Alibaba Cloud HybridDB for MySQL

User Guide

Issue: 20181107

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.
5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade

secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).

6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 **Danger:** Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 **Warning:** Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 **Note:** Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 **Note:** You can use **Ctrl** + **A** to select all files.
>	Multi-level menu cascade.	**Settings** > **Network** > **Set network type**
Bold	It is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *{stand | slave}*

Contents

1 Preface

Overview

HybridDB for MySQL is a distributed hybrid transaction/analytical processing (HTAP) database. This database has the following features:

- Supports both online transaction processing (OLTP) and online analytical processing (OLAP), and allows for real-time analysis and decision making.
- Supports online expansion in the distributed multi-node architecture.
- Provides PB-level storage, and supports data compression.
- Fully compatible with MySQL syntax and functions, and supports common Oracle analysis functions.
- Fully supports the TPC-H and TPC-DS testing benchmarks.

This document describes how to set HybridDB for MySQL using the *HybridDB for MySQL console*, and provides more information about the benefits and features of HybridDB for MySQL. You can also manage the HybridDB for MySQL console using APIs.

To contact Technical Support, click **Ticket System > Open Ticket** in the *HybridDB for MySQL console* or click *Here* to submit a ticket.

For more information about features and pricing of HybridDB for MySQL, see *HybridDB for MySQL details page*.

Declaration

Some product features or services described in this document may be unavailable for certain regions. See the relevant commercial contracts for specific Terms and Conditions. This document serves as a user guide. No content in this document can constitute any express or implied warranty. The content of this document is updated as per the product upgrade and many other respective factors. You must first verify the document with your latest corresponding software version.

Basic concepts

- Instance: a separate database service process that occupies physical memory.
- Database: a logical unit that is created under an instance. Only one database can be created for one HybridDB for MySQL instance.

2 Quick start

For more information about common operations in HybridBD for MySQL, see *Quick Start*.

3 Introduction to the HybridDB for MySQL console

To manage instances and the databases in these instances, log on to the *HybridDB for MySQL console*>.

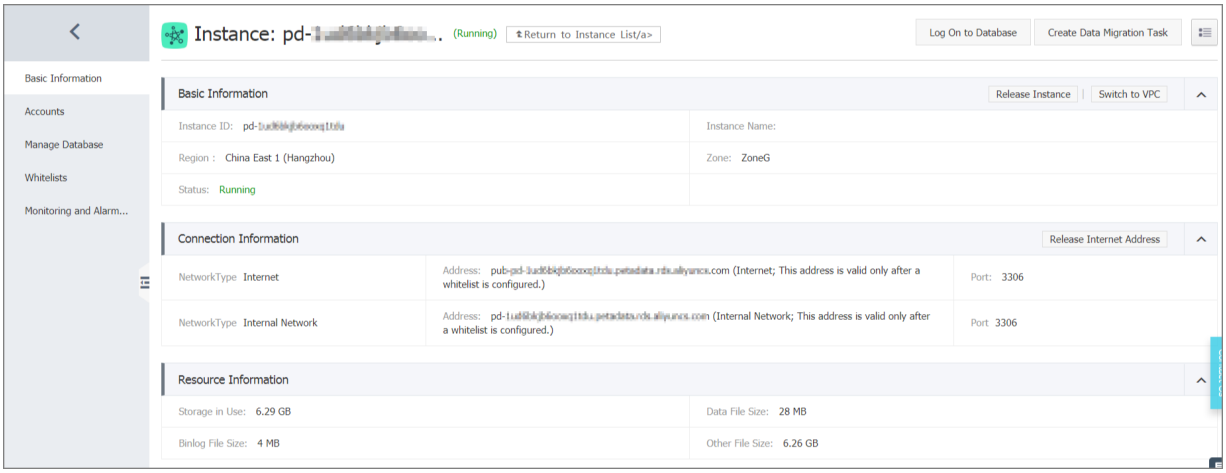
After you enter the console, you can see a list of instances. Click the instance ID or click **Manage** next to the required instance to go to its Basic Information page. To view databases, go to the left-side navigation pane and click **Databases**. Then, click the database name or click **Manage** next to the required database to go to its Basic Information page.

Figure 3-1: HybridDB for MySQL console

On the Basic Information page of an instance, you can choose among the following operations in the left-side navigation pane:

- Basic Information: provides basic information about the instance, including its connection and resource information.
- Accounts: provides account information and related operational options, including creating accounts, deleting accounts, and changing passwords.
- Databases: lists all databases in an instance.
- Monitoring and Alarms: provides monitoring and alarm information about the instance, filters monitoring results based on metrics, and sets alarm rules.

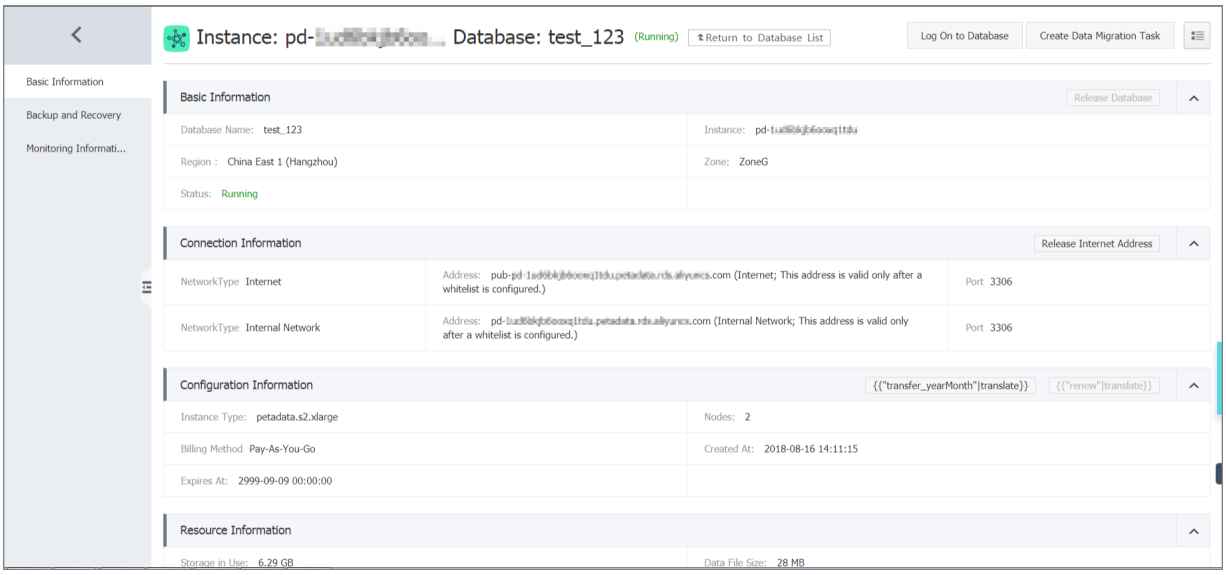
Figure 3-2: Basic information about an instance



On the Basic Information page of a database, you can choose among the following operations in the left-side navigation pane:

- Basic Information: provides basic information about the instance, including its connection and resource information.
- Backup and Recovery: specifies the backup frequency, views backup sets, and recovers data.
- Monitor: provides monitoring information about the database, and filters monitoring results based on metrics.

Figure 3-3: Basic information about a database



4 Manage accounts

1. Log on to the *HybridDB for MySQL console* to go to the **Instances** page.
2. Click **Manage** next to the required instance to go to its **Basic Information** page.
3. In the left-side navigation pane, select **Accounts**. In the Account List page, you can click **Delete**, **Reset password**, or **Create account** as needed.

5 Manage instances

5.1 Create an instance

> **Note:**
>
> - You can use HybridDB for MySQL without RDS instances.
> - To maximize performance while ensuring the stability of public networks, we recommend that you use HybridDB for MySQL together with Elastic Compute Service (ECS). If you have not purchased ECS instances, you can still access the HybridDB for MySQL instance over the public network.
> - To create an instance with more nodes or of higher specifications, *contact us*.

Procedure

1. Log on to the *HybridDB for MySQL console* and click **Create Instance**.
2. Specify **Region** and **Zone**, and set **Database Name**, **Instance Name**, **Network Type**, **Account Name**, and **Password**.

   > **Note:**
   >
   > - To access HybridDB for MySQL from an ECS instance over the Alibaba Cloud internal network, make sure that the created HybridDB for MySQL instance and the ECS instance are in the same region or zone.
   > - When you create an instance, you must also create a database for that instance.
   > - **Instance Name**: We recommend that you use an informative name for the instance. To change the name of an instance that you have created, click the pencil icon under the instance ID.
   > - **Database Name**: The database name cannot be changed once it is set.
   > - HybridDB for MySQL is billed by **node** instead of by instance. You can set the node specifications and node quantity, and create databases, based on your business requirements. For example, you can purchase two 512 GB SSD nodes to build a 1 TB HybridDB for MySQL database.
   > - **Account**: specifies the user's access data. You can set the account name and password as prompted.

3. After the settings are completed, click **Buy Now**.

Initializing a HybridDB for MySQL database can take up to 20 minutes. You can perform subsequent operations on the instance once its status in the console becomes **Running**.

5.2 Set the whitelist

To run a database securely and stably, you must add the IP addresses or CIDR blocks that are used to access the database to a whitelist. You can add up to 1,000 IP addresses.

1. Log on to the *HybridDB for MySQL console*.
2. Click **Manage** next to the required instance.
3. Click **Whitelist Settings** in the left-side navigation pane.
4. On the **Security** page, click **Create Whitelist**.
5. In the **Add Whitelist** dialog box that appears, specify the **Group Name** and **Whitelist Entries**, and then click **OK**.

Note:

You can click **Load ECS Internal IP** and add the IP addresses of ECS instances to the whitelist. IP addresses are separated with commas (,), such as 192.168.0.1,192.168.0.2. If the Whitelist field is empty or set to 0.0.0.0/0, IP addresses for accessing the database are not restricted. This may mistakenly expose the database to risks.

5.3 Switch network type

Background

HybridDB for MySQL supports two network types: classic network and Virtual Private Cloud (VPC) network. The key difference between the classic network and the VPC is as follows:

- **Classic network**: Cloud services in the classic network are not isolated from each other, and they must use a security group or whitelist policy to reject unauthorized access.
- **VPC**: A VPC is an isolated network environment on the Alibaba Cloud platform. You can customize the route table, IP address range, and gateway for the VPC. You can also combine your own IDC with your cloud resources in Alibaba Cloud VPC into a virtual IDC by using a leased line or Virtual Private Network (VPN) in order to smoothly migrate applications to the cloud.

Switch to a VPC

Prerequisites

You have created a VPC in the same region as HybridDB for MySQL. For more information about creating a VPC, see *Create a VPC*.

Procedure

1. Log on to the *HybridDB for MySQL console*.
2. Click **Manage** next to the required instance.
3. Go to the **Instance Information** page, and click **Switch to VPC**.
4. In the **Switch to VPC** dialog box that appears, specify the **VPC** and **VSwitch**, and click **OK**.

Switch to the classic network

1. Log on to the *HybridDB for MySQL console*.
2. Click **Manage** next to the required instance.
3. Go to the **Instance Information** page, and click **Switch to Classic Network**.
4. In the **Switch to Classic Network** dialog box that appears, click **OK**.

5.4 Release an instance

You can manually release Pay-As-You-Go instances based on your business needs. Released instances cannot be recovered, perform this operation cautiously.

Prerequisite

The instance must be a Pay-As-You-Go type instance.

Note:
Pay-As-You-Go instances can be released at any time. Subscription instances cannot be manually released, and they are released automatically only when they are overdue.

Procedure

1. Log on to the *HybridDB for MySQL console*.
2. Click **Manage** next to the required instance.
3. On the **Instance Information** page, click **Release**.
4. In the **Release Instance** dialog box that appears, click **OK**.
5. On the **Mobile Phone Verification** page, click **Obtain Verification Code**. Enter the verification code that you have received on your mobile phone and then click **OK**.

Note:

Released instances cannot be recovered. Make sure if you need the instance before you perform this operation.

5.5 Change Pay-As-You-Go to Subscription

After you purchase a Pay-As-You-Go instance, you can switch to the Subscription billing method as needed.

Note:

- You cannot switch the billing method from Subscription to Pay-As-You-Go. Before switching to the Subscription billing method, make sure that this operation will not cause excessive resource waste.
- After you switch the billing method, the Subscription billing method takes effect immediately. For more information about billing, see *Pricing*.
- The system generates a new order for the Subscription billing method. You must complete payment for this order to use the billing method. If the payment is not received for this order , an unpaid order occurs on the page. If this occurs, you cannot purchase any instance or change the billing method.

Prerequisites

- The required instance belongs to the current user.
- This instance uses the Pay-As-You-Go billing method, and is in **Running** status.

Procedure

1. Log on to the *HybridDB for MySQL console*.
2. Click the required instance name to go to its **Basic Information** page.
3. In the left-side navigation pane, select **Databases**.
4. Click **Subscription Billing** next to the required database.
5. On the **Switch to Subscription Billing** page that appears, drag the **Duration** slider to the required period, read and agree to the Product Terms of Service, and then click **Pay Now**.
6. On the Confirm Order page, click **Pay** to complete the payment.

After the preceding operations are completed, the billing method of the database is displayed as Subscription on the **Databases** page.

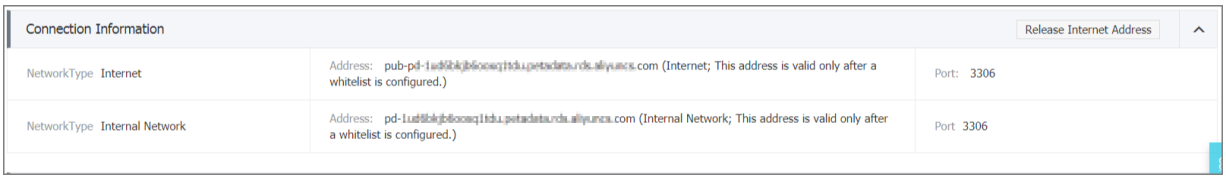
6 Manage databases

6.1 Log in to a database

Databases in HybridDB for MySQL are compatible with MySQL protocols. You can access a HybridDB for MySQL database using the MySQL client or program.

Prerequisites

- You have created an instance and a database in the *HybridDB for MySQL console*, and they are in Running status.
- You have created an instance and a database in the *HybridDB for MySQL console*, and they are in Running status.
- The client IP address is included in the whitelist. For more information about the operation, see *Set the whitelist*.
- You can also choose **Manage** > **Basic Information** in the console to see the IP address and port number of the instance.



Access HybridDB for MySQL using the MySQL client

On the MySQL client, you can use a CLI to access the database.

Note:
By default, you can use the ECS client to access HybridDB for MySQL over the internal network. To access the database over a public network, you must request a public IP address.

```
mysql –h example.petadata.rds.aliyuncs.com –P 3306 –u UserName –p
Password dbname
```

The parameters in the preceding statement are described as follows:

- **-h**: specifies the host name of the instance, that is, the internal network address or public network address of the instance. To access an instance using an internal network address, you must install the MySQL client on the ECS instance.
- **-P**: specifies the port number.

- **-u**: specifies the database account.
- **-p**: specifies the account password.
- **dbname**: specifies the database name.

6.2 View the monitoring details of a database

1. Log on to the *HybridDB for MySQL console* to go to the **Instances** page.
2. Click the required instance to go to its Basic Information page, and then in the left-side navigation pane, click **Databases**.
3. Click **Manage** next to the required database to go the **Basic Information** page of the database.
4. In the left-side navigation pane, click **Monitoring and Alarms** to view monitoring information about the instance.

   Specify the **Period** to view the running status of all statements in the specified period. The maximum monitoring cycle is 30 days.

6.3 Back up a database

You can set a backup policy for backing up a specified database automatically.

Procedure

1. Log on to the *HybridDB for MySQL console* to go to the **Instances** page.
2. Click the required instance **ID** from the list of instances, or click **Manage** next to the required instance to go to its Basic Information page.
3. In the left-side navigation pane, click **Databases**.
4. Click **Manage** next to the required database to go to its **Basic Information** page.
5. In the left-side navigation pane, click **Backup and Recovery**.
6. On the **Backup Settings** tab page, click **Edit** to set related parameters. For more information about setting backup parameters, see the following table.
7. Click **OK** to complete settings.

Table 6-1: Backup settings

Parameter	Description
Data Retention Period	Supports values from 7 days to ~730 days. The default value is 7 days.

Parameter	Description
Backup Cycle	Specifies the backup cycle by weeks.
Backup Time	Specifies the time period in which the database is backed up.
Backup Switch	Switches backup status. By default, backup is switched off.

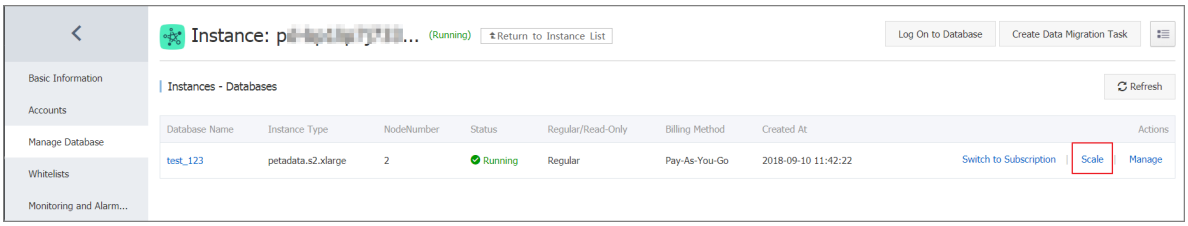
6.4 Scale up a database

Data volume and computing workload increase over time when you use HybridDB for MySQL. However, data processing speed may be limited by computing resource availability, including CPU , disk space, memory, and the number of data processing nodes. HybridDB for MySQL provides online database scaling to support dynamic instance scaling.

Note:
Scaling a database will interrupt your connection. Perform scaling during off-peak hours and make sure your applications have a reconnection mechanism.

1. Log on to the *HybridDB for MySQL console* to go to the **Instances** page.
2. Click the instance **ID**, or click **Manage** next to the required instance to enter the Basic Information page.
3. In the left-side navigation pane, click **Databases** to view the instance databases.
4. Click **Scale** next to the required database to go to the **Configuration upgrade** page.

   You can double the nodes of the database.



5. Click the Agreement of Service and then click **Activate** to finish the operation.

   - By default, HybridDB for MySQL doubles storage capacity during scaling.
   - After you complete the scaling operation, it may take 20 minutes before the service functions normally.
   - Scaling of Pay-As-You-Go instances takes effect immediately. Billing for Pay-As-You-Go instances based on the new specifications will start from the next billing cycle (next hour).

- Scaling of Subscription instances takes effect immediately. Subscription users will need to pay for the scaling immediately.

6.5 Import data

6.5.1 Import data using Data Integration

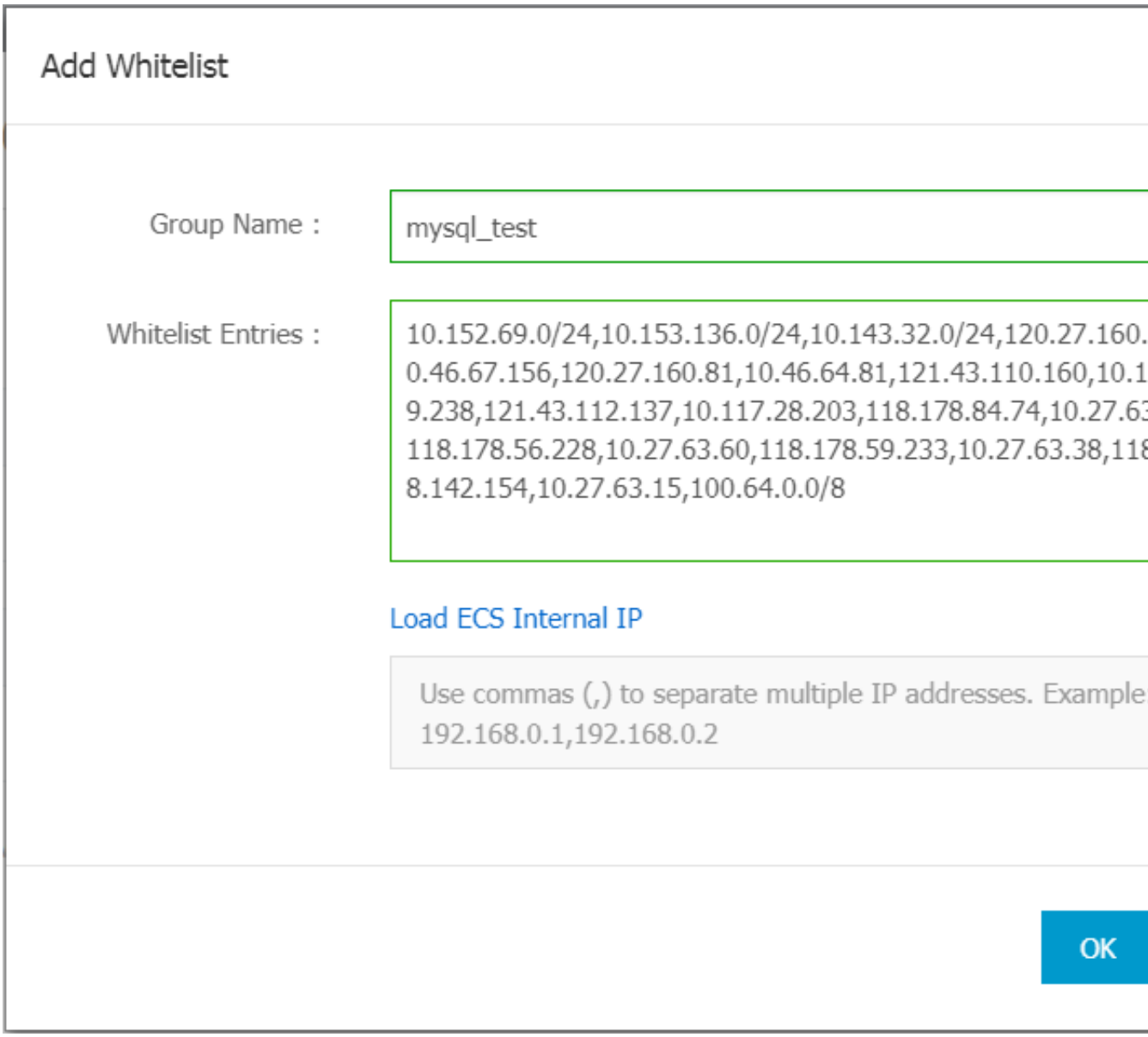
You can use *Data Integration* to import full data or filtered data to HybridDB for MySQL.

Prerequisites

1. You have created the target database and table using the HybridDB for MySQL client before importing data.
2. To import data from ApsaraDB for RDS, you must go to the RDS console and set the **IP whitelist**. For more information, see *Set the whitelist*. Go to the HybridDB for MySQL console and add the following IP addresses to the whitelist:

```
10.152.69.0/24,10.153.136.0/24,10.143.32.0/24,120.27.160.26,10.46.67
.156,120.27.160.81,10.46.64.81,121.43.110.160,10.117.39.238,121.43.
112.137,10.117.28.203,118.178.84.74,10.27.63.41,118.178.56.228,10.27
```

```
.63.60,118.178.59.233,10.27.63.38,118.178.142.154,10.27.63.15,100.64
.0.0/8
```



Note:

If you use a custom resource group to synchronize data in HybridDB for MySQL, you must add the IP addresses in the custom resource group to the whitelist of HybridDB for MySQL.

Add a data source

Note:

Only the project administrator can create a data source. Other roles can only view data sources.

For more information about the operations, see *Configure the data source*. You must select a MySQL data source.

Configure a synchronization task in Wizard Mode

For more information about the operations, see *Configure the synchronization task in wizard mode* .

Submit a data synchronization task

After you save the synchronization task, click **Run** to run this task, or click **Submit** to submit the task to the scheduling system. The scheduling system then runs the task automatically according to the configurations in the next day.

6.5.2 Import data using Data Transmission Service

You can use *Data Transmission Service (DTS)* to migrate existing data or incremental data to HybridDB for MySQL.

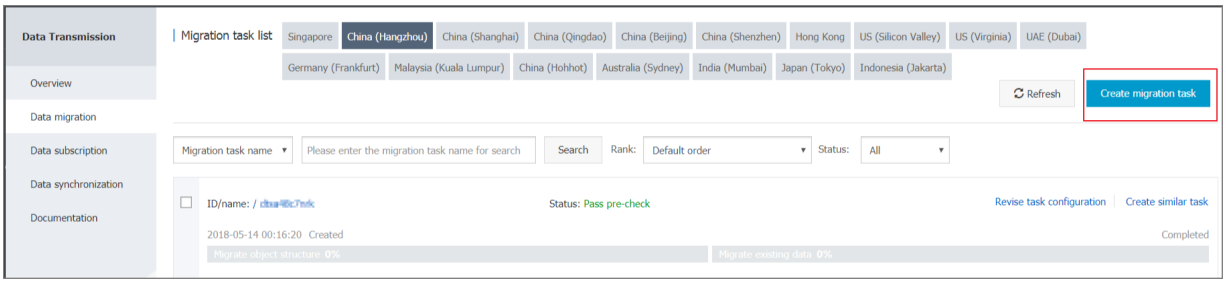
Prerequisites

You have created a database and a table that data can be migrated to in HybridDB for MySQL.

Create a task

Go to the *Data Transmission Service console*.

Go to the Data migration page and click **Create migration task**.



Source database and target database

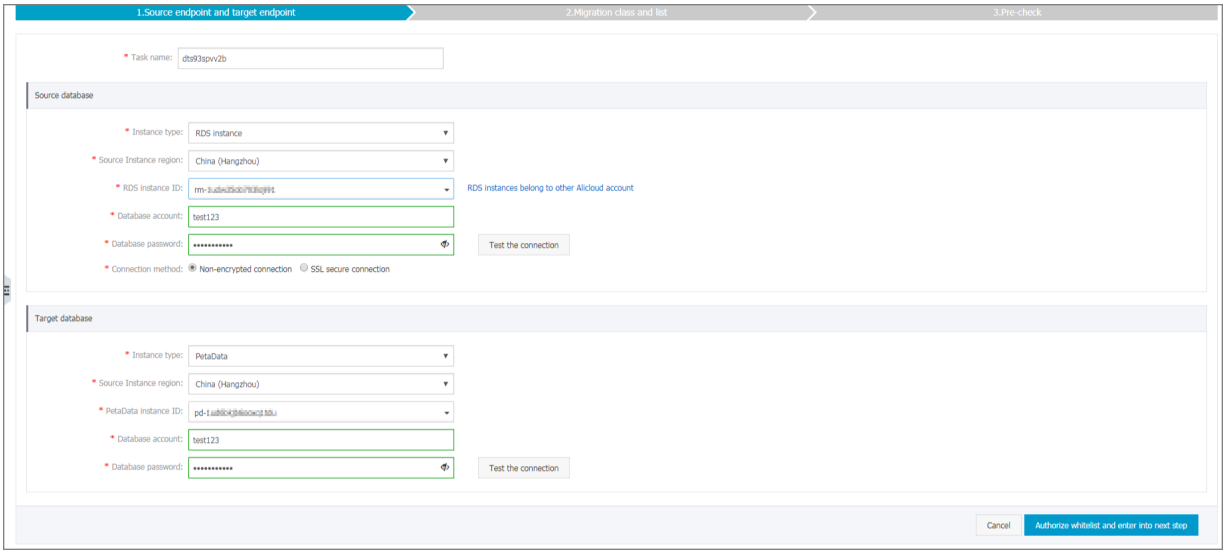
You can use an RDS instance or create a MySQL database as the source database. If you select the **RDS instance** type as the source database, select the RDS instance ID, and enter the database account and password.

Note:

DTS supports **Existing data migration** and **Incremental replication** migration type to migrate data from an RDS instance or a MySQL database to HybridDB for MySQL.

Select **PetaData** as the target database, and specify the instance ID, and database account and password.

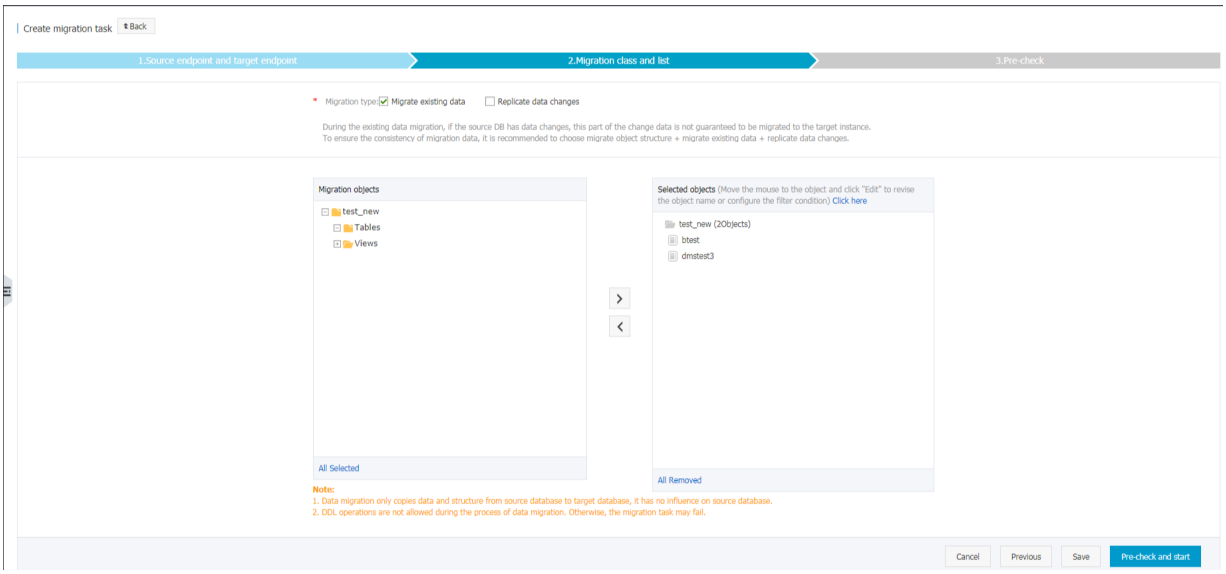
Click **Authorize whitelist and go to the next step** for configuring the source database and table.



Configure the source database and table

Next, configure the migration class and list. In the left-side pane, select the source database and table, and then click the rightwards arrow in the middle of the page to add the database and table to the right pane.

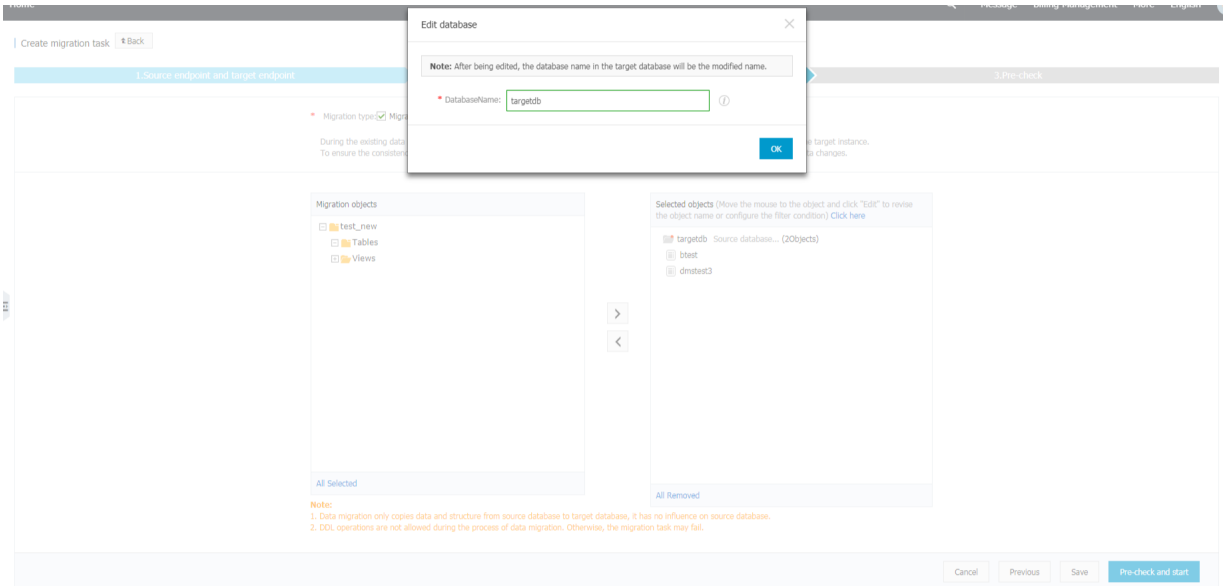
In this example, two tables (btest and dmstest3) in the source database test_new are migrated.



Configure the target database and table

Click the **Edit** next to the source database test_new to edit the database name into the target database name in HybridDB for MySQL.

In this example, data is migrated to the targetdb database in HybridDB for MySQL.



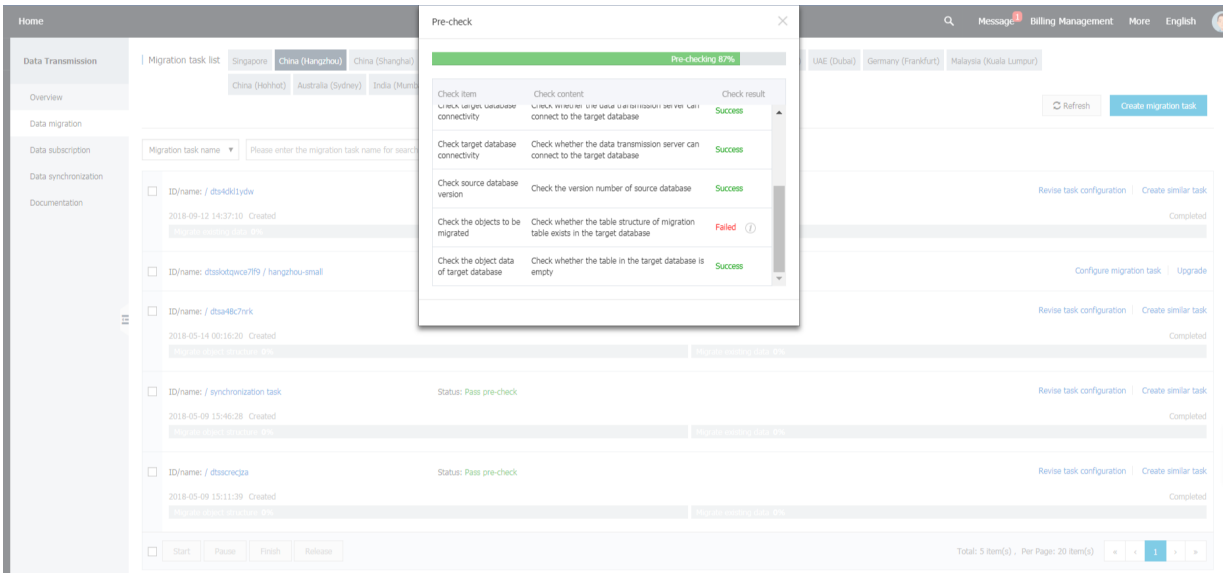
Click **OK**.

Click **Pre-check** and start to start the pre-check.

Pre-check

Note:

You must have created the target database and table before migrating data to HybridDB for MySQL. Otherwise, the following error may occur during the pre-check:



Handle the error according to the prompt, pass the pre-check, and then click **Next**.

Start migration

Migrate data using DTS.

Check the migration result

If you migrate full data only, the migration task is in Finished status after the migration. If you migrate both full and incremental data, after the full data migration is finished, the incremental data migration task stays in Migrating status until the task is finished.

7 Monitoring and alarming

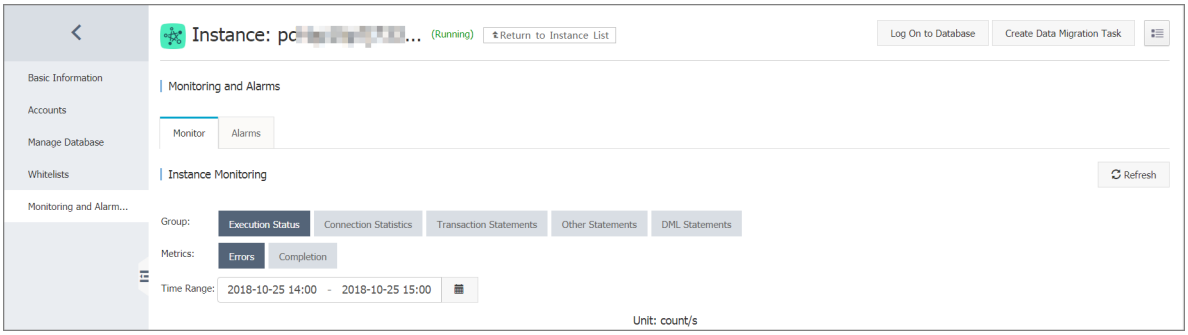
7.1 Monitoring

HybridDB for MySQL provides monitoring and alarm information of instances. You can query the instance performance metrics to analyze the instances usage, collect business trend statistics, and locate and diagnose instance problems.

- The metric data is collected every 5 minutes.
- You can only query the metric data of the last 30 days.

1. Log on to the *HybridDB for MySQL console* to go to the **Instances** page.
2. Click the instance **ID**, or click **Manage** next to the required instance to enter the **Basic Information** page.
3. In the left-side navigation pane, click **Monitoring and Alarms** to enter the **Monitoring** tab page.

   On the **Monitoring** tab page, you can specify the **Group**, **Metrics**, and **Time Range** to filter the monitoring results. For more information about monitoring, see *HybridDB for MySQL monitoring overview*.



7.2 Alarming

1. Log on to the *HybridDB for MySQL console* to go to the **Instances** page.
2. Click the required instance **ID** from the list of instances, or click **Manage** next to the required instance to go to the Basic Information page.
3. Click **Monitoring and Alarms** in the left-side navigation pane to go to the Monitoring tab page.
4. Click the **Alarms** tab to go to the Alarms tab page.

   This page lists the alarm rules that have been defined.
5. To set alarm rules, click **Set Alarm Rules** in the upper-right corner.

For more information about setting alarm rules, see *Alarm Service*.

8 Restrictions

- The data definition language (DDL) currently supports tables and indexes only.
- During the DDL operation, the transaction isolation level is set to READ UNCOMMITTED.
- The `ADD COLUMN BEFORE|AFTER XXXXXX` statement is not supported.
- The `ALTER TABLE ADD|DROP INDEX` syntax should not be used to change an index. This syntax is blocking, and affects normal operations of data manipulation language (DML) statements. Instead, `CREATE|DROP INDEX ON` should be used.
- Currently, no system libraries or system tables are available, such as information_schema.
- The following advanced features are not supported:
  - Tablespaces.
  - MySQL syntax for partitioned tables.
  - Stored procedures and user-defined functions.
  - Triggers.
  - Events.
  - Cursors.
- The following constraints are not supported:
  - The uniqueness constraint. You can create a primary key and a unique index. Only the uniqueness of the auto increment primary key can be guaranteed.
  - The foreign key constraint.
  - The CHECK constraint.

9 SQL Reference

9.1 DDL statements

9.1.1 CREATE TABLE

Standard syntax

```
CREATE TABLE [ IF NOT EXISTS ] table_name
                ( { column_name column_definition | table_constraints
 } [, ... ]  )
                DISTRIBUTE_KEY (column_name)
                [table_attribute]
```

```
DROP TABLE table_name
```

Example:

```
CREATE TABLE mytable (
                id bigint(20) NOT NULL AUTO_INCREMENT,
                name varchar(128) NOT NULL,
                ts timestamp DEFAULT CURRENT_TIMESTAMP,
                title varchar(256) DEFAULT NULL,
                content text DEFAULT NULL,
                view_count int DEFAULT 0,
                PRIMARY KEY (id),
                KEY idx_name(name)
                ) DEFAULT CHARSET=utf8
                DISTRIBUTE_KEY (name);
```

Description:

- The table name must be 1 to 32 bytes in length and can contain letters, numbers, and underscores (_).
- The column name must be 1 to 32 bytes in length and can contain letters, numbers, and underscores (_).
- Currently, the PRIMARY KEY columns and DISTRIBUTE_KEY columns cannot be updated . To change the value of a column, you must delete the column and then re-insert it with the required values.
- A table supports up to 512 columns.
- The data in a row can be up to 16 MB in size.
- The table name and column name cannot contain keywords reserved for the system. If a keyword is required, you must add a backtick (`) to the table name or column name. Make sure that you do not add a single quotation mark (').

Supported data types:

```
TINYINT[(length)] [UNSIGNED] [ZEROFILL]
                SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
                MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
                INT[(length)] [UNSIGNED] [ZEROFILL]
                INTEGER[(length)] [UNSIGNED] [ZEROFILL]
                BIGINT[(length)] [UNSIGNED] [ZEROFILL]
                DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
                FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
                DECIMAL[(length[,decimals])] [UNSIGNED] [ZEROFILL]
                NUMERIC[(length[,decimals])] [UNSIGNED] [ZEROFILL]
                CHAR[(length)] [BINARY]
                VARCHAR(length) [BINARY]
                DATE
                TIME
                DATETIME
                TIMESTAMP
                TINYTEXT [BINARY]
                TEXT [BINARY]
                MEDIUMTEXT [BINARY]
                LONGTEXT [BINARY]
                BOOL
                BOOLEAN
```

Table constraints:

```
[ KEY index_name( index_column_name [, ... ] ) ]
                [ UNIQUE KEY index_name( index_column_name
 [, ... ] ) ]
                [ PRIMARY KEY ( column_name [, ... ] ) ]

                index_column_name:
                column_name [(length)] [ASC | DESC]
```

Description:

- You must specify a primary key for your table. Otherwise, data duplication may occur during data migration.
- You must specify a name when you create an index.
- Currently, PRIMARY KEY columns cannot be updated. To update the value of a PRIMARY KEY column, you must delete the column and then re-insert it with the required value.
- You can only create a unique key using the UNIQUE INDEX index_name (col1, col2, col3, …) statement in advance. The primary key and unique key can ensure only the uniqueness of a specific partition. To ensure global uniqueness, you need to specify a field of the primary key or the unique key as the partitioning key.

Table partition

```
DISTRIBUTE_KEY (column_name)
```

```
```

Description:

- Currently, the partition key must be specified in Data Definition Language (DDL). The partition key can specify only one column, and the column data can either be of integer (TINYINT, SMALLINT, MEDIUMINT, INTEGER, and BIGINT) or character (CHAR and VARCHAR) type. Data is distributed across partitions according to the partition key.
- Currently, DISTRIBUTE_KEY columns cannot be updated. To update the value of a DISTRIBUTE_KEY column, you must delete the column and then re-insert it with the required value.

Table attributes:

```
[ DEFAULT CHARSET = table_charset ]
                [ COMMENT table_comment ]
```

Description:

- The table_comment field supports letters only.
- The character set introducer can only be utf8.

Unsupported constraint check:

Description:

- The unique-key constraint. You can create a primary key and a unique index. Only the uniqueness of the auto increment primary key can be guaranteed.
- The foreign key constraint.
- The CHECK constraint.

System database and table:

Description: Currently, no system databases or system tables are provided, including the information_schema database.

View the table definition

Standard syntax

```
SHOW CREATE TABLE table_name
                DESC table_name
                DESC table_name DISTRIBUTE INFO
```

Parameters:

- SHOW CREATE TABLE table_name: displays the table definition, where no partition key information is included.
- DESC table_name: displays all names of table columns. The partition key is not defined in the column definition.
- DESC table_name DISTRIBUTE INFO: displays the partition key of a table.

9.1.2 ALTER TABLE

Standard syntax

```
ALTER TABLE table_name [alter_specification [, alter_specification
] ...]

alter_specification:
        | ADD [COLUMN] column_name column_definition
        | ADD [COLUMN] (column_name column_definition [, column_name
column_definition] ...)
        | CHANGE [COLUMN] old_column_name new_column_name column_def
inition
        | MODIFY [COLUMN] column_name column_definition
        | DROP [COLUMN] column_name

column_definition:
 data_type [ { NOT NULL | NULL } ][ DEFAULT default_expr ] [
AUTO_INCREMENT ]
```

Parameters

- `ADD [COLUMN]`: adds a column to a table.
- `CHANGE [COLUMN]`: modifies an existing table column. You can change the column name as needed.
- `MODIFY [COLUMN]`: modifies an existing table column. The column name cannot be changed.
- `DROP [COLUMN]`: deletes a column from a table.
- You can add or modify column definitions using a statement that is similar to the CREATE TABLE statement.
- You can update the definition of multiple columns using one table definition change statement.

9.1.3 RENAME TABLE

Standard syntax

```
RENAME TABLE table_name TO new_table_name
```

Note:

- Currently, you cannot change table names using the `ALTER TABLE table_name RENAME [TO|AS] new_tbl_name` statement.
- You cannot access a table while its name is being updated.

9.1.4 CREATE INDEX

Standard syntax

```
CREATE INDEX index_name ON table_name (index_col_name,...)

index_column_name:
    column_name [(length)] [ASC | DESC]

DROP INDEX index_name ON table_name
```

Note:

- You can use the CREATE INDEX statement or the ALTER TABLE table_name ADD INDEX statement to add an index to a table.
- You can also use the DROP INDEX statement or the ALTER TABLE DROP INDEX statement to delete an index from a table.
- An index can specify whether its referenced columns are sorted in ascending or descending order.
- Adding unique index dynamically is not supported because this operation can cause inconsistent data duplicates.

9.2 DML statements

9.2.1 SELECT

Standard syntax:

The syntax of SELECT is as follows:

```
[ WITH with_subquery_table_name AS ( query ) ]
SELECT
[DISTINCT] select_expr [, select_expr ...]
[FROM table_reference [, ...] ]
[WHERE filter_condition]
[GROUP BY { expr | ROLLUP ( expr_list ) | CUBE ( expr_list ) |
GROUPING SETS ( expr_list )} , ...]
[HAVING having_condition]
[ORDER BY {col_name | expr }
[ASC | DESC], ...]
[{ UNION [ ALL ] | INTERSECT | EXCEPT } (SELECT select_expr..)]
```

```
[LIMIT {row_count}]
```

WITH clause

The WITH clause defines one or more subqueries. Each subquery defines a temporary table, similar to a view definition. Other clauses in the current query can reference the temporary table defined in the WITH clause. All temporary tables that are defined in the WITH statement can also be defined in subqueries that use the SELECT clause. When all the subqueries or temporary tables are referenced multiple times in follow-up statements, the WITH statement reuses the first result of the temporary tables to reduce the number of times common table expressions are used.

Syntax:

```
[ WITH with_subquery [, ...] ]
```

Syntax of with_subquery:

```
with_subquery_table_name AS ( query )
```

Parameters:

- with_subquery_table_name: a unique temporary table name in the current query.
- query: all SELECT queries that are supported.

Example:

```
with t as (select x,y from A) select t.y from t order by t.x limit 10
```

SELECT list

Basic structure of the projection expression in the SELECT statement:

```
SELECT [ ALL | DISTINCT ] * | expression [ AS column_alias ] [, ...]
```

Parameters:

- `ALL`: an optional redundant field that is used when the `DISTINCT` parameter is not required.
- `DISTINCT`: deletes duplicate rows.
- `*`: returns all columns.
- `expression`: one or more column references, or a column expression that includes one or more functions.
- `AS column_alias`: defines the alias of the specified column, where the `AS` keyword is optional. If the alias following `AS` is a string that contains one or more spaces, you can enclose this string using two backticks (`).

FROM clause

Syntax:

```
FROM table_reference [, ...]
```

In this statement, table_reference supports the following formats:

```
with_subquery_table_name [ [ AS ] alias ]
table_name [ * ] [ [ AS ] alias ]
( subquery ) [ AS ] alias
table_reference join_type table_reference
[ ON join_condition ]
```

Parameters:

- `with_subquery_table_name` : a subquery name defined in the WITH statement.
- `table_name`: a table name or a view name.
- `alias`: a table alias or a view alias.
- `join_type`: the join type, including [INNER] JOIN, LEFT [OUTER] JOIN, RIGHT [OUTER] JOIN, and CROSS JOIN.
- `join_condition`: the `ON` condition used in a join. The condition following `ON` can only be an equivalence relation. Non-equivalence relations are defined in the `WHERE` clause.

WHERE clause

Syntax:

```
[WHERE filter_condition]
```

In this statement, filter_condition can either be several expressions that are connected with Boolean logical operators such as `AND` and `OR`, or correlated or non-correlated subqueries or semi-joins that contain `IN`, `NOT IN`, `EXIST`, or `NOT EXIST`.

GROUP BY clause

The GROUP BY clause divides the query result into groups of rows. The syntax is as follows:

```
GROUP BY [ROLLUP | CUBE] expression [, ...]
```

The expression following GROUP BY must be a non-aggregate expression in the SELECT list.

Note:

- The GROUP BY clause does not support any alias. For example, the following GROUP BY expression is not recommended: select x+1 as t from table group by t.

HAVING clause

The HAVING clause specifies a search condition for a group or an aggregate after grouping. The syntax is as follows:

```
[ HAVING condition ]
```

Note:

- The HAVING condition must reference the expression that appears in the columns of the GROUP BY clause, or reference an aggregate expression.
- The HAVING condition supports column names rather than column aliases in the SELECT list.
- The HAVING condition does not support subscript references of columns in the SELECT list.

ORDER BY clause

The ORDER BY clause sorts data returned by a query. The syntax is as follows:

```
[ ORDER BY expression
[ ASC | DESC ]
[ LIMIT { count | ALL } ]
```

Parameters:

- `expression`: specifies a column or expression as the sort criterion for the query result set. This parameter can be the column or alias in the SELECT list, or be the column that does not appear in the SELECT list.
- `ASC | DESC`: specifies that the values in the specified column should be sorted in ascending or descending order. The Null values are treated as the lowest possible values.
- `LIMIT number | ALL`: limits the number of rows returned. The number parameter specifies the number of rows, and ALL specifies that all rows are returned.

Note:

The OFFSET clause is not supported.

UNION/INTERSECT/EXCEPT/MINUS clause

The UNION/INTERSECT/EXCEPT/MINUS clause is used to perform set operations including union, intersection, and difference. The syntax is as follows:

```
query
{ UNION [ ALL ] | INTERSECT | EXCEPT | MINUS }
```

```
query
```

Parameters:

- `query`: The query parameter must return the consistent number and type of columns prior to and following the operators.
- `UNION [ALL]`: returns the result of performing the union operation on sets. The ALL parameter specifies that deduplication is not required.
- `INTERSECT`: returns the result of performing the intersection operation on query result sets.
- `EXCEPT`: returns the result of the difference operation on query result sets.
- `MINUS`: returns the result of the difference operation on query result sets. The effect is the same as `EXCEPT`.

Order of the set operations:

`The UNION` and `EXCEPT` operators are left-associative, and start operations from the left to the right. Here is an example:

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

In this statement, the operations t1 union t2 except t3 are finished first, and then followed by global sorting of the preceding operation result according to c1.

`The INTERSECT`operator is prior to `UNION` and `EXCEPT`. Here is an example:

```
select * from t1
union
select * from t2
intersect
select * from t3
order by c1;
```

The preceding statement is equivalent to the following statement:

```
select * from t1
union
(select * from t2
intersect
select * from t3)
order by c1;
```

 **Note:**

- The query result set prior to a set operator cannot contain the ORDER BY statement. If the ORDER BY statement is required, enclose the statement in parentheses.

CONNECT BY clause in hierarchical queries

- The hierarchical query that uses the `START WITH... CONNECT BY PRIOR` clause is supported. Required fields must appear in the projection expression of the `SELECT` statement.
- The `START WITH` clause allows comparison conditions and IN expressions, but does not support subqueries and the `WHERE` statement.
- The `CONNECT BY PRIOR` clause allows one equi-join, but does not support multiple equi-joins or any non-equi joins or subqueries.
- Except `CONNECT BY PRIOR`, other `CONNECT BY` clauses are not supported.
- Other hidden fields or hierarchical functions such as `LEVEL` and `SYS_CONNECT_BY_PATH` are not supported.

```
-- Common query
select id, long_test from test start with id < 100 connect by prior id
 = long_test

-- Subquery
select * from (select id, long_test from test start with id in (1,2,3
) connect by prior id = long_test) as hier order by 1,2
```

MySQL functions supported by the SELECT statement

HybridDB for MySQL supports the following MySQL functions used in the SELECT statement:

- Unless specified otherwise, all the following functions are defined in *MySQL 5.6*.
- Currently, the following functions **can only be used in the SELECT statement**, and do not support other SQL statements, such as UPDATE, DELETE, INSERT, and REPLACE.

Table 9-1: Operator

Name	Description	Alias	Supported
AND, &&			Y
=			Y
BETWEEN AND			Y
COALESCE	return the first non-null arg		Y
&			Y
~			Y

Name	Description	Alias	Supported
^			Y
CASE			Y
DIV			Y
/			Y
=			Y
<=>	NULL-safe equal to		Y
>			Y
>=			Y
GREATEST			Y
LEAST			Y
IN			Y
NOT IN			Y
INVERVAL			Y
IS			Y
IS NOT			Y
IS NOT NULL			Y
IS NULL			Y
<<			Y
<			Y
<=			Y
LIKE			Y
-			Y
%, MOD			Y
NOT, !			Y
NOT BETWEEN AND			Y
! =, <>			Y
NOT LIKE			Y
NOT REGEXP		NOT SIMILAR TO	Y
OR			Y

Name	Description	Alias	Supported
+			Y
REGEXP		SIMILAR TO	Y
>>			Y
RLIKE		REGEXP	Y
NOT RLIKE		NOT REGEXP	Y
*			Y
-			Y
XOR			Y

Table 9-2: Control flow functions

Function name	Description	Alias	Supported
CASE WHEN[test1] THEN [result1]... ELSE [default] END	Returns resultN if testN is true, or otherwise, returns default.		Y
CASE [test] WHEN[val1] THEN [result]... ELSE [default] END	Returns resultN if test is equal to valN, or otherwise, returns default.		Y
IF(test,t,f)	Returns t if test is true , or otherwise, returns f.		Y
IFNULL(arg1,arg2)	Returns arg1 if arg1 is not null, or otherwise, returns arg2.		Y
NULLIF(arg1,arg2)	Returns NULL if arg1 =arg2, or otherwise, returns arg1.		Y

Table 9-3: String functions

Function name	Description	Alias	Supported
ASCII(char)	Returns an ASCII value of a specified character.	ORD(char)	Y

Function name	Description	Alias	Supported
BIN(n)	Returns a binary value of a specified character.	CONV(N, 10, 2)	Y
BIT_LENGTH(str)	Returns the size of a string in bits.		Y
CHAR(N, ...)	Returns the character for each integer passed to the function.		Y
CHAR_LENGTH(str)	Returns the length of the str string , measured in characters.	CHARACTER_ LENGTH(str)	Y
CONCAT(s1,s2...,sn)	Concatenates s1, s2, ..., and sn into a string, and if sn is NULL, returns NULL.		Y
CONCAT_WS(sep,s1, s2...,sn)	Concatenates s1, s2, ..., and sn into a string using the delimiter that is specified in the sep field.		Y
ELT(N, s1, s2, ...)	Returns the Nth element of the list of strings.		Y
EXPORT_SET(bits, on, off [,separator [, number_of_bits]])			Y
FIELD(target, s1, s2 ,...)	Returns the position of target in the s1, s2... string list.		Y
FIND_IN_SET(str, strlist)	Analyzes the strlist list where each element is separated by commas (,), and if str is available, returns the position of str in strlist.		Y

Function name	Description	Alias	Supported
FORMAT(X, D)			Y
FROM_BASE64(str)			Y
HEX(str), HEX(N)			Y
INSERT(str,x,y,instr) INSERT(str,pos,len, newstr)	One of the following results occurs: • Returns the str string after replacing the substring that contains y characters starting from the xth character in str with instr. • Returns the original string if the xth character is not within the length of the string. • Replaces the rest of the string from the xth character if the length of y characters is not within the length of the rest of the string . • Returns NULL if any argument is NULL.		Y
INSTR(str, substr)			Y
LCASE(str)	Returns the str string after converting all characters in str into lower-case characters.	LOWER(str)	Y
LOWER(str)			Y

Function name	Description	Alias	Supported
LEFT(str,x)	Returns the leftmost x characters in the str string.		Y
LENGTH(s)	Returns the size of the str string in bytes.		Y
LOCATE(substr, str), LOCATE(substr, str, pos)			Y
LPAD(str, len, padstr)			Y
LTRIM(str)	Removes the spaces at the beginning of the str string.		Y
MAKE_SET(bits, str1, str2, ...)			Y
MID(str, pos, len)		SUBSTRING(str, pos, len)	Y
OCT(N)		CONV(N, 10, 8)	Y
OCTET_LENGTH(str)		LENGTH(str)	Y
ORD(str)			Y
POSITION(substr IN str)	Returns the position where the substr substring appears in the str string for the first time.	LOCATE(substre, str)	Y
QUOTE(str)	Escapes single quotes (') in the str string using backslashes (\).		Y
REPEAT(str, count)	Returns a string consisting of the str string that is repeated count times.		Y
REPLACE(str, from_str, tp_str)			Y
REVERSE(str)	Returns the str string with the order of the characters reversed.		Y

Function name	Description	Alias	Supported
RIGHT(str, len)	Returns the rightmost len characters from the str string.		Y
RPAD(str, len, padstr)			Y
RTRIM(str)	Returns the str string with trailing space characters removed.		Y
SPACE(N)	Returns a string consisting of N space characters.		Y
SUBSTR(str, pos), SUBSTR(str FROM pos), SUBSTR(str, pos, len), SUBSTR(str FROM pos FOR len)		SUBST(str, pos[, length])	Y
SUBSTRING(str, pos), SUBSTRING (str FROM pos), SUBSTRING(str, pos, len), SUBSTRING(str FROM pos FOR len)		SUBSTRING(str, pos [, length])	Y
SUBSTRING_INDEX(str, delim, count)			Y
TO_BASE64(str)			Y
TRIM([{BOTH / LEADING / TRAILING } [remstr] FROM] str), TRIM([remstr FROM] str)	Deletes all spaces or specified characters at the head and end of the string.	TRIM_STR([remchar,] str)	Y
UCASE(str)	Returns the result of capitalizing all characters in the str string.	UPPER(str)	Y
UNHEX(str)			Y
UPPER(str)			Y

Function name	Description	Alias	Supported
expr LIKE pat [ESCAPE 'escape_char']			Y
expr NOT LIKE pat [ESCAPE 'escape_char']			Y
STRCMP(expr1, expr2)			Y
expr NOT REGEXP pat, expr NOT RLIKE pat		NOT SIMILAR TO	Y
expr REGEXP pat, expr RLIKE pat		SIMILAR TO	Y

Table 9-4: Numeric functions

Function name	Description	Alias	Supported
DIV, /, -, %, MOD, +, *, -	Arithmetic operators		Y
ABS(x)	Returns the absolute value of x.		Y
ACOS(x)			Y
ASIN(x)			Y
ATAN(x)			Y
ATAN(Y, X), ATAN2(Y, X)			Y
CEIL(X)		CEILING(x)	Y
CEILING(x)	Returns the smallest integer not less than x.	CEIL(x)	Y
CONV(N, from_base, to_base)			Y
COS(x)			Y
COT(x)			Y
CRC32(expr)			Y

Function name	Description	Alias	Supported
DEGREES(x)			Y
EXP(x)	Returns e (the base of the natural logarithm) raised to the power of x.		Y
FLOOR(x)	Returns the largest integer not greater than x.		Y
FORMAT(X, D)			Y
HEX(N_or_S)			Y
LN(x)	Returns the natural logarithm of x.		Y
LOG(b,x), LOG(x)	Returns the base-b logarithm of x, where b is approximately 2.718 by default.		Y
LOG2(x)	Returns the base-2 logarithm of x.		Y
LOG10(x)	Returns the base-10 logarithm of x.		Y
MOD(N, M), N % M, N MOD M	Returns the result of N modulo M.		Y
PI()	Returns the value of pi (the ratio of a circle's circumference to its diameter).		Y
POW(x, y)		POWER(x, y)	Y
POWER(x, y)		POW	Y
RADIANS(x)			Y
RAND([N])	Returns a random number between 0 and 1. You can seed the RAND() random number generator to return a specified value.		Y

Function name	Description	Alias	Supported
ROUND(x,[y])	Returns the result of rounding the x parameter to y decimal places. The default y is 0, where x is an integer.		Y
SIGN(x)	Returns the sign of the number x, including -1, 0, and 1, indicating whether x is positive, negative, or zero.		Y
SIN(x)			Y
SQRT(x)	Returns the square root of the number x.		Y
TAN(x)			Y
TRUNCATE(x,y)	Returns the result of truncating the number x to y decimal places.		Y

Table 9-5: Date and time functions

Function name	Description	Alias	Supported
ADDDATE(date, INTERVAL expr unit), ADDDATE(expr, days)	Only supports the format of ADDDATE(expr, days).		Y
ADDTIME(expr1, expr2)	The precision is different between time formats '01:00:00.999999' and '25:00:01'.		Y
CONVERT_TZ(dt, from_tz, to_tz)			Y
CURDATE()	Returns the current date.	CURRENT_DATE()	Y
CURRENT_DATE, CURRENT_DATE()		CURDATE()	Y

Function name	Description	Alias	Supported
CURRENT_TIME([fsp])	Returns the current time.	CURTIME()	Y
CURRENT_TIMESTAMP([fsp])			Y
CURTIME([fsp])			Y
DATE(expr)			Y
DATEDIFF(expr1, expr2)			Y
DATE_ADD(date, INTERVAL int unit)		ADDDATE	Y
DATE_FORMAT(date, format)			Y
DATE_SUB(date, INTERVAL expr, unit)		SUBDATE	Y
DAY(date)		DAYOFMONTH	Y
DAYOFWEEK(date)	Returns the day (1~7) of the week for date (1 = Sunday, 2 = Monday, ..., 7 = Saturday).		Y
DAYOFMONTH(date)	Returns the day (1~31) of the month for date.	DAY(date)	Y
DAYOFYEAR(date)	Returns the day (1~366) of the year for date.		Y
DAYNAME(date)	Returns the name of a day of the week for date.		Y
EXTRACT(unit FROM date)	Currently, the unit field is not fully compatible with all unit values of MySQL.		Y
FROM_DAYS(N)			Y
FROM_UNIXTIME(unix_timestamp), FROM_UNIXTIME	Supports FROM_UNIXTIME(unix_timestamp).		Y

Function name	Description	Alias	Supported
(unix_timestamp, format)			
GET_FORMAT({DATE / TIME / DATETIME}, {'EUR'/'USA'/'JIS'/'ISO'/'INTERNAL'})			Y
HOUR(time)	Returns the hour (0~23) of the time value. Note: The hour can be larger than 24 in MySQL.		Y
LAST_DAY(date)	Does not support any invalid date. For example, the database returns null for '2003-03-32'.		Y
LOCALTIME([fps])		NOW()	Y
LOCALTIMESTAMP([fsp])		NOW()	Y
MAKEDATE(year, dayofyear)			Y
MAKETIME(hour, minute, second)			Y
MICROSECOND(expr)			Y
MINUTE(time)	Returns the minute (0~59) of the time value.		Y
MONTH(date)	Returns the month (1~12) of the date value.		Y
MONTHNAME(date)	Returns the month name of the date value.		Y
NOW([fsp])	Returns the current date and time.	CURRENT_TIMESTAMP(), LOCALTIME(), LOCALTIMESTAMP(), SYSDATE()	Y

Function name	Description	Alias	Supported
PERIOD_ADD(P, N)			Y
PERIOD_DIFF(P1, P2)			Y
QUARTER(date)	Returns the quarter (1~4) in a year of the date value.		Y
SECOND(time)			Y
SEC_TO_TIME(seconds)			Y
STR_TO_DATE(str, format)			Y
SUBDATE(date, INTERVAL expr unit), SUBDATE(expr, days)			Y
SUBTIME(expr1, expr2)			Y
SYSDATE([fsp])			Y
TIME(expr)			Y
TIMEDIFF(expr1, expr2)			Y
TIMESTAMP(expr), TIMESTAMP(expr1, expr2)			Y
TIMESTAMPADD(unit, internal, datetime_expr)			Y
TIMESTAMPDIFF(unit, datetime_expr1, datetime_expr2)			Y
TIME_FORMAT(time, format)			Y
TIME_TO_SEC(time)			Y
TO_DAYS(date)	Supports strings only.		Y
TO_SECONDS(expr)			Y

Function name	Description	Alias	Supported
UNIX_TIMESTAMP([date])			Y
UTC_DATE()			Y
UTC_TIME()			Y
UTC_TIMESTAMP([fsp])			Y
WEEK(date[,mode])	Returns the week (0~53) of the year for date.		Y
WEEKDAY(date)			Y
WEEKOFYEAR(date)			Y
YEAR(date)	Returns the year (1000~9999) of the date value.		Y
YEARWEEK(date[,mode])			Y

Table 9-6: Cast functions

Function name	Description	Alias	Supported
CAST(expr AS type)	Converts only some data types as a subset of MySQL CAST.		Y
CONVERT(expr, type)		CAST(expr AS type)	Y
CONVERT(expr USING transcoding_name)			Y

Table 9-7: Bit functions and operators

Function name	Description	Alias	Supported
BIT_COUNT()			Y
&, ~, , ^, <<, >>			Y

Table 9-8: Information functions

Function name	Description	Alias	Supported
DATABASE()	Returns the name of the current database.	SCHEMA()	Y
BENCHMARK(count, expr)			N
CHARSET(str)			Y
COERCIBILITY(str)			Y
COLLATION(str)			Y
CONNECTION_ID()			Y
SCHEMA()			Y
FOUND_ROWS()	Returns the total number of rows retrieved in the last query using the SELECT statement.		N
LAST_INSERT_ID([expr])			Y
ROW_COUNT()			N
USER()	Returns the current username.	SYSTEM_USER(), SESSION_USER()	Y
VERSION()	Returns the MySQL server version.		Y

Table 9-9: Aggregate functions

Function name	Description	Alias	Supported
AVG([DISTINCT] expr)	Returns the average of the specified column.		Y
BIT_AND(expr)			Y
BIT_OR(expr)			Y
BIT_XOR(expr)			Y
COUNT(expr)			Y

Function name	Description	Alias	Supported
COUNT(DISTINCT expr,[expr...])	Returns the number of non-NULL values in the specified column.		Y
GROUP_CONCAT(expr)	Returns the result of concatenating the column values in the same group.		Y
MAX([DISTINCT] expr)	Returns the maximum of the specified column.		Y
MIN([DISTINCT] expr)			Y
SUM([DISTINCT] expr)	Returns the sum of all values in the specified column.		Y
STD(expr)			Y
STDDEV(expr)			Y
STDDEV_POP(expr)			Y
STDDEV_SAMP(expr)			Y
VAR_POP(expr)			Y
VAR_SAMP(expr)			Y
VARIANCE(expr)			Y
WITH ROLLUP	Uses the statement `GROUP BY ROLLUP( C1, C2,..., Cn)`, which is different from MySQL.		Y

Compatibility

- HybridDB for MySQL does not use any number, which is out of the permissible range of the column data type, from a user-defined function. Otherwise, an error occurs to indicate that the number is out of range.

MySQL functions unsupported by the SELECT statement

Compared with *functions in MySQL 5.6*, HybridDB for MySQL does not support the following MySQL functions:

Table 9-10: Operator

Name	Description	Alias	Supported
:=			N
BINARY		CAST(expr AS BINARY), CONVERT(expr USING BINARY)	N
ANY, SOME			N

Table 9-11: String functions

Function name	Description	Alias	Supported
LOAD_FILE(file_name)			N
SOUNDEX(str)			N
SOUNDS LIKE			N
WEIGHT_STRING(str [AS {CHAR / BINARY }(N)] LEVEL levels flags)			N

Table 9-12: Cast functions

Function name	Description	Alias	Supported
BINARY			

Table 9-13: Information functions

Function name	Description	Alias	Supported
FOUND_ROWS()	Returns the total number of rows retrieved in the last query using the SELECT statement.		
ROW_COUNT()			

Table 9-14: Miscellaneous functions

Function name	Description	Alias	Supported
DEFAULT(col_name)			
FORMAT(X,D)			
GET_LOCK(str, timeout)			
INET_ATON(expr)			
INET_NTOA(expr)			
INET6_ATON(expr)			
INET6_NTOA(expr)			
IS_FREE_LOCK(str)			
IS_IPV4(expr)			
IS_IPV4_COMPAT(expr)			
IS_IPV4_MAPPED(expr)			
IS_IPV6(expr)			
IS_USERD_LOCK(str)			
MASTER_POS_WAIT(log_name, log_pos[, timeout])			
NAME_CONST(name,value)			
RELEASE_LOCK(str)			
SLEEP(duration)			
UUID()			
UUID_SHORT()			
VALUES(col_name)			

Oracle functions supported by the SELECT statement

Currently, HybridDB for MySQL supports the following Oracle functions in the query that uses the SELECT statement:

Table 9-15: Operator

Function name	Description	Alias	Supported
ROLLUP	Used in the GROUP BY clause, such as `GROUP BY ROLLUP( C1, C2, ..., Cn)`.		Y
CUBE	Used in the GROUP BY clause, such as `GROUP BY CUBE(C1 , C2, ..., Cn)`.		Y
GROUPING			Y
OVER	The OVER clause that specifies the window that the window function applies to.		Y
RANK	The rank function that can be used with a window function.		Y
DENSE_RANK	The rank function that can be used with a window function.		Y
ROW_NUMBER	The rank function that can be used with a window function.		Y

9.2.2 DELETE

Standard syntax

```
DELETE FROM table_name WHERE filter_condition
```

Restrictions

- This statement currently does not support distributed transactions. If you delete multiple rows across partitions at a time, the database starts an incomplete distributed transaction. If the transaction is committed successfully only in some of those partitions, data inconsistency may occur after the rollback. Exercise caution when applying distributed transactions, or avoid this transaction type.
- You can delete the whole table. However, we recommend that you avoid this operation.

- The LIMIT clause in the DELETE statement is not supported.

9.2.3 INSERT

Standard syntax

```
INSERT [IGNORE] [INTO] table_name (column_name [, ...]) VALUES (
insert_expr_list) [, insert_expr_list [, ...]] [on duplicate key
update column_name = expr [, ...]]

insert_expr_list:
    insert_expr [, ...]
```

Restrictions

- You can insert multiple records across multiple partitions.
- This statement currently does not support distributed transactions. If you replace multiple rows across partitions at a time, the database starts an incomplete distributed transaction. If the transaction is committed successfully only in some of those partitions, data inconsistency may occur after the rollback. Exercise caution when applying distributed transactions, or avoid this transaction type.
- If your table contains an auto increment primary key, columns are generated based on this primary key when you run the REPLACE statement. The auto increment primary key generates a unique value. This is not necessarily a monotonically increasing or continuous value.
- Character set introducers, such as _utf8'a', must be excluded from the insert_expr column.
- You must ensure that the column types defined in the partition key match the column types that you want to insert. If you define an integer column but insert a float or string value when creating a table, data is unordered due to data truncation.
- Similar to the REPLACE statement, the ON DUPLICATE KEY UPDATE statement cannot ensure the uniqueness of non-partitioning fields. We recommend that you use unique non-partitioning fields, even though duplicate non-partitioning fields are valid in syntax.

INSERT SELECT Clause

The syntax of the INSERT SELECT clause is as follows:

```
INSERT [IGNORE] INTO tbl_name (col_name [, col_name] ...) SELECT ...
REPLACE INTO tbl_name (col_name [, col_name] ...) SELECT ...
```

The restrictions for this clause are as follows:

- The SELECT clause must contain the partitioning key of the INSERT/REPLACE target table.

- The INSERT/REPLACE and SELECT clauses must contain specific column names.
- If the INSERT/REPLACE target table contains an auto-increment ID primary key, the SELECT clause need to explicitly assign the auto-increment ID primary key.

9.2.4 UPDATE

Standard syntax

```
UPDATE table_name SET column_name = update_expr [, ...] WHERE
filter_condition
```

Restrictions

- This statement currently does not support distributed transactions. If you update multiple rows across partitions at a time, the database starts an incomplete distributed transaction. If the transaction is committed successfully only in some of those partitions, data inconsistency may occur after the rollback. Exercise caution when applying distributed transactions, or avoid this transaction type.
- The PRIMARY KEY columns and DISTRIBUTE_KEY columns cannot be updated. To change the value of one of these columns, you must delete the column and then re-insert it with the required value.
- Character set introducers, such as _utf8'a', must be excluded from the update_expr column.
- The LIMIT clause in the UPDATE statement is not supported.

9.2.5 REPLACE

Standard syntax

```
REPLACE [INTO] table_name (column_name [, ...]) VALUES (replace_ex
pr_list) [, (replace_expr_list) [, ...]]
```

Restrictions

- This statement currently does not support distributed transactions. If you replace multiple rows across partitions at a time, the database starts an incomplete distributed transaction. If the transaction is committed successfully only in some of those partitions, data inconsistency may occur after the rollback. Exercise caution when applying distributed transactions, or avoid this transaction type.
- If your table contains an auto increment primary key, columns are generated based on this primary key when you run the REPLACE statement. The auto increment primary key generates a unique value. This is not necessarily a linear increasing or continuous value.

- Character set introducers, such as _utf8'a', must be excluded from the replace_expr column.
- Make sure that the column types defined in the partition key match the column types that you want to replace. If you define an integer column but replace it with a float or string value when creating a table, data is unordered due to data truncation.
- The SET clause in the REPLACE statement is not supported.

9.3 Transactions

9.3.1 Basic SET statements

- **SET autocommit = 0/1**: The `SET autocommit = 0` statement is used to have a transaction open all the time during a session. If you do not explicitly commit the transaction, HybridDB for MySQL does not commit any updates.
- **SET autocommit = 0 converted to SET autocommit = 1**: If you start a transaction by running the `SET autocommit = 0` statement, but do not commit this transaction manually, you can run the `SET autocommit = 1` statement to commit this transaction. In this case, HybridDB for MySQL implicitly sends the `SET autocommit = 1` statement to all partitions to commit the transaction in the same way as manual commitment. MySQL ends a transaction after each commitment. Similarly, after running the `SET autocommit = 1` statement successfully, HybridDB for MySQL returns an OK response to the client, and ends the transaction.
- **SET TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED|READ COMMITTED| REPEATABLE READ|SERIALIZABLE}**: The transaction isolation level of a single-partition transaction is compatible with the MySQL transaction isolation level. The transaction isolation level of a multi-partition transaction is always set to READ_COMMITTED. `SET TRANSACTION ISOLATION LEVEL XXXXXX` and `SET TX_ISOLATION = XXXXXX` only apply to the transaction isolation level of a single-partition transaction.
- **SET TRANSACTION READ ONLY|WRITE**: This statement is used to establish the current transaction as read-only or read/write.
- **SET NAMES XXXXXX**: This statement is used to specify the character set used to establish the current transaction.
- During a transaction, when you set environment variables, HybridDB for MySQL sends an environment variable setting statement immediately to the corresponding partitions of the transaction. Except for converting SET autocommit = 0 to SET autocommit = 1, other SET statements do not affect the transaction status if failed.

- When the transaction accesses a new partition, all environment variables are updated to ensure that all partitions use the latest environment variables.
- HybridDB for MySQL does not support environment variables that are dependent on each other, such as, `SET a = 1`, `SET b = a + 1`, `SET a = b + 1`. You must specify all environment variables by using a self-contained and idempotent single-variable setting statement.

9.3.2 Other SET statements

- The parser intercepts other environment variables, and reports a warning to indicate that the environment variable setting has not been performed.
- In non-transaction scenarios, HybridDB for MySQL delays starting a transaction when setting environment variables or running the BEGIN statement. If an environment variable setting statement or BEGIN statement arrives, HybridDB for MySQL immediately returns an OK response to the client. If an actual request arrives, HybridDB for MySQL runs another environment variable setting statement or BEGIN statement in the corresponding partitions. Then, HybridDB for MySQL delivers the request accordingly. If the delayed transaction fails, HybridDB for MySQL reports an error. When another request arrives, HybridDB for MySQL still runs another environment variable setting statement or BEGIN statement.

9.3.3 BEGIN

The BEGIN statement starts a transaction.

For consecutive BEGIN statements, if you starts the first transaction using `BEGIN/START TRANSACTION/SET autocommit = 0`, and without committing the first transaction, starts the second transaction using BEGIN/START TRANSACTION, HybridDB for MySQL implicitly commits the first transaction. The implicit commitment has the same effect as the common commitment. Then, HybridDB for MySQL continues with the second transaction. The process is the same as MySQL.

For a single-row transaction that is not protected by an explicit transaction statement: If the transaction runs in a single partition, the transaction has the same features as a MySQL transactio n. If the transaction runs in multiple partitions, HybridDB for MySQL uses the one-phase commit (1PC) protocol to commit the transaction. If the transaction is committed successfully only in some of the corresponding partitions, inconsistent rollbacks may occur.

9.3.4 COMMIT

Currently, HybridDB for MySQL only uses the one-phase commit protocol (1PC). If a transaction is committed successfully only in some of the corresponding partitions, and other partitions fail to commit the transaction or they end the transaction due to an exception, data inconsistency may occur. HybridDB for MySQL ends the transaction, no matter whether the commitment is successful or has failed. If the commitment is successful, all updates are visible. If the commitment has failed, all updates roll back automatically.

9.3.5 ROLLBACK

No matter whether partition rollbacks are successful or have failed, or whether partitions are disconnected due to exceptions, HybridDB for MySQL always returns an OK response to the client.

9.3.6 DEADLOCK

- When transactions are in deadlock, HybridDB for MySQL allows the operation for only one transaction, removes the deadlock of the other transactions, and rolls back the deadlocked transactions.
- If an UPDATE statement in a transaction applies to only one partition, but the deadlocked rows exist in two partitions, HybridDB for MySQL cannot immediately detect the deadlock. The UPDATE statements in the deadlocked transactions then request locks. If a lock wait timeout occurs, HybridDB for MySQL reports an UPDATE error. However, this error cannot cause transaction rollbacks in related partitions, so these partitions continue to process the transactions normally. Afterward, HybridDB for MySQL reports the lock wait timeout error.
- If an UPDATE statement in a transaction applies to only one partition, and the deadlocked rows exist in only that partition, HybridDB for MySQL immediately detects the deadlock. HybridDB for MySQL then allows the operation for only one of the deadlocked UPDATE transactions, and rolls back the other deadlocked transactions immediately. UPDATE statements in transactions may occur in multiple partitions. Therefore, HybridDB for MySQL locks all rows for the transactions that have not passed the MySQL deadlock detection. HybridDB for MySQL also removes the deadlock of these transactions, and you can only roll back these transactions. HybridDB for MySQL continues to normally process the transaction that has passed the MySQL deadlock detection. HybridDB for MySQL also reports the deadlock error. Transactions that are not rolled back will cause a rollback error.

- If an UPDATE statement in a transaction applies to multiple partitions, but the deadlocked rows exist in two partitions, HybridDB for MySQL cannot immediately detect the deadlock. The UPDATE statements in the deadlocked transactions then request locks. When a lock wait timeout occurs, HybridDB for MySQL reports an UPDATE error. However, this error cannot cause transaction rollbacks in related partitions, so these partitions continue to process the transactions normally. Afterward, HybridDB for MySQL reports an inconsistent data error.
- If an UPDATE statement in a transaction applies to multiple partitions, but the deadlocked rows exist in one partition, HybridDB for MySQL immediately detects the deadlock. HybridDB for MySQL then allows the operation for only one of the deadlocked UPDATE transactions, and rolls back the other deadlocked transactions immediately. UPDATE transactions may occur in multiple partitions. Therefore, HybridDB for MySQL locks all rows for the transactions that have not passed the MySQL deadlock detection. HybridDB for MySQL also removes the deadlock of these transactions, and you can only roll back these transactions. HybridDB for MySQL continues to normally process the transaction that has passed the MySQL deadlock detection . HybridDB for MySQL also reports the inconsistent data error. Transactions that are not rolled back will cause a rollback error.

9.3.7 Disconnection exceptions

If the current session is in the transaction status, and a partition related to this transaction closes the connection to the session due to an exception, this session closes the connection to the client , and rolls back the transaction.

9.3.8 Inconsistencies

Inconsistencies

HybridDB for MySQL currently only supports complete transactions in a single partition. If a request updates multiple partitions, HybridDB for MySQL cannot guarantee successful commitment in all these partitions. If the transaction is committed successfully only in some of the corresponding partitions, data inconsistency may occur.

Inconsistent transaction status

During a transaction, if a deadlocked partition ends the transaction, but other partitions may continue with the transaction, the transaction status is inconsistent among the corresponding partitions. In this case, the HybridDB for MySQL client can only roll back the partitions in transactio n status by sending a rollback notification to these partitions.

When starting the transaction, if HybridDB for MySQL runs the BEGIN/START TRANSACTION statement successfully only in some of the corresponding partitions, the transaction stops accessing the successful partitions to reduce the number of partitions where transaction commitment is failed, minimizing the impact of the failed commitment. HybridDB for MySQL then returns an error to the client.

When starting the transaction, if HybridDB for MySQL runs the SET autocommit = 0 statement successfully only in some of the corresponding partitions, the transaction stops accessing the failed partitions where the running of the statement is delayed. The successful partitions continue with the transaction. HybridDB for MySQL then returns an error to the client.

Inconsistent committed data

During a single-row multi-partition transaction, if HybridDB for MySQL runs a statement successfully only in some of the corresponding partitions, HybridDB for MySQL returns an exception to the client. Therefore, data is inconsistent among the corresponding partitions, but you cannot roll back the successful partitions to achieve data consistency.

During a multi-row multi-partition transaction, if HybridDB for MySQL runs the COMMIT or implicit COMMIT statement successfully only in some of the corresponding partitions, HybridDB for MySQL returns an exception to the client. Therefore, data is inconsistent among the corresponding partitions, but you cannot roll back the successful partitions to achieve data consistency.

Inconsistent uncommitted data

During a multi-row multi-partition transaction, if HybridDB for MySQL runs a common statement successfully only in some of the corresponding partitions, HybridDB for MySQL returns a data inconsistency error to the client. You can only roll back the failed partitions, but cannot roll back the partitions where the transaction is committed successfully.

9.4 Other statements

Restrictions on other SQL statements:

- The GRANT statement is not supported.
- The desc XXXXXX statement can be used to define a table.
- The SHOW PROCESSLIST statement returns all existing connections. However, the SHOW FULL PROCESSLIST statement is not supported.
- The SHOW SLAVE HOSTS|STATUS statement is not supported.
- The SHOW BINARY|MASTER LOGS statement is not supported.

- The SHOW OPEN TABLES statement is not supported.
- The SHOW MASTER STATUS statement is not supported.
- The SHOW BINLOG EVENTS statement is not supported.
- The SHOW ENGINE XXXXXX STATUS statement is not supported.
- The SHOW GRANTS FOR XXXXXX statement is not supported.
- The KILL/KILL CONNECTION XXXXXX statement can be used to forcibly disable a specified connection. The SELECT CONNECTION_ID() and SHOW PROCESSLIST statements can be used to return the connection ID.

9.5 The connection pool and clients

Connect the client

- To use JDBC or other connectors, you must specify the database by which to connect. Otherwise, the connection fails.
- The client cannot access the database using cursors.
- The multi-partition statement requires a connection to each partition. The connections increase in proportion to the number of partitions.
- Length of the COM_QUERY statement: A SQL statement cannot exceed 16 MB in length. Therefore, the size of a single row also cannot exceed 16 MB.
- The client supports the following command types: COM_PING | COM_QUIT | COM_QUERY | COM_KILL | COM_INIT_DB | COM_PROCESS_INFO | COM_PROCESS_KILL. The client does not support other commands, such as COM_STMT_XXXXXX in the PREPARE statement or COM_BINLOG_XXXXXX in the BINLOG statement.

Server connection pool

- HybridDB for MySQL provides a server connection pool. By default, this connection pool is disabled. You must submit a ticket to request enabling the connection pool.
- The connection pool effectively reduces the connections between the Compute Engine and partitions, and fully reuses existing connections. However, this requires that HybridDB for MySQL runs each statement for a single-row transaction by using the current database name and all current environment variables after the connection pool is enabled.
- After running the statement, HybridDB for MySQL immediately returns the corresponding partitions to the global connection pool.
- After the connection pool is enabled, for a multi-row transaction started by BEGIN/START TRANSACTION/SET autocommit = 0, HybridDB for MySQL uses the current database name

and all current environment variables only when running the first environment variable setting statement. After the commitment, rollback, or implicit commitment, HybridDB for MySQL returns the corresponding partitions to the global connection pool.

- The connection pool may affect the database performance if enabled.