresponding query

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 | ZN1nqzBRSl1rTrr99rnX1aplxhRuOUTLw6so7rzjlRQ317gTPxh0dHljmrARDJ|
|                           | jH7FjRkJW9c7YuUBmWikq7eNgmFKJPreWirDrGrFzUYH4eP6kDtSA3UTnNIIj |
+---------------------------+----------------------------------------------------------------+
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.

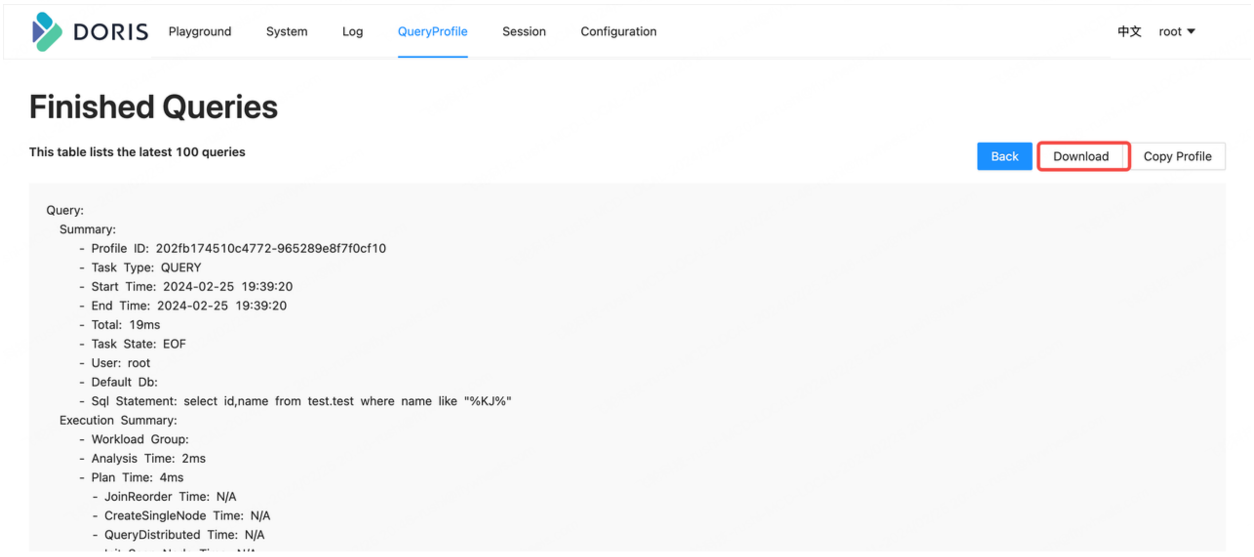


Finished Queries

This table lists the latest 100 queries

Profile ID	Task Type	Start Time	End Time	Total	Task State	User	Default Db	Sql Statement	Action
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%"	Download

1-1 of 1 items 1 30 / page



Finished Queries

This table lists the latest 100 queries

```
Query:
  Summary:
     - Profile ID: 202fb174510c4772-965289e8f7f0cf10
     - Task Type: QUERY
     - Start Time: 2024-02-25 19:39:20
     - End Time: 2024-02-25 19:39:20
     - Total: 19ms
     - Task State: EOF
     - User: root
     - Default Db:
     - Sql Statement: select id,name from test.test where name like "%KJ%"
  Execution Summary:
     - Workload Group:
     - Analysis Time: 2ms
     - Plan Time: 4ms
       - JoinReorder Time: N/A
       - CreateSingleNode Time: N/A
       - QueryDistributed Time: N/A
```

2.6.3.1.4 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 "%Ru0%" |
| 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=1
    ↪ b0bb22689734d30-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 \ \n, which is inconvenient to analyze. You can replace \ \n with \n 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: 0ms
        - Write Result Time: 0ms
        - 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}
```

2.7 Lakehouse

2.7.1 Multi Catalog

2.7.1.1 Multi Catalog

Multi-Catalog is a newly added feature in Doris 1.2.0. It allows Doris to interface with external catalogs more conveniently and thus increases the data lake analysis and federated 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 were large amounts of databases or tables in the external catalog, users would need to create mappings to them one by one, which could be a heavy workload.

With the advent of 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. The currently supported external catalogs include:

1. Hive
2. Iceberg
3. Hudi
4. Elasticsearch
5. JDBC

Multi-Catalog works as an additional and enhanced external table connection method. It helps users conduct multi-catalog federated queries quickly.

2.7.1.1.1 Basic Concepts

1. 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.

2. External Catalog

   Users can create an External Catalog using the CREATE CATALOG command, and view the existing Catalogs via the SHOW CATALOGS command.

3. Switch Catalog

   After login, 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 of switching catalog:

```
    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.

Doris only supports read-only access to data in External Catalogs currently.
```

4. Delete Catalog

   Databases and tables in External Catalogs are for read only, but External Catalogs are deletable via the DROP CATALOG command. (The Internal Catalog cannot be deleted.)

   The deletion only means to remove the mapping in Doris to the corresponding catalog. It doesn' t change the external catalog itself by all means.

5. Resource

   Resource is a set of configurations. Users can create a Resource using the CREATE RESOURCE command, and then apply this Resource for a newly created Catalog. One Resource can be reused for multiple Catalogs.

2.7.1.1.2 Examples

Connect to Hive

The followings are the instruction on how to connect to a Hive catalog using the Catalog feature.

For more information about Hive, please see Hive.

1. Create Catalog

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004'
);
```

```
> Syntax Help: [CREATE CATALOG](https://doris.apache.org/docs/dev/sql-manual/sql-reference/Data-
    ↪ Definition-Statements/Create/CREATE-CATALOG/)
```

2. View Catalog

   View existing Catalogs via the `SHOW CATALOGS` command:

```
mysql> SHOW CATALOGS;
+-----------+-------------+----------+
| CatalogId | CatalogName | Type     |
+-----------+-------------+----------+
|     10024 | hive        | hms      |
|         0 | internal    | internal |
+-----------+-------------+----------+
```

```
> Syntax Help: [SHOW CATALOGS](https://doris.apache.org/docs/dev/sql-manual/sql-reference/Show-
    ↪ Statements/SHOW-CATALOGS/)

> You can view the CREATE CATALOG statement via [SHOW CREATE CATALOG](https://doris.apache.org/
    ↪ docs/dev/sql-manual/sql-reference/Show-Statements/SHOW-CREATE-CATALOG/).

> You can modify the Catalog PROPERTIES via [ALTER CATALOG](https://doris.apache.org/docs/dev/sql
    ↪ -manual/sql-reference/Data-Definition-Statements/Alter/ALTER-CATALOG/).
```

3. 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](https://doris.apache.org/docs/dev/sql-manual/sql-reference/Utility-
    ↪ Statements/SWITCH/)
```

4. 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'}
```

Connect to Iceberg

See Iceberg

Connect to Hudi

See Hudi

Connect to Elasticsearch

See Elasticsearch

Connect to JDBC

See JDBC

2.7.1.1.3 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 to an 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.

2.7.1.1.4 Privilege Management

Access from Doris to databases and tables in an External Catalog is not under the privilege control of the external catalog itself, but is authorized by Doris.

Along with the new Multi-Catalog feature, we also added privilege management at the Catalog level (See Privilege Management for details).

2.7.1.1.5 Database synchronizing management

Setting `include_database_list` and `exclude_database_list` in Catalog properties to specify databases to synchronize.

`include_database_list`: Only synchronize the specified databases. split with ',' , default value is " , means no filter takes effect, synchronizes all databases. db name is case sensitive.

`exclude_database_list`: Specify databases that do not need to synchronize. split with ',' , default value is " , means no filter takes effect, synchronizes all databases. db name is case sensitive.

> When `include_database_list` and `exclude_database_list` specify overlapping databases, `exclude_`
> `↪ database_list` would take effect with higher privilege over `include_database_list`.
>
> To connect JDBC, these two properties should work with `only_specified_database`, see JDBC for more detail.

2.7.1.1.6 Metadata Update

Manual Update

By default, changes in metadata of external data sources, including addition or deletion of tables and columns, will not be synchronized into Doris.

Users need to manually update the metadata using the REFRESH CATALOG command.

Automatic Update

Hive Metastore

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 so they will not be synchronized.
CREATE TABLE	Create a table in the corresponding database.
DROP TABLE	Delete a table in the corresponding database, and invalidate the cache of that table.
ALTER TABLE	If it is a 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:

1. `enable_hms_events_incremental_sync`: This specifies whether to enable automatic incremental synchronization for metadata, which is disabled by default.
2. `hms_events_polling_interval_ms`: This specifies the interval between two readings, which is set to 10000 by default. (Unit: millisecond)
3. `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, you need to modify the hive-site.xml of HMS and then restart HMS:

```
<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>
```

Note: To enable automatic update, whether for existing Catalogs or newly created Catalogs, all you need is to set `enable_hms_events_incremental_sync` to `true`, and then restart the FE node. You don' t need to manually update the metadata before or after the restart.

Timing Refresh

When creating a catalog, specify the refresh time parameter `metadata_refresh_interval_sec` in the properties, in seconds. If this parameter is set when creating a catalog, the master node of FE will refresh the catalog regularly according to the parameter value. Three types are currently supported

- hms: Hive MetaStore -es: Elasticsearch
- jdbc: Standard interface for database access (JDBC)

Example

```
-- 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"
);
```

2.7.1.2 Hive

Once Doris is connected to Hive Metastore or made compatible with Hive Metastore metadata service, it can access databases and tables in Hive and conduct queries.

Besides Hive, many other systems, such as Iceberg and Hudi, use Hive Metastore to keep their metadata. Thus, Doris can also access these systems via Hive Catalog.

2.7.1.2.1 Usage

When connnecting to Hive, Doris:

1. 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. Supports Hive version 1/2/3;
3. Supports both Managed Table and External Table;
4. Can identify metadata of Hive, Iceberg, and Hudi stored in Hive Metastore;
5. Supports Hive tables with data stored in JuiceFS, which can be used the same way as normal Hive tables (put `juicefs-hadoop-x.x.x.jar` in `fe/lib/` and `apache_hdfs_broker/lib/`).
6. Supports Hive tables with data stored in CHDFS, which can be used the same way as normal Hive tables. Follow below steps to prepare doris environment:

    1. put chdfs_hadoop_plugin_network-x.x.jar in fe/lib/ and apache_hdfs_broker/lib/
    2. copy core-site.xml and hdfs-site.xml from hive cluster to fe/conf/ and apache_hdfs_broker/conf

7. Supports Hive / Iceberg tables with data stored in GooseFS(GFS), which can be used the same way as normal Hive tables. Follow below steps to prepare doris environment:

    1. put goosefs-x.x.x-client.jar in fe/lib/ and apache_hdfs_broker/lib/
    2. add extra properties 'fs.AbstractFileSystem.gfs.impl' = 'com.qcloud.cos.goosefs.hadoop.GooseFileSystem', 'fs.gfs.impl' = 'com.qcloud.cos.goosefs.hadoop.FileSystem' when creating catalog

8. If the Hadoop node is configured with hostname, please ensure to add the corresponding mapping relationship to the /etc/hosts file.

2.7.1.2.2 Create Catalog

```sql
CREATE CATALOG hive 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'
);
```

In addition to `type` and `hive.metastore.uris`, which are required, you can specify other parameters regarding the connection.

For example, to specify HDFS HA:

```sql
CREATE CATALOG hive 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'
);
```

To specify HDFS HA and Kerberos authentication information:

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004',
    'hive.metastore.sasl.enabled' = 'true',
    'hive.metastore.kerberos.principal' = 'your-hms-principal',
    'dfs.nameservices'='your-nameservice',
    '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',
    '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'
);
```

Remember `krb5.conf` and `keytab` file should be placed at all BE nodes and FE nodes. The location of `keytab` file should be equal to the value of `hadoop.kerberos.keytab`. As default, `krb5.conf` should be placed at `/etc/krb5.conf`.

Value of `hive.metastore.kerberos.principal` should be same with the same name property used by HMS you are connecting to, which can be found in `hive-site.xml`.

To provide Hadoop KMS encrypted transmission information:

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004',
    'dfs.encryption.key.provider.uri' = 'kms://http@kms_host:kms_port/kms'
);
```

Or to connect to Hive data stored on JuiceFS:

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004',
    'hadoop.username' = 'root',
    'fs.jfs.impl' = 'io.juicefs.JuiceFileSystem',
    'fs.AbstractFileSystem.jfs.impl' = 'io.juicefs.JuiceFS',
    'juicefs.meta' = 'xxx'
);
```

Or to connect to Glue and data stored on S3:

```
CREATE CATALOG hive PROPERTIES (
    "type"="hms",
    "hive.metastore.type" = "glue",
    "aws.region" = "us-east-1",
    "aws.glue.access-key" = "ak",
    "aws.glue.secret-key" = "sk",
    "AWS_ENDPOINT" = "s3.us-east-1.amazonaws.com",
    "AWS_REGION" = "us-east-1",
    "AWS_ACCESS_KEY" = "ak",
    "AWS_SECRET_KEY" = "sk",
    "use_path_style" = "true"
);
```

In Doris 1.2.1 and newer, you can create a Resource that contains all these parameters, and reuse the Resource when creating new Catalogs. Here is an example:

```
#### 1. Create Resource
CREATE RESOURCE hms_resource 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'
);

#### 2. Create Catalog and use an existing Resource. The key and value information in the
    ↪ followings will overwrite the corresponding information in the Resource.
CREATE CATALOG hive WITH RESOURCE hms_resource PROPERTIES(
    'key' = 'value'
);
```

You can use the config `file.meta.cache.ttl-second` to set TTL(Time-to-Live) config of File Cache, so that the stale file info will be invalidated automatically after expiring. The unit of time is second.

You can also set file_meta_cache_ttl_second to 0 to disable file cache.Here is an example:

```
CREATE CATALOG hive 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',
    'file.meta.cache.ttl-second' = '60'
);
```

You can also put the `hive-site.xml` file in the `conf` directories of FE and BE. This will enable Doris to automatically read information from `hive-site.xml`. The relevant information will be overwritten based on the following rules :

- Information in Resource will overwrite that in `hive-site.xml`.
- Information in `CREATE CATALOG PROPERTIES` will overwrite that in Resource.

Hive Versions

Doris can access Hive Metastore in all Hive versions. By default, Doris uses the interface compatible with Hive 2.3 to access Hive Metastore. You can specify a certain Hive version when creating Catalogs, for example:

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://172.21.0.1:7004',
    'hive.version' = '1.1.0'
);
```

2.7.1.2.3 Column Type Mapping

This is applicable 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>`	Support nested array, such as `array<array<int>>`
`map<KeyType, ↪ ValueType>`	`map<KeyType, ↪ ValueType>`	Not support nested map. KeyType and ValueType should be primitive types.

HMS Type	Doris Type	Comment
`struct<col1: ↪ Type1, col2: ↪ Type2, ...>`	`struct<col1: ↪ Type1, col2: ↪ Type2, ...>`	Not support nested struct. Type1, Type2, … should be primitive types.
other	unsupported	

2.7.1.2.4 Use Ranger for permission verification

Apache Ranger is a security framework for monitoring, enabling services, and managing comprehensive data security access on the Hadoop platform.

Currently, Doris supports Ranger’s library, table, and column permissions, but does not support encryption, row permissions, and so on.

Environment configuration

Connecting to Hive Metastore with Ranger permission verification enabled requires additional configuration&configuration environment: 1. When creating a catalog, add:

```
"access_controller.properties.ranger.service.name" = "hive",
"access_controller.class" = "org.apache.doris.catalog.authorizer.
    ↪ RangerHiveAccessControllerFactory",
```

2. Configure all FE environments:

    1. Copy the configuration files ranger-live-audit.xml, ranger-live-security.xml, ranger-policymgr-ssl.xml under the HMS conf directory to/conf directory.
    2. Modify the properties of ranger-live-security.xml. The reference configuration 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>
```

```
3. To obtain the log of Ranger authentication itself, you can click<doris_ Add the configuration
    ↪ file log4j.properties under the home>/conf directory.

4. Restart FE.
```

Best Practices

1.Create user user1 on the ranger side and authorize the query permission of db1.table1.col1

2.Create the role role1 on the ranger side and authorize the query permission of db1.table1.col2

3.Create user user1 with the same name in Doris, and user1 will directly have the query permission of db1.table1.col1

4.Create the role role1 with the same name in Doris and assign role1 to user1. User1 will have query permissions for both db1.table1.col1 and col2

2.7.1.3 Iceberg

2.7.1.3.1 Usage

When connecting to Iceberg, Doris:

1. Supports Iceberg V1/V2 table formats;
2. Supports Position Delete but not Equality Delete for V2 format;
3. Supports Parquet formmat;

3. Supports Hive / Iceberg tables with data stored in GooseFS(GFS), which can be used the same way as normal Hive tables. Follow below steps to prepare doris environment:
    1. put goosefs-x.x.x-client.jar in fe/lib/ and apache_hdfs_broker/lib/
    2. add extra properties 'fs.AbstractFileSystem.gfs.impl' = 'com.qcloud.cos.goosefs.hadoop.GooseFileSystem', 'fs.gfs.impl' = 'com.qcloud.cos.goosefs.hadoop.FileSystem' when creating catalog

2.7.1.3.2 Create Catalog

Hive Metastore Catalog

Same as creating Hive Catalogs. A simple example is provided here. See Hive for more information.

```
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'
);
```

Iceberg Native Catalog

Access metadata with the iceberg API. The Hive, REST, Glue and other services can serve as the iceberg catalog.

Using Iceberg Hive Catalog

```
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'
);
```

Using Iceberg Glue Catalog

```
CREATE CATALOG glue PROPERTIES (
"type"="iceberg",
"iceberg.catalog.type" = "glue",
"glue.endpoint" = "https://glue.us-east-1.amazonaws.com",
"warehouse" = "s3://bucket/warehouse",
"AWS_ENDPOINT" = "s3.us-east-1.amazonaws.com",
"AWS_REGION" = "us-east-1",
"AWS_ACCESS_KEY" = "ak",
"AWS_SECRET_KEY" = "sk",
"use_path_style" = "true"
);
```

`glue.endpoint`: Glue Endpoint. See AWS Glue endpoints and quotas.

`warehouse`: Glue Warehouse Location. To determine the root path of the data warehouse in storage.

The other properties can refer to Iceberg Glue Catalog

- Using Iceberg REST Catalog

RESTful service as the server side. Implementing RESTCatalog interface of iceberg to obtain metadata.

```
CREATE CATALOG iceberg PROPERTIES (
    'type'='iceberg',
    'iceberg.catalog.type'='rest',
    'uri' = 'http://172.21.0.1:8181',
);
```

If you want to use S3 storage, the following properties need to be set.

```
"AWS_ACCESS_KEY" = "ak"
"AWS_SECRET_KEY" = "sk"
"AWS_REGION" = "region-name"
"AWS_ENDPOINT" = "http://endpoint-uri"
"AWS_CREDENTIALS_PROVIDER" = "provider-class-name" // Optional. The default credentials class is
    ↪ based on BasicAWSCredentials.
```

2.7.1.3.3 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

2.7.1.3.4 Time Travel

Doris supports reading the specified Snapshot of Iceberg tables.

Each write operation to an Iceberg table will generate a new Snapshot.

By default, a read request will only read the latest Snapshot.

You can read data of historical table versions using the `FOR TIME AS OF` or `FOR VERSION AS OF` statements based on the Snapshot ID or the timepoint the Snapshot is generated. For example:

`SELECT * FROM iceberg_tbl FOR TIME AS OF "2022-10-07 17:20:37";`

`SELECT * FROM iceberg_tbl FOR VERSION AS OF 868895038966572;`

You can use the iceberg_meta table function to view the Snapshot details of the specified table.

2.7.1.4 Hudi

2.7.1.4.1 Usage

1. Currently, Doris supports Snapshot Query on Copy-on-Write Hudi tables and Read Optimized Query on Merge-on-Read tables. In the future, it will support Snapshot Query on Merge-on-Read tables and Incremental Query.
2. Doris only supports Hive Metastore Catalogs currently. The usage is basically the same as that of Hive Catalogs. More types of Catalogs will be supported in future versions.

2.7.1.4.2 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'
);
```

2.7.1.4.3 Column Type Mapping

Same as that in Hive Catalogs. See the relevant section in Hive.

2.7.1.5 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.

2.7.1.5.1 Usage

1. Doris supports Elasticsearch 5.x and newer versions.

2.7.1.5.2 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
`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.

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.

2.7.1.5.3 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	

ES Type	Doris Type	Comment
text	string	
ip	string	
nested	string	
object	string	
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?pretty" -H 'Content-Type: application/json' -d '
{
    "_doc": {
        "_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.

2.7.1.5.4 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.

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.

2.7.1.5.5 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.

2.7.1.5.6 Appendix

How Doris Conducts Queries in ES

```
+----------------------------------------------+
|                                              |
| Doris      +------------------+              |
|            |       FE         +--------------+-------+
|            |                  |  Request Shard Location
|            +--+-------------+-+              |       |
|               ^             ^                |       |
|               |             |                |       |
|  +-------------------+ +------------------+  |       |
|  |            |      | |    |             |  |       |
|  | +----------+----+ | | +--+-----------+ |  |       |
|  | |      BE       | | | |      BE      | |  |       |
|  | +---------------+ | | +--------------+ |  |       |
+----------------------------------------------+       |
   |        |          | |        |         |          |
   |        |          | |        |         |          |
   |    HTTP SCROLL    | |    HTTP SCROLL   |          |
+-----------+---------------------+------------+       |
|  |        v          | |        v         |  |       |
|   | +-------+--------+ | | +------+-------+ |   |       |
|   | |                | | | |               | |   |       |
|   | |   DataNode     | | | |   DataNode    +<-----------+
|   | |                | | | |               | |   |       |
|   | |                +<--------------------------------+
|   | +----------------+ | | |--------------| |   |       |
|   +-------------------+ +-------------------+   |       |
|    Same Physical Node                           |       |
|                                                 |       |
|           +-----------------------+             |       |
|           |                       |             |       |
|           |      MasterNode       +<-----------------+
| 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.

2.7.1.6 JDBC

JDBC Catalogs in Doris are connected to external data sources using the standard JDBC protocol.

Once connected, Doris will ingest metadata of databases and tables from the external data sources in order to enable quick access to external data.

2.7.1.6.1 Usage

1. Supported datas sources include MySQL, PostgreSQL, Oracle, SQLServer, Clickhouse, Doris, SAP HANA, Trino and OceanBase.

2.7.1.6.2 Create Catalog

1. MySQL

```
CREATE CATALOG jdbc_mysql PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="123456",
    "jdbc_url" = "jdbc:mysql://127.0.0.1:3306/demo",
    "driver_url" = "mysql-connector-java-5.1.47.jar",
    "driver_class" = "com.mysql.jdbc.Driver"
)
```

2. PostgreSQL

```
    CREATE CATALOG jdbc_postgresql PROPERTIES (
        "type"="jdbc",
        "user"="root",
        "password"="123456",
        "jdbc_url" = "jdbc:postgresql://127.0.0.1:5449/demo",
        "driver_url" = "postgresql-42.5.1.jar",
        "driver_class" = "org.postgresql.Driver"
    );
```

Doris obtains all schemas that PG user can access through the SQL statement: `select nspname from pg_namespace where has_schema_privilege('<UserName>', nspname, 'USAGE');` and map these schemas to doris database.

As for data mapping from PostgreSQL to Doris, one Database in Doris corresponds to one schema in the specified database in PostgreSQL (for example, "demo" in `jdbc_url` above), and one Table in that Database corresponds to one table in that schema. To make it more intuitive, the mapping relations are as follows:

Doris	PostgreSQL
Catalog	Database
Database	Schema
Table	Table

3. Oracle

```
    CREATE CATALOG jdbc_oracle PROPERTIES (
        "type"="jdbc",
        "user"="root",
        "password"="123456",
        "jdbc_url" = "jdbc:oracle:thin:@127.0.0.1:1521:helowin",
        "driver_url" = "ojdbc6.jar",
        "driver_class" = "oracle.jdbc.driver.OracleDriver"
    );
```

As for data mapping from Oracle to Doris, one Database in Doris corresponds to one User, and one Table in that Database corresponds to one table that the User has access to. In conclusion, the mapping relations are as follows:

Doris	Oracle
Catalog	Database
Database	User
Table	Table

4. Clickhouse

```
CREATE CATALOG jdbc_clickhouse PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="123456",
    "jdbc_url" = "jdbc:clickhouse://127.0.0.1:8123/demo",
    "driver_url" = "clickhouse-jdbc-0.3.2-patch11-all.jar",
    "driver_class" = "com.clickhouse.jdbc.ClickHouseDriver"
);
```

5. SQLServer

```
CREATE CATALOG sqlserver_catalog PROPERTIES (
 "type"="jdbc",
 "user"="SA",
 "password"="Doris123456",
 "jdbc_url" = "jdbc:sqlserver://localhost:1433;DataBaseName=doris_test",
 "driver_url" = "mssql-jdbc-11.2.3.jre8.jar",
 "driver_class" = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
);
```

As for data mapping from SQLServer to Doris, one Database in Doris corresponds to one schema in the specified database in SQLServer (for example, "doris_test" in `jdbc_url` above), and one Table in that Database corresponds to one table in that schema. The mapping relations are as follows:

Doris	SQLServer
Catalog	Database
Database	Schema
Table	Table

6. Doris

Jdbc Catalog also support to connect another Doris database:

```
CREATE CATALOG doris_catalog PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="123456",
    "jdbc_url" = "jdbc:mysql://127.0.0.1:9030?useSSL=false",
    "driver_url" = "mysql-connector-java-5.1.47.jar",
    "driver_class" = "com.mysql.jdbc.Driver"
);
```

Currently, Jdbc Catalog only support to use 5.x version of JDBC jar package to connect another Doris database. If you use 8.x version of JDBC jar package, the data type of column may not be matched.

7. SAP_HANA

```
CREATE CATALOG hana_catalog PROPERTIES (
    "type"="jdbc",
    "user"="SYSTEM",
    "password"="SAPHANA",
    "jdbc_url" = "jdbc:sap://localhost:31515/TEST",
    "driver_url" = "ngdbc.jar",
    "driver_class" = "com.sap.db.jdbc.Driver"
)
```

Doris	SAP_HANA
Catalog	Database
Database	Schema
Table	Table

8. Trino

```
CREATE CATALOG trino_catalog PROPERTIES (
    "type"="jdbc",
    "user"="hadoop",
    "password"="",
    "jdbc_url" = "jdbc:trino://localhost:9000/hive",
    "driver_url" = "trino-jdbc-389.jar",
    "driver_class" = "io.trino.jdbc.TrinoDriver"
);
```

When Trino is mapped, Doris' s Database corresponds to a Schema in Trino that specifies Catalog (such as "hive" in the 'jdbc_url' parameter in the example). The Table in Doris' s Database corresponds to the Tables in Trino' s Schema. That is, the mapping relationship is as follows:

Doris	Trino
Catalog	Catalog
Database	Schema
Table	Table

9. OceanBase

<<<<<<< HEAD

```
CREATE CATALOG jdbc_oceanbase_mysql PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="123456",
    "jdbc_url" = "jdbc:oceanbase://127.0.0.1:2881/demo",
    "driver_url" = "oceanbase-client-2.4.2.jar",
    "driver_class" = "com.oceanbase.jdbc.Drive",
    "oceanbase_mode" = "mysql"
)

CREATE CATALOG jdbc_oceanbase_oracle PROPERTIES (
    "type"="jdbc",
    "user"="root",
    "password"="123456",
    "jdbc_url" = "jdbc:oceanbase://127.0.0.1:2881/demo",
    "driver_url" = "oceanbase-client-2.4.2.jar",
    "driver_class" = "com.oceanbase.jdbc.Drive",
    "oceanbase_mode" = "oracle"
)
```

Parameter Description

Parameter	Required or Not	Default Value	Description
`user`	Yes		Username in relation to the corresponding database
`password`	Yes		Password for the corresponding database
`jdbc_url`	Yes		JDBC connection string
`driver_url`	Yes		JDBC Driver Jar
`driver_class`	Yes		JDBC Driver Class
`only_specified_database`	No	"false"	Whether only the database specified to be synchronized.
`lower_case_table_names`	No	"false"	Whether to synchronize jdbc external data source table names in lower case.
`oceanbase_mode`	No	""	When the connected external data source is OceanBase, the mode must be specified as mysql or oracle

Parameter	Required or Not	Default Value	Description
`include_database_list`	No	“”	When only_specified_database=true， only synchronize the specified databases. split with ‘,’ . db name is case sensitive.
`exclude_database_list`	No	“”	When only_specified_database=true， do not synchronize the specified databases. split with ‘,’ . db name is case sensitive.

`driver_url` can be specified in three ways:

1. File name. For example, `mysql-connector-java-5.1.47.jar`. Please place the Jar file package in `jdbc_drivers/` under the FE/BE deployment directory in advance so the system can locate the file. You can change the location of the file by modifying `jdbc_drivers_dir` in fe.conf and be.conf.
2. Local absolute path. For example, `file:///path/to/mysql-connector-java-5.1.47.jar`. Please place the Jar file package in the specified paths of FE/BE node.
3. HTTP address. For example, `https://doris-community-test-1308700295.cos.ap-hongkong` ↪ `.myqcloud.com/jdbc_driver/mysql-connector-java-5.1.47.jar`. The system will download the Driver file from the HTTP address. This only supports HTTP services with no authentication requirements.

`only_specified_database`: When the JDBC is connected, you can specify which database/schema to connect. For example, you can specify the DataBase in mysql `jdbc_url`; you can specify the CurrentSchema in PG `jdbc` ↪ `_url`.

`include_database_list`: When `only_specified_database=true`, only synchronize the specified databases. split with ‘,’ , default value is ” , means no filter takes effect, synchronizes all databases. db name is case sensitive.

`exclude_database_list`: When `only_specified_database=true`, specify databases that do not need to synchronize. split with ‘,’ , default value is ” , means no filter takes effect, synchronizes all databases. db name is case sensitive.

When `include_database_list` and `exclude_database_list` specify overlapping databases, `exclude_` ↪ `database_list` would take effect with higher privilege over `include_database_list`.

If you connect the Oracle database when using this property, please use the version of the jar package above 8 or more (such as ojdbc8.jar).

2.7.1.6.3 Query

```
select * from mysql_catalog.mysql_database.mysql_table where k1 > 1000 and k3 ='term';
```

In some cases, the keywords in the database might be used as the field names. For queries to function normally in these cases, Doris will add escape characters to the field names and tables names in SQL statements based on the rules of different databases, such as (“) for MySQL, ([]) for SQLServer, and (“”) for PostgreSQL and Oracle. This might require extra attention on case sensitivity. You can view the query statements sent to these various databases via `explain sql`.

2.7.1.6.4 Write Data

After creating a JDBC Catalog in Doris, you can write data or query results to it using the `insert into` statement. You can also ingest data from one JDBC Catalog Table to another JDBC Catalog Table.

Example:

```
insert into mysql_catalog.mysql_database.mysql_table values(1, "doris");
insert into mysql_catalog.mysql_database.mysql_table select * from table;
```

Transaction

In Doris, data is written to External Tables in batches. If the ingestion process is interrupted, rollbacks might be required. That’s why JDBC Catalog Tables support data writing transactions. You can utilize this feature by setting the session variable: `enable_`
`↪ odbc_transcation`.

```
set enable_odbc_transcation = true;
```

The transaction mechanism ensures the atomicity of data writing to JDBC External Tables, but it reduces performance to a certain extent. You may decide whether to enable transactions based on your own tradeoff.

2.7.1.6.5 Column Type Mapping

MySQL

MYSQL Type	Doris Type	Comment
BOOLEAN	BOOLEAN	
TINYINT	TINYINT	
SMALLINT	SMALLINT	
MEDIUMINT	INT	
INT	INT	
BIGINT	BIGINT	
UNSIGNED TINYINT	SMALLINT	Doris does not support UNSIGNED data types so UNSIGNED TINYINT will be mapped to SMALLINT.
UNSIGNED MEDIUMINT	INT	Doris does not support UNSIGNED data types so UNSIGNED MEDIUMINT will be mapped to INT.
UNSIGNED INT	BIGINT	Doris does not support UNSIGNED data types so UNSIGNED INT will be mapped to BIGINT.
UNSIGNED BIGINT	LARGEINT	
FLOAT	FLOAT	
DOUBLE	DOUBLE	
DECIMAL	DECIMAL	

MYSQL Type	Doris Type	Comment
DATE	DATE	
TIMESTAMP	DATETIME	
DATETIME	DATETIME	
YEAR	SMALLINT	
TIME	STRING	
CHAR	CHAR	
VARCHAR	VARCHAR	
TINYTEXT、 TEXT、 MEDIUMTEXT、 LONGTEXT、 TINYBLOB、 BLOB、 MEDIUMBLOB、 LONGBLOB、 TINYSTRING、 STRING、 MEDIUMSTRING、 LONGSTRING、 BINARY、 VARBINARY、 JSON、 SET、 BIT	STRING	
Other	UNSUPPORTED	

PostgreSQL

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	DATETIME	
date	DATE	
time	STRING	
interval	STRING	
point/line/lseg/box/path/polygon/circle	STRING	
cidr/inet/macaddr	STRING	
bit/bit(n)/bit varying(n)	STRING	`bit` will be mapped to `STRING` in Doris. It will be read as `true/false` instead of `1/0`
uuid/josnb	STRING	
Other	UNSUPPORTED	

Oracle

ORACLE Type	Doris Type	Comment
number(p) / number(p,0)	TINYINT/SMALLINT/INT/BIGINT/LARGEINT	Doris will determine the type to map to based on the value 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 same mapping as `number(p)/ number(p,0)`.
number		Doris does not support Oracle `NUMBER` type that does not specified p and s
float/real	DOUBLE	
DATE	DATETIME	
TIMESTAMP	DATETIME	
CHAR/NCHAR	STRING	
VARCHAR2/NVARCHAR2	STRING	
LONG/ RAW/ LONG RAW/ INTERVAL	STRING	
Other	UNSUPPORTED	

SQLServer

SQLServer Type	Doris Type	Comment
bit	BOOLEAN	
tinyint	SMALLINT	The tinyint type in SQLServer is an unsigned number so it will be mapped to SMALLINT in Doris.
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	
binary/varbinary	STRING	
time/datetimeoffset	STRING	
Other	UNSUPPORTED	

Clickhouse

ClickHouse Type	Doris Type	Comment
Bool	BOOLEAN	
String	STRING	
Date/Date32	DATEV2	JDBC CATLOG uses Datev2 type default when connecting ClickHouse
DateTime/DateTime64	DATETIMEV2	JDBC CATLOG uses DateTimev2 type default when connecting ClickHouse
Float32	FLOAT	
Float64	DOUBLE	
Int8	TINYINT	
Int16/UInt8	SMALLINT	Doris does not support UNSIGNED data types so UInt8 will be mapped to SMALLINT.
Int32/UInt16	INT	Doris does not support UNSIGNED data types so UInt16 will be mapped to INT.
Int64/Uint32	BIGINT	Doris does not support UNSIGNED data types so UInt32 will be mapped to BIGINT.
Int128/UInt64	LARGEINT	Doris does not support UNSIGNED data types so UInt64 will be mapped to LARGEINT.
Int256/UInt128/UInt256	STRING	Doris does not support data types of such orders of magnitude so these will be mapped to STRING.
DECIMAL	DECIMAL/DECIMALV3/STRING	The Data type is based on the DECIMAL field’s (precision, scale) and the `enable_decimal_conversion` configuration.
Enum/IPv4/IPv6/UUID	STRING	Data of IPv4 and IPv6 type will be displayed with an extra `/` as a prefix. To remove the `/`, you can use the `split_part`function.
Array	ARRAY	Array internal basic type adaptation logic refers to the preceding types. Nested types are not supported
Other	UNSUPPORTED	

Doris

Doris Type	Jdbc Catlog Doris Type	Comment
BOOLEAN	BOOLEAN	
TINYINT	TINYINT	
SMALLINT	SMALLINT	
INT	INT	
BIGINT	BIGINT	
LARGEINT	LARGEINT	
FLOAT	FLOAT	
DOUBLE	DOUBLE	

Doris Type	Jdbc Catlog Doris Type	Comment
DECIMAL / DECIMALV3	DECIMAL/DECIMALV3/STRING	The Data type is based on the DECIMAL field’s (precision, scale) and the `enable_decimal_conversion` configuration
DATE	DATEV2	JDBC CATLOG uses Datev2 type default when connecting DORIS
DATEV2	DATEV2	
DATETIME	DATETIMEV2	JDBC CATLOG uses DATETIMEV2 type default when connecting DORIS
DATETIMEV2	DATETIMEV2	
CHAR	CHAR	
VARCHAR	VARCHAR	
STRING	STRING	
TEXT	STRING	
Other	UNSUPPORTED	

SAP HANA

SAP HANA Type	Doris Type	Comment
BOOLEAN	BOOLEAN	
TINYINT	TINYINT	
SMALLINT	SMALLINT	
INTERGER	INT	
BIGINT	BIGINT	
SMALLDECIMAL	DECIMALV3	
DECIMAL	DECIMAL/DECIMALV3/STRING	The Data type is based on the DECIMAL field’s (precision, scale) and the `enable_decimal_conversion` configuration
REAL	FLOAT	
DOUBLE	DOUBLE	
DATE	DATEV2	JDBC CATLOG uses Datev2 type default when connecting HANA
TIME	TEXT	
TIMESTAMP	DATETIMEV2	JDBC CATLOG uses DATETIMEV2 type default when connecting HANA
SECONDDATE	DATETIMEV2	JDBC CATLOG uses DATETIMEV2 type default when connecting HANA
VARCHAR	TEXT	
NVARCHAR	TEXT	
ALPHANUM	TEXT	
SHORTTEXT	TEXT	
CHAR	CHAR	
NCHAR	CHAR	
Other	UNSUPPORTED	

Trino

Trino Type	Doris Type	Comment
boolean	BOOLEAN	
tinyint	TINYINT	
smallint	SMALLINT	
integer	INT	
bigint	BIGINT	
decimal	DECIMAL/DECIMALV3/STRING	The Data type is based on the DECIMAL field's (precision, scale) and the `enable_decimal_conversion` configuration
real	FLOAT	
double	DOUBLE	
date	DATE/DATEV2	JDBC CATLOG uses Datev2 type default when connecting Trino
timestamp	DATETIME/DATETIMEV2	JDBC CATLOG uses DATETIMEV2 type default when connecting Trino
varchar	TEXT	
char	CHAR	
array	ARRAY	Array internal basic type adaptation logic refers to the preceding types. Nested types are not supported
others	UNSUPPORTED	

Note: Currently, only Hive connected to Trino has been tested. Other data sources connected to Trino have not been tested.

OceanBase

For MySQL mode, please refer to MySQL type mapping For Oracle mode, please refer to Oracle type mapping

2.7.1.6.6 FAQ

1. Are there any other databases supported besides MySQL, Oracle, PostgreSQL, SQLServer, ClickHouse, SAP HANA, Trino and OceanBase?

Currently, Doris supports MySQL, Oracle, PostgreSQL, SQLServer, ClickHouse, SAP HANA, Trino and OceanBase. We are planning to expand this list. Technically, any databases that support JDBC access can be connected to Doris in the form of JDBC external tables. You are more than welcome to be a Doris contributor to expedite this effort.

2. Why does Mojibake occur when Doris tries to read emojis from MySQL external tables?

In MySQL, utf8mb3 is the default utf8 format. It cannot represent emojis, which require 4-byte encoding. To solve this, when creating MySQL external tables, you need to set utf8mb4 encoding for the corresponding columns, set the server encoding to utf8mb4, and leave the characterEncoding in JDBC URl empty (because utf8mb4 is not supported for this property, and anything other than utf8mb4 will cause a failure to write the emojis).

You can modify the configuration items globally:

```
Modify the my.ini file in the mysql directory (for linux system, modify the my.cnf file in the
    ↪  etc directory)
[client]
default-character-set=utf8mb4
```

```
[mysql]
Set the mysql default character set
default-character-set=utf8mb4

[mysqld]
Set the mysql character set server
character-set-server=utf8mb4
collation-server=utf8mb4_unicode_ci
init_connect='SET NAMES utf8mb4

Modify the type for the 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
```

3. Why does the error message "CAUSED BY: DataReadException: Zero date value prohibited" pop up when DateTime="0000:00:00 00:00:00" while reading MySQL external tables?

This error occurs because of an illegal DateTime. It can be fixed by modifying the `zeroDateTimeBehavior` parameter.

The options for this parameter include: `EXCEPTION`,`CONVERT_TO_NULL`,`ROUND`. Respectively, they mean to report error, convert to null, and round the DateTime to "0001-01-01 00:00:00" when encountering an illegal DateTime.

You can add `"jdbc_url"="jdbc:mysql://IP:PORT/doris_test?zeroDateTimeBehavior=convertToNull"` to the URL.

4. Why do loading failures happen when reading MySQL or other external tables?

For example:

```
failed to load driver class com.mysql.jdbc.driver in either of hikariconfig class loader
```

Such errors occur because the `driver_class` has been wrongly put when creating the Resource. The problem with the above example is the letter case. It should be corrected as `"driver_class" = "com.mysql.jdbc.Driver"`.

5. How to fix communication link failures?

If you run into the following errors:

```
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
```

Please check the be.out log of BE.

If it contains the following message:

```
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 Catalog. For example, `"jdbc_url" = "` ↪ `jdbc:mysql://127.0.0.1:3306/test?useSSL=false"`.

6. What to do with the `OutOfMemoryError` when querying MySQL databases?

To reduce memory usage, Doris obtains one batch of query results at a time, and has a size limit for each batch. However, MySQL conducts one-off loading of all query results by default, which means the "loading in batches" method won't work. To solve this, you need to specify "jdbc_url" = "jdbc:mysql://IP:PORT/doris_test?useCursorFetch=true" in the URL.

7. What to do with errors such as "CAUSED BY: SQLException OutOfMemoryError" when performing JDBC queries?

If you have set `useCursorFetch` for MySQL, you can increase the JVM memory limit by modifying the value of `jvm_max_heap_` ↪ `size` in be.conf. The current default value is 1024M.

8. When using JDBC to query MySQL large data volume, if the query can occasionally succeed, occasionally report the following errors, and all the MySQL connections are completely disconnected when the error occurs:

```
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 appears, it may be that mysql server's own memory or CPU resources are
    ↪ exhausted and the MySQL service is unavailable. You can try to increase the memory or CPU
    ↪  resources of MySQL Server.
```

2.7.1.7 Alibaba Cloud DLF

Data Lake Formation (DLF) is the unified metadata management service of Alibaba Cloud. It is compatible with the Hive Metastore protocol.

What is DLF

Doris can access DLF the same way as it accesses Hive Metastore.

2.7.1.7.1 Connect to DLF

The First Way, Create a Hive Catalog.

```
CREATE CATALOG hive_with_dlf PROPERTIES (
   "type"="hms",
   "dlf.catalog.proxyMode" = "DLF_ONLY",
   "hive.metastore.type" = "dlf",
   "dlf.catalog.endpoint" = "dlf.cn-beijing.aliyuncs.com",
   "dlf.catalog.region" = "cn-beijing",
   "dlf.catalog.uid" = "uid",
   "dlf.catalog.accessKeyId" = "ak",
   "dlf.catalog.accessKeySecret" = "sk"
);
```

`type` should always be `hms`. If you need to access Alibaba Cloud OSS on the public network, can add `"dlf.catalog.↪ accessPublic"="true"`.

- `dlf.catalog.endpoint`: DLF Endpoint. See Regions and Endpoints of DLF.
- `dlf.catalog.region`: DLF Region. See Regions and Endpoints of DLF.
- `dlf.catalog.uid`: Alibaba Cloud account. You can find the “Account ID” in the upper right corner on the Alibaba Cloud console.
- `dlf.catalog.accessKeyId`: AccessKey, which you can create and manage on the Alibaba Cloud console.
- `dlf.catalog.accessKeySecret`: SecretKey, which you can create and manage on the Alibaba Cloud console.

Other configuration items are fixed and require no modifications.

After the above steps, you can access metadata in DLF the same way as you access Hive MetaStore.

Doris supports accessing Hive/Iceberg/Hudi metadata in DLF.

The Second Way, Configure the Hive Conf

1. Create the `hive-site.xml` file, and put it in the `fe/conf` directory.

```xml
<?xml version="1.0"?>
<configuration>
    <!--Set to use dlf client-->
    <property>
        <name>hive.metastore.type</name>
        <value>dlf</value>
    </property>
    <property>
        <name>dlf.catalog.endpoint</name>
        <value>dlf-vpc.cn-beijing.aliyuncs.com</value>
    </property>
    <property>
        <name>dlf.catalog.region</name>
        <value>cn-beijing</value>
    </property>
    <property>
        <name>dlf.catalog.proxyMode</name>
        <value>DLF_ONLY</value>
    </property>
    <property>
        <name>dlf.catalog.uid</name>
        <value>20000000000000000</value>
    </property>
    <property>
        <name>dlf.catalog.accessKeyId</name>
        <value>XXXXXXXXXXXXXXX</value>
    </property>
    <property>
        <name>dlf.catalog.accessKeySecret</name>
        <value>XXXXXXXXXXXXXXXXX</value>
    </property>
</configuration>
```

2. Restart FE, Doris will read and parse `fe/conf/hive-site.xml`. And then Create Catalog via the `CREATE CATALOG` statement.

```sql
CREATE CATALOG hive_with_dlf PROPERTIES (
    "type"="hms",
    "hive.metastore.uris" = "thrift://127.0.0.1:9083"
)
```

`type` should always be `hms`; while `hive.metastore.uris` can be arbitary since it is not used in real practice, but it should follow the format of Hive Metastore Thrift URI.

2.7.1.8 FAQ

1. What to do with errors such as `failed to get schema` and `Storage schema reading not supported` when accessing Icerberg tables via Hive Metastore?

To fix this, please place the Jar file package of `iceberg` runtime in the `lib/` directory of Hive.

And configure as follows in `hive-site.xml` :

```
metastore.storage.schema.reader.impl=org.apache.hadoop.hive.metastore.SerDeStorageSchemaReader
```

After configuring, please restart Hive Metastore.

2. What to do with the `GSS initiate failed` error when connecting to Hive Metastore with Kerberos authentication?

Usually it is caused by incorrect Kerberos authentication information, you can troubleshoot by the following steps:

1. In versions before 1.2.1, the libhdfs3 library that Doris depends on does not enable gsasl. Please update to a version later than 1.2.2.
2. Confirm that the correct keytab and principal are set for each component, and confirm that the keytab file exists on all FE and BE nodes.
    1. `hadoop.kerberos.keytab`/`hadoop.kerberos.principal`: for Hadoop HDFS
    2. `hive.metastore.kerberos.principal`: for hive metastore.
3. Try to replace the IP in the principal with a domain name (do not use the default _HOST placeholder)
4. Confirm that the `/etc/krb5.conf` file exists on all FE and BE nodes.
5. What to do with the`java.lang.VerifyError: xxx` error when accessing HDFS 3.x?

Doris 1.2.1 and the older versions rely on Hadoop 2.8. Please update Hadoop to 2.10.2 or update Doris to 1.2.2 or newer.

4. An error is reported when using KMS to access HDFS: `java.security.InvalidKeyException: Illegal key size`

   Upgrade the JDK version to a version >= Java 8 u162. Or download and install the JCE Unlimited Strength Jurisdiction Policy Files corresponding to the JDK.

5. When querying a table in ORC format, FE reports an error `Could not obtain block` or `Caused by: java.lang.`
   `↪ NoSuchFieldError: types`

   For ORC files, by default, FE will access HDFS to obtain file information and split files. In some cases, FE may not be able to access HDFS. It can be solved by adding the following parameters:

   `"hive.exec.orc.split.strategy" = "BI"`

   Other options: HYBRID (default), ETL.

6. An error is reported when connecting to SQLServer through JDBC Catalog: `unable to find valid certification`
   `↪ path to requested target`

   Please add `trustServerCertificate=true` option in `jdbc_url`.

7. When connecting to the MySQL database through the JDBC Catalog, the Chinese characters are garbled, or the Chinese character condition query is incorrect

   Please add `useUnicode=true&characterEncoding=utf-8` in `jdbc_url`

> Note: After version 1.2.3, these parameters will be automatically added when using JDBC Catalog to connect to the MySQL database.

8. An error is reported when connecting to the MySQL database through the JDBC Catalog: `Establishing SSL connection ↪ without server's identity verification is not recommended`

   Please add `useSSL=true` in `jdbc_url`

9. An error is reported when connecting Hive Catalog: `Caused by: java.lang.NullPointerException`

   If there is stack trace in fe.log:

```
Caused by: java.lang.NullPointerException
    at org.apache.hadoop.hive.ql.security.authorization.plugin.
        ↪ AuthorizationMetaStoreFilterHook.getFilteredObjects(
        ↪ AuthorizationMetaStoreFilterHook.java:78) ~[hive-exec-3.1.3-core.jar:3.1.3]
    at org.apache.hadoop.hive.ql.security.authorization.plugin.
        ↪ AuthorizationMetaStoreFilterHook.filterDatabases(AuthorizationMetaStoreFilterHook
        ↪ .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 `create catalog` statement.
```

10. An error is reported when connecting to the Hive database through the Hive Catalog: `RemoteException: SIMPLE ↪ authentication is not enabled. Available: [TOKEN, KERBEROS]`

    If both `show databases` and `show tables` are OK, and the above error occurs when querying, we need to perform the following two operations:

    - Core-site.xml and hdfs-site.xml need to be placed in the fe/conf and be/conf directories
    - The BE node executes the kinit of Kerberos, restarts the BE, and then executes the query.

11. If the `show tables` is normal after creating the Hive Catalog, but the query report `java.net.UnknownHostException: ↪ xxxxx`

    Add a property in CATALOG:

```
'fs.defaultFS' = 'hdfs://<your_nameservice_or_actually_HDFS_IP_and_port>'
```

12. The values of the partition fields in the hudi table can be found on hive, but they cannot be found on doris.

    Doris and hive currently query hudi differently. Doris needs to add partition fields to the avsc file of the hudi table structure. If not added, it will cause Doris to query partition_ Val is empty (even if home. datasource. live_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": "名称"
        },
        {
        "name": "create_time",
        "type": "string",
        "doc": "创建时间"
        }
    ]
}
```

13. The table in orc format of Hive 1.x may encounter system column names such as `_col0`, `_col1`, `_col2`… in the underlying orc file schema, which need to be specified in the catalog configuration. Add `hive.version` to 1.x.x so that it will use the column names in the hive table for mapping.

```
CREATE CATALOG hive PROPERTIES (
    'hive.version' = '1.x.x'
);
```

14. When using JDBC Catalog to synchronize MySQL data to Doris, the date data synchronization error occurs. It is necessary to check whether the MySQL version corresponds to the MySQL driver package. For example, the driver com.mysql.cj.jdbc.Driver is required for MySQL8 and above.

15. If an error is reported while configuring Kerberos in the catalog: `SIMPLE authentication is not enabled.` ↪ `Available:[TOKEN, KERBEROS]`.

Need to put `core-site.xml` to the `"${DORIS_HOME}/be/conf"` directory.

If an error is reported while accessing HDFS: `No common protection layer between client and server`, check the `hadoop.rpc.protection` on the client and server to make them 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>
```

16. The solutions when configuring Kerberos in the catalog and encounter an error: `Unable to obtain password from` ↪ `user`.

    - The principal used must exist in the klist, use `klist -kt your.keytab` to check.
    - Ensure the catalog configuration correct, such as missing the `yarn.resourcemanager.principal`.
    - If the preceding checks are correct, the JDK version installed by yum or other package-management utility in the current system maybe have an unsupported encryption algorithm. It is recommended to install JDK by yourself and set JAVA ↪ _HOME environment variable.

17. If an error is reported while querying the catalog with Kerberos: `GSSException: No valid credentials provided` ↪ `(Mechanism level: Failed to find any Kerberos Ticket)`.

    - Restarting FE and BE can solve the problem in most cases.
    - Before the restart all the nodes, can put `-Djavax.security.auth.useSubjectCredsOnly=false` to the JAVA ↪ _OPTS in `"${DORIS_HOME}/be/conf/be.conf"`, which can obtain credentials through the underlying mechanism, rather than through the application.
    - Get more solutions to common JAAS errors from the JAAS Troubleshooting.

18. If an error related to the Hive Metastore is reported while querying the catalog: `Invalid method name`.

    Configure the `hive.version`.

```
  CREATE CATALOG hive PROPERTIES (
      'hive.version' = '2.x.x'
  );
```

19. `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.
- Use telnet to check connectivity with the DataNode.
- Check the DataNode logs.

If encountering 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 indicates that HDFS is configured for encrypted transmission while the client is not, causing the error.

You can use any of the following solutions:

- Copying hdfs-site.xml and core-site.xml to the 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.

2.7.2 External Table

2.7.2.1 Elasticsearch External Table

Please use ES Catalog to access Elasticsearch (ES) data sources, this function will no longer be maintained after version 1.2.2.

Doris-on-ES provides an advanced OLAP solution, where you can benefit from both the distributed query planning capability of Doris and the full-text search capability of ES:

1. Multi-index distributed Join queries in ES;
2. Join queries across Doris and ES as well as full-text search and filter.

This topic is about how ES External Tables are implemented and used in Doris.

2.7.2.1.1 Basic Concepts

Doris-Related Concepts

- FE: Frontend of Doris, responsible for metadata management and request processing
- BE: Backend of Doris, responsible for query execution and data storage

ES-Related Concepts

- DataNode: nodes for data storage and computing in ES
- MasterNode: nodes for managing metadata, nodes, and data distribution in ES
- scroll: built-in dataset cursor in ES, used to stream scan and filter data
- _source: the original JSON file in data ingestion
- doc_values: the columnar storage definition of fields in ES/Lucene
- keyword: string field, ES/Lucene not tokenizing texts
- text: string field, ES/Lucene tokenizing texts using the specified tokenizer (the standard tokenizer, if not specified)

2.7.2.1.2 Usage

Create ES Index

```
PUT test
{
   "settings": {
      "index": {
         "number_of_shards": "1",
         "number_of_replicas": "0"
      }
   },
   "mappings": {
      "doc": { // In ES 7.x or newer, you don't have to specify the type when creating an index.
          ↪ It will come with a unique `_doc` type by default.
         "properties": {
            "k1": {
               "type": "long"
            },
            "k2": {
               "type": "date"
            },
            "k3": {
               "type": "keyword"
            },
            "k4": {
               "type": "text",
               "analyzer": "standard"
            },
            "k5": {
               "type": "float"
            }
         }
      }
   }
}
```

Data Ingestion

```
POST /_bulk
{"index":{"_index":"test","_type":"doc"}}
{ "k1" : 100, "k2": "2020-01-01", "k3": "Trying out Elasticsearch", "k4": "Trying out
    ↪ Elasticsearch", "k5": 10.0}
{"index":{"_index":"test","_type":"doc"}}
{ "k1" : 100, "k2": "2020-01-01", "k3": "Trying out Doris", "k4": "Trying out Doris", "k5": 10.0}
{"index":{"_index":"test","_type":"doc"}}
{ "k1" : 100, "k2": "2020-01-01", "k3": "Doris On ES", "k4": "Doris On ES", "k5": 10.0}
```

```
{"index":{"_index":"test","_type":"doc"}}
{ "k1" : 100, "k2": "2020-01-01", "k3": "Doris", "k4": "Doris", "k5": 10.0}
{"index":{"_index":"test","_type":"doc"}}
{ "k1" : 100, "k2": "2020-01-01", "k3": "ES", "k4": "ES", "k5": 10.0}
```

Create ES External Table in Doris

See CREATE TABLE for syntax details.

```
CREATE EXTERNAL TABLE `test` // You don't have to specify the schema. The system will auto-pull
    ↪ the ES mapping for tabale creation.
ENGINE=ELASTICSEARCH
PROPERTIES (
"hosts" = "http://192.168.0.1:8200,http://192.168.0.2:8200",
"index" = "test",
"type" = "doc",
"user" = "root",
"password" = "root"
);

CREATE EXTERNAL TABLE `test` (
  `k1` bigint(20) COMMENT "",
  `k2` datetime COMMENT "",
  `k3` varchar(20) COMMENT "",
  `k4` varchar(100) COMMENT "",
  `k5` float COMMENT ""
) ENGINE=ELASTICSEARCH // ENGINE should be Elasticsearch.
PROPERTIES (
"hosts" = "http://192.168.0.1:8200,http://192.168.0.2:8200",
"index" = "test",
"type" = "doc",
"user" = "root",
"password" = "root"
);
```

Parameter Description:

Parameter	Description
hosts	One or multiple ES cluster addresses or the load balancer address of ES frontend
index	The corresponding ES index; supports alias, but not when doc_value is used.
type	Type of index (no longer needed in ES 7.x or newer)
user	Username for the ES cluster
password	The corresponding password

- In the ES versions before 7.x, please choose the correct index type when creating tables.

- Only HTTP Basic authentication is supported. Please make sure the user has access to the relevant paths (/_cluster/state/, _nodes/http) and read privilege on the index. If you have not enabled security authentication for the clusters, you don’t have to set the username and password.
- Please ensure that the column names and types in the Doris are consistent with the field names and types in ES.
- The ENGINE should be Elasticsearch.

Predicate Pushdown

A key feature of `Doris On ES` is predicate pushdown: The filter conditions will be pushed down to ES so only the filtered data will be returned. This can largely improve query performance and reduce usage of CPU, memory, and IO in Doris and ES.

Operators will be converted into ES queries as follows:

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	QueryDSL in the ES-native JSON format

Data Type Mapping

Doris vs ES	byte	short	integer	long	float	double	keyword	text	date
tinyint	√								
smallint	√	√							
int	√	√	√						
bigint	√	√	√	√					
float					√				
double						√			
char							√	√	
varchar							√	√	
date									√
datetime									√

Improve Query Speed by Enabling Columnar Scan (enable_docvalue_scan=true)

```
CREATE EXTERNAL TABLE `test` (
  `k1` bigint(20) COMMENT "",
  `k2` datetime COMMENT "",
  `k3` varchar(20) COMMENT "",
  `k4` varchar(100) COMMENT "",
  `k5` float COMMENT ""
) ENGINE=ELASTICSEARCH
PROPERTIES (
"hosts" = "http://192.168.0.1:8200,http://192.168.0.2:8200",
"index" = "test",
"user" = "root",
"password" = "root",
"enable_docvalue_scan" = "true"
);
```

Parameter Description:

Parameter	Description
enable_docvalue_scan	This specifies whether to acquire value from the query field via ES/Lucene columnar storage. It is set to false by default.

If this parameter is set to true, Doris will follow these rules when obtaining data from ES:

- 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.

Sniff Keyword Fields (enable_keyword_sniff=true)

```
CREATE EXTERNAL TABLE `test` (
  `k1` bigint(20) COMMENT "",
  `k2` datetime COMMENT "",
  `k3` varchar(20) COMMENT "",
  `k4` varchar(100) COMMENT "",
  `k5` float COMMENT ""
) ENGINE=ELASTICSEARCH
PROPERTIES (
"hosts" = "http://192.168.0.1:8200,http://192.168.0.2:8200",
"index" = "test",
"user" = "root",
"password" = "root",
"enable_keyword_sniff" = "true"
);
```

Parameter Description:

Parameter	Description
enable_keyword_sniff	This specifies whether to sniff (text) `fields` for untokenized (keyword) fields (multi-fields mechanism)

You can start data ingestion without creating an index since ES will generate a new index automatically. For string fields, ES will create a field of both `text` type and `keyword` type. This is the multi-fields mechanism of ES. The mapping goes as follows:

```
"k4": {
   "type": "text",
   "fields": {
      "keyword": {
         "type": "keyword",
         "ignore_above": 256
      }
   }
}
```

In conditional filtering of k4, "=" filtering for example, Doris-on-ES will convert the query into an ES TermQuery.

SQL filter:

```
k4 = "Doris On ES"
```

Converted query DSL in ES:

```
"term" : {
    "k4": "Doris On ES"

}
```

The primary field type of k4 is `text` so on data ingestion, the designated tokenizer (or the standard tokenizer, if no specification) for k4 will split it into three terms: "doris", "on", and "es".

For example:

```
POST /_analyze
{
  "analyzer": "standard",
  "text": "Doris On ES"
}
```

It will be tokenized as follows:

```
{
   "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
      }
   ]
}
```

The term used in the query is:

```
"term" : {
    "k4": "Doris On ES"
}
```

Since `Doris On ES` does not match any term in the dictionary, no result will be returned. However, if you `enable_keyword_` ↪ `sniff: true` , then `k4 = "Doris On ES"` will be turned into `k4.keyword = "Doris On ES"`. The converted ES query DSL will be:

```
"term" : {
    "k4.keyword": "Doris On ES"
}
```

In this case, `k4.keyword` is of `keyword` type and the data writted into ES is a complete term so the matching can be done.

Enable Node Discovery (nodes_discovery=true)

```
CREATE EXTERNAL TABLE `test` (
  `k1` bigint(20) COMMENT "",
  `k2` datetime COMMENT "",
  `k3` varchar(20) COMMENT "",
  `k4` varchar(100) COMMENT "",
  `k5` float COMMENT ""
) ENGINE=ELASTICSEARCH
PROPERTIES (
"hosts" = "http://192.168.0.1:8200,http://192.168.0.2:8200",
"index" = "test",
"user" = "root",
"password" = "root",
"nodes_discovery" = "true"
);
```

Parameter Description:

Parameter	Description
nodes_discovery	This specifies whether to enable ES node discovery. It is set to true by default.

If this is set to true, Doris will locate all relevant data nodes (the allocated tablets) that are available. If the data node addresses are not accessed by Doris BE, this should be set to false. The deployment of ES clusters is done in an intranet so users require proxy access.

Enable HTTPS Access Mode for ES Clusters (http_ssl_enabled=true)

```
CREATE EXTERNAL TABLE `test` (
  `k1` bigint(20) COMMENT "",
  `k2` datetime COMMENT "",
  `k3` varchar(20) COMMENT "",
  `k4` varchar(100) COMMENT "",
  `k5` float COMMENT ""
) ENGINE=ELASTICSEARCH
PROPERTIES (
"hosts" = "http://192.168.0.1:8200,http://192.168.0.2:8200",
"index" = "test",
"user" = "root",
"password" = "root",
"http_ssl_enabled" = "true"
);
```

Parameter Description:

Parameter	Description
http_ssl_enabled	This specifies whether to enable HTTPS access mode for ES cluster. It is set to false by default.

Currently, the FE and BE implement a trust-all method, which is temporary solution. The actual user configuration certificate will be used in the future.

Query

After creating an ES External Table in Doris, you can query data from ES as simply as querying data in Doris itself, except that you won' t be able to use the Doris data models (rollup, pre-aggregation, and materialized view).

Basic Query

```
select * from es_table where k1 > 1000 and k3 ='term' or k4 like 'fu*z_'
```

Extended esquery (field, QueryDSL)

For queries that cannot be expressed in SQL, such as match_phrase and geoshape, you can use the `esquery(field, QueryDSL` ↪ `)` function to push them down to ES for filtering. The first parameter `field` associates with `index` ; the second one is the Json expression of ES query DSL, which should be surrounded by `{}`. There should be one and only one `root key`, such as match_phrase, geo_shape, and bool.

For example, 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
                     ]
                  }
               }
            ]
         }
      }');
```

2.7.2.1.3 Illustration

```
+----------------------------------------------+
|                                              |
| Doris      +-------------------+             |
|            |       FE          +--------------+-------+
|            |                   |  Request Shard Location
|            +--+-------------+-+              |       |
|               ^             ^                |       |
|               |             |                |       |
|  +-------------------+ +------------------+  |       |
|  |            |      | |    |             |  |       |
|  | +----------+----+ | | +--+-----------+ |  |       |
|  | |      BE       | | | |      BE      | |  |       |
|  | +---------------+ | | +--------------+ |  |       |
+----------------------------------------------+       |
   |        |          | |        |         |          |
   |        |          | |        |         |          |
   |    HTTP SCROLL    | |    HTTP SCROLL   |          |
+-----------+---------------------+------------+       |
|  |        v          | |        v         |  |       |
|  | +------+--------+ | | +------+-------+ |  |       |
|  | |               | | | |              | |  |       |
|  | |   DataNode    | | | |   DataNode   +<-----------+
|  | |               | | | |              | |  |       |
|  | |               +<--------------------------------+
|  | +---------------+ | | |--------------| |  |       |
|  +-------------------+ +------------------+  |       |
|   Same Physical Node                         |       |
|                                              |       |
|           +-----------------------+          |       |
|           |                       |          |       |
|           |      MasterNode       +<-----------------+
| ES        |                       |          |
|           +-----------------------+          |
+----------------------------------------------+
```

1. After an ES External Table is created, Doris FE will send a request to the designated host for information regarding HTTP port and index shard allocation. If the request fails, Doris FE will traverse all hosts until the request succeeds or completely fails.
2. Based on the nodes and metadata in indexes, Doris FE will generate a query plan and send it to the relevant BE nodes.
3. The BE nodes will send requests to locally deployed ES nodes. Via `HTTP Scroll`, BE nodes obtain data in `_source` and `docvalue` concurrently from each tablet in ES index.
4. Doris returns the query results to the user.

2.7.2.1.4 Best Practice

Usage of Time Field

ES allows flexible use of time fields, but improper configuration of time field types can lead to predicate pushdown failure.

When creating an index, allow the greatest format compatibility for time data types:

```
 "dt": {
     "type": "date",
     "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
 }
```

It is recommended to set the corresponding fields in Doris to `date` or `datetime` type (or `varchar`). Then you can use the following SQL statement 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:

- If you don't specify the `format` for time fields in ES, the default format will be:

```
strict_date_optional_time||epoch_millis
```

- Timestamps should be converted to `ms` before they are imported into ES; otherwise errors might occur in Doris-on-ES.

Obtain ES Metadata Field `_id`

You can specify an informative `_id` for a file on ingestion. If not, ES will assign a globally unique `_id` (the primary key) to the file. If you need to acquire the `_id` through Doris-on-ES, you can add a `_id` field of `varchar` type upon table creation.

```
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"
}
```

Note:

1. `_id` fields only support = and `in` filters.
2. `_id` field should be of `varchar` type.

2.7.2.1.5 FAQ

1. What versions of ES does Doris-on-ES support?

Doris-on-ES supports ES 5.x or newer since the data scanning works differently in older versions of ES.

2. Are X-Pack authenticated ES clusters supported？

All ES clusters with HTTP Basic authentication are supported.

3. Why are some queries a lot slower than direct queries oo ES?

For certain queries such as _count, ES can directly read the metadata for the number of files that meet the conditions, which is much faster than reading and filtering all the data.

4. Can aggregation operations be pushed down?

Currently, Doris-on-ES does not support pushing down aggregation operations such as sum, avg, and min/max. Instead, all relevant files from ES will be streamed into Doris in batches, where the computation will be performed.

2.7.2.2 JDBC External Table

> Tips This feature is supported since the Apache Doris 1.2.2 version

Please use JDBC Catalog to access JDBC data sources, this function will no longer be maintained after version 1.2.2.

By creating JDBC External Tables, Doris can access external tables via JDBC, the standard database access inferface. This allows Doris to visit various databases without tedious data ingestion, and give full play to its own OLAP capabilities to perform data analysis on external tables:

> Tips This feature is supported since the Apache Doris 1.2 version

1. Multiple data sources can be connected to Doris;
2. It enables Join queries across Doris and other data sources and thus allows more complex analysis.

This topic introduces how to use JDBC External Tables in Doris.

Create JDBC External Table in Doris

See CREATE TABLE for syntax details.

1. Create JDBC External Table by Creating JDBC_Resource

```
CREATE EXTERNAL RESOURCE jdbc_resource
properties (
    "type"="jdbc",
    "user"="root",
    "password"="123456",
    "jdbc_url"="jdbc:mysql://192.168.0.1:3306/test?useCursorFetch=true",
    "driver_url"="http://IP:port/mysql-connector-java-5.1.47.jar",
    "driver_class"="com.mysql.jdbc.Driver"
);
```

```
CREATE EXTERNAL TABLE `baseall_mysql` (
  `k1` tinyint(4) NULL,
  `k2` smallint(6) NULL,
  `k3` int(11) NULL,
  `k4` bigint(20) NULL,
  `k5` decimal(9, 3) NULL
) ENGINE=JDBC
PROPERTIES (
"resource" = "jdbc_resource",
"table" = "baseall",
"table_type"="mysql"
);
```

Parameter Description:

Parameter	Description
type	"jdbc" ; required; specifies the type of the Resource
user	Username for accessing the external database
password	Password of the user
jdbc_url	JDBC URL protocol, including the database type, IP address, port number, and database name; Please be aware of the different formats of different database protocols. For example, MySQL: "jdbc:mysql://127.0.0.1:3306/test?useCursorFetch=true" .
driver_class	Class of the driver used to access the external database. For example, to access MySQL data: com.mysql.jdbc.Driver.
driver_url	Driver URL for downloading the Jar file package that is used to access the external database, for example, `http://IP:port/mysql-connector-java-5.1.47.jar`. For local stand-alone testing, you can put the Jar file package in a local path: "driver_url" = "file:///home/disk1/pathTo/mysql-connector-java-5.1.47.jar" ; for local multi-machine testing, please ensure the consistency of the paths.
resource	Name of the Resource that the Doris External Table depends on; should be the same as the name set in Resource creation.
table	Name of the external table to be mapped in Doris
table_type	The database from which the external table comes, such as mysql, postgresql, sqlserver, and oracle.

Note:

For local testing, please make sure you put the Jar file package in the FE and BE nodes, too.

In Doris 1.2.1 and newer versions, if you have put the driver in the `jdbc_drivers` directory of FE/BE, you can simply specify the file name in the driver URL: "driver_url" = "mysql-connector-java-5.1.47.jar", and the system will automatically find the file in the jdbc_drivers directory.

Query

```
select * from mysql_table where k1 > 1000 and k3 ='term';
```

In some cases, the keywords in the database might be used as the field names. For queries to function normally in these cases, Doris will add escape characters to the field names and tables names in SQL statements based on the rules of different databases, such as (“) for MySQL, ([]) for SQLServer, and (“”) for PostgreSQL and Oracle. This might require extra attention on case sensitivity. You can view the query statements sent to these various databases via `explain sql`.

Write Data

After creating a JDBC External Table in Doris, you can write data or query results to it using the `insert into` statement. You can also ingest data from one JDBC External Table to another JDBC External Table.

```
insert into mysql_table values(1, "doris");
insert into mysql_table select * from table;
```

Transaction

In Doris, data is written to External Tables in batches. If the ingestion process is interrupted, rollbacks might be required. That’s why JDBC External Tables support data writing transactions. You can utilize this feature by setting the session variable: `enable_↪ odbc_transcation` (ODBC transactions are also controlled by this variable).

```
set enable_odbc_transcation = true;
```

The transaction mechanism ensures the atomicity of data writing to JDBC External Tables, but it reduces performance to a certain extent. You may decide whether to enable transactions based on your own tradeoff.

1.MySQL Test

MySQL Version	MySQL JDBC Driver Version
8.0.30	mysql-connector-java-5.1.47.jar

2.PostgreSQL Test

PostgreSQL Version	PostgreSQL JDBC Driver Version
14.5	postgresql-42.5.0.jar

```
CREATE EXTERNAL RESOURCE jdbc_pg
properties (
    "type"="jdbc",
    "user"="postgres",
    "password"="123456",
    "jdbc_url"="jdbc:postgresql://127.0.0.1:5442/postgres?currentSchema=doris_test",
    "driver_url"="http://127.0.0.1:8881/postgresql-42.5.0.jar",
    "driver_class"="org.postgresql.Driver"
);

CREATE EXTERNAL TABLE `ext_pg` (
  `k1` int
) ENGINE=JDBC
PROPERTIES (
    "resource" = "jdbc_pg",
    "table" = "pg_tbl",
    "table_type"="postgresql"
);
```

3.SQLServer Test

SQLServer Version	SQLServer JDBC Driver Version
2022	mssql-jdbc-11.2.0.jre8.jar

4.Oracle Test

Oracle Version	Oracle JDBC Driver Version
11	ojdbc6.jar

Test information on more versions will be provided in the future.

5.ClickHouse Test

ClickHouse Version	ClickHouse JDBC Driver Version
22	clickhouse-jdbc-0.3.2-patch11-all.jar
22	clickhouse-jdbc-0.4.1-all.jar

6.Sap Hana Test

Sap Hana Version	Sap Hana JDBC Driver Version
2.0	ngdbc.jar

```
CREATE EXTERNAL RESOURCE jdbc_hana
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"
);

CREATE EXTERNAL TABLE `ext_hana` (
  `k1` int
) ENGINE=JDBC
PROPERTIES (
    "resource" = "jdbc_hana",
    "table" = "TEST.HANA",
    "table_type"="sap_hana"
);
```

7.Trino Test

Trino Version	Trino JDBC Driver Version
389	trino-jdbc-389.jar

```
CREATE EXTERNAL RESOURCE jdbc_trino
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"
);

CREATE EXTERNAL TABLE `ext_trino` (
  `k1` int
) ENGINE=JDBC
PROPERTIES (
    "resource" = "jdbc_trino",
    "table" = "hive.test",
    "table_type"="trino"
);
```

8.OceanBase Test

OceanBase Version	OceanBase JDBC Driver Version
3.2.3	oceanbase-client-2.4.2.jar

```
CREATE EXTERNAL RESOURCE jdbc_oceanbase
properties (
    "type"="jdbc",
    "user"="root",
    "password"="",
    "jdbc_url" = "jdbc:oceanbase://localhost:2881/test",
    "driver_url" = "file:///path/to/oceanbase-client-2.4.2.jar",
    "driver_class" = "com.oceanbase.jdbc.Driver",
    "oceanbase_mode" = "mysql" or "oracle"
);

CREATE EXTERNAL TABLE `ext_oceanbase` (
  `k1` int
) ENGINE=JDBC
PROPERTIES (
    "resource" = "jdbc_oceanbase",
    "table" = "test.test",
    "table_type"="oceanbase"
);
```

Note:

When creating an OceanBase external table, you only need to specify the `oceanbase mode` parameter when creating a resource, and the table type of the table to be created is oceanbase

2.7.2.2.1 Type Mapping

The followings list how data types in different databases are mapped in Doris.

MySQL

MySQL	Doris
BOOLEAN	BOOLEAN
BIT(1)	BOOLEAN
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
BIGINT UNSIGNED	LARGEINT
VARCHAR	VARCHAR
DATE	DATE
FLOAT	FLOAT
DATETIME	DATETIME

MySQL	Doris
DOUBLE	DOUBLE
DECIMAL	DECIMAL

PostgreSQL

PostgreSQL	Doris
BOOLEAN	BOOLEAN
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
VARCHAR	VARCHAR
DATE	DATE
TIMESTAMP	DATETIME
REAL	FLOAT
FLOAT	DOUBLE
DECIMAL	DECIMAL

Oracle

Oracle	Doris
VARCHAR	VARCHAR
DATE	DATETIME
SMALLINT	SMALLINT
INT	INT
REAL	DOUBLE
FLOAT	DOUBLE
NUMBER	DECIMAL

SQL server

SQLServer	Doris
BIT	BOOLEAN
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
VARCHAR	VARCHAR
DATE	DATE
DATETIME	DATETIME
REAL	FLOAT
FLOAT	DOUBLE

SQLServer	Doris
DECIMAL	DECIMAL

ClickHouse

ClickHouse	Doris
Boolean	BOOLEAN
String	STRING
Date/Date32	DATE/DATEV2
DateTime/DateTime64	DATETIME/DATETIMEV2
Float32	FLOAT
Float64	DOUBLE
Int8	TINYINT
Int16/UInt8	SMALLINT
Int32/UInt16	INT
Int64/Uint32	BIGINT
Int128/UInt64	LARGEINT
Int256/UInt128/UInt256	STRING
Decimal	DECIMAL/DECIMALV3/STRING
Enum/IPv4/IPv6/UUID	STRING
Array(T)	ARRAY<T>

Note:

- For Array types in ClickHouse, use Doris' s Array type to match them. For basic types in an Array, see Basic type matching rules. Nested arrays are not supported.
- Some data types in ClickHouse, such as UUID, IPv4, IPv6, and Enum8, will be mapped to Varchar/String in Doris. IPv4 and IPv6 will be displayed with an / as a prefix. You can use the `split_part` function to remove the / .
- The Point Geo type in ClickHouse cannot be mapped in Doris by far.

SAP HANA

SAP_HANA	Doris
BOOLEAN	BOOLEAN
TINYINT	TINYINT
SMALLINT	SMALLINT
INTERGER	INT
BIGINT	BIGINT
SMALLDECIMAL	DECIMAL/DECIMALV3
DECIMAL	DECIMAL/DECIMALV3
REAL	FLOAT
DOUBLE	DOUBLE
DATE	DATE/DATEV2

SAP_HANA	Doris
TIME	TEXT
TIMESTAMP	DATETIME/DATETIMEV2
SECONDDATE	DATETIME/DATETIMEV2
VARCHAR	TEXT
NVARCHAR	TEXT
ALPHANUM	TEXT
SHORTTEXT	TEXT
CHAR	CHAR
NCHAR	CHAR

Trino

Trino	Doris
boolean	BOOLEAN
tinyint	TINYINT
smallint	SMALLINT
integer	INT
bigint	BIGINT
decimal	DECIMAL/DECIMALV3
real	FLOAT
double	DOUBLE
date	DATE/DATEV2
timestamp	DATETIME/DATETIMEV2
varchar	TEXT
char	CHAR
array	ARRAY
others	UNSUPPORTED

OceanBase

For MySQL mode, please refer to MySQL type mapping For Oracle mode, please refer to Oracle type mapping

2.7.2.2.2 Q&A

See the FAQ section in JDBC.

2.7.2.3 ODBC External Table

Please use JDBC Catalog to visit external table, this function will no longer be maintained after version 1.2.0.

2.7.2.4 Hive External Table

Please use Hive Catalog to visit Hive, this function will no longer be maintained after version 1.2.0.

2.7.3 File Analysis

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.

2.7.3.1 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

The followings illustrate how file analysis is conducted with the example of S3 Table Value Function.

2.7.3.1.1 Automatic Column Type Inference

```
MySQL [(none)]> DESC FUNCTION s3(
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "ACCESS_KEY"= "minioadmin",
    "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  |
+---------------+--------------+------+-------+---------+-------+
```

An S3 Table Value Function is defined as follows:

```
s3(
    "URI" = "http://127.0.0.1:9312/test2/test.snappy.parquet",
    "ACCESS_KEY"= "minioadmin",
    "SECRET_KEY" = "minioadmin",
    "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',
    'ACCESS_KEY'= 'ak',
    'SECRET_KEY' = 'sk',
    'FORMAT' = 'csv',
    'column_separator' = '|',
    'csv_schema' = 'k1:int;k2:int;k3:int;k4:decimal(38,10)',
    'use_path_style'='true'
)
```

2.7.3.1.2 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",
    "ACCESS_KEY"= "minioadmin",
    "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     |
+-----------+------------------------------------------+----------------+----------+-----------------+
    ↪
```

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.

你也可以用过 `CREATE VIEW` 语句为 Table Value Function 创建一个逻辑视图。这样，你可以想其他视图一样，对这个 Table Value Function 进行访问、权限管理等操作，也可以让其他用户访问这个 Table Value Function。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",
    "ACCESS_KEY"= "minioadmin",
    "SECRET_KEY" = "minioadmin",
    "Format" = "parquet",
    "use_path_style"="true");

DESC v1;

SELECT * FROM v1;
```

```
GRANT SELECT_PRIV ON db1.v1 TO user1;
```

2.7.3.1.3 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" = "${uri}",
    "ACCESS_KEY"= "${ak}",
    "SECRET_KEY" = "${sk}",
    "format" = "${format}",
    "strip_outer_array" = "true",
    "read_json_by_line" = "true",
    "use_path_style" = "true");
```

2.7.4 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.

2.7.4.1 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.

2.7.4.2 Usage

File Cache is disabled by default. You need to set the relevant configuration in FE and BE to enable it.

2.7.4.2.1 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;
```

2.7.4.2.2 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_max_file_segment_size`	Max size of a single cached block, default 4MB, should greater than 4096
`file_cache_path`	Parameters about cache path, json format, for exmaple: `[{"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.
`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

2.7.4.3 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

2.8 Admin Manual

2.8.1 cluster management

2.8.1.1 Cluster upgrade

Doris can upgrade smoothly by rolling upgrades. The following steps are recommended for security upgrade.

The name of the BE binary that appears in this doc is `doris_be`, which was `palo_be` in previous versions.

> Note: 1. Doris does not support upgrading across two-digit version numbers, for example: you cannot upgrade directly from 0.13 to 0.15, only through 0.13.x -> 0.14.x -> 0.15.x, and the three-digit version number can be upgraded across versions, such as from 0.13 .15 can be directly upgraded to 0.14.13.1, it is not necessary to upgrade 0.14.7 or 0.14.12.1 2. The following approaches are based on highly available deployments. That is, data 3 replicas, FE high availability.

2.8.1.1.1 Preparen

1. Turn off the replica repair and balance operation.

   There will be node restarts during the upgrade process, so unnecessary cluster balancing and replica repair logic may be triggered. You can close it first with the following command:

```
    # Turn off the replica ealance logic. After it is closed, the balancing operation of the
        ↪ ordinary table replica will no longer be triggered.
    $ mysql-client> admin set frontend config("disable_balance" = "true");
```

```
# Turn off the replica balance logic of the colocation table. After it is closed, the
    ↪ replica redistribution operation of the colocation table will no longer be triggered
    ↪ .
$ mysql-client> admin set frontend config("disable_colocate_balance" = "true");

# Turn off the replica scheduling logic. After shutting down, all generated replica repair
    ↪ and balancing tasks will no longer be scheduled.
$ mysql-client> admin set frontend config("disable_tablet_scheduler" = "true");
```

```
After the cluster is upgraded, just use the above command to set the corresponding configuration
    ↪  to the original value.
```

2. important! ! Metadata needs to be backed up before upgrading(The entire directory needs to be backed up)! !

2.8.1.1.2 Test the correctness of BE upgrade

1. Arbitrarily select a BE node and deploy the latest doris_be binary file.
2. Restart the BE node and check the BE log be.INFO to see if the boot was successful.
3. If the startup fails, you can check the reason first. If the error is not recoverable, you can delete the BE directly through DROP BACKEND, clean up the data, and restart the BE using the previous version of doris_be. Then re-ADD BACKEND. (This method will result in the loss of a copy of the data, please make sure that three copies are complete, and perform this operation!!!)
4. Install Java UDF function Install Java UDF function: , because Java UDF function is supported from version 1.2, you need to download the JAR package of Java UDF function from the official website and put it in the lib directory of BE, otherwise it may will fail to start.

2.8.1.1.3 Testing FE Metadata Compatibility

0. Important! Exceptional metadata compatibility is likely to cause data cannot be restored!!
1. Deploy a test FE process (It is recommended to use your own local development machine, or BE node. If it is on the Follower or Observer node, you need to stop the started process, but it is not recommended to test on the Follower or Observer node) using the new version alone.
2. Modify the FE configuration file fe.conf for testing and set all ports to different from online.
3. Add configuration in fe.conf: cluster_id=123456
4. Add configuration in fe.conf: metadata_failure_recovery=true
5. Copy the metadata directory doris-meta of the online environment master Fe to the test environment 6.The cluster_ID where copy to the doris-meta/image/VERSION file in the test environment is modified to 123456 (that is, the same as in Step 3)
6. In the test environment,running sh sh bin/start_fe.sh,start FE.
7. Observe whether the start-up is successful through FE log fe.log.
8. If the startup is successful, run sh bin/stop_fe.sh to stop the FE process of the test environment.
9. The purpose of the above 2-6 steps is to prevent the FE of the test environment from being misconnected to the online environment after it starts.

Note: 1.1.x Before upgrading 1.2.x, you need to delete existing Native UDF ; otherwise, FE startup fails ; And since version 1.2 no longer supports Native UDF, please use Java UDF.

2.8.1.1.4 Upgrade preparation

1. After data validation, the new version of BE and FE binary files are distributed to their respective directories.
2. In principle, the version upgrade needs to replace the lib directory and bin directory of FE and BE, and other directories except conf directory, data directory (doris-meta of FE, storage of BE), and log directory.

2.8.1.1.5 rolling upgrade

1. Confirm that the new version of the file is deployed. Restart FE and BE instances one by one.
2. It is suggested that BE be restarted one by one and FE be restarted one by one. Because Doris usually guarantees backward compatibility between FE and BE, that is, the old version of FE can access the new version of BE. However, the old version of BE may not be supported to access the new version of FE.
3. It is recommended to restart the next instance after confirming the previous instance started successfully. Refer to the Installation Deployment Document for the identification of successful instance startup.

2.8.1.1.6 About version rollback

Because the database is a stateful service, Doris cannot support version rollback (version downgrade) in most cases. In some cases, the rollback of the 3-bit or 4-bit version can be supported, but the rollback of the 2-bit version will not be supported.

Therefore, it is recommended to upgrade some nodes and observe the business operation (gray upgrade) to reduce the upgrade risk.

Illegal rollback operation may cause data loss and damage.

2.8.1.2 Elastic scaling

Doris can easily expand and shrink FE, BE, Broker instances.

2.8.1.2.1 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.

2.8.1.2.2 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.

2.8.1.2.3 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.

2.8.1.3 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.

2.8.1.3.1 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.

2.8.1.3.2 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

2.8.1.3.3 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.` dbdatabase 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 -hdoris01
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 10000
    ↪ ms
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.

2.8.1.3.4 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)
```

2.8.1.4 FQDN

2.8.1.4.1 Concept Introduction

A fully qualified domain name (FQDN) is the full domain name of a specific computer or host on the Internet.

After Doris supports FQDN, you can directly specify the domain name when adding various types of nodes. 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_ The host should specify the domain name of the be node.

2.8.1.4.2 Preconditions

1. fe.conf file set `enable_fqdn_mode = true`
2. The fe node can resolve the domain names of all nodes in Doris

2.8.1.4.3 Best Practices

Deployment of Doris for K8S

After an accidental restart of a pod, the K8S cannot ensure that the pod’s IP address does not change, but it can ensure that the domain name remains unchanged. Based on this feature, when Doris starts fqdn, it can ensure that the pod can still provide services normally after an accidental restart. For the method of deploying Doris on the K8S, please refer to K8s Deployment Doris

Server switching network card

For example, a server with a be node has two network cards with corresponding IPs of 192.192.192.2 and 10.10.10.3. Currently, the network card corresponding to 192.192.192.2 is used, and the following steps can be followed:

1. Add a line ‘192.192.192.2 be1’ to the ‘etc/hosts’ file of the machine where fe is located_ fqdn ‘
2. Change be.conf File `priority_Networks=192.192.192.2` and start be
3. Connect and execute the sql command `ALTER SYSTEM ADD BACKEND "be1_fqdn: 9050`

When switching to the network card corresponding to 10.10.10.3 in the future, the following steps can be followed:

1. Configure ‘192.192.192.2 be1’ for ‘etc/hosts’ _ Fqdn ‘is changed to’ 10.10.10.3 be1_ fqdn ‘

Legacy Cluster Enable FQDN

Prerequisite: The current program supports the ’ALTER SYSTEM MODIFY FRONT “:” HOSTNAME “” syntax, If not, you need to upgrade to a version that supports this syntax

Next, follow the steps below:

1. Perform the following operations on the follower and observer nodes one by one (finally, operate the master node):
    1. Stop the node
    2. Check if the node is stopped. Execute ‘show frontends’ on 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>"`
    4. Modify the node configuration. Modify the ‘conf/fe. conf’ file in the FE root directory and add the configuration: ‘enable’ _ fqdn_ mode = true ‘
    5. Start the node.
2. To enable FQDN for a BE node, you only need to execute the following commands through MYSQL, and there is no need to restart the BE.

```
ALTER SYSTEM MODIFY BACKEND "<backend_ip>:<backend_port>" HOSTNAME "<be_hostname>"
```

2.8.2 Data Admin

2.8.2.1 Data Backup

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;`.

2.8.2.1.1 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.

2.8.2.1.2 Start Backup

1. Create a hdfs remote warehouse example_repo:

```
CREATE REPOSITORY `example_repo`
WITH BROKER `hdfs_broker`
ON LOCATION "hdfs://hadoop-name-node:54310/path/to/repo/"
PROPERTIES
(
   "username" = "user",
   "password" = "password"
);
```

2. Create a remote repository for s3 : s3_repo

```
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

1. 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");
```

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
);
```

4. 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)
```

5. 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.

2.8.2.1.3 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.

2.8.2.1.4 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. Common `TaskErrMsg` errors are as follows: Q1: Backup to HDFS, the status shows UPLOADING, TaskErrMsg error message: [13333: Close broker writer failed, broker:TNetworkAddress(hostname=10.10.0.0, port=8000) msg:errors while close file output stream, cause by: DataStreamer Exception :] This is generally a network communication problem. Check the broker log to see if a certain ip or port is blocked. If it is a cloud service, you need to check whether is accessed the intranet. If so, you can add hdfs-site.xml in the broker/conf folder, you need to add dfs.client.use.datanode.hostname=true under the hdfs-site.xml configuration file, and configure the hostname mapping of the HADOOP cluster on the broker node.

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.

2.8.2.1.5 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.

2. 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, or directly back up to HDFS, please refer to Create Remote Warehouse Documentation

3. BACKUP

   Perform a backup operation.

4. SHOW BACKUP

   View the execution of the most recent backup job, including:

   - JobId: The id of this backup job.
   - SnapshotName: The name (Label) of this backup job specified by the user.
   - DbName: Database corresponding to the backup job.
   - State: The current stage of the backup job:
     - PENDING: The initial status of the job.
     - SNAPSHOTING: A snapshot operation is in progress.
     - UPLOAD_SNAPSHOT: The snapshot is over, ready to upload.
     - UPLOADING: Uploading snapshot.
     - SAVE_META: The metadata file is being generated locally.
     - UPLOAD_INFO: Upload metadata files and information about this backup job.
     - FINISHED: The backup is complete.
     - CANCELLED: Backup failed or was canceled.
   - BackupObjs: List of tables and partitions involved in this backup.
   - CreateTime: Job creation time.
   - SnapshotFinishedTime: Snapshot completion time.
   - UploadFinishedTime: Snapshot upload completion time.
   - FinishedTime: The completion time of this job.
   - UnfinishedTasks: During `SNAPSHOTTING`, `UPLOADING` 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.

5. SHOW SNAPSHOT

   View existing backups in the remote repository.

   - Snapshot: The name (Label) of the backup specified during backup.
   - Timestamp: Timestamp of the backup.
   - Status: Whether the backup is normal.

   More detailed backup information can be displayed if a where clause is specified after `SHOW SNAPSHOT`.

   - Database: The database corresponding to the backup.
   - Details: Shows the complete data directory structure of the backup.

6. CANCEL BACKUP

   Cancel the currently executing backup job.

7. 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.

2.8.2.1.6 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.

2.8.2.2 Data Restore

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;`.

2.8.2.2.1 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.

2.8.2.2.2 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.

2.8.2.2.3 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.

2.8.2.2.4 Common mistakes

1. Restore Report An Error:[20181: invalid md5 of downloaded file: /data/doris.HDD/snapshot/20220607095111.862.86400/19962/668322732/199
   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

2.8.2.2.5 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.

2.8.2.3 Data Recover

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.

2.8.2.3.1 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;
```

2.8.2.3.2 More Help

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.

2.8.3 Sql Interception

This function is only used to limit the query statement, and does not limit the execution of the explain 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.

2.8.3.1 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 RULR

  ```
  DROP SQL_BLOCK_RULE test_rule1,test_rule2
  ```

2.8.3.2 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'
```

2.8.4 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 frgment 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

2.8.4.1 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.

2.8.4.2 Basic concepts

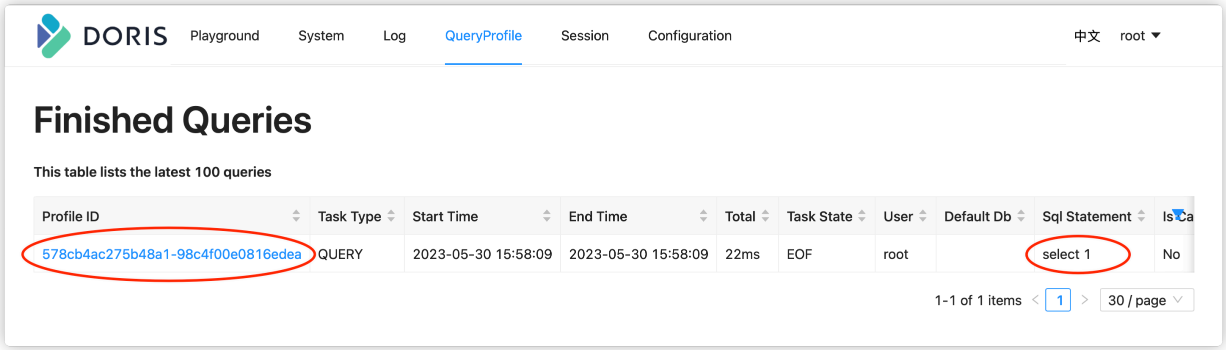
FE splits the query plan into fragments and distributes them to BE for task execution. BE records the statistics of Running State when executing fragment. BE print the outputs statistics of fragment execution into the log. FE can also collect these statistics recorded by each fragment and print the results on FE' s web page.

2.8.4.3 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.



Finished Queries

This table lists the latest 100 queries

Profile ID	Task Type	Start Time	End Time	Total	Task State	User	Default Db	Sql Statement	Is Ca
578cb4ac275b48a1-98c4f00e0816edea	QUERY	2023-05-30 15:58:09	2023-05-30 15:58:09	22ms	EOF	root		select 1	No

1-1 of 1 items 1 30 / page

Figure 24: image.png

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.

2.8.4.4 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 writen 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 duplicatenode 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
    ↪ scaner thread to return rowbatch.
- 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:
    - BitmapIndexFilterTimer: 779ns   # Use bitmap 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.
    - RowsBitmapIndexFiltered: 0      # Only in V2, the number of rows filtered by the Bitmap
        ↪ 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 Bitmap index to perform precise filtering on the columns containing the bitmap index in the query condition, and the number of filtered rows is recorded in `RowsBitmapIndexFiltered`.
- 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 Bitmap index) and the number of remaining rows after BloomFilter, so the data filtered by BloomFilter may overlap with the data filtered by Bitmap.
- 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

2.8.5 tracing

Tracing records the life cycle of a request execution in the system, including the request and its sub-procedure call links, execution time and statistics, which can be used for slow query location, performance bottleneck analysis, etc.

> Please note that Tracing is only supported in Doris 1.2 and is no longer avialable since Doris 2.0.

2.8.5.1 Principle

doris is responsible for collecting traces and exporting them to a third-party tracing analysis system, which is responsible for the presentation and storage of traces.

2.8.5.2 Quick Start

doris currently supports exporting traces directly to zipkin.

2.8.5.2.1 Deploy zipkin

```
curl -sSL https://zipkin.io/quickstart.sh | bash -s
java -jar zipkin.jar
```

2.8.5.2.2 Configuring and starting Doris

Add configuration to fe.conf

```
enable_tracing = true


trace_export_url = http://127.0.0.1:9411/api/v2/spans
```

Add configuration to be.conf

```
enable_tracing = true

### Configure traces to export to zipkin
trace_export_url = http://127.0.0.1:9411/api/v2/spans
```

```
### Queue size for caching spans. span export will be triggered once when the number of spans
    ↪ reaches half of the queue capacity. spans arriving in the queue will be discarded when
    ↪ the queue is full.
max_span_queue_size=2048

### The maximum number of spans to export in a single pass.
max_span_export_batch_size=512

### Maximum interval for exporting span
export_span_schedule_delay_millis=500
```

Start fe and be

```
sh fe/bin/start_fe.sh --daemon
sh be/bin/start_be.sh --daemon
```

2.8.5.2.3 Executing a query

```
...
```

2.8.5.2.4 View zipkin UI

The browser opens `http://127.0.0.1:9411/zipkin/` to view the query tracing.

2.8.5.3 Using opentelemetry collector

Use the opentelemetry collector to export traces to other systems such as zipkin, jaeger, skywalking, or to database systems and files. For more details, refer to collector exporter.

Meanwhile, opentelemetry collector provides a rich set of operators to process traces. For example, filterprocessor , tailsampling-processor. For more details, refer to collector processor.

traces export path: doris->collector->zipkin etc.

2.8.5.3.1 Deploy opentelemetry collector

opentelemetry has released collector core and contrib, contrib provides richer features, here is an example of contrib version.

Download collector

Download otelcol-contrib, available on the official website more precompiled versions for more platforms

```
wget https://github.com/open-telemetry/opentelemetry-collector-releases/releases/download/v0
    ↪ .55.0/otelcol-contrib_0.55.0_linux_amd64.tar.gz

tar -zxvf otelcol-contrib_0.55.0_linux_amd64.tar.gz
```

Generate configuration file

The collector configuration file is divided into 5 parts: `receivers`, `processors`, `exporters`, `extensions`, and `service`. Among them, receivers, processors and exporters define the way to receive, process and export data respectively; extensions are optional and are used to extend tasks that do not involve processing telemetry data; service specifies which components are used in the collector. See collector configuration.

The following configuration file uses the otlp (OpenTelemetry Protocol) protocol to receive traces data, perform batch processing and filter out traces longer than 50ms, and finally export them to zipkin and file.

```
cat > otel-collector-config.yaml << EOF
receivers:
  otlp:
    protocols:
      http:

exporters:
  zipkin:
    endpoint: "http://10.81.85.90:8791/api/v2/spans"
  file:
    path: ./filename.json

processors:
  batch:
  tail_sampling:
    policies:
      {
        name: duration_policy,
        type: latency,
        latency: {threshold_ms: 50}
      }

extensions:

service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch, tail_sampling]
      exporters: [zipkin, file]
EOF
```

Start collector

```
nohup ./otelcol-contrib --config=otel-collector-config.yaml &
```

2.8.5.3.2 Configuring and starting Doris

Add configuration to fe.conf

```
enable_tracing = true

### enable opentelemetry collector
trace_exporter = collector

### Configure traces export to collector, 4318 is the default port for collector otlp http
trace_export_url = http://127.0.0.1:4318/v1/traces
```

Add configuration to be.conf

```
enable_tracing = true

### enable opentelemetry collector
trace_exporter = collector

### Configure traces export to collector, 4318 is the default port for collector otlp http
trace_export_url = http://127.0.0.1:4318/v1/traces

### Queue size for caching spans. span export will be triggered once when the number of spans
    ↪ reaches half of the queue capacity. spans arriving in the queue will be discarded when
    ↪ the queue is full.
max_span_queue_size=2048

### The maximum number of spans to export in a single pass.
max_span_export_batch_size=512

### Maximum interval for exporting span
export_span_schedule_delay_millis=500
```

Start fe and be

```
sh fe/bin/start_fe.sh --daemon
sh be/bin/start_be.sh --daemon
```

2.8.5.3.3 Executing a query

```
...
```

2.8.5.3.4 View zipkin UI

```
...
```

2.8.6 performance optimization

2.8.7 TLS certificate

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/`.

2.8.7.1 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.

```
### 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
```

reference documents

2.8.8 FE SSL certificate

Certificate Configuration

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`

2.8.9 Maintenance and Monitor

2.8.9.1 Monitor

2.8.9.1.1 Monitor Metrics

(TODO) There is no English document, please visit the Chinese document.

2.8.9.1.2 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.

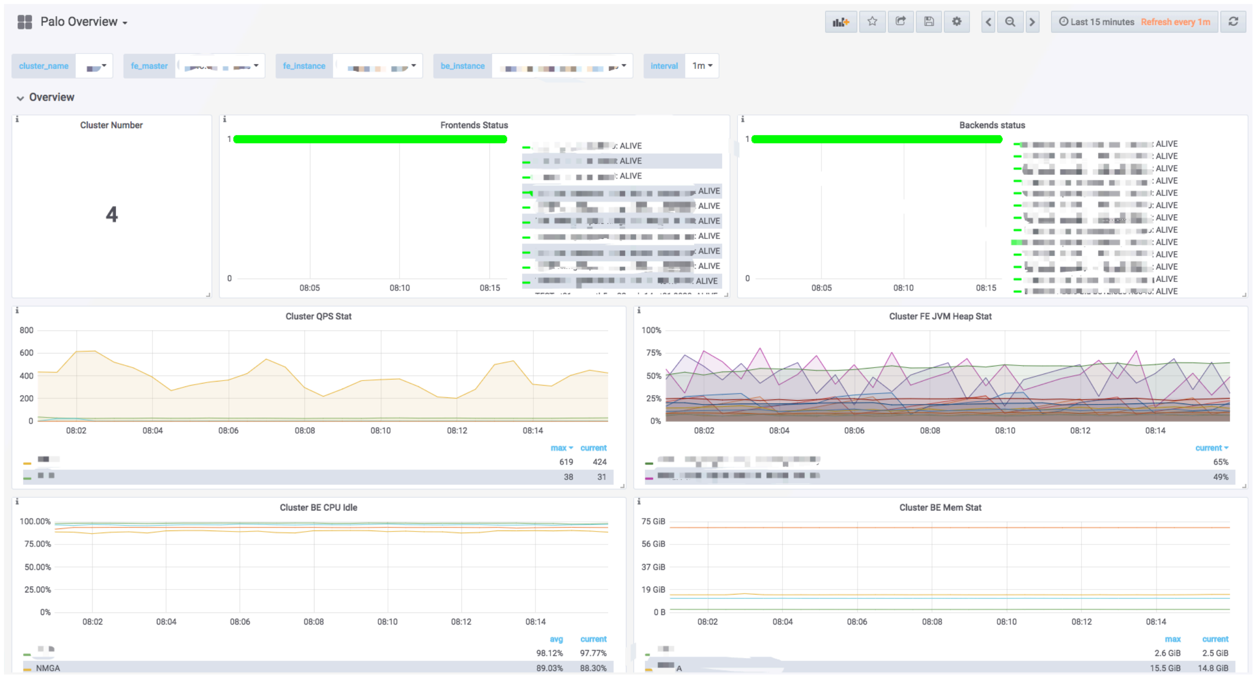


Figure 25: 组件

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:

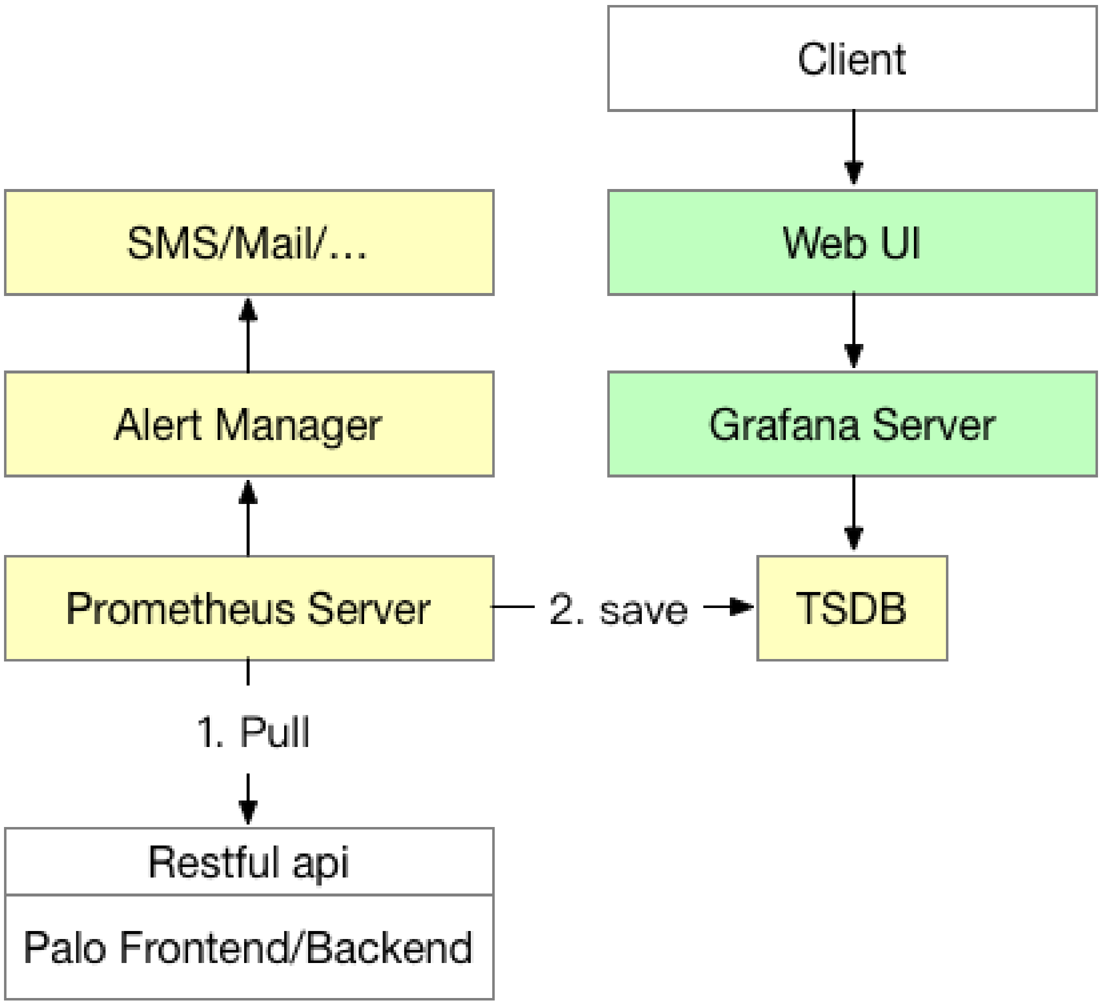


Figure 26: Monitoring Architecture

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

   `nohuo ./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

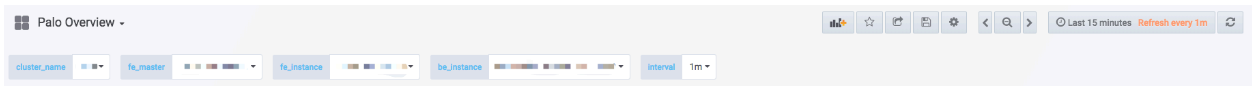


Figure 27: Top Bar

- 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.

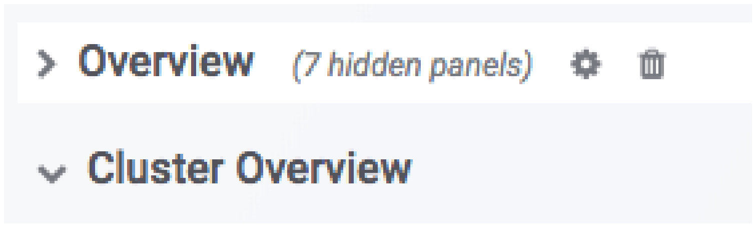


Figure 28: Row

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

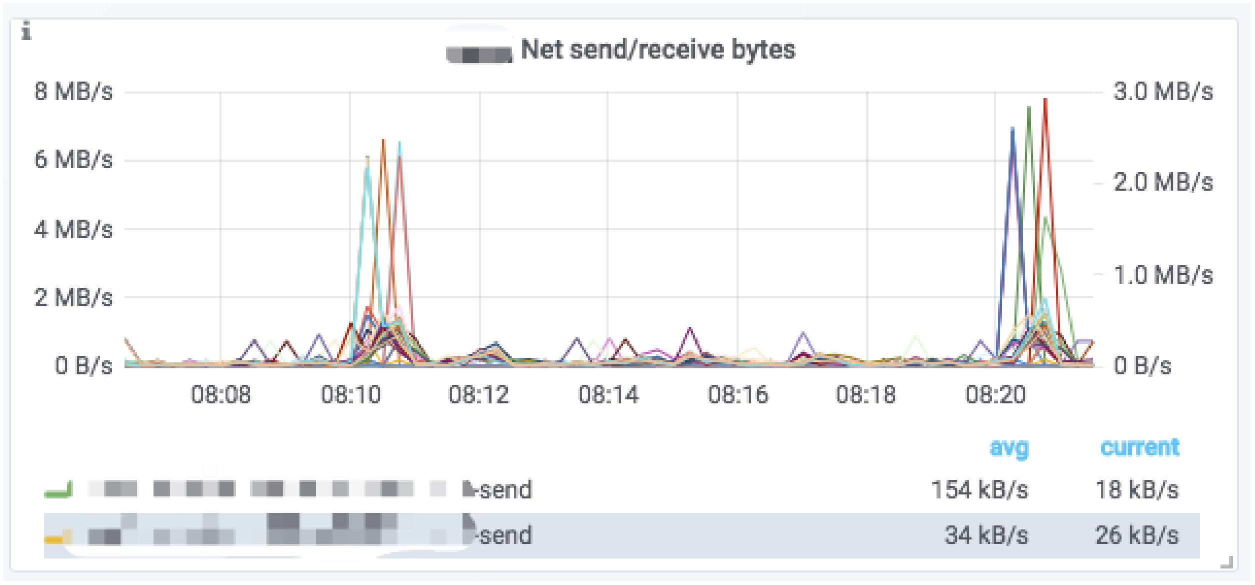


Figure 29: Charts

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.

2.8.9.2 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.

2.8.9.2.1 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.

2.8.9.2.2 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.

2.8.9.2.3 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

2.8.9.2.4 BE Parameter

Flood Stage:

```
capacity_used_percent_flood_stage: default value is 95 (95%).
capacity_min_left_bytes_flood_stage: 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.)

2.8.9.2.5 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
```

2.8.9.3 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 Collocation attributes can be referred to HERE

2.8.9.3.1 Noun Interpretation

1. Tablet: The logical fragmentation of a Doris table, where a table has multiple fragmentations.
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.

```

              +--------+              +-----------+
              |  Meta  |              |  Backends |
              +---^----+              +------^----+
                  | |                        | 3. Send clone tasks
 1. Check tablets | |                        |
           +--------v------+        +-----------------+
           | TabletChecker +--------> TabletScheduler |
           +---------------+        +-----------------+
                   2. Waiting to be scheduled
```

The figure above is a simplified workflow.

2.8.9.3.2 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.

2.8.9.3.3 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 REPLICAMISSING.

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.

2.8.9.3.4 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.

2.8.9.3.5 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.

2.8.9.3.6 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           |
| 13402 | default_cluster:ssb1           | 709       | 708        | 1                 | 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           |
| Total | 3                              | 1071      | 1070       | 1                 | 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.
```

4. 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 REPLICAMISSING (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.

2.8.9.3.7 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           | 2.459007474009069
        ↪ E7   |
    | 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           | 2.417217089806745
        ↪ E7   |
    | 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.

2.8.9.3.8 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-locotion 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 1048576000 (1GB)
  - 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.

2.8.9.3.9 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.

2.8.9.3.10 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.

2.8.9.4 Tablet metadata management tool

2.8.9.4.1 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.

2.8.9.4.2 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
```

2.8.9.5 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.

2.8.9.5.1 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/SchemHash.

```
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.

2.8.9.5.2 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.

2.8.9.6 Tablet Restore Tool

2.8.9.6.1 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
```

2.8.9.6.2 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.

2.8.9.7 Metadata Operations and Maintenance

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.

2.8.9.7.1 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.

2.8.9.7.2 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).

2.8.9.7.3 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.

2.8.9.7.4 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.

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. 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)
    2. Add configuration in fe.conf: `metadata_failure_recovery=true`.
    3. Run `sh bin/start_fe.sh --daemon` to start the FE
    4. 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.
    5. 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.
    6. If successful, through the `show frontends;` command, you should see all the FEs you added before, and the current FE is master.
    7. Delete the `metadata_failure_recovery=true` configuration item in fe.conf, or set it to `false`, and restart the FE (Important).

> 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 = true` 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

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=true`.

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.

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.

2.8.9.7.5 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.

2.8.9.7.6 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 memroy 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.

2.8.10 Memory Management

2.8.10.1 Memory Tracker

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.

2.8.10.1.1 principle

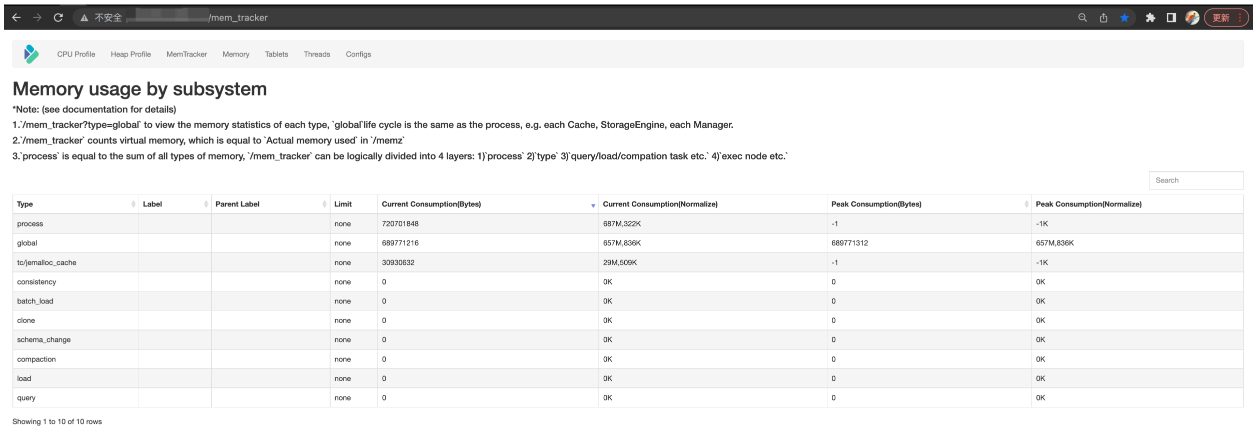
Each query, import and other tasks in the system will create its own Memory Tracker when it is initialized, and put the Memory Tracker into TLS (Thread Local Storage) during execution, and each memory application and release of the BE process will be in the Mem Hook Consume the Memory Tracker in the 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+
https://shimo.im/docs/DT6JXDRkdTvdyV3G

2.8.10.1.2 View statistics

The real-time memory statistics results can be viewed through Doris BE's Web page `http://ip:http_port/mem_tracker`. 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`



Memory usage by subsystem

*Note: (see documentation for details)

1.`/mem_tracker?type=global` to view the memory statistics of each type, `global`life cycle is the same as the process, e.g. each Cache, StorageEngine, each Manager.

2.`/mem_tracker` counts virtual memory, which is equal to `Actual memory used` in `/memz`

3.`process` is equal to the sum of all types of memory, `/mem_tracker` can be logically divided into 4 layers: 1)`process` 2)`type` 3)`query/load/compation task etc.` 4)`exec node etc.`

Type	Label	Parent Label	Limit	Current Consumption(Bytes)	Current Consumption(Normalize)	Peak Consumption(Bytes)	Peak Consumption(Normalize)
process			none	720701848	687M,322K	-1	-1K
global			none	689771216	657M,836K	689771312	657M,836K
tc/jemalloc_cache			none	30930632	29M,509K	-1	-1K
consistency			none	0	0K	0	0K
batch_load			none	0	0K	0	0K
clone			none	0	0K	0	0K
schema_change			none	0	0K	0	0K
compaction			none	0	0K	0	0K
load			none	0	0K	0	0K
query			none	0	0K	0	0K

Showing 1 to 10 of 10 rows

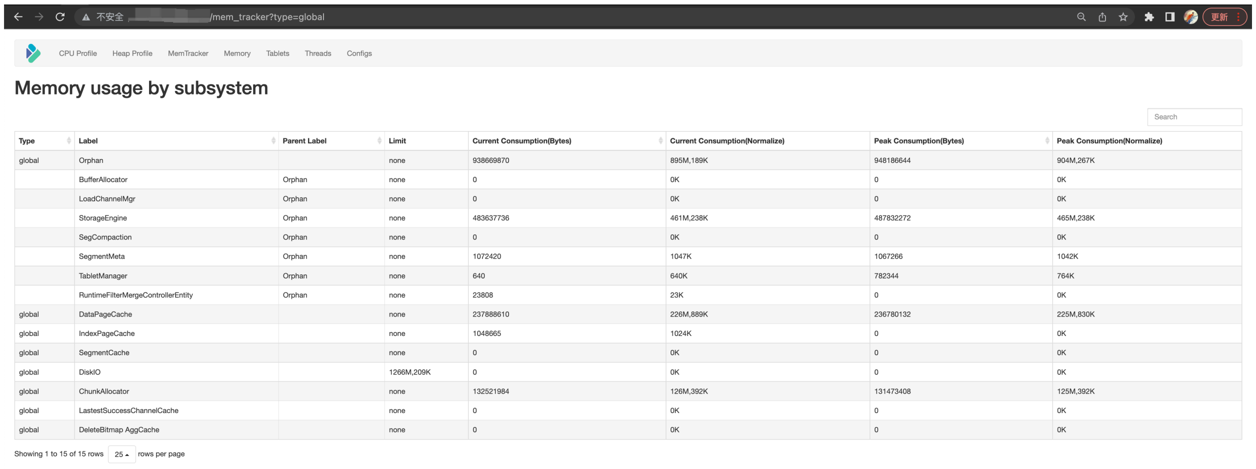
Figure 30: image

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://ip:http_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`



Memory usage by subsystem

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

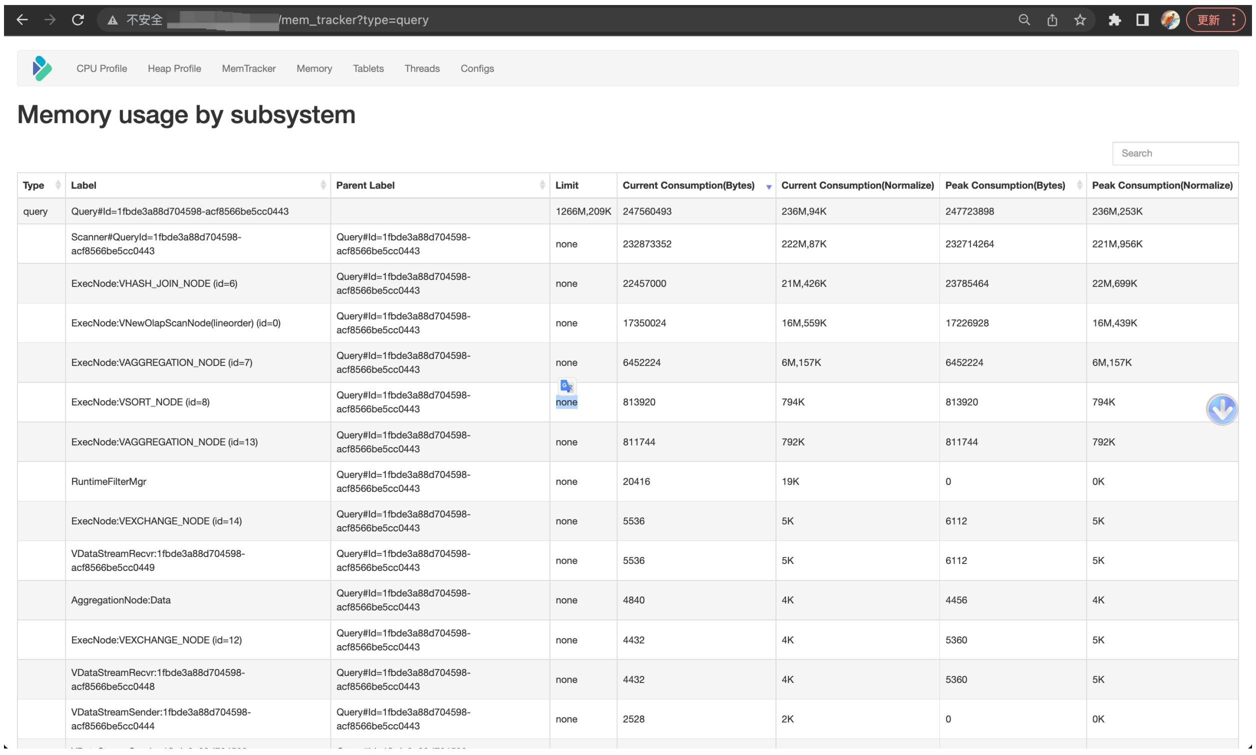
Showing 1 to 15 of 15 rows 25 rows per page

Figure 31: 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 delte Tablet, a subset of Orphan.
- BufferAllocator: Only used for memory multiplexing in the non-vectorized Partitioned Agg process, 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.
- DiskIO: Used to cache Disk IO data, only used in non-vectorization.
- 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`



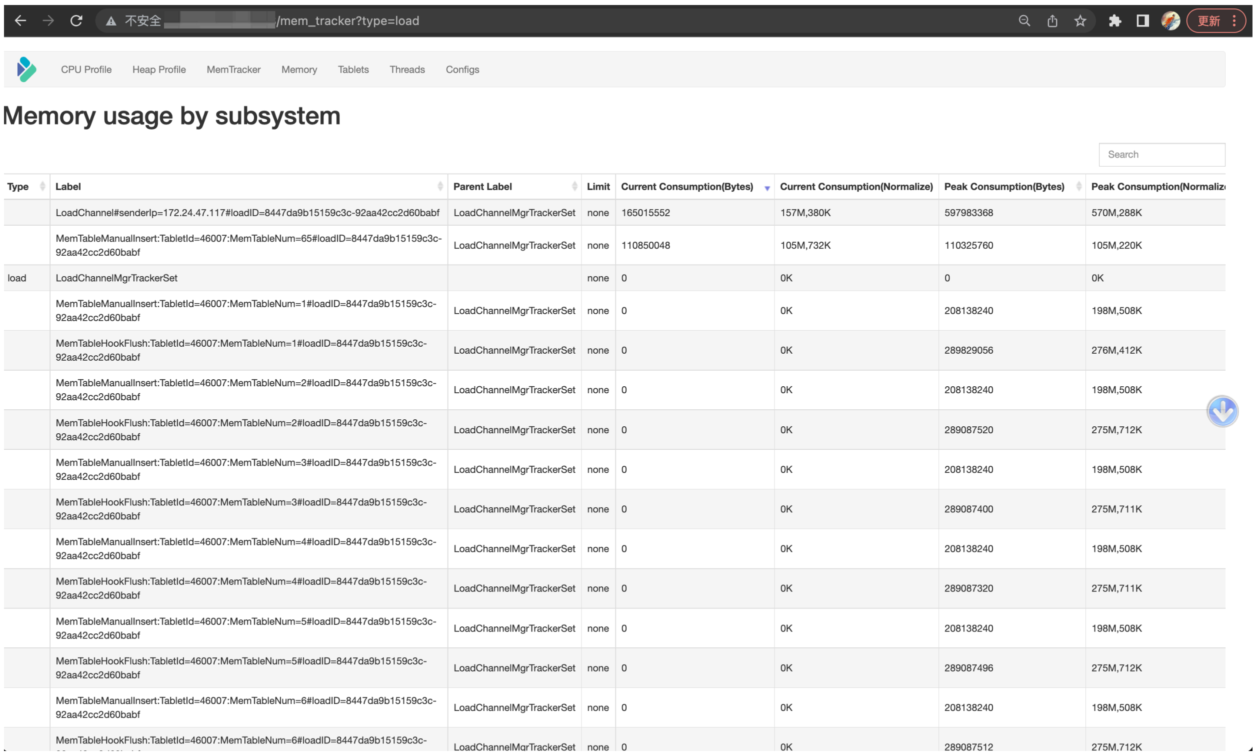
Memory usage by subsystem

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

Figure 32: 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`



不安全 [redacted]/mem_tracker?type=load

CPU Profile Heap Profile MemTracker Memory Tablets Threads Configs

Memory usage by subsystem

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

Figure 33: 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.

2.8.10.2 Memory Limit Exceeded Analysis

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.
```

2.8.10.2.1 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 Faild, 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.

2.8.10.2.2 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 ex 98.92 GB cery tracker If , `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 veriables, 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=8G`to 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 attention`Alloc 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=78208
        ↪ b15e064527-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=78208
        ↪ b15e064527-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)
```

2.8.10.3 BE OOM Analysis

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.

2.8.10.3.1 Solution

Refer to BE Configuration Items to reduce `mem_limit` and increase `max_sys_mem_available_low_water_mark_bytes` in `be`
↪ `.conf`.

2.8.10.3.2 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;

2.8.11 Config

2.8.11.1 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.

2.8.11.1.1 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.

2.8.11.2 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.

2.8.11.2.1 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.

2.8.11.2.2 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.

2.8.11.2.3 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.

2.8.11.2.4 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： 1

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/sleepyca

`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_deploy_manager`

Default: disable

Set to true if you deploy Doris using thirdparty deploy manager

Valid options are:

- disable: no deploy manager
- k8s: Kubernetes
- ambari: Ambari
- local: Local File (for test or Boxer2 BCC version)

`with_k8s_certs`

Default**:** false

If use k8s deploy manager locally, set this to true and prepare the certs files

`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

Service

`query_port`

Default**:** 9030

FE MySQL 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.

`max_connection_scheduler_threads_num`

Default: 4096

Maximal number of thread in connection-scheduler-pool.

The current strategy is to apply for a separate thread for service when there is a request

`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.

`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

When FeEstarts the MySQL server based on NIO model, the number of threads responsible for Task events. Only `mysql_service` ↪ `_nio_enabled` is true takes effect.

`mysql_service_io_threads_num`

Default: 4

When FeEstarts 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 hours）

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: 5000 （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: 10240 （10K）

http header size configuration parameter, the default value is 10K

`enable_tracing`

Default: false

IsMutable: false

MasterOnly: false

Whether to enable tracking

If this configuration is enabled, you should also specify the trace_export_url.

`trace_exporter`

Default: zipkin

IsMutable: false

MasterOnly: false

Current support for exporting traces: zipkin: Export traces directly to zipkin, which is used to enable the tracing feature quickly. collector: The collector can be used to receive and process traces and support export to a variety of third-party systems. If this configuration is enabled, you should also specify the enable_tracing=true and trace_export_url.

`trace_export_url`

Default: `http://127.0.0.1:9411/api/v2/spans`

IsMutable: false

MasterOnly: false

trace export to zipkin like: `http://127.0.0.1:9411/api/v2/spans`

trace export to collector like: `http://127.0.0.1:4318/v1/traces`

`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： 1

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

`max_multi_partition_num`

Default： 4096

IsMutable： false

MasterOnly： true

Used to limit the maximum number of partitions that can be created when multi creating partitions, to avoid creating too many partitions at one time.

`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, 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 2000 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 v_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_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: `3 * 1024 * 1024 * 1024L` (3G)

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: 5

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 ‘routine_load_thread_pool_size’ (default 10), which is the routine load task thread pool size on BE.

`max_routine_load_task_concurrent_num`

Default: 5

IsMutable: true

MasterOnly: true

the max concurrent routine load task num of a single routine load job

`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.

`load_straggler_wait_second`

Default: 300

IsMutable: true

MasterOnly: true

Maximal wait seconds for straggler node in load eg. there are 3 replicas A, B, C load is already quorum finished(A,B) at t1 and C is not finished if (current_time - t1) > 300s, then palo will treat C as a failure node will call transaction manager to commit the transaction and tell transaction manager that C is failed

This is also used when waiting for publish tasks

Note: this parameter is the default value for all job and the DBA could specify it for separate job

`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.

`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`

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 feature allows users to separate hot and cold data. This feature is still under development. Recommended for test environments only.

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.streami
}

`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=
} }

`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

`backup_upload_task_num_per_be`

Default: 3

IsMutable: true

MasterOnly: true

The max number of upload tasks assigned to each be during the backup process, the default value is 3.

`restore_download_task_num_per_be`

Default: 3

IsMutable: true

MasterOnly: true

The max number of download tasks assigned to each be during the restore process, the default value is 3.

`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.

`enable_quantile_state_type`

Default: false

IsMutable: true

MasterOnly: true

Whether to enable the quantile_state data type

`enable_date_conversion`

Default: false

IsMutable: true

MasterOnly: true

If set to TRUE, FE will convert date/datetime to datev2/datetimev2(0) automatically.

`enable_decimal_conversion`

Default: false

IsMutable: true

MasterOnly: true

If set to 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`

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.

2.8.11.3 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.

2.8.11.3.1 View configuration items

Users can view the current configuration items by visiting BE’s web page:

`http://be_host:be_webserver_port/varz`

2.8.11.3.2 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
```

2.8.11.3.3 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
```

2.8.11.3.4 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

`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

`single_replica_load_brpc_port`

- Type: int32
- Description: The port of BRPC on BE, used for single replica load. There is a independent BRPC thread pool for the communication between the Master replica and Slave replica during single replica load, which prevents data synchronization between the replicas from preempt the thread resources for data distribution and query tasks when the load concurrency is large.
- Default value: 9070

`single_replica_load_download_port`

- Type: int32
- Description: The port of http for segment download on BE, used for single replica load. There is a independent HTTP thread pool for the Slave replica to download segments during single replica load, which prevents data synchronization between the replicas from preempt the thread resources for other http tasks when the load concurrency is large.
- Default value: 8050

`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 to many the machine’s memory. This parameter must be greater than 0. When the percentage is greater than 100%, the value will default to 100%.
  - `auto` means process mem limit is equal to max(physical_mem * 0.9, physical_mem - 6.4G), 6.4G is the maximum memory reserved for the system by default.
- Default value: auto

`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

`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 -1, which means the number of bthreads is #cpu-cores.
- User can set this configuration to a larger value to get better QPS performance. For more information, please refer to `https` `↪ ://github.com/apache/incubator-brpc/blob/master/docs/cn/benchmark.md`

- Default value: -1

`thrift_rpc_timeout_ms`

- Description: thrift default timeout time
- Default value: 10000

`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.

`txn_commit_rpc_timeout_ms`

- Description:txn submit rpc timeout
- Default value: 10,000 (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: 2048

`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: 512

`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_queue_size`

- Type: int32

- Description: The length of the RowBatch buffer queue between TransferThread and OlapScanner. When Doris performs data scanning, it is performed asynchronously. The Rowbatch scanned by OlapScanner will be placed in the scanner buffer queue, waiting for the upper TransferThread to take it away.
- Default value: 1024

`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

`serialize_batch`

- Type: bool

- Description: Whether the rpc communication between BEs serializes RowBatch for data transmission between query layers
- Default value: false

`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: bool
- Description: In vertical compaction, column number for every group
- Default value: true

`vertical_compaction_max_row_source_memory_mb`

- Type: bool
- Description: In vertical compaction, max memory usage for row_source_buffer
- Default value: true

`vertical_compaction_max_segment_size`

- Type: bool
- Description: In vertical compaction, max dest segment file size
- Default value: true

`enable_ordered_data_compaction`

- Type: bool
- Description: Whether to enable ordered data compaction
- Default value: true

`ordered_data_compaction_min_segment_size`

- Type: bool
- Description: In ordered data compaction, min segment size for input rowset
- Default value: true

`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] Trace:
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
Metrics: {"filtered_rows":0,"input_row_num":3346807,"input_rowsets_count":42,"input_rowsets_data_
    ↪ size":1256413170,"input_segments_num":44,"merge_rowsets_latency_us":101574444,"merged_
    ↪ rows":0,"output_row_num":3346807,"output_rowset_data_size":1228439659,"output_segments_
    ↪ num":6}
```

`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: 2

`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: 4

`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_threshold_segment_num`

- Type: int32
- Description: Trigger segcompaction if the num of segments in a rowset exceeds this threshold
- Default value: 10

`segcompaction_small_threshold`

- Type: int32

- Description: The segment whose row number above the threshold will be compacted during segcompaction
- Default value: 1048576

`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

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

`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

`single_replica_load_brpc_num_threads`

- Type: int32
- Description: This configuration is mainly used to modify the number of bthreads for single replica load brpc. When the load concurrency increases, you can adjust this parameter to ensure that the Slave replica synchronizes data files from the Master replica timely.
- Default value: 64

`single_replica_load_download_num_workers`

- Type: int32
- Description:This configuration is mainly used to modify the number of http 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.
- Default value: 64

`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

`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_limitation_per_thread_for_schema_change`

- Description: The maximum memory allowed for a single schema change task
- Default value: 2 (GB)

`memory_max_alignment`

- Description: Maximum alignment memory
- Default value: 16

`mmap_buffers`

- Description: Whether to use mmap to allocate memory
- Default value: false

`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

`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: 16MB

`segment_cache_capacity`

- Type: int32
- Description: The maximum number of Segments cached by Segment Cache.
- The default value is currently only an empirical value, and may need to be modified according to actual scenarios. Increasing this value can cache more segments and avoid some IO. Decreasing this value will reduce memory usage.
- Default value: 1000000

`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: 用于保存 SmallFileMgr 下载的文件的目录
- 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

`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 2ˆn, 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 and the disk time is insufficient, you can configure less tablet_rowset_stale_sweep_time_sec. However, if this time is less than 5 minutes, it may cause fe to query the version that has been merged, causing a query -230 error.
- Default value: 1800

`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

`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

`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: 空

`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: 空

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: 100

`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

`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: false

2.8.11.4 User Property

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.

2.8.11.4.1 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;`.

2.8.11.4.2 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.

2.8.11.4.3 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.

2.8.11.4.4 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

2.8.12 User Privilege and Ldap

2.8.12.1 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.

2.8.12.1.1 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.

2.8.12.1.2 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.

```
┌--------┐        ┌--------┐         ┌--------┐
│  user1 ├----┬---□  role1 ├----┬----□  priv1 │
└--------┘    │   └--------┘    │    └--------┘
              │                 │
              │                 │
              │   ┌--------┐    │
              │   │  role2 ├----┤
┌--------┐    │   └--------┘    │    ┌--------┐
│  user2 ├----┘                 │  ┌─□  priv2 │
└--------┘                      │  │ └--------┘
                  ┌--------┐    │  │
```

```
          ┌------►  role3 ├----┘   │
          │      └--------┘        │
          │                        │
          │                        │
┌--------┐ │      ┌--------┐        │ ┌--------┐
│  userN ├-┴------►  roleN ├--------┴-►  privN │
└--------┘        └--------┘          └--------┘
```

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.

2.8.12.1.3 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`.

2.8.12.1.4 Permission type

Doris currently supports the following permissions

1. Node_priv

   Nodes change permissions. Including FE, BE, BROKER node addition, deletion, offline operations. Currently, this permission can only be granted to Root 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.

2.8.12.1.5 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.

2.8.12.1.6 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.

2.8.12.1.7 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.

2.8.12.1.8 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`.

2.8.12.1.9 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.

2.8.12.2 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.

2.8.12.2.1 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.

2.8.12.2.2 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.
- 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.

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

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. “ ‘sql mysql -hDORIS_HOST -PDORIS_PORT -u user -p –enable-cleartext-plugin

Enter ldap password “ ‘

2.8.12.2.3 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 DLAP:

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.

2.8.12.2.4 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.

2.8.13 System Table

2.8.13.1 rowsets

2.8.13.1.1 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 | 02000000000000994847fbd41a42297d7c7a57d3bcb46f8c |     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 | 02000000000000894847fbd41a42297d7c7a57d3bcb46f8c |     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 | 02000000000000884847fbd41a42297d7c7a57d3bcb46f8c |     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 | 02000000000000874847fbd41a42297d7c7a57d3bcb46f8c |     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
```

2.8.14 Multi-tenancy

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.

2.8.14.1 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.

2.8.14.2 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:
```

```
      ┌----------------------------------------------------┐
      |                                                    |
      |         ┌------------------┐  ┌------------------┐ |
      |         | host1            |  | host2            | |
      |         |  ┌-------------┐ |  |                  | |
      | group_a |  |   replica1  | |  |                  | |
      |         |  └-------------┘ |  |                  | |
      |         |                  |  |                  | |
      |         └------------------┘  └------------------┘ |
      |                                                    |
      ├----------------------------------------------------┤
      ├----------------------------------------------------┤
      |                                                    |
      |         ┌------------------┐  ┌------------------┐ |
      |         | host3            |  | host4            | |
      |         |                  |  |  ┌-------------┐ | |
      | group_b |                  |  |  |   replica2  | | |
      |         |                  |  |  └-------------┘ | |
      |         |                  |  |                  | |
      |         └------------------┘  └------------------┘ |
      |                                                    |
      ├----------------------------------------------------┤
      ├----------------------------------------------------┤
      |                                                    |
      |         ┌------------------┐  ┌------------------┐ |
      |         | host5            |  | host6            | |
      |         |                  |  |  ┌-------------┐ | |
      | group_c |                  |  |  |   replica3  | | |
      |         |                  |  |  └-------------┘ | |
      |         |                  |  |                  | |
      |         └------------------┘  └------------------┘ |
      |                                                    |
      └----------------------------------------------------┘
```

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';
```

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.

```
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.

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.

2.8.14.3 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. Then the variable only affects this SQL.
    select /*+ SET_VAR(exec_mem_limit=1G) */ 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.

2.8.14.4 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.

2.8.15 HTTP API

2.8.15.1 FE

2.8.15.1.1 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.

2.8.15.1.2 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"
        }],
        "removedFronteds": [],
        "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
}
```

2.8.15.1.3 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.

2.8.15.1.4 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
}
```

2.8.15.1.5 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
}
```

2.8.15.1.6 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
    }
```

2.8.15.1.7 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
}
```

2.8.15.1.8 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>&nbsp;&nbsp;&nbsp;&nbsp;Summary:</br>...",
    "count": 0
}
```

```
`data` is the text content of the profile.
```

2.8.15.1.9 Session Action

Request

```
GET /rest/v1/session
```

Description

Session Action is used to obtain the current session information.

Path parameters

None

Query parameters

None

Request body

None

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
}
```

The returned result is the same as `System Action`. Is a description of the table.

2.8.15.1.10 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.

2.8.15.1.11 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

2.8.15.1.12 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

2.8.15.1.13 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
    }
```

2.8.15.1.14 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",
            "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": [
            "配置项",
            "节点",
            "节点类型",
            "配置值类型",
            "MasterOnly",
            "配置值",
            "可修改"
        ],
        "rows": [
            [
                ""
            ]
        ]
    },
    "count": 0
}
```

POST /rest/v2/manager/node/configuration_info?type=be

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "column_names": [
            "配置项",
            "节点",
            "节点类型",
            "配置值类型",
            "配置值",
            "可修改"
        ],
        "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": [
               "配置项",
               "节点",
               "节点类型",
               "配置值类型",
               "MasterOnly",
               "配置值",
               "可修改"
           ],
           "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

```
{
    "hostPorts": ["127.0.0.1:9050"]
}
```

Response

```
{
    "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

```
    {
        "role": "FOLLOWER",
        "hostPort": "127.0.0.1:9030"
    }
```

Response

```
    {
        "msg": "success",
        "code": 0,
        "data": null,
        "count": 0
    }
```

2. drop FOLLOWER node

post /rest/v2/manager/node/DROP/fe Request body

```
{
    "role": "FOLLOWER",
    "hostPort": "127.0.0.1:9030"
}
```

Response

```
{
    "msg": "success",
    "code": 0,
    "data": null,
    "count": 0
}
```

2.8.15.1.15 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
}
```

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
}
```

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 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": {
                "": ""
            }
        }
    ],
    "count": 0
}
```

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": "36.169ms"
                }
            },
            {
                "fragment_id": "1",
                "time": "20.710ms",
                "instance_id": {
                    "d7c93d9275334c35-9e6ac5f295a7134c": "20.710ms"
                }
            },
            {
                "fragment_id": "2",
                "time": "7.83ms",
                "instance_id": {
                    "d7c93d9275334c35-9e6ac5f295a7134d": "7.83ms"
                }
            }
        ],
        "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
}
```

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", "1
                ↪ a35f62f4b14b9d7961b057b77c3102f", "select sleep(60)"],
            ["172.19.0.11", "3606cad4e34b49c6-867bf6862cacc645", "3", "", "root", "9306", "1
                ↪ a35f62f4b14b9d7961b057b77c3102f", "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
}
```

2.8.15.1.16 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
}
```

2.8.15.1.17 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,
        "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.
```

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,
        "maxReplayedJournal": 17287
    },
    "count": 0
}
```

2.8.15.1.18 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
    }
```

2.8.15.1.19 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
    }
```

2.8.15.1.20 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
    }
```

2.8.15.1.21 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
    }
```

2.8.15.1.22 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
}
```

2.8.15.1.23 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
}
```

2.8.15.1.24 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
    }
```

2.8.15.1.25 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
    }
```

2.8.15.1.26 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
    }
```

2.8.15.1.27 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 ...
```

2.8.15.1.28 Get Small File Action

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 ...
```

2.8.15.1.29 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
}
```

2.8.15.1.30 Meta Info 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.

2.8.15.1.31 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

2.8.15.1.32 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

2.8.15.1.33 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
    }
```

2.8.15.1.34 Query Detail Action

Request

`GET /api/query_detail`

Description

Used to obtain information about all queries after a specified time point

Path parameters

None

Query parameters

- `event_time`

  At the specified time point (Unix timestamp, in milliseconds), obtain query information after that time point.

Request body

无

Response

```
{
    "msg": "success",
    "code": 0,
    "data": {
        "query_details": [{
            "eventTime": 1596462699216,
            "queryId": "f732084bc8e74f39-8313581c9c3c0b58",
            "startTime": 1596462698969,
            "endTime": 1596462699216,
            "latency": 247,
            "state": "FINISHED",
            "database": "db1",
            "sql": "select * from tbl1"
        }, {
            "eventTime": 1596463013929,
            "queryId": "ed2d0d80855d47a5-8b518a0f1472f60c",
            "startTime": 1596463013913,
            "endTime": 1596463013929,
            "latency": 16,
            "state": "FINISHED",
            "database": "db1",
            "sql": "select k1 from tbl1"
        }]
    },
    "count": 0
}
```

Examples

1. Get query details after the specified time point.

```
GET /api/query_detail?event_time=1596462079958

Response:
{
    "msg": "success",
    "code": 0,
    "data": {
        "query_details": [{
            "eventTime": 1596462699216,
            "queryId": "f732084bc8e74f39-8313581c9c3c0b58",
            "startTime": 1596462698969,
            "endTime": 1596462699216,
            "latency": 247,
            "state": "FINISHED",
            "database": "db1",
            "sql": "select * from tbl1"
        }, {
            "eventTime": 1596463013929,
            "queryId": "ed2d0d80855d47a5-8b518a0f1472f60c",
            "startTime": 1596463013913,
            "endTime": 1596463013929,
            "latency": 16,
            "state": "FINISHED",
            "database": "db1",
            "sql": "select k1 from tbl1"
        }]
    },
    "count": 0
}
```

2.8.15.1.35 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
```

2.8.15.1.36   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
}
```

2.8.15.1.37 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
```

2.8.15.1.38 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
}
```

2.8.15.1.39 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
}
```

2.8.15.1.40 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
    }
```

2.8.15.1.41 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
}
```

2.8.15.1.42 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
}
```

2.8.15.1.43 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.
```

2.8.15.1.44 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//////////
        w8ABQgAAAABAAAAAAA8ABgIAAAABAAIACAAMABIIAAEAAAAADwACCwAAAAIAAAACazEAAAACazIPAAMIAAAAA
        gAAAAUAAAAFAgAEAQAAAA8ABAwAAAACDwABDAAAAAEIAAEAAAAQDAACDwABDAAAAAEIAAEAAAAADAACCAABA
        AAABQAAAAgABAAAAAAMAA8IAAEAAAAACAACAAAAAAAIABT/////CAAX/////wAADwABDAAAAAEIAAEAAAAQD
        AACDwABDAAAAAEIAAEAAAAADAACCAABAAAABQAAAAgABAAAAAAMAA8IAAEAAAABCAACAAAAAAAIABT
        /////CAAX/////
        wAADAAFCAABAAAABgwACAAADAAGCAABAAAAAA8AAgwAAAAAAAoABwAAAAAAAAAACgAIAAAAAAAAAAAADQACC
        gwAAAABAAAAAAAAJzcKAAEAAAAAAAAnNwoAAgAAAAAAAACCgADDaJlqbrVdwgIAARNJAZvAAwAAw8AAQwAA
        AACCAABAAAAAAgAAgAAAAAMAAMPAAEMAAAAAQgAAQAAAAAMAAIIAAEAAAAFAAAACAAE
        /////
        wgABQAAAAQIAAYAAAAACAAHAAAAAAsACAAAAAJrMQgACQAAAAACAAoBAAgAAQAAAAEIAAIAAAAADAADDwABDA
        AAAAEIAAEAAAAADAACCAABAAAABQAAAAgABP
        ////
        8IAAUAAAAICAAGAAAAAAgABwAAAAELAAgAAAACazIIAAkAAAABAgAKAQAPAAIMAAAAAQgAAQAAAAAIAAIAAAA
        MCAADAAAAAQoABAAAAAAAACc1CAAFAAAAAgAPAAMMAAAAAQoAAQAAAAAAACc1CAACAAAAAQgAAwAAAAIIAAQA
        AAAACwAHAAAABHRibDELAAgAAAAADAALCwABAAAABHRibDEAAAAMAAQKAAFfL5rpxl1I4goAArgs6f+h6eMxA
        AA=",
        "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
    }
```

2.8.15.1.45 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
    }
```

2.8.15.1.46 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
}
```

2.8.15.1.47 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
```

2.8.15.1.48 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

2.8.15.1.49 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":["databese list" / "table list" / "table schema"],
    "count":0
}
```

2.8.15.1.50 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
}
```

2.8.15.2 BE

2.8.15.2.1 Be Version Info Action

Request

`GET be_host:be_http_port/api/be_version_info`

Description

Get be version info from be host.

Path parameters

None.

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

```
GET be_host:be_http_port/api/be_version_info

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
}
```

2.8.15.2.2 RESTORE TABLET

description

```
To restore the tablet data from trash dir on BE

METHOD: POST
URI: http://be_host:be_http_port/api/restore_tablet?tablet_id=xxx&schema_hash=xxx
```

example

```
curl -X POST "http://hostname:8088/api/restore_tablet?tablet_id=123456\&schema_hash=1111111"
```

keyword

```
RESTORE,TABLET,RESTORE,TABLET
```

2.8.15.2.3 PAD ROWSET

description

```
Pad one empty rowset as one substitute for error replica.

METHOD: POST
URI: http://be_host:be_http_port/api/pad_rowset?tablet_id=xxx&start_version=xxx&end_version=xxx
```

example

```
curl -X POST "http://hostname:8088/api/pad_rowset?tablet_id=123456\&start_version=1111111\$end_
    ↪ version=1111112"
```

keyword

```
ROWSET,TABLET,ROWSET,TABLET
```

2.8.15.2.4 MIGRATE SINGLE TABLET TO A PARTICULAR DISK

Migrate single tablet to a particular disk.

Submit the migration task:

```
curl -X GET http://be_host:webserver_port/api/tablet_migration?goal=run&tablet_id=xxx&schema_hash
    ↪ =xxx&disk=xxx
```

The return is the submission result of the migration task:

```
    {
        status: "Success",
        msg: "migration task is successfully submitted."
    }
```

or

```
    {
        status: "Fail",
        msg: "Migration task submission failed"
    }
```

Show the status of migration task:

```
curl -X GET http://be_host:webserver_port/api/tablet_migration?goal=status&tablet_id=xxx&schema_
    ↪ hash=xxx
```

The return is the execution result of the migration task:

```
    {
        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"
    }
```

2.8.15.2.5 GET TABLETS DISTRIBUTION BETWEEN DIFFERENT DISKS

Get the distribution of tablets under each partition between different disks on BE node

```
curl -X GET http://be_host:webserver_port/api/tablets_distribution?group_by=partition
```

The return is the number distribution of tablets under each partition between different disks on BE node, which only include tablet number.

```
{
    msg: "OK",
    code: 0,
    data: {
        host: "***",
        tablets_distribution: [
            {
                partition_id:***,
                disks:[
                    {
                        disk_path:"***",
                        tablets_num:***,
                    },
                    {
                        disk_path:"***",
                        tablets_num:***,
                    },

                    ...

                ]
            },
            {
                partition_id:***,
                disks:[
                    {
                        disk_path:"***",
                        tablets_num:***,
                    },
                    {
                        disk_path:"***",
                        tablets_num:***,
                    },

                    ...

                ]
            },
            ...

        ]
    },
    count: ***
}
```

```
curl -X GET http://be_host:webserver_port/api/tablets_distribution?group_by=partition&partition_
    ↪ id=xxx
```

The return is the number distribution of tablets under the particular partition between different disks on BE node, which include tablet number, tablet id, schema hash and tablet size.

```
{
    msg: "OK",
    code: 0,
    data: {
        host: "***",
        tablets_distribution: [
            {
                partition_id:***,
                disks:[
                    {
                        disk_path:"***",
                        tablets_num:***,
                        tablets:[
                            {
                                tablet_id:***,
                                schema_hash:***,
                                tablet_size:***
                            },

                            ...

                        ]
                    },

                    ...

                ]
            }
        ]
    },
    count: ***
}
```

2.8.15.2.6 Compaction Action

This API is used to view the overall compaction status of a BE node or the compaction status of a specified tablet. It can also be used to manually trigger Compaction.

View Compaction status

The overall compaction status of the node

```
curl -X GET http://be_host:webserver_port/api/compaction/run_status
```

Return JSON:

```
{
  "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

```
curl -X GET http://be_host:webserver_port/api/compaction/show?tablet_id=xxxx
```

If the tablet does not exist, an error in JSON format is returned:

```
{
    "status": "Fail",
    "msg": "Tablet not found"
}
```

If the tablet exists, the result is returned in JSON format:

```
{
    "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": "2019-12-16 18:12:15.110",
    "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 -X GET http://192.168.10.24:8040/api/compaction/show?tablet_id=10015
```

Manually trigger Compaction

```
curl -X POST http://be_host:webserver_port/api/compaction/run?tablet_id=xxxx\&compact_type=
    ↪ cumulative
```

The only one manual compaction task that can be performed at a moment, and the value range of compact_type is base or cumulative

If the tablet does not exist, an error in JSON format is returned:

```
{
    "status": "Fail",
    "msg": "Tablet not found"
}
```

If the compaction execution task fails to be triggered, an error in JSON format is returned:

```
{
    "status": "Fail",
    "msg": "fail to execute compaction, error = -2000"
}
```

If the compaction execution task successes to be triggered, an error in 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.

Examples

```
curl -X POST http://192.168.10.24:8040/api/compaction/run?tablet_id=10015\&compact_type=
    ↪ cumulative
```

Manual Compaction execution status

```
curl -X GET http://be_host:webserver_port/api/compaction/run_status?tablet_id=xxxx
```

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,
    "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,
    "compact_type" : "cumulative"
}
```

Explanation of results:

- run_status: Get the current manual compaction task execution status.

Examples

```
curl -X GET http://192.168.10.24:8040/api/compaction/run_status?tablet_id=10015
```

2.8.15.2.7 GET TABLETS ON A PARTICULAR BE

Get the tablet id and schema hash for a certain number of tablets on a particular BE node

```
curl -X GET http://be_host:webserver_port/tablets_page?limit=XXXXX
```

The return is the tablet id and schema hash for a certain number of tablets on the BE node. The data is returned as a rendered Web page. The number of returned tablets is determined by the parameter limit. If parameter limit does not exist, none tablet will be returned. if the value of parameter limit is “all”, all the tablets on the BE node will be returned. if the value of parameter limit is non-numeric type other than “all”, none tablet will be returned.

```
curl -X GET http://be_host:webserver_port/tablets_json?limit=XXXXX
```

The return is the tablet id and schema hash for a certain number of tablets on the BE node. The returned data is organized as a Json object. The number of returned tablets is determined by the parameter limit. If parameter limit does not exist, none tablet will be returned. if the value of parameter limit is “all”, all the tablets on the BE node will be returned. if the value of parameter limit is non-numeric type other than “all”, none tablet will be returned.

```
{
    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
}
```

2.8.15.2.8 CHECK/RESET Stub Cache

description

Check Stub Cache

```
Check whether the connection cache is available

Description: Check whether the connection cache is available, the maximum load is 10M
METHOD: GET
URI: http://be_host:be_http_port/api/check_rpc_channel/{host_to_check}/{remot_brpc_port}/{payload
    ↪ _size}
```

Reset Stub Cache

```
This api is used to reset the connection cache of brpc. Endpoints can be in the form of `all` to
    ↪ clear all caches, `host1:port1,host2:port2,...`: clear to the cache of the specified
    ↪ target

Description: Reset connection cache
METHOD: GET
URI: http://be_host:be_http_port/api/reset_rpc_channel/{endpoints}
```

example

```
curl -X GET "http://host:port/api/check_rpc_channel/host2/8060/1024000"
curl -X GET "http://host:port/api/reset_rpc_channel/all"
```

2.8.15.2.9 CHECK ALL TABLET SEGMENT LOST

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.

```
curl -X POST http://be_host:webserver_port/api/check_tablet_segment_lost?repair=xxx
```

When parameter `repair` is set to `true`, tablets with lost segment will be set to SHUTDOWN status and treated as bad replica, which can be detected and repaired by FE. Otherwise, all tablets with missing segment are returned and nothing is done.

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"
}
```

3 Benchmark

3.1 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.

3.1.1 1. Hardware Environment

Hardware	Configuration Instructions
Number of Machines	4 Aliyun Virtual Machine (1FE， 3BEs)
CPU	Intel Xeon (Ice Lake) Platinum 8369B 32C

Hardware	Configuration Instructions
Memory	128G
Disk	Enterprise SSD (PL0)

3.1.2 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"

3.1.3 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

3.1.4 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

3.1.5 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
q4.2	840
q4.3	400
Total	19390

3.1.6 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).

3.1.7 7. Data Preparation

3.1.7.1 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.

3.1.7.2 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.

3.1.7.3 7.3 Create Table

3.1.7.3.1 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'
```

3.1.7.3.2 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.

3.1.7.4 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
```

3.1.7.5 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;
```

3.1.7.6 7.6 Query Test

- SSB-Flat Query Statement: ssb-flat-queries
- Standard SSB Queries: ssb-queries

3.1.7.6.1 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;
```

3.1.7.6.2 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;
```

3.2 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.

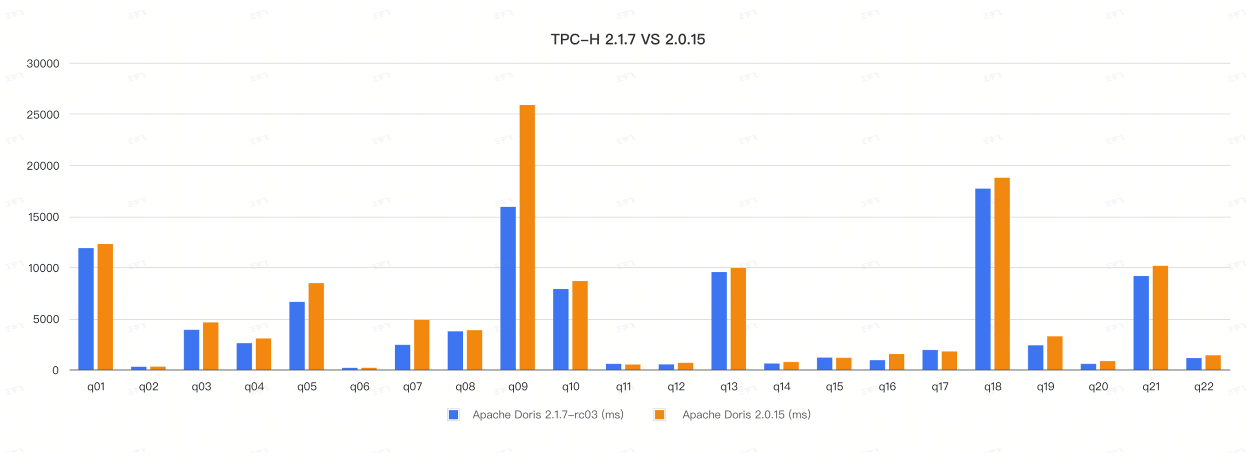


Figure 34: image-20220614114351241

3.2.1 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)

3.2.2 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”

3.2.3 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

TPC-H Table Name	Rows	Annotation
PARTSUPP	800,000,000	Parts Supply
CUSTOMER	150,000,000	Customer
ORDERS	1,500,000,000	Orders
LINEITEM	5,999,989,709	Order Details

3.2.4 4. Test SQL

TPC-H 22 test query statements : TPCH-Query-SQL

3.2.5 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
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

3.2.6 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).

3.2.7 7. Data Preparation

3.2.7.1 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.

3.2.7.2 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.

3.2.7.3 7.3 Create Table

3.2.7.3.1 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'
```

3.2.7.3.2 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 excute it in Doris.

3.2.7.4 7.4 Import Data

Please perform data import with the following command:

```
sh bin/load-tpch-data.sh
```

3.2.7.5 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;
```

3.2.7.6 7.6 Query Test

3.2.7.6.1 7.6.1 Executing Query Scripts

Execute the above test SQL or execute the following command

```
sh bin/run-tpch-queries.sh -s 1000
```

3.2.7.6.2 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;
```

3.3 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.

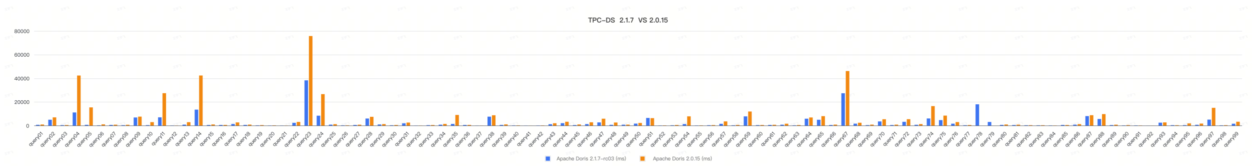


Figure 35: TPCDS_1000G

3.3.1 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)

3.3.2 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”

3.3.3 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

TPC-DS Table Name	Rows
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

3.3.4 4. Test SQL

TPC-DS 99 test query statements : TPC-DS-Query-SQL

3.3.5 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

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
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
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

Query	Apache Doris 2.1.7-rc03 (ms)	Apache Doris 2.0.15.1 (ms)
query60	380	490
query61	540	670
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
query95	350	1810
query96	660	1680
query97	5020	14990
query98	190	330
query99	1560	3230
Total	261320	507380

3.3.6 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).

3.3.7 7. Data Preparation

3.3.7.1 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
```

3.3.7.2 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.

3.3.7.3 7.3 Create Table

3.3.7.3.1 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'
```

3.3.7.3.2 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 excute it in Doris.

3.3.7.4 7.4 Import Data

Please perform data import with the following command:

```
sh bin/load-tpcds-data.sh
```

3.3.7.5 7.5 Query Test

3.3.7.5.1 7.5.1 Executing Query Scripts

Execute the above test SQL or execute the following command

```
sh bin/run-tpcds-queries.sh -s 1000
```

3.3.7.5.2 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.

4 Ecosystem

4.1 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.

4.1.1 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.

4.1.2 Installation

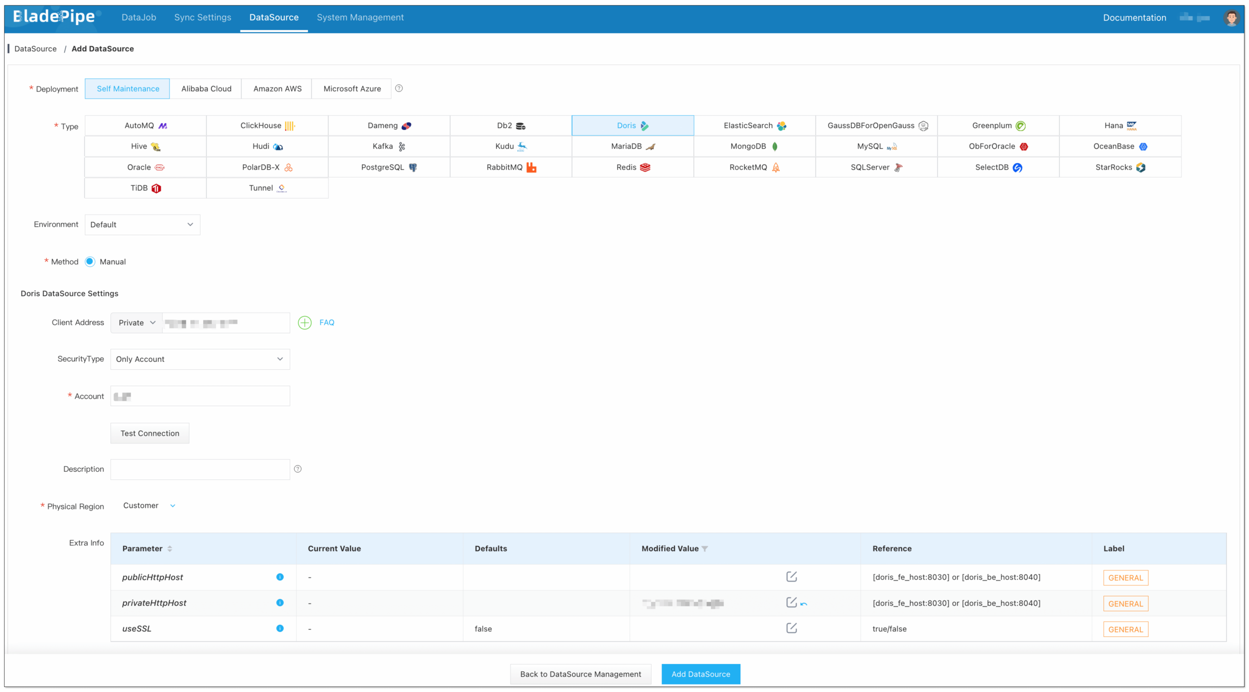
Follow the instructions in Install Worker (Docker) or Install Worker (Binary) to download and install a BladePipe Worker.

4.1.3 Example

Taking a MySQL instance as an example, the following part describes how to move data from MySQL to Doris.

4.1.3.1 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.



DataSource / **Add DataSource**

* Deployment: [✓ Self Maintenance] Alibaba Cloud Amazon AWS Microsoft Azure

* Type

AutoMQ	ClickHouse	Dameng	Db2	[✓ Doris]	ElasticSearch	GaussDBForOpenGauss	Greenplum	Hana
Hive	Hudi	Kafka	Kudu	MariaDB	MongoDB	MySQL	ObForOracle	OceanBase
Oracle	PolarDB-X	PostgreSQL	RabbitMQ	Redis	RocketMQ	SQLServer	SelectDB	StarRocks
TiDB	Tunnel							

Environment: Default

* Method: [✓ Manual]

Doris DataSource Settings

Client Address: Private [redacted] FAQ

SecurityType: Only Account

* Account: [redacted]

Test Connection

Description:

* Physical Region: Customer

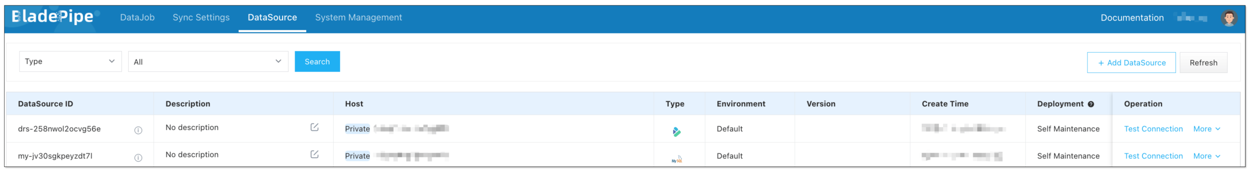
Extra Info

Parameter	Current Value	Defaults	Modified Value	Reference	Label
publicHttpHost	-			[doris_fe_host:8030] or [doris_be_host:8040]	GENERAL
privateHttpHost	-		[redacted]	[doris_fe_host:8030] or [doris_be_host:8040]	GENERAL
useSSL	-	false		true/false	GENERAL

Back to DataSource Management | Add DataSource

Figure 36: Add DataSources-1

3. Click Test Connection. After successful connection, click Add DataSource to add the DataSource.

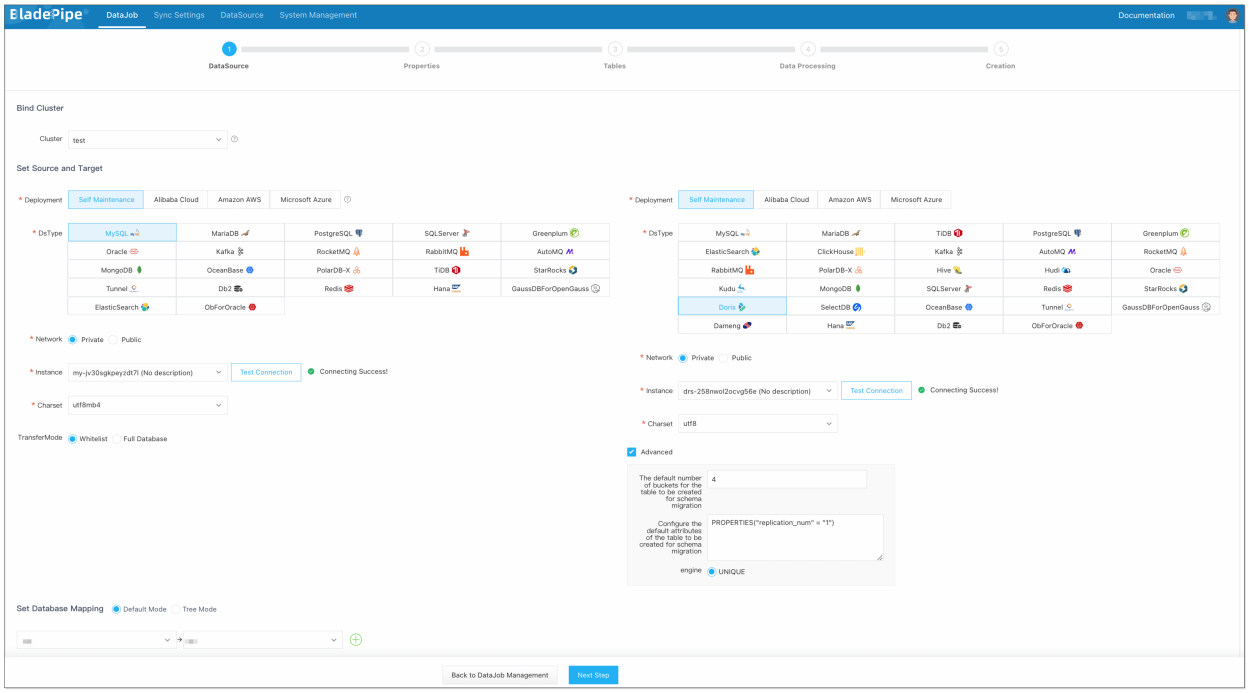


DataSource ID	Description	Host	Type	Environment	Version	Create Time	Deployment	Operation
drs-258nwol2ocvg56e	No description	Private [redacted]		Default		[redacted]	Self Maintenance	Test Connection More
my-jv30sgkpeyzdt7l	No description	Private [redacted]		Default		[redacted]	Self Maintenance	Test Connection More

Figure 37: Add DataSources-2

4.1.3.2 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.


BladePipe | DataJob | Sync Settings | DataSource | System Management | Documentation

1 DataSource — 2 Properties — 3 Tables — 4 Data Processing — 5 Creation

Bind Cluster

Cluster: test

Set Source and Target

Source:

- * Deployment: [✓ Self Maintenance] Alibaba Cloud Amazon AWS Microsoft Azure
- * DsType: [✓ MySQL] MariaDB PostgreSQL SQLServer Greenplum Oracle Kafka RocketMQ RabbitMQ AutoMQ MongoDB OceanBase PolarDB-X TiDB StarRocks Tunnel Db2 Redis Hana GaussDBForOpenGauss ElasticSearch ObForOracle
- * Network: [✓ Private] Public
- * Instance: my-jv30sgkpeyzdt7l (No description) | Test Connection | Connecting Success!
- * Charset: utf8mb4
- TransferMode: [✓ Whitelist] Full Database

Target:

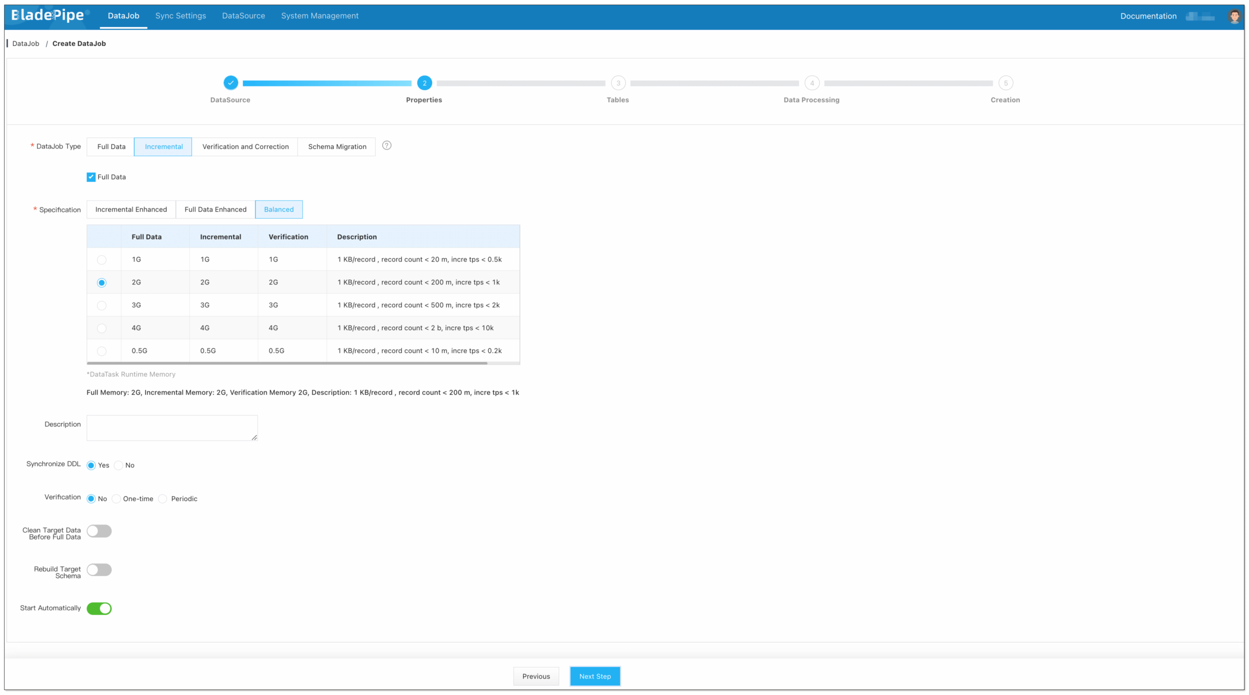
- * Deployment: [✓ Self Maintenance] Alibaba Cloud Amazon AWS Microsoft Azure
- * DsType: MySQL MariaDB TiDB PostgreSQL Greenplum ElasticSearch ClickHouse Kafka AutoMQ RocketMQ RabbitMQ PolarDB-X Hive Hudi Oracle Kudu MongoDB SQLServer Redis StarRocks [✓ Doris] SelectDB OceanBase Tunnel GaussDBForOpenGauss Dameng Hana Db2 ObForOracle
- * Network: [✓ Private] Public
- * Instance: drs-258nwol2ocvg56e (No description) | Test Connection | Connecting Success!
- * Charset: utf8
- [✓ Advanced]
  - The default number of buckets for the table to be created for schema migration: 4
  - Configure the default attributes of the table to be created for schema migration: PROPERTIES("replication_num" = "1")
  - engine: [✓ UNIQUE]

Set Database Mapping [✓ Default Mode] Tree Mode

Back to DataJob Management | Next Step

Figure 38: Create a DataJob-1

1. Select Incremental for DataJob Type, together with the Full Data option.


BladePipe | DataJob | Sync Settings | DataSource | System Management | Documentation

DataJob / **Create DataJob**

✓ DataSource — 2 Properties — 3 Tables — 4 Data Processing — 5 Creation

- * DataJob Type: Full Data [✓ Incremental] Verification and Correction Schema Migration
- [✓ Full Data]
- * Specification: Incremental Enhanced Full Data Enhanced [✓ Balanced]

	Full Data	Incremental	Verification	Description
○	1G	1G	1G	1 KB/record , record count < 20 m, incre tps < 0.5k
●	2G	2G	2G	1 KB/record , record count < 200 m, incre tps < 1k
○	3G	3G	3G	1 KB/record , record count < 500 m, incre tps < 2k
○	4G	4G	4G	1 KB/record , record count < 2 b, incre tps < 10k
○	0.5G	0.5G	0.5G	1 KB/record , record count < 10 m, incre tps < 0.2k

*DataTask Runtime Memory

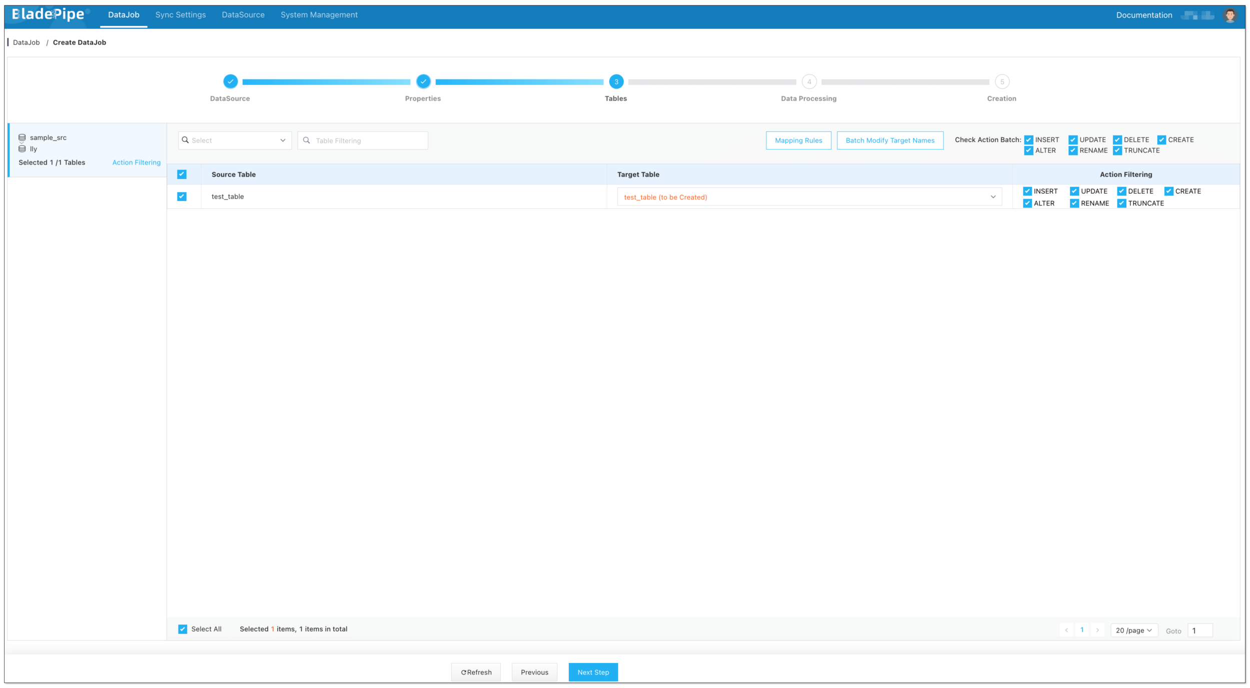
Full Memory: 2G, Incremental Memory: 2G, Verification Memory 2G, Description: 1 KB/record , record count < 200 m, incre tps < 1k

- Description:
- Synchronize DDL: [✓ Yes] No
- Verification: [✓ No] One-time Periodic
- Clean Target Data Before Full Data: off
- Rebuild Target Schema: off
- Start Automatically: on

Previous | Next Step

Figure 39: Create a DataJob-2

1. Select the tables to be replicated.



DataJob / Create DataJob

sample_src
lly
Selected 1 /1 Tables

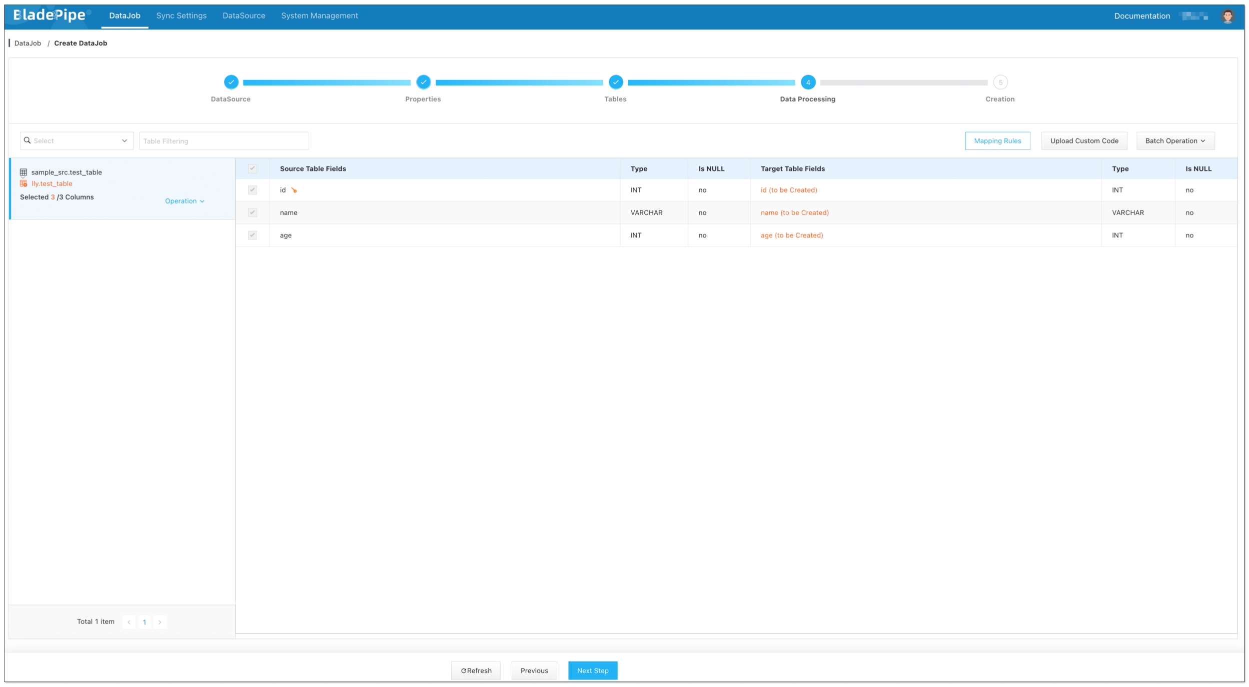
Check Action Batch: ✓ INSERT ✓ UPDATE ✓ DELETE ✓ CREATE ✓ ALTER ✓ RENAME ✓ TRUNCATE

✓	Source Table	Target Table	Action Filtering
✓	test_table	test_table (to be Created)	✓ INSERT ✓ UPDATE ✓ DELETE ✓ CREATE ✓ ALTER ✓ RENAME ✓ TRUNCATE

✓ Select All   Selected 1 items, 1 items in total

Figure 40: Create a DataJob-3

1. Select the columns to be replicated.



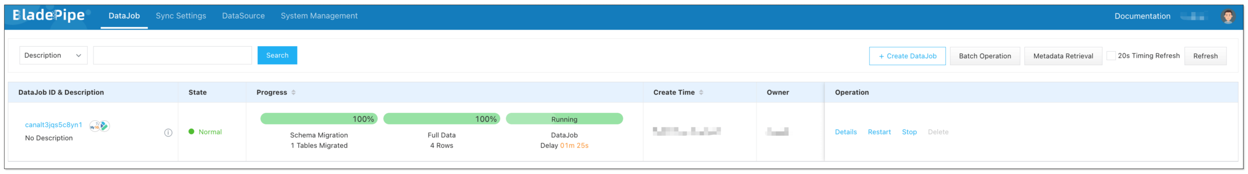
sample_src.test_table
lly.test_table
Selected 3 /3 Columns

Source Table Fields	Type	Is NULL	Target Table Fields	Type	Is NULL
id	INT	no	id (to be Created)	INT	no
name	VARCHAR	no	name (to be Created)	VARCHAR	no
age	INT	no	age (to be Created)	INT	no

Figure 41: Create a DataJob-4

1. Confirm the DataJob creation.
2. 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.



DataJob ID & Description	State	Progress	Create Time	Owner	Operation
canalt3jqs5c8yn1 No Description	Normal	100% Schema Migration 1 Tables Migrated; 100% Full Data 4 Rows; Running DataJob Delay 01m 25s	[redacted]	[redacted]	Details Restart Stop Delete

Figure 42: Create a DataJob-5

4.2 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.

4.2.1 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

4.2.2 How To Use

4.2.2.1 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.

4.2.2.2 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
```

4.2.3 Example

4.2.3.1 Read

4.2.3.1.1 SQL

```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;
```

4.2.3.1.2 DataFrame

```scala
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)
```

4.2.3.1.3 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()
```

4.2.3.1.4 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)
```

4.2.3.2 Write

4.2.3.2.1 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
```

4.2.3.2.2 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()
```

4.2.4 Configuration

4.2.4.1 General

Key	Default Value	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
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.

Key	Default Value	Comment
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
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).
doris.sink.batch.interval.ms	50	The interval time of each batch sink, unit ms.

Key	Default Value	Comment
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.

4.2.4.2 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’

4.2.4.3 Structured Streaming Configuration

Key	Default Value	Comment
doris.sink.streaming.passthrough	false	Write the value of the first column directly without processing.

4.2.4.4 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.

4.2.5 Doris & Spark Column Type Mapping

Doris Type	Spark Type
NULL_TYPE	DataTypes.NullType

Doris Type	Spark Type
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.StringType1
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.

4.2.6 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 >`

4.3 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

4.3.1 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	-

4.3.2 USE

4.3.2.1 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.

4.3.2.2 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.

4.3.3 Instructions

4.3.3.1 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();
```

4.3.3.2 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

4.3.3.3 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
```

4.3.4 Configuration

4.3.4.1 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

4.3.4.2 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

4.3.4.2.1 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.

4.3.4.3 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, \x01 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.

4.3.4.4 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

4.3.5 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
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.

4.3.6 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

4.3.7 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;
```

4.3.8 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;
```

4.3.9 Use Flink CDC to access multiple tables or the entire database (Supports MySQL, Oracle, PostgreSQL, SQLServer, MongoDB)

4.3.9.1 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_.

Key	Comment
–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.
–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

Key	Comment
–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.jar2.TheFlinkCDCversionthatConnector24.0.0dependsonmustbeabove3.1.IfFlinkCDCistobeusedforsync

4.3.9.2 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
```

4.3.9.3 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
```

4.3.9.4 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
```

4.3.9.5 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
```

4.3.9.6 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
```

4.3.9.7 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
```

4.3.10 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.

4.3.10.1 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 .

4.3.11 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.

4.3.11.1 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;
```

4.3.12 Java example

`samples/doris-demo/` An example of the Java version is provided below for reference, see here

4.3.13 Best Practices

4.3.13.1 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

4.3.13.2 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.

4.3.14 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

Occurred in the Commit phase, the transaction ID recorded in the checkpoint has expired on the FE side, and the above error will occur when committing again at this time. At this time, it cannot be started from the checkpoint, and the expiration time can be extended by modifying the streaming_label_keep_max_second configuration in fe.conf, which defaults to 12 hours.

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.

4.4 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.

4.4.1 Usage

4.4.1.1 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.

4.4.1.2 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.
```

4.4.1.3 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

4.4.1.4 Example

4.4.1.4.1 1. Stream reads the data and imports it to Doris

For instructions on using the doriswriter plug-in, please refer to here.

4.4.1.4.2 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: t, row separator: n

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_writer_c4e08cb9-c157-4689-932f-
    ↪ db34acc45b6f","LoadBytes":441,"StreamLoadPutTimeMs":1,"NumberTotalRows":2,"
    ↪ WriteDataTimeMs":11,"TxnId":217056,"LoadTimeMs":31,"TwoPhaseCommit":"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
```

4.5 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

4.5.1 version

doris	python	dbt-core
>=1.2.5	>=3.8,<=3.10	>=1.5.0

4.5.2 dbt-doris adapter Instructions

4.5.2.1 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
```

4.5.2.2 dbt-doris adapter project init

```
dbt init
```

Users need to prepare the following information to init dbt project

name	default	meaning
project		project name
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)

4.5.2.3 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

4.5.2.3.1 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") }}
```

4.5.2.3.2 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

4.5.2.3.3 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

4.5.2.4 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)
```

4.5.3 Usage Examples

4.5.3.1 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
```

4.5.3.2 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
```

4.5.3.3 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
```

4.5.3.4 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
```

4.5.3.5 Incremental model full refresh sample reference

```
{{ config(
    materialized='incremental',
    full_refresh = true
)}}

select * from
 {{ source('dbt_source', 'sell_user') }}
```

4.5.3.6 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 %}
```

4.5.3.7 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
```

4.5.3.8 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 %}
```

4.5.3.9 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 %}
```

4.5.3.10 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 %}
```

4.6 Seatunnel Doris Sink

4.6.1 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.

4.6.2 Connector-V2

The connector-v2 for SeaTunnel supports Doris Sink since version 2.3.1 and supports exactly-once write and CDC data synchronization

4.6.2.1 Plugin Code

SeaTunnel Doris Sink Plugin Code

4.6.2.2 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

4.6.2.3 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 = ","
        }
    }
}
```

4.6.3 Connector-V1

4.6.3.1 Flink Sink Doris

4.6.3.1.1 Plugin Code

Seatunnel Flink Sink Doris plugin code

4.6.3.1.2 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

4.6.3.1.3 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
    }
}
```

4.6.3.1.4 Start command

```
sh bin/start-seatunnel-flink.sh --config config/flink.streaming.conf
```

4.6.3.2 Spark Sink Doris

4.6.3.2.1 Plugin Code

Seatunnel Spark Sink Doris plugin code

4.6.3.2.2 Options

name	type	required	default value	engine
fenodes	string	yes	-	Spark
database	string	yes	-	Spark
table	string	yes	-	Spark

name	type	required	default value	engine
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

4.6.3.2.3 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"
   }
}
```

4.6.3.2.4 Start command

```
sh bin/start-waterdrop-spark.sh --master local[4] --deploy-mode client --config ./config/spark.
    ↪ conf
```

4.7 Kettle Doris Plugin

4.7.1 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.

4.7.2 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/`

4.7.3 User Manual

4.7.3.1 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

4.7.3.2 Compile Kettle Doris Plugin

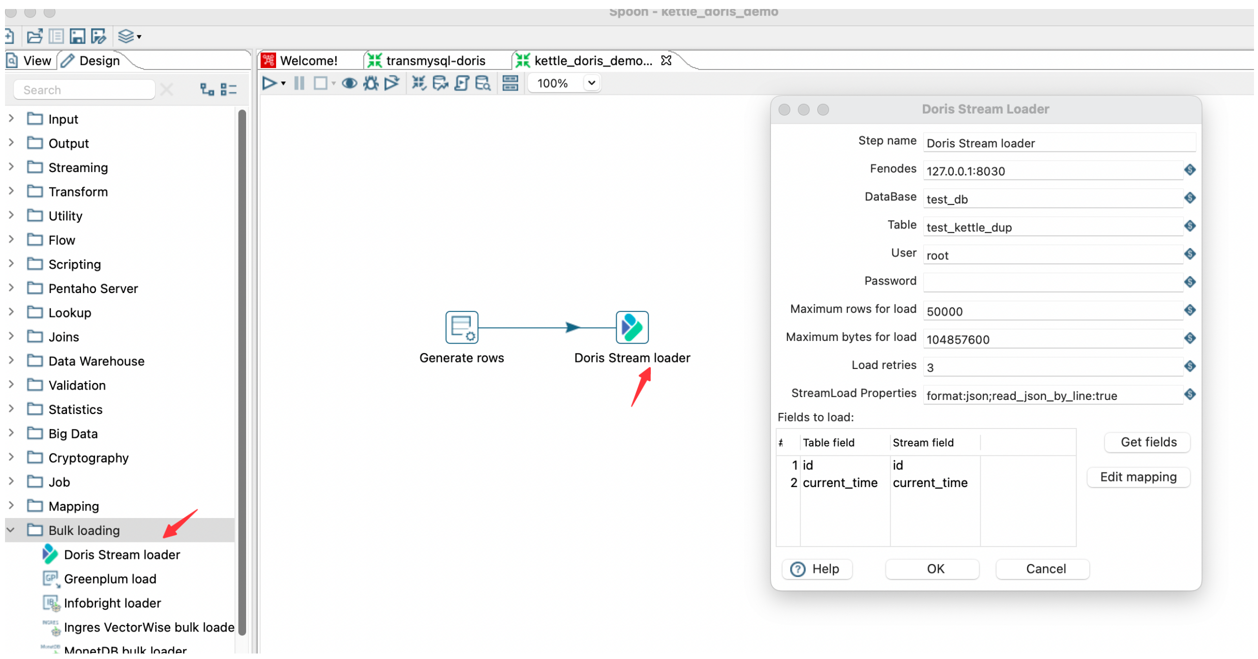
```
cd doris/extension/kettle
mvn clean package -DskipTests
```

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
```

4.7.3.3 Build a job

Find Doris Stream Loader in the batch loading in Kettle and build a job



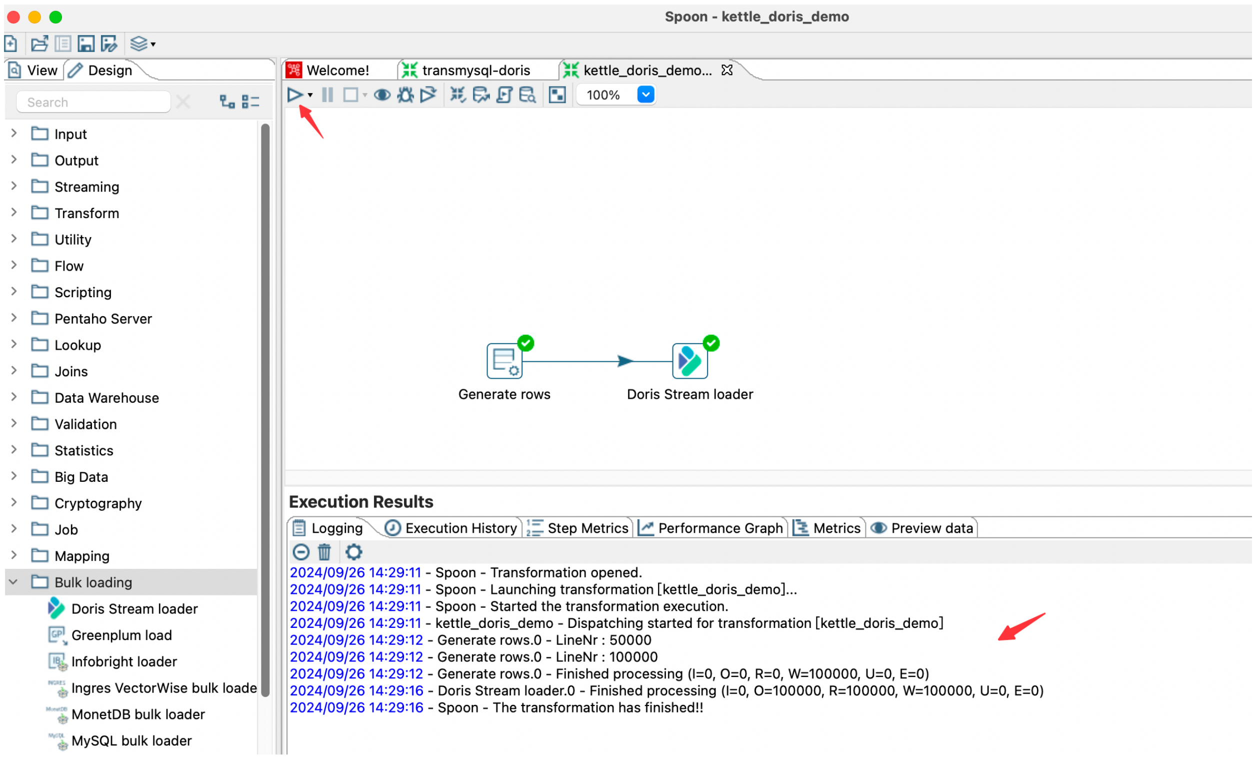
Step name	Doris Stream loader
Fenodes	127.0.0.1:8030
DataBase	test_db
Table	test_kettle_dup
User	root
Password	
Maximum rows for load	50000
Maximum bytes for load	104857600
Load retries	3
StreamLoad Properties	format:json;read_json_by_line:true

Fields to load:

#	Table field	Stream field
1	id	id
2	current_time	current_time

Figure 43: create_zh.png

Click Start Running the Job to complete data synchronization



```
2024/09/26 14:29:11 - Spoon - Transformation opened.
2024/09/26 14:29:11 - Spoon - Launching transformation [kettle_doris_demo]...
2024/09/26 14:29:11 - Spoon - Started the transformation execution.
2024/09/26 14:29:11 - kettle_doris_demo - Dispatching started for transformation [kettle_doris_demo]
2024/09/26 14:29:12 - Generate rows.0 - LineNr : 50000
2024/09/26 14:29:12 - Generate rows.0 - LineNr : 100000
2024/09/26 14:29:12 - Generate rows.0 - Finished processing (I=0, O=0, R=0, W=100000, U=0, E=0)
2024/09/26 14:29:16 - Doris Stream loader.0 - Finished processing (I=0, O=100000, R=100000, W=100000, U=0, E=0)
2024/09/26 14:29:16 - Spoon - The transformation has finished!!
```

Figure 44: running_zh.png

4.7.3.4 Parameter Description

Key	Default Value	Required	Comment
Step name	–	Y	Step name
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

Key	Default Value	Required	Comment
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.

{ "title": "Kyuubi", "language": "en" }

4.8 Kyuubi

4.8.1 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.

4.8.2 Usage

4.8.2.1 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.

4.8.2.2 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`

Configuration	Description
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 .

4.8.2.3 Add MySQL JDBC Driver

Copy the Mysql JDBC Driver `mysql-connector-j-8.X.X.jar` to `$KYUUBI_HOME/externals/engines/jdbc`.

4.8.2.4 Start Kyuubi Server

Run `$KYUUBI_HOME/bin/kyuubi start`. After started, port 10009 by default is listened by Kyuubi Server with Thrift protocol.

4.8.3 Example

The following example shows basic example of querying Doris with Kyuubi with beeline CLI in Thrift protocol.

4.8.3.1 Connect to Kyuubi with Beeline

```
$ $KYUUBI_HOME/bin/beeline -u "jdbc:hive2://xxxx:10009/"
```

4.8.3.2 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-4
    ↪ c02-8c3a-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)
```

4.9 Logstash Doris Output Plugin

4.9.1 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

4.9.2 Installation

4.9.2.1 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 without dependencies https://apache-doris-releases.oss-accelerate.aliyuncs.com/logstash-output-doris-1.0.0.gem
- Installation package with dependencies https://apache-doris-releases.oss-accelerate.aliyuncs.com/logstash-output-doris-1.0.0.zip
- Compile from source code

```
cd extension/logstash/

gem build logstash-output-doris.gemspec
```

4.9.2.2 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-1.0.0.gem

Validating logstash-output-doris-1.0.0.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

```
${LOGSTASH_HOME}/bin/logstash-plugin install file:///tmp/logstash-output-doris-1.0.0.zip

Installing file: logstash-output-doris-1.0.0.zip
Resolving dependencies.........................
Install successful
```

4.9.3 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

Configuration	Description
`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 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

4.9.4 Usage Example

4.9.4.1 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
```

4.9.4.2 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
```

4.10 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.

4.10.1 Installation

4.10.1.1 Download from the Official Website

https://apache-doris-releases.oss-accelerate.aliyuncs.com/filebeat-doris-2.0.0

4.10.1.2 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
```

4.10.2 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
`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

4.10.3 Usage Example

4.10.3.1 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
```

4.10.3.2 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
```

4.11 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.

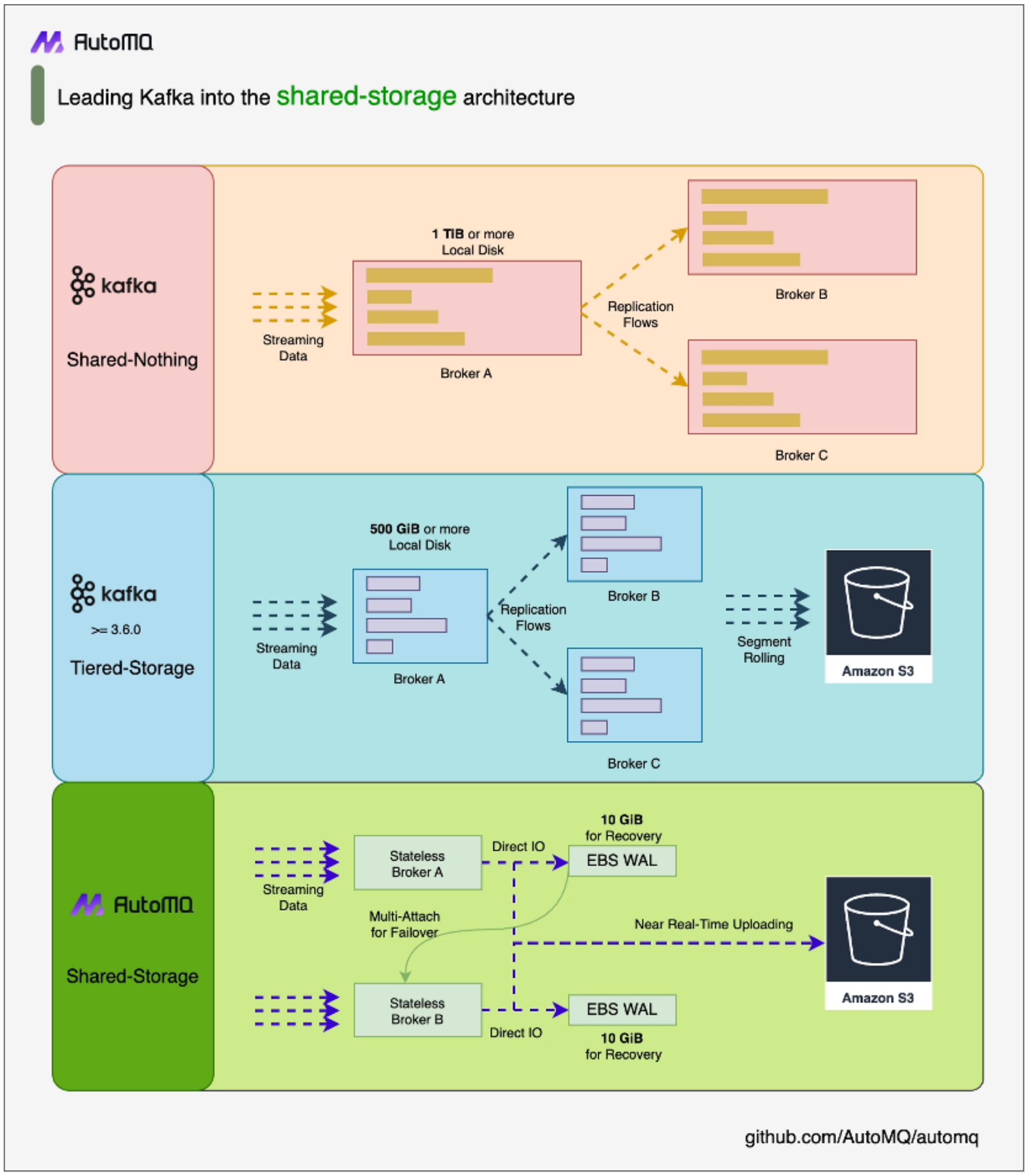


Figure 45: AutoMQ Storage Architecture

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.

4.11.1 Environment Preparation

4.11.1.1 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');
```

4.11.1.2 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.

4.11.1.3 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.

4.11.2 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.

4.11.3 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)
```

4.12 Doris Streamloader

4.12.1 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.

4.12.2 Installation

Source Code: https://github.com/apache/doris-streamloader/ Binary File: https://doris.incubator.apache.org/download

The obtained result is the executable binary.

4.12.3 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
–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: (LoadBytes1000)/(LoadTimeMs1024*1024)

Parameter	Description	Default Value	Suggestions
–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 |

Whether to check the encoding of the data that has been loaded:

1) false, direct load of raw data without checking; 2) true, replacing non UTF-8 characters with �
   | true |Remain as default| |–debug |Print debug log | false | Remain as default | |–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". |

4.12.4 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"
```

```
    ]
}
```

4.12.5 Best practice

4.12.5.1 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.

4.12.5.2 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
```

4.12.5.3 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`.

4.13 Hive Bitmap 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;

4.13.1 How To Use

4.13.1.1 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'
```

4.13.1.2 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
```

4.13.1.3 Hive Bitmap UDF Description

4.13.2 Hive Bitmap import into Doris

4.13.2.1 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;
```

4.13.2.2 Method 2: Spark Load

see details: Spark Load -> Basic operation -> Create load(Example 3: when the upstream data source is hive binary type table)

4.14 Hive HLL UDF

The Hive HLL UDF provides a set of UDFs for generating HLL operations in Hive tables, which are identical to Doris HLL. Hive HLL can be imported into Doris through Spark HLL Load. For more information about HLL, please refer to Using HLL for Approximate Deduplication.:Approximate Deduplication Using HLL

Function Introduction: 1. UDAF

```
· to_hll: An aggregate function that returns a Doris HLL column, similar to the to_bitmap
    ↪ function

· hll_union: An aggregate function that calculates the union of groups, returning a Doris HLL
    ↪ column, similar to the bitmap_union function
```

2. UDF

   ·hll_cardinality: Returns the number of distinct elements added to the HLL, similar to the bitmap_count function

Main Purpose: 1. Reduce data import time to Doris by eliminating the need for dictionary construction and HLL pre-aggregation 2. Save Hive storage by compressing data using HLL, significantly reducing storage costs compared to Bitmap statistics 3. Provide flexible HLL operations in Hive, including union and cardinality statistics, and allow the resulting HLL to be directly imported into Doris

Note: HLL statistics are approximate calculations with an error rate of around 1% to 2%.

4.14.1 Usage

4.14.1.1 Create a Hive table and insert test data

```
-- Create a test database, e.g., hive_test
use hive_test;

-- Create a Hive HLL table
CREATE TABLE IF NOT EXISTS `hive_hll_table`(
  `k1`   int       COMMENT '',
  `k2`   String    COMMENT '',
  `k3`   String    COMMENT '',
  `uuid` binary    COMMENT 'hll'
) comment  'comment'

-- Create a normal Hive table and insert test data
CREATE TABLE IF NOT EXISTS `hive_table`(
    `k1`   int       COMMENT '',
    `k2`   String    COMMENT '',
    `k3`   String    COMMENT '',
    `uuid` int       COMMENT ''
) comment  'comment'
```

```
insert into hive_table select 1, 'a', 'b', 12345;
insert into hive_table select 1, 'a', 'c', 12345;
insert into hive_table select 2, 'b', 'c', 23456;
insert into hive_table select 3, 'c', 'd', 34567;
```

4.14.1.2 Use Hive HLL UDF:

Hive HLL UDF needs to be used in Hive/Spark. First, compile the FE to obtain the hive-udf.jar file. Compilation preparation: If you have compiled the ldb source code, you can directly compile the FE. If not, you need to manually install thrift, refer to Setting Up Dec Env for FE - IntelliJ IDEA for compilation and installation.

```
-- Clone the Doris source code
git clone https://github.com/apache/doris.git
cd doris
git submodule update --init --recursive

-- Install thrift (skip if already installed)
-- Enter the FE directory
cd fe

-- Execute the Maven packaging command (all FE submodules will be packaged)
mvn package -Dmaven.test.skip=true
-- Or package only the hive-udf module
mvn package -pl hive-udf -am -Dmaven.test.skip=true

-- The packaged hive-udf.jar file will be generated in the target directory
-- Upload the compiled hive-udf.jar file to HDFS, e.g., to the root directory
hdfs dfs -put hive-udf/target/hive-udf.jar /
```

Then, enter Hive and execute the following SQL statements:

```
-- Load the hive hll udf jar package, modify the hostname and port according to your actual
    ↪ situation
add jar hdfs://hostname:port/hive-udf.jar;

-- Create UDAF functions
create temporary function to_hll as 'org.apache.doris.udf.ToHllUDAF' USING JAR 'hdfs://hostname:
    ↪ port/hive-udf.jar';
create temporary function hll_union as 'org.apache.doris.udf.HllUnionUDAF' USING JAR 'hdfs://
    ↪ hostname:port/hive-udf.jar';


-- Create UDF functions
create temporary function hll_cardinality as 'org.apache.doris.udf.HllCardinalityUDF' USING JAR '
    ↪ hdfs://node:9000/hive-udf.jar';
```

```
-- Example: Use the to_hll UDAF to aggregate and generate HLL, and write it to the Hive HLL table
insert into hive_hll_table
select
    k1,
    k2,
    k3,
    to_hll(uuid) as uuid
from
    hive_table
group by
    k1,
    k2,
    k3

-- Example: Use hll_cardinality to calculate the number of elements in the HLL
select k1, k2, k3, hll_cardinality(uuid) from hive_hll_table;
+-----+-----+-----+------+
| k1  | k2  | k3  | _c3  |
+-----+-----+-----+------+
| 1   | a   | b   | 1    |
| 1   | a   | c   | 1    |
| 2   | b   | c   | 1    |
| 3   | c   | d   | 1    |
+-----+-----+-----+------+

-- Example: Use hll_union to calculate the union of groups, returning 3 rows
select k1, hll_union(uuid) from hive_hll_table group by k1;

-- Example: Also can merge and then continue to statistics
select k3, hll_cardinality(hll_union(uuid)) from hive_hll_table group by k3;
+-----+------+
| k3  | _c1  |
+-----+------+
| b   | 1    |
| c   | 2    |
| d   | 1    |
+-----+------+
```

4.14.1.3 Hive HLL UDF Explanation

4.14.2 Importing Hive HLL to Doris

4.14.2.1 Method 1: Catalog (Recommended)

Create Hive table specified as TEXT format. For Binary type, Hive will save it as a base64 encoded string. At this time, you can use the Hive Catalog to directly import the HLL data into Doris using the hll_from_base64 function.

Here is a complete example:

1. Create a Hive table

```
CREATE TABLE IF NOT EXISTS `hive_hll_table`(
`k1`   int       COMMENT '',
`k2`   String    COMMENT '',
`k3`   String    COMMENT '',
`uuid` binary    COMMENT 'hll'
) stored as textfile

-- then reuse the previous steps to insert data from a normal table into it using the to_hll
    ↪ function
```

2. Create a Doris catalog

```
CREATE CATALOG hive PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://127.0.0.1:9083'
);
```

3. Create a Doris internal table

```
CREATE TABLE IF NOT EXISTS `doris_test`.`doris_hll_table`(
    `k1`   int                   COMMENT '',
    `k2`   varchar(10)           COMMENT '',
    `k3`   varchar(10)           COMMENT '',
    `uuid` HLL  HLL_UNION  COMMENT 'hll'
)
AGGREGATE KEY(k1, k2, k3)
DISTRIBUTED BY HASH(`k1`) BUCKETS 1
PROPERTIES (
    "replication_allocation" = "tag.location.default: 1"
);
```

4. Import data from Hive to Doris

```
insert into doris_hll_table select k1, k2, k3, hll_from_base64(uuid) from hive.hive_test.hive_hll
    ↪ _table;

-- View the imported data, combining hll_to_base64 for decoding
select *, hll_to_base64(uuid) from doris_hll_table;
+------+------+------+------+---------------------+
| k1   | k2   | k3   | uuid | hll_to_base64(uuid) |
+------+------+------+------+---------------------+
|    1 | a    | b    | NULL | AQFw+a9MhpKhoQ==    |
|    1 | a    | c    | NULL | AQFw+a9MhpKhoQ==    |
|    2 | b    | c    | NULL | AQGyB7kbWBxh+A==    |
|    3 | c    | d    | NULL | AQFYbJB5VpNBhg==    |
+------+------+------+------+---------------------+

-- Also can use Doris's native HLL functions for statistics, and see that the results are
    ↪ consistent with the previous statistics in Hive
select k3, hll_cardinality(hll_union(uuid)) from doris_hll_table group by k3;
+------+----------------------------------+
| k3   | hll_cardinality(hll_union(uuid)) |
+------+----------------------------------+
| b    |                                1 |
| d    |                                1 |
| c    |                                2 |
+------+----------------------------------+

-- At this time, querying the external table data, i.e., the data before import, can also verify
    ↪ the correctness of the data
select k3, hll_cardinality(hll_union(hll_from_base64(uuid))) from hive.hive_test.hive_hll_table
    ↪ group by k3;
+------+---------------------------------------------------+
| k3   | hll_cardinality(hll_union(hll_from_base64(uuid))) |
+------+---------------------------------------------------+
| d    |                                                 1 |
| b    |                                                 1 |
| c    |                                                 2 |
+------+---------------------------------------------------+
```

4.14.2.2 Method 2: Spark Load

See details: Spark Load -> Basic operation -> Creating Load (Example 3: when the upstream data source is hive binary type table)

4.15 BI

4.15.1 Apache Superset

4.15.1.1 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.

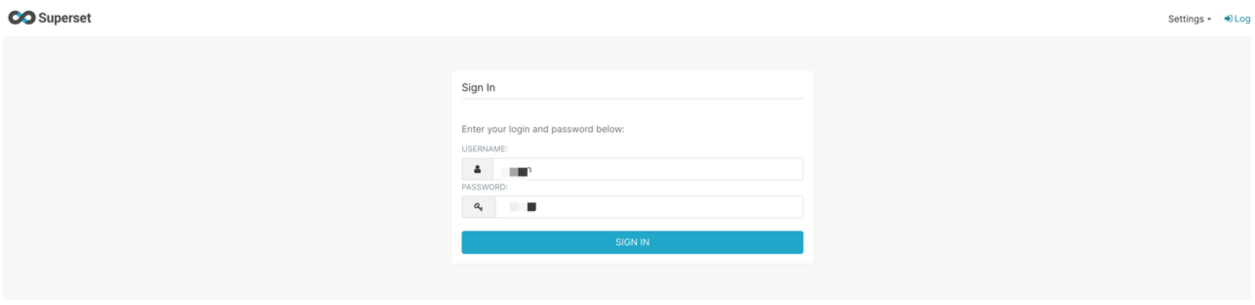


Figure 46: login page

2. Select the “Add Database Connection” option after logging into Superset.

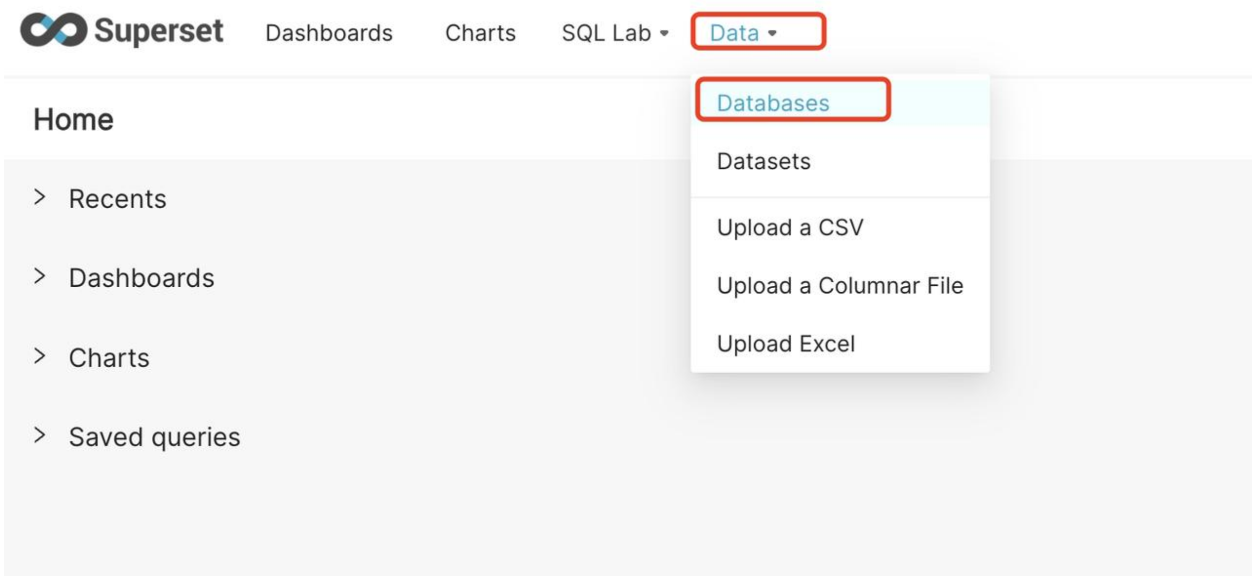
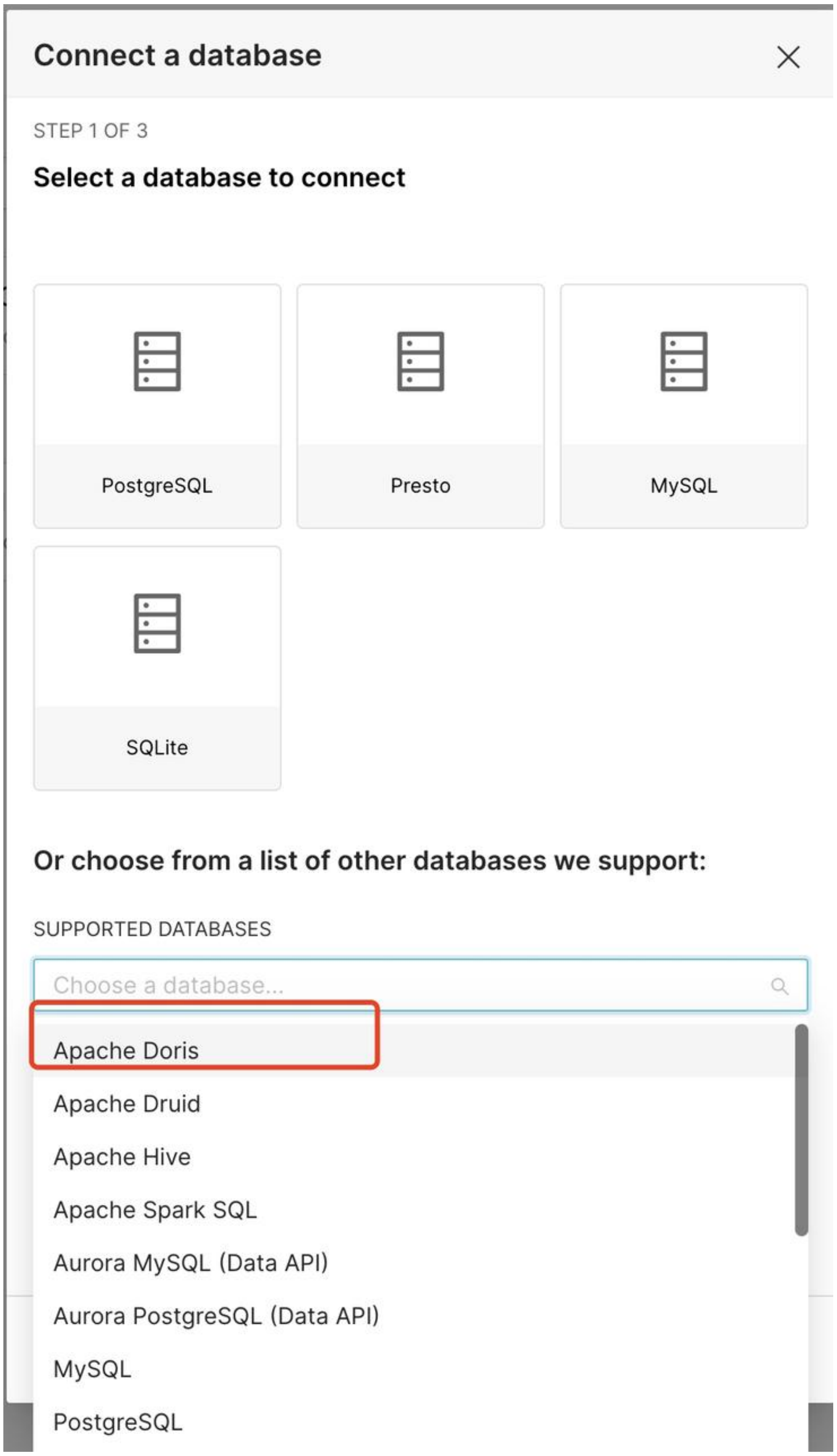


Figure 47: add databases

3. In the connection page, select the “Apache Doris” option.


Connect a database

STEP 1 OF 3

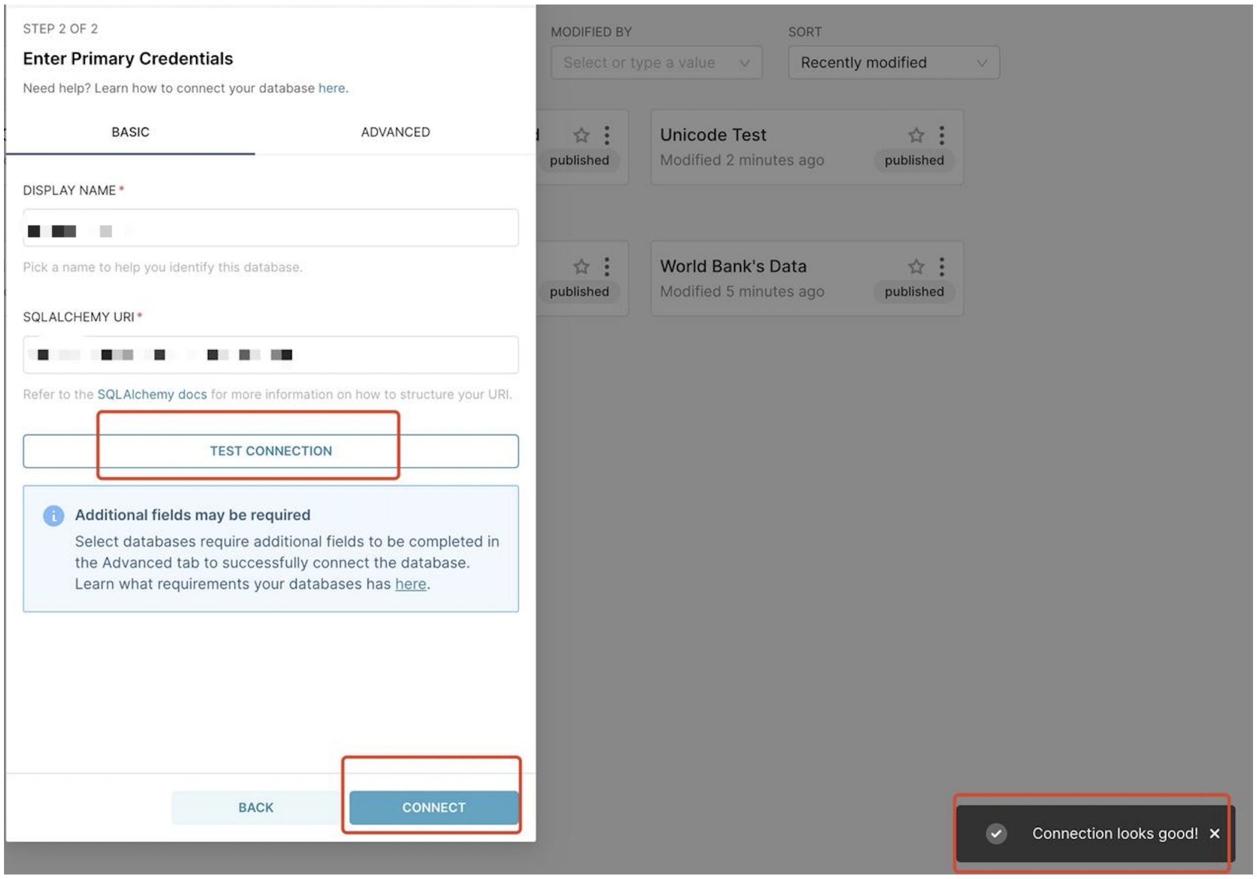
Select a database to connect

Or choose from a list of other databases we support:

SUPPORTED DATABASES

Figure 48: select databases

4. Fill in the SQLAlchemy URI information in the connection details and proceed with the relevant connectivity verification.


STEP 2 OF 2

Enter Primary Credentials

Need help? Learn how to connect your database here.

DISPLAY NAME * [redacted]

Pick a name to help you identify this database.

SQLALCHEMY URI * [redacted]

Refer to the SQLAlchemy docs for more information on how to structure your URI.

Additional fields may be required

Select databases require additional fields to be completed in the Advanced tab to successfully connect the database. Learn what requirements your databases has here.

Unicode Test — Modified 2 minutes ago — published

World Bank's Data — Modified 5 minutes ago — published

Connection looks good!

Figure 49: 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.

4.15.2 DBeaver

4.15.2.1 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.

4.15.2.2 Preconditions

Dbeaver installed You can visit https://dbeaver.io to download and install DBeaver

4.15.2.3 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.

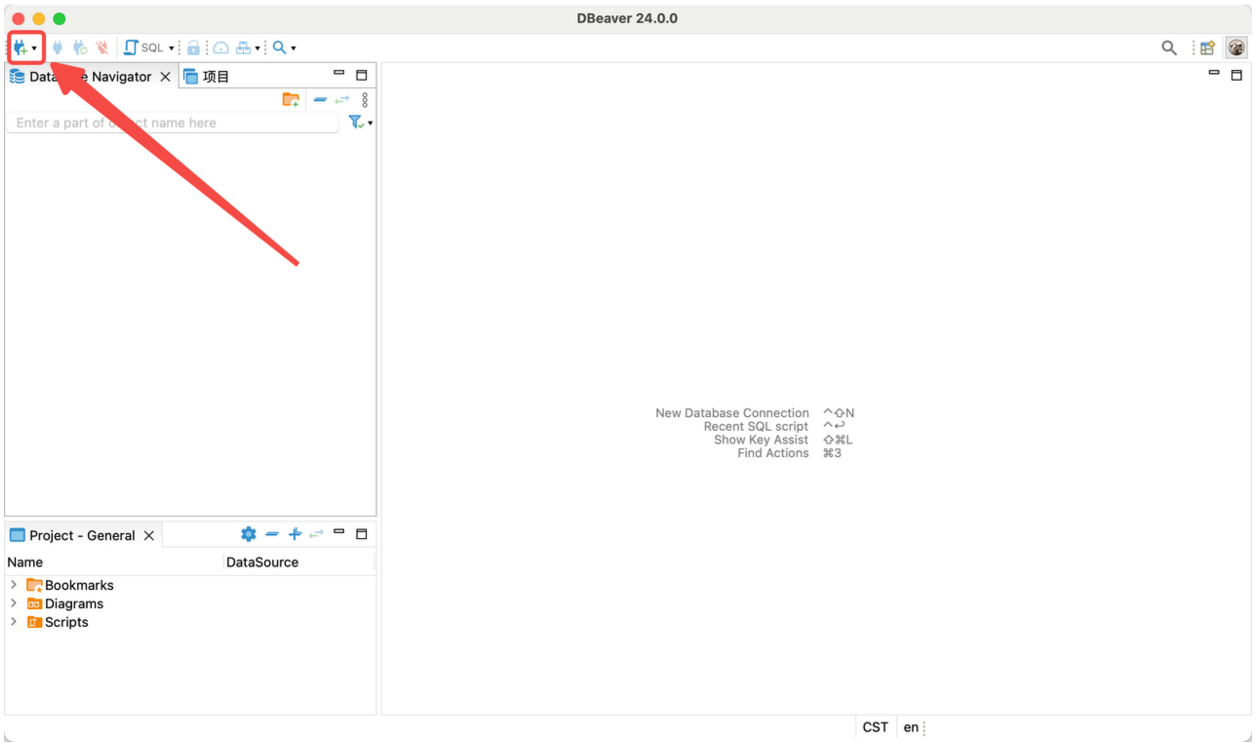


Figure 50: add connection 1

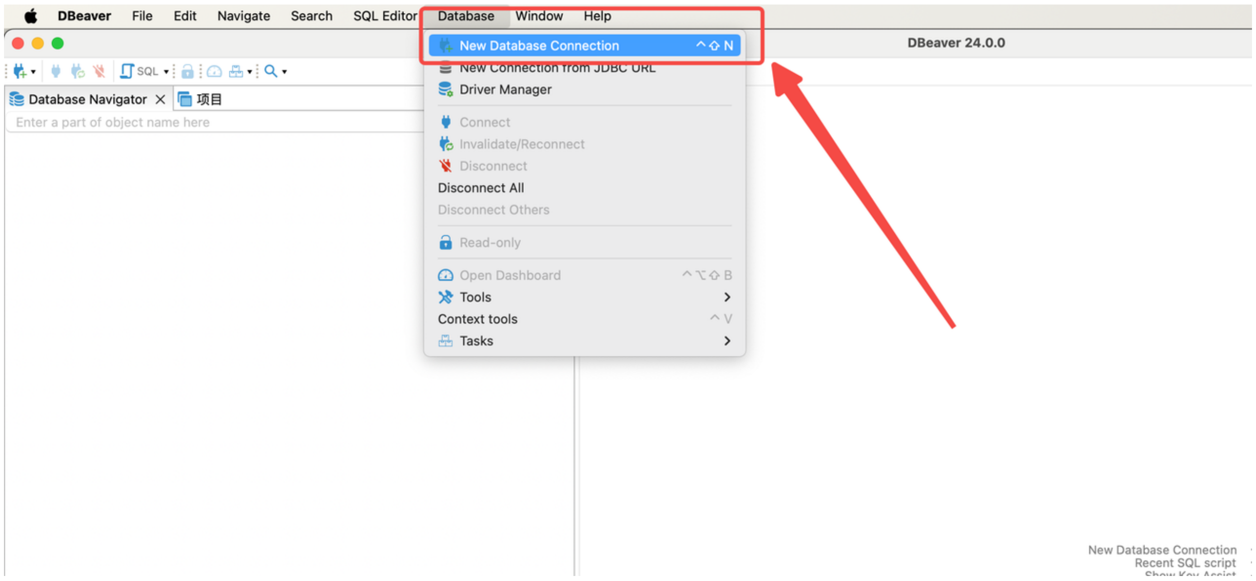


Figure 51: add connection 2

3. Select the MySQL driver

   In the Select your database window, select MySQL.

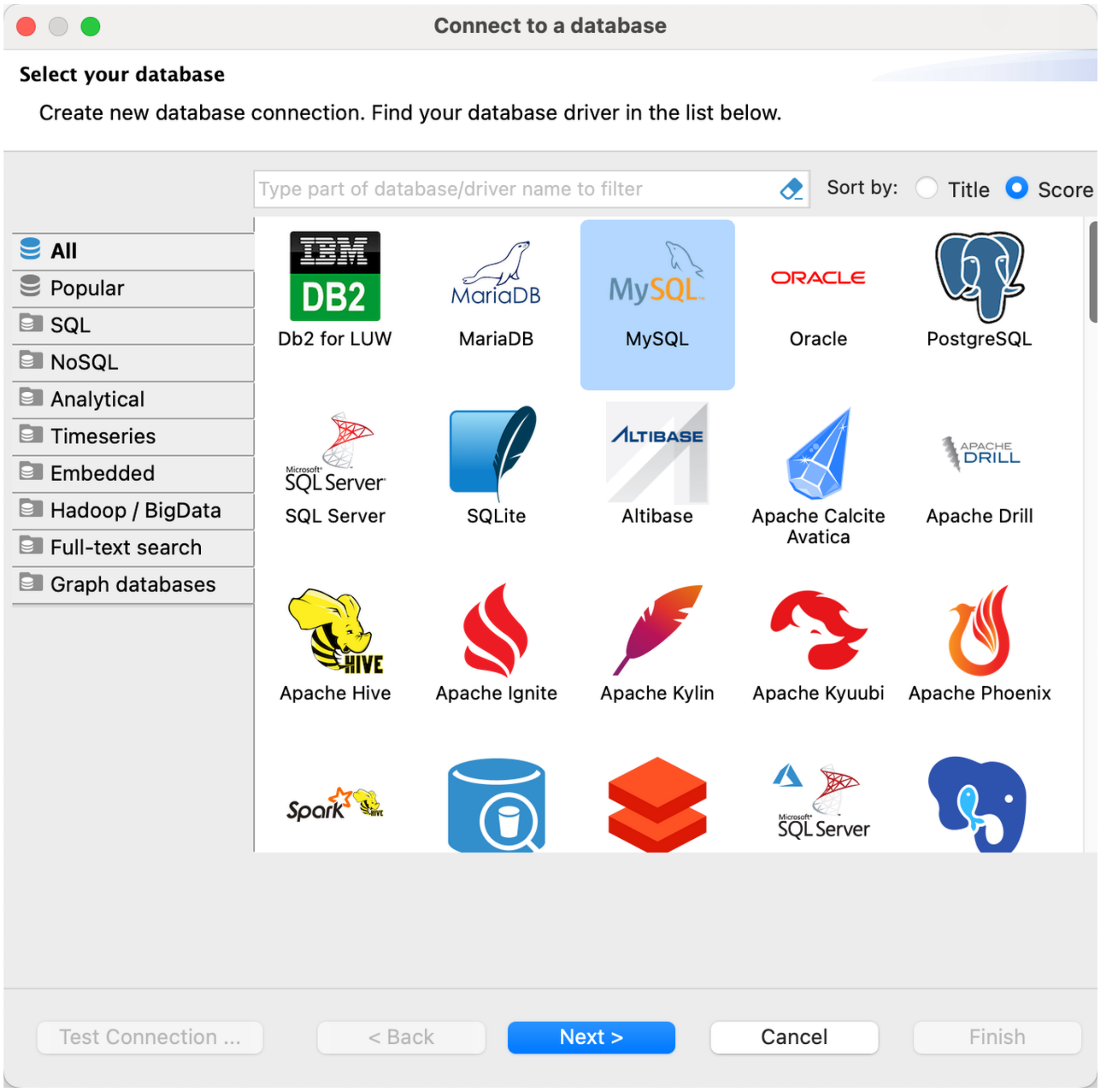


Figure 52: chose driver

4. Configure Doris connection

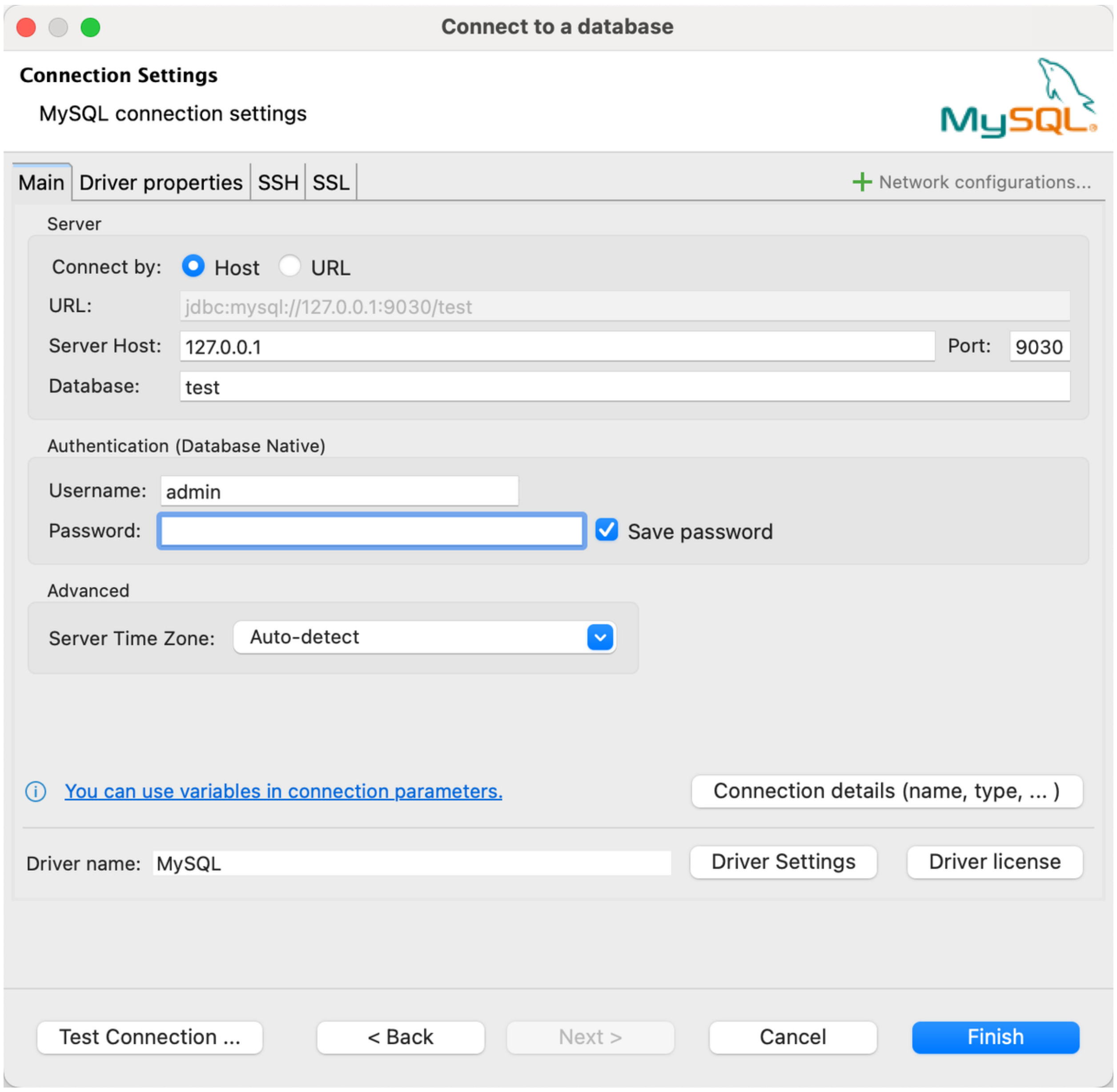
   In the main tab of the Connection Settings window, configure the following connection information:

- 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.

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.

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 to a database

Connection Settings

MySQL connection settings

Server

Connect by: [✓ Host] URL

URL: jdbc:mysql://127.0.0.1:9030/test

Server Host: 127.0.0.1 Port: 9030

Database: test

Authentication (Database Native)

Username: admin

Password:

[✓] Save password

Advanced

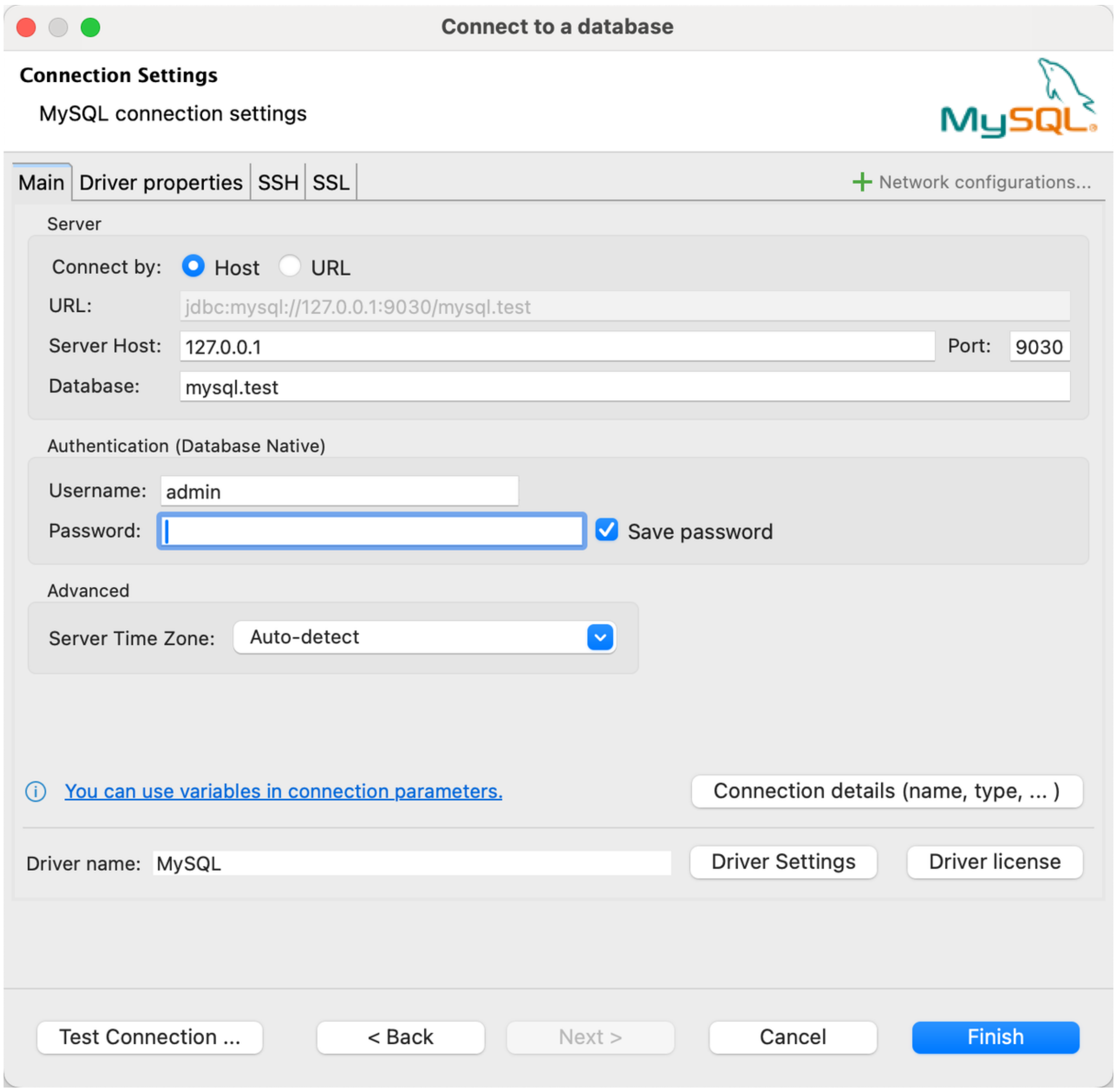
Server Time Zone: Auto-detect

You can use variables in connection parameters.

Driver name: MySQL

Figure 53: connect internal catalog

- external catalog


Connect to a database

Connection Settings

MySQL connection settings

Main | Driver properties | SSH | SSL

Server

Connect by: [✓ Host] URL

URL: jdbc:mysql://127.0.0.1:9030/mysql.test

Server Host: 127.0.0.1   Port: 9030

Database: mysql.test

Authentication (Database Native)

Username: admin

Password:

[✓] Save password

Advanced

Server Time Zone: Auto-detect

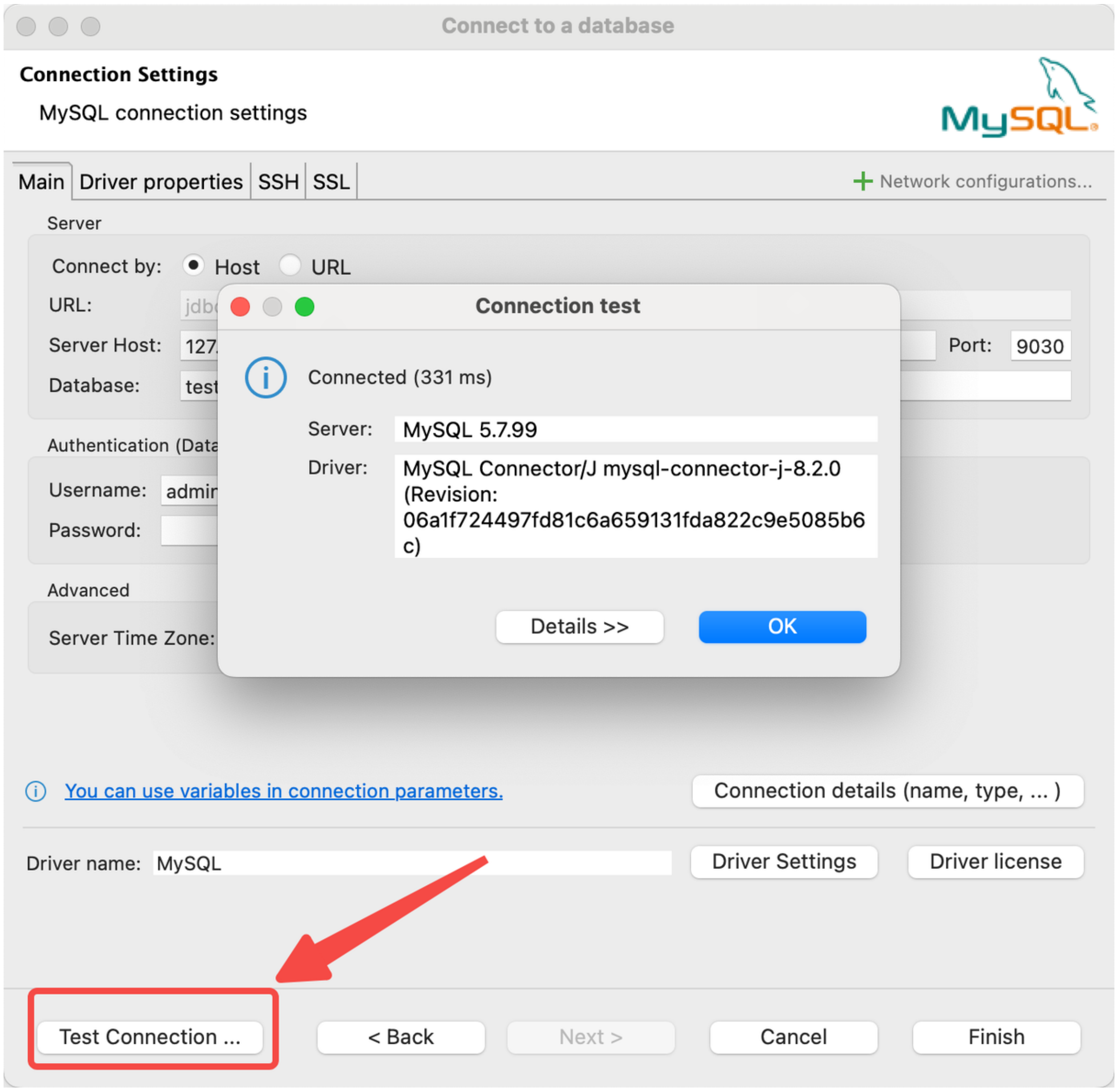
You can use variables in connection parameters.

Driver name: MySQL

Figure 54: 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. 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.


Connect to a database

Connection Settings

MySQL connection settings

Main | Driver properties | SSH | SSL

Server

Connect by: [✓ Host] URL

URL: jdb

Server Host: 127.

Port: 9030

Database: test

Authentication (Data

Username: admin

Password:

Advanced

Server Time Zone:

Connection test

Connected (331 ms)

Server: MySQL 5.7.99

Driver: MySQL Connector/J mysql-connector-j-8.2.0 (Revision: 06a1f724497fd81c6a659131fda822c9e5085b6c)

Details >> | OK

You can use variables in connection parameters.

Connection details (name, type, ...)

Driver name: MySQL

Driver Settings | Driver license

Test Connection ... | < Back | Next > | Cancel | Finish

Figure 55: 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 DBeaver.

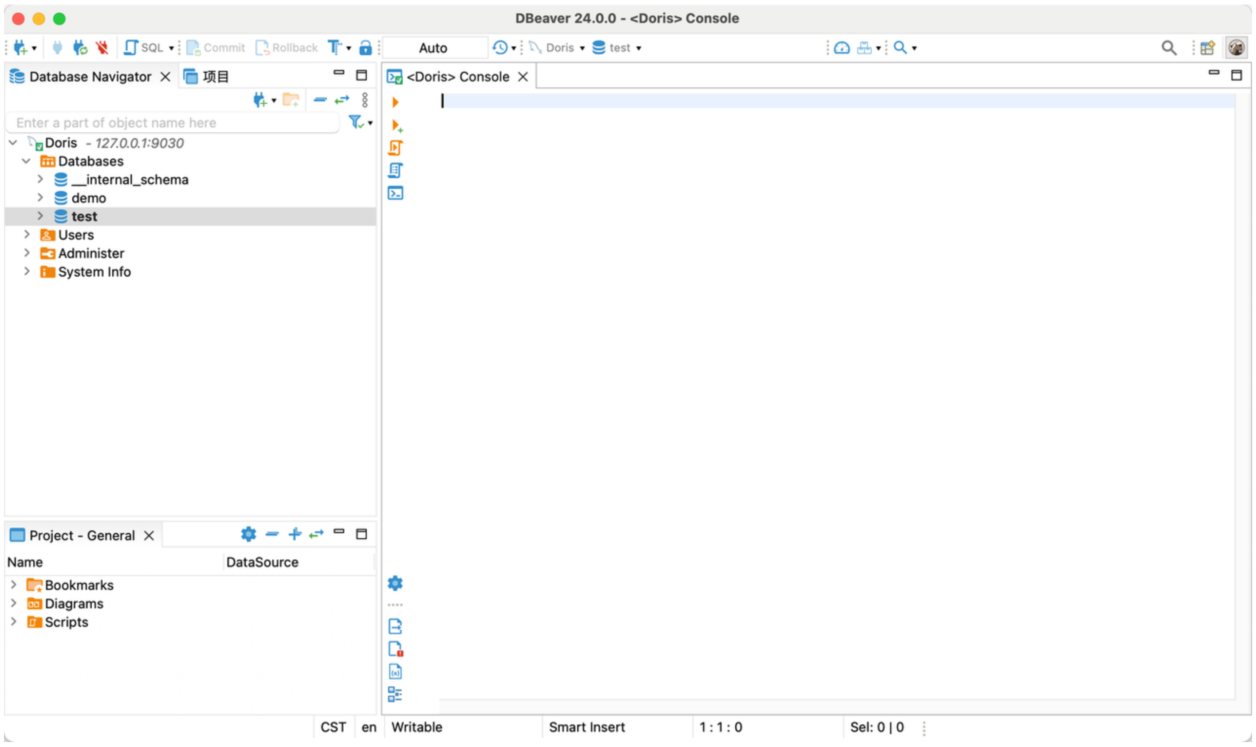


Figure 56: create connection

4.15.2.4 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.

4.15.3 DataGrip

4.15.3.1 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.

4.15.3.2 Preconditions

DataGrip installed You can visit www.jetbrains.com/datagrip/ to download and install DataGrip

4.15.3.3 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

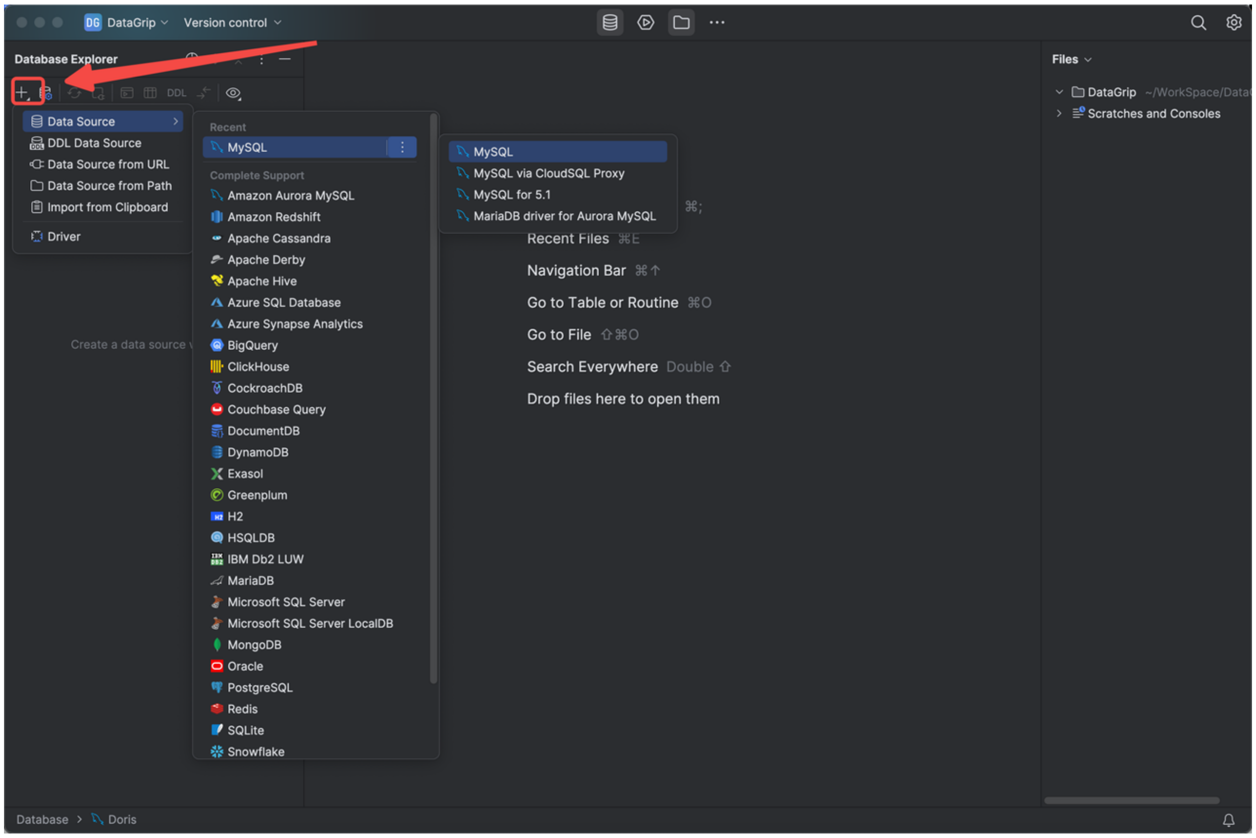


Figure 57: add data source

3. Configure Doris connection

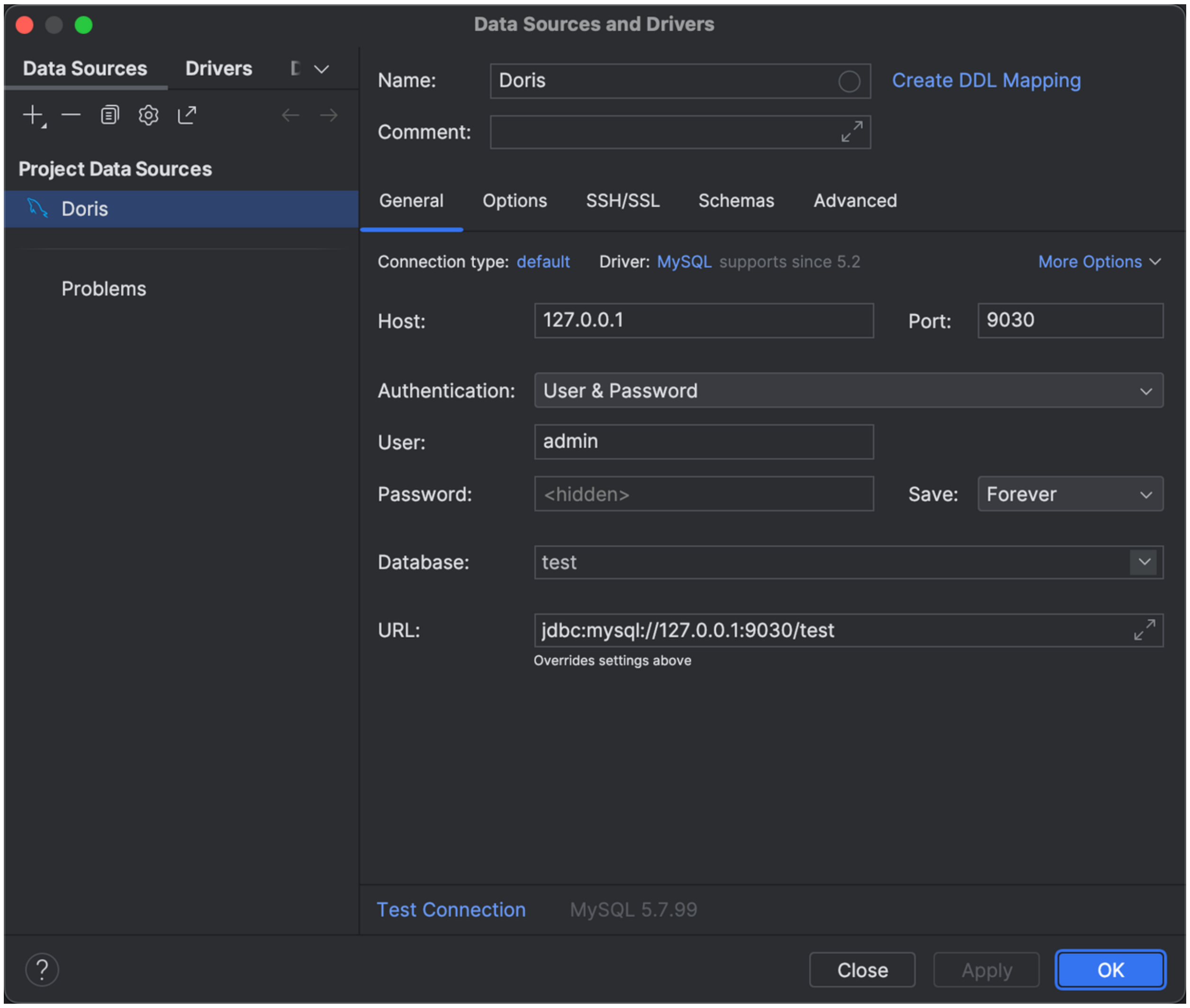
   On the General tab of the Data Sources and Drivers window, configure the following connection information:

- 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.
```

Note Managing the external catalog connected to Doris through the Database form of catalog.db requires Doris version 2.1.0 and above.

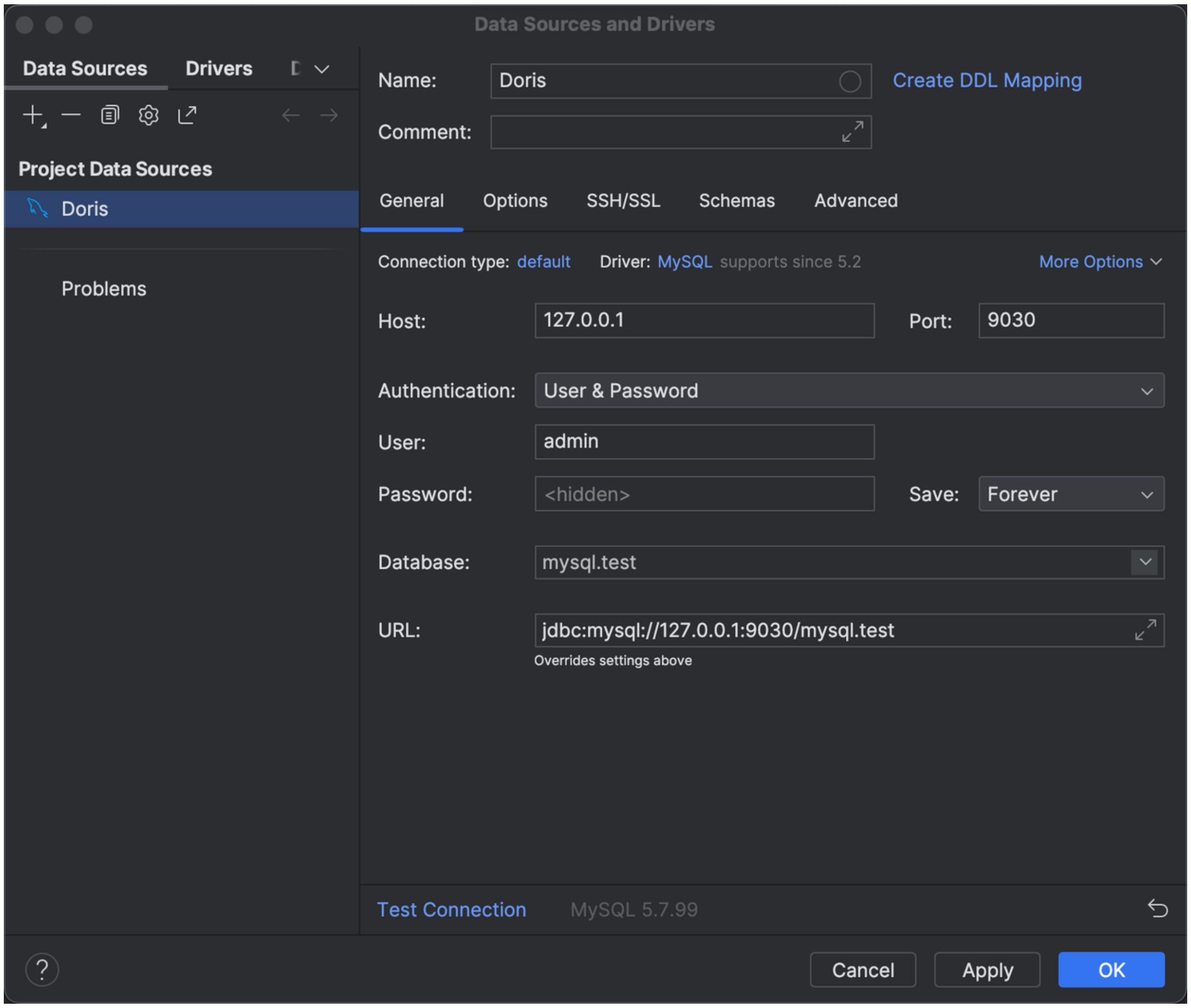
- internal catalog



Name: Doris
Comment:
Connection type: default   Driver: MySQL supports since 5.2
Host: 127.0.0.1   Port: 9030
Authentication: User & Password
User: admin
Password: <hidden>   Save: Forever
Database: test
URL: jdbc:mysql://127.0.0.1:9030/test
Overrides settings above
MySQL 5.7.99

Figure 58: connect internal catalog

- external catalog


Data Sources and Drivers

Name: Doris

Comment:

Connection type: default Driver: MySQL supports since 5.2

Host: 127.0.0.1 Port: 9030

Authentication: User & Password

User: admin

Password: <hidden> Save: Forever

Database: mysql.test

URL: jdbc:mysql://127.0.0.1:9030/mysql.test

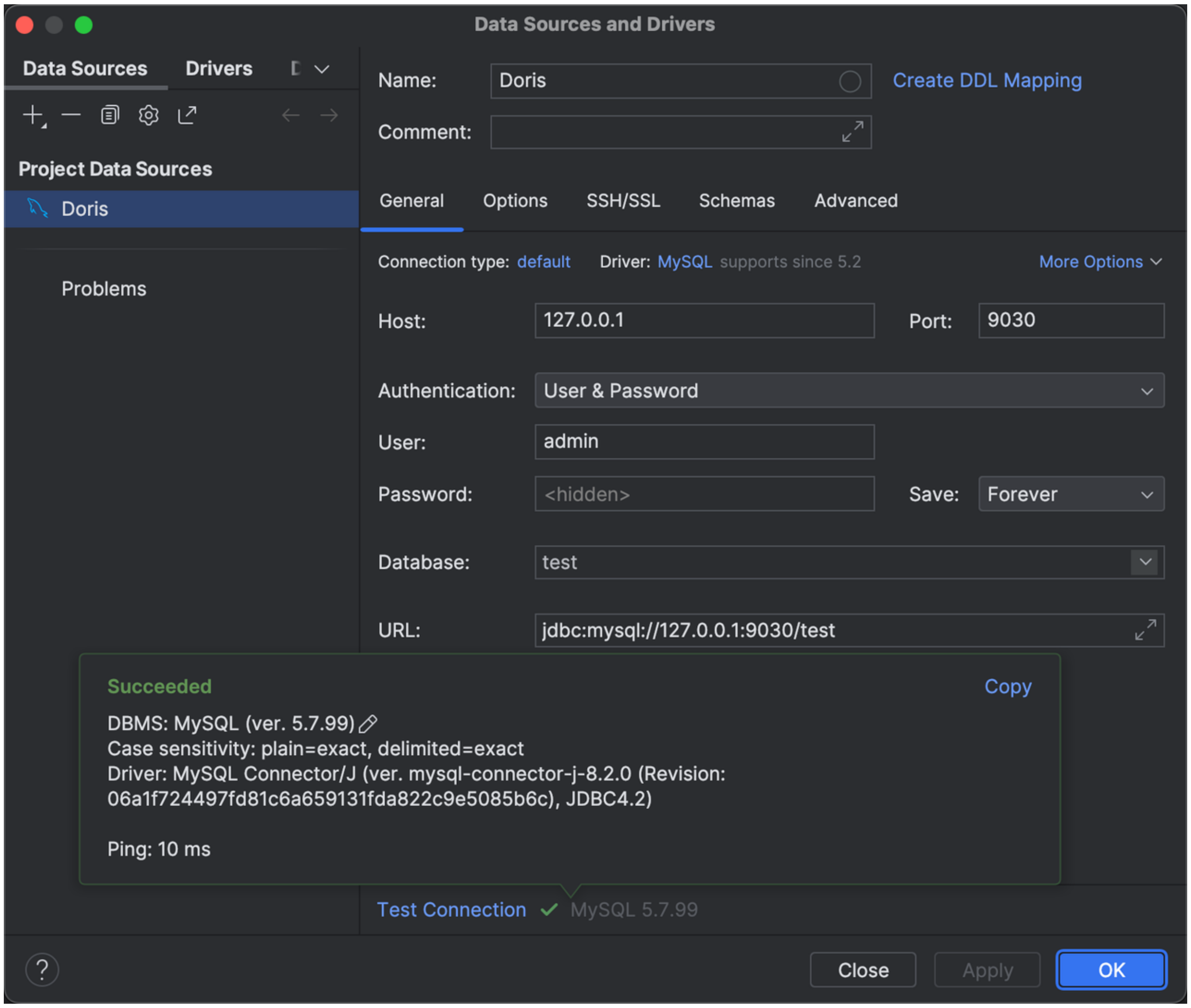
Overrides settings above

Test Connection MySQL 5.7.99

Figure 59: 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.



Name: Doris
Comment:
Connection type: default Driver: MySQL supports since 5.2
Host: 127.0.0.1 Port: 9030
Authentication: User & Password
User: admin
Password: <hidden> Save: Forever
Database: test
URL: jdbc:mysql://127.0.0.1:9030/test

```
Succeeded
DBMS: MySQL (ver. 5.7.99)
Case sensitivity: plain=exact, delimited=exact
Driver: MySQL Connector/J (ver. mysql-connector-j-8.2.0 (Revision:
06a1f724497fd81c6a659131fda822c9e5085b6c), JDBC4.2)

Ping: 10 ms
```

Test Connection ✓ MySQL 5.7.99

Figure 60: 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.

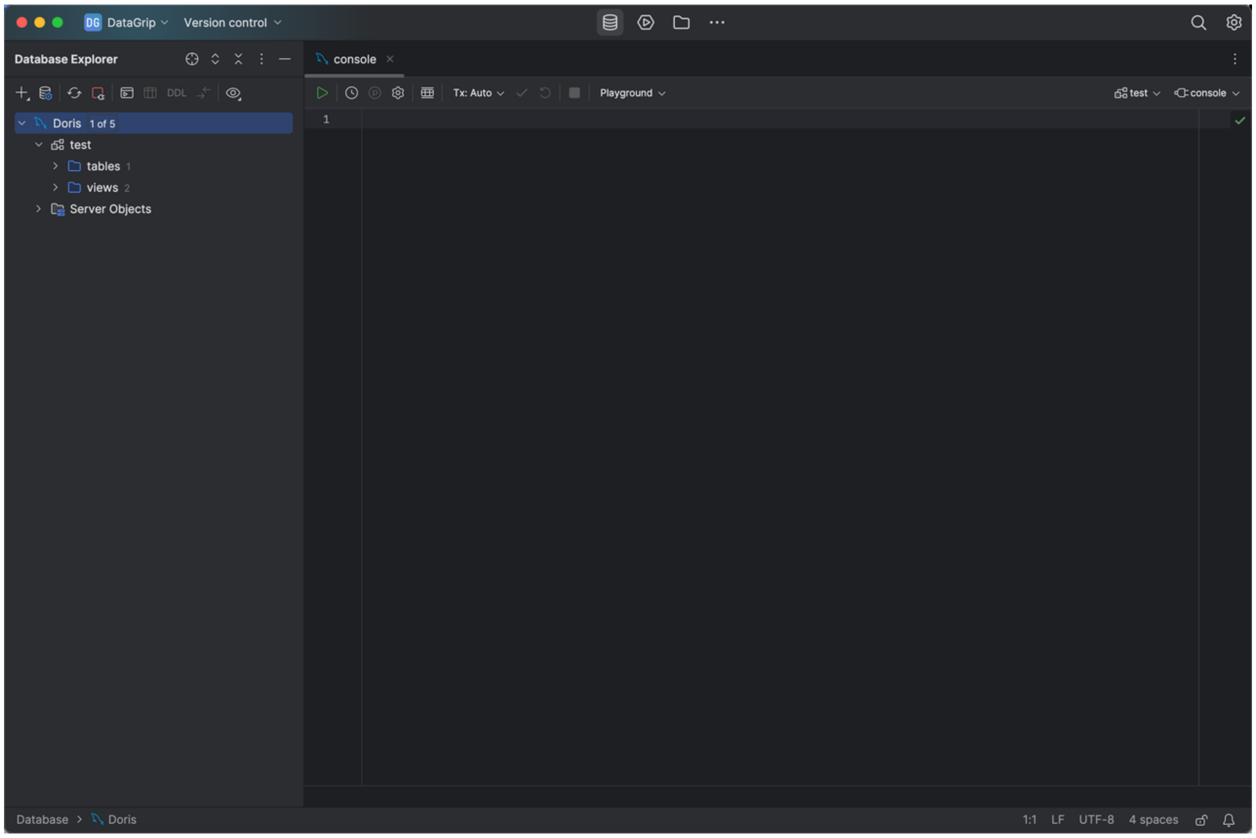


Figure 61: create connection

4.15.3.4 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.

4.15.4 FineBI

4.15.4.1 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.

4.15.4.2 Precondition

Install FineBI 5.0 or later, Download link: https://intl.finebi.com/

4.15.4.3 Login and Connection

1. Create account and log in FineBI

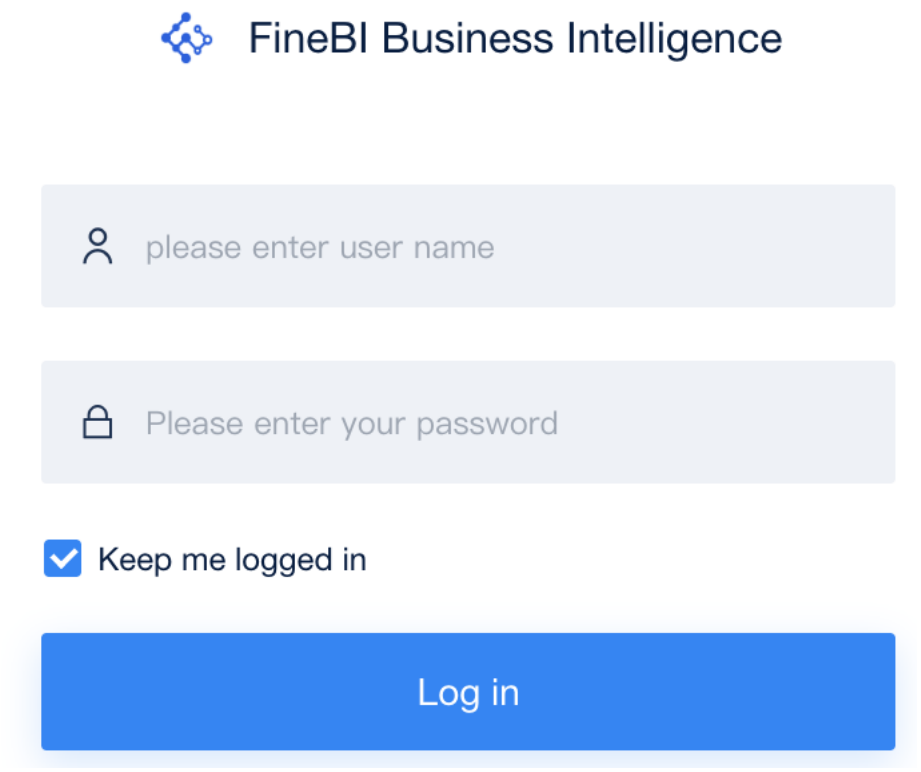
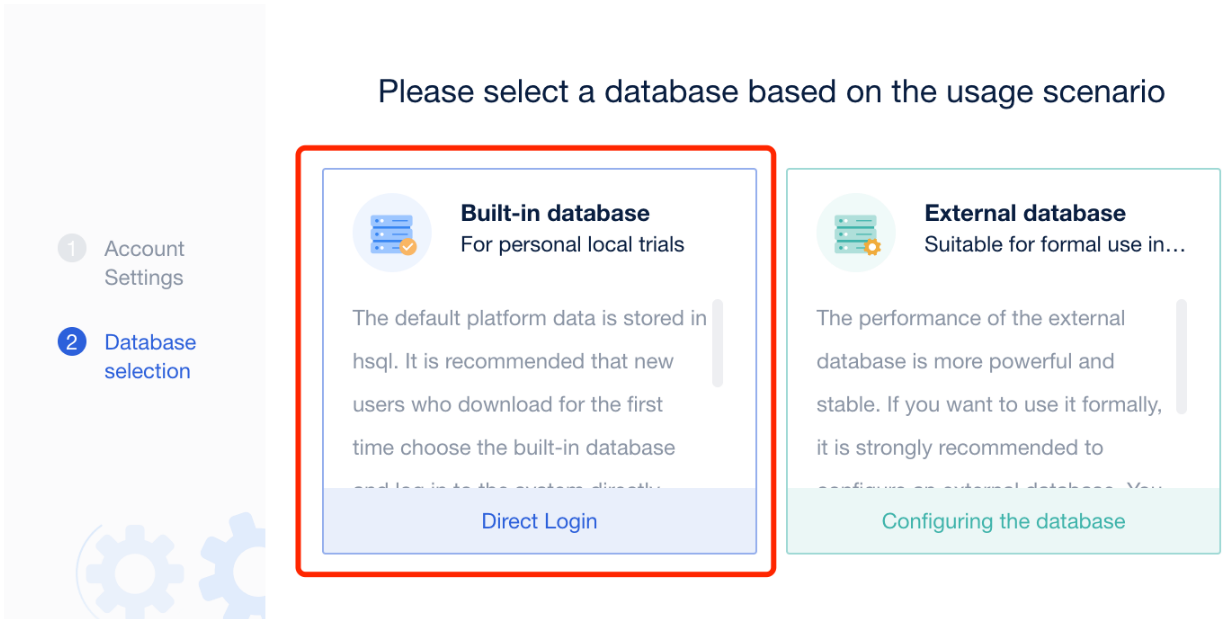


Figure 62: login page

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

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.

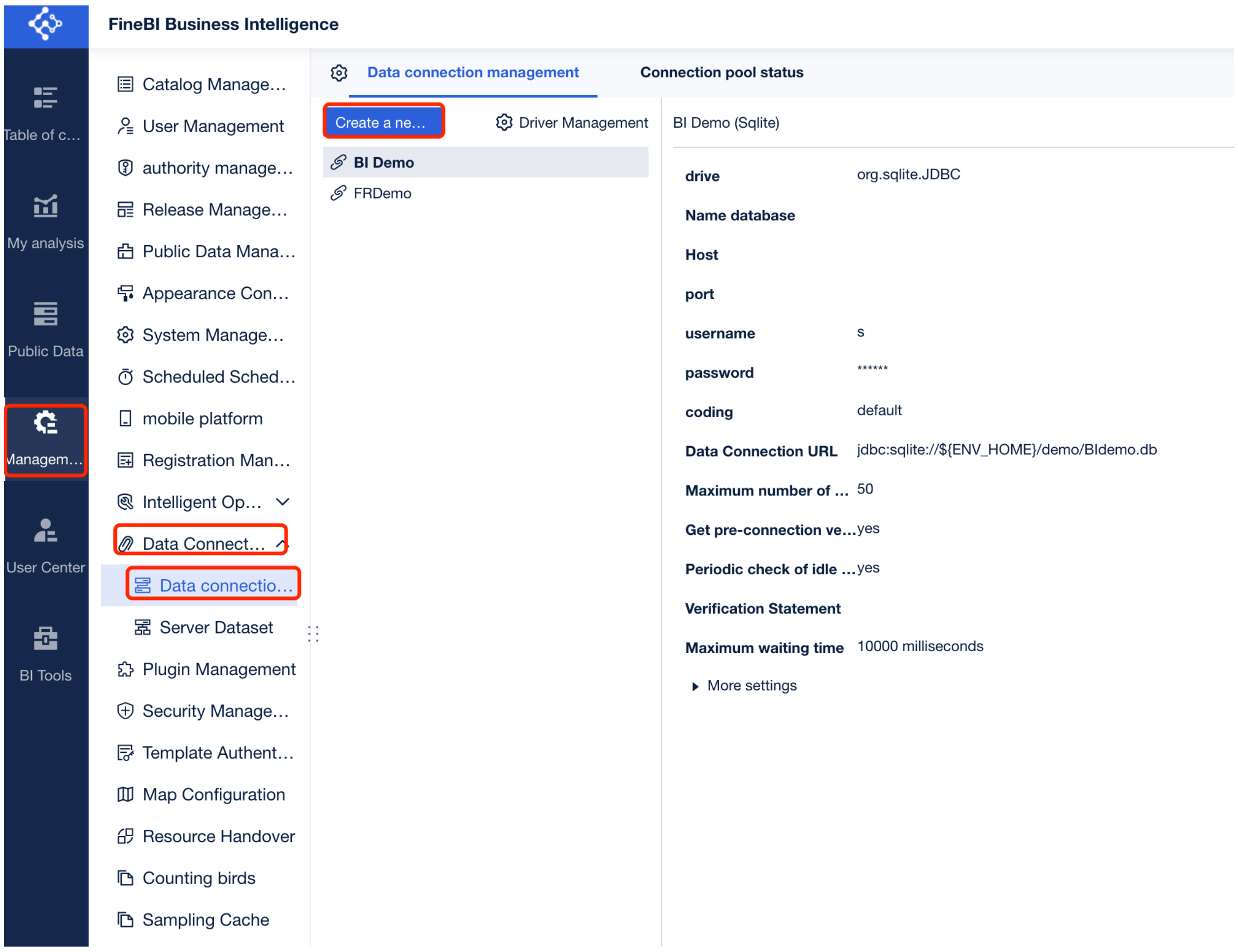


The default platform data is stored in hsql. It is recommended that new users who download for the first time choose the built-in database

The performance of the external database is more powerful and stable. If you want to use it formally, it is strongly recommended to

Figure 63: select database

3. Click the Management System button and select the database connection management in Data Connections to create a new database connection.



drive	org.sqlite.JDBC
Name database	
Host	
port	
username	s
password	******
coding	default
Data Connection URL	jdbc:sqlite://${ENV_HOME}/demo/BIdemo.db
Maximum number of …	50
Get pre-connection ve…	yes
Periodic check of idle …	yes
Verification Statement	
Maximum waiting time	10000 milliseconds

Figure 64: data connection

4. On the new database connection page, select MySQL database

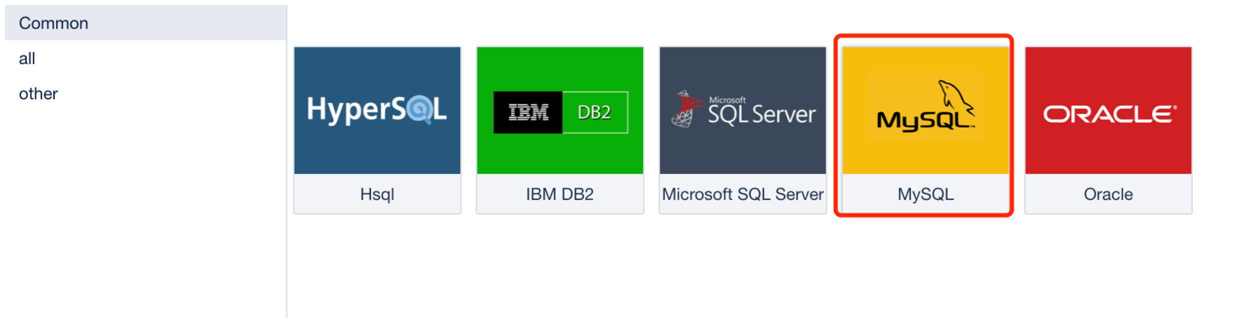
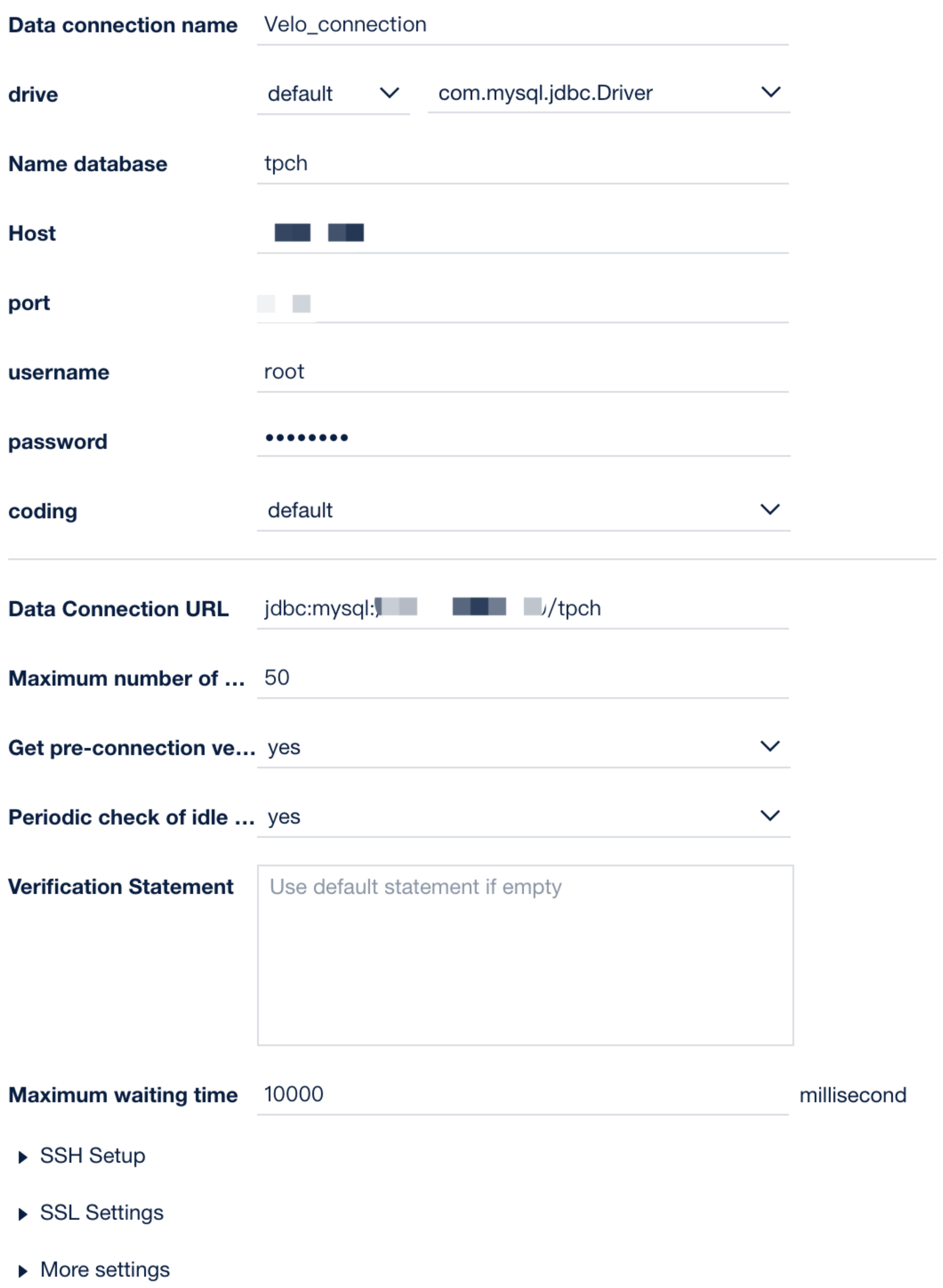


Figure 65: select connection

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。



Data connection name	Velo_connection
drive	default ⌄ com.mysql.jdbc.Driver ⌄
Name database	tpch
Host	[redacted]
port	[redacted]
username	root
password	•••••••••
coding	default ⌄
Data Connection URL	jdbc:mysql:[redacted]/tpch
Maximum number of ...	50
Get pre-connection ve...	yes ⌄
Periodic check of idle ...	yes ⌄
Verification Statement	Use default statement if empty
Maximum waiting time	10000 millisecond

▸ SSH Setup

▸ SSL Settings

▸ More settings

Figure 66: connection information

6. Click the test link. Connection succeeded is displayed when the connection information is correct

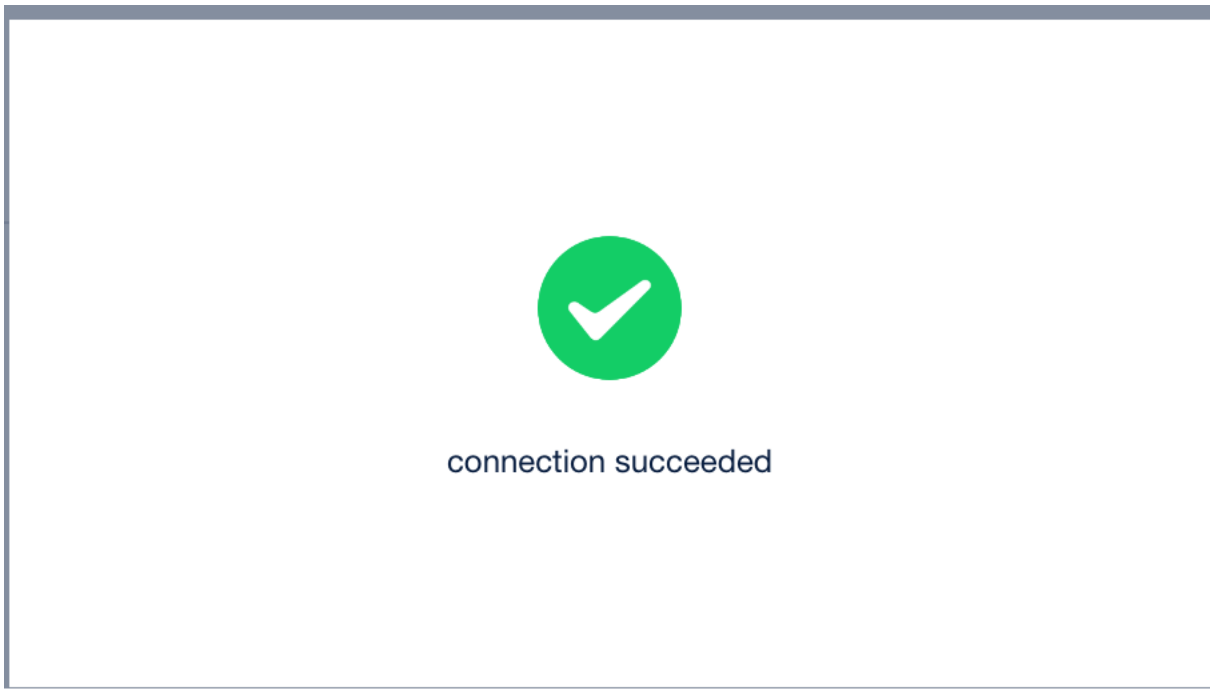


Figure 67: connection test

4.15.4.4 Create model

1. In the “Public Data” section, click to create a new dataset. Then click the database table

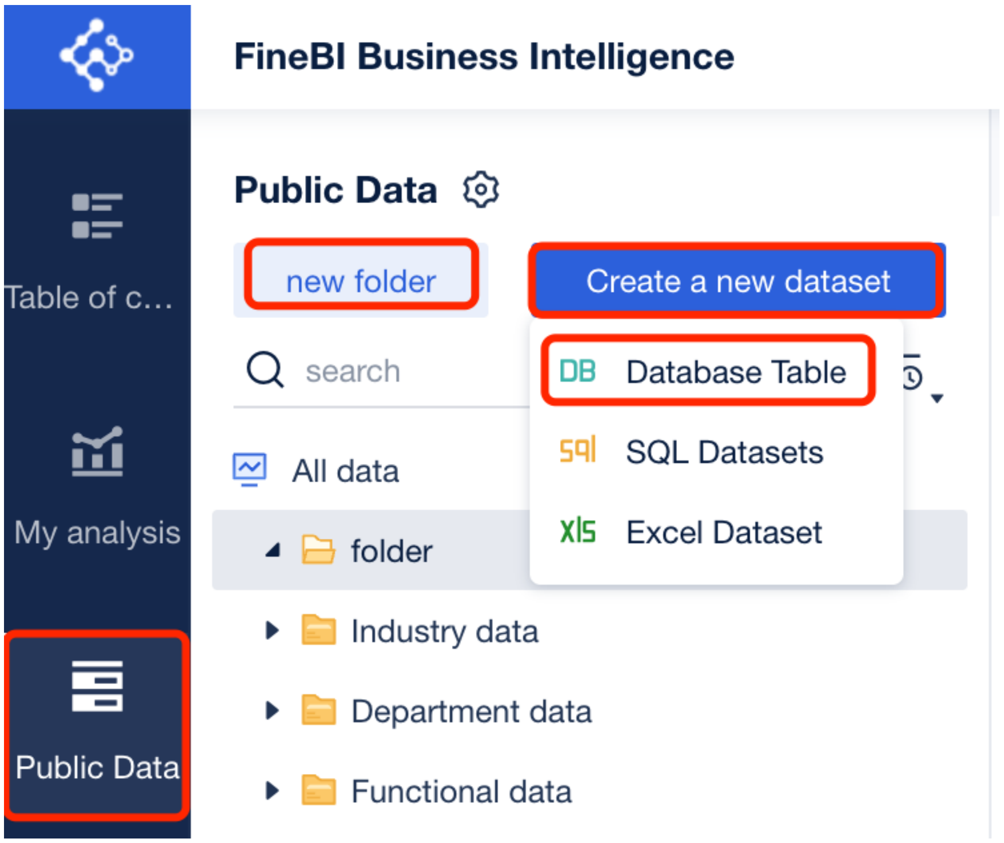


Figure 68: new dataset

2. You need to import tables in the existing database connection

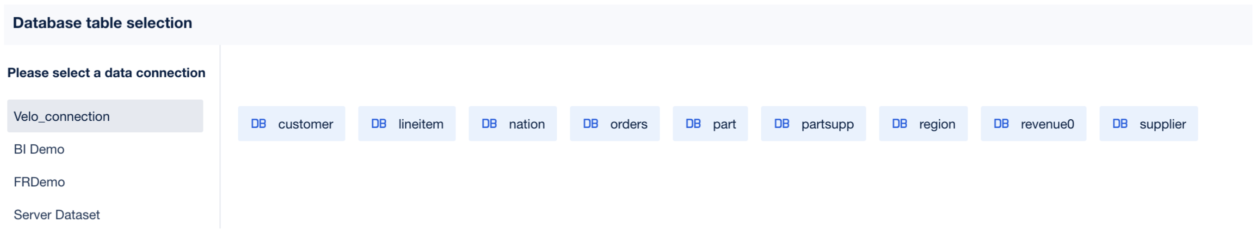


Figure 69: select table

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

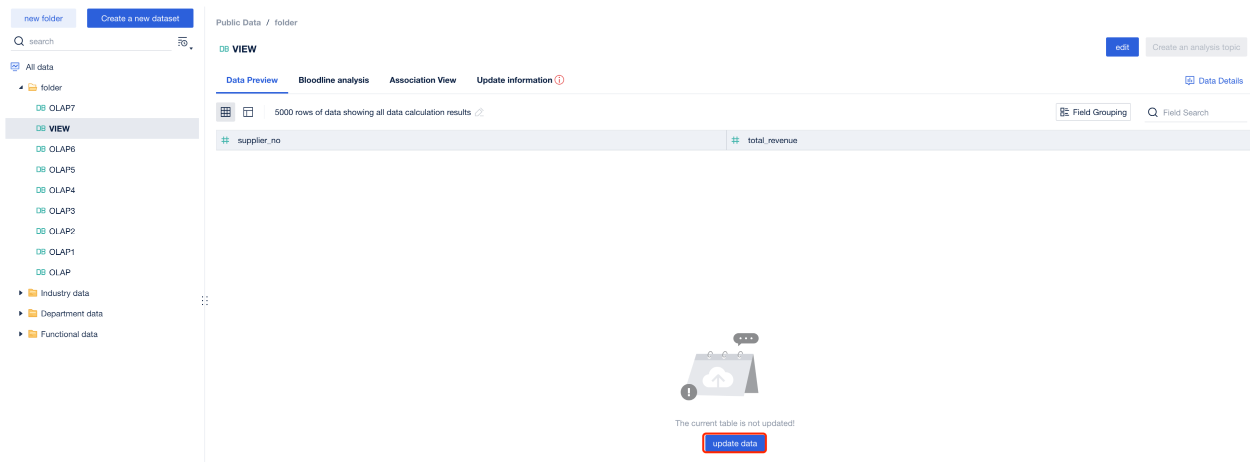
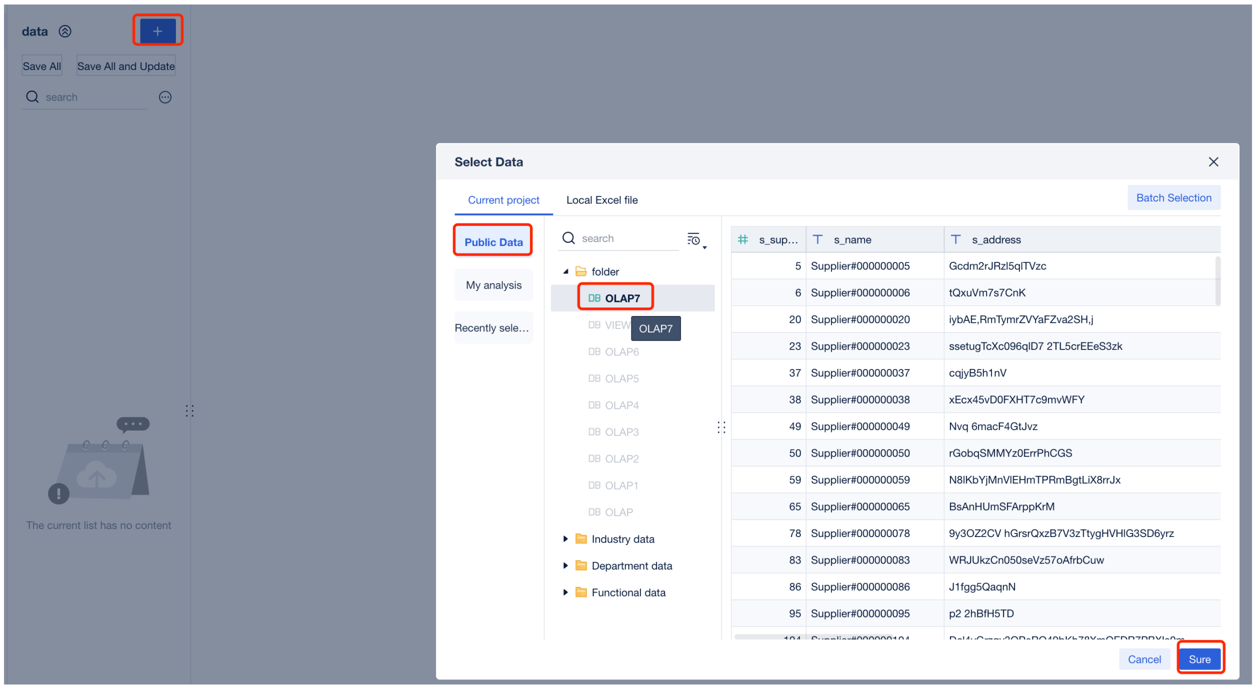


Figure 70: refresh table

4. Add the imported public data to the edited topic, and then conduct compass analysis and configuration according to business logic.



s_sup…	s_name	s_address
5	Supplier#000000005	Gcdm2rJRzl5qlTVzc
6	Supplier#000000006	tQxuVm7s7CnK
20	Supplier#000000020	iybAE,RmTymrZVYaFZva2SH,j
23	Supplier#000000023	ssetugTcXc096qlD7 2TL5crEEeS3zk
37	Supplier#000000037	cqjyB5h1nV
38	Supplier#000000038	xEcx45vD0FXHT7c9mvWFY
49	Supplier#000000049	Nvq 6macF4GtJvz
50	Supplier#000000050	rGobqSMMYz0ErrPhCGS
59	Supplier#000000059	N8lKbYjMnVlEHmTPRmBgtLiX8rrJx
65	Supplier#000000065	BsAnHUmSFArppKrM
78	Supplier#000000078	9y3OZ2CV hGrsrQxzB7V3zTtygHVHlG3SD6yrz
83	Supplier#000000083	WRJUkzCn050seVz57oAfrbCuw
86	Supplier#000000086	J1fgg5QaqnN
95	Supplier#000000095	p2 2hBfH5TD

Figure 71: data analysis

4.15.5 Power BI

4.15.5.1 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 and Apache Doris through MySQL Driver. At present, internal data modeling, data query and visualization processing of Apache Doris have been officially supported in Power BI.

4.15.5.2 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

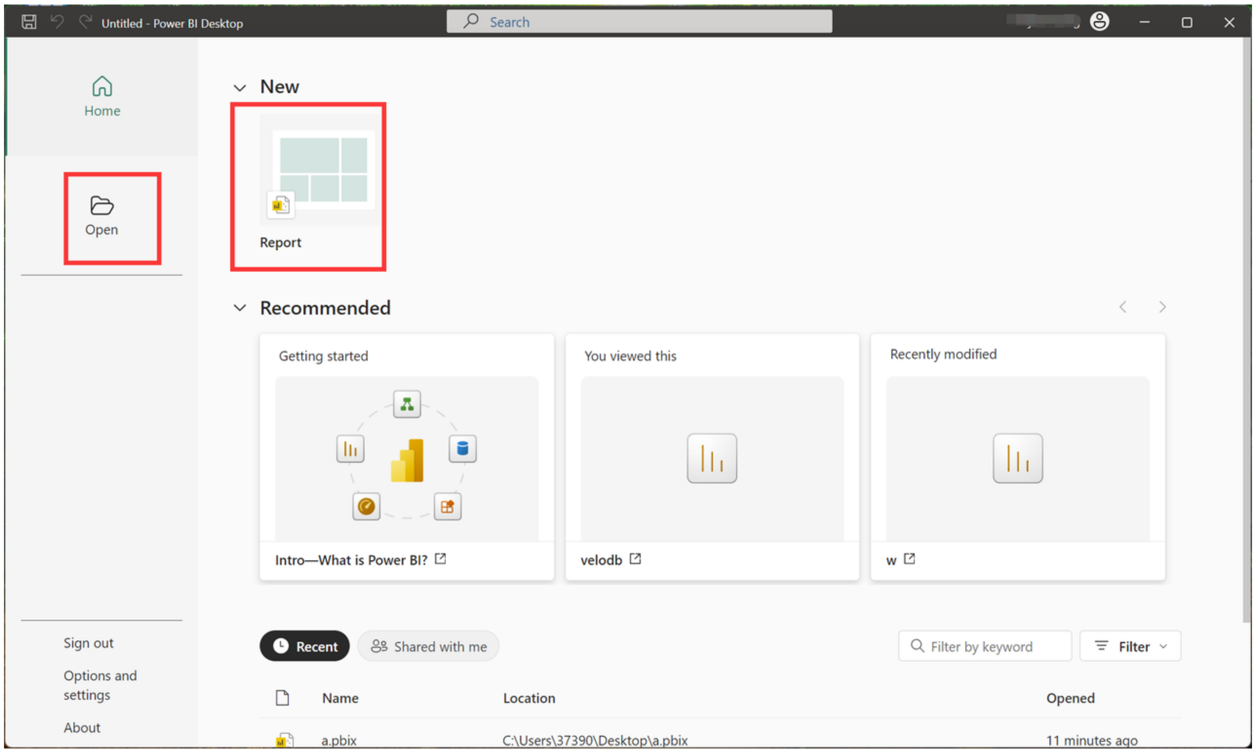
4.15.5.3 Connector configuration of Power BI and Doris

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

4.15.5.4 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.



Name	Location	Opened
a.pbix	C:\Users\37390\Desktop\a.pbix	11 minutes ago

Figure 72: start page

3. Click get data. In the dialog box that is displayed, select MySQL database.

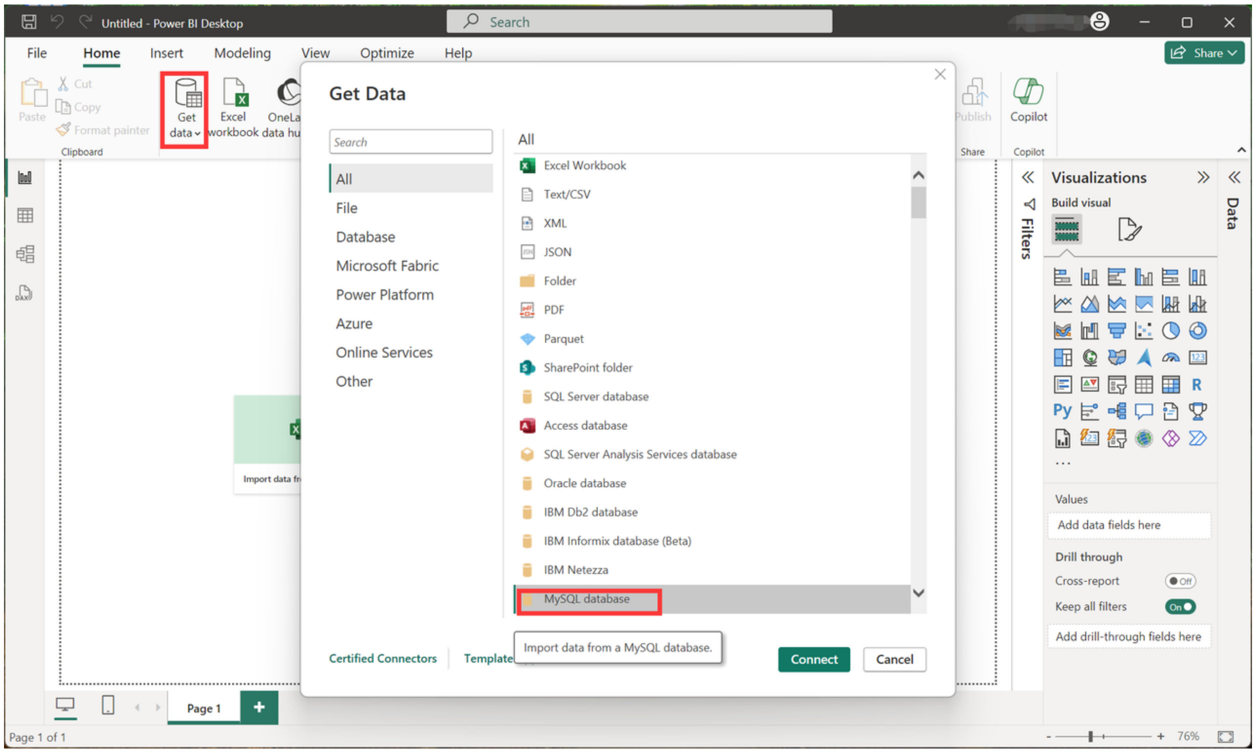


Figure 73: get data

4. Configure the database connection information and enter ip:port in the server text box. The default port number for Doris is 9030.

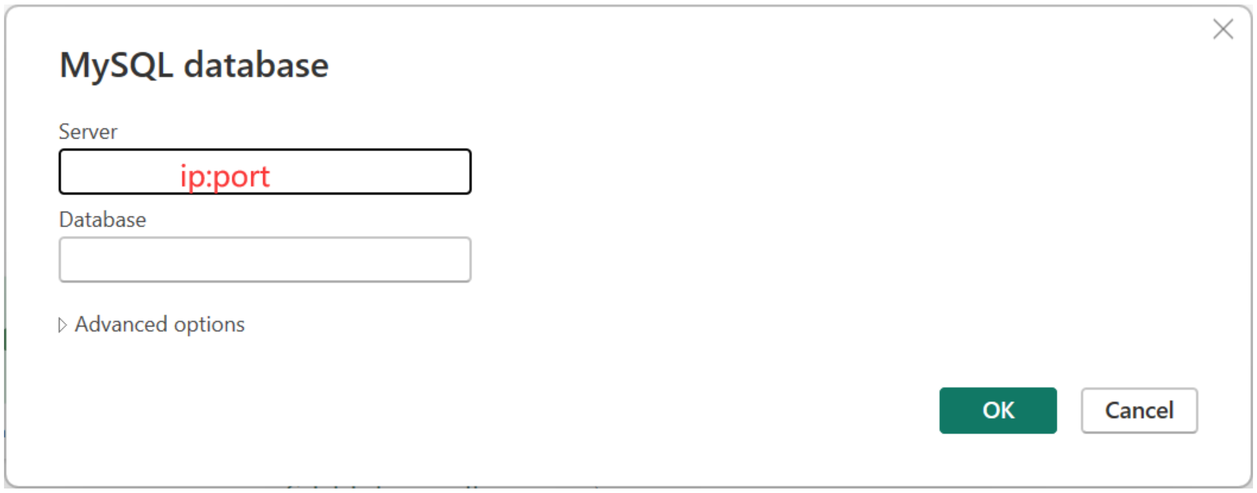


Figure 74: connection information

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.

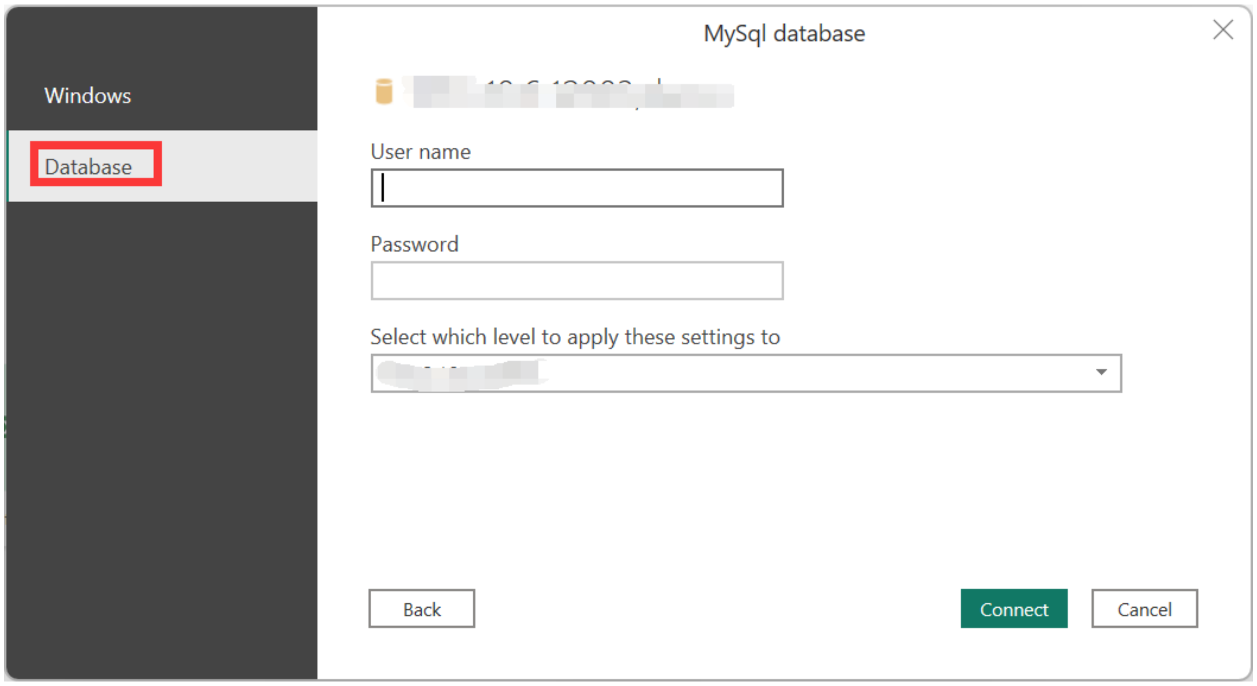
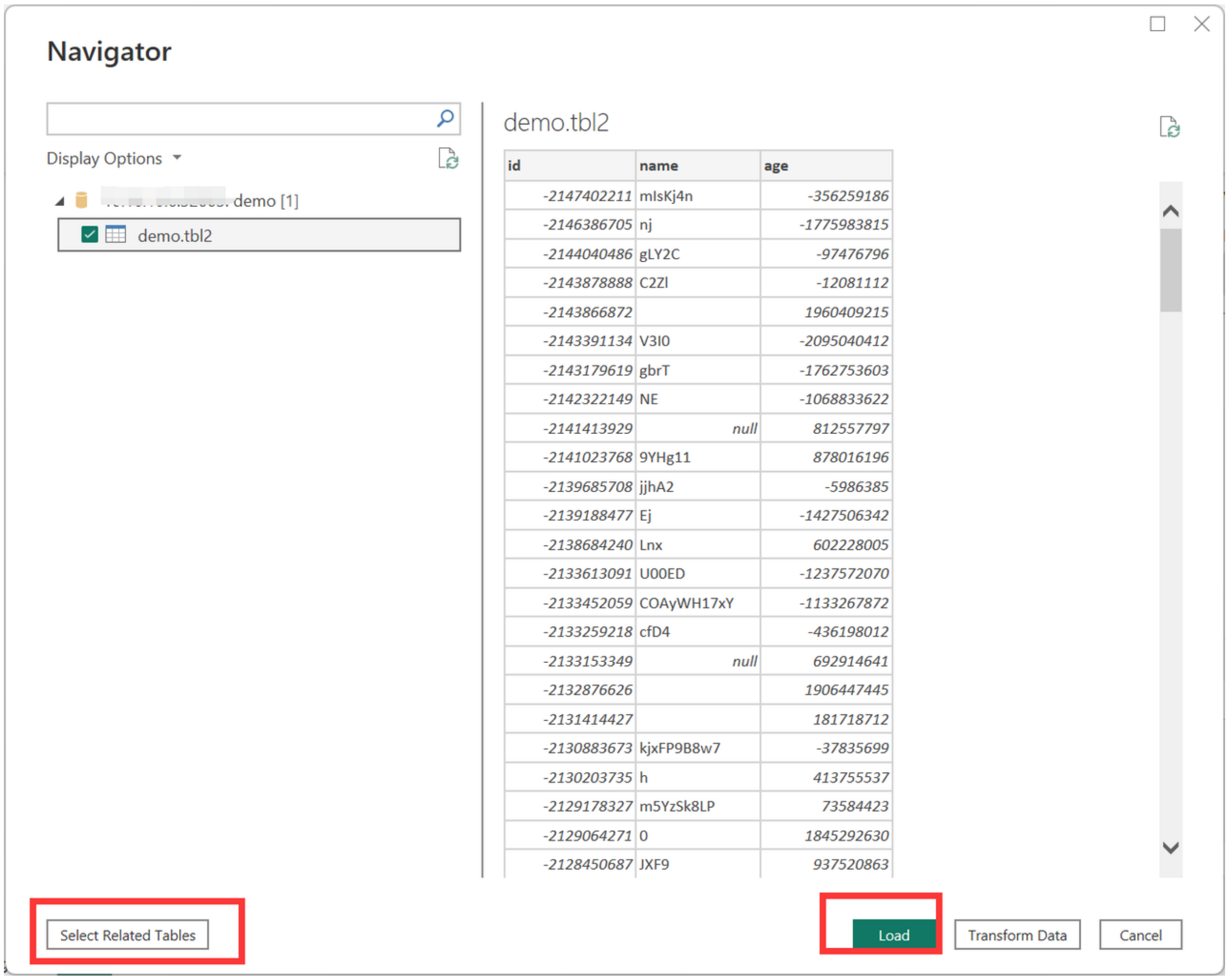


Figure 75: uname and pwd

6. Load the selected table to the Power BI Desktop



Navigator

Display Options

[redacted] demo [1]

demo.tbl2

demo.tbl2

id	name	age
-2147402211	mIsKj4n	-356259186
-2146386705	nj	-1775983815
-2144040486	gLY2C	-97476796
-2143878888	C2Zl	-12081112
-2143866872		1960409215
-2143391134	V3I0	-2095040412
-2143179619	gbrT	-1762753603
-2142322149	NE	-1068833622
-2141413929	null	812557797
-2141023768	9YHg11	878016196
-2139685708	jjhA2	-5986385
-2139188477	Ej	-1427506342
-2138684240	Lnx	602228005
-2133613091	U00ED	-1237572070
-2133452059	COAyWH17xY	-1133267872
-2133259218	cfD4	-436198012
-2133153349	null	692914641
-2132876626		1906447445
-2131414427		181718712
-2130883673	kjxFP9B8w7	-37835699
-2130203735	h	413755537
-2129178327	m5YzSk8LP	73584423
-2129064271	0	1845292630
-2128450687	JXF9	937520863

Figure 76: load data

7. Configure statistical compass

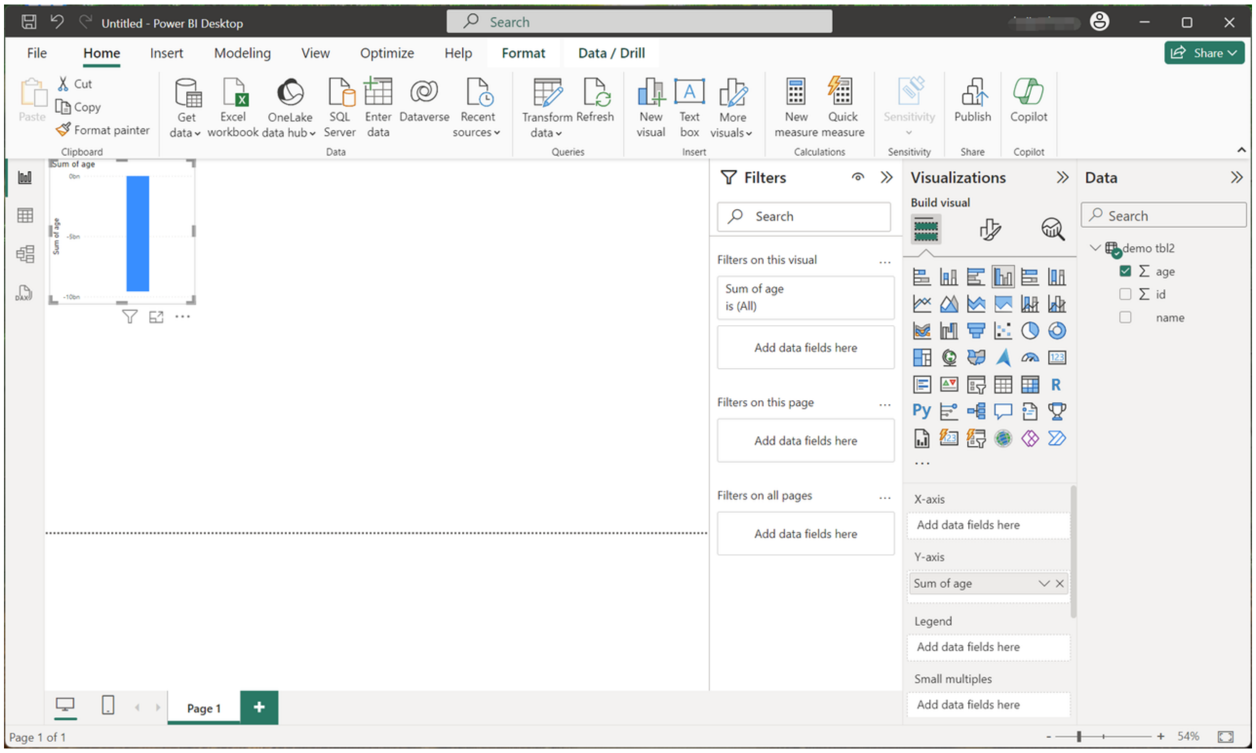
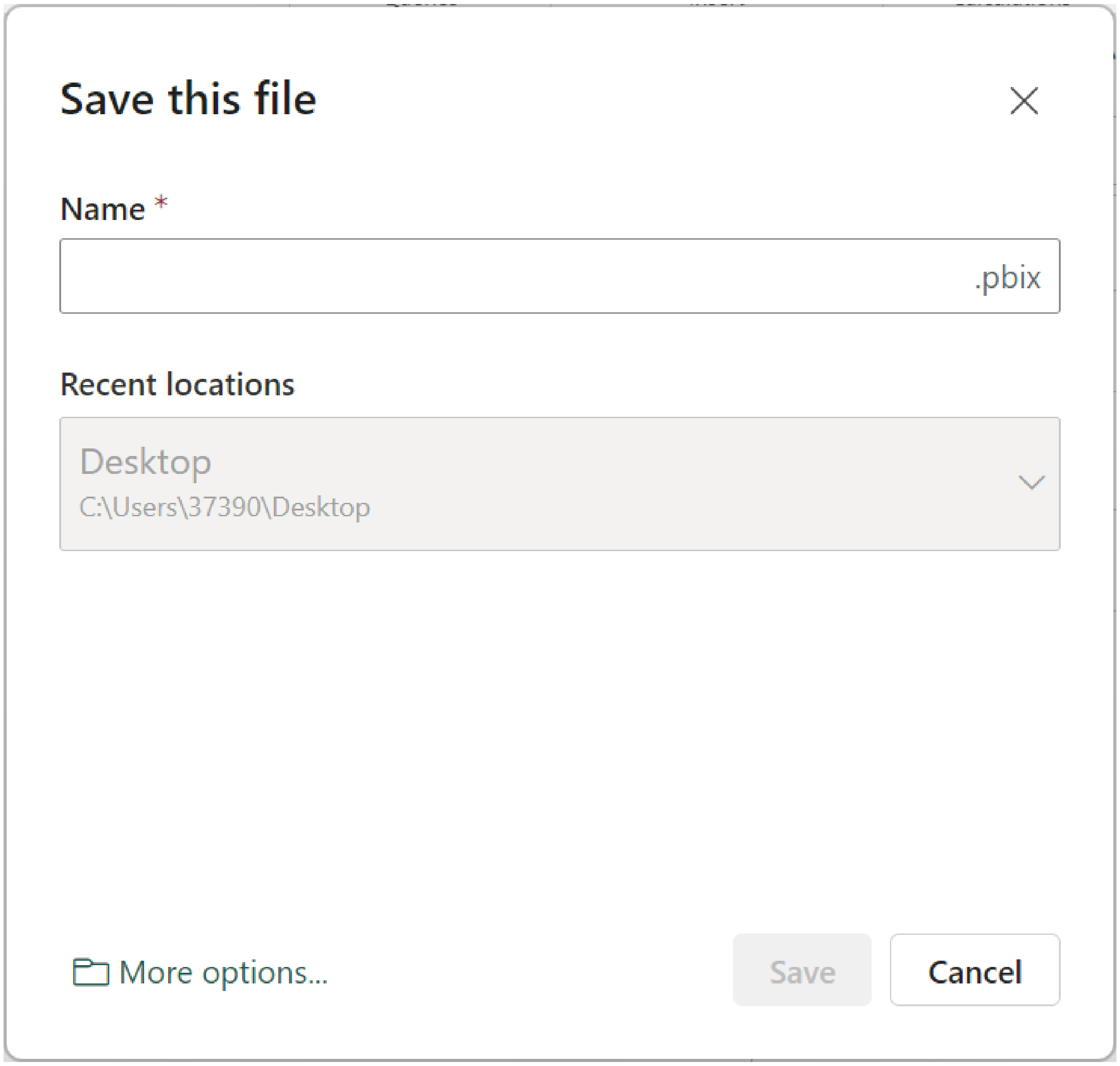


Figure 77: create compass

8. Save statistical compass to location


Save this file

Name *

.pbix

Recent locations

Desktop
C:\Users\37390\Desktop

More options...

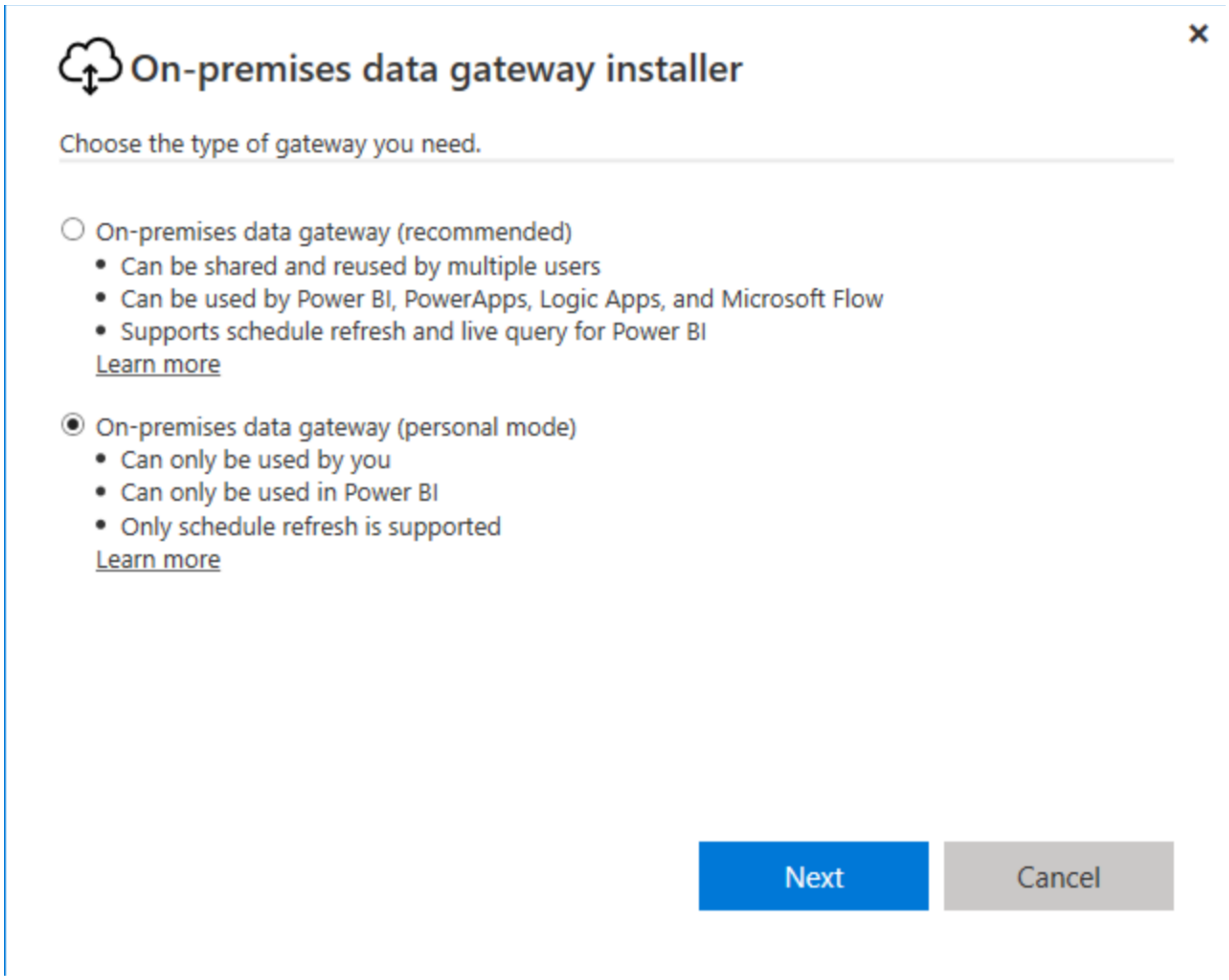
Save

Cancel

Figure 78: save file

4.15.5.5 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



On-premises data gateway installer

Choose the type of gateway you need.

( ) On-premises data gateway (recommended)

- Can be shared and reused by multiple users
- Can be used by Power BI, PowerApps, Logic Apps, and Microsoft Flow
- Supports schedule refresh and live query for Power BI

Learn more

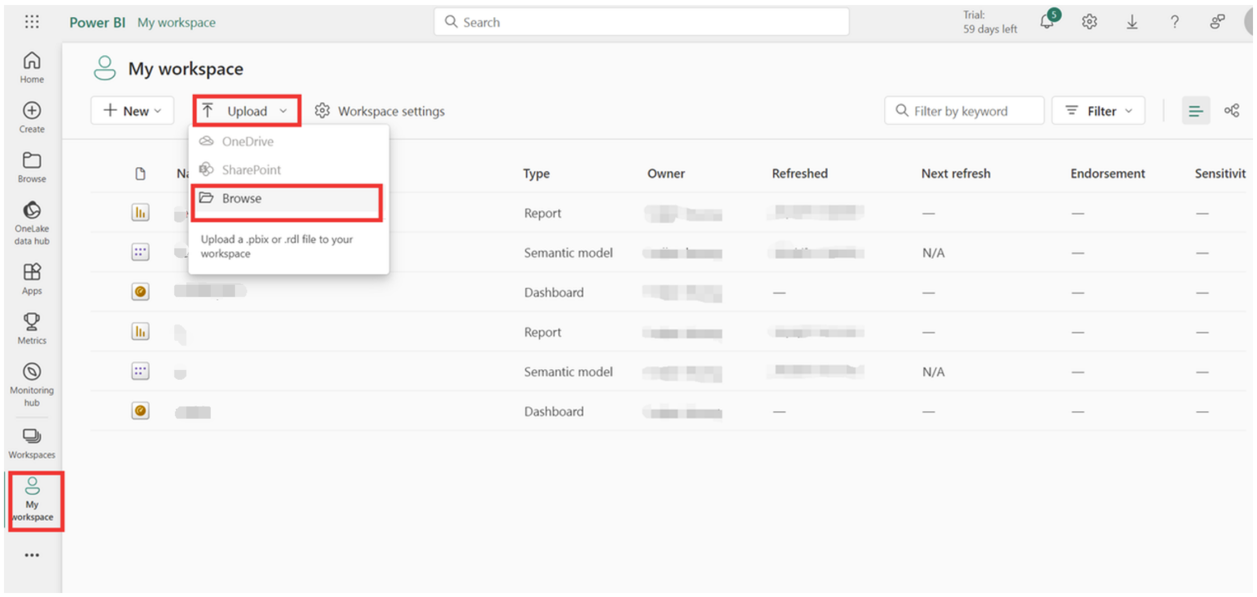
[✓ On-premises data gateway (personal mode)]

- Can only be used by you
- Can only be used in Power BI
- Only schedule refresh is supported

Learn more

Figure 79: gateway install

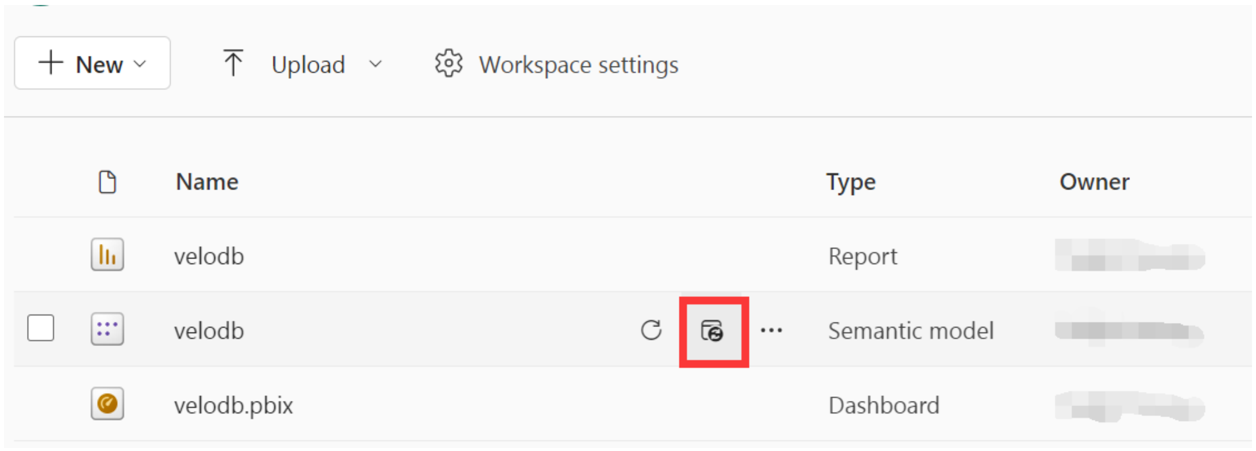
3. Log into Power BI Online and import the local model in your workspace



	Type	Owner	Refreshed	Next refresh	Endorsement	Sensitivit
	Report			—	—	—
	Semantic model			N/A	—	—
	Dashboard		—	—	—	—
	Report			—	—	—
	Semantic model			N/A	—	—
	Dashboard		—	—	—	—

Figure 80: upload

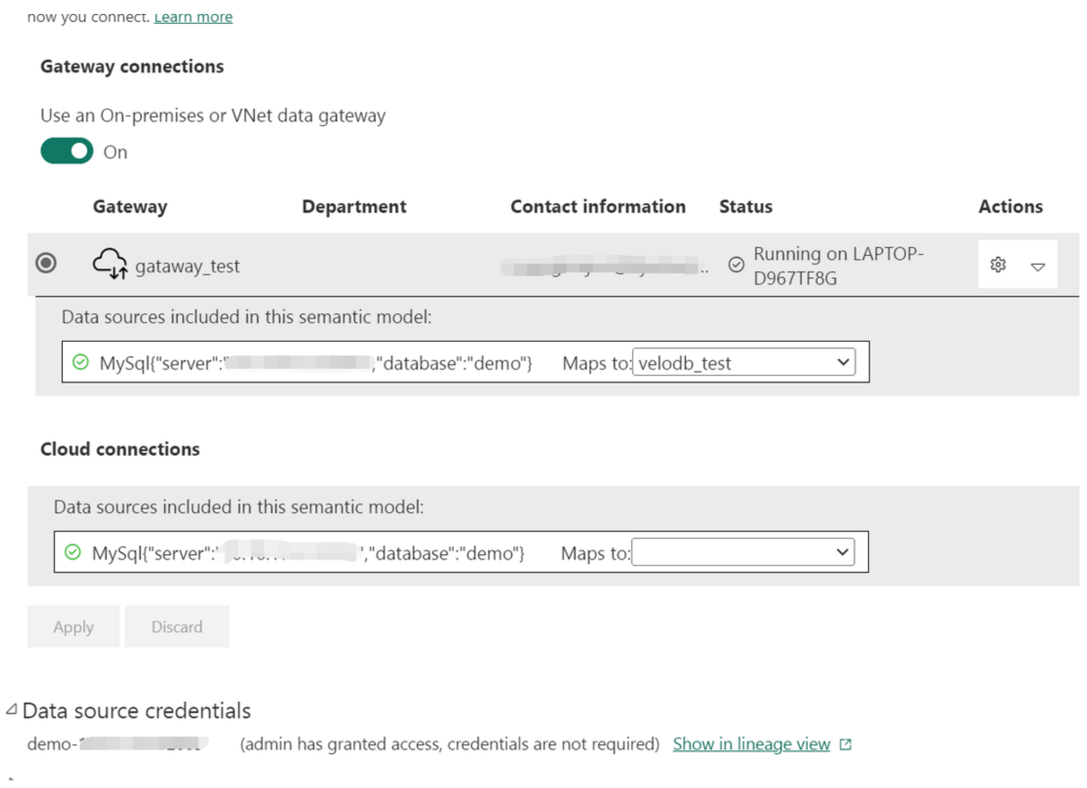
4. Click the model to set the automatic refresh time



Name	Type	Owner
velodb	Report	
velodb	Semantic model	
velodb.pbix	Dashboard	

Figure 81: 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.



how you connect. Learn more

Gateway connections

Use an On-premises or VNet data gateway

On

	Gateway	Department	Contact information	Status	Actions
●	gataway_test		[redacted]..	Running on LAPTOP-D967TF8G	

Data sources included in this semantic model:

MySql{"server":"[redacted]","database":"demo"} Maps to: velodb_test

Cloud connections

Data sources included in this semantic model:

MySql{"server":"[redacted]","database":"demo"} Maps to:

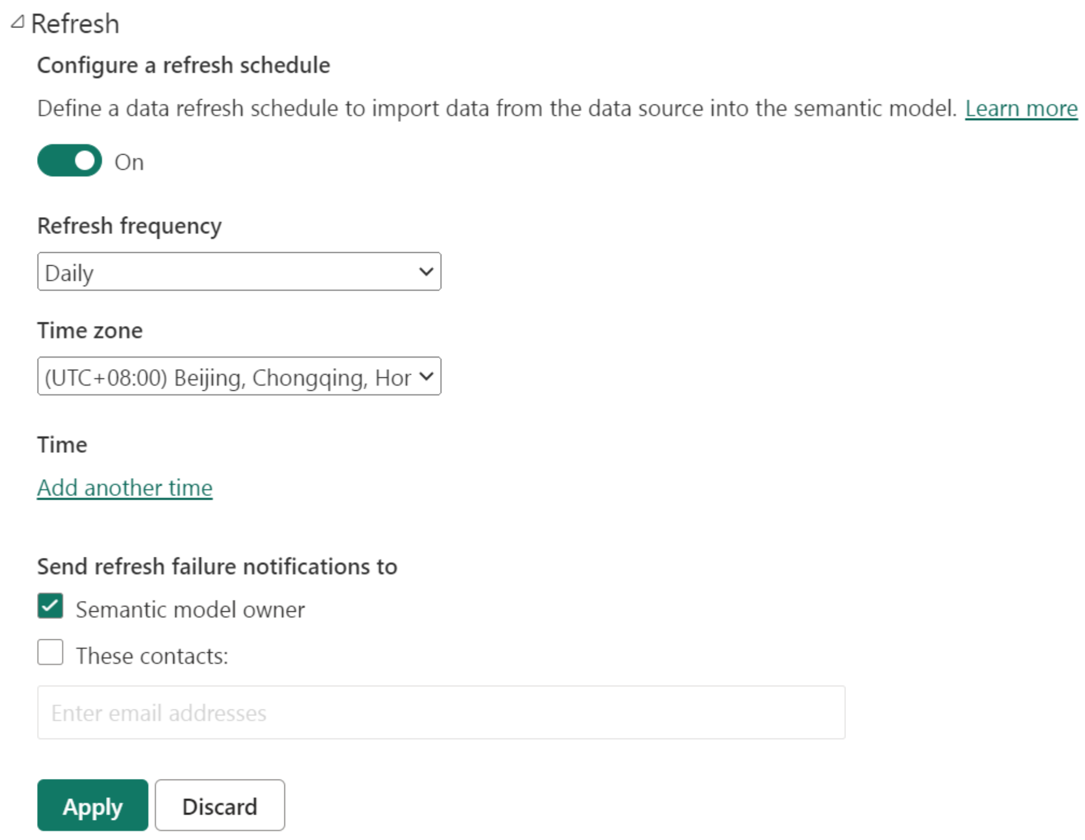
Apply Discard

Data source credentials

demo-[redacted] (admin has granted access, credentials are not required) Show in lineage view

Figure 82: 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

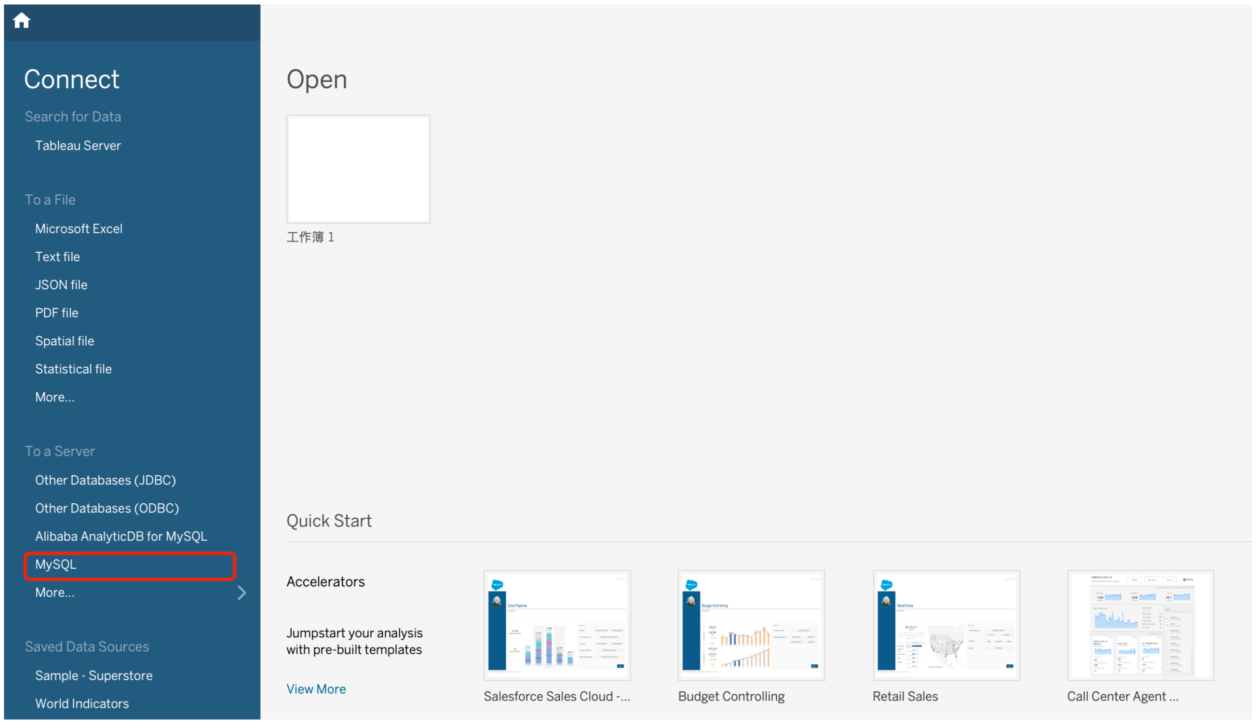
Figure 83: make plan

4.15.6 Tableau

4.15.6.1 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. #### Precondition Tableau Desktop via the following link to download: https://www.tableau.com/products/desktop/download #### Driver installation 1. Install iODBC 1. Close the Tableau Desktop 2. Install iODBC Driver Manager. Obtain the latest version (mxkozzz.dmg) from iODBC.org 3. Click on the downloaded dmg file to install 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


Connect

Search for Data

Tableau Server

To a File

Microsoft Excel

Text file

JSON file

PDF file

Spatial file

Statistical file

More...

To a Server

Other Databases (JDBC)

Other Databases (ODBC)

Alibaba AnalyticDB for MySQL

MySQL

More...

Saved Data Sources

Sample - Superstore

World Indicators

Open

工作簿 1

Quick Start

Accelerators

Jumpstart your analysis with pre-built templates

View More

Salesforce Sales Cloud -...

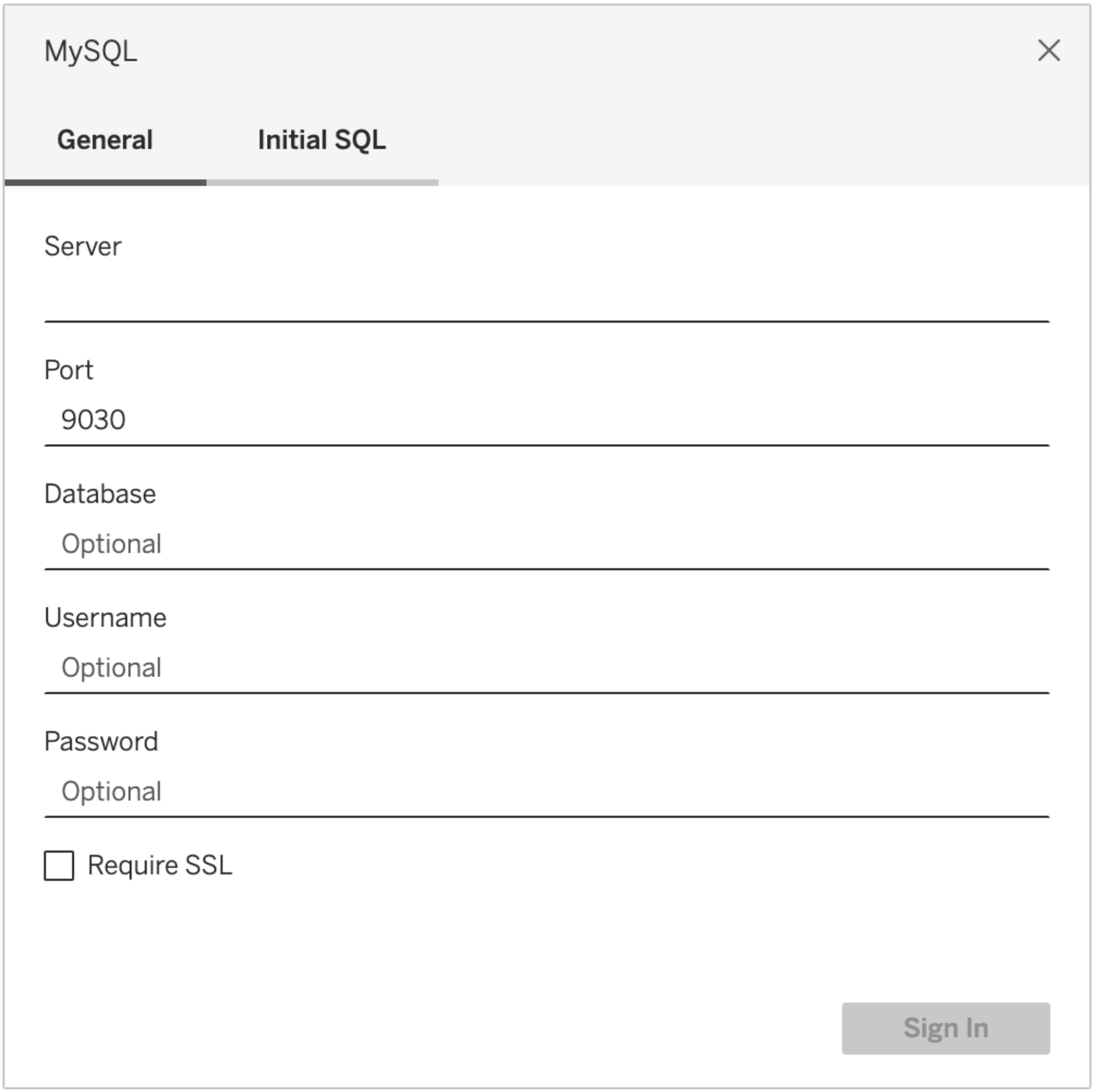
Budget Controlling

Retail Sales

Call Center Agent ...

Figure 84: main page

2. Fill in the Doris server address, port and other relevant information, and click sigin in button after correctly filling



MySQL

General | Initial SQL

Server

Port
9030

Database
Optional

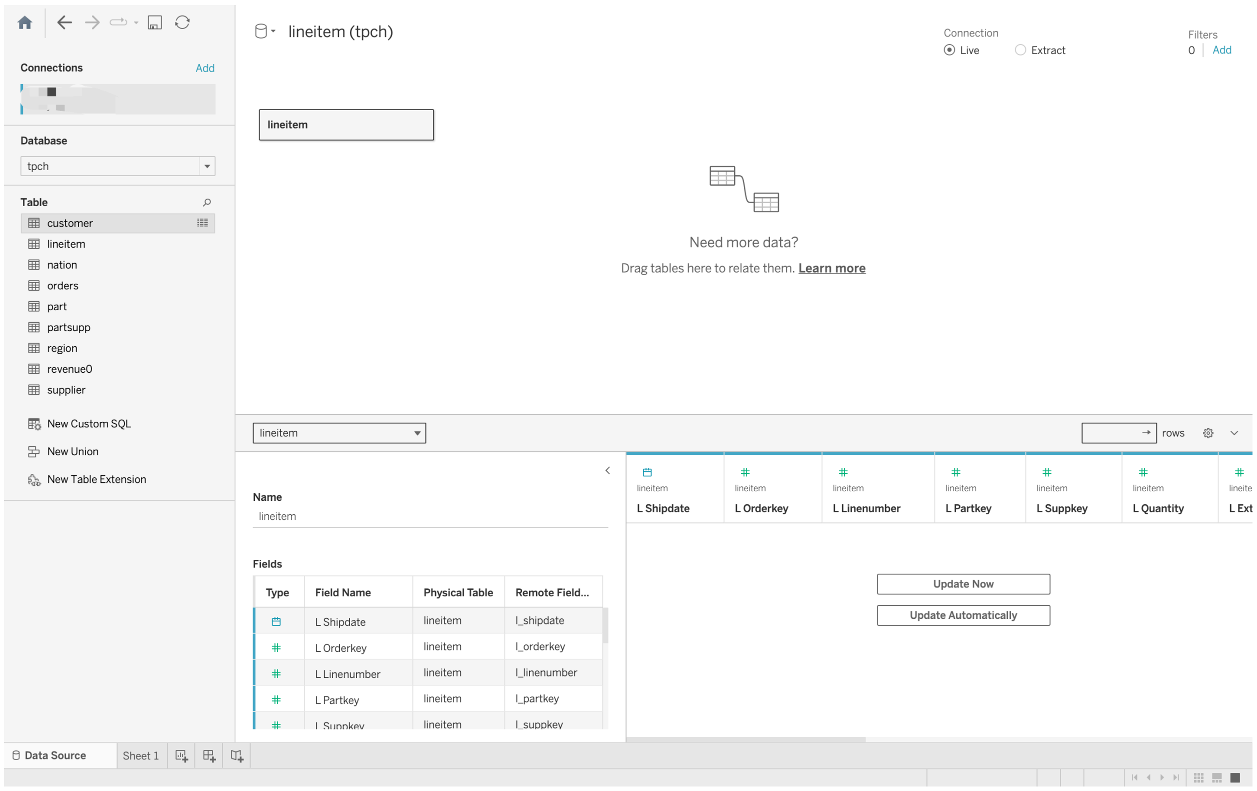
Username
Optional

Password
Optional

☐ Require SSL

Figure 85: sign in page

3. After entering Tableau, select the corresponding library table to carry out the relevant compass processing.



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

Figure 86: usage page

4.15.7 QuickSight

4.15.7.1 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.

4.15.7.2 Data configuration

1. Upon completing the registration process, please proceed to sign in to QuickSight and choose the desired dataset.

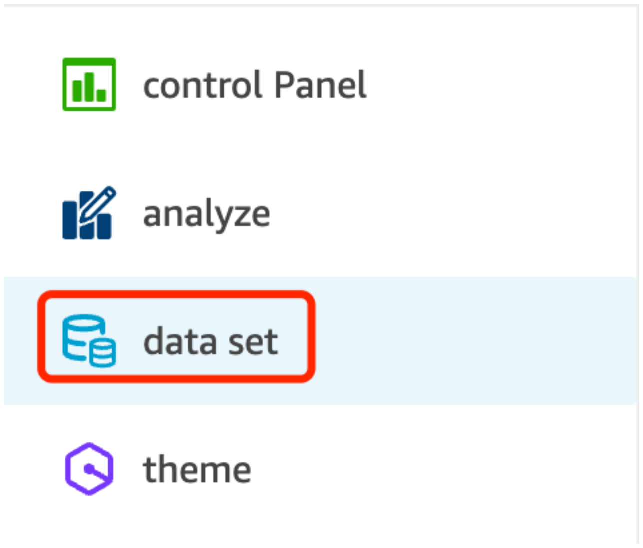


Figure 87: dataset select

2. Select MySQL on the dataset selection page.

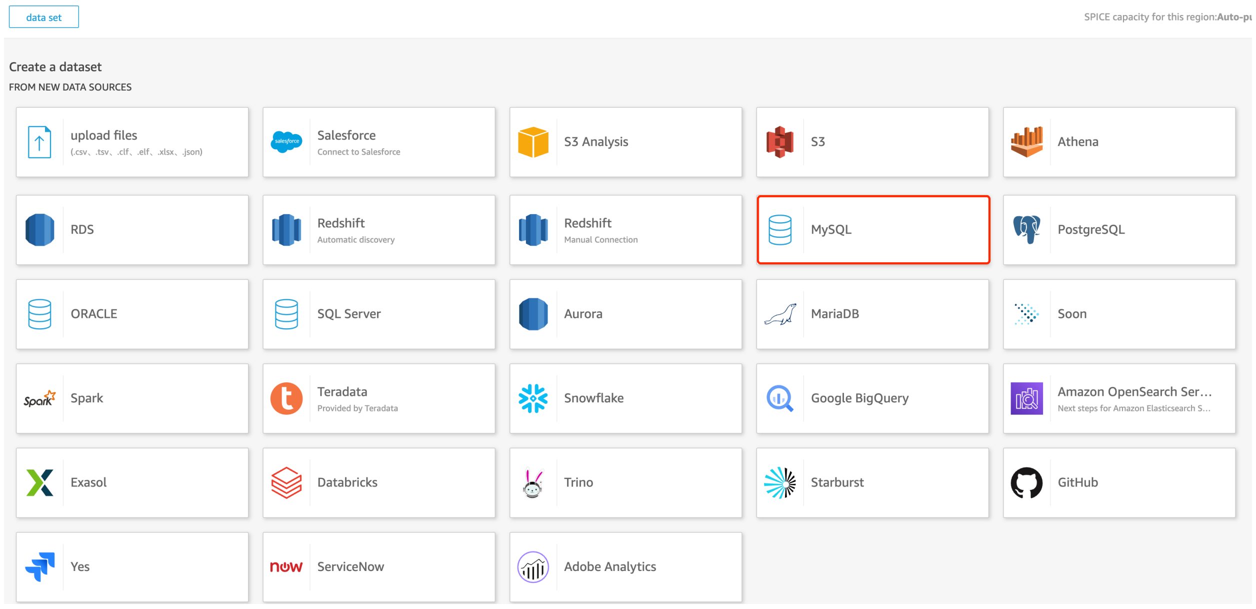
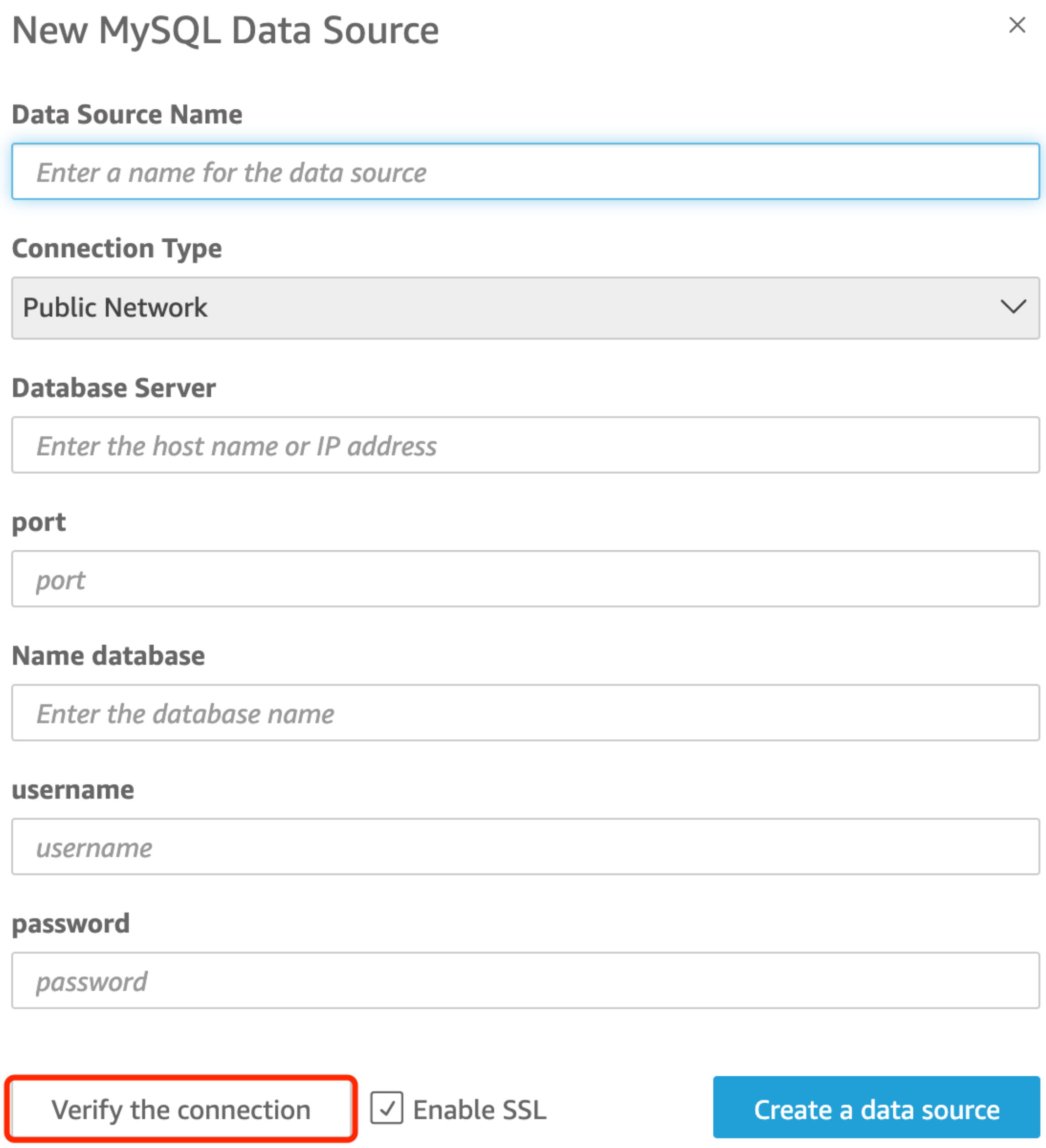


Figure 88: database select

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.



New MySQL Data Source

Data Source Name

Enter a name for the data source

Connection Type

Public Network

Database Server

Enter the host name or IP address

port

port

Name database

Enter the database name

username

username

password

password

Figure 89: connection information

4. Select the table for which you want to create the dashboard.

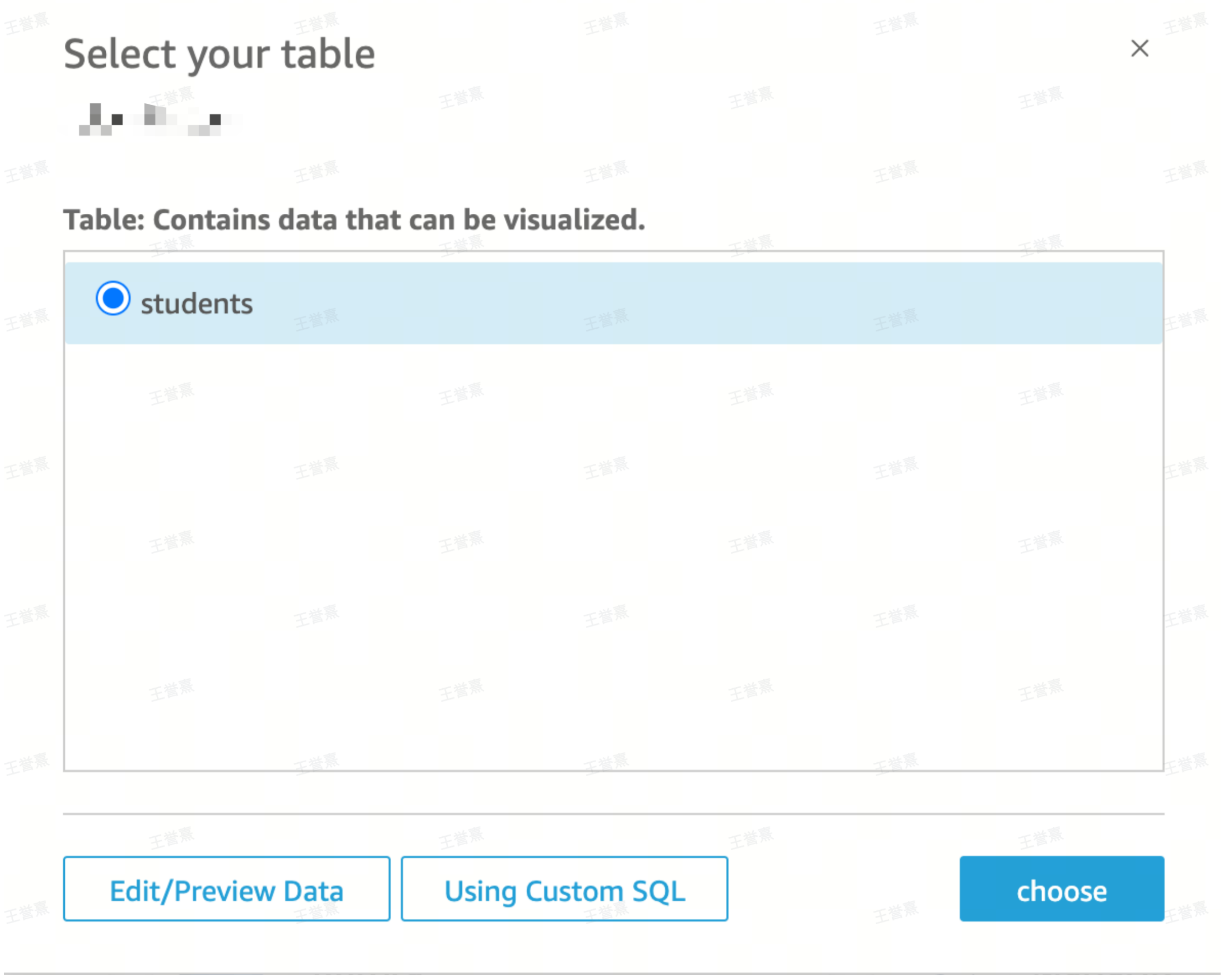
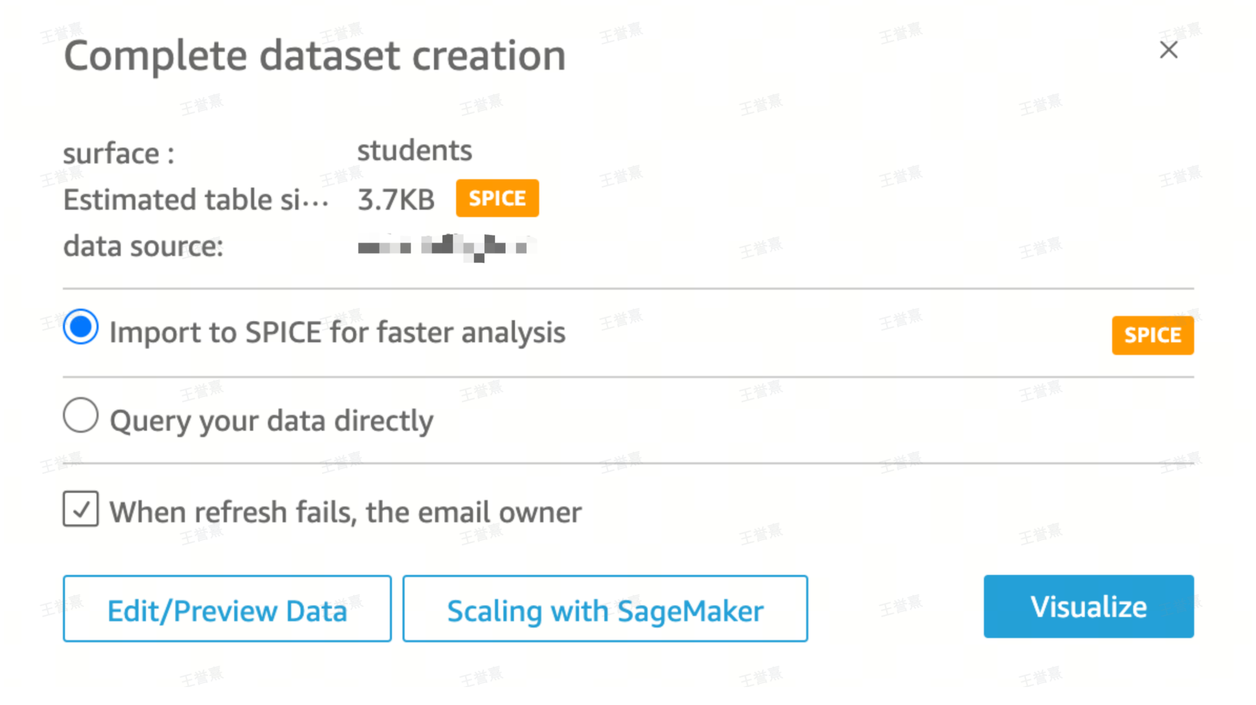


Figure 90: jdbc connector download

5. QuickSight offers two methods for importing tables. You can choose the appropriate import method based on your needs.


Complete dataset creation

surface : students

Estimated table si… 3.7KB SPICE

data source: [redacted]

(●) Import to SPICE for faster analysis SPICE

( ) Query your data directly

☑ When refresh fails, the email owner

Edit/Preview Data | Scaling with SageMaker | Visualize

Figure 91: table select

6. After importing the table, you can create the desired dashboard.

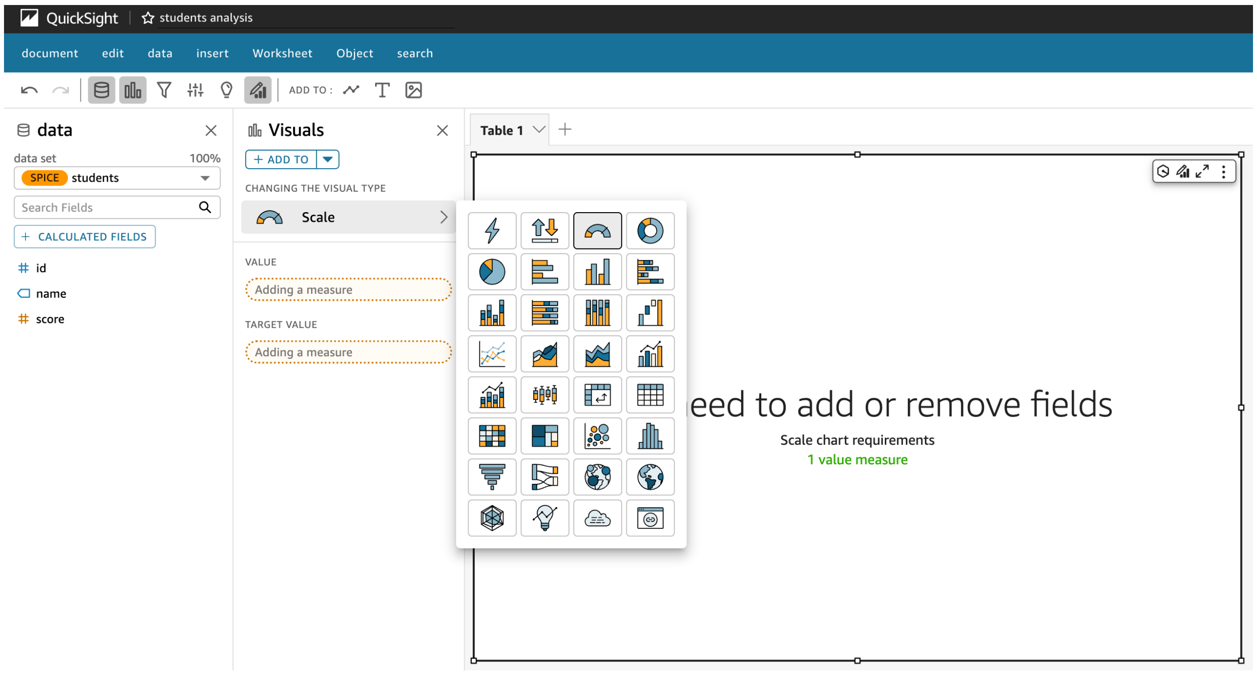


Figure 92: table import

4.15.8 Quick BI

4.15.8.1 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.

4.15.8.2 Data connection and application

1. Login Quick BI and create a workspace.
2. Click Data Source under the current workspace.

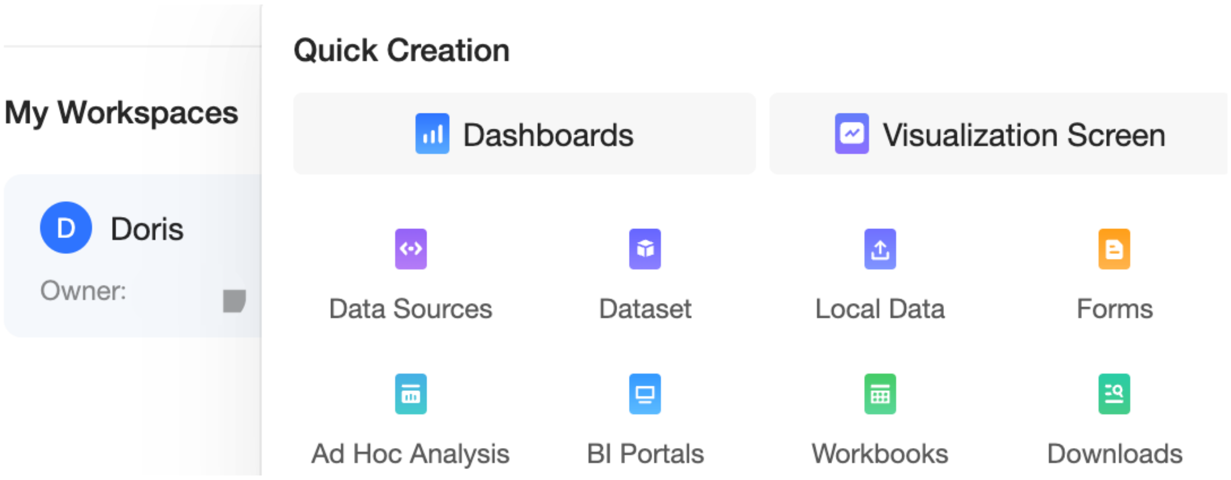
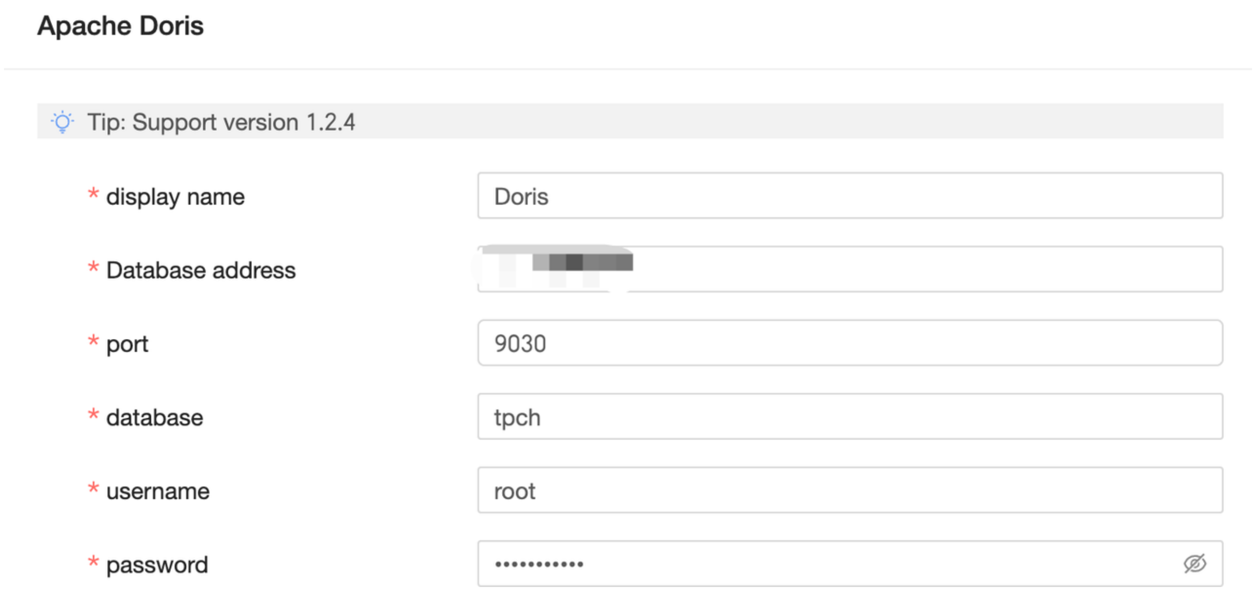


Figure 93: create workspace

3. Select Apache Doris in the already created data source and fill in the corresponding Apache Doris connection information.

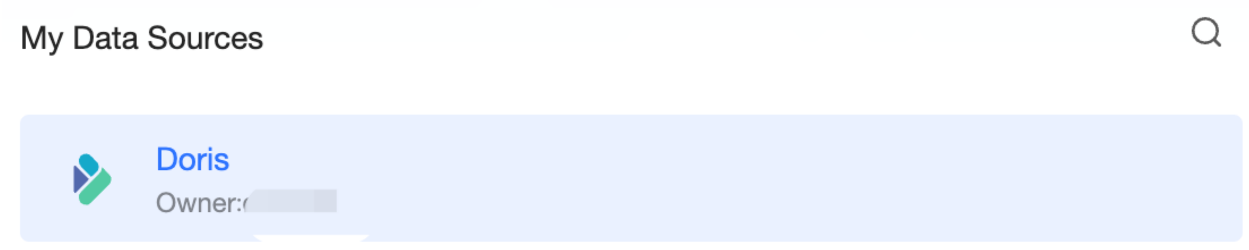

Apache Doris

Tip: Support version 1.2.4

* display name	Doris
* Database address	[redacted]
* port	9030
* database	tpch
* username	root
* password	•••••••••••

Figure 94: Doris information

4. Once the connection is successful, you can see the data source we created in the data source list.


My Data Sources

Doris
Owner: [redacted]

Figure 95: 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.

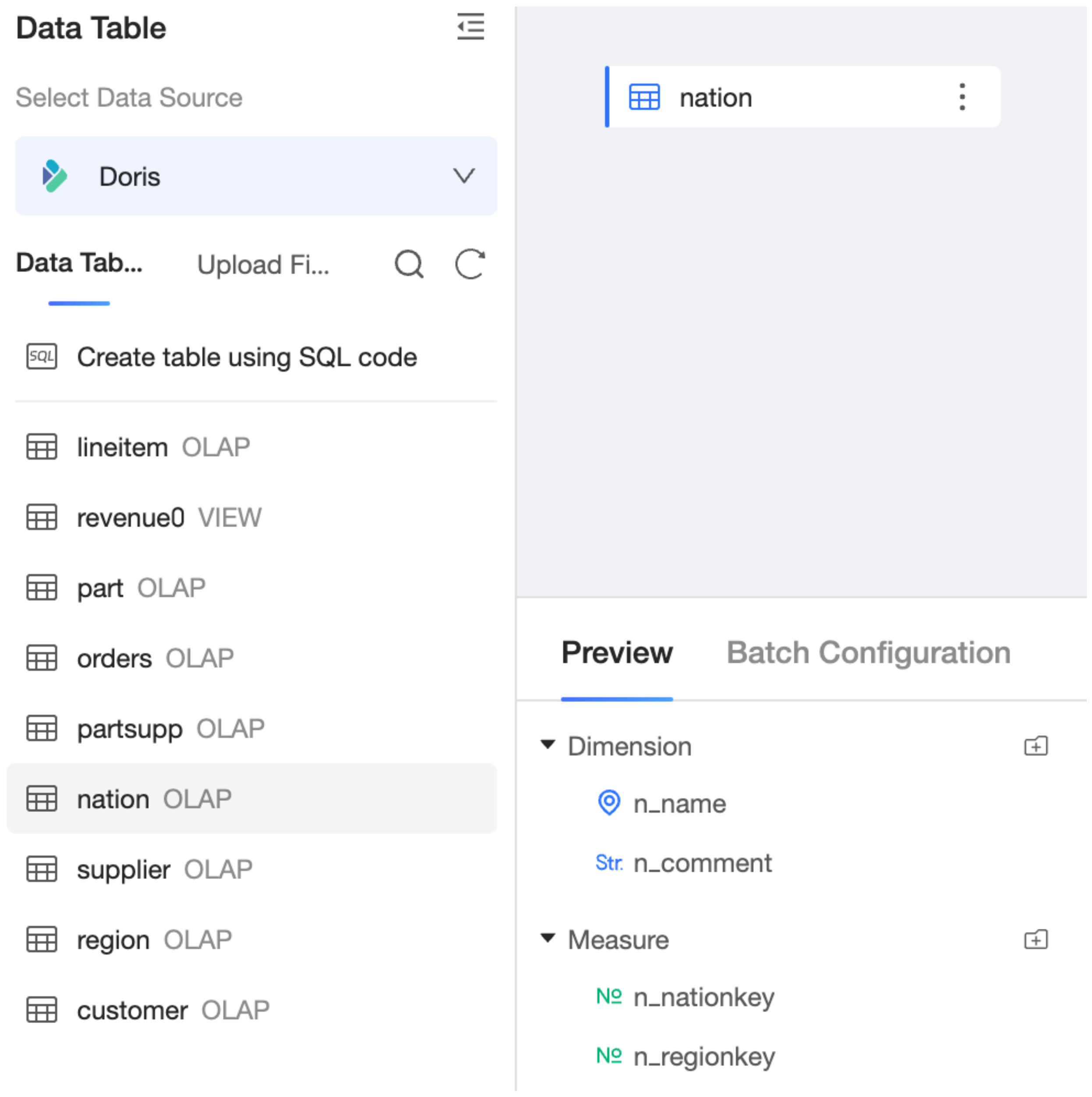


Figure 96: Doris table

4.15.9 Smartbi

4.15.9.1 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.

4.15.9.2 Precondition

you can visit https://www.smartbi.info/download to download and install Smartbi.

4.15.9.3 Data connection and application

1. Log into Smartbi and click Data Connections.

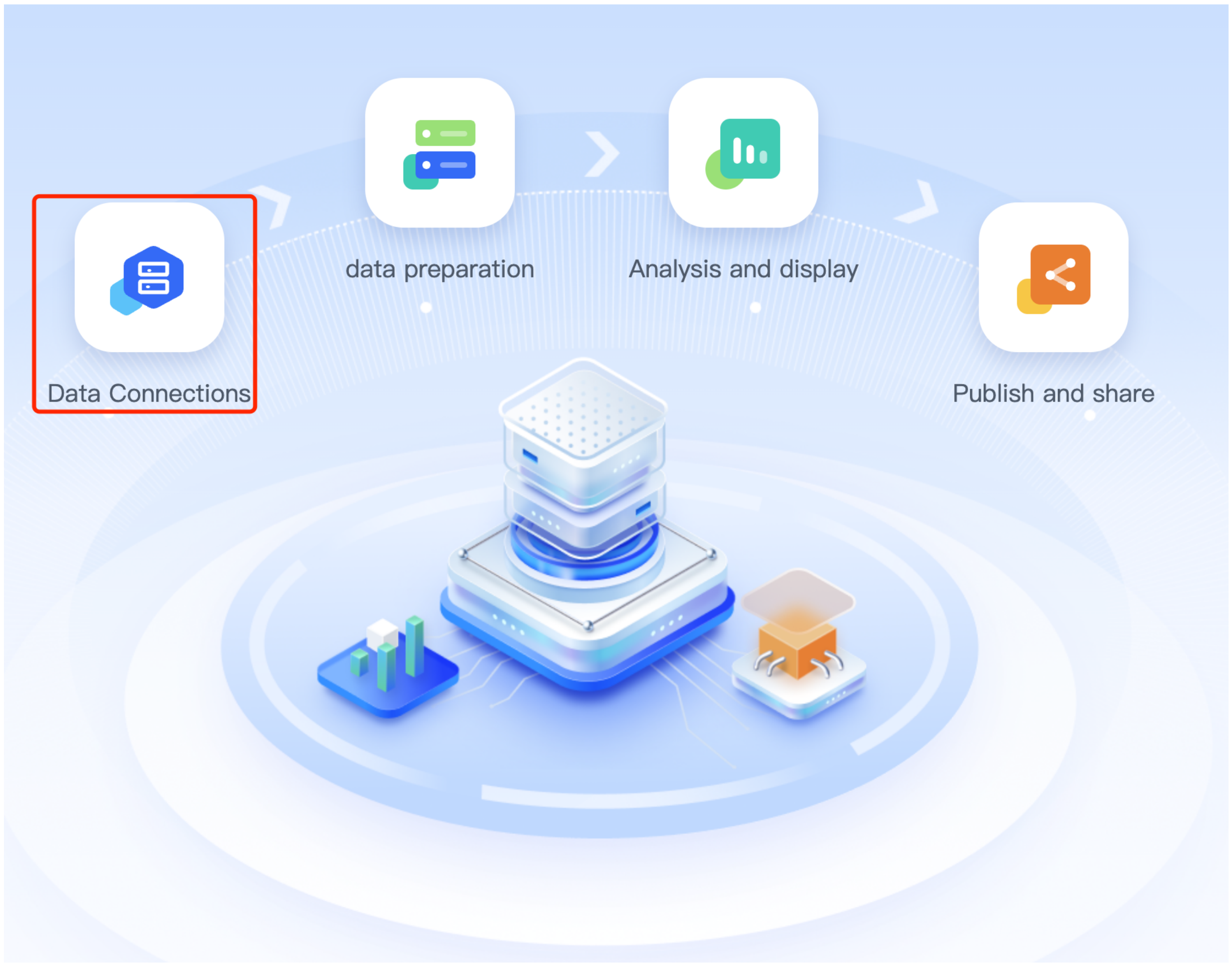
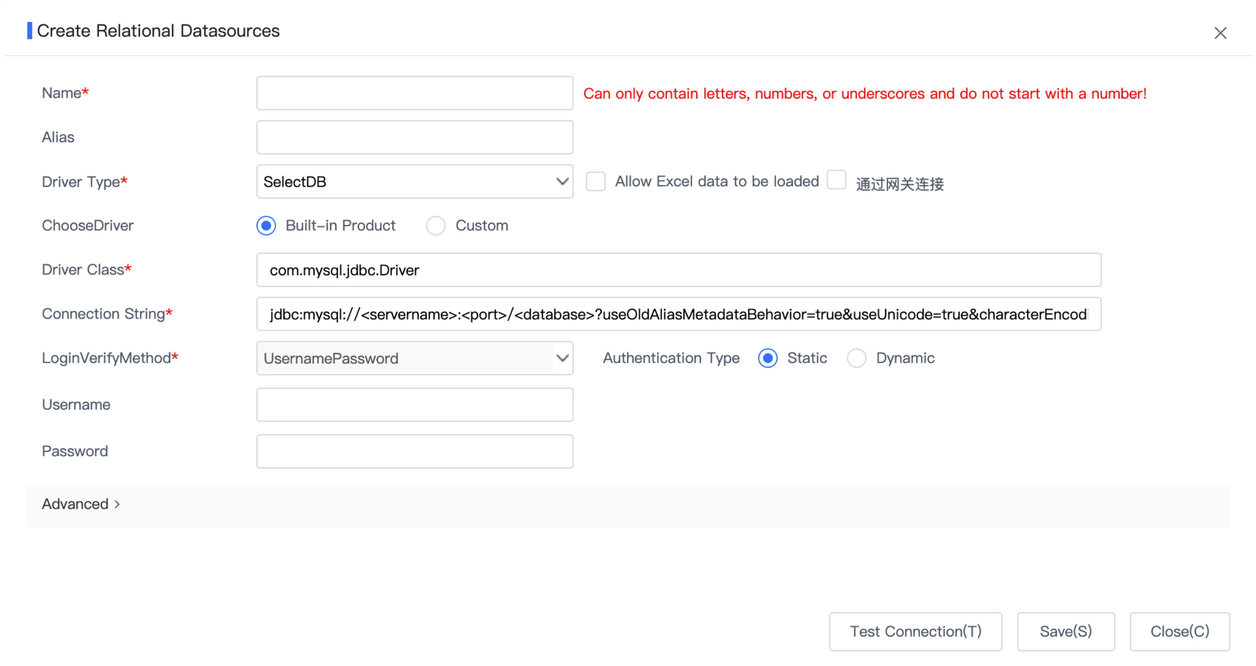


Figure 97: main page

2. After clicking the data connections, you can find a list of databases to be connected with and select the Doris database.

Figure 98: selectdb

3. After selecting the database, fill in the connection information of the Doris database

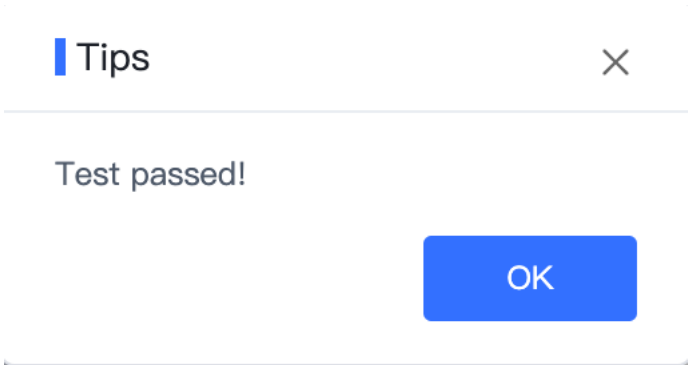


Create Relational Datasources

Name*	Can only contain letters, numbers, or underscores and do not start with a number!
Alias	
Driver Type*	SelectDB ☐ Allow Excel data to be loaded ☐ 通过网关连接
ChooseDriver	● Built–in Product ○ Custom
Driver Class*	com.mysql.jdbc.Driver
Connection String*	jdbc:mysql://<servername>:<port>/<database>?useOldAliasMetadataBehavior=true&useUnicode=true&characterEncod
LoginVerifyMethod*	UsernamePassword Authentication Type ● Static ○ Dynamic
Username	
Password	

Figure 99: data source connection

4. If the information is correct and the network is connected, a message is displayed indicating that the Test passed.


Tips

Test passed!

Figure 100: 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.

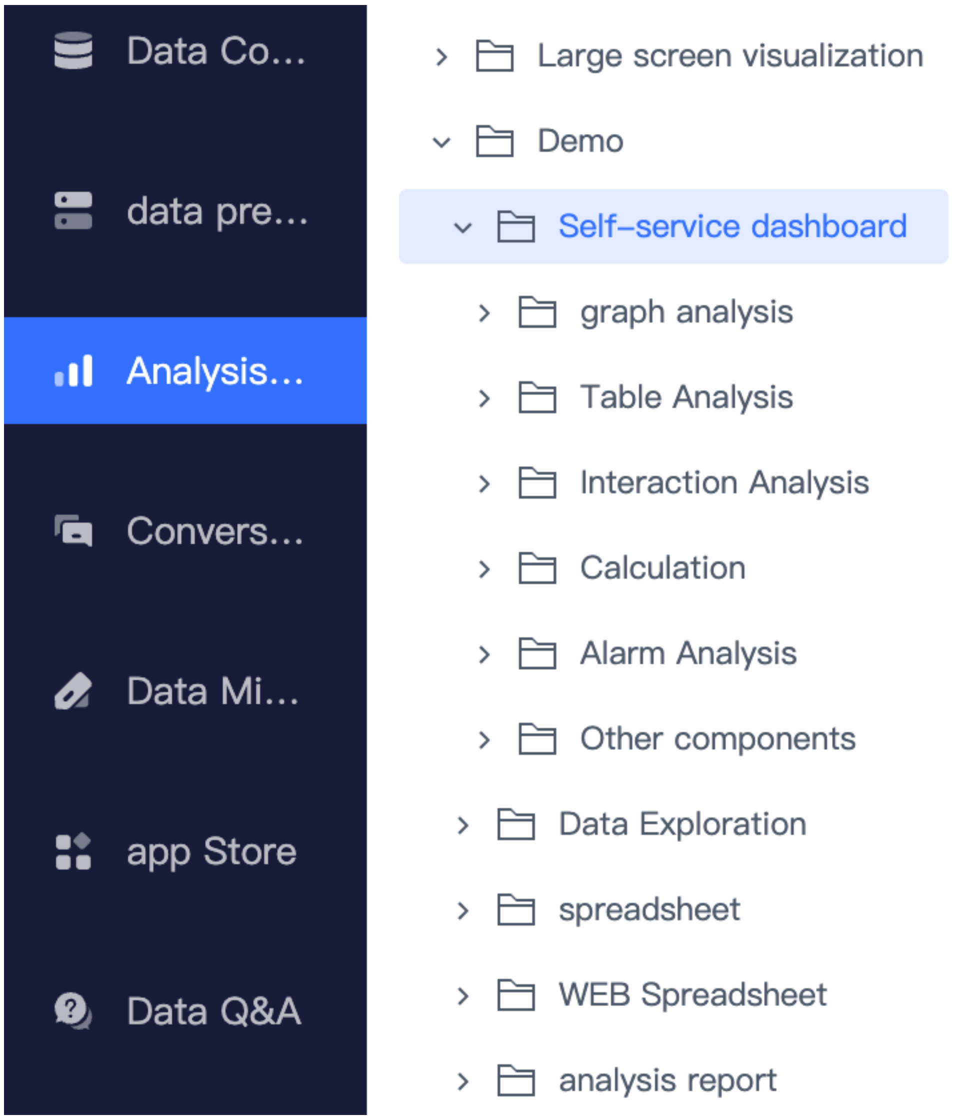


Figure 101: data analysis

5 FAQ

5.1 Install 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.

5.1.0.1 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.

5.1.0.2 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.

5.1.0.3 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.

5.1.0.4 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.

5.1.0.5 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.

5.1.0.6 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.

4. 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.

5.1.0.7 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.

5.1.0.8 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;
        }
    }
}
```

5.1.0.9 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.

5.1.0.10 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).

5.1.0.11 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.

5.1.0.12 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

5.1.0.13 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
```

5.1.0.14 ### 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

5.1.0.15 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

5.1.0.16 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.

5.1.0.17 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.

5.2 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.

5.2.0.1 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 .

5.2.0.2 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.

5.2.0.3 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.

5.2.0.4 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.

5.2.0.5 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.

5.2.0.6 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.

5.2.0.7 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.

5.2.0.8 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)

5.2.0.9 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.
```

5.2.0.10 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.

5.2.0.11 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"
);
```

5.2.0.12 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
```

5.2.0.13 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.

5.3 SQL Error

5.3.0.1 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.

5.3.0.2 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.

5.3.0.3 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.

5.3.0.4 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.

5.3.0.5 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

5.3.0.6 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

5.3.0.7 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.

5.3.0.8 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.

5.4 Data Lakehouse FAQ

5.4.1 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
```

5.4.2 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

5.4.3 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.

5.4.4 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(
        ↪ AuthorizationMetaStoreFilterHook.java:78) ~[hive-exec-3.1.3-core.jar:3.1.3]
    at org.apache.hadoop.hive.ql.security.authorization.plugin.
        ↪ AuthorizationMetaStoreFilterHook.filterDatabases(AuthorizationMetaStoreFilterHook
        ↪ .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.
```

5.4.5 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.ConfiguredFailoverProxyProvi`
   `↪`

   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.

5.4.6 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.

5.4.7 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

5.5 BI FAQ

5.5.1 Power BI

5.5.1.1 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.

5.5.1.2 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.

5.5.1.3 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.

5.5.1.4 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.

5.5.2 Tableau

5.5.2.1 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

5.5.2.2 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.

5.5.2.3 Q3. Connection error Unsupported command(COM_STMT_PREPARED)

The MySQL driver version is improperly installed. Install the MySQL 5.1.x connection driver instead.

6 Releases

6.1 v1.2

6.1.1 Release 1.2.8

Quick Download: https://doris.apache.org/download/

GitHub: https://github.com/apache/doris/releases

6.1.1.1 Bug Fixes

- Fixed several issues with query execution.
- Fixed several issues with Spark Load.
- Fixed several issues with Parquet Reader.
- Fixed several issues with Orc Reader.
- Fixed Broker “FileSystem closed” problem.
- Fixed several issues with Broker Load.
- Fixed several issues with CTAS execution.

- Fixed several issues with backup and restore.
- Added “Catalog” column in audit log.
- Optimized the metadata cache of Iceberg Catalog.
- Fixed several issues with outfile/export feature.
- Fixed an issue with “replayEraseTable” edit log causing FE start to fail.
- Fixed some security issues.

6.1.2 Release 1.2.7

- Fixed some query issues.
- Fix some storage issues.
- Fix some decimal precision issues.
- Fix query error caused by invalid `sql_select_limit` session variable’s value.
- Fix the problem that hdfs short-circuit read cannot be used.
- Fix the problem that Tencent Cloud cosn cannot be accessed.
- Fix several issues with hive catalog kerberos access.
- Fix the problem that stream load profile cannot be used.
- Fix promethus monitoring parameter format problem.
- Fix the table creation timeout issue when creating a large number of tablets.

6.1.3 New Features

- Unique Key model supports array type as value column
- Added `have_query_cache` variable for compatibility with MySQL ecosystem.
- Added `enable_strong_consistency_read` to support strong consistent read between sessions
- FE metrics supports user-level query counter

6.1.4 Release 1.2.6

- Add a BE configuration item `allow_invalid_decimalv2_literal` to control whether can import data that exceeding the decimal’s precision, for compatibility with previous logic.

6.1.5 Query

- Fix several query planning issues.
- Support `sql_select_limit` session variable.
- Optimize query cold run performance.
- Fix expr context memory leak.
- Fix the issue that the `explode_split` function was executed incorrectly in some cases.

6.1.5.1 Multi Catalog

- Fix the issue that synchronizing hive metadata caused FE replay edit log to fail.
- Fix `refresh catalog` operation causing FE OOM.

- Fix the issue that jdbc catalog cannot handle `0000-00-00` correctly.
- Fixed the issue that the kerberos ticket cannot be refreshed automatically.
- Optimize the partition pruning performance of hive.
- Fix the inconsistent behavior of trino and presto in jdbc catalog.
- Fix the issue that hdfs short-circuit read could not be used to improve query efficiency in some environments.
- Fix the issue that the iceberg table on CHDFS could not be read.

6.1.6 Storage

- Fix the wrong calculation of delete bitmap in MOW table.
- Fix several BE memory issues.
- Fix snappy compression issue.
- Fix the issue that jemalloc may cause BE to crash in some cases.

6.1.7 Others

- Fix several java udf related issues.
- Fix the issue that the `recover table` operation incorrectly triggered the creation of dynamic partitions.
- Fix timezone when importing orc files via broker load.
- Fix the issue that the newly added PERCENT keyword caused the replay metadata of the routine load job to fail.
- Fix the issue that the `truncate` operation failed to acts on a non-partitioned table.
- Fix the issue that the mysql connection was lost due to the `show snapshot` operation.
- Optimize the lock logic to reduce the probability of lock timeout errors when creating tables.
- Add session variable `have_query_cache` to be compatible with some old mysql clients.
- Optimize the error message when encountering an error of loading.

6.1.8 Big Thanks

Thanks all who contribute to this release:

@amorynan

@BiteTheDDDDt

@caoliang-web

@dataroaring

@Doris-Extras

@dutyu

@Gabriel39

@HHoflittlefish777

@htyoung

@jacktengg

@jeffreys-cat

@kaijchen

@kaka11chen

@Kikyou1997

@KnightLiJunLong

@liaoxin01

@LiBinfeng-01

@morningman

@mrhhsg

@sohardforaname

@starocean999

@vinlee19

@wangbo

@wsjz

@xiaokang

@xinyiZzz

@yiguolei

@yujun777

@Yulei-Yang

@zhangstar333

@zy-kkk

6.1.9 Release 1.2.5

In version 1.2.5, the Doris team has fixed nearly 210 issues or performance improvements since the release of version 1.2.4. At the same time, version 1.2.5 is also an iterative version of version 1.2.4, which has higher stability. It is recommended that all users upgrade to this version.

- The `start_be.sh` script will check that the maximum number of file handles in the system must be greater than or equal to 65536, otherwise the startup will fail.
- The BE configuration item `enable_quick_compaction` is set to true by default. The Quick Compaction is enabled by default. This feature is used to optimize the problem of small files in the case of large batch import.
- After modifying the dynamic partition attribute of the table, it will no longer take effect immediately, but wait for the next task scheduling of the dynamic partition table to avoid some deadlock problems.

6.1.10 Improvement

- Optimize the use of bthread and pthread to reduce the RPC blocking problem during the query process.
- A button to download Profile is added to the Profile page of the FE web UI.
- Added FE configuration `recover_with_skip_missing_version`, which is used to query to skip the problematic replica under certain failure conditions.
- The row-level permission function supports external Catalog.
- Hive Catalog supports automatic refreshing of kerberos tickets on the BE side without manual refreshing.
- JDBC Catalog supports tables under the MySQL/ClickHouse system database (`information_schema`).

6.1.11 Bug Fixes

- Fixed the problem of incorrect query results caused by low-cardinality column optimization
- Fixed several authentication and compatibility issues accessing HDFS.
- Fixed several issues with float/double and decimal types.
- Fixed several issues with date/datetimev2 types.
- Fixed several query execution and planning issues.
- Fixed several issues with JDBC Catalog.
- Fixed several query-related issues with Hive Catalog, and Hive Metastore metadata synchronization issues.
- Fix the problem that the result of `SHOW LOAD PROFILE` statement is incorrect.
- Fixed several memory related issues.
- Fixed several issues with `CREATE TABLE AS SELECT` functionality.
- Fix the problem that the jsonb type causes BE to crash on CPU that do not support avx2.
- Fixed several issues with dynamic partitions.
- Fixed several issues with TOPN query optimization.
- Fixed several issues with the Unique Key Merge-on-Write table model.

6.1.12 Big Thanks

58 contributors participated in the improvement and release of 1.2.5, and thank them for their hard work and dedication:

@adonis0147

@airborne12

@AshinGau

@BePPPower

@BiteTheDDDDt

@caiconghui

@CalvinKirs

@cambyzju

@caoliang-web

@dataroaring

@Doris-Extras

@dujl

@dutyu

@fsilent

@Gabriel39

@gitccl

@gnehil

@GoGoWen

@gongzexin

@HappenLee

@herry2038

@jacktengg

@Jibing-Li

@kaka11chen

@Kikyou1997

@LemonLiTree

@liaoxin01

@LiBinfeng-01

@luwei16

@Moonm3n

@morningman

@mrhhsg

@Mryange

@nextdreamblue

@nsnhuang

@qidaye

@Shoothzj

@sohardforaname

@stalary

@starocean999

@SWJTU-ZhangLei

@wsjz

@xiaokang

@xinyiZzz

@yangzhg

@yiguolei

@yixiutt

@yujun777

@Yulei-Yang

@yuxuan-luo

@zclllyybb

@zddr

@zenoyang

@zhangstar333

@zhannngchen

@zxealous

@zy-kkk

@zzzzzzzs

6.1.13 Release 1.2.4

- For `DateV2`/`DatetimeV2` and `DecimalV3` type, in the results of `DESCRIBLE` and `SHOW CREATE TABLE` statements, they will no longer be displayed as `DateV2`/`DatetimeV2` or `DecimalV3`, but directly displayed as `Date`/`Datetime` or `Decimal`.
  - This change is for compatibility with some BI tools. If you want to see the actual type of the column, you can check it with the `DESCRIBE ALL` statement.
- When querying tables in the `information_schema` database, the meta information(database, table, column, etc.) in the external catalog is no longer returned by default.
  - This change avoids the problem that the `information_schema` database cannot be queried due to the connection problem of some external catalog, so as to solve the problem of using some BI tools with Doris. It can be controlled by the FE configuration `infodb_support_ext_catalog`, and the default value is `false`, that is, the meta information of external catalog will not be returned.

6.1.14 Improvement

6.1.14.0.1 JDBC Catalog

- Supports connecting to Trino/Presto via JDBC Catalog

Refer to: https://doris.apache.org/docs/dev/lakehouse/multi-catalog/jdbc#trino

- JDBC Catalog connects to Clickhouse data source and supports Array type mapping

Refer to: https://doris.apache.org/docs/dev/lakehouse/multi-catalog/jdbc#clickhouse

6.1.14.0.2 Spark Load

- Spark Load supports Resource Manager HA related configuration

Refer to: https://github.com/apache/doris/pull/15000

6.1.14.1 Bug Fixes

- Fixed several connectivity issues with Hive Catalog.
- Fixed ClassNotFound issues with Hudi Catalog.
- Optimize the connection pool of JDBC Catalog to avoid too many connections.
- Fix the problem that OOM will occur when importing data from another Doris cluster through JDBC Catalog.
- Fixed serveral queries and imports planning issues.
- Fixed several issues with Unique Key Merge-On-Write data model.
- Fix several BDBJE issues and solve the problem of abnormal FE metadata in some cases.
- Fix the problem that the `CREATE VIEW` statement does not support Table Valued Function.
- Fixed several memory statistics issues.
- Fixed several issues reading Parquet/ORC format.
- Fixed several issues with DecimalV3.
- Fixed several issues with SHOW QUERY/LOAD PROFILE.

6.1.15 Release 1.2.3

6.1.15.0.1 JDBC Catalog

- Support connecting to Doris clusters through JDBC Catalog.

Currently, Jdbc Catalog only support to use 5.x version of JDBC jar package to connect another Doris database. If you use 8.x version of JDBC jar package, the data type of column may not be matched.

Reference: https://doris.apache.org/docs/dev/lakehouse/multi-catalog/jdbc/#doris

- Support to synchronize only the specified database through the `only_specified_database` attribute.
- Support synchronizing table names in the form of lowercase through `lower_case_table_names` to solve the problem of case sensitivity of table names.

Reference: https://doris.apache.org/docs/dev/lakehouse/multi-catalog/jdbc

- Optimize the read performance of JDBC Catalog.

6.1.15.0.2 Elasticsearch Catalog

- Support Array type mapping.
- Support whether to push down the like expression through the `like_push_down` attribute to control the CPU overhead of the ES cluster.

Reference: https://doris.apache.org/docs/dev/lakehouse/multi-catalog/es

6.1.15.0.3 Hive Catalog

- Support Hive table default partition `_HIVE_DEFAULT_PARTITION_`.
- Hive Metastore metadata automatic synchronization supports notification event in compressed format.

6.1.15.0.4 Dynamic Partition Improvement

- Dynamic partition supports specifying the `storage_medium` parameter to control the storage medium of the newly added partition.

Reference: https://doris.apache.org/docs/dev/advanced/partition/dynamic-partition

6.1.15.0.5 Optimize BE’s Threading Model

- Optimize BE’s threading model to avoid stability problems caused by frequent thread creation and destroy.

6.1.16 Bugfix

- Fixed issues with Merge-On-Write Unique Key tables.
- Fixed compaction related issues.
- Fixed some delete statement issues causing data errors.
- Fixed several query execution errors.
- Fixed the problem of using JDBC catalog to cause BE crash on some operating system.
- Fixed Multi-Catalog issues.
- Fixed memory statistics and optimization issues.
- Fixed decimalV3 and date/datetimev2 related issues.
- Fixed load transaction stability issues.
- Fixed light-weight schema change issues.
- Fixed the issue of using `datetime` type for batch partition creation.
- Fixed the problem that a large number of failed broker loads would cause the FE memory usage to be too high.
- Fixed the problem that stream load cannot be canceled after dropping the table.
- Fixed querying `information_schema` timeout in some cases.
- Fixed the problem of BE crash caused by concurrent data export using `select outfile`.
- Fixed transactional insert operation memory leak.
- Fixed several query/load profile issues, and supports direct download of profiles through FE web ui.
- Fixed the problem that the BE tablet GC thread caused the IO util to be too high.
- Fixed the problem that the commit offset is inaccurate in Kafka routine load.

6.1.17 Release 1.2.2

6.1.17.0.1 Lakehouse

- Support automatic synchronization of Hive metastore.
- Support reading the Iceberg Snapshot, and viewing the Snapshot history.
- JDBC Catalog supports PostgreSQL, Clickhouse, Oracle, SQLServer
- JDBC Catalog supports Insert operation

Reference: https://doris.apache.org/docs/dev/lakehouse/multi-catalog/

6.1.17.0.2 Auto Bucket

Set and scale the number of buckets for different partitions to keep the number of tablet in a relatively appropriate range.

6.1.17.0.3 New Functions

Add the new function `width_bucket`.

Reference: https://doris.apache.org/zh-CN/docs/dev/sql-manual/sql-functions/width-bucket/#description

6.1.18 Behavior Changes

- Disable BE’s page cache by default: `disable_storage_page_cache=true`

Turn off this configuration to optimize memory usage and reduce the risk of memory OOM. But it will reduce the query latency of some small queries. If you are sensitive to query latency, or have high concurrency and small query scenarios, you can configure disable_storage_page_cache=false to enable page cache again.

- Add new session variable `group_by_and_having_use_alias_first`, used to control whether the group and having clauses use alias.

Reference: https://doris.apache.org/docs/dev/advanced/variables

6.1.19 Improvement

6.1.19.0.1 Compaction

- Support `Vertical Compaction`. To optimize the compaction overhead and efficiency of wide tables.
- Support `Segment ompaction`. Fix -238 and -235 issues with high frequency imports.

6.1.19.0.2 Lakehouse

- Hive Catalog can be compatible with Hive version 1/2/3
- Hive Catalog can access JuiceFS based HDFS with Broker.
- Iceberg Catalog Support Hive Metastore and Rest Catalog type.
- ES Catalog support _id column mapping.
- Optimize Iceberg V2 read performance with large number of delete rows.
- Support for reading Iceberg tables after Schema Evolution
- Parquet Reader handles column name case correctly.

6.1.19.0.3 Other

- Support for accessing Hadoop KMS-encrypted HDFS.
- Support to cancel the Export export task in progress.
- Optimize the performance of `explode_split` with 1x.
- Optimize the read performance of nullable columns with 3x.
- Optimize some problems of Memtracker, improve memory management accuracy, and optimize memory application.

6.1.20 Bug Fix

- Fixed memory leak when loading data with Doris Flink Connector.
- Fixed the possible thread scheduling problem of BE and reduce the `Fragment sent timeout` error caused by BE thread exhaustion.
- Fixed various correctness and precision issues of column type datetimev2/decimalv3.
- Fixed the problem data correctness issue with Unique Key Merge-on-Read table.
- Fixed various known issues with the Light Schema Change feature.
- Fixed various data correctness issues of bitmap type Runtime Filter.
- Fixed the problem of poor reading performance of csv reader introduced in version 1.2.1.
- Fixed the problem of BE OOM caused by Spark Load data download phase.
- Fixed possible metadata compatibility issues when upgrading from version 1.1 to version 1.2.
- Fixed the metadata problem when creating JDBC Catalog with Resource.
- Fixed the problem of high CPU usage caused by load operation.
- Fixed the problem of FE OOM caused by a large number of failed Broker Load jobs.
- Fixed the problem of precision loss when loading floating-point types.
- Fixed the problem of memory leak when useing 2PC stream load

6.1.21 Other

Add metrics to view the total rowset and segment numbers on BE

- doris_be_all_rowsets_num and doris_be_all_segments_num

6.1.22 Big Thanks

Thanks to ALL who contributed to this release!

@adonis0147

@AshinGau

@BePPPower

@BiteTheDDDDt

@ByteYue

@caiconghui

@cambyzju

@chenlinzhong

@DarvenDuan

@dataroaring

@Doris-Extras

@dutyu

@englefly

@freemandealer

@Gabriel39

@HappenLee

@Henry2SS

@htyoung

@isHuangXin

@JackDrogon

@jacktengg

@Jibing-Li

@kaka11chen

@Kikyou1997

@Lchangliang

@LemonLiTree

@liaoxin01

@liqing-coder

@luozenglin

@morningman

@morrySnow

@mrhhsg

@nextdreamblue

@qidaye

@qzsee

@spaces-X

@stalary

@starocean999

@weizuo93

@wsjz

@xiaokang

@xinyiZzz

@xy720

@yangzhg

@yiguolei

@yixiutt

@Yukang-Lian

@Yulei-Yang

@zclllyybb

@zddr

@zhangstar333

@zhannngchen

@zy-kkk

6.1.23 Release 1.2.1

6.1.23.0.1 Supports new type DecimalV3

DecimalV3, which supports higher precision and better performance, has the following advantages over past versions.

- Larger representable range, the range of values are significantly expanded, and the valid number range [1,38].
- Higher performance, adaptive adjustment of the storage space occupied according to different precision.
- More complete precision derivation support, for different expressions, different precision derivation rules are applied to the accuracy of the result.

DecimalV3

6.1.23.0.2 Support Iceberg V2

Support Iceberg V2 (only Position Delete is supported, Equality Delete will be supported in subsequent versions).

Tables in Iceberg V2 format can be accessed through the Multi-Catalog feature.

6.1.23.0.3 Support OR condition to IN

Support converting OR condition to IN condition, which can improve the execution efficiency in some scenarios.#15437 #12872

6.1.23.0.4 Optimize the import and query performance of JSONB type

Optimize the import and query performance of JSONB type. #15219 #15219

6.1.23.0.5 Stream load supports quoted csv data

Search trim_double_quotes in Document:https://doris.apache.org/docs/dev/sql-manual/sql-reference/Data-Manipulation-Statements/Load/STREAM-LOAD

6.1.23.0.6 Broker supports Tencent Cloud CHDFS and Baidu Cloud BOS, AFS

Data on CHDFS, BOS, and AFS can be accessed through Broker. #15297 #15448

6.1.23.0.7 New function

Add function `substring_index`. #15373

6.1.24 Bug Fix

- In some cases, after upgrading from version 1.1 to version 1.2, the user permission information will be lost. #15144
- Fix the problem that the partition value is wrong when using datev2/datetimev2 type for partitioning. #15094
- Bug fixes for a large number of released features. For a complete list see: PR List

6.1.25 Upgrade Notice

6.1.25.0.1 Known Issues

- Do not use JDK11 as the runtime JDK of BE, it will cause BE Crash.
- The reading performance of the csv format in this version has declined, which will affect the import and reading efficiency of the csv format. We will fix it as soon as possible in the next three-digit version

6.1.25.0.2 Behavior Changed

- The default value of the BE configuration item `high_priority_flush_thread_num_per_store` is changed from 1 to 6, to improve the write efficiency of Routine Load. (https://github.com/apache/doris/pull/14775)
- The default value of the FE configuration item `enable_new_load_scan_node` is changed to true. Import tasks will be performed using the new File Scan Node. No impact on users.#14808
- Delete the FE configuration item `enable_multi_catalog`. The Multi-Catalog function is enabled by default.
- The vectorized execution engine is forced to be enabled by default.#15213

The session variable enable_vectorized_engine will no longer take effect. Enabled by default.

To make it valid again, set the FE configuration item `disable_enable_vectorized_engine` to false, and restart FE to make `enable_vectorized_engine` valid again.

6.1.26 Big Thanks

Thanks to ALL who contributed to this release!

@adonis0147

@AshinGau

@BePPPower

@BiteTheDDDDt

@ByteYue

@caiconghui

@cambyzju

@chenlinzhong

@dataroaring

@Doris-Extras

@dutyu

@eldenmoon

@englefly

@freemandealer

@Gabriel39

@HappenLee

@Henry2SS

@hf200012

@jacktengg

@Jibing-Li

@Kikyou1997

@liaoxin01

@luozenglin

@morningman

@morrySnow

@mrhhsg

@nextdreamblue

@qidaye

@spaces-X

@starocean999

@wangshuo128

@weizuo93

@wsjz

@xiaokang

@xinyiZzz

@xutaoustc

@yangzhg

@yiguolei

@yixiutt

@Yulei-Yang

@yuxuan-luo

@zenoyang

@zhangstar333

@zhannngchen

@zhengshengjun

6.1.27 Release 1.2.0

6.1.27.1 Highlight

1. Full Vectorizied-Engine support, greatly improved performance

   In the standard ssb-100-flat benchmark, the performance of 1.2 is 2 times faster than that of 1.1; in complex TPCH 100 benchmark, the performance of 1.2 is 3 times faster than that of 1.1.

2. Merge-on-Write Unique Key

   Support Merge-On-Write on Unique Key Model. This mode marks the data that needs to be deleted or updated when the data is written, thereby avoiding the overhead of Merge-On-Read when querying, and greatly improving the reading efficiency on the updateable data model.

3. Multi Catalog

   The multi-catalog feature provides Doris with the ability to quickly access external data sources for access. Users can connect to external data sources through the `CREATE CATALOG` command. Doris will automatically map the library and table information of external data sources. After that, users can access the data in these external data sources just like accessing ordinary tables. It avoids the complicated operation that the user needs to manually establish external mapping for each table.

   Currently this feature supports the following data sources:

   1. Hive Metastore: You can access data tables including Hive, Iceberg, and Hudi. It can also be connected to data sources compatible with Hive Metastore, such as Alibaba Cloud' s DataLake Formation. Supports data access on both HDFS and object storage.
   2. Elasticsearch: Access ES data sources.
   3. JDBC: Access MySQL through the JDBC protocol.

   Documentation: https://doris.apache.org//docs/dev/lakehouse/multi-catalog)

> Note: The corresponding permission level will also be changed automatically, see the "Upgrade Notes" section for details.

4. Light table structure changes

In the new version, it is no longer necessary to change the data file synchronously for the operation of adding and subtracting columns to the data table, and only need to update the metadata in FE, thus realizing the millisecond-level Schema Change operation. Through this function, the DDL synchronization capability of upstream CDC data can be realized. For example, users can use Flink CDC to realize DML and DDL synchronization from upstream database to Doris.

Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Definition-Statements/Create/CREATE-TABLE

When creating a table, set `"light_schema_change"="true"` in properties.

5. JDBC facade

   Users can connect to external data sources through JDBC. Currently supported:

   - MySQL
   - PostgreSQL
   - Oracle
   - SQL Server
   - Clickhouse

   Documentation: https://doris.apache.org/en/docs/dev/lakehouse/multi-catalog/jdbc

Note: The ODBC feature will be removed in a later version, please try to switch to the JDBC.

6. JAVA UDF

   Supports writing UDF/UDAF in Java, which is convenient for users to use custom functions in the Java ecosystem. At the same time, through technologies such as off-heap memory and Zero Copy, the efficiency of cross-language data access has been greatly improved.

   Document: https://doris.apache.org//docs/dev/ecosystem/udf/java-user-defined-function

   Example: https://github.com/apache/doris/tree/master/samples/doris-demo

7. Remote UDF

   Supports accessing remote user-defined function services through RPC, thus completely eliminating language restrictions for users to write UDFs. Users can use any programming language to implement custom functions to complete complex data analysis work.

   Documentation: https://doris.apache.org//docs/ecosystem/udf/remote-user-defined-function

   Example: https://github.com/apache/doris/tree/master/samples/doris-demo

8. More data types support

   - Array type

     Array types are supported. It also supports nested array types. In some scenarios such as user portraits and tags, the Array type can be used to better adapt to business scenarios. At the same time, in the new version, we have also implemented a large number of data-related functions to better support the application of data types in actual scenarios.

   Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Types/ARRAY

   Related functions: https://doris.apache.org//docs/dev/sql-manual/sql-functions/array-functions/array_max

   - Jsonb type

     Support binary Json data type: Jsonb. This type provides a more compact json encoding format, and at the same time provides data access in the encoding format. Compared with json data stored in strings, it is several times newer and can be improved.

   Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Types/JSONB

   Related functions: https://doris.apache.org//docs/dev/sql-manual/sql-functions/json-functions/jsonb_parse

   - Date V2

     Sphere of influence:

     1. The user needs to specify datev2 and datetimev2 when creating the table, and the date and datetime of the original table will not be affected.
     2. When datev2 and datetimev2 are calculated with the original date and datetime (for example, equivalent connection), the original type will be cast into a new type for calculation
     3. The example is in the documentation

     Documentation: https://doris.apache.org/docs/1.2/sql-manual/sql-reference/Data-Types/DATEV2

6.1.27.2 More

1. A new memory management framework

   Documentation: https://doris.apache.org//docs/dev/admin-manual/maint-monitor/memory-management/memory-tracker

2. Table Valued Function

   Doris implements a set of Table Valued Function (TVF). TVF can be regarded as an ordinary table, which can appear in all places where "table" can appear in SQL.

   For example, we can use S3 TVF to implement data import on object storage:

```
    insert into tbl select * from s3("s3://bucket/file.*", "ak" = "xx", "sk" = "xxx") where c1 >
        ↪ 2;
```

```
Or directly query data files on HDFS:
```

```
    insert into tbl select * from hdfs("hdfs://bucket/file.*") where c1 > 2;
```

```
TVF can help users make full use of the rich expressiveness of SQL and flexibly process various
    ↪ data.

Documentation:

https://doris.apache.org//docs/dev/sql-manual/sql-functions/table-functions/s3

https://doris.apache.org//docs/dev/sql-manual/sql-functions/table-functions/hdfs
```

3. A more convenient way to create partitions

   Support for creating multiple partitions within a time range via the `FROM TO` command.

4. Column renaming

   For tables with Light Schema Change enabled, column renaming is supported.

   Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Definition-Statements/Alter/ALTER-TABLE-RENAME

5. Richer permission management

   - Support row-level permissions
     Row-level permissions can be created with the `CREATE ROW POLICY` command.
     Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Definition-Statements/Create/CREATE-POLICY
   - Support specifying password strength, expiration time, etc.
   - Support for locking accounts after multiple failed logins.
     Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Account-Management-Statements/ALTER-USER

6. Import

    - CSV import supports csv files with header.

      Search for `csv_with_names` in the documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Manipulation-Statements/Load/STREAM-LOAD/
    - Stream Load adds `hidden_columns`, which can explicitly specify the delete flag column and sequence column.

      Search for `hidden_columns` in the documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Manipulation-Statements/Load/STREAM-LOAD
    - Spark Load supports Parquet and ORC file import.
    - Support for cleaning completed imported Labels

    Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Manipulation-Statements/Load/CLEAN-LABEL

    - Support batch cancellation of import jobs by status

      Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Manipulation-Statements/Load/CANCEL-LOAD
    - Added support for Alibaba Cloud oss, Tencent Cloud cos/chdfs and Huawei Cloud obs in broker load.

      Documentation: https://doris.apache.org//docs/dev/advanced/broker
    - Support access to hdfs through hive-site.xml file configuration.

      Documentation: https://doris.apache.org//docs/dev/admin-manual/config/config-dir

7. Support viewing the contents of the catalog recycle bin through `SHOW CATALOG RECYCLE BIN` function.

    Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Show-Statements/SHOW-CATALOG-RECYCLE-BIN

8. Support `SELECT * EXCEPT` syntax.

    Documentation: https://doris.apache.org//docs/dev/data-table/basic-usage

9. OUTFILE supports ORC format export. And supports multi-byte delimiters.

    Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Manipulation-Statements/OUTFILE

10. Support to modify the number of Query Profiles that can be saved through configuration.

    Document search FE configuration item: max_query_profile_num

11. The DELETE statement supports IN predicate conditions. And it supports partition pruning.

    Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Manipulation-Statements/Manipulation/DELETE

12. The default value of the time column supports using `CURRENT_TIMESTAMP`

    Search for “CURRENT_TIMESTAMP” in the documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Definition-Statements/Create/CREATE-TABLE

13. Add two system tables: backends, rowsets

    Documentation:

    https://doris.apache.org//docs/dev/admin-manual/system-table/backends

    https://doris.apache.org//docs/dev/admin-manual/system-table/rowsets

14. Backup and restore

    - The Restore job supports the `reserve_replica` parameter, so that the number of replicas of the restored table is the same as that of the backup.
    - The Restore job supports `reserve_dynamic_partition_enable` parameter, so that the restored table keeps the dynamic partition enabled.

    Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Definition-Statements/Backup-and-Restore/RESTORE

    - Support backup and restore operations through the built-in libhdfs, no longer rely on broker.

    Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Definition-Statements/Backup-and-Restore/CREATE-REPOSITORY

15. Support data balance between multiple disks on the same machine

    Documentation:

    https://doris.apache.org//docs/dev/sql-manual/sql-reference/Database-Administration-Statements/ADMIN-REBALANCE-DISK

    https://doris.apache.org//docs/dev/sql-manual/sql-reference/Database-Administration-Statements/ADMIN-CANCEL-REBALANCE-DISK

16. Routine Load supports subscribing to Kerberos-authenticated Kafka services.

    Search for kerberos in the documentation: https://doris.apache.org//docs/dev/data-operate/import/import-way/routine-load-manual

17. New built-in-function

    Added the following built-in functions:

    - `cbrt`
    - `sequence_match/sequence_count`
    - `mask/mask_first_n/mask_last_n`
    - `elt`
    - `any/any_value`
    - `group_bitmap_xor`
    - `ntile`
    - `nvl`
    - `uuid`
    - `initcap`
    - `regexp_replace_one/regexp_extract_all`
    - `multi_search_all_positions/multi_match_any`
    - `domain/domain_without_www/protocol`
    - `running_difference`
    - `bitmap_hash64`
    - `murmur_hash3_64`
    - `to_monday`
    - `not_null_or_empty`

  - `window_funnel`
  - `group_bit_and/group_bit_or/group_bit_xor`
  - `outer combine`
  - and all array functions

6.1.28 Upgrade Notice

6.1.28.1 Known Issues

- Use JDK11 will cause BE crash, please use JDK8 instead.

6.1.28.2 Behavior Changed

- Permission level changes

  Because the catalog level is introduced, the corresponding user permission level will also be changed automatically. The rules are as follows:

  - GlobalPrivs and ResourcePrivs remain unchanged
  - Added CatalogPrivs level.
  - The original DatabasePrivs level is added with the internal prefix (indicating the db in the internal catalog)
  - Add the internal prefix to the original TablePrivs level (representing tbl in the internal catalog)

- In GroupBy and Having clauses, match on column names in preference to aliases. (#14408)

- Creating columns starting with `mv_` is no longer supported. `mv_` is a reserved keyword in materialized views (#14361)

- Removed the default limit of 65535 rows added by the order by statement, and added the session variable `default_order_by_limit` to configure this limit. (#12478)

- In the table generated by "Create Table As Select", all string columns use the string type uniformly, and no longer distinguish varchar/char/string (#14382)

- In the audit log, remove the word `default_cluster` before the db and user names. (#13499) (#11408)

- Add sql digest field in audit log (#8919)

- The union clause always changes the order by logic. In the new version, the order by clause will be executed after the union is executed, unless explicitly associated by parentheses. (#9745)

- During the decommission operation, the tablet in the recycle bin will be ignored to ensure that the decomission can be completed. (#14028)

- The returned result of Decimal will be displayed according to the precision declared in the original column, or according to the precision specified in the cast function. (#13437)

- Changed column name length limit from 64 to 256 (#14671)

- Changes to FE configuration items

- The `enable_vectorized_load` parameter is enabled by default. (#11833)

- Increased `create_table_timeout` value. The default timeout for table creation operations will be increased. (#13520)
- Modify `stream_load_default_timeout_second` default value to 3 days.
- Modify the default value of `alter_table_timeout_second` to one month.
- Increase the parameter `max_replica_count_when_schema_change` to limit the number of replicas involved in the alter job, the default is 100000. (#12850)
- Add `disable_iceberg_hudi_table`. The iceberg and hudi appearances are disabled by default, and the multi catalog function is recommended. (#13932)
- Changes to BE configuration items
- Removed `disable_stream_load_2pc` parameter. 2PC's stream load can be used directly. (#13520)
- Modify `tablet_rowset_stale_sweep_time_sec` from 1800 seconds to 300 seconds.
- Redesigned configuration item name about compaction (#13495)
- Revisited parameter about memory optimization (#13781)
- Session variable changes
- Modify the variable `enable_insert_strict` to true by default. This will cause some insert operations that could be executed before, but inserted illegal values, to no longer be executed. (11866)
- Modified variable `enable_local_exchange` to default to true (#13292)
- Default data transmission via lz4 compression, controlled by variable `fragment_transmission_compression_codec` (#11955)
- Add `skip_storage_engine_merge` variable for debugging unique or agg model data (#11952)

  Documentation: https://doris.apache.org//docs/dev/advanced/variables
- The BE startup script will check whether the value is greater than 200W through `/proc/sys/vm/max_map_count`. Otherwise, the startup fails. (#11052)
- Removed mini load interface (#10520)
- FE Metadata Version

  FE Meta Version changed from 107 to 114, and cannot be rolled back after upgrading.

6.1.28.3 During Upgrade

1. Upgrade preparation

- Need to replace: lib, bin directory (start/stop scripts have been modified)
- BE also needs to configure JAVA_HOME, and already supports JDBC Table and Java UDF.
- The default JVM Xmx parameter in fe.conf is changed to 8GB.

2. Possible errors during the upgrade process

- The repeat function cannot be used and an error is reported: `vectorized repeat function cannot be executed`, you can turn off the vectorized execution engine before upgrading. (#13868)
- schema change fails with error: `desc_tbl is not set. Maybe the FE version is not equal to the BE` (#13822)
- Vectorized hash join cannot be used and an error will be reported. `vectorized hash join cannot be executed`. You can turn off the vectorized execution engine before upgrading. (#13753)

  The above errors will return to normal after a full upgrade.

6.1.28.4 Performance Impact

- By default, JeMalloc is used as the memory allocator of the new version BE, replacing TcMalloc (#13367)
- The batch size in the tablet sink is modified to be at least 8K. (#13912)
- Disable chunk allocator by default (#13285)

6.1.28.5 Api change

- BE's http api error return information changed from `{"status": "Fail", "msg": "xxx"}` to more specific `{"status ↪ ": "Not found", "msg": "Tablet not found. tablet_id=1202"}`(#9771)
- In `SHOW CREATE TABLE`, the content of comment is changed from double quotes to single quotes (#10327)
- Support ordinary users to obtain query profile through http command. (#14016) Documentation: https://doris.apache.org//docs/dev/admin-manual/http-actions/fe/manager/query-profile-action
- Optimized the way to specify the sequence column, you can directly specify the column name. (#13872) Documentation: https://doris.apache.org//docs/dev/data-operate/update-delete/sequence-column-manual
- Increase the space usage of remote storage in the results returned by `show backends` and `show tablets` (#11450)
- Removed Num-Based Compaction related code (#13409)
- Refactored BE's error code mechanism, some returned error messages will change (#8855) other
- Support Docker official image.
- Support compiling Doris on MacOS(x86/M1) and ubuntu-22.04 Documentation: https://doris.apache.org//docs/dev/install/source-install/compilation-mac/
- Support for image file verification.

Documentation: https://doris.apache.org//docs/dev/admin-manual/maint-monitor/metadata-operation/

- script related

- The stop scripts of FE and BE support exiting FE and BE via the `--grace` parameter (use kill -15 signal instead of kill -9)
- FE start script supports checking the current FE version via –version (#11563)
- Support to get the data and related table creation statement of a tablet through the `ADMIN COPY TABLET` command, for local problem debugging (#12176)

  Documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Database-Administration-Statements/ADMIN-COPY-TABLET
- Support to obtain a table creation statement related to a SQL statement through the http api for local problem reproduction (#11979)

  Documentation: https://doris.apache.org//docs/dev/admin-manual/http-actions/fe/query-schema-action
- Support to close the compaction function of this table when creating a table, for testing (#11743)

  Search for "disble_auto_compaction" in the documentation: https://doris.apache.org//docs/dev/sql-manual/sql-reference/Data-Definition-Statements/Create/CREATE-TABLE

6.1.29 Big Thanks

Thanks to ALL who contributed to this release! (alphabetically)

```
@924060929
@a19920714liou
@adonis0147
@Aiden-Dong
@aiwenmo
@AshinGau
@b19mud
@BePPPower
@BiteTheDDDDt
@bridgeDream
@ByteYue
@caiconghui
@CalvinKirs
@cambyzju
@caoliang-web
@carlvinhust2012
@catpineapple
@ccoffline
@chenlinzhong
@chovy-3012
@coderjiang
@cxzl25
@dataalive
@dataroaring
@dependabot[bot]
@dinggege1024
```

@DongLiang-0
@Doris-Extras
@eldenmoon
@EmmyMiao87
@englefly
@FreeOnePlus
@Gabriel39
@gaodayue
@geniusjoe
@gj-zhang
@gnehil
@GoGoWen
@HappenLee
@hello-stephen
@Henry2SS
@hf200012
@huyuanfeng2018
@jacktengg
@jackwener
@jeffreys-cat
@Jibing-Li
@JNSimba
@Kikyou1997
@Lchangliang
@LemonLiTree
@lexoning
@liaoxin01
@lide-reed
@link3280
@liutang123
@liuyaolin
@LOVEGISER
@lsy3993
@luozenglin
@luzhijing
@madongz
@morningman
@morningman-cmy
@morrySnow
@mrhhsg
@Myasuka
@myfjdthink
@nextdreamblue
@pan3793
@pangzhili

```
@pengxiangyu
@platoneko
@qidaye
@qzsee
@SaintBacchus
@SeekingYang
@smallhibiscus
@sohardforaname
@song7788q
@spaces-X
@ssusieee
@stalary
@starocean999
@SWJTU-ZhangLei
@TaoZex
@timelxy
@Wahno
@wangbo
@wangshuo128
@wangyf0555
@weizhengte
@weizuo93
@wsjz
@wunan1210
@xhmz
@xiaokang
@xiaokangguo
@xinyiZzz
@xy720
@yangzhg
@Yankee24
@yeyudefeng
@yiguolei
@yinzhijian
@yixiutt
@yuanyuan8983
@zbtzbtzbt
@zenoyang
@zhangboya1
@zhangstar333
@zhannngchen
@ZHbamboo
@zhengshiJ
@zhenhb
@zhqu1148980644
```

```
@zuochunwei
@zy-kkk
```

Copyright © 2024 The Apache Software Foundation,Licensed under the Apache License, Version 2.0. Apache, Doris, Apache Doris, the Apache feather logo and the Apache Doris logo are trademarks of The Apache Software Foundation.