

Distributed Database Middleware

User Guide

Issue 01
Date 2021-10-13



Copyright © Huawei Technologies Co., Ltd. 2023. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Service Overview.....	1
1.1 Overview.....	1
1.2 Basic Concepts.....	2
1.3 Product Architecture.....	3
1.4 Functions.....	3
1.5 DDM Instance Classes.....	5
1.6 Restrictions.....	5
1.6.1 Network Access.....	5
1.6.2 MySQL DB Instances.....	5
1.6.3 SQL Syntax.....	6
1.7 Regions and AZs.....	7
1.8 Application Scenarios.....	7
2 Getting Started.....	9
2.1 Overview.....	9
2.2 Step 1: Log In to the DDM Console.....	10
2.3 Step 2: Create a DDM Instance.....	11
2.4 Step 3: Create an RDS DB Instance.....	12
2.5 Step 4: Create a Schema.....	13
2.6 Step 5: Create a DDM Account.....	14
2.7 Step 6: Log In to the DDM Instance or Schema.....	15
3 Function Overview.....	21
4 Instance Management.....	22
4.1 Creating a DDM Instance.....	22
4.2 Node Groups.....	23
4.2.1 Group Introduction.....	23
4.2.2 Group Management.....	24
4.3 Changing Class of a DDM Instance.....	25
4.4 Scaling Out a DDM Instance.....	26
4.5 Scaling In a DDM Instance.....	26
4.6 Configuring Access Control.....	27
4.7 Restarting a DDM Instance.....	28
4.8 Deleting a DDM Instance.....	28

4.9 Reloading Table Data.....	29
4.10 Parameter Management.....	29
4.11 Read/Write Splitting.....	34
4.12 Configuring a Parameter Template.....	34
5 Parameter Template Management.....	36
5.1 Creating a Parameter Template.....	36
5.2 Editing a Parameter Template.....	37
5.3 Comparing Two Parameter Templates.....	38
5.4 Viewing Parameter Change History.....	39
5.5 Replicating a Parameter Template.....	39
5.6 Applying a Parameter Template.....	40
5.7 Viewing Application Records of a Parameter Template.....	41
5.8 Modifying the Description of a Parameter Template.....	41
5.9 Deleting a Parameter Template.....	42
6 Task Center.....	43
7 Schema Management.....	45
7.1 Creating a Schema.....	45
7.2 Exporting Schema Information.....	47
7.3 Importing Schema Information.....	47
7.4 Deleting a Schema.....	48
7.5 Configuring the SQL Blacklist.....	49
7.6 Scaling Out a Schema.....	50
7.6.1 Rebalancing Shards Across All DB Instances.....	51
7.6.2 Doubling Shards and Resharding Data Records.....	52
8 Account Management.....	55
8.1 Creating an Account.....	55
8.2 Modifying an Account.....	57
8.3 Deleting an Account.....	58
8.4 Resetting the Password of an Account.....	59
9 Backup Management.....	61
9.1 Restoring from Current Instance.....	61
9.2 Creating a Consistent Backup.....	62
9.2.1 Creating a Backup.....	62
9.2.2 Restoring from a Backup.....	62
10 Monitoring Management.....	64
10.1 Metrics.....	64
10.2 Viewing Metrics.....	65
11 Auditing.....	68
11.1 Key Operations Recorded by CTS.....	68
11.2 Querying Traces.....	70

12 SQL Syntax.....	71
12.1 Introduction.....	71
12.2 DDL.....	73
12.2.1 Creating a Table.....	74
12.2.2 Sharding Algorithm Overview.....	74
12.2.3 Sharding Algorithms.....	76
12.2.3.1 MOD_HASH.....	76
12.2.3.2 MOD_HASH_CI.....	78
12.2.3.3 UNI_HASH.....	80
12.2.3.4 RIGHT_SHIFT.....	82
12.2.3.5 YYYYMM.....	83
12.2.3.6 YYYYDD.....	85
12.2.3.7 YYYYWEEK.....	86
12.2.3.8 HASH.....	87
12.2.3.9 Range.....	89
12.2.3.10 MM.....	91
12.2.3.11 DD.....	92
12.2.3.12 MMDD.....	93
12.2.3.13 WEEK.....	94
12.3 DML.....	95
12.3.1 INSERT.....	96
12.3.2 REPLACE.....	96
12.3.3 DELETE.....	97
12.3.4 UPDATE.....	97
12.3.5 SELECT.....	97
12.3.6 SELECT JOIN Syntax.....	98
12.3.7 SELECT UNION Syntax.....	99
12.3.8 SELECT Subquery Syntax.....	99
12.4 Functions.....	100
12.5 Unsupported Items.....	103
12.6 Supported SQL Statements.....	104
12.6.1 CHECK TABLE.....	104
12.6.1.1 Checking DDL Consistency of Table Shards in All Logical Tables.....	104
12.6.1.2 Checking DDL Consistency of Table Shards in One Logical Table.....	105
12.6.2 SHOW RULE.....	107
12.6.3 SHOW TOPOLOGY.....	108
12.6.4 SHOW DATA NODE.....	108
12.6.5 TRUNCATE TABLE.....	108
12.6.5.1 HINT-DB.....	108
12.6.5.2 HINT-TABLE.....	109
12.6.5.3 HINT-DB/TABLE.....	109
12.6.6 HINT- ALLOW_ALTER_RERUN.....	109

12.6.7 LOAD DATA.....	109
12.6.8 SHOW PHYSICAL PROCESSLIST.....	111
12.6.9 Customized Hints for Read/Write Isolation.....	111
12.6.10 Setting a Hint to Skip the Cached Execution Plan.....	112
12.7 Global Sequence.....	112
12.7.1 Using NEXTVAL and CURRVAL to Query Global Sequence Numbers.....	114
12.7.2 Using Global Sequences in INSERT or REPLACE Statements.....	116
12.8 Database Management Syntax.....	118
12.9 Advanced SQL Functions.....	119
13 FAQs.....	120
13.1 General Questions.....	120
13.1.1 What High-Reliability Mechanisms Does DDM Provide?.....	120
13.1.2 Does DDM Store Service-Related Data?.....	120
13.1.3 How Do I Select and Configure a Security Group?.....	120
13.2 DDM Usage.....	122
13.2.1 How Can I Rectify a Fault in Accessing DDM by Using the JDBC Driver?.....	122
13.2.2 What Version and Parameters Should I Select?.....	123
13.2.3 Why It Takes So Long Time to Export Data from MySQL Using mysqldump?.....	125
13.2.4 How Should I Handle the Duplicate Primary Key Error Occurring After Data Is Imported into DDM?.....	125
13.2.5 What Should I Do If an Error Message Is Returned After I Specify an Auto-Increment Primary Key?.....	125
13.2.6 How Do I Handle the Error Reported When Parameter Configuration Does Not Time Out?.....	125
13.2.7 Which Should I Delete First, Schema or Associated RDS DB Instances?.....	125
13.2.8 Should I Manually Delete Databases and Accounts Remained in the Associated RDS DB Instances After a Schema Is Deleted?.....	125
13.3 SQL Syntax.....	125
13.3.1 Does DDM Support Cross-Database Access of SQL?.....	126
13.3.2 Does DDM Support Distributed JOIN?.....	126
13.3.3 How Do I Optimize SQL Statements?.....	126
13.3.4 Does DDM Support Forced Data Type Conversion?.....	126
13.3.5 What Should I Do If an Error Is Reported When Multiple Data Records Are Inserted into Batches Using the INSERT Statement?.....	126
13.4 RDS-related Questions.....	126
13.4.1 Is the Name of a Database Table Case-Sensitive?.....	127
13.4.2 What Risky Operations on RDS for MySQL Will Affect DDM?.....	127
13.4.3 How Do I Handle Data with Duplicate Primary Keys in a Table?.....	128
13.4.4 How Can I Query RDS Information by Running Command show full innodb status ?.....	130
13.5 Connection Management.....	130
13.5.1 Can I Connect to DDM Instances from My Local Environment?.....	130
13.5.2 What Should I Do If Garbled Characters Are Displayed When I Connect to DDM Using MySQL?.....	131

1 Service Overview

1.1 Overview

Distributed Database Middleware (DDM) is a MySQL-compatible database middleware service. It uses a storage-compute decoupled architecture to make it easy to scale out compute and storage resources to handle a huge number of concurrent requests.

How DDM Works

DDM is scalable in compute and data storage resources, freeing you from worrying about scalability and O&M issues as your services grow.

Advantages

DDM is highly scalable, easy to use, quick to deploy, and cost-effective and can deliver high performance.

- **Highly scalable**
 - DDM supports automatic horizontal sharding, so you can:
 - Change your instance class within minutes.
 - Add instance nodes without impacts on applications.
 - Add DB instances with minimal downtime.
- **Easy to use**
 - Compatible with MySQL licenses, syntax, and clients.
 - Makes it easy to import data and migrate databases to the cloud.
 - Splits read and write requests with no need to modify service code.
- **Quick to deploy**

DDM instances are easy to deploy online. This shortens project cycles and helps migrate your workloads to the cloud. You do not need to purchase, deploy, or configure a DDM instance as you do for a self-built database.
- **Cost-effective**

DDM provides stable performance, comprehensive O&M, and excellent technical support. It is more cost-effective than open-source products. Different instance classes are available to meet your needs.

1.2 Basic Concepts

DDM Instance

A DDM instance is the smallest management unit of DDM. Each DDM instance is an independent database. You can create or modify a DDM instance using the DDM console or API. Then create multiple schemas in the DDM instance and associate each schema with multiple MySQL DB instances. Each MySQL DB instance can be accessed independently.

NOTE

MySQL DB instances in this document include RDS for MySQL DB instances and GaussDB(for MySQL) DB instances.

VPC

A Virtual Private Cloud (VPC) is a private and isolated virtual network. You can configure IP address ranges, subnets, and security groups, assign EIPs, and allocate bandwidth for DDM instances.

Subnet

A subnet is a range of IP addresses, a logical subdivision of an IP network. Subnets are created for a VPC where you will place your DDM instances. Every subnet is defined by a unique CIDR block which cannot be modified once the subnet is created.

Security Group

A security group is a collection of rules for ECSs that have the same security protection requirements and are mutually trusted. After a security group is created, you can add different access rules to the security group, and these rules will apply to all ECSs added to this security group.

Your account automatically comes with a security group by default. The default security group allows all outbound traffic and denies all inbound traffic. Your ECSs in this security group can communicate with each other without the need to add rules.

Parameter Template

A parameter template acts as a container for configuration values that can be applied to one or more DDM instances. If you want to use your own parameter template, you only need to create a custom parameter template and select it when creating a DDM instance. You can also apply the parameter template to an existing DDM instance.

EIP

The Elastic IP (EIP) service provides independent public IP addresses and bandwidth for Internet access. EIPs can be bound to and unbound from DDM instances.

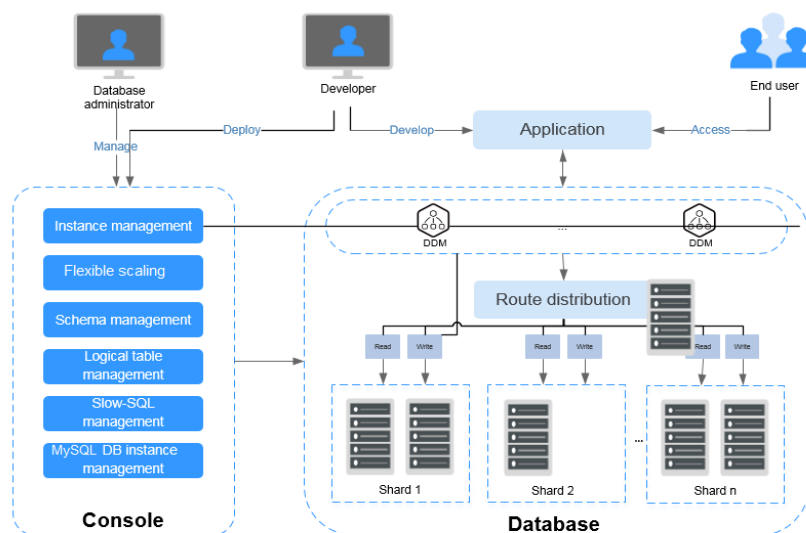
Region and Endpoint

1.3 Product Architecture

As a database middleware, DDM manages underlying database storage engines in clusters.

Using a DDM instance is as if you are still using a single-node database no matter how many database shards there are. You can perform database O&M on the DDM console, and log in to a database using the Java database connectivity (JDBC) driver or a MySQL client to read and write data.

Figure 1-1 DDM architecture



1.4 Functions

DDM supports a variety of functions, including horizontal sharding, flexible scaling, high SQL compatibility, read/write splitting, and global sequence.

Table 1-1 DDM main functions

Function	Description
Horizontal sharding	Select a sharding key when creating a distributed database. DDM will generate a sharding rule and horizontally shard data.

Function	Description
Scaling	<p>DDM supports both compute and storage scaling. You can scale up or out a DDM instance dynamically with zero downtime. Compute scaling is undetectable to your applications, and storage scaling minimizes service interruption to seconds.</p> <p>Storage scaling can be actualized with two methods: rebalancing or resharding.</p> <ul style="list-style-type: none"> • Rebalancing keeps the total database shards unchanged. • Resharding doubles the number of database shards.
Distributed transactions	<p>DDM processes three types of transactions, including single-shard, FREE, and Extended Architecture (XA).</p> <ul style="list-style-type: none"> • Single-shard: Transactions cannot be committed across shards. • FREE: Transactions are committed across shards. A transaction is not rolled back when it fails to be committed by any shard, causing data inconsistency. • XA: Transactions are committed in two phases. If a transaction fails to be committed by any shard, all work done will be rolled back to ensure data consistency.
Data import and export	<p>External data can be imported into DDM to help you migrate databases to the cloud, and DDM instance data can be exported based on requirements.</p>
SQL syntax	<p>DDM is highly compatible with the MySQL licenses and syntaxes.</p>
Read and write splitting	<p>Read and write requests can be split without modifying the application code, and this is totally transparent to applications. You only need to create read replicas for a MySQL DB instance associated with your DDM instance and configure a read policy, so a large number of concurrent requests can read data from those read replicas.</p>
Global sequence	<p>DDM allows you to use globally unique, distributed, and ascending SNs as primary or unique keys or to meet your requirements in specific scenarios.</p>
DDM console	<p>The DDM console enables you to manage and maintain DDM instances and schemas.</p>

Related Services

- VPC

DDM instances are deployed in an isolated VPC and you can configure IP addresses and bandwidth for accessing these DDM instances and use a security group to control access to them.

- ECS
You can access your DDM instance through an ECS.
- Relational Database Service (RDS)
After creating a DDM instance, you can associate it with RDS for MySQL DB instances in the same VPC to obtain separated storage resources.
- Cloud Trace Service (CTS)
CTS records operations on DDM resources for query, audit, and backtrack.
- Elastic Load Balance (ELB)
Elastic Load Balance (ELB) is a traffic distribution control service. It distributes incoming traffic to multiple backend servers based on the forwarding policy to balance workloads. So, it can expand external service capabilities of DDM and eliminate single points of failure (SPOFs) to improve service availability.

1.5 DDM Instance Classes

- General-enhanced DDM instances use Intel® Xeon® Scalable processors. Working in high-performance networks, these DDM instances can offer high and stable computing performance, meeting enterprise-class application requirements.

Table 1-2 DDM instance classes

Class	Architecture	vCPUs	Memory (GB)
General-enhanced	x86	8	16
		16	32
		32	64

1.6 Restrictions

1.6.1 Network Access

DDM usage restrictions are as follows:

- The MySQL DB instances and ECSs running your applications must be in the same VPC as your DDM instance.
- To access DDM from your computer, you need to configure an EIP for your DDM instance and then use the EIP to access the DDM instance.

1.6.2 MySQL DB Instances

Restrictions on MySQL DB instances are as follows:

- DDM can work with only MySQL DB instances of versions 5.7 and 8.0.
- DDM cannot connect to MySQL DB instances using SSL connections.
- Modifying configurations of an associated MySQL DB instance may result in an exception in using your DDM instance. After the modification, click **Synchronize DB Instance Data** on the DDM instance details page to synchronize changes from the DB instance to DDM.

1.6.3 SQL Syntax

DDM is fully compatible with MySQL licenses and syntax, but there are differences between distributed databases and single-node databases.

You can manually submit a transaction to execute SQL statements in batches to add, delete, or modify data records to, from or in data tables.

For example, use the following settings to execute SQL statements on a client:

```
set autocommit=0;
```

```
{sql operations};
```

```
commit;
```

 **NOTE**

This release supports operations only on the connected database using SQL statements. Otherwise, an exception may occur.

Table 1-3 Restrictions on DDL syntax

Item	Restriction
DDL syntax	<ul style="list-style-type: none"> • ALTER statements cannot be used to modify databases or dbpartition/tbpartition columns. • TEMPORARY sharded and broadcast tables cannot be created. • Sharded and broadcast tables cannot be created from another table. • Only unsharded tables support foreign keys. Sharded and broadcast tables do not support such keys. • Unsharded tables can be created only from another unsharded or broadcast table. • CREATE TABLE Syntax and DROP TABLE Syntax do not support annotations and are not restricted by the sql_execute_timeout parameter.

Table 1-4 Syntax restrictions on DML

Item	Restriction
DELETE statement	<ul style="list-style-type: none"> • PARTITION clauses are not supported. • Subqueries are not supported.

Item	Restriction
UPDATE statement	<ul style="list-style-type: none"> • PARTITION clauses are not supported. • Subqueries are not supported.

Table 1-5 Restrictions on database administration statements

Item	Restriction
Database administration statements	<ul style="list-style-type: none"> • SET syntax cannot be used to modify global variables. • SHOW TRIGGERS Syntax is not supported.

Table 1-6 Restrictions on advanced SQL functions

Item	Restriction
Advanced SQL functions	<ul style="list-style-type: none"> • Customized data types and functions are not supported. • Views, stored procedures, triggers, and cursors are not supported. • Compound statements such as BEGIN...END, LOOP...END LOOP, REPEAT...UNTIL...END REPEAT, and WHILE...DO...END WHILE are not supported. • Process control statements such as IF and WHILE are not supported.

1.7 Regions and AZs

Concepts

The combination of a region and an availability zone (AZ) identifies the location of a data center. You can create resources in a specific AZ in a region.

Selecting an AZ

When determining whether to deploy resources in the same AZ, consider your applications' requirements on disaster recovery (DR) and network latency.

- For high DR capability, deploy resources in different AZs in the same region.
- For low network latency, deploy resources in the same AZ.

1.8 Application Scenarios

DDM is an OLTP-oriented service for accessing distributed relational databases across a variety of industries.

It is especially suitable for applications requiring high-concurrency access to large volumes of data. Typical application scenarios are as follows:

- **Internet**

E-commerce, finance, O2O, retail, and social networking applications usually face challenges such as large user base, frequent marketing events, and slow response of core transactional systems. DDM can scale compute and storage resources to improve database processing of high-concurrency transactions and ensure fast access to data.

- **IoT**

In industrial monitoring, remote control, smart city, smart home, and Internet of Vehicles (IoV) scenarios, a large number of sensors and monitoring devices frequently collect data and generate huge amounts of data, which may exceed the storage capability of single-node databases. DDM provides horizontal expansion to help you store massive data at low costs.

- **Other traditional industries**

Government agencies, large-sized enterprises, and banks use commercial solutions to support high-concurrency access to large volumes of data. These solutions are expensive because they need to rely on mid-range computers and high-end storage devices. DDM, deployed in clusters with common ECSs, provides cost-efficient database solutions with the same or even higher performance than traditional commercial database solutions.

2 Getting Started

2.1 Overview

Scenarios

This section uses an RDS DB instance as an example to describe how to use DDM.

Process of Using DDM

Step 1: Log In to the DDM Console

Step 2: Create a DDM Instance

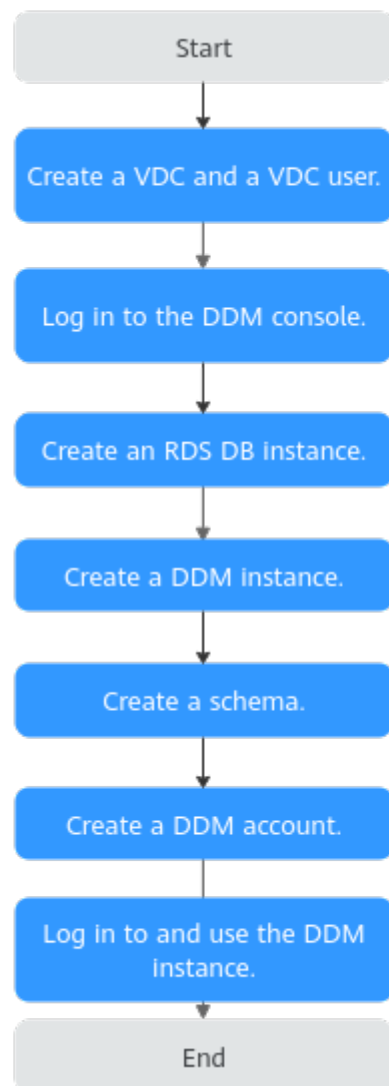
Step 3: Create an RDS DB Instance

Step 4: Create a Schema

Step 5: Create a DDM Account


Step 6: Log In to the DDM Instance or Schema

Figure 2-1 Flowchart for using DDM



2.2 Step 1: Log In to the DDM Console

Procedure

- Step 1** Log in to the management console.
 - Step 2** Click  in the upper left corner and select the required region.
 - Step 3** Click **Service List** and choose **Database > Distributed Database Middleware**.
- End

2.3 Step 2: Create a DDM Instance

Procedure

- Step 1** On the RDS console, click **Service List** and choose **Database > Distributed Database Middleware**.
- Step 2** On the displayed page, in the upper right corner, click **Create DDM Instance**.
- Step 3** On the displayed page, configure the required parameters.

Table 2-1 Required parameters

Parameter	Description
AZ	<p>Availability zone where the DDM instance is deployed.</p> <p>Currently, some regions support cross-AZ deployment to enhance availability of DDM instances.</p> <p>You can select cross-AZ deployment for your DDM instance. In this case, instance nodes will be deployed in different AZs.</p> <p>NOTE Deploy your application, DDM instance, and required RDS DB instances in the same AZ to reduce network latency. Cross-AZ deployment may increase network latency.</p>
Instance Name	<p>Name of the DDM instance, which:</p> <ul style="list-style-type: none"> • Cannot be left blank. • Must start with a letter. • Must be 4 to 64 characters long. • Can contain only letters, digits, and hyphens (-).
Instance Class	<p>Class of the DDM instance. You can select General-enhanced or Kunpeng general computing-plus.</p> <p>NOTE Estimate compute and storage requirements of your applications based on your service type and scale before you create a DDM instance, and then select an appropriate class so that the CPU and memory specifications of your DDM instance can better meet your needs.</p>
Instance Nodes	<p>Number of nodes in a DDM instance. Up to 32 nodes are supported.</p>

Parameter	Description
VPC	VPC that the DDM instance belongs to. This VPC isolates networks for different services. It allows you to manage and configure private networks, simplifying network management. Click View VPC to show more details and security group rules. NOTE The DDM instance should be in the same VPC as the required RDS DB instance. To ensure network connectivity, the DDM instance you purchased must be in the same VPC as your applications and RDS DB instances.
Subnet	Name and IP address range of the subnet
Security Group	Select an existing security group. You are advised to select the same security group for your DDM instance, application, and RDS MySQL DB instances so that they can communicate with each other. If different security groups are selected, add security group rules to enable network access.
Parameter Template	Select an existing parameter template. You can also click View Parameter Template to set parameters on the displayed page.

Step 4 After the configuration is complete, click **Next** at the bottom of the page.

Step 5 Confirm the configurations and click **Submit**.

----End

Creating a DDM Instance

2.4 Step 3: Create an RDS DB Instance

Procedure

Step 1 On the DDM console, click **Service List** and choose **Database > Relational Database Service**.

Step 2 On the displayed page, click **Create DB Instance** in the upper right corner, specify the required parameters, and click **Create Now**.

Step 3 On the displayed page, confirm the configurations.

Step 4 Confirm the configurations and click **Submit**.

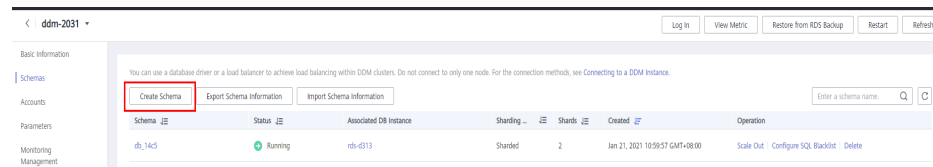
Step 5 Wait 1 to 3 minutes for the RDS DB instance to be created.

----End

2.5 Step 4: Create a Schema

You can create a schema on the **Instances** or **Schemas** page. This section uses the **Instances** page as an example to describe how to create a schema.

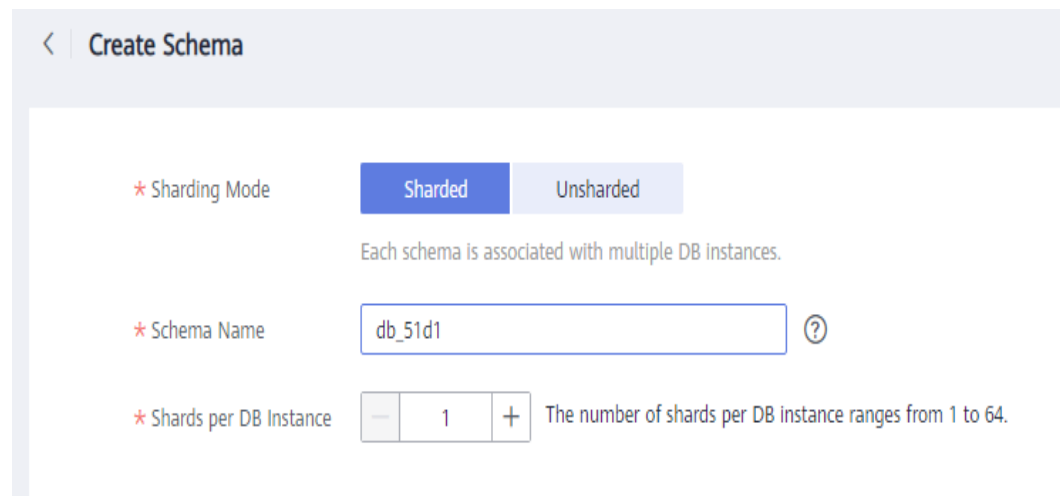
Figure 2-2 Schemas page



Procedure

- Step 1** Log in to the DDM console, and in the navigation pane, choose **Instances**. In the instance list, locate the required DDM instance and click **Create Schema** in the **Operation** column.
- Step 2** On the displayed page, specify a sharding mode, enter a schema name, select the number of shards per DB instance, and click **Next: Select DB Instance**.

Figure 2-3 Creating a schema



NOTE

Sharding mode:

- Sharded: One schema corresponds to multiple MySQL DB instances.
- Unsharded: One schema corresponds only to one MySQL DB instance, and only one database shard is created for that instance.

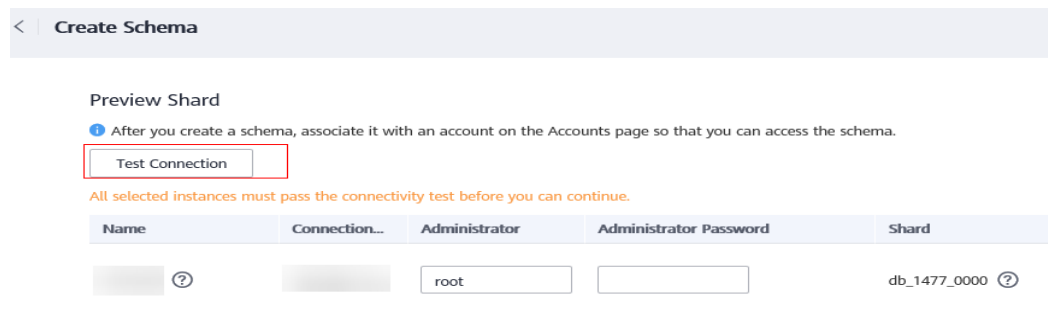
The schema name contains 2 to 24 characters and must start with a letter. Only letters, digits, and underscores (_) are allowed.

The shards per DB instance ranges from 1 to 64.

- Step 3** On the **RDS DB Instances** tab page, select the required RDS DB instances. You can choose to configure the instances as the unsharded table storage nodes. Click **Preview**.

- Step 4** On the displayed page, enter the administrator password and click **Test Connection**. After all the test is successful, click **Finish**.

Figure 2-4 Test Connection



----End

2.6 Step 5: Create a DDM Account

Procedure

- Step 1** Log in to the DDM console, in the instance list, locate the required DDM instance and click its name.
- Step 2** In the navigation pane, choose **Accounts**.
- Step 3** On the displayed page, click **Create Account** and configure the required parameters.

Table 2-2 Required parameters

Parameter	Description
Username	Account name. The name: <ul style="list-style-type: none"> • Must start with a letter. • Is case-insensitive. • Must be 6 to 32 characters long. • Contains only letters, digits, and underscores (_).
Password	Password of the account. The password: <ul style="list-style-type: none"> • Can contain 8 to 32 characters. • Must contain at least three of the following character types: letters, digits, spaces, and special characters <code>^~!@#%\$%^&*()-_+=\ []{};:\\"<.>/?</code> • Cannot be a weak password. It cannot be overly simple and easily guessed.
Confirm Password	-

Parameter	Description
Schema	Schema to be associated with the DDM account. You can select an existing schema from the drop-down list. Only the associated schemas can be accessed using the account.
Permissions	Permissions of a DDM account, including: <ul style="list-style-type: none"> • Basic Permissions: CREATE, DROP, ALTER, INDEX, INSERT, DELETE, UPDATE, and SELECT • Extended Permissions: Table query, Table delete, and Table update • When configuring permissions: Select basic permissions based on your needs. When you select a basic permission, the corresponding extended permissions are selected automatically. Not each basic permission corresponds to an extended permission. The extended permissions for each basic permission are as follows: <ul style="list-style-type: none"> - Table query corresponds to SELECT. - Table delete corresponds to DELETE. - Table update corresponds to UPDATE.
Description	Description of the account, which cannot exceed 256 characters.

Step 4 Click **OK**.

----End

2.7 Step 6: Log In to the DDM Instance or Schema

After you create a DDM instance, you can log in to it using a client such as Navicat, or connect to the required schema in the instance using the CLI or JDBC driver.

This section describes how to log in to a DDM instance or a schema.

Preparations

Before login to your DDM instance or schema, you have to obtain the connection address of your DDM instance or schema.

Obtaining the Instance Connection Address

Step 1 Log in to the management console.

Step 2 Hover on the left menu to display **Service List** and choose **Database > Distributed Database Middleware**.

- Step 3** In the navigation pane, choose **Instances**. In the instance list, locate the required instance and click its name.
- Step 4** In the **Network Information** area, view the connection address.

Network Information

Connection Address	192.168.2.119:5065
VPC	vpc-74f8
Subnet	subnet-74f8(192.168.1.0/24)
Security Group	ces-test

NOTE

If you enter an incorrect password for five times or more when attempting to access your DDM instance, the system will be locked for 20 minutes.

----End

Obtaining the Schema Connection Address

- Step 1** Log in to the management console.
- Step 2** Hover on the left menu to display **Service List** and choose **Database > Distributed Database Middleware**.
- Step 3** In the navigation pane, choose **Instances**. In the instance list, locate the required DDM instance and click its name.
- Step 4** In the navigation pane, choose **Schemas**.
- Step 5** In the schema list, locate the required schema and click its name.
- Step 6** At the **Connection Address** area, view the CLI and JDBC connection addresses.

Figure 2-5 Schema connection address



 **NOTE**

If you enter an incorrect password for five times or more when attempting to access your DDM instance, the system will be locked for 20 minutes.

----End

Connection Methods

For details about method 1, see [Using Navicat to Log In to a DDM Instance](#).

For details about method 2, see [Using the MySQL CLI to Log In to a Schema](#).

For details about method 3, see [Using a JDBC Driver to Log In to a Schema](#).

For details about method 4, see [Log In to a DDM Instance on the DDM Console](#).

 **NOTE**

1. For security purposes, use an ECS in the same VPC as your DDM instance.
2. Ensure that a MySQL client has been installed on the required ECS or the MySQL connection driver has been configured.
3. Before you log in to a DDM instance, configure its information on the client or connection driver.

Using Navicat to Log In to a DDM Instance

Step 1 Log in to the DDM console, locate the required DDM instance, and click its name.

Step 2 In the **Instance Information** area, click **Bind** and select an EIP available.

Step 3 In the left pane, click the VPC icon and choose **Access Control > Security Groups**.

Step 4 On the **Security Groups** page, locate the required security group and click **Manage Rule** in the **Operation** column. On the displayed page, click **Add Rule**. Configure the security group rule as needed and click **OK**.

 **NOTE**

After binding an EIP to your DDM instance, set strict inbound and outbound rules for the security group to enhance database security.

Step 5 Open Navicat and click **Connection**. In the displayed dialog box, enter the host IP address (EIP), username, and password (DDM account).

Step 6 Click **Test Connection**. If a message is returned indicating that the connection is successful, click **OK**. The connection will succeed 1 to 2 minutes later. If the connection fails, the failure cause is displayed. Modify the required information and try again.

----End

 **NOTE**

Using Navicat to access a DDM instance is similar to using other visualized MySQL tools such as MySQL Workbench. Therefore, the procedure of using other visualized MySQL tools to connect to a DDM instance has been omitted.

Using the MySQL CLI to Log In to a Schema

Step 1 Log in to the required ECS, open the CLI, and run the following command:

```
mysql -h ${DDM_SERVER_ADDRESS} -P${DDM_SERVER_PORT} -u${DDM_USER} -p [-D${DDM_DBNAME}]
[--default-character-set=utf8][--default_auth=mysql_native_password]
```

Table 2-3 Parameter description

Example Parameter	Description	Example Value
DDM_SERVER_ADDRES S	IP address of the DDM instance	192.168.0. 200
DDM_SERVER_PORT	Connection port of the DDM instance	5066
DDM_USER	Account of the DDM instance	dbuser01
DDM_DBNAME	(Optional) Name of the target schema in the DDM instance	-
default-character- set=utf8	(Optional) Select character set UTF-8 for encoding. Configure this parameter if garbled characters are displayed during parsing due to inconsistency between MySQL connection code and actually used code.	-
default_auth=mysql_nat ive_password	The password authentication plug-in is used by default.	-

 **NOTE**

If you use the MySQL 8.0 client, set **default_auth** to **mysql_native_password**.

Step 2 View the command output. The following is an example output of running a MySQL command in the Windows CLI.

```
C:\Users\testDDM>mysql -h192.168.0.200 -P5066 -Ddb_5133 -udbuser01 -p
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.6.29

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

----End

Using a JDBC Driver to Log In to a Schema

Step 1 Load the required JDBC driver.

```
Class.forName("com.mysql.jdbc.Driver");
```

NOTE

JDBC drivers 5.1.35 to 5.1.45 are recommended.

Step 2 Create a database connection.

```
String username = "dbuser01" ;  
String password = "xxxxxx" ;  
String url = "jdbc:mysql://192.168.0.200:5066/db_5133";  
Connection con = DriverManager.getConnection(url , username , password);
```

Step 3 Create a Statement object.

```
Statement stmt = con.createStatement();
```

Step 4 Execute the required SQL statement.

```
ResultSet rs = stmt.executeQuery("select now() as Systemtime");  
con.close();
```

Step 5 (Optional) Optimize code as needed.

```
loadBalanceAutoCommitStatementThreshold=5&loadBalanceHostRemovalGracePeriod=15000&loadBalance  
BlacklistTimeout=60000&loadBalancePingTimeout=5000&retriesAllDown=10&connectTimeout=10000";
```

NOTE

- Parameters **loadBalanceAutoCommitStatementThreshold** and **retriesAllDown** must be configured based on the example in [Step 5](#). Otherwise, an infinite loop may occur during the connection switchover, resulting in stack overflow.
- **loadBalanceAutoCommitStatementThreshold**: defines the number of matching statements which will trigger the driver to potentially swap physical server connections.
- **loadBalanceHostRemovalGracePeriod**: indicates the grace period to wait for a host being removed from a load-balanced connection, to be released when it is the active host.
- **loadBalanceBlacklistTimeout**: indicates the time in milliseconds between checks of servers which are unavailable, by controlling how long a server lives in the global blacklist.
- **loadBalancePingTimeout**: indicates the time in milliseconds that the connection will wait for a response to a ping operation when you set **loadBalanceValidateConnectionOnSwapServer** to **true**.
- **retriesAllDown**: indicates the maximum number of connection attempts before an exception is thrown when a valid host is searched. `SQLException` will be returned if the threshold of retries is reached with no valid connections obtained.
- **connectTimeout**: indicates the maximum amount of time in milliseconds that the JDBC driver is willing to wait to set up a socket connection. **0** indicates that the connection does not time out. Only JDK-1.4 or later is supported. The default value **0**.

----End

Log In to a DDM Instance on the DDM Console

Step 1 Log in to the DDM console.

Step 2 In the navigation pane, choose **Instances**.

Step 3 In the instance list, locate the required instance and click **Log In** in the **Operation** column.

On the displayed page, enter the required username and password.

Step 4 On the displayed page, enter username and password of the DDM account.

Step 5 Click **Test Connection**.

Step 6 (Optional) Enable **Collect Metadata Periodically** and **Show Executed SQL Statements**.

Step 7 Ensure that all settings are correct and click **Log In**.

----End

3 Function Overview

Distributed Database Middleware (DDM) is a MySQL-compatible distributed relational database. It uses a storage-compute decoupled architecture that makes it easy to scale out compute and storage resources to handle millions of concurrent requests.

Table 3-1 lists the functions supported by DDM.

Table 3-1 DDM functions

Category	Function
Instances	Creating, deleting, restarting, renewing, and unsubscribing from a DDM instance, and changing class of a DDM instance. For details, see Instance Management .
Backups	Backing up a DDM instance, restoring a DDM instance from a backup, and restoring data from the current DDM instance. For details, see Backup Management .
Parameter templates	Creating, editing, replicating, and applying a parameter template, and comparing two parameter templates. For details, see Parameter Template Management .
Task center	Enabling you to view progress and statuses of asynchronous tasks submitted on the console. For details, see Task Center .
Schemas	Creating, exporting, importing, and deleting schemas. For details, see Schema Management .
Schema scaling	Scaling out a schema by rebalancing or resharding. For details, see Scaling Out a Schema .
Accounts	Creating, modifying, and deleting a DDM account, and resetting its password. For details, see Account Management .
Monitoring	Providing metrics and methods of viewing metrics. For details, see Monitoring Management .
SQL syntax	Describing DDL, DML, global sequence, SQL statements, and sharding algorithms. For details, see SQL Syntax .

4 Instance Management

4.1 Creating a DDM Instance

Prerequisites

You have logged in to the DDM console.

Procedure

Step 1 On the displayed page, in the upper right corner, click **Create DDM Instance**.

Step 2 On the displayed page, configure the required parameters.

Table 4-1 Parameter description

Parameter	Description
AZ	Availability zone where the DDM instance is deployed. Currently, some regions support cross-AZ deployment to enhance availability of DDM instances. When you create a DDM instance, you can select one or more AZs and then the DDM instance will be deployed across different AZs. NOTE Deploy your application, DDM instance, and required RDS DB instances in the same AZ to reduce network latency. Cross-AZ deployment may increase network latency.
Instance Name	Name of the DDM instance, which: <ul style="list-style-type: none">• Cannot be left blank.• Must start with a letter.• Must be 4 to 64 characters long.• Contain only letters, digits, and hyphens (-).

Parameter	Description
Instance Class	Class of the DDM instance. You can select general-enhanced (x86) or Kunpeng general computing-plus (Arm). NOTE Estimate compute and storage requirements of your application and select an appropriate instance class based on your service type and scale so that the DDM instance you will create can better meet your needs. The instance class includes vCPUs and memory.
Instance Nodes	Number of nodes for deploying the DDM instance. You can apply for up to 32 nodes at a time.
VPC	VPC that the DDM instance belongs to. This VPC isolates networks for different services. It allows you to manage and configure private networks, simplifying network management. Click View VPC to show more details and security group rules. NOTE The DDM instance should be in the same VPC as the required MySQL DB instances.
Security Group	Select an existing security group. The same security group is recommended for your DDM instance, application, and required MySQL DB instances so that network access is not restricted. If different security groups are selected, add security group access rules to enable such network access.
Parameter Template	A parameter template acts as a container for engine configuration values that can be applied to one or more DDM instances. You can modify parameters in a parameter template to manage configurations of the DDM instance that the template applies to.

Step 3 After the configuration is complete, click Create Now at the bottom of the page.

Step 4 Confirm the configurations and click **Submit**.

----End

4.2 Node Groups

4.2.1 Group Introduction

Overview

For a DDM instance, its nodes can be divided into different groups to expand its service capabilities. There are two types of node groups, read-only and read/write, and they are responsible for handling read-only and read/write requests, respectively.

Read-only and read/write groups use the same data. When there are a large number of concurrent requests, read-only groups handle complex queries or

extract data offline from the primary RDS DB instance or its read replicas to reduce response time and provide faster access.

It is easy to use node groups without the need of establishing complex links or synchronizing data.

 **NOTE**

The node group function is only available for whitelisted users. To use this function, submit a service ticket first.

4.2.2 Group Management

Procedure

Step 1 Log in to the DDM console and choose **Instances** in the navigation pane. In the instance list, locate the required instance and click its name.

Step 2 On the **Basic Information** page, view the default group, named **group-default**. The nodes you have created are classified into this group.

NOTICE

Only DDM instances with the kernel version of 2.4.1.2 or later support node groups.

Step 3 Click **Create Group** to create a read/write or read-only group.

 **NOTE**

- One DDM instance supports a maximum of 32 nodes that can be divided into multiple read/write and read-only groups. Each group contains at least 2 nodes.
- One node belongs to only one group, and its group cannot be changed once determined. Nodes in the same group must be of the same class.

Step 4 On the **Create Group** page, select the required role and class, specify the quantity of new nodes, and click **Next**.

 **NOTE**

Read-only groups handle only read requests.

Step 5 Confirm the information and click **Submit**.

Step 6 After the creation is complete, view the group information at the **Node Information** area on the **Basic Information** page. After a group is created, a connection address is configured for it.

 **NOTE**

Deleting groups is not supported for yearly/monthly DDM instances.

To delete a group, locate the group that you want to delete and click **Delete**. The corresponding connection address becomes invalid once the group is deleted. This may affect your services. Retain at least one read/write group.

----End

4.3 Changing Class of a DDM Instance

Prerequisites

- You have logged in to the DDM console.
- The DDM instance is in the **Running** state.

NOTICE

Change instance class during off-peak hours because services will be temporarily interrupted during class changing.

Procedure

NOTICE

If the read-only group function is not used, perform step **Step 1** to change the class.

If the read-only group function is used, go to step **Step 2** to change the class.

- Step 1** In the navigation pane, choose **Instances**; in the instance list, locate the DDM instance whose class you want to change, and choose **More > Change Instance Class** in the **Operation** column.
- Step 2** On the DDM console, choose **Instances** in the navigation pane. In the instance list, locate the DDM instance whose class you want to change, and click its name. On the displayed **Basic Information** page, click **Change Instance Class** at the **Node Information** area.
- Step 3** On the displayed page, view the current instance configuration and select the required instance class in the lower part of the page.
- Step 4** Ensure that the settings are correct and click **Next**.
- Step 5** Confirm the configurations and click **Submit**.
- Step 6** Switch back to the instance list and check whether the status of the instance changes to **Changing class**.

NOTE

- Once the change operation is performed, it cannot be undone. To change the class again, submit another request after the class change is complete.
- Instance class can be upgraded or downgraded.

----End

4.4 Scaling Out a DDM Instance

Scenarios

As service data increases, you can scale out a DDM instance by adding nodes to improve service stability.

 **NOTE**

- Scale out your DDM instance during off-peak hours.
- Make sure that the associated DB instances are normal and not undergoing other operations.

Procedure

NOTICE

If the read-only group function is not used, perform [Step 1](#) to add nodes.

If the read-only group function is used, perform [Step 2](#) to add nodes.

- Step 1** Log in to the DDM console; in the instance list, locate the DDM instance that you want to scale out, and choose **More > Scale Out** in the **Operation** column.
- Step 2** On the DDM console, choose **Instances** in the navigation pane. In the instance list, locate the DDM instance that you want to scale out, and click its name. On the displayed **Basic Information** page, click **Scale Out** at the **Node Information** area.
- Step 3** On the displayed page, view the current instance configuration, select the required AZ, and specify the number of new nodes.

 **NOTE**

Each DDM instance supports up to 32 nodes.

- Step 4** Click **Next**.
- Step 5** On the displayed page, click **Submit** if all configurations are correct.

 **NOTE**

After the scaling task is submitted, the instance status changes to **Scaling out**.

----End

4.5 Scaling In a DDM Instance

Scenarios

This section describes how to scale in a DDM instance as service data volume decreases.

 NOTE

- Scale in your DDM instance during off-peak hours.
- Make sure that the associated DB instance is normal and not undergoing other operations.

Procedure

NOTICE

If the read-only group function is not used, perform [Step 1](#) to scale in your DDM instance.

If the read-only group function is used, perform [Step 2](#) to scale in your DDM instance.

- Step 1** Log in to the DDM console, in the instance list, locate the DDM instance that you want to scale in, and choose **More > Scale In** in the **Operation** column.
- Step 2** On the DDM console, choose **Instances** in the navigation pane. In the instance list, locate the DDM instance that you want to scale out, and click its name. On the displayed **Basic Information** page, click **Scale In** at the **Node Information** area.
- Step 3** On the displayed page, view the current instance configuration and specify the number of nodes to be removed.

 NOTE

At least one node should be left for a DDM instance.

- Step 4** Click **Next**.
- Step 5** On the displayed page, click **Submit** if all configurations are correct.

 NOTE

After the scaling task is submitted, the instance status changes to **Scaling in**.

----End

4.6 Configuring Access Control

Scenarios

After load balancing is enabled on DDM, the IP addresses that can access DDM are not restricted by default. You need to use the access control function to control the access. The security group is still valid for access requests directly sent to DDM nodes.

Procedure

- Step 1** On the DDM console, choose **Instances** in the navigation pane. In the instance list, locate the DDM instance that you want to configure access control, and click its

name. On the displayed **Basic Information** page, locate the **Access Control** switch at the **Node Information** area.

Step 2 Click **Configure**.

Step 3 In the **Configure Access Control** dialog box, specify **Access Policy**, enter the required IP addresses, and click **OK**.

----End

4.7 Restarting a DDM Instance

Prerequisites

- You have logged in to the DDM console.
- The DDM instance is in the **Running** state.

NOTICE

The DDM instance is not available during restart, and the restart operation cannot be undone. Exercise caution when performing this operation.

Procedure

Step 1 In the instance list, locate the DDM instance that you want to restart and choose **More > Restart** in the **Operation** column.

Step 2 In the displayed dialog box, click **Yes**.

Step 3 Wait until the instance is restarted.

----End

4.8 Deleting a DDM Instance

Prerequisites

You have logged in to the DDM console.

NOTICE

Deleted DDM instances cannot be recovered. Exercise caution when performing this operation.

Procedure

Step 1 In the instance list, locate the DDM instance that you want to delete and choose **More > Delete** in the **Operation** column.

Step 2 In the displayed dialog box, click **Yes**.

 **NOTE**

- To delete data stored on associated MySQL DB instances, select **Delete data on associated DB instances**.

----End

4.9 Reloading Table Data

Prerequisites

You have logged in to the DDM console.

Data has been migrated to a new DDM instance using DRS.

Scenarios

If you want to deploy a DDM instance across regions for DR, use DRS to migrate service data and then reload table data after the migration is complete so that DDM can detect where logical table information is located.

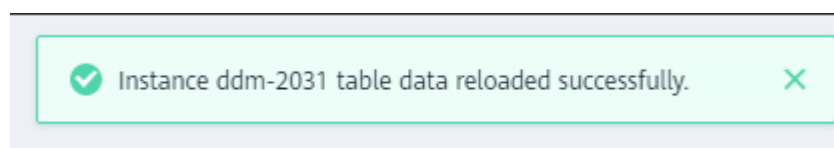
Procedure

Step 1 In the instance list, locate the DDM instance whose table data you want to reload.

Step 2 Choose **More > Reload Table Data** in the **Operation** column.

A message is displayed, indicating that table data of instance *XXX* has been reloaded.

Figure 4-1 Example message



----End

4.10 Parameter Management

Scenarios

Configure parameters of a DDM instance based on your needs to keep the instance running well.

Prerequisites

There is a DDM instance available and running properly.

Procedure

- Step 1** Log in to the DDM console.
- Step 2** In the navigation pane, choose **Instances**.
- Step 3** In the instance list, locate the DDM instance whose parameters you want to configure and click its name.
- Step 4** In the left pane, click **Parameters** and modify parameter values as needed.

Figure 4-2 Parameter management page

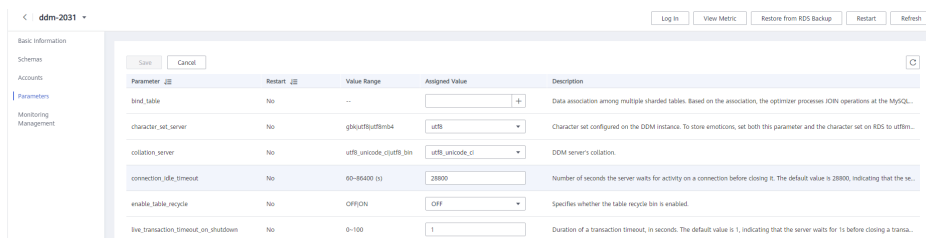


Table 4-2 Parameters of a DDM instance

Parameter	Default Value	Value Range	Description
bind_table	-	The value should be in format [[tb.col1,tb2.col2], [tb.col2,tb3.col1],...] . <i>tb.col1,tb2.col2</i> indicates a table name.column name pair, and the value may contain multiple pairs. The version should be: DDM 2.3.2.7 or later	Data association among multiple sharded tables. Based on the association, the optimizer processes JOIN operations at the MySQL layer based on these associations. For details about parameter examples, see the description below the table.

Parameter	Default Value	Value Range	Description
max_connections	20000	10-40000	<p>Maximum number of concurrent client connections allowed per DDM instance.</p> <p>This value depends on specifications and processing capabilities of the target RDS MySQL DB instance. Too many connections may cause connection waiting, affecting performance. The consumption of DDM connections is related to the number of shards and SQL design.</p> <p>For example, If a SQL statement contains a sharding key, each DDM connection consumes one RDS MySQL instance connection. If the SQL statement contains no sharding key and the number of shards is N, N RDS MySQL connections are consumed.</p> <p>If SQL design is appropriate and DDM and RDS processing capabilities are good enough, you can set this parameter to a value slightly smaller than "backend RDS MySQL DB instances" x "maximum connections supported by each RDS MySQL DB instance".</p> <p>Carry out pressure tests on your services and then select a proper value.</p>
connection_idle_timeout	28800	60-86400 (s)	Number of seconds the server waits for activity on a connection before closing it.
long_query_time	1	0.1-3600 (s)	Minimum duration of a query to be logged as slow.
sql_execute_timeout	28800	100-28800 (s)	Number of seconds to wait for a SQL statement to execute before timing out.
connection_failed_threshold	50	1-10000	Number of consecutive failed connection attempts allowed before your account and IP address are locked.
connection_failed_delay	1200	1-86400	Number of seconds before a locked account and IP address are unlocked.

Parameter	Default Value	Value Range	Description
max_allowed_packet	16777216	1024–1073741824	Maximum size of one packet or any generated intermediate string. The packet message buffer is initialized to net_buffer_length bytes, but can grow up to max_allowed_packet bytes when needed. The value is small by default. You need to set a larger value to catch large (and possibly incorrect) packets. The value must be a multiple of 1024.
character_set_server	utf8	gbk, utf8, utf8mb4	Character set configured on the DDM instance. To store emoticons, set both this parameter and the character set on the RDS MySQL DB instance to utf8mb4 . NOTE The character sets of the DDM instance and RDS MySQL DB instance must be consistent, including values of parameters character_set_client , character_set_connection , character_set_database , character_set_results , and character_set_server .
collation_server	The parameter value range is based on the character_set_server value.	The parameter value range is based on the character_set_server value.	Collation on the DDM server. You need to configure this parameter after configuring character_set_server as follows: <ul style="list-style-type: none"> • If character_set_server is set to gbk, collation_server can be set to gbk_chinese_ci or gbk_bin. • If character_set_server is set to utf8, collation_server can be set to utf8_unicode_ci or utf8_bin. • If character_set_server is set to utf8mb4, collation_server can be set to utf8mb4_unicode_ci or utf8mb4_bin.
sql_audit	OFF	OFF or ON	Whether to enable or disable SQL audit.

Parameter	Default Value	Value Range	Description
dmm_instance_type	-	SINGLE,MASTER,SLAVE	SINGLE: indicates that the DDM instance is a single instance. SLAVE: indicates that the DDM instance is the standby instance for across-region DR. MASTER: indicates that the DDM instance is the primary instance for across-region DR.
transaction_policy	XA	XA, FREE, NO_DTX	XA: XA transaction that attempts to ensure atomicity and visibility. FREE: Best-effort commit transaction that allows data to be written to multiple shards, without impacting performance. This setting does not ensure atomicity. NO_DTX: With this setting, single-shard transactions are performed.

By default, DDM only allows you to modify the preceding instance parameters. If you need to modify other parameters for some special scenarios, such as data migration, contact customer service.

Parameter configuration examples:

Figure 4-3 Result if `bind_table` is not used


```
mysql> explain select * from bill b join orderbl o on b.cid = o.orderid where b.id > 2;
+-----+
| Logical Execution Plan |
+-----+
|
| LookUpJoin(condition='bill.cid = orderbl.orderid', joinType=inner)
|
+-----+
|
| ExtTableScan(table='db_4581 [0-7].bill', tableType=SHARDING, shardCount=8, pushdownSql='SELECT id, cid, sid, amount FROM db_4581.bill WHERE id > 2')
|
| ExtTableScan(table='db_4581 [0-7].orderbl', tableType=SHARDING, shardCount=8, pushdownSql='SELECT orderid, ordermaster, totalamount, createtime, updateTime, payTime, payid, unsubscribeid, unsubscribeamount, unsubscribeTime, unsubscribeReason, status, userid, merchantid, commodityid FROM db_4581.orderbl')
|
+-----+
```

Figure 4-4 Result if `bind_table` is used

```
mysql> explain select * from bill b join orderbl o on b.cid = o.orderid where b.id > 2;
+-----+
| Logical Execution Plan |
+-----+
|
| ExtTableScan(schemas=db_4421 [0-15], joinTypeList=INNER, pushdownSql='SELECT bill.id, bill.cid, bill.sid, bill.amount, orderbl.orderid, orderbl.ordermaster, orderbl.totalamount, orderbl.createtime, orderbl.updateTime, orderbl.payTime, orderbl.payid, orderbl.unsubscribeid, orderbl.unsubscribeamount, orderbl.unsubscribeTime, orderbl.unsubscribeReason, orderbl.status, orderbl.userid, orderbl.merchantid, orderbl.commodityid FROM db_4421.bill INNER JOIN db_4421.orderbl ON bill.cid = orderbl.orderid WHERE bill.id > 2')
|
+-----+
| plan cache: hit |
+-----+
```

Step 5 Click **Save** in the upper left corner and then **Yes** in the displayed dialog box.

Figure 4-5 Confirming the parameter modification

 **Are you sure you want to modify parameters of this DDM instance?**

Modifying the parameters may affect access to the DDM instance. Exercise caution when performing this operation.



 **NOTE**

- Modifying DDM parameters may affect access to the instance. Exercise caution when performing this operation.
- It takes 20s to 60s to have the modifications to take effect.

----End

4.11 Read/Write Splitting

Prerequisites

- You have created a DDM instance and a MySQL DB instance with read replicas.
- You have created a schema.

Procedure

Step 1 On the **Instances** page, locate the required DDM instance and click its name.

Step 2 At the **Associated DB Information** area, locate the DB instance.

Step 3 Click **Modify Read Policy** in the **Operation** column to change read/write weights of the primary DB instance and its read replicas.

 **NOTE**

Example: If the weight of the primary DB instance is 40 and that of the read replica is 50, the read ratio of the primary DB instance is $40/(40+50)$, and that of the read replica is $50/(40+50)$.

The SELECT statements that contain hints or modify data in transactions are all executed by the primary DB instance.

----End

4.12 Configuring a Parameter Template

Prerequisites

You have logged in to the DDM console.

Procedure

Step 1 In the instance list, locate the DDM instance that you want to configure a parameter template for and choose **More > Configure Parameter Template** in the **Operation** column.

The **Configure Parameter Template** dialog box is displayed.

Step 2 Select the required parameter template and click **OK**.

Figure 4-6 Configuring a parameter template

Configure Parameter Template

Instance Name ddm-8644-

Parameter Template

----End

5 Parameter Template Management

5.1 Creating a Parameter Template

A database parameter template acts as a container for engine configuration values that can be applied to one or more DDM instances. You can manage configurations of a DDM instance by managing parameters in the parameter template applied to the instance.

If you do not specify a parameter template when creating a DDM instance, the system uses the default parameter template for your instance. The default parameter template contains multiple default values, which are determined based on the computing level and the storage space allocated to the instance. You cannot modify parameter settings of a default parameter template. You must create your own parameter template to change parameter settings.

If you want to use your custom parameter template, you simply create a parameter template and select it when you create a DDM instance or apply it to an existing DDM instance following the instructions provided in [Applying a Parameter Template](#).

When you have already created a parameter template and want to provide most of its custom parameters and values in a new parameter template, you can replicate the template you created following the instructions provided in [Replicating a Parameter Template](#).

The following are the key points you should know when using parameters from a parameter template:


- When you change a parameter value in a parameter template that has been applied to a DB instance, the change applies only to the current DB instance and does not affect other DB instances.
- When you change a parameter value in a parameter template and save the change, the change will take effect only after you apply the parameter template to a DDM instance and manually restart the instance.
- Improper parameter settings may have unintended adverse effects, including degraded performance and system instability. Exercise caution when modifying parameters and you need to back up data before modifying parameters in a parameter template. Before applying parameter template

changes to a production DDM instance, you should try out these changes on a test DDM instance.

Procedure

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and a project.

Step 3 Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.

Step 4 Choose **Parameter Templates** and click **Create Parameter Template**.

Step 5 In the displayed dialog box, enter a template name and description and click **OK**.

- The template name is case-sensitive and consists of 1 to 64 characters. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), and periods (.).
- The template description consists of a maximum of 256 characters and cannot include carriage return characters and special characters >!<"&'=

NOTE

Each user can create up to 100 parameter templates.

----End

5.2 Editing a Parameter Template

You can modify parameters in custom parameter templates to improve DDM performance.

You can change parameter values in custom parameter templates only and cannot change parameter values in default parameter templates.



The following are the key points you should know when using parameters from a parameter template:

- When you change a parameter value in a parameter template and save the change, the change will take effect only after you apply the parameter template to a DDM instance and manually restart the instance. For details, see [Applying a Parameter Template](#).
- In conclusion, after you modify a parameter, the time when the modification takes effect is determined by the type of the parameter.

NOTE

Parameters in default parameter templates cannot be modified. You can view these parameters by clicking template names. If a custom parameter template is set incorrectly and causes an instance restart to fail, you can re-configure the custom parameter template according to configurations of the default parameter template.

Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.
- Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the required parameter template, and click its name.
- Step 5** On the **Parameter Details** page, modify parameters as needed.
Available operations are as follows:
 - To save the modifications, click **Save**.
 - To cancel the modifications, click **Cancel**.
- Step 6** After the parameter values are modified, you can click **Template History** to view details.

NOTICE

The modifications take effect only after you apply the parameter template to DDM instances. For details, see [Applying a Parameter Template](#).

If you have modified certain parameters or collations, you need to manually restart the DDM instance for the modifications to take effect because the restart caused by instance class changes (if any) will not make these modifications take effect.

----End



5.3 Comparing Two Parameter Templates

Scenarios

You can apply different parameter templates to the same DDM instance to view impacts of parameter settings on the instance.

You can also apply the same parameter template to different DDM instances to learn about the impacts.

Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.

Step 4 Choose **Parameter Templates**, click the **Custom Templates** tab, locate the required parameter template, and click **Compare** in the **Operation** column.

Step 5 In the displayed dialog box, select a parameter template and click **OK**.

 **NOTE**

You can also click the **Default Templates** tab and compare a default template with a customer template. The procedure is the same as comparing parameter templates on the **Custom Templates** page.

- If their settings are different, the parameter names and values of both parameter templates are displayed.
- If their settings are the same, no data is displayed.

----End

5.4 Viewing Parameter Change History

Scenarios


You can view change history of parameter templates applied to the current DDM instance or customer parameter templates


 **NOTE**

In a newly exported or created parameter template, the change history is blank.

Procedure

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and a project.

Step 3 Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.

Step 4 Choose **Parameter Templates**, click the **Customer Templates** tab, locate the required parameter template, and choose **More > View Change History**.

You can view the name, original parameter value, new parameter value, modification status, and modification time of each parameter.

----End

5.5 Replicating a Parameter Template



Scenarios

You can replicate a parameter template you have created. When you have already created a parameter template and want to provide most of its custom parameters and values in a new parameter template, you can replicate the template you created.

After a parameter template is replicated, the new template may be displayed about 5 minutes later.

Default parameter templates cannot be replicated. You can create parameter templates based on the default ones.

Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.
- Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the required parameter template, and click **Replicate** in the **Operation** column.
- Step 5** In the displayed dialog box, configure required details and click **OK**.
 - The template name is case-sensitive and consists of 1 to 64 characters. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), and periods (.).
 - The template description consists of a maximum of 256 characters and cannot include carriage return characters and special characters >!<"&'=

After the parameter template is replicated, a new template is generated in the list.



----End

5.6 Applying a Parameter Template

Scenarios

You can apply parameter templates to DDM instances as needed.

Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.
- Step 4** Choose **Parameter Templates** in the left navigation pane and proceed with subsequent operations based on the type of the required parameter template.
 - To apply a default template, click the **Default Templates** tab, locate the required parameter template, and click **Apply** in the **Operation** column.
 - To apply a custom template, click the **Custom Templates** tab, locate the required parameter template, and choose **More > Apply** in the **Operation** column.

A parameter template can be applied to one or more DDM instances.

- Step 5** In the displayed dialog box, select one or more DDM instances to which the parameter template will be applied and click **OK**.

After the parameter template is applied to DDM instances successfully, you can view its application history by referring to [Viewing Application Records of a Parameter Template](#).



----End

5.7 Viewing Application Records of a Parameter Template

Scenarios

You can apply parameter templates to DDM instances as needed.

Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.
- Step 4** Choose **Parameter Templates** in the navigation pane on the left.
- Step 5** On the **Default Templates** page, locate the target parameter template and click **View Application Record** in the **Operation** column. Alternatively, on the **Custom Templates** page, choose **More > View Application Record** in the **Operation** column.

You can view the name or ID of the DDM instance to which the parameter template is applied, as well as the application status, application time, and failure cause.

----End

5.8 Modifying the Description of a Parameter Template






Scenarios

You can modify the description of a parameter template that you have created.

 **NOTE**

You cannot modify the description of any default parameter template.

Procedure

- Step 1** Log in to the management console.
 - Step 2** Click  in the upper left corner and select a region and a project.
 - Step 3** Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.
 - Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the parameter template whose description you want to modify, and click  in the **Description** column.
 - Step 5** Enter a new description. You can click  to submit or  to cancel the modification.
 - The description contains up to 256 characters but cannot contain special characters >!<"&'=
 - After the modification is successful, you can view the new description in the **Description** column.
- End

5.9 Deleting a Parameter Template



Scenarios

You can delete custom parameter templates that will not be used any more.

NOTICE

- Deleted parameter templates cannot be recovered. Exercise caution when performing this operation.
 - Default parameter templates cannot be deleted.
-

Procedure

- Step 1** Log in to the management console.
 - Step 2** Click  in the upper left corner and select a region and a project.
 - Step 3** Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.
 - Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the template that you want to delete, and click **Delete** in the **Operation** column.
 - Step 5** In the displayed dialog box, click **Yes**.
- End

6 Task Center

You can view the progress and results of asynchronous tasks on the **Task Center** page.


NOTE


The following tasks can be viewed:

- Creating a DDM instance
- Deleting a DDM instance
- Changing class of a DDM instance
- Scaling out a DDM instance
- Scaling in a DDM instance
- Restarting a DDM instance
- Binding an EIP to a DDM instance
- Unbinding an EIP from a DDM instance
- Restoring data from current instance
- Importing schema information
- Scaling out a schema
- Creating a consistent backup
- Deleting a consistent backup
- Restoring data using a consistent backup

Procedure


Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and a project.

Step 3 Click  in the upper left corner of the page and choose **Database > Distributed Database Middleware**.

Step 4 Choose **Task Center** in the left navigation pane, locate the required task, and view its details.

- You can use the task name, order ID, or DB instance name/ID to locate the required task, or simply enter the target task name in the search box in the upper right corner.

- You can click  in the upper right corner to search for tasks executed within a specific period. The default time range is seven days. All tasks in the list can be retained up to one month.
- You can view the tasks that are in the following statuses:
 - Running
 - Completed
 - Failed
- You can view the task creation and completion time.

----End

7 Schema Management

7.1 Creating a Schema

Prerequisites

- You have logged in to the DDM console.
- The DDM instance is in the **Running** state.
- DDM supports RDS and GaussDB(for MySQL) DB instances. The following uses an RDS MySQL DB instance as an example.
- Do not modify or delete RDS internal accounts (DDMRW*, DDMR*, and DDMREP*). Otherwise, services are affected.

NOTE

- The internal account is in the format: fixed prefix (DDMRW, DDMR, or DDMREP) +hash value of the RDS DB instance ID.
- Password rule: The password is generated randomly and must contain 16 to 32 characters.

You can create a schema in two ways: on the **Instances** page or on the **Schemas** page. This section uses the DDM instance page as an example to describe how to create a schema.

Figure 7-1 On the **Instances** page



Figure 7-2 On the **Schemas** page



Procedure

- Step 1** In the navigation pane, choose **Instances**. In the instance list, locate the DDM instance that you want to create a schema for and click **Create Schema** in the **Operation** column.
- Step 2** On the displayed page, specify a sharding mode, enter a schema name, select the number of shards per DB instance, and click **Next: Select DB Instance**.

Figure 7-3 Create Schema

NOTE

Sharding mode:

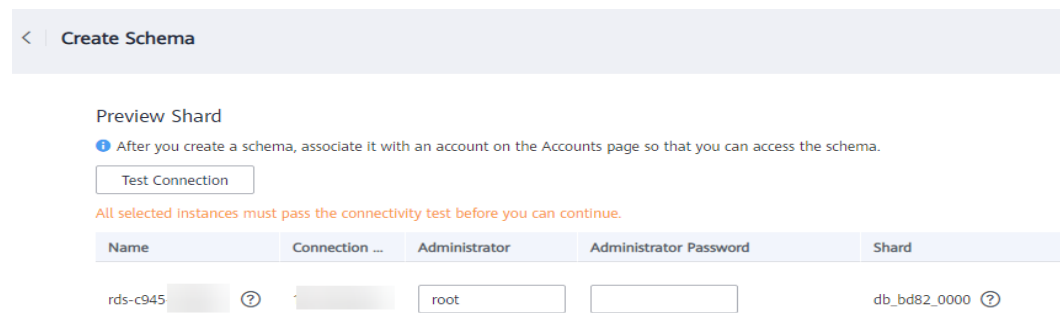
- **Sharded:** One schema corresponds to multiple DB instances.
- **Unsharded:** One schema corresponds only to one DB instance, and only one database shard is created for that instance.

The schema name contains 2 to 24 characters and must start with a letter. Only letters, digits, and underscores (_) are allowed.

The shards per DB instance ranges from 1 to 64.

- Step 3** On the **RDS DB Instances** tab page, select the required RDS DB instances. You can choose to configure the instances as the unsharded table storage nodes. Click **Preview**.
- Step 4** On the displayed page, enter the administrator password and click **Test Connection**. After the test is successful, click **Finish**.

Figure 7-4 Testing Connection



----End

7.2 Exporting Schema Information

Scenarios

Export schema information from the primary DDM instance for cross-region DR.

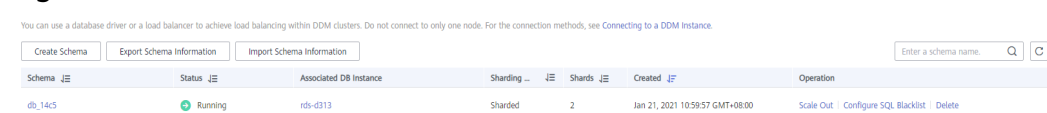
Prerequisites

There are schemas available in the DDM instance that you want to export schema information from.

Procedure

- Step 1** Log in to the DDM console; in the instance list, locate the required DDM instance and click its name.
- Step 2** On the displayed page, in the navigation pane, choose **Schemas**.

Figure 7-5 Schema list



- Step 3** On the displayed page, click **Export Schema Information**. All schema information of the current DDM instance is exported as a JSON file.

----End

7.3 Importing Schema Information

Scenarios

Create a schema in the standby DDM instance for cross-region DR.

Prerequisites

The standby DDM instance has no schemas.

Procedure

- Step 1** Log in to the DDM console; in the instance list, locate the DDM instance that you want to import schema information into and click its name.
- Step 2** On the displayed page, in the navigation pane, choose **Schemas**.

Figure 7-6 Schema list

You can use a database driver or a load balancer to achieve load balancing within DDM clusters. Do not connect to only one node. For the connection methods, see [Connecting to a DDM Instance](#).

Create Schema
Export Schema Information
Import Schema Information

Enter a schema name. Q C

Schema	Status	Associated DB Instance	Sharding ...	Shards	Created	Operation
db_145	Running	rds-d313	Sharded	2	Jan 21, 2021 10:59:57 GMT+08:00	Scale Out Configure SQL Blacklist Delete

- Step 3** On the displayed page, click **Import Schema Information**.

NOTE

More than one JSON file can be imported into a DDM instance in the premise that the DDM instance does not have schemas with the same names as those in the JSON file.

- Step 4** On the displayed page, click **Select File** to select the JSON file exported in [Exporting Schema Information](#), choose the required DB instances, enter the required password, and click **Finish**.

NOTE

The number of selected DB instances must be the same as the number of DB instances associated with the imported schemas.

----End

7.4 Deleting a Schema

Prerequisites

- You have logged in to the DDM console.
- You have created a schema.

NOTICE

Deleted schemas cannot be recovered. Exercise caution when performing this operation.

Procedure

- Step 1** In the instance list, locate the DDM instance that you want to delete and click its name.
- Step 2** In the navigation pane, choose **Schemas**.
- Step 3** In the schema list, locate the schema that you want to delete and click **Delete** in the **Operation** column.
- Step 4** In the displayed dialog box, click **Yes**.

 NOTE

- Your schema will become faulty if its associated RDS DB instances are deleted on the RDS console.
- To delete data stored on associated RDS MySQL DB instances, select **Delete data on associated RDS DB instances**.
- Before you delete a schema, check whether there are DB instances associated with this schema. If the associated DB instances have been deleted, click **Synchronize DB Instance Data** and delete the schema.
- If the associated RDS DB instances are not deleted but their information is modified, such as the instance name, engine, engine version, maximum connections, port number, or IP address, click **Synchronize DB Instance Data** and delete the schema.

----End

7.5 Configuring the SQL Blacklist

Overview

To prohibit executing some SQL statements, you can configure a blacklist and add those statements to it.

Prerequisites


- You have logged in to the DDM console.
- A DDM instance is running properly and has available schemas.

Procedure

- Step 1** In the instance list, locate the instance that contains schemas you require and click the instance name.
- Step 2** On the displayed page, choose **Schemas**.
- Step 3** In the schema list, locate the schema that you want to configure a blacklist for and click **Configure SQL Blacklist** in the **Operation** column.

Figure 7-7 Configure SQL Blacklist

Configure SQL Blacklist


 The total size of data in Prefix Match, Full-text Match, and Regular Expression Match text boxes cannot exceed 1 KB.

[^](#)

Prefix Match

Full-text Match

Regular Expression Match

 Use a backslash (\) before the characters that need to be escaped.

Step 4 In the displayed dialog box, click **Edit**, enter the required SQL statements or regular expressions in prefix match, full-text, and regular expression match boxes, and click **OK**.

 **NOTE**

- **Prefix Match:** Enter SQL statements that contain keywords such as DROP XXXX or DELETE XXX and are not allowed by the current schema.
- **Full-text Match:** Enter full-text SQL statements that are not allowed by the current schema.
- Separate SQL statements in the blacklist with commas (,). The total size of SQL statements for prefix match and full-text match cannot exceed 1 KB.
- If you want to clear all the SQL statements in prefix match and full-text match areas, clear them separately and click **OK**.

----End

7.6 Scaling Out a Schema

Service growth requires schemas to provide more storage space and support higher concurrency. To expand storage and improve concurrency, you can scale out a schema by adding DB instances.

This section uses an RDS for MySQL DB instance as an example to describe how to scale out a schema.

7.6.1 Rebalancing Shards Across All DB Instances

Overview

With rebalancing, database shards are rebalanced between original and new DB instances.

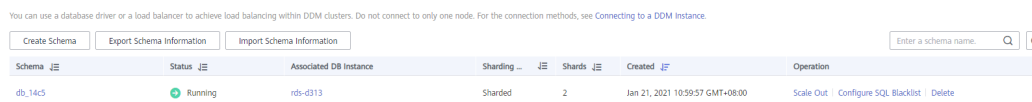
Prerequisites

- There are schemas available in a running DDM instance.
- You have purchased a MySQL DB instance. The DB instance and your DDM instance are in the same VPC, and the DB instance is not in use by other DDM instances.

Procedure

- Step 1** In the instance list, locate the instance that contains schemas you require and click the instance name.
- Step 2** On the displayed page, in the navigation pane, choose **Schemas**.
- Step 3** In the schema list, locate the schema that you want to scale out and click **Scale Out** the **Operation** column.

Figure 7-8 Scale Out



- Step 4** On the displayed page, select **Rebalance** for **Scaling Method**, configure **Traffic Switch** and **Automatic Clearance Required**, select the required DB instances, enter the administrator password, and click **Test Connection**. After the connection is successful, click **Next: Confirm Information**.

Figure 7-9 Rebalance

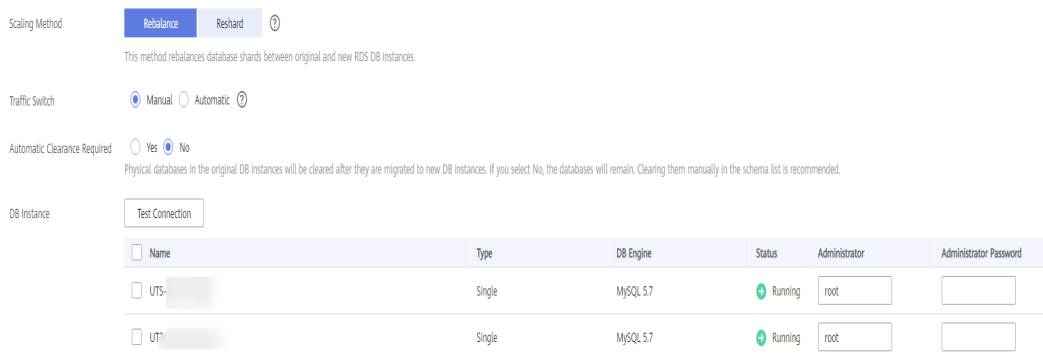
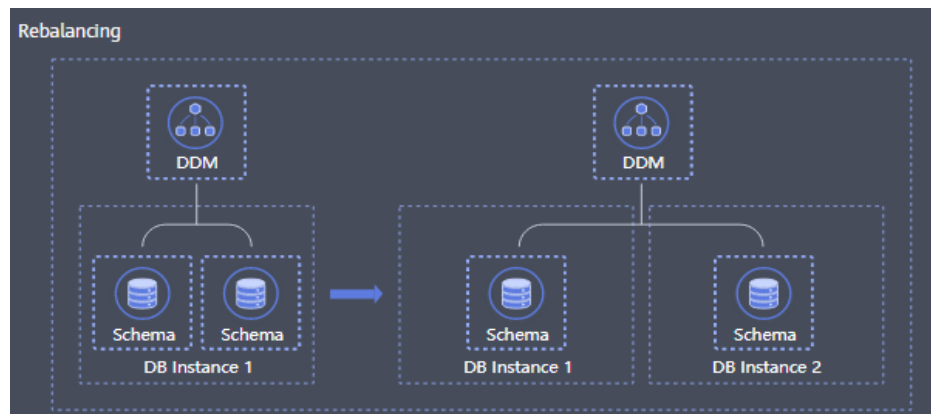


Figure 7-10 How rebalancing works



Step 5 Click **Next: Pre-check**. After all check items are complete, click **Scale out**.

If the schema status changes to **Running**, the scaling out succeeds and the newly-added MySQL DB instance is displayed in the **Associated DB Instance** column.

NOTE

If the DDM instance version is earlier than 2.4.1.3, do not set **sql_mode** to **ANSI_QUOTES**. If you do so, do not use double quotation marks to quote text strings because the quotation marks will be translated into an identifier that makes the quoted string invalid. For example, **logic** in **select * from test where tb = "logic"** cannot be parsed correctly.

Step 6 If **Status** changes to **Scaling failed**, the schema failed to be scaled out. Click **Retry** or **Roll back**.

NOTE

- Scale out a schema during off-peak hours.
- Storage scaling does not support tables without primary keys.
- Only schemas whose status is **Running** can be scaled out.
- Only one schema can be scaled out in a DDM instance at a time.
- A maximum of 256 MySQL DB instances can be selected for scaling.
- You can view the scaling progress on the **Schemas** page. The whole scaling process takes about 5 to 30 minutes depending on the amount of data to be migrated.
- During scaling, DDM automatically enables the LOAD DATA function. After scaling, you can disable the function in parameter template settings of the MySQL DB instance as needed.

----End

7.6.2 Doubling Shards and Resharding Data Records

Overview

With resharding, physical databases corresponding to the schema are doubled and rebalanced to each DB instance as much as possible. During this process, data records are redistributed.

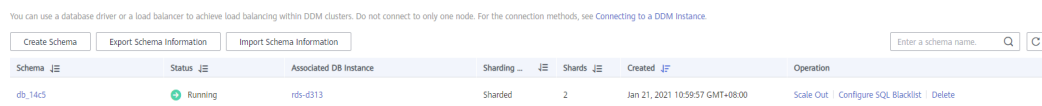
Prerequisites

- There are schemas available in a running DDM instance.
- You have purchased a MySQL DB instance. The DB instance and your DDM instance are in the same VPC, and the DB instance is not in use by other DDM instances.

Procedure

- Step 1** In the instance list, locate the instance that contains schemas you require and click the instance name.
- Step 2** On the displayed page, in the navigation pane, choose **Schemas**.
- Step 3** In the schema list, locate the schema that you want to scale out and click **Scale Out** the **Operation** column.

Figure 7-11 Scale Out



- Step 4** On the displayed page, select **Reshard** for **Scaling Method**, configure **Traffic Switch** and **Automatic Clearance Required**, select the required DB instances, enter the administrator password, and click **Test Connection**. After the connection is successful, click **Next: Confirm Information**.

Figure 7-12 Reshard

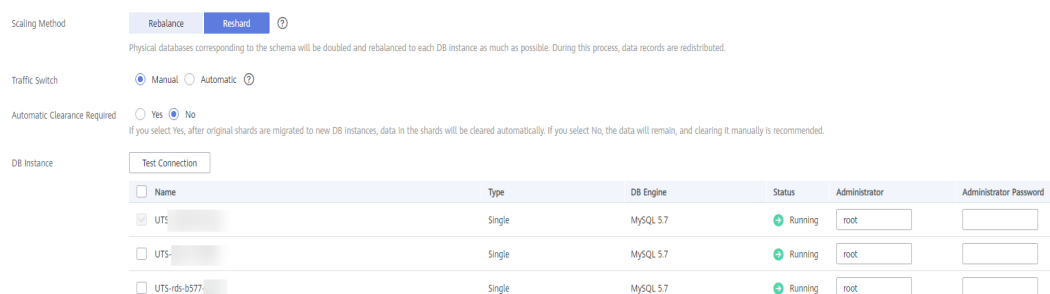
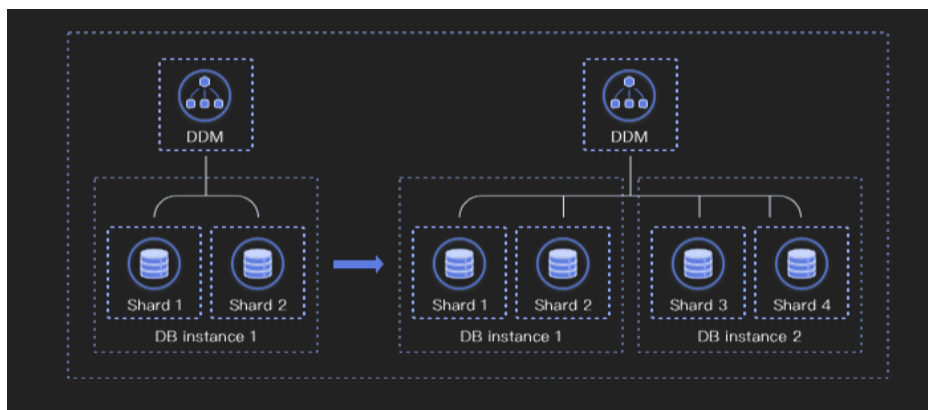


Figure 7-13 Reshard



Step 5 Click **Next: Pre-check**. After all check items are complete, click **Scale out**.

If the schema status changes to **Running**, the scaling out succeeds and the newly-added MySQL DB instance is displayed in the **Associated DB Instance** column.

 **NOTE**

If the DDM instance version is earlier than 2.4.1.3, do not set **sql_mode** to **ANSI_QUOTES**. If you do so, do not use double quotation marks to quote text strings because the quotation marks will be translated into an identifier that makes the quoted string invalid. For example, **logic** in **select * from test where tb = "logic"** cannot be parsed correctly.

Step 6 If **Status** changes to **Scaling failed**, the schema failed to be scaled out. Click **Retry** or **Roll Back**.

 **NOTE**

- Scale out a schema during off-peak hours.
- Storage scaling does not support tables without primary keys.
- Only schemas whose status is **Running** can be scaled out.
- Only one schema can be scaled out in a DDM instance at a time.
- A maximum of 256 MySQL DB instances can be selected for scaling.
- You can view the scaling progress on the **Schemas** page. The whole scaling process takes about 5 to 30 minutes depending on the amount of data to be migrated.
- During scaling, DDM automatically enables the LOAD DATA function. After scaling, you can disable the function in parameter template settings of the MySQL DB instance as needed.

----End

8 Account Management

8.1 Creating an Account

Prerequisites

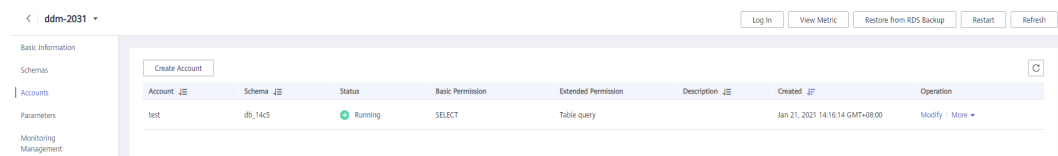
- You have logged in to the DDM console.
- There are schemas available in the DDM instance that you want to create an account for.

Procedure

Step 1 In the instance list, locate the required DDM instance and click its name.

Step 2 In the navigation pane, choose **Accounts**.

Figure 8-1 Account management



Step 3 On the displayed page, click **Create Account** and configure the required parameters.

Figure 8-2 Creating a DDM account

Create Account

* Username ?

* Password

* Confirm Password

Schema

* Permissions

Basic Permissions

CREATE DROP ALTER INDEX INSERT DELETE

UPDATE SELECT

Extended Permissions

Table query Table delete Table update

Description 0/256

Table 8-1 Required parameters

Parameter	Description
Username	Username of the account. The username consists of 1 to 32 characters and starts with a letter. Only letters, digits, and underscores (_) are allowed.
Password	Password of the account. The password: <ul style="list-style-type: none"> • Contains 8 to 32 characters. • The password must contain at least three types of lowercase letters, uppercase letters, digits, and special characters ~! @#%^*_-=+?
Confirm Password	None

Parameter	Description
Permissions	<p>Permissions of a DDM account, including:</p> <ul style="list-style-type: none"> • Basic Permissions: CREATE, DROP, ALTER, INDEX, INSERT, DELETE, UPDATE, and SELECT • Extended Permissions: Table query, Table delete, and Table update <p>When you configure permissions, select basic permissions based on your needs.</p> <p>After you select a basic permission, the corresponding extended permissions are selected automatically. Not every basic permission corresponds to an extended permission. The extended permissions for each basic permission are as follows:</p> <ul style="list-style-type: none"> - Table query corresponds to SELECT. - Table delete corresponds to DELETE. - Table update corresponds to UPDATE.
Schema	<p>Schema to be associated with the account. You can select an existing schema from the drop-down list.</p> <p>The account can access only the associated schemas.</p>
Description	<p>Description of the account, which cannot exceed 256 characters.</p>

Step 4 Confirm the settings and click **OK**.

----End

8.2 Modifying an Account

Prerequisites

You have logged in to the DDM console.

Procedure

Step 1 In the instance list, locate the DDM instance that you want to modify and click its name.

Step 2 In the navigation pane, choose **Accounts**.

Step 3 In the account list, locate the required account and click **Modify** in the **Operation** column.

Figure 8-3 Modifying an account

Account	Schema	Status	Basic Permission	Extended Permission	Description	Created	Operation
test	db_14c5	Running	SELECT	Table query		Jan 21, 2021 14:16:14 GMT+08:00	Modify More

Step 4 In the displayed dialog box, modify the associated schemas, permissions, and description.

Figure 8-4 Modifying a DDM account

Modify Account

Username test

Schema db_14c5

* Permissions Basic Permissions

CREATE DROP ALTER INDEX INSERT

DELETE UPDATE SELECT

Extended Permissions

Table query Table delete Table update

Description Enter a description.

0/256

OK Cancel

Step 5 Click **OK**.

----End

8.3 Deleting an Account

Prerequisites

You have logged in to the DDM console.

NOTE

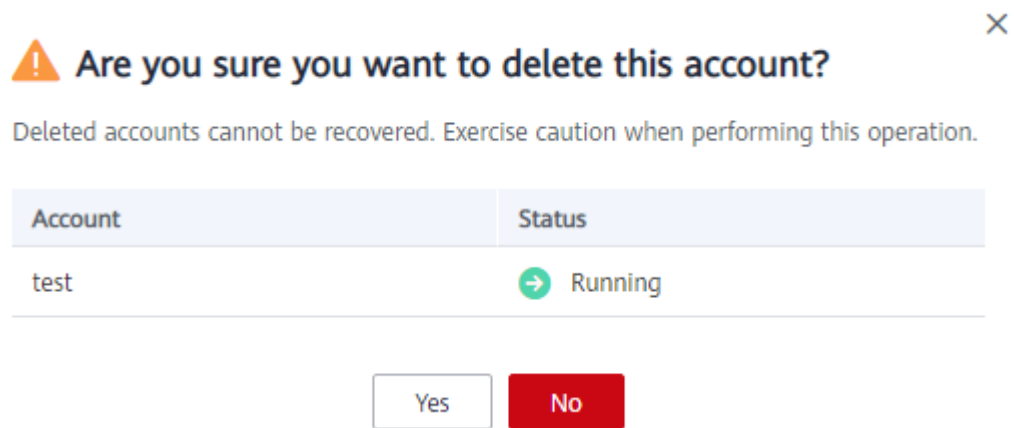
Deleted accounts cannot be recovered. Exercise caution when performing this operation.

Procedure

Step 1 In the instance list, locate the DDM instance with the account that you want to delete and click its name.

- Step 2** In the navigation pane, choose **Accounts**.
- Step 3** In the account list, locate the account that you want to delete and choose **More > Delete** in the **Operation** column.

Figure 8-5 Deleting an account



- Step 4** In the displayed dialog box, click **Yes**.
- End

8.4 Resetting the Password of an Account

Prerequisites

You have logged in to the DDM console.

Procedure

- Step 1** In the instance list, locate the DDM instance with the account whose password you want to reset and click its name.
- Step 2** In the navigation pane, choose **Accounts**.
- Step 3** In the account list, locate the required account and choose **More > Reset Password** in the **Operation** column.

Figure 8-6 Resetting the password of an account

Reset Password

Account Name

* Password

* Confirm Password

Step 4 In the displayed dialog box, enter the new password, confirm the new password, and click **OK**.

----End

9 Backup Management

9.1 Restoring from Current Instance

Prerequisites

- You have logged in to the DDM console.
- The current (source) DDM instance is in the **Running** state.
- Both GaussDB (for MySQL) and RDS DB instances are supported. The following uses RDS DB instances as an example.

Procedure

Step 1 Create a DDM instance or select an existing DDM instance that meets the requirements in the region where the source DDM instance is located.

Step 2 Create as many RDS DB instances as there are in the source DDM instance.

NOTE

- The new DDM instance has no RDS DB instances associated.
- The selected existing DDM instance has no schemas or accounts created.
- Select engine version MySQL 5.7 for each new RDS DB instance and ensure that its minor version is not earlier than the RDS DB instance version in the source DDM instance.
- Ensure that the storage space of each new RDS DB instance is not less than that in the source DDM instance.

Step 3 In the DDM instance list, click the name of the instance created in [1](#).

Step 4 In the upper right corner of the **Basic Information** area, click **Restore from RDS Backup**.

Step 5 On the displayed **Restore from RDS Backup** page, specify a time range and a point in time and select the DDM instance created in [1](#) as the destination DDM instance.

Step 6 Select the RDS DB instances created in [2](#) as destination DB instances and check the confirmation box. Click **OK**. Wait for 1 to 3 minutes for the data restoration to complete.

 **NOTE**

- Restoring to an existing DDM instance will overwrite data on it and cause the instance to be unavailable during restoration.
- Ensure that no schema has been created or deleted since the time point you specify.
- Ensure that the destination DB instances have the same engines and versions as or later versions than and at least as much as storage space as source DB instances.

----End

9.2 Creating a Consistent Backup

9.2.1 Creating a Backup

Prerequisites

- You have logged in to the DDM console.
- The DDM instance that you want to back up is in the **Running** state.
- There are schemas available in the DDM instance.

Procedure

- Step 1** In the instance list, locate the DDM instance that you want to back up and click its name.
- Step 2** In the navigation pane, choose **Backups**.
- Step 3** On the displayed page, click **Create Backup**.
- Step 4** In the displayed dialog box, enter a name for the backup and a brief description and click **OK**.

 **NOTE**

The time required depends on the amount of data to be backed up.

----End

9.2.2 Restoring from a Backup

Prerequisites

- You have logged in to the DDM console.
- The DDM instance that you want to restore is in the **Running** state.
- There are schemas available in the DDM instance.
- The DDM instance has been fully backed up.

Procedure

- Step 1** In the instance list, locate the DDM instance that you want to restore and click its name.

Step 2 In the navigation pane, choose **Backups**.

Step 3 In the backup list, locate the required backup and click **Restore** in the **Operation** column.

Step 4 On the displayed page, select the destination DDM instance and destination RDS DB instances. Confirm the configurations and click **OK**.

 **NOTE**

- The time required depends on the amount of data to be restored.
- The destination DDM instance has no DDM accounts created.

----**End**

10 Monitoring Management

10.1 Metrics

Description

This section describes metrics reported by DDM to Cloud Eye as well as their namespaces, metric list, and dimensions. You can use APIs provided by Cloud Eye to query the metric information generated for DDM.

Metrics

Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
CPU Usage	CPU usage of the DDM instance node	0-100%	DDM nodes	1 minute
Memory Usage	Memory usage of the DDM instance node.	0-100%	DDM nodes	1 minute
Network Input Throughput	Incoming traffic per second of the DDM instance node	≥ 0 byte/s	DDM nodes	1 minute
Network Output Throughput	Outgoing traffic per second of the DDM instance node	≥ 0 byte/s	DDM nodes	1 minute
QPS	Requests per second of the DDM instance node	≥ 0 counts	DDM nodes	1 minute
Read/Write Ratio	Read/Write ratio of the DDM instance node within n each monitoring period	0-100%	DDM nodes	1 minute

Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
Reads	Read operations of the DDM instance node within each monitoring period	≥ 0 counts/s	DDM nodes	1 minute
Writes	Write operations of the DDM instance node within a monitoring period	≥ 0 counts/s	DDM nodes	1 minute
Slow SQL Logs	Number of slow SQL queries added within a monitoring period	≥ 0 counts	DDM nodes	1 minute
Average Response Latency	Average SQL response latency of the DDM instance node	≥ 0 ms	DDM nodes	1 minute
Connections	Connections of the DDM instance node	≥ 0 counts	DDM nodes	1 minute
Tuning Backend Connections	Percentage of active connections (from a DDM node to the target RDS DB instance) in maximum backend connections	0-100%	DDM nodes	1 minute

Dimensions

Key	Value
ddm_node_id	DDM nodes

10.2 Viewing Metrics

Scenarios

The DDM console supports monitoring and management of DDM instances, including the read/write ratio and slow SQL logs. You can optimize databases based on the monitoring results.

- Slow SQL refers to SQL statements that take a long time to run on your database. Statistical analysis of SQL helps you locate performance bottlenecks of your database.

- The read/write ratio is the proportion of read requests to read and write requests on a logical table.

Prerequisites

- You have logged in to the DDM console.
- There is a DDM instance available, which has available schemas.

Procedure

Step 1 In the instance list, locate the DDM instance whose metrics you want to view and click its name.

Step 2 In the navigation pane, choose **Monitoring Management**.

Step 3 Select **Slow SQL Log** and **Read/Write Ratio** tabs to view monitoring details as follows:

- Slow SQL logs

Specify a time range to filter logs. After you locate the required log, view the account, schema, SQL statement, execution start time, execution duration, shard, and affected lines.

Figure 10-1 Slow SQL log information

Account	Schema	SQL Statement	Executed	Execution Time (ms)	Shard	Affected Lines
---------	--------	---------------	----------	---------------------	-------	----------------

Table 10-1 Parameter description

Parameter	Description
Account	DDM account used for executing SQL statements
Schema	Name of the schema
SQL Statement	SQL statements that take a long time to run on a DDM instance
Executed	Time when the first slow SQL statement is executed within the specified period of time
Execution Time (ms)	Unit of milliseconds
Shard	All shards involved in the SQL execution
Affected Lines	Lines involved in SQL execution

- Read/Write ratio

Specify a time range to filter read/write ratio information. Obtain the reads, writes, and the last execution time.

Figure 10-2 Read/Write ratio information

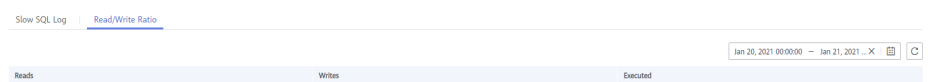
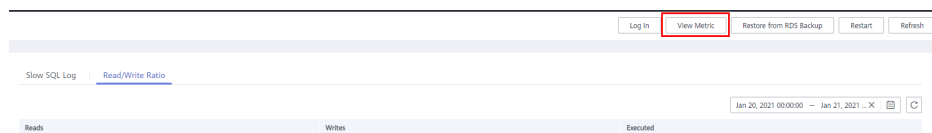


Table 10-2 Parameter description


Parameter	Description
Reads	Number of read operations on a logical table. You can specify a period of time for querying and collecting reads.
Writes	Number of write operations on a logical table. You can specify a period of time for querying and collecting reads.
Executed	Time when the last SQL statement is executed within the specified period of time

Step 4 Click **View Metric** in the upper right corner.

Figure 10-3 View Metric



Step 5 On the Cloud Eye console, view monitoring metrics of the DDM instance.

1. In the navigation pane, choose **Cloud Service Monitoring > Distributed Database Middleware**.
2. In the instance list, locate the target DDM instance, click  to view instance details, and click **View Metric** in the **Operation** column.
3. On the displayed page, select the required monitoring period to view metric information.

----End

11 Auditing

11.1 Key Operations Recorded by CTS

Cloud Trace Service (CTS) records operations related to DDM for further query, audit, and backtrack.

Table 11-1 DDM operations that can be recorded by CTS

Operation	Resource Type	Trace Name
Applying a parameter template	parameterGroup	applyParameterGroup
Clearing metadata after a schema is scaled out	logicDB	cleanMigrateLogicDB
Clearing user resources	all	cleanupUserAllResources
Comparing two parameter templates	parameterGroup	compareParameterGroup
Replicating a parameter template	parameterGroup	copyParameterGroup
Creating a DDM instance	instance	createInstance
Creating a schema	instance	createLogicDB
Creating a parameter template	parameterGroup	createParameterGroup
Creating an account	instance	createUser
Modifying an enterprise project	all	dealResourceTag
Deleting a DDM instance	instance	deleteInstance
Deleting a schema	logicDB	deleteLogicDB

Operation	Resource Type	Trace Name
Deleting a parameter template	parameterGroup	deleteParameterGroup
Deleting an account	instance	deleteUser
Scaling out a DDM instance	instance	enlargeNode
Restarting a DDM instance	instance	instanceRestart
Importing schema information	logicDB	loadMetadata
Switching the route during scaling	logicDB	manualSwitchRoute
Scaling out a schema	logicDB	migrateLogicDB
Modifying a parameter template	parameterGroup	modifyParameter-GroupTempate
Changing the route switching time	logicDB	modifyRouteSwitch-Time
Modifying an account	instance	modifyUser
Scaling in a DDM instance	instance	reduceNode
Resetting a parameter template	parameterGroup	resetParameterGroup
Resetting the password of an account	instance	resetUserPassword
Changed the class	instance	resizeFlavor
Restoring DB instance data	instance	restoreInstance
Retrying to scale out a schema	logicDB	retryMigrateLogicDB
Rolling back the version upgrade of a DDM instance	instance	rollback
Rolling back a schema scaling task	logicDB	rollbackMigrateLogicDB
Saving parameters in a parameter template	parameterGroup	saveParameterGroup
Configuring access control	instance	switchIpGroup
Synchronizing DB data	instance	synRdsinfo
Upgrading the version of a DDM Instance	instance	upgrade


11.2 Querying Traces

Scenarios

After CTS is enabled, the tracker starts recording operations on cloud resources. Operation records for the last 7 days are stored on the CTS console.

This section describes how to query operation records for the last 7 days on the CTS console.

Procedure

- Step 1** Log in to the management console.
- Step 2** Under **Management & Deployment**, click **Cloud Trace Service**.
- Step 3** Choose **Trace List** in the navigation pane on the left.
- Step 4** Specify filter criteria to search for the required traces. The following filters are available:
 - **Trace Source, Resource Type, and Search By:** Select a filter from the drop-down list.
When you select **Resource ID** for **Search By**, you also need to select or enter a resource ID.
 - **Operator:** Select a specific operator from the drop-down list.
 - **Trace Status:** Available options include **All trace statuses, Normal, Warning, and Incident**. You can only select one of them.
 - In the upper right corner of the page, you can specify a time range for querying traces.
- Step 5** Click **Query**.
- Step 6** Locate the required trace and click  on the left of the trace to view details.
- Step 7** Click **View Trace** in the **Operation** column. On the displayed dialog box, the trace structure details are displayed.
- Step 8** Click **Export** on the right. CTS exports traces collected in the past seven days to a CSV file. The CSV file contains all information related to traces on the management console.

For details about key fields in the trace structure, see sections "Trace Structure" and "Trace Examples" in the *Cloud Trace Service User Guide*.

----End

12 SQL Syntax

12.1 Introduction

DDM is compatible with the MySQL license and syntax, but the use of SQL statements is limited due to differences between distributed databases and single-node databases.

Before evaluating a DDM solution, evaluate the compatibility between SQL syntax in your application and syntax supported by DDM.

MySQL EXPLAIN

When you add **EXPLAIN** before a SQL statement is executed, you will see a specific execution plan. Analyze the time required based on the plan and modify the SQL statement for optimization.

Table 12-1 Description of the **EXPLAIN** column

Column Name	Description
table	Table that the data record belongs to
type	Type of the connection. The connections from the best to the worst are const , eq_reg , ref , range , index , and ALL .
possible_keys	Index that may be applied to the table
key	Index that is actually used. If the value is NULL , no index is used. In some cases, MySQL may select indexes that are not fully optimized. Add USE INDEX(indexname) to the SELECT statement to forcibly use an index, or IGNORE INDEX(indexname) to force MySQL to ignore the index.
key_len	Length of the used index. The shorter the length is, the better the index is if accuracy is not affected.

Column Name	Description
ref	Column where the index is used. The value is generally a constant.
rows	Rows of the data returned by MySQL
Extra	Additional information about how MySQL parses queries

SQL Restrictions

- Temporary tables are not supported.
- Foreign keys, views, cursors, triggers, and stored procedures are not supported.
- Customized data types and functions are not supported.
- Process control statements such as IF and WHILE are not supported.
- Compound statements such as BEGIN...END, LOOP...END LOOP, REPEAT...UNTIL...END REPEAT, and WHILE...DO...END WHILE are not supported.

DDL Syntax

- Sharded and broadcast tables do not support foreign keys.
- Modifying sharding keys is not supported.
- ALTER DATABASE Syntax is not supported.
- Creating sharded or broadcast tables from another table is not supported.
- The CREATE TABLE statement does not support GENERATED COLUMN.
- Modifying sharding keys or global sequence fields using the **ALTER** command is not supported.
- Creating TEMPORARY sharded or broadcast tables is not supported.
- The logical table name contains only letters, digits, and underscores (_).
- CREATE TABLE tbl_name LIKE old_tbl_name is not supported.
- The CREATE TABLE tbl_name SELECT statement is not supported.
- Updating the sharding key by executing INSERT INTO ON DUPLICATE KEY UPDATE is not supported.
- Cross-schema DDL is not supported, for example, CREATE TABLE db_name.tbl_name (...)
- Reverse quotation marks are required to quote identifiers such as table names, column names, and index names that are MySQL key words or reserved words.

DML Syntax

- PARTITION clauses are not supported.
- Nesting a subquery in an UPDATE statement is not supported.
- INSERT DELAYED Syntax is not supported.

- STRAIGHT_JOIN and NATURAL JOIN are not supported.
- Multiple-table UPDATE is supported if all tables joined across shards have primary keys.
- Multiple-table DELETE is supported if all tables joined across shards have primary keys.
- Manipulating variables in SQL statements is not supported, for example, SET @c=1, @d=@c+1; SELECT @c, @d.

Unsupported Functions

- XML functions
- GTID functions
- Full-text search functions
- Enterprise encryption functions
- Function **row_count()**

Subqueries

Using subqueries in the HAVING clause and the JOIN ON condition is not supported.

Data Types

Spatial data

12.2 DDL

DDM supports common DDL operations, such as creating databases, creating tables, and modifying table structure, but the implementation method is different from that in common MySQL databases.

DDL Statements that Can Be Executed on a MySQL Client

- **TRUNCATE Syntax**
Example:
TRUNCATE TABLE t1
Deletes all data from table t1.
TRUNCATE TABLE is used to delete all data from a table and has the DROP permission. In logic, TRUNCATE TABLE is similar to the DELETE statement for deleting all data from a table.
- **ALTER TABLE Syntax**
Example:
ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
Deletes columns c and d from table t2.
ALTER is used to add or delete a column, create or drop an index, change the type of an existing column, rename columns or tables, or change the storage engine or comments of a table.
- **DROP INDEX Syntax**
Example:
DROP INDEX 'PRIMARY' ON t;
Deletes primary key from table t.
DROP INDEX is used to delete index *index_name* from table *tbl_name*.
- **CREATE INDEX Syntax**
Example:
CREATE INDEX part_of_name ON customer (name(10));
Creates an index using the first 10 characters in column name (assuming that there are non-binary

character strings in column name).
CREATE INDEX is used to add an index to an existing table.

- **CREATE, DROP, UPDATE**

Example:

```
CREATE DATABASE t shard 100;
```

Creates database t that contains 100 shards.

```
CREATE DATABASE is used to create a database with a specified name.
```

12.2.1 Creating a Table

Database and Table Sharding

The following is an example when HASH is used for database sharding and MOD_HASH for table sharding:

```
CREATE TABLE tpartition_tbl (  
id int NOT NULL AUTO_INCREMENT COMMENT 'Primary key ID',  
name varchar(128),  
PRIMARY KEY(id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci  
DBPARTITION BY HASH(id)  
TBPARTITION BY mod_hash(name) tpartitions 8;
```

Database Sharding

The following is an example when HASH is used:

```
CREATE TABLE dbpartition_tbl (  
id int NOT NULL AUTO_INCREMENT COMMENT 'Primary key ID',  
name varchar(128),  
PRIMARY KEY(id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci  
DBPARTITION BY HASH(id);
```

Creating a Broadcast Table

The following is an example statement:

```
CREATE TABLE broadcast_tbl (  
id int NOT NULL AUTO_INCREMENT COMMENT 'Primary key ID',  
name varchar(128),  
PRIMARY KEY(id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci  
BROADCAST;
```

Creating a Table When Sharding Is not Used

The following is an example statement:

```
CREATE TABLE single(  
id int NOT NULL AUTO_INCREMENT COMMENT 'Primary key ID',  
name varchar(128),  
PRIMARY KEY(id)  
);
```

12.2.2 Sharding Algorithm Overview

Supported Sharding Algorithms

DDM supports database sharding, table sharding, and a variety of sharding algorithms.

Table 12-2 Sharding algorithms

Function	Description	Database Sharding	Table Sharding
MOD_HASH	Sharding using modulo operation	Yes	Yes
UNI_HASH	Sharding using modulo operation	Yes	Yes
RIGHT_SHIFT	Sharding using right-shift operation of the database sharding key	Yes	Yes
YYYYMM	Sharding by year and month	Yes	Yes
YYYYDD	Sharding by year and day	Yes	Yes
YYYYWEEK	Sharding by year and week	Yes	Yes
HASH	Sharding using modulo operation	Yes	Yes
RANGE	Sharding by range	Yes	No
MM	Sharding by month	No	Yes
DD	Sharding by date	No	Yes
MMDD	Sharding by month and day	No	Yes
WEEK	Sharding by week	No	Yes

 **NOTE**

- Database and table sharding keys cannot be left blank.
- In DDM, sharding of a logical table is defined by the sharding function (number of shards and routing algorithm) and the sharding key (including MySQL data type).
- When database and table sharding algorithms of a logical table are different, full-shard or full-table scanning is performed if you do not specify conditions for database and table sharding in SQL queries.

Data Type of Sharding Algorithms

Different sharding algorithms support different data types. The following table lists supported data types.

Table 12-3 Supported Data types

Sharding Algorithm	TINYINT	SMALLINT	MEDIUMINT	INT	BIGINT	CHAR	VARCHAR	DATE	DATETIME	TIMESTAMP	OTHERS
HASH	√	√	√	√	√	√	√	√	√	√	×
UNI_HASH	√	√	√	√	√	√	√	×	×	×	×
RIGHT_SHIFT	√	√	√	√	√	×	×	×	×	×	×
YYYYDD	×	×	×	×	×	×	×	√	√	√	×
YYYYWEEK	×	×	×	×	×	×	×	√	√	√	×
YYYYMM	×	×	×	×	×	×	×	√	√	√	×

Table Creation Syntax of Sharding Algorithms

DDM is compatible with table creation syntax of MySQL databases and adds keyword **partition_options** for databases and tables sharding.

```
CREATE TABLE [IF NOT EXISTS] tbl_name
(create_definition,...)
[table_options]
[partition_options]
partition_options:
DBPARTITION BY
    {{RANGE|HASH|MOD_HASH|RIGHT_SHIFT|YYYYMM|YYYYWEEK|YYYYDD}}(column)}
[TPARTITION BY
    {{HASH|MOD_HASH|UNI_HASH|RIGHT_SHIFT|YYYYMM|YYYYWEEK|YYYYDD}}(column)}
[TPARTITIONS num]
]
```

12.2.3 Sharding Algorithms

12.2.3.1 MOD_HASH

Instructions

The sharding key must be the numeric data type (INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL, or NUMERIC) or the string data type.

Data Routing

The data route generally depends on the remainder of the sharding key value divided by database shards. If the value is a string, convert the string into a hashed value and divide the value by database shards or table shards to calculate the remainder.

For example, `MOD_HASH('6')` is equivalent to $6\%D$. D is the number of shards.

If the same key is used for database and table sharding, the algorithm calculates the remainder by the total number of table shards.

For example, if there are 2 database shards and each shard has 4 table shards, table shards 0 to 3 are stored on shard 0, and table shards 4 to 7 are stored on database shard 1. If the sharding key value is 15, data is distributed to table shard 7 on database shard 1 according to $15\% (2 \times 4) = 7$.

Calculation Method

Method 1: Use an Integer as the Sharding Key

Table 12-4 Required calculation methods

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Database routing result = Database sharding key value % Database shards Table routing result = Table sharding key value % Table shards	Database shard: $16 \% 8 = 0$ Table shard: $16 \% 3 = 1$
Database sharding key = Table sharding key	Table routing result = Sharding key value % (Database shards x Table shards) Database routing result = Table routing result / Table shards	Table shard: $16 \% (8 \times 3) = 16$ Database shard: $16 / 3 = 5$

Method 2: Use a String as the Sharding Key

Table 12-5 Required calculation methods

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Database routing result = <code>hash(Database sharding key value) % Database shards</code> Table routing result = <code>hash(Table sharding key value) % Table shards</code>	<code>hash('abc') = 'abc'.hashCode()=96354</code> Database shard: $96354 \% 8 = 2$; Table shard: $96354 \% 3 = 0$;

Condition	Calculation Method	Example
Database sharding key = Table sharding key	Table routing result = $\text{hash}(\text{Sharding key value}) \% (\text{Database shards} \times \text{Table shards})$ Database routing result = $\text{Table routing result} / \text{Table shards}$	$\text{hash}('abc') = 'abc'.\text{hashCode}()=96354$ Table shard: $96354 \% (8 * 3) = 18$ Database shard: $18 / 3=6$

Syntax for Creating Tables

```
create table mod_hash_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by mod_hash (ID)
tbpartition by mod_hash (ID) tpartitions 6;
```

Precautions

- The sharding key and its value cannot be modified.
- The MOD_HASH algorithm is a simple way to find the remainder of the sharding key value divided by shards. This algorithm features even distribution of sharding key values to ensure even results.

12.2.3.2 MOD_HASH_CI

Application Scenarios

This algorithm applies if you want to route data to different database shards by user ID or order ID.

Instructions

The sharding key must be CHAR, VARCHAR, INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, or DECIMAL (the precision can be 0).

Data Routing

The data route depends on the remainder of the sharding key value divided by database or table shards. MOD_HASH is case-sensitive, but MOD_HASH_CI is not.

Calculation Method

Method 1: Use an Integer as the Sharding Key

Table 12-6 Required calculation methods when the sharding key is the integer data type

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Database routing result = Database sharding key value % Database shards Table routing result = Table sharding key value % Table shards	Database shard: $16 \% 8 = 0$ Table shard: $16 \% 3 = 1$
Database sharding key = Table sharding key	Table routing result = Sharding key value % (Database shards x Table shards) Database routing result = Table routing result / Table shards	Table shard: $16 \% (8 \times 3) = 16$ Database shard: $16 / 3 = 5$

Method 2: Use a String as the Sharding Key

Table 12-7 Required calculation methods when the sharding key is the string data type

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Database routing result = hash(Database sharding key value) % Database shards Table routing result = hash(Table sharding key value) % Table shards	hash('abc') = 'abc'.toUpperCase().hashCode()=64578 Database shard: $64578 \% 8 = 2$; Table shard: $64578 \% 3 = 0$;
Database sharding key = Table sharding key	Table routing result = hash(Sharding key value) % (Database shards x Table shards) Database routing result = Table routing result / Table shards	hash('abc') = 'abc'.toUpperCase().hashCode()=64578 Table shard: $64578 \% (8 \times 3) = 18$ Database shard: $18 / 3 = 6$

Syntax for Creating Tables

- Assume that you use field **ID** as the sharding key to shard databases based on MOD_HASH_CI:

```
create table mod_hash_ci_tb(
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8 dbpartition by mod_hash_ci(ID);
```

- Assume that you use field **ID** as the sharding key to shard databases and tables based on MOD_HASH_CI:

```
create table mod_hash_ci_tb(
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by mod_hash_ci(ID)
tbpartition by mod_hash_ci(ID) tpartitions 4;
```

Precautions

- The MOD_HASH_CI algorithm is a simple way to find the remainder of the sharding key value divided by shards. This algorithm features even distribution of sharding key values to ensure even results.

12.2.3.3 UNI_HASH

Application Scenarios

This algorithm applies if you want to route data to different shards by user ID or order ID.

Instructions

The sharding key must be the numeric data type (INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL, or NUMERIC) or the string data type.

Data Routing

The data route depends on the remainder of the sharding key values divided by the number of database shards or table shards. If the value is a string, convert the string into a hashed value and use the value to calculate remainders.

For example, UNI_HASH('6') is equivalent to $6\%D$. D is the number of shards. This example only applies to the sharding key value that is an integer.

Calculation Method

Method 1: Use an Integer as the Sharding Key

Table 12-8 Required calculation methods

Condition	Calculation Method	Example
Database sharding key ≠ Table sharding key	Database routing result = Database sharding key value % Database shards Table routing result = Table sharding key value % Table shards	Database shard: $16 \% 8 = 0$ Table shard: $16 \% 3 = 1$

Condition	Calculation Method	Example
Database sharding key = Table sharding key	Database routing result = Sharding key value % Database shards Table routing result = (Sharding key value % Database shards) x Table shards + (Sharding key value / Database shards) % Table shards	Database shard: $16 \% 8 = 0$ Table shard: $(16 \% 8) \times 3 + (16 / 8)\%3 = 2$

Method 2: Use a String as the Sharding Key

Table 12-9 Required calculation methods

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Database routing result = $\text{hash}(\text{Database sharding key value}) \% \text{Database shards}$ Table routing result = $\text{hash}(\text{Table sharding key value}) \% \text{Table shards}$	$\text{hash}('abc') = 'abc'.\text{hashCode}()=96354$ Database shard: $96354 \% 8 = 2$; Table shard: $96354 \% 3 = 0$;
Database sharding key = Table sharding key	Database routing result = $\text{hash}(\text{Sharding key value}) \% \text{Database shards}$ Table routing result = $\{\text{hash}(\text{Sharding key value}) \% \text{Database shards}\} \times \text{Table shards} + \{\text{hash}(\text{Sharding key value}) / \text{Database shards}\} \% \text{Table shards}$	Database shard: $96354 \% 8 = 2$ Table shard: $:(96354 \% 8) \times 3 + (96354 / 8)\%3 = 8$

Syntax for Creating Tables

```
create table uni_hash_tb(
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by UNI_HASH(ID)
tbpartition by UNI_HASH(name) tpartitions 3;
```

Precautions

- The sharding key and its value cannot be modified.
- The UNI_HASH algorithm is a simple way to obtain the remainder of the sharding key divided by the number of shards. This algorithm features even distribution of sharding key values to ensure even results.

Comparison with MOD_HASH

- When you use UNI_HASH to shard databases, UNI_HASH applies the same data routing method as MOD_HASH, that is, shard databases depending on the remainder of the sharding key value divided by database shards.
- If you use MOD_HASH and the same sharding key to shard databases and tables, the database sharding result changes as you specify different table shards per database shard.
- If you use UNI_HASH and the same sharding key to shard databases and tables, the database sharding result is fixed whatever table shards per database shard you specify.
- If two logical tables with different table shards use the same sharding key to shard databases and tables, a cross-shard JOIN operation is required to join the two logical tables when MOD_HASH is used, but not when UNI_HASH is used.

12.2.3.4 RIGHT_SHIFT

Application Scenarios

This algorithm applies if a large difference appears in high-digit part but a small difference in low-digit part of sharding key values. Using this algorithm ensures uniform distribution of remainders calculated from sharding key values. Therefore, data is evenly routed to different shards.

Instructions

The sharding key must be the numeric data type (INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL, or NUMERIC).

Data Routing

The data route depends on the remainder of the changed sharding key value divided by the number of shards or sharded tables. To change the sharding key value, you need to convert the value into a binary number and right shift its bits to gain a new binary number. The number of moved bits is specified in DDL statements. Then, convert the new binary number into a decimal number. This decimal number is the changed sharding key value.

Calculation Method

Table 12-10 Required calculation methods

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Database routing result = Database sharding key value % Database shards Table routing result = Table sharding key value % Table shards	Database shard: $(123456 \gg 4) \% 8 = 4$ Table shard: $(123456 \gg 4) \% 3 = 0$
Database sharding key = Table sharding key	Database routing result = Sharding key value % Database shards Table routing result = (Sharding key value % Database shards) x Table shards + (Sharding key value / Database shards) % Table shards	Database shard: $(123456 \gg 4) \% 8 = 4$ Table table: $((123456 \gg 4) \% 8) * 3 + ((123456 \gg 4) / 8) \% 3 = 13$

Syntax for Creating Tables

```
create table RIGHT_SHIFT (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by RIGHT_SHIFT(id, 4)
tbpartmention by RIGHT_SHIFT(id, 4) tpartitions 2;
```

Precautions

- The sharding key and its value cannot be modified.
- The number of shifts cannot exceed the number of bits occupied by the integer type. If it exceeds, data will be routed to shard 0.

12.2.3.5 YYYYMM

Application Scenarios

This algorithm applies when data is routed to shards by year and month.

Instructions

The data type of the sharding key value is **DATE**, **DATETIME**, or **TIMESTAMP**.

Data Routing

The data route depends on the remainder of the hash value divided by the number of shards. Enter the year and month into the hash function to obtain the hash value.

For example, YYYYMM ('2018-12-31 12:12:12') is equivalent to $(2018 \times 12 + 12) \% D$. D is the number of shards.

Calculation Method

Table 12-11 Required calculation methods

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Sharding key: yyyy-MM-dd Database routing result = $(yyyy \times 12 + MM) \% \text{Database shards}$ Table routing result = $(yyyy \times 12 + MM) \% \text{Table shards}$	Sharding key: 2012-11-20 Database shard: $(2012 \times 12 + 11) \% 8 = 3$ Table shard: $(2012 \times 12 + 11) \% 3 = 2$
Database sharding key = Table sharding key	Sharding key: yyyy-MM-dd Table routing result = $(yyyy \times 12 + MM) \% (\text{Database shards} \times \text{Table shards})$ Database routing result = $\text{Table routing result} / \text{Table shards}$	Sharding key: 2012-11-20 Table shard: $(2012 \times 12 + 11) \% (8 \times 3) = 11$ Database shard: $11 \% 3 = 2$

Syntax for Creating Tables

```
create table yyyyymm_tb(
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  update_time datetime DEFAULT NULL,
  primary key(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by YYYYMM(create_time)
tbpartition by YYYYMM(update_time) tpartitions 12;
```

Precautions

- The sharding key and its value cannot be modified.
- This YYYYMM algorithm does not apply if each month of a year corresponds to one shard.
- Data of the same month in different years may be routed to the same shard.

12.2.3.6 YYYYDD

Application Scenarios

This algorithm applies when data is routed to shards by the year and the day of the year specified in the sharding key value.

Instructions

The data type of the sharding key value is **DATE**, **DATETIME**, or **TIMESTAMP**.

Data Routing

The data route depends on the remainder of the hash value divided by the number of shards or sharded tables. Use the hash function and enter the year and the day of the year specified in the sharding key value to calculate the hash value.

For example, `YYYYDD('2018-12-31 12:12:12')` is equivalent to $(2018 \times 366 + 365) \% D$. D is the number of shards.

NOTE

2018-12-31 is the 365th day of 2018. In the formula, 366 is a fixed value.

Calculation Method

Table 12-12 Required calculation methods

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Sharding key: yyyy-MM-dd Database routing result = $(yyyy \times 366 + \text{Day of the current year}) \% \text{Database shards}$ Table routing result = $(yyyy \times 366 + \text{Day of the current year}) \% \text{Table shards}$	Sharding key: 2012-12-31 Database shard: $(2012 \times 366 + 366) \% 8 = 6$ Table shard: $(2012 \times 366 + 366) \% 3 = 0$
Database sharding key = Table sharding key	Sharding key: yyyy-MM-dd Table routing result = $(yyyy \times 366 + \text{Day of the current year}) \% (\text{Database shards} \times \text{Table shards})$ Database routing result = $\text{Table routing result} / \text{Table shards}$	Sharding key: 2012-12-31 Database shard: $(2012 \times 366 + 366) \% (8 \times 3) = 6$ Database shard: $6 / 3 = 2$

Syntax for Creating Tables

```
create table yyyydd_tb(
  id int,
  name varchar(30) DEFAULT NULL,
```

```
create_time datetime DEFAULT NULL,  
update_time datetime DEFAULT NULL,  
primary key(id)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8  
dbpartition by YYYYDD(create_time)  
tbpartition by YYYYDD(update_time) tpartitions 30;
```

Precautions

- The sharding key and its value cannot be modified.
- The YYYYDD algorithm does not apply if each day of a year corresponds to one shard.
- Data of the same day in different years may be routed to the same shard.

12.2.3.7 YYYYWEEK

Application Scenarios

This algorithm applies when data is routed to shards by the year and the week of the year specified in the sharding key value.

Instructions

The data type of the sharding key value is **DATE**, **DATETIME**, or **TIMESTAMP**.

Data Routing

The data route depends on the remainder of the hash value divided by the number of shards or sharded tables. Use the hash function and enter the year and the week of the year specified in the sharding key value to calculate the hash value.

For example, `YYYYWEEK('2018-12-31 12:12:12')` is equivalent to $(2019 \times 54 + 1) \% D$. `D` is the number of shards.

NOTE

2018-12-31 is the first week of 2019. In the formula, 54 is a fixed value.

Calculation Method

Table 12-13 Required calculation methods

Condition	Calculation Method	Example
Database sharding key \neq Table sharding key	Sharding key: yyyy-MM-dd Database routing result = $(yyyy \times 54 + \text{Week of the current year}) \% \text{Database shards}$ Table routing result = $(yyyy \times 54 + \text{Week of the current year}) \% \text{Table shards}$	Sharding key: 2012-12-31 Database shard: $(2013 \times 54 + 1) \% 8 = 7$ Table shard: $(2013 \times 54 + 1) \% 3 = 1$
Database sharding key = Table sharding key	Sharding key: yyyy-MM-dd Table routing result = $(yyyy \times 54 + \text{Week of the current year}) \% (\text{Database shards} \times \text{Table shards})$ Database routing result = $\text{Table routing result} / \text{Table shards}$	Sharding key: 2012-12-31 Database shard: $(2013 \times 54 + 1) \% (8 \times 3) = 7$ Database shard: $7 / 3 = 2$

Syntax for Creating Tables

```
create table yyyyweek_tb(
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  update_time datetime DEFAULT NULL,
  primary key(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by YYYYWEEK(create_time)
tbpartition by YYYYWEEK(update_time) tpartitions 54;
```

Precautions

- The sharding key and its value cannot be modified.
- The YYYYWEEK algorithm does not apply if each week of a year corresponds to one shard.
- Data of the same week in different years may be routed to the same shard.

12.2.3.8 HASH

Application Scenarios

This algorithm features even distribution of data or sharding tables by year, month, week, day, and a combination of them. Arithmetic operators such as equality (=) and IN operators are often used in SQL queries.

Instructions

A sharding key is generally the table field or the table field+date function. If the sharding key is the table field+date function, the table field must be **DATE**, **DATETIME**, or **TIMESTAMP**. The date function applies to routing data to different shards by year, month, week, day, or a combination of them.

Data Routing

Determine the range of each database or table shard.

For example, if there are 8 shards in each schema, use formula $102400/8=12800$ to calculate the range of each shard as follows: 0=0–12799, 1=12800–25599, 2=25600–38399, 3=38400–51199, 4=51200–63999, 5=64000–76799, 6=76800–89599, and 7=89600–102399.

To determine the route, you need to first calculate CRC32 value based on the sharding key value and divide CRC value by 102400. Then check which range the remainder belongs to.

Calculation Method

Method 1: Use a Non-date Sharding Key

Table 12-14 Required calculation methods

Condition	Calculation Method	Example
Non-date sharding key	Database routing result = $\text{crc32}(\text{Database sharding key}) \% 102400$ Table routing result = $\text{crc32}(\text{Table sharding key}) \% 102400$	Database/Table shard: $\text{crc32}(16) \% 102400 = 49364$; 49364 belongs to range 3=38400-51199, so data is routed to shard 3.

Method 2: Use a Date Sharding Key

Table 12-15 Supported date functions

Date Function	Calculation Method	Example
year()	$\text{year}(\text{yyyy-MM-dd})=\text{yyyy}$	$\text{year}('2019-10-11')=2019$
month()	$\text{month}(\text{yyyy-MM-dd})=\text{MM}$	$\text{month}('2019-10-11')=10$
weekofyear()	$\text{weekofyear}(\text{yyyy-MM-dd})=\text{Week of the current year}$	$\text{weekofyear}('2019-10-11')=41$
day()	$\text{day}(\text{yyyy-MM-dd})=\text{Day of the current month}$	$\text{day}('2019-10-11')=11$

Table 12-16 Required calculation methods

Condition	Calculation Method	Example
Date sharding key	Database routing result = crc32(Date function(Database sharding key)) % 102400 Table routing result = crc32(Date function(Database sharding key)) % 102400	Database/Table shard: crc32(year('2019-10-11')) % 102400 = 5404; 5404 belongs to range 0=0-12799, so data is routed to shard 0.

Syntax for Creating Tables

Assume that you use field ID as the sharding key and the HASH algorithm to shard databases:

```
create table hash_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash (ID);
```

Assume that you use field ID as the sharding key and the hash algorithm to shard databases and tables:

```
create table mod_hash_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by hash (ID)
tbpartmention by hash (ID) tbpartitions 4;
```

Precautions

The sharding key and its value cannot be modified.

12.2.3.9 Range

Application Scenarios

This algorithm applies to routing data in different ranges to different shards. Less-than signs (<), greater-than signs (>), and BETWEEN ... AND ... are frequently used in SQL queries.

Instructions

A sharding key can only be an integer, a date, or used in combination with a date function. If a date function is used, the sharding key must be of the DATE, DATETIME, or TIMESTAMP type.

A sharding key is generally the table field or the table field+date function. If the sharding key is the table field+date function, the table field must be **DATE**, **DATETIME**, or **TIMESTAMP**. The date function applies to routing data to different shards by year, month, week, day, or a combination of them.

Data Routing

Data is routed to different shards by the sharding key value based on algorithm metadata rules.

Calculation Method

Method 1: Use an Integer as the Sharding Key

Table 12-17 Required calculation methods

Condition	Calculation Method	Example
Integer sharding keys	<p>Database routing result: Data is routed to different shards based on the sharding key and the preset metadata range.</p> <p>Table routing result: Data is routed to different sharded tables based on the sharding key and the preset metadata range.</p>	<p>Data is routed to shard1 if the sharding key value is 3 and the preset metadata range is 3-4.</p> <p>Data is routed to shard2 if the sharding key value is 5 and the preset metadata range is 5-6.</p>

Method 2: Use a Date as the Sharding Key

Table 12-18 Supported date functions

Date Function	Calculation Method	Example
year()	year(yyyy-MM-dd)=yyyy	year('2019-10-11')=2019
month()	month(yyyy-MM-dd)=MM	month('2019-10-11')=10
weekofyear()	weekofyear(yyyy-MM-dd)=Week of the current year	weekofyear('2019-10-11')=41
day()	day(yyyy-MM-dd)=Day of the current month	day('2019-10-11')=11

Table 12-19 Required calculation methods

Condition	Calculation Method	Example
Date sharding key	<p>Database routing: Data is routed to different database shards based on the date function (database sharding key value) and the preset metadata range.</p> <p>Table routing: Data is routed to different table shards based on the date function (sharding key value) and the preset metadata range.</p>	Data is routed to shard4 based on the metadata range 9–10 when the sharding key value is 10: month(2019-10-11)=10 belongs to 9–10=4.

Syntax for Creating Tables

```
create table range_tb(
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
)
dbpartition by range(id)
{
  1-2=0,
  3-4=1,
  5-6=2,
  7-8=3,
  9-10=4,
  11-12=5,
  13-14=6,
  default=7
};
```

Precautions

- The sharding key and its value cannot be modified.
- The range algorithm can be used only for database sharding.

12.2.3.10 MM

Application Scenarios

This algorithm applies if you want to shard data by month. One table shard for one month is recommended.

Instructions

- The sharding key must be **DATE**, **DATETIME**, or **TIMESTAMP**.
- This algorithm can be used only for table sharding, instead of database sharding.

Data Routing

Use the month number in the sharding key value to find the remainder. This remainder determines which table shard your data is routed to and serves as the name suffix of each table shard.

For example, if the sharding key value is **2019-01-15**, the calculation of the table shard is: Month mod Shards or $1 \text{ mod } 12 = 1$.

Calculation Method

Table 12-20 Required calculation methods

Condition	Calculation Method	Example
None	Table routing result = Table sharding key value % Table shards	Sharding key value: 2019-01-15 Table shard: $1 \text{ mod } 12 = 1$

Syntax for Creating Tables

```
create table test_mm_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH(id)
tbpartition by MM(create_time) tpartitions 12;
```

Precautions

Table shards in each database shard cannot exceed 12 because there are only 12 months a year.

12.2.3.11 DD

Application Scenarios

This algorithm applies if you want to shard data by date. One table shard for one day is recommended.

Instructions

- The sharding key must be **DATE**, **DATETIME**, or **TIMESTAMP**.
- This algorithm can be used only for table sharding, instead of database sharding.

Data Routing

Use the day number in the sharding key value to find the remainder. This remainder determines which table shard your data is routed to and serves as the name suffix of the table shard.

For example, if the sharding key value is **2019-01-15**, the calculation of the table shard is: Day number in a month mod Table shards, that is, $15 \text{ mod } 31 = 15$.

Calculation Method

Table 12-21 Required calculation methods

Condition	Calculation Method	Example
None	Table routing result = Table sharding key value % Table shards	Sharding key value: 2019-01-15 Table shard: $15 \text{ mod } 31 = 15$

Syntax for Creating Tables

```
create table test_dd_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH(id)
tbpartition by DD(create_time) tpartitions 31;
```

Precautions

Table shards in each database shard cannot exceed 31 because there are at most 31 days in a month.

12.2.3.12 MMDD

Application Scenarios

This algorithm applies when you want to shard data by day in a year. One table shard for one day is recommended.

Instructions

- The sharding key must be **DATE**, **DATETIME**, or **TIMESTAMP**.
- This algorithm can be used only for table sharding, instead of database sharding.

Data Routing

Use the day number of a year in the sharding key value to find the remainder. This remainder determines which table shard your data is routed to and serves as the name suffix of each table shard.

For example, if the sharding key value is **2019-01-15**, the calculation of the table shard is: Day number in a year mod Table shards, that is, $15 \text{ mod } 366 = 15$.

Calculation Method

Table 12-22 Required calculation methods

Condition	Calculation Method	Example
None	Table routing result = Table sharding key value % Table shards	Sharding key value: 2019-01-15 Table shard: $15 \% 366 = 15$

Syntax for Creating Tables

```
create table test_mmdd_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH(name)
tbpartition by MMDD(create_time) tpartitions 366;
```

Precautions

Table shards in each database shard cannot exceed 366 because there are at most 366 days in a year.

12.2.3.13 WEEK

Application Scenarios

This algorithm applies when you want to shard data by day in a week. One table shard for one day is recommended.

Instructions

- The sharding key must be **DATE**, **DATETIME**, or **TIMESTAMP**.
- This algorithm can be used only for table sharding, instead of database sharding.

Data Routing

Use the day number of a week in the sharding key value to find the remainder. This remainder determines which table shard your data is routed to and serves as the name suffix of each table shard.

For example, if the sharding key value is **2019-01-15**, the calculation of the table shard is: Day number in a week mod Table shards, that is, 3 mod 7 = 3.

Calculation Method

Table 12-23 Required calculation methods

Condition	Calculation Method	Example
None	Table routing result = Table sharding key value % Table shards	Sharding key value: 2019-01-15 Table shard: 3 mod 7= 3

Syntax for Creating Tables

```
create table test_week_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by HASH(name)
tbpartition by WEEK(create_time) tpartitions 7;
```

Precautions

Table shards in each database shard cannot exceed 7 because there are 7 days in a week.

12.3 DML

Unsupported DML Statements

Table 12-24 Syntax restrictions on DML

DML Syntax	Restriction
DELETE statement	PARTITION clauses are not supported.
UPDATE statement	<ul style="list-style-type: none"> • Cross-shard subquery is not supported. • Updating sharding keys is not supported.
SELECT statement	User-defined sequencing similar to ORDER BY FIELD(id,1,2,3) is not supported.

DML Syntax	Restriction
System database query	<p>The following system database queries are supported:</p> <pre>SELECT version()</pre> <ul style="list-style-type: none"> information_schema.SCHEMA_PRIVILEGES information_schema.TABLE_PRIVILEGES information_schema.USER_PRIVILEGES information_schema.SCHEMATA information_schema.tables information_schema.columns <pre>SHOW KEYS FROM `table` FROM `database`</pre> <p>NOTE</p> <ul style="list-style-type: none"> Supported operators include =, IN, and LIKE. These operators can be associated using AND. Complex queries such as subquery, JOIN, sorting, aggregate query, and LIMIT are not supported. information_schema.tables and information_schema.columns support operators < and >.

12.3.1 INSERT

INSERT is used to insert data into database objects.

Common Syntax

```
INSERT [INTO] tbl_name
[(col_name,...)]
{VALUES | VALUE} ({expr },...),(...),...
[ ON DUPLICATE KEY UPDATE
col_name=expr
[, col_name=expr] ... ]
OR
INSERT [INTO] tbl_name
SET col_name={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
col_name=expr [, col_name=expr] ... ]
```

Syntax Restrictions

- INSERT DELAYED... is not supported.
- Only INSERT statements that contain sharding fields are supported.
- PARTITION syntax is not supported. Partitioned tables are not recommended.

12.3.2 REPLACE

REPLACE is used to insert rows into or replace rows in a table.

Common Syntax

```
replace into table(col1,col2,col3)
values(value1,value2,value3)
```

Syntax Restrictions

- PARTITION syntax is not supported.
- If an auto-increment table has no ID, you can insert a data record with a specified ID using REPLACE, but no ID is generated.

12.3.3 DELETE

DELETE is used to delete rows that meet conditions from a table.

Common Syntax

```
DELETE [IGNORE]
FROM tbl_name [WHERE where_condition]
```

Syntax Restrictions

- The WHERE clause does not support subqueries, including correlated subqueries and non-correlated subqueries.
- Data in reference tables cannot be deleted when multiple tables are deleted.

12.3.4 UPDATE

Common Syntax

```
UPDATE table_reference
SET col_name1={expr1} [, col_name2={expr2}] ...
[WHERE where_condition]
```

Syntax Restrictions

- Subqueries are not supported, including correlated subqueries and non-correlated subqueries.
- The WHERE condition in the UPDATE statement does not support arithmetic expressions and their subqueries.
- Modifying reference tables is not supported during update of multiple tables.
- Updating the sharding key field of a logical table is not supported. Updating the sharding key field may cause data redistribution.

12.3.5 SELECT

SELECT is used to query data in one or more tables.

Common Syntax

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
select_expr
[, select_expr ...]
[FROM table_references [WHERE where_condition]
[GROUP BY {col_name | expr | position} [ASC | DESC], ...]
[HAVING where_condition] [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Table 12-25 Supported syntax

Syntax	Description
select_expr	Indicates a column that you want to query.
FROM table_references	Indicates the table or tables that you want to query.
WHERE	Followed by an expression to filter for rows that meet certain criteria.
GROUP BY	Groups the clauses used in SQL in sequence. GROUP BY indicates relationships between statements and supports column names. For example, the HAVING clause must be placed after the GROUP BY clause and before the ORDER BY clause.
ORDER BY	Indicates relationships between statements and supports column names and specified sorting orders such as ASC or DESC.
LIMIT/OFFSET	Restrains the offset and size of output result sets, for example, one or two values can be input after LIMIT.

Syntax Description

- Using an empty string as an alias is not supported.
- SELECT ... GROUP BY ... WITH ROLLUP is not supported.
- Neither STRAIGHT_JOIN nor NATURAL JOIN is supported.
- The SELECT FOR UPDATE statement supports only simple queries and does not support JOIN, GROUP BY, ORDER BY, or LIMIT.
- Each SELECT statement in UNION does not support multiple columns with the same name, for example,
SELECT id, id, name FROM t1 UNION SELECT pk, pk, name FROM t2 is not supported because this statement has duplicate column names.

12.3.6 SELECT JOIN Syntax

Common Syntax

table_references:

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

table_reference:

```
table_factor | join_table
```

table_factor:

```
tbl_name [[AS] alias]
| table_subquery [AS] alias
| ( table_references )
```

join_table:


```
table_reference [INNER | CROSS] JOIN table_factor [join_condition]
| table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
| table_reference [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

join_condition:

```
ON conditional_expr
| USING (column_list)
```

Syntax Restrictions

SELECT STRAIGHT_JOIN and NATURAL JOIN are not supported.

Example

```
select id,name from test1 where id=1;
select distinct id,name from test1 where id>=1;
select id,name from test1 order by id limit 2 offset 2;
select id,name from test1 order by id limit 2,2;
select 1+1,'test',id,id*1.1,now() from test1 limit 3;
select current_date,current_timestamp;
select abs(sum(id)) from test1;
```

12.3.7 SELECT UNION Syntax

Common Syntax

```
SELECT ...UNION [ALL | DISTINCT]
SELECT ...[UNION [ALL | DISTINCT] SELECT ...]
```

Example

```
select userid from user union select orderid from ordertbl order by userid;
select userid from user union (select orderid from ordertbl group by orderid) order by userid;
```

Syntax Restrictions

SELECT statements in UNION do not support multiple columns with the same name.

12.3.8 SELECT Subquery Syntax

Subquery as Scalar Operand

Example

```
SELECT (SELECT id FROM test1 where id=1);
SELECT (SELECT id FROM test2 where id=1)FROM test1;
SELECT UPPER((SELECT name FROM test1 limit 1)) FROM test2;
```

Comparisons Using Subqueries

Syntax

```
non_subquery_operand comparison_operator (subquery)
comparison_operator: = > < >= <= <> != <=> like
```

Example

```
select name from test1 where id > (select id from test2 where id=1);
select name from test1 where id = (select id from test2 where id=1);
select id from test1 where name like (select name from test2 where id=1);
```

Subqueries with ANY, IN, NOT IN, SOME, ALL, Exists, NOT Exists

Syntax

```
operand comparison_operator SOME (subquery)
operand comparison_operator ALL (subquery)
operand comparison_operator ANY (subquery)
operand IN (subquery)
operand not IN (subquery)
operand exists (subquery)
operand not exists (subquery)
```

Example

```
select id from test1 where id > any (select id from test2);
select id from test1 where id > some (select id from test2);
select id from test1 where id > all (select id from test2);
select id from test1 where id in (select id from test2);
select id from test1 where id not in (select id from test2);
select id from test1 where exists (select id from test2 where id=1);
select id from test1 where not exists (select id from test2 where id=1);
```

Derived Tables (Subqueries in the FROM Clause)

Syntax

```
SELECT ... FROM (subquery) [AS] tbl_name ...
```

Example

```
select id from (select id,name from test2 where id>1) a order by a.id;
```

Syntax Restrictions

- Each derived table must have an alias.
- A derived table cannot be a correlated subquery.
- In some cases, correct results cannot be obtained using a scalar subquery. Using JOIN instead is recommended to improve query performance.
- Using subqueries in the HAVING clause and the JOIN ON condition is not supported.
- Row subqueries are not supported.

12.4 Functions

Supported Functions

Table 12-26 Operator functions

Expression	Example
IN	SELECT * FROM Products WHERE vendor_id IN ('V000001', 'V000010') ORDER BY product_price

Expression	Example
NOT IN	SELECT product_id, product_name FROM Products WHERE NOT vendor_id IN ('V000001', 'V000002') ORDER BY product_id
BETWEEN	SELECT id, product_id, product_name, product_price FROM Products WHERE id BETWEEN 000005 AND 000034 ORDER BY id
NOT...BETWEEN	SELECT product_id, product_name FROM Products WHERE NOT vendor_id BETWEEN 'V000002' and 'V000005' ORDER BY product_id
IS NULL	SELECT product_name FROM Products WHERE product_price IS NULL
IS NOT NULL	SELECT id, product_name FROM Products WHERE product_price IS NOT NULL ORDER BY id
AND	SELECT * FROM Products WHERE vendor_id = 'V000001' AND product_price <= 4000 ORDER BY product_price
OR	SELECT * FROM Products WHERE vendor_id = 'V000001' OR vendor_id = 'V000009'
NOT	SELECT product_id, product_name FROM Products WHERE NOT vendor_id = 'V000002'
LIKE	SELECT * FROM Products WHERE product_name LIKE 'NAME %' ORDER BY product_name
NOT LIKE	SELECT * FROM Products WHERE product_name NOT LIKE 'NAME%' ORDER BY product_name
CONCAT	SELECT product_id, product_name, Concat(product_id , '(', product_name ,')') AS product_test FROM Products ORDER BY product_id
+	SELECT 3 * 2+5-100/50
-	SELECT 3 * 2+5-100/50
*	SELECT order_num, product_id, quantity, item_price, quantity*item_price AS expanded_price FROM OrderItems WHERE order_num BETWEEN 000009 AND 000028 ORDER BY order_num
/	SELECT 3 * 2+5-100/50
UPPER	SELECT id, product_id, UPPER(product_name) FROM Products WHERE id > 10 ORDER BY product_id
LOWER	SELECT id, product_id, LOWER(product_name) FROM Products WHERE id <= 10 ORDER BY product_id
SOUNDEX	SELECT * FROM Vendors WHERE SOUNDEX(vendor_name) = SOUNDEX('Huawee') ORDER BY vendor_name

Expression	Example
IFNULL	<p>SELECT IFNULL(product_id, 0) FROM Products;</p> <p>NOTE</p> <ul style="list-style-type: none"> For DDM instances created before March 20, sharded tables do not support the calling of functions nested in the IFNULL and aggregation functions. For example, if you execute function select IFNULL(sum(yan),0) from shenhai, the result differs from the expected result. For DDM instances created after March 20, sharded tables support only the calling of functions nested in the IFNULL and aggregation functions.

Table 12-27 Time and date functions

Function Expression	Example
DAY()	<p>SELECT * FROM TAB_DT WHERE DAY(dt)=21</p> <p>SELECT * FROM TAB_DT WHERE dt='2018-12-21'</p> <p>INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')</p>
MONTH()	<p>SELECT * FROM TAB_DT WHERE MONTH(dt)=12</p> <p>SELECT * FROM TAB_DT WHERE dt='2018-12-21'</p> <p>INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')</p>
YEAR()	<p>SELECT * FROM TAB_DT WHERE YEAR(dt)=2018</p> <p>SELECT * FROM TAB_DT WHERE dt='2018-12-21'</p> <p>INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')</p>
DAYOFYEAR()	<p>SELECT * FROM TAB_DT WHERE DAYOFYEAR(dt)=365</p> <p>SELECT * FROM TAB_DT WHERE dt='2018-12-31'</p> <p>INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')</p>
DAYOFWEEK()	<p>SELECT * FROM TAB_DT WHERE DAYOFWEEK(dt)=6</p> <p>SELECT * FROM TAB_DT WHERE dt='2018-12-21'</p> <p>INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')</p>
WEEKOFYEAR()	<p>SELECT * FROM TAB_DT WHERE WEEKOFYEAR(dt)=51</p> <p>SELECT * FROM TAB_DT WHERE dt='2018-12-21'</p> <p>INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')</p>

Table 12-28 Mathematical functions

Expression	Example
SQRT()	SELECT id, product_price, SQRT(product_price) AS price_sqrt FROM Products WHERE product_price < 4000 ORDER BY product_price
AVG()	SELECT AVG(product_price) AS avg_product FROM Products
COUNT()	SELECT COUNT(*) AS num_product FROM Products
MAX()	SELECT id, product_id, product_name, MAX(product_price) AS max_price FROM Products ORDER BY id
MIN()	SELECT id, product_id, product_name, MIN(product_price) AS min_price FROM Products ORDER BY id
SUM()	SELECT SUM(product_price) AS sum_product FROM Products

Unsupported Functions

Table 12-29 Function restrictions

Item	Restriction
Functions	Function row_count() is not supported.

12.5 Unsupported Items

- Triggers
- Temporary tables
- DO statement
- Association with foreign keys
- RESET statement
- FLUSH statement
- BINLOG statement
- HANDLER statement
- SHOW WARNINGS statement
- Assignment operator :=
- Operators less than (<), assignment (=), and more than (>)
- Expression IS UNKNOWN
- Statements for managing users and permissions
- INSTALL and UNINSTALL PLUGIN statements
- Cross-shard stored procedures and custom functions
- Statements for modifying database names, table names, and sharding field names and types

- Most of SHOW statements such as SHOW PROFILES and SHOW ERRORS
- Table maintenance statements, including ANALYZE, CHECK, CHECKSUM, OPTIMIZE, and REPAIR TABLE
- Statements for assigning a value to or querying variable **session**, for example, set @rowid=0;select @rowid:=@rowid+1,id from user
- SQL statements that use -- or /*...*/ to comment out a single line or multiple lines of code

Supported Permission Levels

- Global level (not supported)
- Database level (supported)
- Table level (supported)
- Column level (not supported)
- Subprogram level (not supported)

12.6 Supported SQL Statements

12.6.1 CHECK TABLE

12.6.1.1 Checking DDL Consistency of Table Shards in All Logical Tables

Purpose: To view DDL consistency of all logical tables in one schema

Command Format:

```
check table
```

Command Output:

The following output is returned if DDL check results of all logical tables are consistent.

```
mysql> check table;
```

ID	DATABASE_NAME	TABLE_NAME	TABLE_TYPE	DDL_CONSISTENCY	TOTAL_COUNT	INCONSISTENT_COUNT	DETAILS
1	test	p1	SHARDING	Y	8	0	
2	test	b	BROADCAST	Y	8	0	
3	test	p2	SHARDING	Y	32	0	
4	test	s1	SINGLE	Y	1	0	
5	test	p20	SHARDING	Y	160	0	

5 rows in set (0.24 sec)

The following output is returned if there are logical tables with inconsistent DDL check results.

```
mysql> check table;
```

ID	DATABASE_NAME	TABLE_NAME	TABLE_TYPE	DDL_CONSISTENCY	TOTAL_COUNT	INCONSISTENT_COUNT	DETAILS
1	test	p2	SHARDING	N	32	2	'test_0004'.p2_0', 'test_0006'.p2_2'
2	test	p1	SHARDING	Y	8	0	
3	test	b	BROADCAST	Y	8	0	
4	test	s1	SINGLE	Y	1	0	
5	test	p20	SHARDING	Y	160	0	

5 rows in set (0.23 sec)

Output Details:

Each row indicates the check result of a logical table.

- **DATABASE_NAME**: indicates the schema name.
- **TABLE_NAME**: indicates the logical table name.
- **TABLE_TYPE**: indicates the logical table type.
 - **SINGLE**: indicates that the logical table is unsharded.
 - **BROADCAST**: indicates that the table is a broadcast table.
 - **SHARDING**: indicates that the table is sharded.
- **DDL_CONSISTENCY**: indicates whether DDL results of all table shards corresponding to the logical table are consistent.
- **TOTAL_COUNT**: indicates the number of table shards in the logical table.
- **INCONSISTENT_COUNT**: indicates the number of table shards with inconsistent DDL results.
- **DETAILS**: indicates names of the table shards with inconsistent DDL check results.

12.6.1.2 Checking DDL Consistency of Table Shards in One Logical Table

Purpose: to check DDL consistency of table shards in a specific logical table

Command format:

```
check table <table_name>
```

Command Output:

If the returned result set is empty, DDL results of table shards in this logical table are consistent.

```
mysql> check table p1;  
Empty set (0.02 sec)
```

If the returned result set is not empty, there are table shards with inconsistent DDL results.

```
mysql> check table p2\G
***** 1. row *****
      ID: 1
      DATABASE_NAME: test_0006
      TABLE_NAME: p2_2
      TABLE_TYPE: SHARDING
      EXTRA_COLUMNS:
      MISSING_COLUMNS:
      DIFFERENT_COLUMNS:
      KEY_DIFF:
      ENGINE_DIFF:
      CHARSET_DIFF:
      COLLATE_DIFF:
      EXTRA_PARTITIONS:
      MISSING_PARTITIONS:
      DIFFERENT_PARTITIONS:
      EXTRA_INFO: TABLE NOT EXISTS
***** 2. row *****
      ID: 2
      DATABASE_NAME: test_0004
      TABLE_NAME: p2_0
      TABLE_TYPE: SHARDING
      EXTRA_COLUMNS:
      MISSING_COLUMNS: `id2` int(11) DEFAULT NULL
      DIFFERENT_COLUMNS:
      KEY_DIFF:
      ENGINE_DIFF:
      CHARSET_DIFF:
      COLLATE_DIFF:
      EXTRA_PARTITIONS:
      MISSING_PARTITIONS:
      DIFFERENT_PARTITIONS:
      EXTRA_INFO:
2 rows in set (0.03 sec)
```

Output Details:

Each row displays details of a table shard with inconsistent DDL results.

- **DATABASE_NAME:** indicates the database shard containing the table shard.
- **TABLE_NAME:** indicates the name of the table shard.
- **TABLE_TYPE:** indicates the type of the logical table that the table shard belongs to.
- **EXTRA_COLUMNS:** indicates extra columns in the table shard.
- **MISSING_COLUMNS:** indicates missing columns in the table shard.
- **DIFFERENT_COLUMNS:** indicates name and type columns whose attributes are inconsistent in the table shard.
- **KEY_DIFF:** indicates inconsistent indexes in the table shard.
- **ENGINE_DIFF:** indicates inconsistent engines in the table shard.
- **CHARSET_DIFF:** indicates inconsistent character sets in the table shard.
- **COLLATE_DIFF:** indicates inconsistent collations in the table shard.
- **EXTRA_PARTITIONS:** indicates extra partitions in the table shard. This field is only available to partitioned tables.

- **MISSING_PARTITIONS:** indicates missing partitions in the table shard. This field is only available to partitioned tables.
- **DIFFERENT_PARTITIONS:** indicates partitions with inconsistent attributes in the table shard. This field is only available to partitioned tables.
- **EXTRA_INFO:** indicates other information such as missing table shards.

12.6.2 SHOW RULE

Command Format:

show rule: used to view the sharding rule of each logical table in a certain schema.

show rule from *table_name*: displays the sharding rule of a specified logical table in a certain schema.

Command Output:

The following is an example output of **show rule**.

```
mysql> show rule.
```

ID	TABLE_NAME	BROADCAST	DB_PARTITION_KEY	DB_PARTITION_POLICY	DB_PARTITION_COUNT	DB_PARTITION_OFFSET	PARTITION_RANGE
			TB_PARTITION_KEY	TB_PARTITION_POLICY	TB_PARTITION_COUNT	TB_PARTITION_OFFSET	
0	history	0			1	1	
1	tbtest_hash_forerror	0			1	1	
2	single_table_1	0			1	1	
3	carrier	0			1	1	
4	employee3	0			1	1	
5	employee4	0			1	1	
6	supplier	1			1	8	
7	broadcast_table_1	1			1	8	
8	test5	1			1	8	
9	employee1	1			1	8	
10	category	1			1	8	
11	employee2	1			1	8	
12	range_id_dpt_1	0	id_col	range	1	8	0-10=0, 11-20=1, 21-30=2, 31-
13	mhash_bigint_dpt_1	0	bigint_col	mod_hash	1	8	40=3, 41-50=4, 51-60=5, 61-70=6, 71-120=7, default=3
14	hash_id_dpt_1	0	id_col	hash	1	8	
15	mhash_varchar_dpt_1	0	varchar_col	mod_hash	1	8	

The following is an example output of **show rule from *table_name***.

```
mysql> show rule from range_id_yd_datetime_3_tpt_1 ;
```

ID	TABLE_NAME	BROADCAST	DB_PARTITION_KEY	DB_PARTITION_POLICY	DB_PARTITION_COUNT	DB_PARTITION_OFFSET	PARTITION_RANGE
			TB_PARTITION_KEY	TB_PARTITION_POLICY	TB_PARTITION_COUNT	TB_PARTITION_OFFSET	
1	range_id_yd_datetime_3_tpt_1	0	id_col	range	3	8	0-10=0, 11-20=1, 21-30=2, 31-40=3, 41-
			datetime_col	yyyydd			50=4, 51-60=5, 61-70=6, 71-120=7, default=3

1 row in set (0.00 sec)

Output Details:

TABLE_NAME: indicates the name of the logical table.

BROADCAST: specifies whether the table is a broadcast table. **0** indicates that the table is not a broadcast table. **1** indicates the table is a broadcast table.

DB_PARTITION_KEY: indicates the database sharding key. Leave this field blank if database sharding is not required.

DB_PARTITION_POLICY: indicates the database sharding algorithm. The value can be **HASH**, **YYYYMM**, **YYYYDD**, and **YYYYWEEK**.

DB_PARTITION_COUNT: indicates the number of database shards.

DB_PARTITION_OFFSET: indicates where a new database partition starts from.

PARTITION_RANGE: indicates the sharding range when the database sharding algorithm is range.

TB_PARTITION_KEY: indicates the table sharding key. Leave this field blank if table sharding is not required.

TB_PARTITION_POLICY: indicates the table sharding algorithm. The value can be **HASH**, **MM**, **DD**, **MMDD**, or **WEEK**.

TB_PARTITION_COUNT: indicates the number of physical tables on each database shard.

TB_PARTITION_OFFSET: indicates where a new table partition starts from.

12.6.3 SHOW TOPOLOGY

Command Format:

show topology from xxxtable_name: used to view physical tables corresponding to a specified logical table.

Output Details:

Rds_instance_id: indicates the ID of the RDS DB instance.

HOST: indicates the IP address of the RDS DB instance.

PORT: indicates the port number of the RDS DB instance.

DATABASE: indicates the physical database in the RDS DB instance.

TABLE: indicates the physical table.

ROW_COUNT: indicates the estimated number of data records in each physical table. The value is obtained from information_schema.TABLES.

12.6.4 SHOW DATA NODE

Command Format:

show data node: used to view data about database shards in the RDS DB instance.

Output Details:

Rds_instance_id: indicates the ID of the RDS DB instance.

PHYSICAL_NODE: used to view physical databases in the RDS DB instance.

HOST: indicates the IP address of the RDS DB instance.

PORT: indicates the port number of the RDS DB instance.

12.6.5 TRUNCATE TABLE

Hints are valid only for sharded tables.

12.6.5.1 HINT-DB

Command Format:

```
/*+db=<physical_db_name>*/ TRUNCATE TABLE <table_name>
```

Description:

Deleting data records in physical tables corresponding to *<table_name>* in *<physical_db_name>* does not affect physical tables in other database shards.

12.6.5.2 HINT-TABLE

Command Format:

```
/*+table=<physical_table_name>*/ TRUNCATE TABLE <table_name>
```

Description:

Deleting data records in physical tables named *<physical_table_name>* in the current database shard does not affect other physical tables.

12.6.5.3 HINT-DB/TABLE

Command Format:

```
/*+db=<physical_db_name>,table=<physical_table_name>*/ TRUNCATE TABLE <table_name>
```

Description:

Deleting data records in physical table *<physical_table_name>* in database shard *<physical_db_name>* does not affect other physical tables.

12.6.6 HINT- ALLOW_ALTER_RERUN

Command Format:

```
/*+ allow_alter_rerun=true*/ALTER TABLE aaa_tb ADD schoolroll varchar(128) not null  
comment 'Enrollment data'
```

Description:

Using this hint ensures that commands can be repeatedly executed, and no error is reported. This hint supports the following ALTER TABLE statements: ADD COLUMN, MODIFY COLUMN, DROP COLUMN, ADD INDEX, DROP INDEX, CHANGE COLUMN, ADD PARTITION, and DROP PARTITION.

12.6.7 LOAD DATA

Standard Example

```
LOAD DATA LOCAL INFILE '/data/qq.txt' IGNORE INTO TABLE test CHARACTER  
SET 'gbk' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES  
TERMINATED BY '\n'(id,sid,asf)
```

 **NOTE**

If the data contains special characters, such as separators or escape characters, enclose the characters with quotation marks (") and specify them using OPTIONALLY ENCLOSED BY "".

If the preceding method does not work, replace quotation marks (") with special characters (\) and marks (").

- If keyword **LOCAL** is specified, the file is read from the client host. If keyword **LOCAL** is not specified, this function is not supported for security purposes.
- You can use **FIELDS TERMINATED BY** to specify a separator between characters. The default value is `\t`.
- You can use **OPTIONALLY ENCLOSED BY** to ignore symbols in the data source fields.
- You can use **LINES TERMINATED BY** to specify a newline character between lines. The default value is `\n`.

 **NOTE**

On some hosts running the Windows OS, the newline character of text files may be `\r\n`. The newline character is invisible, so you may need to check whether it is there.

- You can use **CHARACTER SET** to specify a file code which should be the same as the code used by physical databases in the target RDS MySQL DB instance, to avoid garbled characters. The character set code shall be enclosed in quotation marks to avoid parsing errors.
- You can use **IGNORE** or **REPLACE** to specify whether repeated records are replaced or ignored.
- Currently, the column name must be specified, and the sharding field must be included. Otherwise, the route cannot be determined.
- For other parameters, see the [LOAD DATA INFILE Syntax](#) on the MySQL official website. The sequence of other parameters must be correct. For more information, visit [the MySQL official website](#).

NOTICE

1. Importing data affects performance of DDM instances and RDS MySQL DB instances. Import data during off-peak hours.
2. Do not to send multiple LOAD DATA requests at the same time. If you do so, SQL transactions may time out due to highly concurrent data write operations, table locking, and system I/O occupation, resulting in failure of all LOAD DATA requests.
3. Manually submit transactions when using LOAD DATA to import data so that data records are modified correctly.

For example, configure your client as follows:

```
mysql> set autocommit=0;
```

```
mysql> LOAD DATA LOCAL INFILE '/data/qq.txt' IGNORE INTO TABLE test  
CHARACTER SET 'gbk' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED  
BY '"' LINES TERMINATED BY '\n'(id,sid,asf);
```

```
mysql> commit;
```

Restrictions

The following are not supported:

- [IGNORE number {LINES | ROWS}] clauses
- SET clauses

12.6.8 SHOW PHYSICAL PROCESSLIST

Command Format 1:

show physical processlist: Displays the processes that run on the associated RDS DB instance.

Command Format 2:

show physical processlist with info: Filters out the data whose **info** is empty from the result set of command 1 and displays only the data whose **info** is not empty.

The command output is as follows:

Figure 12-1 Command output

ip	Port	Instance_id	Type	Physical_thread_id	User	Host	Db	Command	Time	State	Info
localhost	33062	shard2	master	4	DDMRV	localhost:1world_0007		Sleep	24373		
localhost	33062	shard2	master	5	DDMRV	localhost:1world_0007		Sleep	24373		
localhost	33062	shard2	master	6	DDMRV	localhost:1world_0004		Sleep	769		
localhost	33062	shard2	master	7	DDMRV	localhost:1world_0006		Sleep	769		
localhost	33062	shard2	master	8	DDMRV	localhost:1world_0007		Sleep	24373		
localhost	33062	shard2	master	9	DDMRV	localhost:1world_0007		Sleep	24373		
localhost	33062	shard2	master	10	DDMRV	localhost:1world_0004		Sleep	24373		
localhost	33062	shard2	master	11	DDMRV	localhost:1world_0005		Sleep	769		
localhost	33062	shard2	master	12	DDMRV	localhost:1world_0007		Query	0	starting	SHOW FULL
localhost	33062	shard2	master	13	DDMRV	localhost:1world_0004		Sleep	24373		
localhost	33061	shard1	master	7	DDMRV	localhost:1world_0000		Sleep	24372		
localhost	33061	shard1	master	8	DDMRV	localhost:1world_0000		Sleep	19576		

Output details:

ip: indicates the IP address of the associated RDS DB instance.

port: indicates the port number of the associated RDS DB instance.

instance id: indicates the ID of the associated RDS DB instance.

type:master: indicates that the associated DB instance is the primary type, and **readreplica** indicates that the associated DB instance is the read-only type.

Columns after column **type** indicate the information about processes running on the associated RDS DB instance, and such information is the same as the output of command **show processlist** executed on the associated RDS DB instance.

Command Format 3:

kill physical physical_thread_id@rds_ip:rds_port: kills the execution thread on the associated RDS DB instance.

12.6.9 Customized Hints for Read/Write Isolation

DDM allows you to customize a hint to specify whether SQL statements are executed on the primary instance or its read replicas.

The hint is in the format of **/*mycat:db_type=host*/**.

In the format, **host** can be "master" or "slave". "master" refers to the primary DB instance, and "slave" refers to read replicas.

Currently, this function only applies to SELECT statements.

 **NOTE**

Generally, after read/write isolation, write operations are performed only on the primary DB instance, and read operations are performed only on its read replicas. In some special scenarios where data needs to be read from the primary DB instance, you can customize a hint to forcibly perform read operations on the primary DB instance.

12.6.10 Setting a Hint to Skip the Cached Execution Plan

DDM allows you to configure a hint to control whether the SELECT statement skips the cached execution plan.

The hint is in the format of `/*!GAUSS:skip_plancache=flag*/`.

flag can be set to **true** or **false**. **true** indicates that the statement skips the cached execution plan. **false** indicates that the statement does not skip the cached execution plan.

Currently, this function only applies to SELECT statements.

12.7 Global Sequence

Global sequences are mainly database-based global sequences.

 **NOTE**

- The start auto-increment SN can be modified.
- Global sequence provides sequence numbers that are globally unique but may not increase continuously.

Table 12-30 Tables supported by global sequence

Table Type	Sharded Table	Broadcast Table	Unsharded Table
DB-based	Supported	Supported	Not supported

Creating an Auto-Increment Sequence

Step 1 Connect to the target DDM instance using a client.

Step 2 Open the target schema.

Step 3 Run the following command to create an auto-increment sequence:

```
create sequence xxxxx ;
```

 **NOTE**

- *xxxxx* indicates the sequence name.

----**End**

Deleting an Auto-Increment Sequence

Step 1 Connect to the target DDM instance using a client.

Step 2 Open the target schema.

Step 3 Run **show sequences** to view all global sequences.

Step 4 Run the following command to delete an auto-increment sequence:

```
drop sequence xxxxx;
```

```
drop sequence DB.xxx;
```

NOTE

- The sequence name is case-insensitive.
- If an auto-increment sequence is inherent to a table, the sequence cannot be deleted.

----End

Modifying the Start Value of an Auto-Increment Global Sequence

Step 1 Connect to the target DDM instance using a client.

Step 2 Open the target schema.

Step 3 Run **show sequences** to view all global sequences.

Step 4 Run the command to change the start value:

```
alter sequence xxxxx START WITH yyyyy;
```

NOTE

- *xxxxx* indicates the sequence name.
- *yyyyy* indicates the start value of the target sequence.

----End

Querying an Auto-Increment Sequence

Step 1 Connect to the target DDM instance using a client.

Step 2 Open the target schema.

Step 3 Run the following command to view all global sequences:

```
show sequences;
```

```
mysql> show sequences;
```

NAME	START_WITH	INCREMENT
WORLD.WORLDDYML	1	1000
WORLD.YML	1	1000

----End

Modifying the Auto-Increment Cache Value

NOTICE

Only kernel 3.0.3 and later versions are supported.

- Step 1** Connect to the target DDM instance using a client.
- Step 2** Open the target schema.
- Step 3** Run command **alter sequence test cache 5000** to modify the global sequence cache value of table **test**.
- Step 4** Run command **show sequences** to view the cache value (**INCREMENT** value) of table **test**.
----End

12.7.1 Using NEXTVAL and CURRVAL to Query Global Sequence Numbers

- NEXTVAL returns the next sequence number, and CURRVAL returns the current sequence number. NEXTVAL(N) returns *n* unique sequence numbers.
- NEXTVAL(N) can be used only in **select sequence.nextval(n)** and does not support cross-schema operations.
- CURRVAL(N) is not supported.

Procedure

- Step 1** Connect to the required DDM instance using a client.
- Step 2** Open the required schema.
- Step 3** Run the following command to create a global sequence:

```
create sequence seq_test;
```

```
mysql> create sequence seq_test;  
Query OK, 0 rows affected (0.28 sec)
```

- Step 4** Run the following command to obtain the next sequence number:
select seq_test.nextval;


```
mysql> select seq_test.nextval;
+-----+
| seq_test.NEXTVAL |
+-----+
|          1 |
+-----+
1 row in set (0.05 sec)

mysql> select seq_test.nextval;
+-----+
| seq_test.NEXTVAL |
+-----+
|          2 |
+-----+
1 row in set (0.04 sec)

mysql> select seq_test.nextval;
+-----+
| seq_test.NEXTVAL |
+-----+
|          3 |
+-----+
1 row in set (0.03 sec)
```

Step 5 Run the following command to obtain the current sequence number:

```
select seq_test.currval;
```

```
mysql> select seq_test.currval;
+-----+
| seq_test.CURRVAL |
+-----+
|          3 |
+-----+
1 row in set (0.31 sec)

mysql> select seq_test.currval;
+-----+
| seq_test.CURRVAL |
+-----+
|          3 |
+-----+
1 row in set (0.03 sec)
```

Step 6 Run the following command to obtain sequence numbers in batches:

```
select seq_test.nextval(n);
```

```
mysql> select seq_test.nextval(5);
+-----+
| seq_test.NEXTVAL |
+-----+
|                4 |
|                5 |
|                6 |
|                7 |
|                8 |
+-----+
5 rows in set (0.04 sec)
```

NOTE

- Cross-schema operations are not supported when sequence numbers are obtained in batches.
- If no global sequence is used, CURRVAL returns 0.

----End

12.7.2 Using Global Sequences in INSERT or REPLACE Statements

You can use global sequences in INSERT or REPLACE statements to provide unique global sequence across schemas in a DDM instance. Generating sequence numbers with NEXTVAL and CURRVAL is supported in INSERT or REPLACE statements. NEXTVAL returns the next sequence number, and CURRVAL returns the current sequence number, for example, schema.seq.nextval and schema.seq.currval. If no schema is specified, use the global sequence of the currently connected schema.

Concurrently executing schema.seq.nextval in multiple sessions is supported to obtain unique global sequence numbers.

Example

There are two schemas dml_test_1 and dml_test_2, and both of them have table test_seq.

Table Definition

```
CREATE TABLE test_seq(col1 BIGINT,col2 BIGINT) DBPARTITION BY HASH(col1)
```

Procedure

- Step 1** Connect to the required DDM instance using a client.
- Step 2** Open the required schema.
- Step 3** Run the following command to create a global sequence for a schema:

```
use dml_test_1;
create sequence seq_test;
```

```
mysql> use dml_test_1;
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> create table test_seq(coll1 bigint,col2 bigint) dbpartition by hash(coll1);
Query OK, 0 rows affected (0.58 sec)

mysql> create sequence seq_test;
Query OK, 0 rows affected (0.73 sec)
```

Step 4 Run the following statement to use the global sequence in an INSERT or REPLACE statement:

- **use dml_test_1;**

insert into test_seq(col1,col2)values(seq_test.nextval,seq_test.currval);

```
mysql> insert into test_seq(coll1,col2)values(seq_test.nextval,seq_test.currval);
Query OK, 1 row affected (0.31 sec)

mysql> select * from test_seq;
+-----+-----+
| coll1 | col2 |
+-----+-----+
| 1 | 1 |
+-----+-----+
1 row in set (0.05 sec)
```

- **use dml_test_2;**

insert into test_seq(col1,col2)values(dml_test_1.seq_test.nextval,dml_test_1.seq_test.currval);

```
mysql> use dml_test_2;
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> create table test_seq(coll1 bigint,col2 bigint) dbpartition by hash(coll1);
Query OK, 0 rows affected (0.52 sec)

mysql> insert into test_seq(coll1,col2)values(dml_test_1.seq_test.nextval,dml_test_1.seq_test.currval);
Query OK, 1 row affected (0.04 sec)

mysql> select * from test_seq;
+-----+-----+
| coll1 | col2 |
+-----+-----+
| 2 | 2 |
+-----+-----+
1 row in set (0.05 sec)
```

The global sequence is created in schema `dml_test_1`. To use the global sequence in schema `dml_test_2`, you need to specify the schema name, for example, `dml_test_1.seq_test.nextval` and `dml_test_1.seq_test.currval`.

NOTE

- Using global sequences in INSERT and REPLACE statements is supported only in sharded tables, but not broadcast or unsharded tables.
- NEXTVAL and CURRVAL are executed from left to right in INSERT and REPLACE statements. If NEXTVAL is referenced more than once in a single statement, the sequence number is incremented for each reference.
- Each global sequence belongs to a schema. When you delete a schema, the global sequence of the schema is also deleted.

----End

12.8 Database Management Syntax

Supported Database Management Syntax

- SHOW Syntax
- SHOW COLUMNS Syntax
- SHOW CREATE TABLE Syntax
- SHOW TABLE STATUS Syntax
- SHOW TABLES Syntax
- SHOW DATABASES
- SHOW INDEX FROM
- SHOW VARIABLES Syntax

Supported Database Tool Commands

- DESC Syntax
- USE Syntax
- EXPLAIN Syntax

Unlike EXPLAIN in MySQL, the output of DDM EXPLAIN describes the nodes that the current statement is routed to.

Unsupported Database Management Syntax

Table 12-31 Restrictions on database administration statements

Item	Restriction
Database administration statements	<ul style="list-style-type: none"> • Executing SET Syntax to modify global variables is not supported. • SHOW TRIGGERS Syntax is not supported. <p>The following SHOW statements are randomly sent to a physical shard. If physical shards are on different RDS MySQL DB instances, the returned variables or table information may be different.</p> <ul style="list-style-type: none"> • SHOW TABLE STATUS • SHOW VARIABLES Syntax • CHECK TABLE does not support sharding tables by hash or sharding key. • SHOW WARNINGS Syntax does not support the combination of LIMIT and COUNT. • SHOW ERRORS Syntax does not support the combination of LIMIT and COUNT.

12.9 Advanced SQL Functions

Table 12-32 Restrictions on advanced SQL functions

Item	Restriction
SQL functions	<ul style="list-style-type: none"> ● PREPARE and EXECUTE syntax is not supported. ● Customized data types and functions are not supported. ● Views, stored procedures, triggers, and cursors are not supported. ● Compound statements such as BEGIN...END, LOOP...END LOOP, REPEAT...UNTIL...END REPEAT, and WHILE...DO...END WHILE are not supported. ● Process control statements such as IF and WHILE are not supported. ● The following prepared statements are not supported: PREPARE Syntax EXECUTE Syntax ● Comments for indexes are not supported in table creation statements. ● Configuring user permissions is not supported. For example, user privilege settings such as grant all on *.* to 'test'@'%' and identified by 'test123' are not supported. User privileges can be set on the console.

13 FAQs

13.1 General Questions

13.1.1 What High-Reliability Mechanisms Does DDM Provide?

Protection of Data Integrity

DDM instance faults do not affect data integrity.

- Service data is stored in shards of RDS MySQL DB instances, but not on DDM.
- Configuration information of schemas and logical tables is stored in DDM databases. Active and standby DDM databases are highly available.

Fault Tolerance

DDM uses the RDS MySQL fault tolerance mechanism. RDS for MySQL ensures that data is written to a database. DDM considers that data is written to the database if the RDS MySQL DB instance returns a message indicating that the SQL statements is executed successfully.

13.1.2 Does DDM Store Service-Related Data?

DDM instances do not store service-related data, which is stored in shards of the RDS MySQL DB instances.

The hard disks on DDM nodes are used to store log files and temporary files, which are periodically cleared to ensure sufficient space.

13.1.3 How Do I Select and Configure a Security Group?

DDM uses VPCs and security groups to ensure security of your instances. The following provides guidance for you on how to correctly configure a security group.

Intra-VPC Access to DDM Instances

Access to a DDM instance includes access to the DDM instance from the ECS where a client is located and access to its associated RDS MySQL DB instance.

The ECS, DDM instance, and RDS MySQL DB instance must be in the same VPC. In addition, correct rules should be configured for their security groups to allow network access.

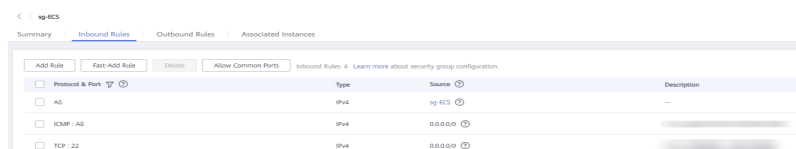
1. You are advised to configure the same security group for the ECS, DDM instance, and RDS MySQL DB instance. After a security group is created, network access in the group is not restricted by default.
2. If different security groups are configured, you may need to refer to the following configurations:

NOTE

- Assume that the ECS, DDM instance, and RDS MySQL DB instance are configured with security groups **sg-ECS**, **sg-DDM**, and **sg-RDS**, respectively.
- Assume that the service port of the DDM instance is **5066** and that of the RDS MySQL DB instance is **3306**.
- The remote end should be a security group or an IP address.

Add the rules in [Figure 13-1](#) to the security group of the ECS to ensure that your client can access the DDM instance.

Figure 13-1 ECS security group rules

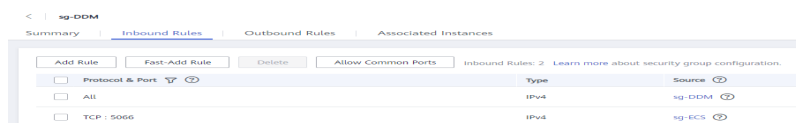


The screenshot shows the 'Inbound Rules' configuration for security group 'sg-ECS'. It displays four rules:

Protocol & Port	Type	Source	Description
All	IPv4	sg-ECS	--
ICMP - All	IPv4	0.0.0.0	
TCP : 22	IPv4	0.0.0.0	

Add the rules in [Figure 13-2](#) and [Figure 13-3](#) to the security group of the ECS where your DDM instance is located so that your DDM instance can access the associated RDS MySQL DB instance and can be accessed by your client.

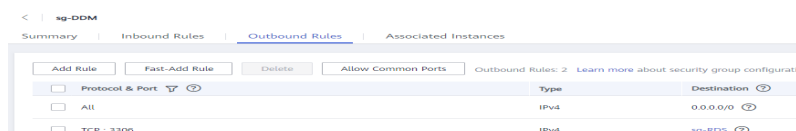
Figure 13-2 Configuring security group inbound rules for your DDM instance



The screenshot shows the 'Inbound Rules' configuration for security group 'sg-DDM'. It displays two rules:

Protocol & Port	Type	Source	Description
All	IPv4	sg-DDM	
TCP : 5066	IPv4	sg-ECS	

Figure 13-3 Configuring security group outbound rules for your DDM instance

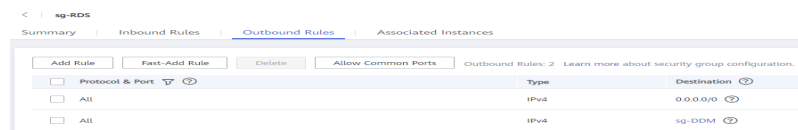


The screenshot shows the 'Outbound Rules' configuration for security group 'sg-DDM'. It displays two rules:

Protocol & Port	Type	Destination	Description
All	IPv4	0.0.0.0/0	
TCP : 3306	IPv4	sg-RDS	

Add the rules in [Figure 13-4](#) to the security group of the ECS where the RDS MySQL DB instance is located so that your DDM instance can access the RDS DB instance.

Figure 13-4 Configuring security group rules of the RDS MySQL DB instance



13.2 DDM Usage

13.2.1 How Can I Rectify a Fault in Accessing DDM by Using the JDBC Driver?

When you access DDM using the MySQL driver (JDBC) in load balancing mode, an infinite loop may occur during connection switchover, resulting in stack overflow.

Fault Locating

1. Query the application logs and locate the fault cause.

For example, the following logs show that the fault is caused by stack overflow.

```
Caused by: java.lang.StackOverflowError
  at java.nio.HeapByteBuffer.<init>(HeapByteBuffer.java:57)
  at java.nio.ByteBuffer.allocate(ByteBuffer.java:335)
  at java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:795)
  at java.nio.charset.Charset.encode(Charset.java:843)
  at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:2362)
  at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:2344)
  at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:568)
  at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:626)
  at com.mysql.jdbc.Buffer.writeStringNotNull(Buffer.java:670)
  at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2636)
```

2. Analyze the overflow source.

For example, the following logs show that the overflow results from an infinite loop inside the driver.

```
at
com.mysql.jdbc.LoadBalancedConnectionProxy.pickNewConnection(LoadBalancedConnectionProxy.java:
344)
at
com.mysql.jdbc.LoadBalancedAutoCommitInterceptor.postProcess(LoadBalancedAutoCommitIntercepto
r.java:104)
at com.mysql.jdbc.MysqlIO.invokeStatementInterceptorsPost(MysqlIO.java:2885)
at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2808)
at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2483)
at com.mysql.jdbc.ConnectionImpl.setReadOnlyInternal(ConnectionImpl.java:4961)
at com.mysql.jdbc.ConnectionImpl.setReadOnly(ConnectionImpl.java:4954)
at com.mysql.jdbc.MultiHostConnectionProxy.syncSessionState(MultiHostConnectionProxy.java:381)
at com.mysql.jdbc.MultiHostConnectionProxy.syncSessionState(MultiHostConnectionProxy.java:366)
at
com.mysql.jdbc.LoadBalancedConnectionProxy.pickNewConnection(LoadBalancedConnectionProxy.java:
344)
```

3. Query the MySQL version, which is 5.1.44.

According to the source code of the version, when a connection is obtained, **LoadBalance** updates the connection based on the load balancing policy and copies the configurations of the old connection to the new connection. If **AutoCommit** is **true** for the new connection, parameters of the new connection are inconsistent with those of the old connection, and **loadBalanceAutoCommitStatementThreshold** is not configured, an infinite

loop occurs. The connection update function calls the parameter synchronization function, and the parameter synchronization function calls the connection update function at the same time, resulting in stack overflow.

Solution

Add the **loadBalanceAutoCommitStatementThreshold=5&retriesAllDown=10** parameter to the URL for connecting to DDM.

```
//Connection example when load balancing is used
//jdbc:mysql:loadbalance://ip1:port1,ip2:port2..ipN:portN/{db_name}
String url = "jdbc:mysql:loadbalance://192.168.0.200:5066,192.168.0.201:5066/db_5133?loadBalanceAutoCommitStatementThreshold=5&retriesAllDown=10";
```

- **loadBalanceAutoCommitStatementThreshold** indicates the number of statements executed before a reconnection.

If **loadBalanceAutoCommitStatementThreshold** is set to **5**, a reconnection is initiated after five SQL statements (queries or updates) are executed. A value of **0** indicates a sticky connection, and no reconnection is required. When automatic submission is disabled (**autocommit** is set to **false**), the system waits for the transaction to complete and then determines whether to initiate a reconnection.

13.2.2 What Version and Parameters Should I Select?

Currently, you cannot connect to DDM using JDBC driver 5.1.46. Versions 5.1.35 to 5.1.45 are recommended.

JDBC driver download address: <https://dev.mysql.com/doc/index-connectors.html>

Table 13-1 describes the recommended parameters for the JDBC URL.

Table 13-1 Parameters

Parameter	Description	Recommended Value
ip:port	Indicates the connection address and port number for connecting to DDM.	Query the connection address from the DDM instance details page.
db_name	Indicates the name of a schema.	Query the schema name on the Schemas page of the DDM instance details page.

Parameter	Description	Recommended Value
loadBalanceAutoCommitStatementThreshold	<p>Indicates the number of statements executed before a reconnection.</p> <ul style="list-style-type: none"> If the parameter value is set to 5, after five SQL statements (queries or updates) are executed, a reconnection is initiated. A value of 0 indicates a sticky connection, and no reconnection is required. <p>When automatic submission is disabled (autocommit is set to false), the system waits for the transaction to complete and then determines whether to initiate a reconnection.</p>	5
loadBalanceHostRemovalGracePeriod	Sets the grace period for removing a host from the load balancing connection.	15000
loadBalanceBlacklistTimeout	Sets the time for retaining a service in the global blacklist.	60000
loadBalancePingTimeout	Indicates the time (unit: ms) for waiting for the ping response of each load balancing connection.	5000
retriesAllDown	<p>Indicates the maximum number of polling retries when all connection addresses fail.</p> <p>If the threshold for retries has been reached but no valid address can be obtained, "SQLException" will be displayed.</p>	10
connectTimeout	<p>Specifies the timeout interval for establishing a socket connection with a database server.</p> <p>Unit: ms. A value of 0 indicates that connection establishment never times out. This parameter setting is used for JDK 1.4 or later versions.</p>	10000
socketTimeout	<p>Specifies the timeout interval for a socket operation (read and write).</p> <p>Unit: ms. A value of 0 indicates that a socket operation never times out.</p>	Set this parameter based on your service requirements.

13.2.3 Why It Takes So Long Time to Export Data from MySQL Using mysqldump?

The version of the mysqldump client may be inconsistent with that of the supported MySQL server.

Using the same version of the mysqldump client and MySQL server is recommended.

13.2.4 How Should I Handle the Duplicate Primary Key Error Occurring After Data Is Imported into DDM?

When creating a table, set the start value for automatic increment and ensure that the start value is greater than the maximum auto-increment value of imported data.

13.2.5 What Should I Do If an Error Message Is Returned After I Specify an Auto-Increment Primary Key?

Execute the following SQL statement to modify the start value of the auto-increment primary key so that the value is greater than the maximum value of primary keys in existing tables:

```
ALTER SEQUENCE Schema name.SEQ name START WITH New start value
```

13.2.6 How Do I Handle the Error Reported When Parameter Configuration Does Not Time Out?

Adjust the **SocketTimeOut** value or leave this parameter blank. The default value is 0, indicating that the client is not disconnected.

13.2.7 Which Should I Delete First, Schema or Associated RDS DB Instances?

You should delete a schema first and then its associated RDS DB instances. If you delete the associated RDS DB instances before deleting the schema, the schema cannot be deleted. You need to delete it by deleting the associated DDM instance.

13.2.8 Should I Manually Delete Databases and Accounts Remained in the Associated RDS DB Instances After a Schema Is Deleted?

If you do not need to delete the databases or accounts, you can manually delete them to free up space.

13.3 SQL Syntax

13.3.1 Does DDM Support Cross-Database Access of SQL?

DDM does not support cross-schema access of SQL statements which contain database names.

DDM automatically deletes the database names in SQL statements, for example, the statement **select * from dn1.item** will be automatically changed to **select * from item**.

13.3.2 Does DDM Support Distributed JOIN?

DDM supports distributed JOIN.

- Redundant fields are added during table design.
- Cross-shard JOIN is implemented by using broadcast tables, ER shards, and ShareJoin.
- Currently, DDM does not allow cross-database update or deletion of multiple tables.

13.3.3 How Do I Optimize SQL Statements?

- You are advised to use INNER instead of LEFT JOIN or RIGHT JOIN.
- When LEFT JOIN or RIGHT JOIN is used, ON is preferentially executed, and WHERE is executed at the end. Therefore, when using LEFT JOIN or RIGHT JOIN, ensure that the conditions are judged in the ON statement to reduce the execution of WHERE.
- When possible, use JOIN instead of subqueries to avoid full scanning of large tables.

13.3.4 Does DDM Support Forced Data Type Conversion?

Data type conversion is an advanced function. DDM will be gradually improved to be compatible with more SQL syntax. If necessary, submit a service ticket for processing.

13.3.5 What Should I Do If an Error Is Reported When Multiple Data Records Are Inserted into Batches Using the INSERT Statement?

Solution

Split the INSERT statement into multiple small statements, or configure parameters on the console, modify the value of **max_allowed_packet**, and restart the instance to have the modification take effect.

13.4 RDS-related Questions

13.4.1 Is the Name of a Database Table Case-Sensitive?

By default, DDM and RDS MySQL are case-insensitive to data table names and sequence names.

13.4.2 What Risky Operations on RDS for MySQL Will Affect DDM?

[Table 13-2](#) lists risky operations on RDS for MySQL.

Table 13-2 Risky operations on RDS for MySQL

Operation Type	Operation	Impact of the Operation
Operations on the RDS console	Deleting an RDS MySQL DB instance	After an RDS MySQL DB instance is deleted, all schemas and logical tables of the DDM instance associated with the DB instance become unavailable.
	Performing the primary/standby switchover of an RDS MySQL DB instance	RDS may be intermittently interrupted during the primary/standby switchover. In addition, a small amount of data may be lost in case of long delay in primary/standby synchronization. <ul style="list-style-type: none"> • Creating schemas or logical tables or scaling out schemas is not allowed on DDM during the primary/standby switchover of the RDS MySQL DB instance. • After a primary/standby switchover of an RDS MySQL instance, the RDS MySQL instance ID in DDM remains unchanged.
	Restarting an RDS MySQL DB instance	An RDS MySQL DB instance being restarted is unavailable, and the DDM service will be affected.
	Resetting a password	After the RDS DB instance password is reset, enter the new password on DDM when creating a schema.
	Modifying a parameter template	The following parameters are set to fixed values. If their values are changed, DDM will not function properly. <ul style="list-style-type: none"> • lower_case_table_names: Set this parameter to 1, indicating that data table names and sequence names are case-insensitive. • local_infile: Set this parameter to ON in scale-out scenarios.
	Modifying a security group	The DDM instance cannot connect to the RDS MySQL DB instance.

Operation Type	Operation	Impact of the Operation
	Modifying a VPC	The DDM instance and RDS MySQL DB instance cannot communicate with each other if they are in different VPCs.
	Restoring data	Restoring data may damage data integrity.
Operations through an RDS MySQL client	Deleting a physical database created on DDM	After a physical database is deleted, the original data will be lost and new data cannot be written into the database.
	Deleting a physical account created on DDM	After a physical account is deleted, logical tables cannot be created on DDM.
	Deleting a physical table created on DDM	After a logical table is deleted, data will be lost on DDM, and the table cannot be used on DDM.
	Modifying the name of a logical table created on DDM	After the name of a logical table is modified, DDM cannot obtain the table data, and the table cannot be used on DDM.
	Changing a record	Changing a record in a broadcast table will affect the data consistency of shards.
	Modifying a whitelist	The DDM instance cannot access the RDS MySQL DB instance if the DDM instance is not in the DB instance whitelist.

13.4.3 How Do I Handle Data with Duplicate Primary Keys in a Table?

Scenario 1

The logical table of a DDM instance contains a data record whose type value of the primary key is a boundary value. When you enter a data record with the value exceeding the required range, data with duplicate primary keys is displayed.

Procedure

- Step 1** Log in to the RDS console.
- Step 2** On the **Instance Management** page, locate the target RDS MySQL DB instance and click its name.
- Step 3** On the **Basic Information** page, choose **Parameters** in the left pane.
- Step 4** Click the **Parameters** tab and enter **sql_mode** in the text box. Then click the expanding button in the **Value** column, select **STRICT_ALL_TABLES** or **STRICT_TRANS_TABLES**, and click **Save**.

NOTE

STRICT_ALL_TABLES and **STRICT_TRANS_TABLES** are both strict modes. The strict mode controls how MySQL handles invalid or missing values.

- An invalid value might have the wrong data type for the column, or might be out of range.
- A value is missing when a new row to be inserted does not contain a value for a non-NULL column that has no explicit DEFAULT clause in its definition.
- If the DDM instance version is earlier than 2.4.1.3, do not set **sql_mode** to **ANSI_QUOTES**. If you set it to **ANSI_QUOTES**, double quotation marks used for each string will be translated into an identifier during SQL execution, making the string invalid.

For example, **logic** in **select * from test where tb = "logic"** cannot be parsed correctly.

For more information about SQL modes, see [Server SQL Modes](#).

- Step 5** On the **Instances** page, restart the DDM instance.

----End

Scenario 2

For sharded tables (**hash\range\mod**) with composite primary keys, the primary key is duplicated when you insert **0** and **1** to data with the sharding key.

Procedure

- Step 1** Log in to the RDS console.
- Step 2** On the **Instance Management** page, locate the target RDS MySQL DB instance and click its name.
- Step 3** On the **Basic Information** page, choose **Parameters** in the left pane.
- Step 4** Click the **Parameters** tab and enter **sql_mode** in the text box. Then click the expanding button in the **Value** column, select **NO_AUTO_VALUE_ON_ZERO** in the drop-down list, and click **Save**.

- Step 5** On the **Instances** page, restart the DDM instance.

----End

13.4.4 How Can I Query RDS Information by Running Command `show full innodb status`?

After connecting to a DDM instance through the MySQL client, you can run command `show full innodb status` to query information about the associated RDS MySQL DB instances. The following information can be queried:

- Current time and duration since the last output.
- Status of the master thread.
- SEMAPHORES including event counts and available waiting threads when there is high-concurrency workload. You can use the information to locate performance bottlenecks if any.

13.5 Connection Management

13.5.1 Can I Connect to DDM Instances from My Local Environment?

Yes. To access a DDM instance from your local environment, you first need to bind an EIP to the DDM instance and then access it using the EIP on a connection tool, for example, Navicat.

Using Navicat to Log In to a DDM Instance

Step 1 Log in to the DDM console, locate the required DDM instance, and click its name.

Step 2 In the **Instance Information** area, click **Bind** and select the EIP that you have applied for

Step 3 On the DDM console, click the VPC icon on the left. Choose **Access Control > Security Groups**.

Step 4 On the **Security Groups** page, locate the target security group and click **Manage Rule** in the **Operation** column. On the displayed page, click **Add Rule**. Configure the security group rule as needed and click **OK**.

NOTE

After binding an EIP to your DDM instance, set strict inbound and outbound rules for the security group to enhance database security.

Step 5 Open Navicat and click **Connection**. In the displayed dialog box, enter the host IP address (EIP), username, and password (DDM account).

Step 6 Click **Test Connection**. If a message is returned indicating that the connection is successful, click **OK**. The connection will be succeeded 1 to 2 minutes later. If the connection fails, the failure cause is displayed. Modify the required information and try again.

----End

 NOTE

Using Navicat to access a DDM instance is similar to using other visualized MySQL tools such as MySQL Workbench. Therefore, the procedure of using other visualized MySQL tools to connect to a DDM instance has been omitted.

13.5.2 What Should I Do If Garbled Characters Are Displayed When I Connect to DDM Using MySQL?

If the MySQL connection code is inconsistent with the actual one, garbled characters may be displayed during parsing on DDM.

In this case, configure **default-character-set=utf8** to specify the encoding system.

Example:

```
mysql -h127.0.0.1 -P5066 -Dbase --default-character-set=utf8 -uddmuser -p
```