Distributed Database Middleware

User Guide

Issue 01

Date 2026-01-23

HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property 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 Cloud 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.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Service Overview

1.1 What Is DDM?

Definition

Distributed Database Middleware (DDM) is a MySQL-compatible, distributed middleware service designed for relational databases. It can resolve distributed scaling issues to break through capacity and performance bottlenecks of databases, helping handle highly concurrent access to massive volumes of data.

DDM is reliable, stable, scalable, and maintainable. It uses an architecture with decoupled storage and compute and can provide functions like database and table sharding, read/write splitting, and elastic scaling. Management of instance nodes has no impacts on your workloads. You can perform O&M on your databases and read and write data from and to them on the DDM console, just like as operating a single-node database.

Advantages

- Automatic database and table sharding

  MySQL databases are usually deployed on single nodes. Once a fault occurs, all data may be lost, and your workloads are 100% affected.

  DDM supports automatic database and table sharding to distribute data across multiple data nodes, so impacts on your services are greatly reduced once a fault occurs. It also supports explosive growth of services.

- Read/Write splitting

  DDM can leverage data nodes. If there is still great query pressure after horizontal sharding, you can enable read/write splitting to speed up database processing and access, without the need to reconstruct your service system.

- Elastic scaling

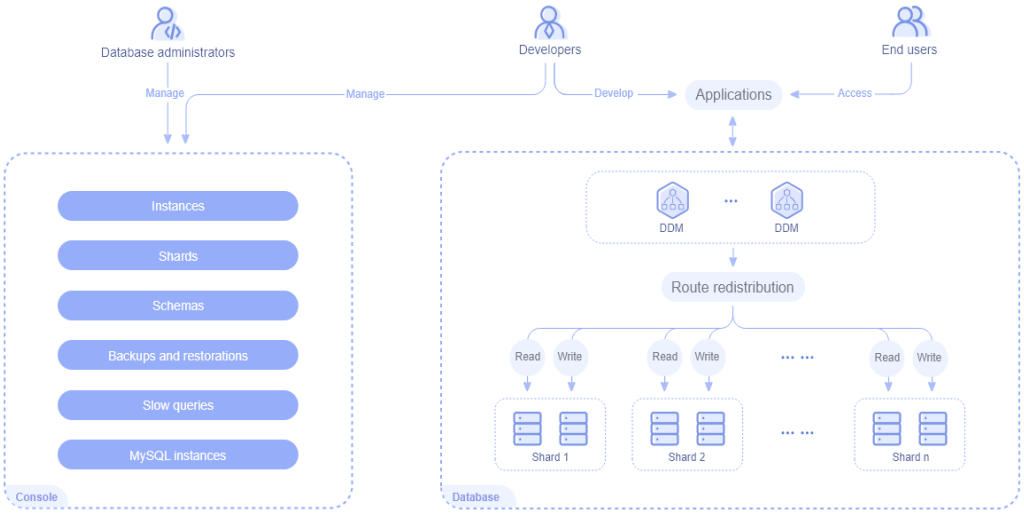
  MySQL databases can support only medium- and small-scale service systems because their CPU, memory, and network processing are limited by server configurations and their storage depends on the size of SSD or EVS disks.

  DDM supports both compute and storage scaling. You can add nodes to a DDM instance or scale up its node class. Alternatively, increase shards or data

nodes to distribute data from one large table to multiple tables or scale out storage resources as services grow, without worrying about O&M.

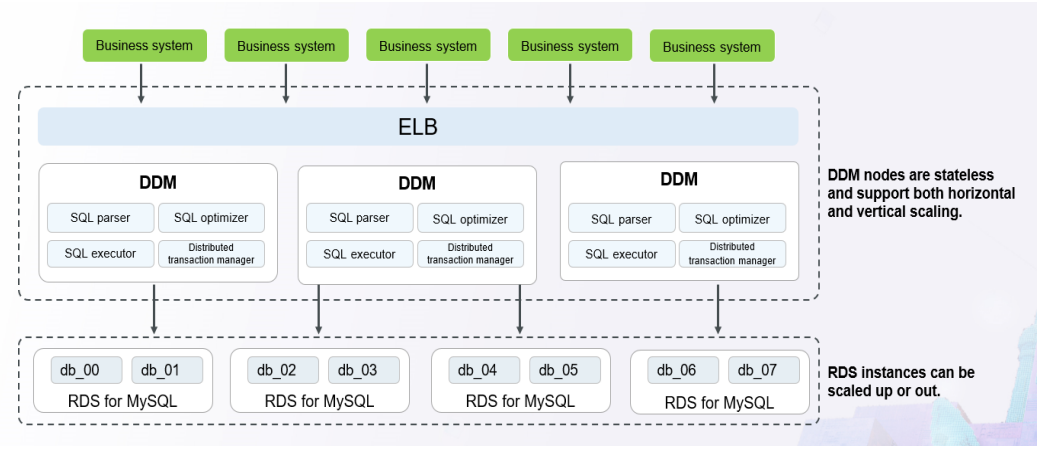
Service Architecture

Figure 1-1 DDM service architecture



How DDM Works

Figure 1-2 DDM working diagram



1.2 Basic Concepts

Data Node

A data node is the minimum management unit of DDM. Each data node represents an independently running database, and it may be an RDS for MySQL instance that is associated with your DDM instance. You can create multiple schemas in a DDM instance to manage data nodes and access each data node independently.

A data node is the minimum management unit of DDM. Each data node is an RDS for MySQL instance, which represents an independently running database. You can create multiple schemas in a DDM instance to manage data nodes and access each data node independently.

NOTE

DDM instances do not store service-related data, which is stored in shards of data nodes.

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.

1.3 Core Functions

DDM is compatible with SQL syntax of multiple versions of MySQL and supports a variety of functions, including horizontal sharding, shard configuration, read/write splitting, and global sequence.

Table 1-1 DDM core functions

Function	Description
Horizontal sharding	Select a sharding key when creating a logical table. DDM will generate a sharding rule and horizontally shard data. **NOTE** A sharding key is a table field used to generate a route during horizontal partitioning of logical tables. After specifying a table field, you can select a date function or manually enter *date function (field name)*. The table field must be **date**, **datetime**, or **timestamp**. Select a date function if data needs to be redistributed by year, month, day, week, or some combinations thereof.
Flexible shard configuration	DDM supports both compute and storage scaling. You can add nodes to a DDM instance or scale up its node class. Alternatively, increase shards or data nodes to distribute data from one large table to multiple tables or scale out storage resources. Compute scaling is undetectable to your applications. Storage scaling minimizes service interruption to seconds.
Distributed transactions	DDM processes three types of transactions, including single-shard, FREE, and Extended Architecture (XA). ● 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 executed by any shard, causing data inconsistency. ● XA: Transactions are committed in two phases. If a transaction fails to be executed by any shard, all work done will be rolled back to ensure data consistency.
Data import and export	External data can be imported into DDM instances to help you migrate databases to the cloud, and DDM instance data can be exported based on service requirements.
Highly compatible SQL syntax	DDM is compatible with MySQL 5.7 and 8.0 licenses and syntax.
Read and write splitting	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 instance associated with your DDM instance and configure a read policy, and a large number of concurrent requests can read data from those read replicas.
Global sequence	DDM allows you to use globally unique, distributed, and ascending SNs as primary or unique keys or to meet your requirements in specific scenarios.

Function	Description
Online monitoring	DDM monitors reads, writes, and slow query logs of your DDM instance on the console, helping you detect resource and performance bottlenecks as fast as possible.
O&M console	DDM enables you to manage and maintain DDM instances, schemas, and accounts on a visualized console.

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 you create a DDM instance, you can associate it with RDS for MySQL instances in the same VPC to obtain separated storage resources.
- Cloud Eye

  Cloud Eye monitors DDM metrics and sends notifications in the event of an alarm or event.
- Elastic Load Balance (ELB)

  ELB distributes incoming traffic to multiple backend servers based on forwarding policies to balance workloads. So, it can expand external service capabilities of DDM and eliminate single points of failure (SPOFs) to improve service availability.

1.4 Product Specifications

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 Product specifications

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

1.5 Use Constraints

1.5.1 Network Access

Restrictions on DDM network access are as follows:

- The data nodes and ECSs running your applications must be in the same VPC as your DDM instance.
- To access DDM from your computer, you need to bind an EIP to your DDM instance and then use the EIP to access the DDM instance.

1.5.2 Data Nodes

Constraints on data nodes are as follows:

- DDM can work with only RDS for MySQL 5.7 and 8.0 instances.
- DDM does not support SSL for MySQL instances.
- Do not enable case sensitivity support for MySQL instances.

  NOTE

  - Select **Case insensitive** for **Table Name** when you create an instance.
- When you modify configurations of a MySQL instance associated with DDM, an exception may occur. After the modification, click **Synchronize Data Node Information** on the **Data Nodes** page to synchronize changes from the data node to DDM.
- Do not use character set GBK for data nodes.

1.5.3 Unsupported Features and Limitations

Unsupported Features

- Stored procedures
- Triggers
- Views
- Events
- User-defined functions
- Foreign key references and associations
- Full-text indexes and SPACE functions
- Temporary tables
- Compound statements such as BEGIN...END, LOOP...END LOOP, REPEAT...UNTIL...END REPEAT, and WHILE...DO...END WHILE
- Process control statements such as IF and WHILE
- RESET and FLUSH statements
- BINLOG statement
- HANDLER statement

- INSTALL and UNINSTALL PLUGIN statements
- Character sets other than ASCII, Latin1, binary, utf8, and utf8mb4
- SYS schema
- MySQL Optimizer Trace
- X-Protocol
- CHECKSUM TABLE syntax
- Table maintenance statements, including CHECK, CHECKSUM, OPTIMIZE, and REPAIR TABLE
- Statements for assigning a value to or querying variable **session**

  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
- Incomplete support for system variable queries. The returned values are variable values of RDS instances, instead of DDM kernel variable values. For example, the returned values of SELECT @@autocommit do not indicate the current transaction status. For example, the value returned by select @@autocommit does not indicate the current transaction status of DDM.
- Executing SET syntax to modify global variables
- PARTITION syntax. Partitioned tables are not recommended.
- LOAD XML statement
- Inline comments
- CREATE TABLE AS WITH SELECT syntax
- ZEROFILL CREATE syntax

Unsupported Operators

- Assignment operator **:=**.
- Arrow operator **->**. This operator can be executing on a single table. Errors will be displayed if the operator is executed on other types of tables.
- Arrow operator **->>**. This operator can be executing on a single table. Errors will be displayed if the operator is executed on other types of tables.
- Expression IS UNKNOWN
- Operators =, <, <=, >, >=, <>, !=, <=> that require comparing JSON fields at the DDM compute layer

Unsupported Functions

The compute layer does not support the following functions.

- XML functions
- Function **ANY_VALUE()**
- Function **ROW_COUNT()**
- Function **COMPRESS()**
- Function **SHA()**

- Function **SHA1()**
- Function **AES_ENCRYPT()**
- Function **AES_DECRYPT()**
- Aggregate function **JSON_OBJECTAGG()**
- Aggregate function **JSON_ARRAYAGG()**
- Function **JSON_CONTAINS()**
- Aggregate function **STD()**
- Aggregate function **STDDEV()**
- Aggregate function **STDDEV_POP()**
- Aggregate function **STDDEV_SAMP()**
- Aggregate function **VAR_POP()**
- Aggregate function **VAR_SAMP()**
- Aggregate function **VARIANCE()**
- Function **MICROSECOND()**
- Function **TO_DAYS()**
- Function **TO_SECONDS()**
- Function **UNCOMPRESS()**
- Function **UNCOMPRESSED_LENGTH()**
- Function **UNHEX()**
- Function **YEARWEEK()**
- Function **TIME_FORMAT()**

SQL Syntax Limitations

Unsupported SELECT syntax

- DISTINCTROW
- Configuring options [HIGH_PRIORITY], [STRAIGHT_JOIN], [SQL_SMALL_RESULT], [SQL_BIG_RESULT], [SQL_BUFFER_RESULT], and [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] in SELECT statements on DDM instances
- SELECT ... GROUP BY ... WITH ROLLUP
- SELECT ... ORDER BY ... WITH ROLLUP
- WITH
- JOIN statements with different collations
- Window functions
- SELECT FOR UPDATE supports only simple queries and does not support statements such as JOIN, GROUP BY, ORDER BY, and LIMIT. Option [NOWAIT | SKIP LOCKED] for modifying FOR UPDATE is invalid for DDM.
- DDM does not support multiple columns with the same name for each SELECT statement in UNION.

  For example, duplicate column names are used in the following SELECT statement:

  ```
  SELECT id, id, name FROM t1 UNION SELECT pk, pk, name FROM t2;
  ```


SORT and LIMIT

LIMIT/OFFSET, allowed value range: 0–2147483647

Aggregation

Function **asc** or **desc** cannot be used in the GROUP BY statement to sort out results.

NOTE

- DDM automatically ignores keyword **asc** or **desc** after GROUP BY.
- In MySQL versions earlier than 8.0.13, function **asc** or **desc** can be used in the GROUP BY statement to sort out results. In MySQL 8.0.13 or later, a syntax error is reported if you use function **asc** or **desc** this way. ORDER BY is recommended for sorting.

Subqueries

- Subqueries that join grandparent queries are not supported.
- Subqueries in the HAVING clause and the JOIN ON condition are not supported.
- Each derived table must have an alias.
- A derived table cannot be a correlated subquery.

LOAD DATA

- LOW_PRIORITY is not supported.
- CONCURRENT is not supported.
- PARTITION (partition_name [, partition_name] ...) is not supported.
- LINES STARTING BY 'string' is not supported.
- User-defined variables are not supported.
- ESCAPED BY supports only '\\'.
- If you have not specified a value for your auto-increment key when you insert a data record, DDM will not fill a value for the key. The auto-increment keys of data nodes of a DDM instance all take effect, so the auto-increment key values may be duplicate.
- If the primary key or unique index is not routed to the same physical table, REPLACE does not take effect.
- If the primary key or unique index is not routed to the same physical table, IGNORE does not take effect.
- Executing LOAD DATA statements is not allowed on the tables that contain global secondary indexes.

INSERT and REPLACE

- INSERT DELAYED is not supported.
- Only INSERT statements that contain sharding fields are supported.
- PARTITION syntax is not supported. Partitioned tables are not recommended.
- Setting **YYYY** of **datetime** (in the format of **YYYY-MM-DD HH:MM:SS**) to **1582** or any value smaller in INSERT statements is not supported.
- Nesting a subquery in ON DUPLICATE KEY UPDATE of an INSERT statement is not supported. The following is an example:


```
INSERT INTO t1(a, b)
SELECT * FROM(SELECT c, d FROM t2 UNION SELECT e, f FROM t3) AS dtest
ON DUPLICATE KEY UPDATE b = b + c;
```

Subquery c is used in the ON DUPLICATE KEY UPDATE clause.

- The sharding key values in INSERT and REPLACE statements cannot be **DEFAULT**.

UPDATE and DELETE

- Updating a sharding key value to **DEFAULT** is not supported.
- Repeatedly updating the same field in one SQL statement is not supported.
- Updating a sharding key using UPDATE JOIN is not supported. The following is an example:

```
UPDATE tbl_1 a, tbl_2 b set a.name=b.name where a.id=b.id;
```

name indicates the sharding key of table **tbl_1**.

- Updating a sharding key by executing INSERT ON DUPLICATE KEY UPDATE is not supported.
- Updating self-joins is not supported.

```
UPDATE tbl_1 a, tbl_1 b set a.tinyblob_col=concat(b.tinyblob_col, 'aaabbb');
```

- Updating sharding keys in subqueries is not allowed for secondary sharded tables that contain JSON fields.
- UPDATE JOIN supports only joins with WHERE conditions.
- Referencing other object columns in assignment statements or expressions is not supported when UPDATE JOIN syntax is used. The following is an example:

```
UPDATE tbl_1 a, tbl_2 b SET a.name=concat(b.name, 'aaaa'),b.name=concat(a.name, 'bbbb') ON
a.id=b.id;
```

- You can update a sharding field by two steps: delete the original sharding field and then insert a new field. During this process, the results of querying the sharding fields involved in the target table may be inconsistent.

DDL

- Modifying database names and sharding field names and types is not allowed.
- Creating or deleting schemas by executing SQL statements is not supported.
- Index FULL_TEXT is not supported.
- AS SELECT clause of the CREATE TABLE statement is not supported.
- CREATE TABLE ... LIKE statement is not supported.
- Dropping multiple tables with one SQL statement is not supported.
- Executing multiple DDL statements at the same time is not supported.
- Creating foreign keys for broadcast and sharded tables is not supported.
- Creating tables whose names are prefixed by **_ddm** is not supported.
- Creating temporary sharded or broadcast tables is not supported.
- Specifying globally unique keys in the CREATE TABLE statement is not supported.
- Modifying global secondary indexes is not supported.

Indexes

- Global unique indexes are not supported. Unique keys and primary keys may not be globally unique.
- Values of primary key and sharding key fields cannot exceed the corresponding data range.
- If you want to use a global secondary index, set **sql_mode** to **STRICT_TRANS_TABLES**.

Table Recycle Bins: Unsupported Functions

- Hints
- Deleting tables by schema
- Deleting tables by logical table
- After a table is recovered, its globally unique sequence increases automatically but may not follow the last sequence value.
- Shard configuration
- Retaining copies with no time limit
- Recovering data to a table with any name
- Unlimited copies

Transactions: Unsupported Functions

- Savepoints
- XA syntax. DDM has implemented distributed transactions through XA, so the user layer does not need to process the syntax.
- Customizing the isolation level of a transaction. Currently, DDM supports only the READ COMMITTED isolation level. In consideration of compatibility, DDM does not report errors for any SQL statement (such as SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ) to set the database isolation level, but will ignore the modifications to the transaction isolation level.
- Setting a transaction to read-only (START TRANSACTION READ ONLY). DDM can enable read/write of a transaction, instead of enabling read-only, to ensure compatibility.

Permissions

- Column-level permissions
- Subprogram-level permissions

Database Management Statements

- SHOW TRIGGERS statements are not supported.
- Most of SHOW statements such as SHOW PROFILES, SHOW ERRORS, and SHOW WARNINGS are not supported.
- The following SHOW statements are randomly sent to a database shard. If database shards are on different RDS for MySQL instances, the returned variables or table information may be different.

  - SHOW TABLE STATUS
  - SHOW VARIABLES Syntax
  - SHOW WARNINGS Syntax does not support the combination of LIMIT and COUNT.
  - SHOW ERRORS Syntax does not support the combination of LIMIT and COUNT.

INFORMATION_SCHEMA

Only simple queries of SCHEMATA, TABLES, COLUMNS, STATISTICS, and PARTITIONS are supported. No subqueries, JOINs, aggregate functions, ORDER BY, and LIMIT are allowed.

Broadcast Tables

The DDM broadcast table mechanism is that each SQL statement can be executed on all shards of a broadcast table. When you use such a broadcast table, do not use any function that has different results returned each time it is executed. Otherwise, data inconsistency will occur between different broadcast table shards. If such functions are indeed required, calculate their results, write the results to your SQL statements, and then execute the SQL statements on the broadcast table. Functions of this type include but are not limited to the following:

- CONNECTION_ID()
- CURDATE()
- CURRENT_DATE()
- CURRENT_TIME()
- CURRENT_TIMESTAMP()
- CURTIME()
- LAST_INSERT_ID()
- LOCALTIME()
- LOCALTIMESTAMP()
- NOW()
- UNIX_TIMESTAMP()
- UTC_DATE()
- UTC_TIME()
- UTC_TIMESTAMP()
- CURRENT_ROLE()
- CURRENT_USER()
- FOUND_ROWS()
- GET_LOCK()
- IS_FREE_LOCK()
- IS_USED_LOCK()
- JSON_TABLE()
- LOAD_FILE()

- MASTER_POS_WAIT()
- RAND()
- RELEASE_ALL_LOCKS()
- RELEASE_LOCK()
- ROW_COUNT()
- SESSION_USER()
- SLEEP()
- SYSDATE()
- SYSTEM_USER()
- USER()
- UUID()
- UUID_SHORT()

1.5.4 High-risk Operations

Pay attention to the following when you use DDM:

- Do not connect to any data node for data operations to avoid deleting system catalogs or metadata by mistake.
- Do not clear system tables **TBL_DRDS_TABLE** and **MYCAT_SEQUENCE** to prevent metadata loss.

1.6 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.

shows the relationship between regions and AZs.

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.

1.7 Application Scenarios

As an OLTP-oriented service, DDM allows you to access 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 management, smart city extension, 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.

- **Traditional sectors**

  Government agencies, large-sized enterprises, banks, and the like usually 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.

1.8 Permissions Management

If your account does not need individual IAM users for permissions management, you may skip this section.

If you need to assign different permissions to employees in your enterprise to access your DDM resources, IAM is a good choice for fine-grained permissions management. IAM provides functions like identity authentication, permissions management, and access control, helping you secure access to your cloud resources.

You can create IAM users for your employees, and assign permissions to these users to control their access to specific types of resources. For example, you can create IAM users for software developers and assign specific permissions to allow them to use DDM resources but disallow them to delete the resources or perform any high-risk operations.

DDM Permissions

By default, new IAM users do not have any permissions assigned. To assign permissions to these new users, you need to add them to one or more groups, and attach permissions policies or roles to these groups.

DDM is a project-level service deployed in specific physical regions. When you assign DDM permissions to a user group, you need to specify region-specific projects where the permissions will take effect. If you select **All projects**, the permissions will be granted for all region-specific projects. To access DDM, you need to switch to the region where you are authorized.

You can grant users permissions using roles and policies.

- Roles: A type of coarse-grained authorization mechanism that provides only a limited number of service-level roles. When using roles to grant permissions, you also need to assign other dependent roles. Roles are not ideal for fine-grained authorization and secure access control.

- Policies: A fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization and more secure access control. For example, you can grant IAM users only the permissions for managing a certain type of DDM resources.

  Table 1-3 System-defined policies

Policy Name	Description	Type	Dependency
DDM FullAccess	Full permissions for Distributed Database Middleware	System-defined policy	None
DDM CommonOperations	Common permissions for Distributed Database Middleware, excluding the permissions to create, delete, and add nodes, configure shards, and roll back shard configuration tasks	System-defined policy	None
DDM ReadOnlyAccess	Read-only permissions for Distributed Database Middleware	System-defined policy	None

The following are permission configurations of supported system-defined policies:

- DDM FullAccess

```
{
    "Version": "1.1",
     "Statement": [{
      "Action": ["ddm:*:*",
      "rds:instance:list",
          "rds:instance:modify",
          "rds:instance:modifyParameter",
      "vpc:*:*",
          "ecs:*:get*",
          "ecs:*:list*",
          "ecs:cloudServerNics:update",
          "ecs:serverInterfaces:use"],
       "Effect": "Allow"
      }]
  }
```

- DDM CommonOperations

```
{
   "Version": "1.1",
   "Statement": [{
              "Action": [
                  "vpc:*:list*",
                  "vpc:*:get*",
                  "vpc:ports:update",
                  "ecs:*:get*",
                  "ecs:*:list*",
                  "rds:instance:list",
                  "rds:instance:modify",
                  "rds:instance:modifyParameter"
              ],
```



```
            "Effect": "Allow"
            },
            {
                "Condition": {
                    "StringEqualsIgnoreCase": {
                        "g:ServiceName": [
                            "ddm"
                        ]
                    }
                },
                "NotAction": [
                    "ddm:instance:create",
                    "ddm:instance:delete",
                    "ddm:database:migrate*",
                    "ddm:instance:resize",
                    "ddm:instance:extendNode"
                ],
                "Effect": "Allow"
        }]
    }
```

- DDM ReadOnlyAccess

```
{
    "Version": "1.1",
    "Statement": [{
                "Action": [
                    "rds:instance:list",
                    "vpc:*:*list*",
                    "vpc:*:*get*",
                    "ecs:*:get*",
                    "ecs:*:list*",
                    "ddm:*:list*",
                    "ddm:*:get*",
                    "ddm:instance:listParameter",
                    "ddm:instance:listRwInfo",
                    "ddm:instance:listSlowSqlInfo",
                    "ddm:rds:connectivity"
                ],
                "Effect": "Allow"
        }]
    }
```

Table 1-4 lists the common operations supported by each DDM system-defined policy. Choose appropriate system-defined policies based on your requirements.

Table 1-4 Common operations supported by each system-defined policy

Operation	DDM FullAccess	DDM CommonOperations	DDM ReadOnlyAccess
Querying DDM instances	Supported	Supported	Supported
Querying details of a DDM instance	Supported	Supported	Supported
Modifying instance information, including the name and security group	Supported	Supported	Not supported
Restarting a DDM instance	Supported	Supported	Not supported

Operation	DDM FullAccess	DDM CommonOperations	DDM ReadOnlyAccess
Creating a DDM instance	Supported	Not supported	Not supported
Deleting a DDM Instance	Supported	Not supported	Not supported
Changing node class	Supported	Not supported	Not supported
Scaling out a DDM instance	Supported	Not supported	Not supported
Creating a schema	Supported	Supported	Not supported
Querying schemas	Supported	Supported	Supported
Querying details of a schema	Supported	Supported	Supported
Viewing the size of a logical table	Supported	Supported	Supported
Viewing physical table fragments	Supported	Supported	Supported
Performing a rollback if configuring shards fails Deleting source data if configuring shards fails Retrying if configuring shards fails	Supported	Not supported	Not supported
Deleting a schema	Supported	Supported	Not supported
Querying accounts	Supported	Supported	Supported
Creating an account	Supported	Supported	Not supported
Modifying an account	Supported	Supported	Not supported
Resetting a password	Supported	Supported	Not supported
Deleting an account	Supported	Supported	Not supported
Synchronizing data node information	Supported	Supported	Not supported
Querying data nodes	Supported	Supported	Supported
Querying details of a data node	Supported	Supported	Supported
Modifying the read policy of a data node	Supported	Supported	Not supported
Viewing products	Supported	Supported	Supported

Operation	DDM FullAccess	DDM CommonOperations	DDM ReadOnlyAccess
Creating a parameter template	Supported	Supported	Not supported
Deleting a parameter template	Supported	Supported	Not supported
Applying a parameter template	Supported	Supported	Not supported
Modifying a parameter template	Supported	Supported	Not supported
Replicating a parameter template	Supported	Supported	Not supported
Comparing two parameter templates	Supported	Supported	Supported
Querying parameter templates	Supported	Supported	Supported
Viewing all tags	Supported	Supported	Supported
Adding, modifying, or deleting a tag	Supported	Supported	Not supported
Querying a session	Supported	Supported	Supported
Killing a session	Supported	Supported	Not supported
Querying slow query logs	Supported	Not supported	Not supported
Exporting slow query logs	Supported	Not supported	Not supported
Creating a slow query log download task	Supported	Not supported	Not supported
Querying whether SQL concurrency control is enabled for DNs	Supported	Supported	Supported
Configuring whether SQL concurrency control is enabled for DNs	Supported	Supported	Not supported
Viewing SQL concurrency control rules for DNs	Supported	Supported	Supported
Creating a SQL concurrency control rule for DNs	Supported	Supported	Not supported

Operation	DDM FullAccess	DDM CommonOperations	DDM ReadOnlyAccess
Deleting a SQL concurrency control rule for DNs	Supported	Supported	Not supported
Generating keywords for a physical SQL statement	Supported	Supported	Supported
Generating keywords for a logical SQL statement	Supported	Supported	Supported

Table 1-5 Common operations and supported actions

Operation Category	Operation	Action
DDM routine operations	Applying for a DDM instance	ddm:instance:create Before you create a DDM instance, obtain the following dependent permissions: • ecs:*:get* • ecs:*:list* • vpc:vpcs:list • vpc:securityGroups:get • vpc:subnets:get • ecs:cloudServerNics:update • ecs:serverInterfaces:use • **vpc:ports:*** for a global or regional DDM instance
	Querying DDM instances	ddm:instance:list
	Querying details of a DDM instance	ddm:instance:get To view details of a DDM instance, you need to configure the following permissions: • vpc:*:get* • vpc:*:list*

Operation Category	Operation	Action
	Modifying instance information, including modifying the name, changing the security group, or adding, modifying, or deleting a tag of a DDM instance	ddm:instance:modify To modify a security group, you need to configure the following permissions: • vpc:*:get* • vpc:*:list* • vpc:ports:update
	Restarting a DDM instance	ddm:instance:reboot
	Deleting a DDM instance	ddm:instance:delete vpc:ports:delete
	Changing node class	ddm:instance:resize
	Scaling out a DDM instance	ddm:instance:extendNode
	Monitoring the read/write ratio	ddm:instance:listRwInfo
	Querying slow query logs	ddm:instance:listSlowSqlInfo
Schema operations	Creating a schema	ddm:database:create
	Querying schemas	ddm:database:list
	Querying details of a schema	ddm:database:get

Operation Category	Operation	Action
	Viewing the size of a logical table	ddm:database:listTable
	Viewing physical table fragments	ddm:table:list
	Performing a rollback if configuring shards fails Deleting source data if configuring shards fails Retrying if configuring shards fails	ddm:database:migrateRollback
	Deleting a schema	ddm:database:delete
DDM account operations	Querying accounts	ddm:user:list
	Creating an account	ddm:user:create
	Modifying an account	ddm:user:modify
	Resetting a password	ddm:user:modify
	Deleting an account	ddm:user:delete
Data node management (using an RDS for MySQL instance as an example)	Synchronizing data node information	ddm:rds:synchro To synchronize data node information, you need to configure the following permissions: ● rds:instance:list ● rds:instance:modify ● rds:instance:modifyParameter
	Querying data nodes	ddm:rds:list

Operation Category	Operation	Action
	Querying details of a data node	ddm:rds:get
	Modifying the read policy of a data node	ddm:rds:modifyReadPolicy
DDM product operations	Viewing products	ddm:product:list
Parameter template operations	Creating a parameter template	ddm:param:create
	Deleting a parameter template	ddm:param:delete
	Applying a parameter template	ddm:param:apply
	Modifying a parameter template	ddm:param:update
	Replicating a parameter template	ddm:param:create
	Comparing two parameter templates	ddm:param:list
	Querying parameter templates	ddm:param:list
Tag operations	Querying the tag list	ddm:tag:list
	Adding, modifying, or deleting a tag	ddm:tag:modify

Operation Category	Operation	Action
Session operations	Querying a session	ddm:instance:queryProcessList
	Killing a session	ddm:instance:killProcessList
SQL concurrency control	Querying whether SQL concurrency control is enabled for DNs	ddm:instance:get
	Configuring whether SQL concurrency control is enabled for DNs	ddm:instance:modify
	Viewing SQL concurrency control rules for DNs	ddm:instance:get
	Creating a SQL concurrency control rule for DNs	ddm:instance:modify
	Deleting a SQL concurrency control rule for DNs	ddm:instance:modify
	Generating keywords for a physical SQL statement	ddm:instance:get

Operation Category	Operation	Action
	Generating keywords for a logical SQL statement	ddm:instance:get

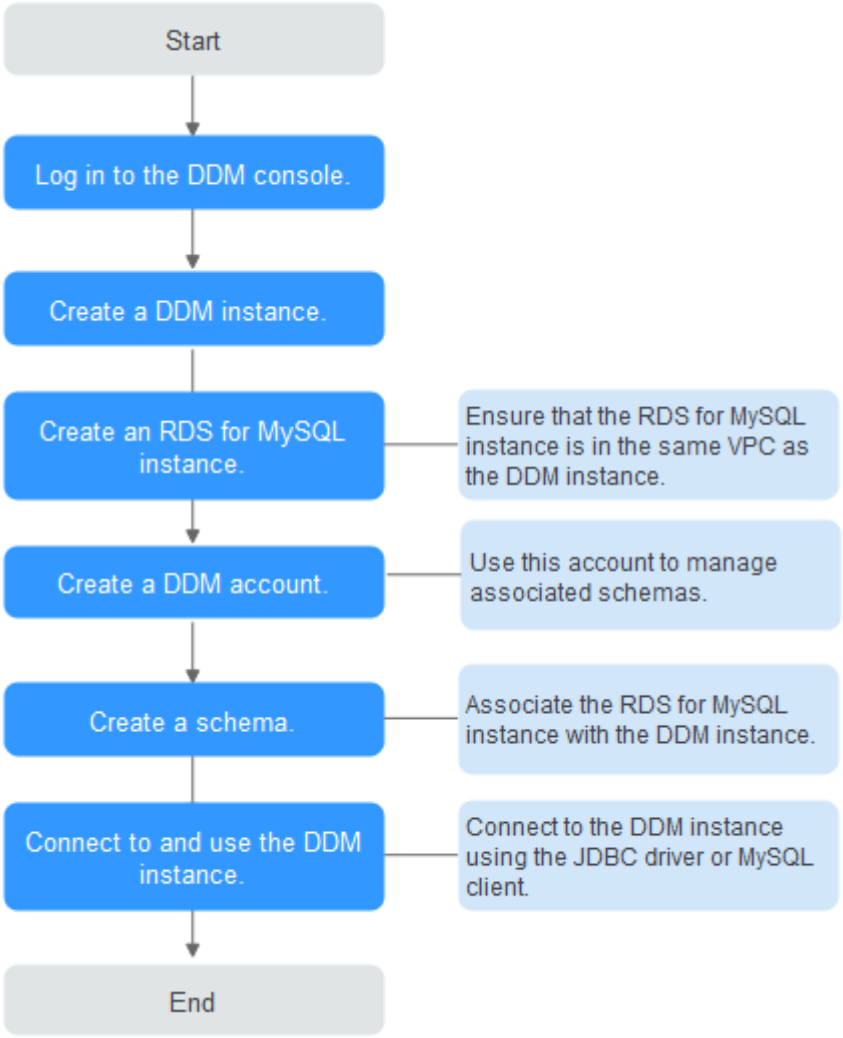
2 Getting Started

2.1 Process of Using DDM

This section uses an RDS for MySQL instance as an example to describe how to associate a DDM instance with a data node (RDS for MySQL instance) and how to use DDM.

Process of Using DDM

Figure 2-1 Flowchart of Using DDM



2.2 Precautions for Creating a DDM Instance

Before creating a DDM instance, you need to make the required preparations, including evaluating the instance class, determining the VPC and AZ to be used, and applying for an account.

After registering an account, use IAM to create an IAM user and a user group, and grant specific permissions to the IAM user so that you can perform refined management on your resources.

Selecting an Instance Class

Estimate compute and storage requirements of your application and then select an appropriate instance class based on these requirements and your service type and scale so that the DDM instance you will buy can better meet your application requirements. The instance class includes vCPUs and memory.

Determining a VPC

A VPC provides logical isolation for network access. When creating a VPC, you can define its network features, such as the security group, VPN, and subnet. Using a VPC, you can easily manage and configure the intranet, and modify networks quickly and securely.

The DDM instance you purchase must be in the same VPC as your application and RDS for MySQL instance to ensure network connectivity.

You are advised to select the same security group for your DDM instance, application, and RDS MySQL instance so that they can communicate with each other. If different security groups are selected, add security group access rules to enable such network access.

Choosing an AZ

Configure your application, DDM instance, and required RDS instances in the same AZ to reduce network latency.

2.3 Step 1: Create a DDM Instance and an RDS for MySQL Instance

This section describes how to create a DDM instance and an RDS for MySQL instance on the console.

Creating a DDM instance

Step 1 Log in to the console.

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

Step 3 Click **Service List** and choose **Databases** > **Distributed Database Middleware**.

Step 4 On the displayed page, in the upper right corner, click **Create DDM Instance**.

Step 5 On the displayed page, configure required parameters.

Table 2-1 Parameter description

Parameter	Description
Region	Region where the DDM instance is located. Select the required region.
Project	Project that the DDM instance belongs to.

Parameter	Description
Instance Name	Name of the DDM instance, which: • Cannot be left blank. • Must start with a letter. • Must be 4 to 64 characters long. • Can contain letters, digits, underscores (_), and hyphens (-).
Time Zone	You need to select a time zone for your instance based on the region hosting your instance.
Quantity	Number of nodes in a DDM instance. Up to 32 nodes are supported. **NOTE** • At least 2 nodes are recommended because using a single node cannot guarantee high availability. • Disk usage of a single node: system disk 40 GB and data disk 1 GB
AZ	Availability zone where the DDM instance is deployed. Nodes in a DDM instance can be deployed on different physical servers in the same AZ to keep services always available even if one physical server becomes faulty. A DDM instance can be deployed across AZs to provide cross-AZ DR. If necessary, you can select multiple AZs when you create a DDM instance. Then nodes of the instance will be deployed in multiple different AZs. **NOTE** Deploy your application, DDM instance, and required RDS instances in the same AZ to reduce network latency. Cross-AZ deployment may increase network latency.
Node Class	Class of the DDM instance node. You can select **General-enhanced** and then specify a node class. **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 node class so that the CPU and memory specifications of your DDM instance can better meet your needs.
Password	Select **Configure** or **Skip**. **NOTE** Configure information about an administrator account. If you select **Skip**, you can configure information about an administrator account later as needed by following **Creating an Administrator Account**.
Administrator	This parameter is mandatory when **Configure** is selected for **Password**. Enter an administrator account name, which cannot be the same as any account on the **Accounts** page. It cannot be deleted after being created.

Parameter	Description
Password	This parameter is mandatory when **Configure** is selected for **Password**. Set the password of the administrator account. **NOTE** You can reset the password later as needed by following **Resetting the Administrator Password**.
Confirm Password	This parameter is mandatory when **Configure** is selected for **Password**. Enter the password of the administrator account again.
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 for MySQL instance. • To ensure network connectivity, the DDM instance you created must be in the same VPC as your applications and RDS for MySQL instances. • After the DDM instance is created, the VPC cannot be changed. Exercise caution when performing this operation.
Subnet	A subnet provides dedicated network resources that are logically isolated from other networks for network security. A floating IP address is automatically assigned when you create a DDM instance. After the instance is created, you can change the floating IP address.
Security Group	Select an existing security group. You are advised to select the same security group for your DDM instance, application, and RDS for MySQL instances so that they can communicate with each other. If different security groups are selected, add security group rules to enable network access.
Enterprise Project	EPS provides a unified method to manage cloud resources and personnel by enterprise project.
Parameter Template	Select an existing parameter template. You can also click **View Parameter Template** to set parameters on the displayed page.

Parameter	Description
Tags	(Optional) Adding tags helps you better identify and manage your DDM resources. You can add tags to your DDM instance. Each instance can have a maximum of 20 tags. • Creating a tag You can create tags on the DDM console. A tag key and a value are required when you create a tag. **Tag key: This parameter is mandatory and cannot be null.** – Must be unique for each instance. – Can include 1 to 36 characters. – Cannot be an empty string, or start with **_sys_**, and cannot start or end with a space. – Cannot contain the following characters: Non-printable ASCII characters (0-31), "*", "<", ">", "\", ",", "|", "/" **Tag value: This parameter is mandatory.** – Is an empty string by default. – Can contain 0 to 43 characters. – Cannot contain the following characters: Non-printable ASCII characters (0-31), "*", "<", ">", "\", ",", "|" After an instance is created, you can click the instance name to view its tags, modify or delete the tags on the **Tags** tab page. In addition, you can quickly search for and filter specified instances by tag. You can add a tag to an instance after the instance is created.

Step 6 After the configuration is complete, click **Next** at the bottom of the page.

Step 7 Confirm the configurations and click **Submit**.

Step 8 To view and manage the instance, go to the **Instances** page.

The DDM service port is **5066** by default and can be changed after a DDM instance is created.

For details, see **Changing a DDM Service Port**.

----End

Creating an RDS for MySQL Instance

Step 1 Log in to the console.

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

Step 3 Click ☰ and choose **Databases** > **Relational Database Service**.

Step 4 On the **Instances** page, click **Create DB Instance** in the upper right corner.

Step 5 On the displayed page, configure required parameters.

For details, see section "Create a DB Instance" in *RDS for MySQL User Guide*.

CAUTION

- A DDM instance can be associated with RDS for MySQL instances of versions 5.7 and 8.0.
- The RDS for MySQL instance must be in the same VPC and subnet as your DDM instance. If they are not in the same subnet, configure routes to ensure network connectivity.
- Specifications of associated RDS for MySQL instances should be greater than that of the DDM instance. Otherwise, the performance will be affected.

Step 6 After the configuration is complete, click **Next** at the bottom of the page.

Step 7 Confirm the configurations and click **Submit**. Wait 1 to 3 minutes for the RDS instance to be created.

----End

2.4 Step 2: Create a DDM Account

This section describes how to create a DDM account on the console.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the required DDM instance and click its name.

Step 3 In the navigation pane, choose **Accounts**.

Step 4 On the displayed page, click **Create Account**.

Step 5 In the displayed dialog box, configure parameters as required.

Table 2-2 Parameter description

Parameter	Description
Username	Username of the account. The username can include 1 to 32 characters and must start with a letter. Only letters, digits, and underscores (_) are allowed.

Parameter	Description
Password	Password of the account. The password: • Can include 8 to 32 characters. • Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters ~!@#%^*-_+? • Cannot be a weak password. It cannot be overly simple and easily guessed. • Cannot be the username or the username spelled backwards.
Confirm Password	The confirm password must be the same as the entered password.
Password Validity	Password validity period. The value must be an integer ranging from 0 to 65535, in days. An expired account cannot be used to log in to the system. You need to reset the password and log in again. • If the value is **0**, the password never expires. • If this parameter is not set, the password will always be valid. **NOTE** The DDM kernel version must be 3.0.4 or later.
Schema	Schema to be associated with the DDM account. You can select an existing schema from the drop-down list. The account can be used to access only the associated schemas.
Permissions	Options: **CREATE**, **DROP**, **ALTER**, **INDEX**, **INSERT**, **DELETE**, **UPDATE**, and **SELECT**. You can select any or a combination of them.
Description	Description of the account, which cannot exceed 256 characters.

Step 6 Click **OK**.

----End

2.5 Step 3: Create a Schema and Associate It with an RDS for MySQL Instance

This section describes how to create a schema on the DDM console and associate it with an RDS for MySQL instance.

Precautions

- When you create a schema, the RDS for MySQL instance and your DDM instance must be in the same VPC, and the RDS for MySQL instance is not

used by other DDM instances. DDM will create databases on the selected RDS for MySQL instances, without affecting their existing databases and tables.

- All instances associated with one schema must have the same major RDS for MySQL version.
- Multiple schemas can be created in a DDM instance. Multiple schemas can be associated with an RDS for MySQL instance.
- One RDS for MySQL instance cannot be associated with schemas in different DDM instances.
- If you create a sharded schema, more than one shard will be generated in the schema. Shard names will follow the rule: *<schemaname>_<number>*. *<number>* here indicates a four-digit number starting from 0000. This number will be incremented by one. For example, if a schema name is **db_cbb5** and there are 2 shards, the shard names are **db_cbb5_0000** and **db_cbb5_0001**.
- Read-only instances cannot be associated with the schema as data nodes.
- Do not modify or delete the internal accounts (DDMRW*, DDMR*, and DDMREP*) created on RDS for MySQL instances. Otherwise, services will be affected.

Prerequisites

- A DDM instance has been created and is running normally.
- A DDM account has been created. For details, see **Creating an Account**.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required DDM instance and click **Create Schema** in the **Operation** column.

Alternatively, click the instance name to go to the **Basic Information** page. On the displayed page, choose **Schemas** in the navigation pane and click **Create Schema** in the upper left corner of the page.

Step 3 On the displayed page, configure required parameters.

Table 2-3 Parameter description

Parameter	Description
Sharding	• **Sharded**: indicates that one schema can be associated with multiple data nodes, and all shards will be evenly distributed across the nodes. • **Unsharded**: indicates that one schema can be associated with only one data node, and only one shard can be created on the data node.
Schema	The name contains 2 to 48 characters and must start with a lowercase letter. Only lowercase letters, digits, and underscores (_) are allowed.

Parameter	Description
Account	The DDM account that needs to be associated with the schema.
Data Nodes	Select only RDS for MySQL instances that are in the same VPC as the DDM instance and are not used by other DDM instances. Databases can be created on the data nodes you select, without impacting existing databases and tables.
Shards	The total shards are the shards on all data nodes. There cannot be more data nodes than there are shards in the schema. Each data node must have at least one shard assigned. Recommended shards per data node: 8 to 64.

Step 4 Click **Next**.

Step 5 On the displayed page, enter a database account with the required permissions and click **Test Availability**.

NOTE

Required permissions: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, and TRIGGER WITH GRANT OPTION

You can create a database account for the RDS for MySQL instance and assign it the above permissions in advance.

Step 6 After the test becomes successful, click **Finish**.

----End

2.6 Step 4: Log In to the DDM 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 you log in to your DDM instance or schema, you have to obtain its connection address.

Obtaining the Schema Connection Address

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the required DDM instance and click its name.

Step 3 In the navigation pane on the left, choose **Schemas**. On the displayed page, click **View** in the **Connection Address** column.

Alternatively, locate the target schema, click its name to go to the **Basic Information** page. In the **Connection Address** tab page, obtain connection addresses for CLI and JDBC.

NOTE

- If load balancing is enabled, one floating IP address will be assigned to a DDM instance even if it has multiple nodes. You can use this address to connect to the DDM instance for load balancing.
- There are some historical instances that do not support load balancing, so they have multiple IP addresses. For load balancing, you can use JDBC connection strings to connect to them.
- If read-only groups are created, each group will be assigned a load balancing address for service isolation.

----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**.

NOTE

- For security purposes, select an ECS in the same VPC as your DDM instance.
- Ensure that a MySQL client has been installed on the required ECS or the MySQL connection driver has been configured.
- 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 Ask technical support to add an EIP to the feature whitelist. In the **Instance Information** area, click **Bind**. In the displayed dialog box, select the EIP and click **OK**. Bind the EIP with your DDM instance.

Step 3 In the navigation pane on the left, 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).

NOTE

Navicat12 is recommended for Navicat clients.

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 2-4 Parameter description

Example Parameter	Description	Example Value
DDM_SERVER_ADDRESS	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) UTF-8 character encoding system Configure this parameter if garbled characters are displayed during parsing due to inconsistency between MySQL connection code and actually used code.	-
default_auth=mysql_native_password	(Optional) Password authentication plug-in 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

NOTE

To perform the following operations, install the JDBC driver of 5.1.49 or later versions. JDBC driver download address: **https://dev.mysql.com/doc/index-connectors.html**.

Step 1 Load the required JDBC driver.

```
Class.forname(com.mysql.jdbc.Driver);
```

Step 2 Create a database connection.

```
String username = "EXAMPLE_USERNAME_ENV" ;
String password = "EXAMPLE_PASSWORD_ENV" ;
String url = "jdbc:mysql://ip:port/db_name";
Connection con = DriverManager.getConnection(url , username , password);
```

NOTE

- There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
- In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.

For details, see the example parameter value in **Table 2-5**.

Table 2-5 URL parameter description

Parameter	Description	Example Value
ip:port	Indicates the connection address and port number for connecting to the DDM instance.	192.168.0.200:5066 See **Obtaining the Schema Connection Address** to obtain the JDBC connection address and port number.
db_name	Indicates the name of a schema.	db_5505 On the DDM console, locate the required instance and click its name. In the navigation pane on the left, choose **Schemas** to view the schema name.

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&useSS
L=true";
```

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.
- For details about parameter configurations, see **Table 2-6**.

Table 2-6 Parameter description

Parameter	Description	Example Value
loadBalanceAutoCommitStatementThreshold	Indicates the number of statements executed before a reconnection. • If the parameter value is set to **5**, a reconnection is required after five SQL statements (queries or updates) are executed. • If it is set to **0**, a sticky connection is initiated instead of a reconnection. 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.	5
loadBalanceHostRemovalGracePeriod	Indicates the grace period to wait for a host being removed from a load balancing connection, and to be released when it is the active host.	15000
loadBalanceBlacklistTimeout	Indicates the time in milliseconds between checks of servers which are unavailable, by controlling how long a server lives in the global blacklist.	60000
loadBalancePingTimeout	Indicates the time in milliseconds for waiting for the ping response of each load balancing connection.	5000

Parameter	Description	Example Value
retriesAllDown	Indicates the maximum number of polling retries when all connection addresses fail. SQLException will be returned if the threshold of retries is reached with no valid connections obtained.	10
connectTimeout	Indicates the timeout interval for establishing a socket connection with a database server. Unit: millisecond. A value of **0** indicates that connection establishment never times out. This parameter setting is used for JDK 1.4 or later versions.	10000
socketTimeout	Indicates the timeout interval for a socket operation (read and write). Unit: millisecond. A value of **0** indicates that a socket operation never times out.	Set this parameter based on your service requirements.
useSSL	Indicates whether an SSL connection is used to connect to DDM.	true

----End

3 Function Overview

Distributed Database Middleware (DDM) is a MySQL-compatible, distributed middleware service designed for relational databases. It can resolve distributed scaling issues to break through capacity and performance bottlenecks of traditional databases, helping handle highly concurrent access to massive volumes of data.

Table 3-1 lists the functions supported by DDM.

Table 3-1 DDM functions

Category	Function
Permissions	Creating a user and granting the user permissions to use DDM and DDM custom policies. For details, see **Permissions Management**.
Instances	Creating, deleting, and restarting a DDM instance, and changing class of a DDM instance. For details, see **Instance Management**.
Backups	Restoring data to a new DDM instance and restoring metadata. For details, see **Backups and Restorations**.
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**.
Shard configuration	You can increase shards or data nodes to scale out storage. For details, see **Shard Configuration**.
Data nodes	Managing data nodes is managing RDS for MySQL instances that are associated with your DDM instance. You can configure read weights, synchronize data node information, and enable read/write splitting. For details, see **Data Nodes**.

Category	Function
Accounts	Creating, modifying, and deleting a DDM account, and resetting its password. For details, see **Account Management**.
Data migration	Migrating data from a third-party cloud to DDM or exporting data from DDM. For details, see **Data Migration**.
Monitoring	Providing metrics and methods of viewing metrics. For details, see **Monitoring**.
Tags	Using tags to manage resources on the management console. For details, see **Tags**.
SQL syntax	Describing DDL, DML, global sequence, SQL statements, and sharding algorithms. For details, see **SQL Syntax**.

4 Permissions Management

4.1 Creating a User and Granting Permissions

This chapter describes how to use **Identity and Access Management (IAM)** for fine-grained permissions management for your DDM resources. With IAM, you can:

- Create IAM users for employees based on the organizational structure of your enterprise. Each IAM user has their own security credentials for accessing DDM resources.
- Grant users only the permissions required to perform a given task based on their job responsibilities.
- Entrust an account or cloud service to perform professional and efficient O&M on your DDM resources.

If your account does not need individual IAM users, then you may skip over this section.

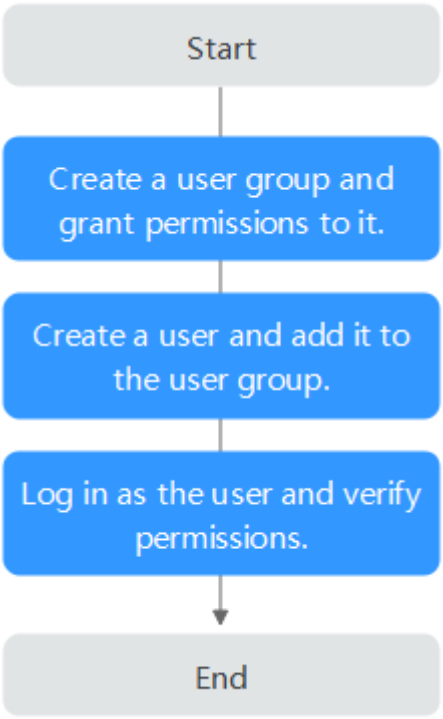
This section describes the procedure for granting user permissions. **Figure 4-1** shows the process flow.

Prerequisites

Before assigning permissions to a user group, you need to know the DDM system policies that can be added to the user group and select permissions as required. For system-defined policies for other services, see **Permissions**.

Process Flow

Figure 4-1 Process for granting DDM permissions



1. **Create a user group and assign permissions** to it.

   Create a user group on the IAM console and attach the **DDM ReadOnlyAccess** policy to the group.
2. **Create an IAM user and add it to the user group**.

   Create a user on the IAM console and add the user to the group created in **1**.
3. **Log in** and verify permissions.

   In the authorized region, perform the following operations:
   - Choose **Service List** > **Distributed Database Middleware** and click **Buy DDM Instance** to buy a DDM instance. If you cannot buy a DDM instance, the **DDM ReadOnlyAccess** permission has taken effect.
   - Choose any other service in the **Service List** (for example, there is only the **DDM ReadOnlyAccess** policy). If a message appears indicating insufficient permissions to access the service, the **DDM ReadOnlyAccess** policy has already taken effect.

4.2 Creating a Custom Policy

You can create custom policies if system-defined policies cannot meet your permission requirements.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.

- JSON: Edit JSON policies from scratch or based on an existing policy.

This section provides examples of common DDM custom policies.

Example Policies

- **Example: Denying DDM instance deletion**

  A deny policy must be used together with other policies. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions. The following is an example of a deny policy:

```
{
    "Version": "1.1",
    "Statement": [
        {
        "Effect": "Deny",
            "Action": [
                "ddm:instance:delete"
            ]
        }
    ]
}
```

  The following is an example custom policy with both Allow and Deny permissions:

```
{
   "Version": "1.1",
   "Statement": [{
                        "Effect": "Allow"
        "Action": [
            "*:*:*"
        ],
      },
      {
        "Action": [
            "ddm:instance:create",
        ],
        "Effect": "Deny"
      }
   ]
}
```

4.3 Database Accounts and Permissions

To create a schema, import schema information, or configure shards, you can use the administrator account of data nodes or create a database account with the following permissions:

SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, and TRIGGER WITH GRANT OPTION

5 Instance Management

5.1 Instance Statuses

The instance status indicates the running status of a DDM instance. You can view the status of your DDM instance on the management console.

Table 5-1 Instance statuses

Category	Status	Description
Normal	Running	The DDM instance is normal and available.
Abnormal	Creation failed	The DDM instance failed to be created.
	Abnormal	The DDM instance is unavailable.
	Some nodes abnormal	Some modes of the DDM instance are unavailable.
Actions in progress	Creating	The DDM instance is being created.
	Backing up	The instance is being backed up.
	Restoring	The instance is being restored from a backup.
	Changing the DDM service port	The service port of the DDM instance is being changed.
	Deleting	The DDM instance is being deleted.
	Restarting	The DDM instance is being restarted.

Category	Status	Description
	Adding node	Nodes are being added to an instance.
	Deleting node	Nodes are being deleted from an instance.
	Changing instance class	The vCPUs and memory of the DDM instance are being changed.
	Creating a group	A node group is being created.
	Deleting a group	A node group is being deleted.
	Creating a schema	A schema is being created.
	Deleting a schema	A schema is being deleted.

5.2 Creating a DDM instance

This section describes how to create a DDM instance on the DDM console.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the displayed page, in the upper right corner, click **Create DDM Instance**.

Step 3 On the displayed page, configure required parameters.

Table 5-2 Parameter description

Parameter	Description
Region	Region where the DDM instance is located. Select the required region.
Project	Project that the DDM instance belongs to.
Instance Name	Name of the DDM instance, which: • Cannot be left blank. • Must start with a letter. • Must be 4 to 64 characters long. • Can contain letters, digits, underscores (_), and hyphens (-).

Parameter	Description
Time Zone	You need to select a time zone for your instance based on the region hosting your instance.
Quantity	Number of nodes in a DDM instance. Up to 32 nodes are supported. **NOTE** • At least 2 nodes are recommended because using a single node cannot guarantee high availability. • Disk usage of a single node: system disk 40 GB and data disk 1 GB
AZ	Availability zone where the DDM instance is deployed. Nodes in a DDM instance can be deployed on different physical servers in the same AZ to keep services always available even if one physical server becomes faulty. A DDM instance can be deployed across AZs to provide cross-AZ DR. If necessary, you can select multiple AZs when you create a DDM instance. Then nodes of the instance will be deployed in multiple different AZs. **NOTE** Deploy your application, DDM instance, and required RDS instances in the same AZ to reduce network latency. Cross-AZ deployment may increase network latency.
Node Class	Class of the DDM instance node. You can select **General-enhanced** and then specify a node class. **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 node class so that the CPU and memory specifications of your DDM instance can better meet your needs.
Password	Select **Configure** or **Skip**. **NOTE** Configure information about an administrator account. If you select **Skip**, you can configure information about an administrator account later as needed by following **Creating an Administrator Account**.
Administrator	This parameter is mandatory when **Configure** is selected for **Password**. Enter an administrator account name, which cannot be the same as any account on the **Accounts** page. It cannot be deleted after being created.
Password	This parameter is mandatory when **Configure** is selected for **Password**. Set the password of the administrator account. **NOTE** You can reset the password later as needed by following **Resetting the Administrator Password**.
Confirm Password	This parameter is mandatory when **Configure** is selected for **Password**. Enter the password of the administrator account again.

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 for MySQL instance. • To ensure network connectivity, the DDM instance you created must be in the same VPC as your applications and RDS for MySQL instances. • After the DDM instance is created, the VPC cannot be changed. Exercise caution when performing this operation.
Subnet	A subnet provides dedicated network resources that are logically isolated from other networks for network security. A floating IP address is automatically assigned when you create a DDM instance. After the instance is created, you can change the floating IP address.
Security Group	Select an existing security group. You are advised to select the same security group for your DDM instance, application, and RDS for MySQL instances so that they can communicate with each other. If different security groups are selected, add security group rules to enable network access.
Enterprise Project	EPS provides a unified method to manage cloud resources and personnel by enterprise project.
Parameter Template	Select an existing parameter template. You can also click **View Parameter Template** to set parameters on the displayed page.

Parameter	Description
Tags	(Optional) Adding tags helps you better identify and manage your DDM resources. You can add tags to your DDM instance. Each instance can have a maximum of 20 tags. • Creating a tag You can create tags on the DDM console. A tag key and a value are required when you create a tag. **Tag key: This parameter is mandatory and cannot be null.** – Must be unique for each instance. – Can include 1 to 36 characters. – Cannot be an empty string, or start with _**sys**_, and cannot start or end with a space. – Cannot contain the following characters: Non-printable ASCII characters (0-31), "*", "<", ">", "\", ",", "|", "/" **Tag value: This parameter is mandatory.** – Is an empty string by default. – Can contain 0 to 43 characters. – Cannot contain the following characters: Non-printable ASCII characters (0-31), "*", "<", ">", "\", ",", "|" After an instance is created, you can click the instance name to view its tags, modify or delete the tags on the **Tags** tab page. In addition, you can quickly search for and filter specified instances by tag. You can add a tag to an instance after the instance is created.

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

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

Step 6 To view and manage the instance, go to the **Instances** page.

The DDM service port is **5066** by default and can be changed after a DDM instance is created.

For details, see **Changing a DDM Service Port**.

----End

5.3 Splitting Read-only and Read-Write Services

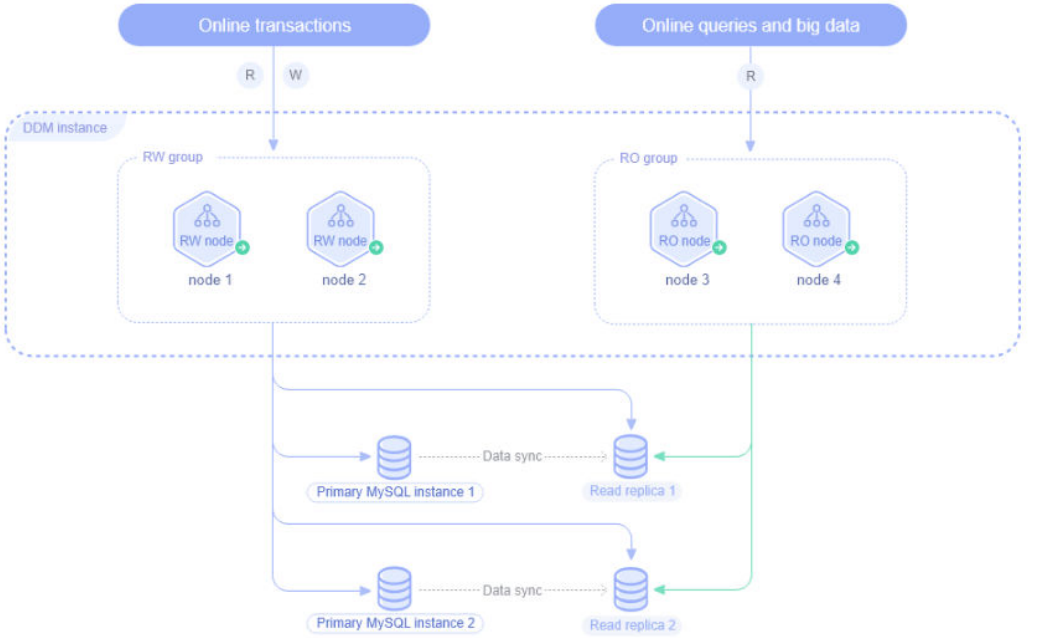
5.3.1 What Is Read-only Service Isolation?

DDM provides read-only service isolation by grouping nodes of a DDM instance to provide physically separated compute and storage resources.

DDM provides two types of node groups, read-only and read/write, which handle read-only and read/write requests, respectively. By default, read-only groups handle read requests sent to read replicas at the storage layer, relieving the read pressure of core workloads in the DDM cluster. 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 read replicas of data nodes to reduce query response time and provide faster access. It is easy to use node groups without the need of establishing complex links or synchronizing data.

How Does Read-only Service Isolation Works

Figure 5-1 Isolation diagram



5.3.2 How Are Read-only Services Split from Read-Write Services

This section describes how to isolate read-only services.

Precautions

- The kernel version must be 2.4.1.2 or later.
- If you want to use read-only groups to handle SQL queries, make sure that the associated data node is available and has available read replicas. If there are no available read replicas, the following error messages may be returned:
  - backend database connection error;
  - query has been canceled
  - execute error: No read-only node

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, click the instance where you want to create a node group and click its name.

Step 3 In the **Node Information** area, click **Create Group**.

Step 4 On the displayed page, configure required parameters.

Table 5-3 Parameter description

Parameter	Description
Group Name	The name must start with a letter and contains only letters, digits, and hyphens (-).
Role	There are two types of groups: read/write and read-only. Each group contains instance nodes with the same role and has a connection address.
VPC	The VPC is the same as the VPC where your instance is deployed, and cannot be changed. You can specify a subnet for the node group.
Node Class	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.
AZ	Select an AZ.
Quantity	One DDM instance supports multiple read-only groups. Each group contains at least 2 nodes, and each instance contains up to 32 nodes. You can click the plus icon to add nodes.

Step 5 Click **Next**.

Step 6 Confirm the configuration and click **Submit**.

Step 7 After the creation is complete, check whether the original **Node Information** area becomes the **Group Information** area. Then you can manage nodes in the group.

NOTE

- After you create a node group, you can change node class, add or remove nodes, and configure access control.
- After a node group is created for a DDM instance with no groups, existing nodes of the instance are included into another read/write group by default, responsible for handling read/write requests to core services.
- Read traffic to read-only groups is forwarded to read replicas of data nodes by default. There may be noticeable latency because read replica data is asynchronously replicated from primary instances of the data nodes. If the latency exceeds the default or preset threshold, an access error is returned.
- To delete a group of a DDM instance, locate the group that you want to delete and click **More** > **Delete Group** in the **Operation** column. The corresponding floating IP address becomes invalid once the group is deleted. This may affect your services. The default group cannot be deleted.

----End

5.4 Changing Node Class

You can change CPUs or memory of your instance node based on service requirements. This section describes how to change the CPU or memory of a DDM instance node.

Precautions

- Change node class during off-peak hours because services will be interrupted for a while during class changing.
- After a read-only group is created, the entry for changing node class will be moved to the operation column of the group.
- Once the change operation is performed, it cannot be undone. To change the class again, submit another request after the change is complete.
- Node class can be upgraded or downgraded.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the instance whose node class you want to change and click its name.

Step 3 Click **Change**. If a group is created, click **Change Node Class** in the **Operation** column in the **Group Information** area.

Step 4 On the displayed page, select the required class.

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

Step 6 Switch back to the instance list and check whether the instance status changes to **Changing class**. You can also view the change task at Task Center.

----End

5.5 Scaling Out a DDM Instance

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

Precautions

- Scaling out a DDM instance will not interrupt services.
- Scale out your DDM instance during off-peak hours.
- Make sure that the associated data nodes are normal and not undergoing other operations.
- Each DDM instance supports up to 32 nodes.
- After a read-only group is created, the entry for adding nodes will be moved to the operation column of the group.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the instance that you want to add nodes to and click its name.

Step 3 Click **Scale Out**. If a group is created, click **Scale Out** in the **Operation** column in the **Group Information** area.

Step 4 On the displayed page, select AZs and specify the number of nodes that you want to add.

You can click the plus icon to add multiple nodes. Each DDM instance supports up to 32 nodes.

Step 5 On the displayed page, click **Submit** if all configurations are correct. If you want to modify the quantity of nodes, click **Previous**.

----End

5.6 Scaling In a DDM Instance

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

Precautions

- Scaling in a DDM instance will not interrupt services.
- Scale in your DDM instance during off-peak hours.
- Make sure that the associated data nodes are normal and not undergoing other operations.
- At least one node should be left for a DDM instance.
- After a read-only group is created, the entry for removing nodes will be moved to the operation column of the group.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the instance that you want to remove nodes from and click its name.

Step 3 Click **Scale In**. If a group is created, click **More** > **Scale In** in the **Operation** column in the **Group Information** area.

Step 4 On the displayed page, view the current instance configuration and specify the number of nodes to be removed.

Step 5 Click **Next**.

Step 6 On the displayed page, click **Submit** if all configurations are correct.

----End

5.7 Restarting a DDM Instance or an Instance Node

You may need to restart an instance to perform maintenance. You can restart a DDM instance or one of its nodes on the DDM console.

Precautions

The DDM instance is not available during restart, and the restart operation cannot be rolled back.

Restarting an Instance

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance that you want to restart and choose **More** > **Restart** in the **Operation** column.

Alternatively, click the instance name to go to the **Basic Information** page. Click **Restart** in the upper right corner of the page.

Step 3 In the displayed dialog box, click **Yes**.

Step 4 Wait until the instance is restarted.

----End

Restarting a Node of a DDM Instance

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance that you want to restart and click its name.

Step 3 In the **Node Information** area, locate the node that you want to restart and click **Restart** in the **Operation** column.

After a read-only group is created for a DDM instance, click  next to the group in the **Group Information** area, then click **Restart** in the **Operation** column.

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

Step 5 Wait until the node is restarted.

----End

5.8 Deleting a DDM Instance

You can manually delete a DDM instance billed on a pay-per-use basis on the **Instances** page.

Precautions

- Instances that an operation is being performed on cannot be deleted. They can be deleted only after the operations are complete.
- Deleted DDM instances cannot be recovered. Exercise caution when performing this operation.
- If there are RDS for MySQL instances associated with the DDM instance, the system notifies you of instance information (including instance name, instance status, and database type) when you delete the DDM instance.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance that you want to delete and choose **More** > **Delete** in the **Operation** column.

NOTE

- To delete data stored on data nodes, select **Delete data on data nodes**.
- If there are RDS for MySQL instances associated with the DDM instance, the system notifies you of instance information (including instance name, instance status, and database type) when you delete the DDM instance.

Step 3 Click **Yes**.

----End

5.9 Reloading Table Data

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 stored.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the instance whose table data you want to reload and click its name.

Step 3 Choose **More** > **Reload Table Data** in the **Operation** column.

----End

5.10 Changing a Parameter Template

You can associate a parameter template with a DDM instance on the DDM console.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the DDM instance that you want to configure a parameter template for.

Step 3 Choose **More** > **Change Parameter Template** in the **Operation** column.

Step 4 Select the required parameter template and click **OK**.

----End

5.11 Modifying Parameters of a DDM Instance

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

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance whose parameters you want to configure and click its name.

Step 3 In the navigation pane on the left, choose **Parameters**.

On the displayed page, modify parameters as needed.

By default, DDM allows you to modify the instance parameters in **Instance Parameters**.

The following are parameter configuration examples.

Figure 5-2 Results displayed when **bind_table** is not used

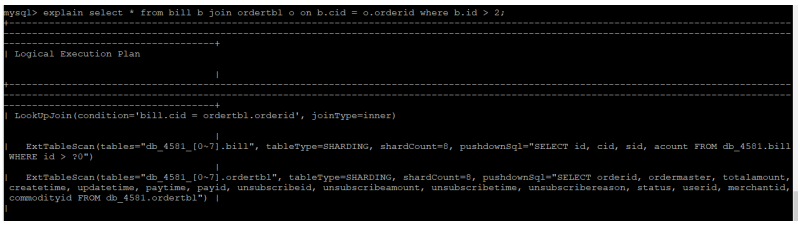
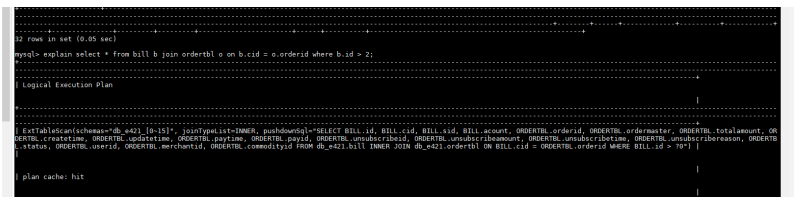


Figure 5-3 Results displayed when **bind_table** is used



Step 4 Click **Save** in the upper left corner and then **Yes** in the displayed dialog box.

NOTE

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

----End

5.12 Upgrading the Version of a DDM Instance

What Is the Preferred Version of DDM Version Series?

A DDM kernel version usually consists of four digits, for example, 3.0.8.x. The first three digits indicate the major version number, for example, 3.0.8. A series of minor versions will be released for each major version. The preferred version of this DDM series is a recommended minor version. It is also the latest and the most stable version. For DDM instances of the same major version, upgrading the kernel version to this preferred version is a minor version upgrade. This operation involves rectifying issues, and there is a low risk of syntax incompatibility. You are advised to upgrade your instance to the preferred version.

Which Is the Latest Version of DDM?

The latest version of DDM is the preferred version of the current latest DDM major version.

Scenarios

- DDM allows you to manually upgrade the kernel version to the preferred version of the current version series or the latest version.
  - Preferred version: recommended version under a major version The smaller the modification is, the lower the compatibility risk.
  - Latest version: recommended version under the latest major version A kernel version upgrade is a major version upgrade and may involve updates of new features and performance and rectification of issues. There are compatibility risks. So thorough testing is required before such an upgrade.
- By default, a newly created DDM instance uses the latest version. When a new version is released, you will see **Upgrade** in the **Version** column on the **Instances** page. You can click **Upgrade** to go to the version upgrade page.

Precautions

- Upgrading the kernel version will restart your DDM instance, and this may interrupt your workloads for a while. Perform this upgrade during off-peak hours or make sure that your applications can be automatically reconnected.
- If your instance is already of the preferred version of the current version series, it can be upgraded to only the latest version.
- If the current version differs greatly from the target version, remember to perform thorough compatibility testing on a test instance before upgrading

the version of your production instances, so that production services are not affected.

- If there are incompatibility issues after a version upgrade, roll back your instance to the original version by referring to **Rolling Back the Version of a DDM Instance**.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the instance whose version you want to upgrade and click its name.

Step 3 On the **Basic Information** page, in the **Instance Information** area, click **Upgrade** beside the **Version** field.

Alternatively, click **Upgrade** in the **Version** column on the **Instances** page.

Step 4 In the displayed dialog box, select the target version and click **OK**.

- Preferred version: recommended version under a major version
- Latest version: recommended version under the latest major version

Step 5 Confirm and click **Yes** to upgrade the version.

Step 6 When upgrading the version, the instance status changes to "Upgrading".

Step 7 After the upgrade is completed, the instance status changes from "Upgrading" to "Running". The target version is displayed.

----End

5.13 Version Change

DDM allows you to manually upgrade or downgrade the kernel version of a DDM instance.

- Version upgrade: Select a kernel version later than that of the DDM instance and perform the upgrade.
- Version downgrade: Select a kernel version earlier than that of the DDM instance and perform the downgrade.

Precautions

- Changing the kernel version will restart your DDM instance, and this may interrupt your workloads for a while. Perform this upgrade during off-peak hours or make sure that your applications can be automatically reconnected.
- If the current version differs greatly from the target version, remember to perform thorough compatibility testing on a test instance before changing the version of your production instances, so that production services are not affected.
- If there are incompatibility issues after a version upgrade, roll back your instance to the original version by referring to **Rolling Back the Version of a DDM Instance**.

- Version rollback is not supported after a downgrade. If there are compatibility issues after a version change, you can upgrade the instance to the original version by referring to **Upgrading the Version of a DDM Instance**.
- The version can be downgraded to 3.0.4.3 or later.
- Downgrading the instance to a version earlier than 3.0.9 will delete existing administrator accounts. Make sure the downgrade has no impact on your services.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the target instance and click its name.

Step 3 On the **Basic Information** page, in the **Instance Information** area, click **Change** beside the **Version** field.

Step 4 In the displayed dialog box, select **Target Version**. The system will perform an upgrade or a downgrade based on the target version you selected.

Step 5 Click **Change Now**. In the displayed dialog box, click **Yes**.

Step 6 When the version is being changed, the instance status becomes **Upgrading** or **Downgrading**. After the change is complete, the instance status changes to **Running**, and the instance version becomes the target version.

----End

5.14 Rolling Back the Version of a DDM Instance

Scenarios

After a DDM instance is upgraded to a new version, the kernel version can be rolled back to the version before the last update.

Precautions

- Rolling back to the kernel version will restart your DDM instance, and this may interrupt your workloads for a while. Perform this upgrade during off-peak hours or make sure that your applications can be automatically reconnected.
- The version can only be rolled back to the version before the last upgrade.
- If nodes are added after version updates, you need to scale in the new nodes and then roll back the version.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the instance whose version you want to upgrade and click its name.

Step 3 On the **Basic Information** page, in the **Instance Information** area, click **Roll Back Version** beside the **Version** field.

Step 4 In the displayed dialog box, click **OK**.

Step 5 Confirm and click **Yes**.

Step 6 When the rollback is under progress, the instance status changes to **Rolling back**.

Step 7 After the rollback is complete, the instance status changes from **Rolling back** to **Running**, and the instance version becomes the target version.

----End

6 Connection Management

6.1 Configuring Access Control

Scenarios

DDM supports load balancing by default, but some regions may not support. If an application accesses DDM using a private IP address, there are no traffic restrictions. To control access, you need to configure access control for your DDM instance. The security group is still valid for access requests directly sent to DDM nodes.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the instance that you want to configure access control for and click its name. The **Basic Information** page is displayed.

Step 3 Toggle on **Access Control**.

- If the instance has only one group, in the **Network Information** area, click  on the right of button **Access Control**.
- If the instance has multiple groups, the access control button is moved to the group information area. On the **Basic Information** page, in the group list, click  in the **Access Control** column.

Step 4 Click **Configure**. In the **Configure Access Control** dialog box, specify **Access Policy**, enter the required IP addresses, and click **OK**.

----End

6.2 Binding and Unbinding an EIP

After a DDM instance is created, it is not accessible over public networks by default (because it has no EIPs bound). You can bind an EIP to a DDM instance or for public access and then unbind the EIP when public access is no longer required.

Precautions

- If a DDM instance has already been bound with an EIP, you must unbind the EIP from the instance first before binding a new EIP to it.
- If a DDM instance contains both read-only and read/write groups, you can bind an EIP to a node of the read/write groups.

Prerequisites

You have applied for an EIP in the VPC.

Binding an EIP to an Instance

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, select the instance that you want to bind an EIP for and click its name.

Step 3 In the **Instance Information** area, click **Bind** next to **EIP**.

Step 4 In the displayed dialog box, all EIPs in the unbound status are listed. Select the required EIP and click **OK**.

If no available EIPs are displayed, click **View EIP** and obtain an EIP.

Step 5 On the **Basic Information** page, view the EIP that has been bound to the instance.

----End

Unbinding an EIP from an Instance

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, select the instance that you want to bind an EIP for and click its name.

Step 3 In the **Instance Information** area, click **Unbind**.

Step 4 In the displayed dialog box, click **Yes** to unbind the EIP.

Step 5 On the **Basic Information** page, check whether the EIP is unbound.

----End

6.3 Changing a DDM Service Port

DDM allows you to change a service port of your DDM instance and will restart the instance after the change.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, select the DDM instance whose service port you want to change and click its name.

Step 3 On the **Basic Information** page, in the **Connection Information** area, click next to **DDM Service Port**.

For DDM instances, the service port number ranges from 1025 to 65534 except for ports 1033, 7009, 8888, and 12017 because they are in use by DDM. The default value is **5066**.

Step 4 Click to apply the changes.

- In the dialog box, click **Yes**. Changing the service port of a DDM instance requires a restart of the instance.
- In the dialog box, click **No**.

Step 5 View the results on the **Basic Information** page.

----End

6.4 Changing the Security Group of a DDM Instance

DDM allows you to change the security group of a DDM instance.

Precautions

Changing the security group of a DDM instance may disconnect the instance from its associated data nodes.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, select the DDM instance whose security group you want to change and click its name.

Step 3 In the **Network Information** area on the **Basic Information** page, click beside field **Security Group**.

Step 4 Click to apply the changes.

Step 5 View the results on the **Basic Information** page.

----End

7 Schema Management

7.1 Creating a Schema

This section describes how to create a schema on the DDM console.

Precautions

- When you create a schema, select only data nodes that are in the same VPC as the DDM instance and are not used by other DDM instances. DDM will create databases on the selected data nodes without affecting their existing databases and tables.
- All instances associated with one schema must be of the same major MySQL version.
- Specifications of associated data nodes should be greater than that of the DDM instance. Otherwise, the performance will be affected.
- Multiple schemas can be created in a DDM instance. Multiple schemas can be associated with the same data node.
- One data node cannot be associated with schemas in different DDM instances.
- If you create a sharded schema, more than one shard will be generated in the schema. Shard names will follow the rule: *<schema name>_<number>*. *<number>* here indicates a four-digit number starting from 0000. This number will be incremented by one. For example, if a schema name is **db_cbb5** and there are 2 shards, the shard names are **db_cbb5_0000** and **db_cbb5_0001**.
- Read-only instances cannot be associated with the schema as data nodes.
- Do not modify or delete the internal accounts (DDMRW*, DDMR*, and DDMREP*) created on data nodes. Otherwise, services will be affected.

  NOTE

  - The internal account name is in the format: Fixed prefix (such as DDMRW, DDMR, or DDMREP) + Hash value of the data node ID.
  - A random password is generated, which can contain 16 to 32 characters.

Prerequisites

- You have **created a DDM instance** and the instance is in the **Available** state.
- You have created an account. For details, see **Creating an Account**.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required DDM instance and click **Create Schema** in the **Operation** column.

Alternatively, click the instance name to go to the **Basic Information** page. On the displayed page, choose **Schemas** in the navigation pane and click **Create Schema** in the upper left corner of the page.

Step 3 On the displayed page, configure required parameters.

Table 7-1 Parameter description

Parameter	Description
Sharding	• **Sharded**: indicates that one schema can be associated with multiple data nodes, and all shards will be evenly distributed across the nodes. • **Unsharded**: indicates that one schema can be associated with only one data node, and only one shard can be created on the data node.
Schema	The name can contain 2 to 48 characters and must start with a lowercase letter. Only lowercase letters, digits, and underscores (_) are allowed.
Account	The DDM account that needs to be associated with the schema.
Data Nodes	Select only data nodes that are in the same VPC as the DDM instance and are not used by other DDM instances. Databases can be created on the data nodes you select, without impacting existing databases and tables.
Shards	The total shards are the shards on all data nodes. There cannot be more data nodes than there are shards in the schema. Each data node must have at least one shard assigned. Recommended shards per data node can be 8 to 64.

Step 4 Click **Next**.

Step 5 On the displayed page, enter a database account with the required permissions and click **Test Availability**.

NOTE

Required permissions: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, and TRIGGER WITH GRANT OPTION

You can create a database account for the RDS for MySQL instance and assign it the above permissions in advance.

Step 6 After the test becomes successful, click **Finish**.

After the schema is created, the schema status is as follows:

- Creating
- Creation failed
- Running

----End

7.2 Exporting Schema Information

When you deploy DR or migrate data across regions, you can export schema information from source DDM instances. The export information includes schema information and shard information, excluding service data and index data.

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.

Step 2 In the instance list, locate the instance that you want to export schema information from and click its name.

Step 3 On the displayed page, in the navigation pane, choose **Schemas**.

Step 4 Click **Export Schema Information**. All schema information of the current DDM instance is exported to a JSON file.

----End

7.3 Importing Schema Information

When you deploy DR or migrate data across regions, you can import schema information into destination DDM instances. The imported information includes schema information and shard information, excluding service data and index data.

Prerequisites

The DDM instance has no schemas.

The kernel version of the destination instance must be 3.1.1 or later.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate a target instance and click its name.

Step 3 On the displayed page, in the navigation pane, choose **Schemas**.

Step 4 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 5 On the displayed page, click **Upload** to select the JSON file exported in **Exporting Schema Information**, choose the required data nodes, enter a database account with the required permissions, and click **Finish**.

NOTE

- The number of selected data nodes is the same as the number of data nodes imported into the DDM instance.
- Required permissions: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER WITH GRANT OPTION

----End

7.4 Deleting a Schema

You can delete schemas that are no longer needed.

Precautions

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

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance whose schemas you want to delete and click its name.

Step 3 On the displayed page, in the navigation pane, choose **Schemas**.

Step 4 In the schema list, locate the schema that you want to delete and click **Delete** in the **Operation** column.

Step 5 In the displayed dialog box, click **Yes**.

NOTE

- Your schema will become faulty if you delete its associated data nodes by clicking the **Delete** button in the schema list.
- To delete data stored on the associated data nodes, select **Delete data on data nodes** in the displayed dialog box.
- If you want to delete a schema, check whether there are data nodes associated with this schema. If the associated instances have been deleted, click **Synchronize Data Node Information** and delete the schema.
- If the associated data nodes 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 Data Node Information** and delete the schema.

----End

7.5 Configuring a SQL Blacklist

To prevent some SQL statements being executed, you can configure a blacklist and add those statements to it.

This function aims to keep DDM instances running stably even if there is a sudden increase in concurrent SQL statements.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate a target instance and click its name.

Step 3 On the displayed page, in the navigation pane on the left, choose **Schemas**.

Step 4 In the schema list, locate the schema that you want to configure a blacklist for and click **More** > **Configure SQL Blacklist** in the **Operation** column.

Step 5 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 or DELETE and are not allowed by the current schema.
- **Full-text Match**: Enter full-text SQL statements that are not allowed by the current schema. Multiple spaces and line breaks will not be treated as if they were replaced or truncated as a single space.
- **Regular Expression Match**: Enter regular expressions that are not allowed by the current schema.
- Separate SQL statements in the blacklist with semicolons (;). The size of SQL statements for prefix match, full-text match, and regular expression match cannot exceed 1 KB, respectively.
- 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 Viewing Schemas

You can view schemas of a DDM instance on the DDM console without using any code.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the required instance and click its name.

Step 3 On the displayed page, in the navigation pane on the left, choose **Schemas**.

Step 4 On the displayed page, click the name of the target schema or click **Manage** in the **Operation** column.

- On the displayed page, view information about logical tables of the current schema.

  You can view the table name, type, database sharding key, database sharding algorithm, number of shards, table sharding key, table sharding algorithm, number of table shards, auto-increment field, start value for auto-increment field, auto-increment step, status, primary key, table fragmentation rate, disk size, creation time, update time, and global secondary index of a logical table.

  Step for Auto-Increment indicates the auto-increment cache value of a global sequence, instead of the number generated for the auto-increment field each time a new record is inserted.

  You can click **View Table Structure** in the **Operation** column to view the SQL statement used for creating this table.

- On the **Associated DB Instances** page, view instance information of the current schema.
- On the **Connection Address** tab page, obtain CLI and JDBC connection addresses of the DDM instance.

----End

8 Shard Configuration

8.1 Overview and Application Scenarios

Shard configuration is a core function of DDM. With this function, you can increase data nodes or shards to improve database storage and concurrency as services grow. Shard configuration has little impacts on your services, so you do not need to worry about database scaling and subsequent O&M as your services burst.

Application Scenarios

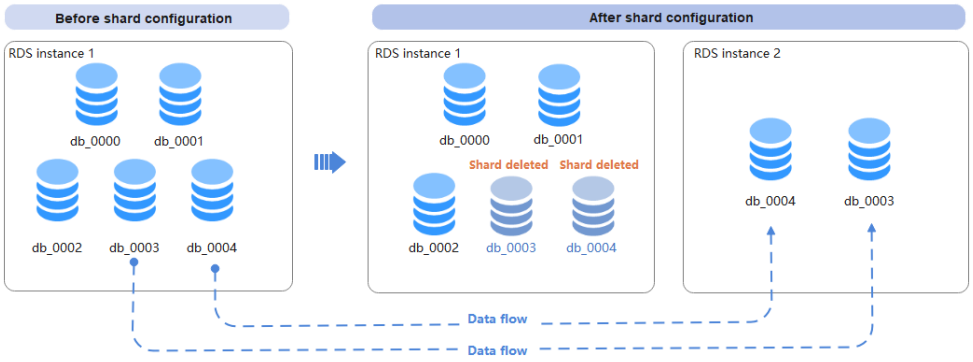
DDM provides the following methods of configuring shards to meet different service needs.

Method 1: Keep shards unchanged and increase data nodes

This method does not change the number of shards and only increases data nodes. Some shards are migrated from original data nodes to new data nodes, but shard data is not redistributed. So this method is the fastest one among all three methods and is recommended.

This method underpins rapid service growth after horizontal sharding and can reduce costs at the early stage of services. It is also suitable if RDS for MySQL instances cannot meet storage space and read/write performance requirements.

Figure 8-1 Adding RDS for MySQL instances with shards unchanged

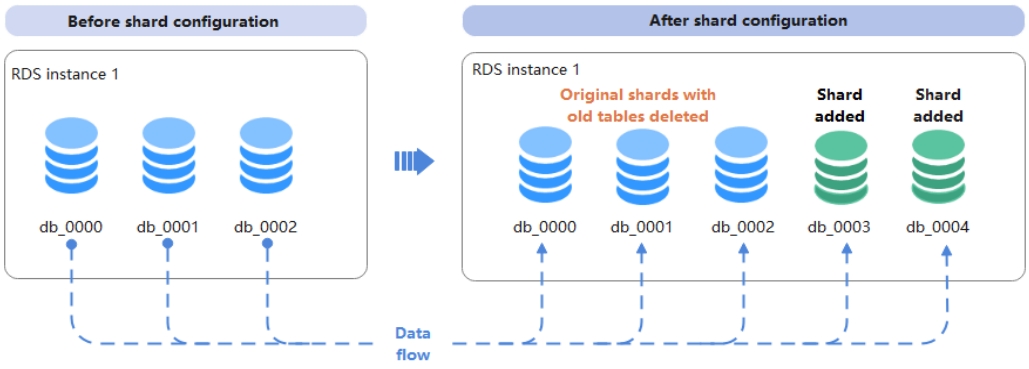


Method 2: Add shards with data nodes unchanged

This method adds shards, but not data nodes. It changes total shards, total table shards, and table sharding rules. Data is redistributed to all shards. Old tables in original shards will be deleted, and broadcast tables are increased.

This method is suitable if associated RDS for MySQL instances have sufficient storage space but one of its tables contains a large amount of data, with query performance limited.

Figure 8-2 Adding shards with RDS for MySQL instances unchanged

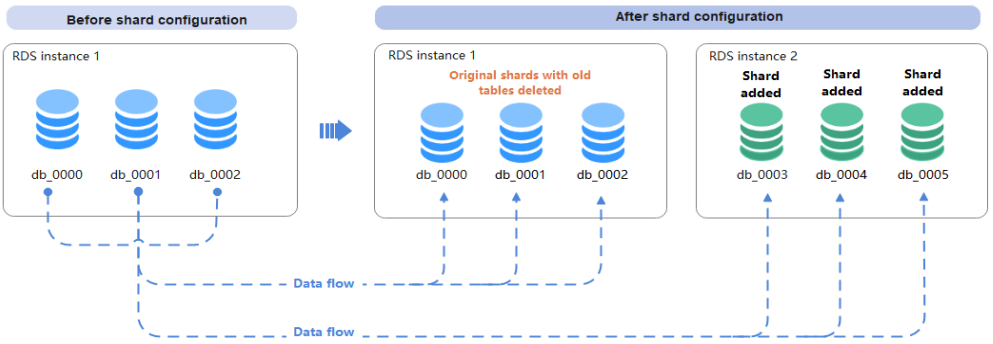


Method 3: Add both shards and data nodes

This method increases both shards and data nodes. It changes total shards, total table shards, and table sharding rules. Data is redistributed to all shards. Old tables in original shards will be deleted, and broadcast tables are increased.

This method is suitable if associated RDS for MySQL instances cannot meet storage space and read/write requirements and there is a physical table containing a large amount of data with query performance limited.

Figure 8-3 Adding shards and RDS for MySQL instances



8.2 Assessment

Before changing shards, you need to carry out a preliminary evaluation and determine the number of new shards, whether to scale up the current DDM node class, and the number of required data nodes and their specifications.

- Data volume: Run **show db status** to query the volume of data involved.
- DDM node class: Determine nodes of the DDM instance and vCPUs and memory size of each node.
- Data node class: Determine the number of data nodes and vCPUs and memory size of each node.
- Business scale:
  - Analyze current service scale and growth trend. Changing shards is a critical operation for data change. You are advised to postpone this operation if the storage space of a data node is sufficient.
  - Do not update the shard key (including global secondary indexes) during shard configuration to ensure data consistency.
  - You are advised to perform shard configuration during off-peak hours.
- Whether to add shards: Adding shards leads to sharding rule changes. All data in the current schema needs to be rebalanced based on the new sharding rule. This process is slow and requires more resources.
- Whether to perform DDL operations during shard change: Read and write services are not affected. To ensure data consistency, do not perform DDL operations during shard change.

8.3 Pre-check

Check items in the table below one day before performing a shard configuration task.

Pre-check Items

Table 8-1 Pre-check items involved

Item	Purpose	Solution to Check Failure
Table name length	Redistributing data (for example, adding shards) will generate a temporary table whose name is longer than the original table. Ensure that the name of a temporary table does not exceed the maximum length defined by MySQL.	Modify the table name that is too long.
Binlog full backup time of the data node	Whether your full backups are retained for a time period long enough	Log in to the data node console and ensure that the full backup retention period is more than 30 days.

Item	Purpose	Solution to Check Failure
Binlog enabled on data nodes	Whether binlog is enabled to support online shard configuration	If your data node is an RDS instance, ensure that Binlog is enabled.
Retention period of binlogs on data nodes	The retention period of binlogs on data nodes must be long enough.	If your data node is an RDS instance, no further action is required.
Broadcast table consistency	Ensure broadcast table consistency before performing a shard configuration task.	In the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket** to contact DDM O&M personnel.
Character set and collation of source shards	Ensure that character set and collation are consistent before and after the shard configuration.	In the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket** to contact DDM O&M personnel.
SQL statements for creating physical stables.	Ensure that table structure on physical shards is consistent.	Execute CHECK TABLE to check for table structure inconsistencies and execute ALTER to rectify the inconsistencies.
Primary keys	All tables in the source database have primary keys, and the sharding key is a part of the primary keys to ensure data consistency after shards are changed.	Add primary keys for tables using ALTER if the tables have no primary keys.
DB instance connectivity	Check whether data nodes can be connected.	Check security group configurations.
DB instance parameters	The source data nodes have the same DB parameter settings as the destination data nodes.	Modify parameter configurations on the data node console.

Item	Purpose	Solution to Check Failure
DB instance storage space	The disk space of data nodes is sufficient during shard configuration.	Scale up storage space of data nodes. **CAUTION** This check item is based on the estimated value that may be different from the actual value.
DB instance time zone	The source data nodes have the same time zone requirements as the destination data nodes.	Modify the time zone on the data node console.
Maximum number of physical tables	When you add shards, each data record in the source table will be rerouted to a new physical table. It takes a long time for sharding if there are too many physical tables. Check whether the number of physical tables on each node exceeds the upper limit.	In the upper right corner of the console, choose **Service Tickets** > **Create Service Ticket** and contact the customer service.

Common Issues and Solutions

- The shard configuration fails due to table structure inconsistency.

  Solution: Execute CHECK TABLE to query table structure inconsistencies and execute ALTER TABLE to rectify the inconsistencies. In the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket** to contact O&M personnel if the inconsistencies cannot be rectified using DDL, for example, the primary or unique keys cannot be modified for data reasons.

- Tables without primary keys cannot be migrated. If a table has no primary keys, it cannot be correctly located and recorded. After a retry is performed during shard configuration, duplicate data may be generated.

  Solution: Add a primary key to the table.

- If the sharding key is not part of a primary key, there may be data records (in different physical tables) with duplicate primary key values in a logical table. When these data records are redistributed, they will be routed to the same physical table, and only one record is retained because they have the same primary keys. As a result, data becomes inconsistent before and after the migration, causing the shard configuration failure.

NOTE

This error does not occur when the primary key is a globally unique sequence and the number of shards does not change.

Solution: Rectify the data and check again.

8.4 Operation Guide

This section uses an RDS for MySQL instance as an example to describe how to configure shards for a schema.

Prerequisites

- There is a DDM instance with available schemas.
- There is an RDS for MySQL instance in the same VPC as the DDM instance, and is not associated with any other DDM instances. If adding data nodes is required, ensure that the new data nodes are in the same VPC as the DDM instance.
- The kernel version of the DDM instance must be 3.0.8.3 or later. The latest kernel version is recommended.
- Ensure that the instances to be associated with your schema cannot be in read-only states.
- Unsharded schemas do not support shard configuration.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the required instance and click its name.

Step 3 On the displayed page, choose **Schemas** to view schemas of the DDM instance.

Step 4 In the schema list, locate the schema that you want to configure shards for and click **Configure Shards** in the **Operation** column.

Step 5 On the **Configure Shards** page, configure the required parameters and click **Test Availability**.

Table 8-2 Parameter description

Parameter	Description
Current DB Instances and Shards	Displays the information of current DB instances and shards.
Total Shards After Configuration	Defaults to the total number of existing shards in the schema. If you want to increase shards, change the default value to the new total number of shards, and DDM will distribute all shards evenly to all data nodes.

Parameter	Description
Execution Node Type	The default value is **Shared**. You can change the execution node type based on service requirements. • **Shared**: The node is responsible not only for configuring shards, but also for handling service requests. So, shard configuration is slower. • **Dedicated**: The execution node will be dedicated to configuring shards, so shard configuration is faster. If you are worried about not having enough nodes to handle your services, add more nodes before configuring shards. If you select **Dedicated** for the execution node type, enable load balancing for the default group and ensure that the number of nodes exceeds one.
Route Switchover	Both manual and automatic switchover are supported. Read and write requests involved will be switched to new DB instances. During the switchover, one or two intermittent disconnections may occur but do not affect your services. You are advised to perform the switchover during peak-off hours.
DB Instance	An existing instance is selected by default. You can determine whether to add DB instances based on service requirements. You need to input the required password for testing connectivity. Only after the connectivity test is successful, you can go to the next step.

NOTE

- Tables without primary keys do not support shard configuration.
- The number of physical shards per data node in the schema cannot exceed 64. If more than 64 shards are required, in the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket** to contact DDM technical support personnel.
- Required permissions: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER WITH GRANT OPTION

Step 6 After the test is successful, click **Next** to go to the **Precheck** page.

NOTE

- Precheck is not the start of shard configuration. The configuration task does not start until you click **OK**.
- Handle risks first if any. You can also ignore the risks if you ensure that they do not affect your services.

Step 7 After all check items are complete, click **Configure shards**.

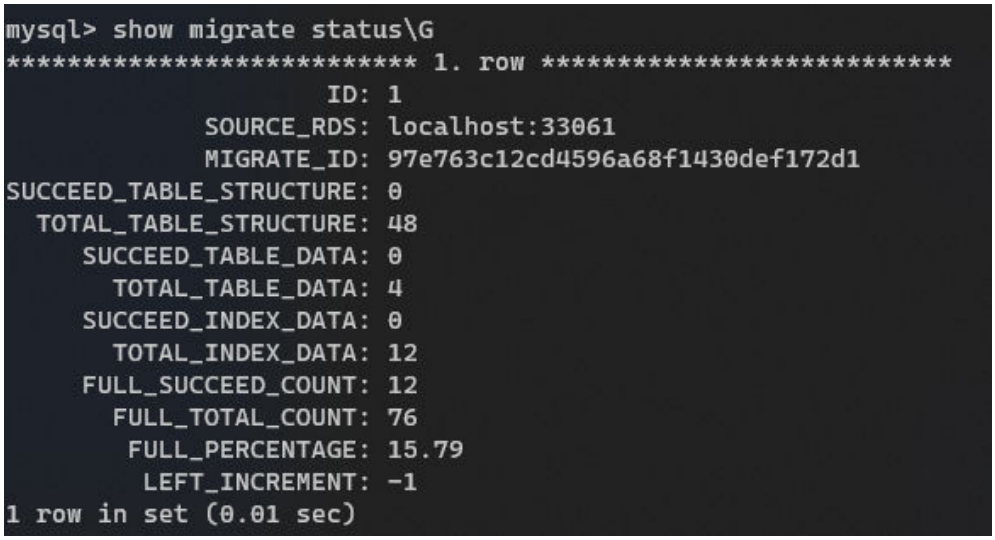
- The shard configuration task is in progress. A data migration task is performed in two phases, full migration and increment migration, respectively.

  You can view the migration progress on the **Task Center** page.

  You can click ⌄ next to the task to view task details, including shards and data nodes before and after the shard configuration, and switchover policy.

  You can run **show migrate status** on a SQL client to view progress of the shard configuration task.

  Figure 8-4 Running the required command to view task progress



NOTE

The number of returned records corresponds to the number of source RDS instances.

SOURCE_RDS: indicates the source RDS instance.

MIGRATE_ID: indicates the ID of the task of configuring shards.

SUCCEED_TABLE_STRUCTURE: indicates the number of physical tables whose structure data has been migrated.

TOTAL_TABLE_STRUCTURE: indicates the total number of physical tables whose structure data is to be migrated.

SUCCEED_TABLE_DATA: indicates the number of physical tables whose data records have been migrated.

TOTAL_TABLE_DATA: indicates the number of physical tables whose data records are to be migrated.

SUCCEED_INDEX_DATA: indicates the number of physical tables whose indexes have been migrated.

TOTAL_INDEX_DATA: indicates the number of physical tables whose data records are to be migrated.

FULL_SUCCEED_COUNT: indicates the objects that have finished a full migration in the current scale-out subtask.

FULL_TOTAL_COUNT: indicates all objects that need to be migrated by a full migration in the current scale-out subtask.

FULL_PERCENTAGE: indicates the percentage of migrated objects in the full migration in the current scale-out subtask.

Aggregate total objects to be migrated in a full migration and migrated objects in each scale-out subtask. The total objects to be migrated and migrated in all subtasks are displayed in the progress bar at Task Center.

- On the **Task Center** page, choose **More** > **Modify Switch Policy** in the **Operation** column.
- On the displayed page, click **More** > **View Log** in the **Operation** column to view task logs.
- If you select **Manual** for route switchover, click **Switch Route** at Task Center after data is completely migrated. If you select **Automatic**, the route is automatically switched over within the specified time.

  NOTE

  - Switching route is critical for a shard configuration task. Before the route is switched, you can cancel a shard configuration task, and data in original databases is not affected.
  - If a new RDS for MySQL instances is added, write operations will be disabled during route switchover. If the number of shards is increased, read and write operations are both disabled during route switchover.
  - Switching route during off-peak hours is recommended. This is because data validation is required during this process, increasing the switchover time. How long route switchover requires depends on the volume of the data involved.

Step 8 Click **Clear** in the **Operation** column to delete the data migrated from original RDS for MySQL instances.

Step 9 Carefully read information in the dialog box, confirm that the task is correct, and click **Yes**.

Step 10 Wait till the source data is cleared.

> **NOTE**
>
> Old data can be deleted by the DROP statement. After the deletion, storage space may not be released immediately due to MySQL constraints.

Step 11 Run the following commands after the shard configuration is complete:

show data node: used to view the relationship between new data nodes and shards.

show db status: used to view the estimated usage of schema disks.

----End

9 Data Nodes

9.1 Overview

Managing data nodes is managing RDS for MySQL instances that are associated with your DDM instance. You can view the RDS for MySQL instance status, storage, node class, and read weight, configure read weights, synchronize data node information, and enable read/write splitting.

Managing data nodes is managing RDS for MySQL instances that are associated with your DDM instance. You can view the RDS for MySQL instance status, storage, node class, and read weight, configure read weights, synchronize data node information, and enable read/write splitting.

Table 9-1 Functions

Function	Description
Synchronizing data node information	This function is used to synchronize data node changes to your DDM instance after you make changes such as adding or deleting a read replica, changing the connection address, port number, or specifications, or deleting a data node.
Enabling or disabling read/write splitting	This function is used for DDM kernel version 3.1.0 or later. If the DDM kernel version is 3.1.0 or later, you need to manually enable read/write splitting and then adjust read/write weights of your primary instance and read replicas. You can also manually disable read/write splitting as needed. If the DDM kernel version is earlier than 3.1.0, read/write splitting is enabled by default and cannot be disabled. You just need to adjust read/write weights of your primary instance and read replicas.

Function	Description
Configuring read weights	This function is used when you need to adjust read/write weights of your primary instance and read replicas to split read and write requests. • You can configure read/write weights of multiple instances in batches. • If a data node (associated DB instance) has no read replicas, you cannot configure a read/write weight for it.

NOTE

- If a data node is an RDS for MySQL instance, the SQL_MODE parameter **PAD_CHAR_TO_FULL_LENGTH** is not supported.

9.2 Synchronizing Data Node Information

Synchronize data node changes to your DDM instance after you make changes such as adding or deleting a read replica, changing the connection address, port number, or specifications, or deleting a data node.

After you change your data nodes, you need to click **Synchronize Data Node Information** to synchronize the change to your DDM instance.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the DDM instance that you want to synchronize data node information to.

Step 3 Click the instance name and go to the **Basic Information** page.

Step 4 Choose **Data Nodes** in the left navigation pane and click **Synchronize Data Node Information**.

Step 5 The system automatically synchronizes data node information. This operation takes 1 to 3 minutes to complete.

----End

9.3 Enabling Read/Write Splitting

DDM optimizes its read/write splitting function. In earlier versions, read/write splitting is automatically enabled after read replicas are added. From this version on, read/write splitting needs to be manually enabled after read replicas are added, and you can set read weights for your primary instance and read replicas.

You can enable or disable read/write splitting based as needed.

Precautions

The DDM kernel version must be 3.1.0 or later.

Enabling Read/Write Splitting

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the DDM instance that you want to enable read/write splitting for.

Step 3 Click the instance name and go to the **Basic Information** page.

Step 4 Choose **Data Nodes** in the left navigation pane and click **Enable Read/Write Splitting**.

Step 5 In the displayed dialog box, click **OK**.

NOTE

- For an RDS for MySQL instance, enabling read/write splitting will set the read weight of the primary instance to 100. You can adjust read weights of the primary instance and its read replicas based on your service requirements.
- There may be a noticeable latency in the data because read replica data is asynchronously replicated from the primary instance.
- After read/write splitting is enabled, read queries will be distributed to the primary or a read replica based on the configured **read weight**. There may be a replication delay for queries distributed to the read replica. For query statements that are not in the same transaction but logically depend on the data written in the previous transaction, you can add **/*+ db_type=master*/** to the query statement. This hint forcibly enables the primary node to process the query, ensuring that the latest data written by the previous transaction can be queried.

----End

Disabling Read/Write Splitting

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the DDM instance that you want to disable read/write splitting for.

Step 3 Click the instance name and go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Data Nodes**.

Step 5 Click **Disable Read/Write Splitting**.

NOTE

After read/write splitting is disabled, the preset weights will not work any more. Read/Write requests are processed only on the primary instance, so workloads using the primary instance may be affected.

Step 6 In the displayed dialog box, click **OK**.

----End

9.4 Configuring Read Weights

This section describes how to adjust read/write weights of a primary instance and its read replicas. If one DDM instance is associated with multiple data nodes, you can configure read weights of the data nodes in batches.

Precautions

If a data node (associated DB instance) has no read replicas, you cannot configure a read/write weight for it.

Batch Configuring Read Weights for Multiple DB Instances

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the DDM instance whose associated DB instances you want to configure read weights for.

Step 3 Click the instance name and go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Data Nodes** and click **Configure Read Weight**.

Step 5 Configure read weights for DB instances associated with the DDM instance

In the displayed dialog box, you can click **Synchronize** to apply the read weight of the first DB instance to other instances. This operation requires that each of the DB instances has the same number of read replicas.

Otherwise, you need to manually configure the read weight for each DB instance.

- The read weight can be 0 to 100.
- If the kernel version is 3.1.0, read replicas of one of the DB instances handle all separated read requests by default. To re-assign read/write requests, you can configure read weights of the instance and its read replicas.
- If the kernel version is 3.1.0 or later, the primary instance that read replicas belong to handle all read requests by default. To re-assign read/write requests, enable read/write splitting and configure read weights.
- After the read weights are configured, the primary instance and its read replica will handle read requests according to the following formulas:
  - Primary instance: Read weight of primary instance/Total read weights of primary instance and read replica
  - Read replica: Read weight of read replica or total read weights of primary instance and read replica

  For example, if an RDS for MySQL instance contains one primary instance and one read replica and read weights of the primary instance and its read replica are 20 and 80 respectively, they will process read requests in the ratio of 1:4. In other words, the primary instance processes 1/4 of read requests and the read replica processes 3/4. Write requests are automatically routed to the primary instance.

Step 6 After the read weight is configured, you can view the updated read weight in the data node list.

----End

Configuring the Read Weight for a Single DB Instance

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the DDM instance whose associated DB instance you want to configure a read weight for.

Step 3 Click the instance name and go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Data Nodes**.

Step 5 Locate the required instance and click **Configure Read Weight** in the **Operation** column.

- The read weight can be 0 to 100.
- If the kernel version is 3.1.0, read replicas of one of the DB instances handle all separated read requests by default. To re-assign read/write requests, you can configure read weights of the instance and its read replicas.
- If the kernel version is 3.1.0 or later, the primary instance that read replicas belong to handle all read requests by default. To re-assign read/write requests, enable read/write splitting and configure read weights.
- After the read weights are configured, the primary instance and its read replica will handle read requests according to the following formulas:
  - Primary instance: Read weight of primary instance/Total read weights of primary instance and read replica
  - Read replica: Read weight of read replica or total read weights of primary instance and read replica

  For example, if an RDS for MySQL instance contains one primary instance and one read replica and read weights of the primary instance and its read replica are 20 and 80 respectively, they will process read requests in the ratio of 1:4. In other words, the primary instance processes 1/4 of read requests and the read replica processes 3/4. Write requests are automatically routed to the primary instance.

Step 6 After the read weight is configured, you can view the updated read weight in the data node list.

----End

9.5 Configuring Read/Write Splitting

Read/Write splitting offloads read requests from associated primary DB instances to its read replicas at a ratio, improving processing of read/write transactions. This function is transparent to applications, and you do not need to modify service code. Configure read weights of primary instances and their read replicas on the DDM console, and read traffic will be distributed at the preset ratio and write traffic will be forwarded to the primary instances by default. The ratio is generally based on service requirements and loads of associated data nodes.

Data is asynchronously replicated from the primary instance to read replicas, and there is a delay between them in milliseconds. Set weights of the primary instance and its read replicas to 0 and 100, respectively, that is, distribute all read requests to read replicas if sub-second latency is allowed for read requests and these requests require high query costs that may impact read/write transactions. In other scenarios, adjust the ratio based on service requirements.

Precautions

- If the DDM kernel version is 3.1.0 or later, you need to manually enable read/write splitting and then adjust read weights of your primary instance and read replicas.
- If the DDM kernel version is earlier than 3.1.0, read/write splitting is enabled by default. You just need to adjust read weights of your primary instance and read replicas.
- The SELECT statements that contain hints or modify data in transactions are all executed by the primary instance.
- If the primary instance becomes faulty and parameter **Seconds_Behind_Master** on its read replicas is set to **NULL**, read-only requests are still forwarded to the primary instance. Recover the faulty instance as soon as possible.

Prerequisites

- You have created a DDM instance and a data node with read replicas.
- You have created a schema.

Procedure

Step 1 Log in to the DDM console.

Step 2 Locate the DDM instance that you want to configure read/write splitting for and click its name.

Step 3 In the navigation pane on the left, choose **Data Nodes**.

Step 4 Enabling read/write splitting affects only the nodes in read/write groups.

- If the DDM kernel version is 3.1.0 or later, you need to manually enable read/write splitting by referring to **Enabling Read/Write Splitting**. Then, go to **Step 5**.
- If the kernel version is earlier than 3.1.0, read/write splitting is enabled by default. Skip this step and go to **Step 5**.

NOTE

You can create a read-only group and use the private IP address of the read-only group to process read requests without enabling read/write splitting. For details, see **Creating a Read-Only Group**.

Step 5 Configure read weights.

For details, see **Configuring Read Weights**.

----End

10 Parameter Template Management

10.1 Instance Parameters

By default, DDM allows you to modify only the following parameters. If you need to modify other parameters in some special scenarios, such as data migration, contact technical support.

Table 10-1 Parameter description

Parameter	Description	Value Range	Default Value
bind_table	Data association among multiple sharded tables. The optimizer processes JOIN operations at the MySQL layer based on these associations. For details about parameter examples, see the description below the table.	• The format should be: [{<table1>.<column1>,<table2>.<column2>}, {<table1>.<column2>,<table3>.<column1>},...] The value is in the format of *<table1>.<column1>,<table2>.<column2>* and can contain multiple objects. • The version should be: DDM 2.3.2.7 or later.	-

Parameter	Description	Value Range	Default Value
character_set_server	DDM server's character set. To store emoticons, set both this parameter and the character set on RDS to **utf8mb4**. For a DDM instance 3.0.9 or later, you can execute **show variables like '%char%'** to query its character set. You will find that **character_set_client**, **character_set_results**, and **character_set_connection** in the command output all have a fixed value, **utf8mb4**. **CAUTION** To change the value of this parameter, you must also change the value of **collation_server**.	gbk, utf8, utf8mb4	utf8mb4
collation_server	Collation on the DDM server. This parameter determines the collation of a schema. **CAUTION** To change the value of this parameter, you must also change the value of **character_set_server**.	utf8mb4_unicode_ci, utf8mb4_bin, utf8mb4_general_ci	utf8mb4_unicode_ci
concurrent_execution_level	Concurrency level of scanning table shards in a logical table: • **DATA_NODE**: indicates that database shards are scanned in parallel and table shards in each database shard are scanned in serial. • **RDS_INSTANCE**: indicates that RDS DB instances are scanned in parallel and shards in each DB instance are scanned in serial. • **PHY_TABLE**: indicates that all table shards are scanned in parallel.	RDS_INSTANCE, DATA_NODE, PHY_TABLE	DATA_NODE

Parameter	Description	Value Range	Default Value
connection_idle_timeout	Number of seconds the server waits for activity on a connection before closing it. The default value is **28800**, indicating that the server waits for 28,800 seconds before closing a connection where there is no activity.	60~86400	28800
contains_shard_key	Whether the SELECT, UPDATE, and DELETE statements must contain sharding keys in filter conditions.	OFF or ON	OFF
ddl_precheck_mdl_threshold_time	Threshold of the MDL duration in DDL pre-check. The unit is second. The default value is **120**. If the duration exceeds the threshold, the DDL statement fails to be executed.	1~3600	120
enable_table_recycle	• **ON**: indicates that the table recycle bin is enabled. • **OFF**: indicates that the table recycle bin is disabled. After the table recycle bin is enabled, deleted tables are moved to the recycle bin and can be recovered by running the RESTORE command within seven days.	OFF or ON	OFF
long_query_time	Minimum duration of a query to be logged as slow, in seconds. The default value is **1**, indicating that the query is considered as a slow query if its execution duration is greater than or equal to 1 second.	0.01~10	1

Parameter	Description	Value Range	Default Value
max_allowed_packet	Maximum size allowed for the packets transferred between the server and the client at a time. The value must be a multiple of **1024**.	1024~1073741824	1073741824
max_backend_connections	Maximum of concurrent client connections allowed per DDM instance. When this parameter is set to **0** (default), the maximum concurrent connections from a DDM node to an RDS instance is: (RDS instance's maximum connections - 20)/DDM nodes. This parameter takes effect only after maximum connections are set on RDS.	0~10000000	0

Parameter	Description	Value Range	Default Value
max_connections	Minimum concurrent connections from a DDM instance node to the client. This value depends on node classes and processing capabilities of the target data node. Too many connections may cause connection waiting, affecting performance. The consumption of DDM connections varies with the number of shards and SQL design. For example, if a SQL statement contains a sharding key, each DDM connection consumes one data node connection. If the SQL statement contains no sharding keys and the number of shards is N, N data node connections are consumed. If SQL design is appropriate and processing capabilities of DDM and its data nodes are good enough, you can set this parameter to a value slightly smaller than the product of backend data nodes x maximum connections supported by each data node. Carry out pressure tests on your services and then select a proper value.	10~40000	20000
min_backend_connections	Minimum of concurrent client connections allowed per DDM instance. The default value is **10**.	0~10000000	10

Parameter	Description	Value Range	Default Value
seconds_behind_master	Threshold in seconds of the replication lag between a primary RDS instance to its read replica. The default value is **30**, indicating that the time for data replication between the primary RDS instance and its read replicas cannot exceed 30 seconds. If the time exceeds 30 seconds, the data read requests are no longer forwarded to the read replicas.	0~7200	30
sql_execute_timeout	Number of seconds to wait for a SQL statement to execute before it times out. The default value is **28800**, indicating that the SQL statement times out if its execution time is greater than or equal to 28,800 seconds. For data nodes, ensure that **net_write_timeout** has a greater value than **sql_execute_timeout**.	100~28800	28800
temp_table_size_limit	Number of rows in a temporary table. The default value is **1000000**.	500000~2000000000	1000000
transfer_hash_to_mod_hash	Whether the hash algorithm must be converted into mod_hash during table creation.	OFF or ON	OFF
ultimate_optimize	Whether the SQL execution plan is optimized based on parameter values.	OFF or ON	ON

Parameter	Description	Value Range	Default Value
force_read_master_in_transaction	Whether SQL statements involved in each transaction are read from the master node. **CAUTION** This parameter is available in version 3.0.9 or later. If this feature is enabled in version 3.0.9 but the version is downgraded to earlier than 3.0.9, the feature keeps enabled when the version returns to 3.0.9 or later.	OFF or ON	OFF
ddl_flowcontrol_threshold	Maximum number of DDL executions (including successful and failed executions) allowed per hour.	10~100000	10000
batch_insert_policy	Whether a batch INSERT statement is split into multiple statements and executed in series	**NONE** or **SPLIT** • The version should be: DDM 3.1.5.0 or later	NONE
enable_ccl	Whether to enable SQL concurrency control for a DDM instance. **CAUTION** This parameter is available in version 3.1.7.0 or later. If this feature is enabled in version 3.1.7.0 but the version is downgraded to earlier than 3.1.7.0, the feature keeps enabled when the version returns to 3.1.7.0 or later.	OFF or ON	OFF

10.2 Creating a Parameter Template

A database parameter template acts as a container for parameter configurations 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**.

If 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:

- Changing parameter values in a parameter template only changes parameters in the DDM instance where the template has been applied to.
- 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. Remember 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.
- Each user can create up to 100 parameter templates.
- The parameter template quota is shared by all DDM instances in a project.

Procedure

Step 1 Log in to the DDM console.

Step 2 Choose **Parameter Templates** and click **Create Parameter Template**.

Step 3 In the displayed dialog box, enter a template name and description and click **OK**.

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

----End

10.3 Modifying a Custom Parameter Template

When using DDM instances, you can modify parameters in the parameter template you created as needed.

You cannot change parameter values in default parameter templates.

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

- After you change a parameter value 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**.
- The time when the modification takes effect is determined by the type of the parameter.
- 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 based on configurations of the default parameter template.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the navigation pane on the left, choose **Parameter Templates**. On the **Custom Templates** page, click the name of a parameter template.

Step 3 On the **Parameter Details** tab page, modify parameters as required. For details, see **Instance Parameters**.

Available operations are as follows:

- To save the modifications, click **Save**.
- To cancel the modifications, click **Cancel**.
- To preview your changes, click **Preview**.

Step 4 After the parameter values are modified, click **Template History** to view details.

- 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. However, the restart caused by node class changes (if any) does not make these modifications take effect.
- Modifying parameters may affect access to the DDM instance. Exercise caution when performing this operation.
- It takes 20s to 60s to have the modifications to take effect.
- After you modify parameters in a parameter template, some modifications immediately take effect for the DDM instance to which the parameter template applies. Exercise caution when performing this operation.

----End

10.4 Comparing Two Parameter Templates

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

Procedure

Step 1 Log in to the DDM console.

Step 2 Choose **Parameter Templates**, click the **Custom Templates** tab, locate the required parameter template, and click **Compare** in the **Operation** column.

You can also click the **Default Templates** tab and compare a default template with a custom template.

Step 3 In the displayed dialog box, select a parameter template and click **OK**.

- 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

10.5 Viewing Parameter Change History

You can view parameters of a DDM instance and change history of custom templates.

Precautions

An exported or custom parameter template has no change history if no parameters in it are modified.

Procedure

Step 1 Log in to the DDM console.

Step 2 Choose **Parameter Templates**, click the **Custom 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

10.6 Replicating a Parameter Template

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 DDM console.

Step 2 Choose **Parameter Templates**, click the **Custom Templates** tab, locate the required parameter template, and click **Replicate** in the **Operation** column.

Step 3 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 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

10.7 Applying a Parameter Template

After you create a parameter template and modify parameters in it, you can apply it to your DDM instances.

Procedure

Step 1 Log in to the DDM console.

Step 2 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 3 In the displayed dialog box, select one or more DDM instances that you want to apply the parameter template to 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

10.8 Viewing Application Records of a Parameter Template

Scenarios

After a parameter template is applied to DDM instances, you can view its application records.

Procedure

Step 1 Log in to the DDM console.

Step 2 Choose **Parameter Templates** in the navigation pane on the left.

Step 3 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

10.9 Modifying the Description of a Parameter Template

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

Precautions

You cannot modify the description of any default parameter template.

Procedure

Step 1 Log in to the DDM console.

Step 2 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 3 Enter a new description. You can click to submit or to cancel the modification.

- The description can contain a maximum of 256 characters but cannot contain special characters >!<"&'=
- After the modification is successful, you can view the new description in the **Description** column.

----End

10.10 Deleting a Parameter Template

You can delete parameter templates that are no longer needed.

Precautions

- 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 DDM console.

Step 2 Choose **Parameter Templates**, click the **Custom Templates** tab, locate the template that you want to delete, and click **More** > **Delete** in the **Operation** column.

Step 3 In the displayed dialog box, click **Yes**.

----End

11 Account Management

11.1 Adding an Administrator Account

DDM allows you to create an administrator account for your instance. This account has the superuser permissions to modify permissions of accounts displayed on the **Accounts** page. The administrator account has read/write permissions for all schemas and tables by default, including schemas being created. Once an administrator account is created, it cannot be deleted.

You can configure an administrator account when you create an instance, or create one on the instance details page after your instance has been created.

Scenarios

- If you forget the password of the administrator account, reset it by referring to **Resetting the Administrator Password**.
- If you select **Skip** when you create a DDM instance, you can create an administrator by referring to **Creating an Administrator Account** on the instance basic information page.

Prerequisites

The DDM kernel version must be 3.0.9 or later.

Precautions

- After an administrator account is created, its username cannot be modified.
- The administrator account cannot be duplicated with any DDM account on the **Accounts** page.

Resetting the Administrator Password

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance that you want to reset the administrator password for and click its name.

Step 3 In the **Instance Information** area, click **Reset Password** in the **Administrator** field.

Step 4 In the displayed dialog box, enter a new administrator password and confirm the password.

Step 5 Click **Yes** and wait the request is submitted.

----End

Creating an Administrator Account

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance that you want to create an administrator password for and click its name.

Step 3 In the **Instance Information** area, click **Create** in the **Administrator** field.

Step 4 In the displayed dialog box, enter an administrator, password, and confirm password.

Step 5 Click **Yes** and wait the request is submitted.

----End

11.2 Creating an Account

Multiple users with different permissions can be created to access a DB instance or a database. You can create an account with required permissions on the DDM console.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance that you want to create an account for and click its name.

Step 3 In the navigation pane, choose **Accounts**.

Step 4 On the displayed page, click **Create Account** and configure the required parameters.

Table 11-1 Required parameters

Parameter	Description
Username	Username of the account. The username can include 1 to 32 characters and must start with a letter. Only letters, digits, and underscores (_) are allowed.

Parameter	Description
Password	Password of the account. The password: • Can include 8 to 32 characters. • Must contain at least three of the following character types: lowercase letters, uppercase letters, digits, and special characters ~!@#%^*-_+? • Cannot be a weak password. It cannot be overly simple and easily guessed. • The password cannot be the same as username or username in reverse order.
Confirm Password	The confirm password must be the same as the entered password.
Password Validity (Days)	Password validity period. The value must be an integer ranging from 0 to 65535, in days. An expired account cannot be used to log in to the system. You need to reset the password and log in again. If an account is created using the administrator account by executing the CREATE USER statement, before you set the password validity period for this account, reset its password first. • If the value is **0**, the password never expires. • If this parameter is not set, the password will always be valid. **NOTE** The DDM kernel version must be 3.0.4 or later.
Schema	Schema to be associated with the DDM account. You can select an existing schema from the drop-down list. The account can be used to access only the associated schemas.
Permissions	Options: **CREATE**, **DROP**, **ALTER**, **INDEX**, **INSERT**, **DELETE**, **UPDATE**, and **SELECT**. You can select any or a combination of them.
Description	Description of the account, which cannot exceed 256 characters.

Step 5 Confirm the settings and click **OK**.

----End

11.3 Modifying an account

You can modify a DDM account on the DDM console.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance that you want to modify an account for and click its name.

Step 3 In the navigation pane, choose **Accounts**.

Step 4 In the account list, locate the required account and click **Modify** in the **Operation** column.

Step 5 In the displayed dialog box, modify the password validity period, associated schemas, permissions, and description.

The value of the password validity period must be an integer ranging from 0 to 65535, in days. If the value is **0**, the password never expires.

Step 6 Click **OK**.

----End

11.4 Deleting an Account

You can delete DDM accounts that are no longer needed.

Precautions

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

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance whose DDM account you want to delete and click the instance name.

Step 3 In the navigation pane, choose **Accounts**.

Step 4 In the account list, locate the account that you want to delete and choose **More** > **Delete** in the **Operation** column.

Step 5 In the displayed dialog box, click **Yes**.

----End

11.5 Resetting the Password of an Account

You can reset the password of an account on the DDM console.

Precautions

- Resetting the DDM account password is a high-risk operation. Ensure that you have the IAM permission to modify DDM accounts.

- An expired account of DDM cannot be used to log in to the system. You need to reset the password and log in again.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the DDM instance whose account password you want to reset and click its name.

Step 3 In the navigation pane, choose **Accounts**.

Step 4 In the account list, locate the required account and choose **More** > **Reset Password** in the **Operation** column.

Step 5 In the displayed dialog box, enter the new password, confirm the new password, and click **OK**.

----End

11.6 Account Permissions

11.6.1 Overview

DDM permissions management is based on MySQL permissions management. DDM supports most of MySQL syntax and permissions. For more information about MySQL accounts and permissions, see **MySQL documentation**.

This document describes DDM account rules, permission levels, permission items, and permission operations.

NOTE

DDM accounts created by CREATE USER or GRANT have nothing to do with accounts of associated RDS instances, and will not be synchronized to the RDS instances either.

11.6.2 Account Requirements

Account

Different from MySQL, DDM identifies an account by username, not by 'username'@'host'.

Username

- Must be case-sensitive.
- Can contain 1 to 32 characters and must start with a letter. Only letters, digits, and underscores (_) are allowed.

Password

- Can include 8 to 32 characters.
- Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters ~!@#%^*-_+?

- Cannot be a weak password. It cannot be overly simple and easily guessed.

Account Locking Rules

- If an account fails to log in to the system for 10 consecutive times, the username and IP address of the account are locked. If a locked account is used to log in to the system for a long time, the system considers that a brute force attack occurs regardless of whether the username and password are correct and continuously updates the lockout duration.
- If the locked account and IP address are not used for more than 180s, they will be automatically unlocked.

  NOTE

  Account locking is supported in version 3.0.8.5 or later.

11.6.3 Managing Permissions

Permission Levels

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

Permission Types

DDM supports different permission types by using the GRANT statement.

Permission Type	Description
ALL	All permissions
DROP	Deleting a table
INDEX	Creating/Deleting an index
ALTER	Executing ALTER statements
CREATE	Creating a table
SELECT	Reading table data
INSERT	Inserting data to a table
UPDATE	Updating data in a table
GRANT	Granting permissions to users
REVOKE	Deleting a user permission
SET	Setting user's passwords

Permission Type	Description
FILE	Uploading database permissions from a file
CREATE USER	Creating a user

Precautions

- Basic permissions of a DDM account can only be modified on the DDM console.
- If a DDM account has table or database permissions on a schema, the schema will be displayed in the row where the account is located.
- Users created by the CREATE USER statement support only user-level permissions.
- If a DDM account has been associated with a schema, deleting this schema or tables in it does not affect the permissions assigned to the account.
- You can execute the GRANT statement to assign permissions to a DDM account. The following is an example statement:
  ```
  grant grant option on {user-level, database-level, and table-level} to DDM account
  ```
- Permissions cannot be assigned to a DDM account unless the account is associated with a schema.

Permission Operations

NOTICE

SHOW GRANTS is supported in versions in 3.0.2 or later. Other functions are available in versions 2.4.1.4 or later.

CREATE USER

Syntax:

```
CREATE USER username IDENTIFIED BY 'password'
```

Example: **Creating an account (username: Jenny; password: *xxxxxx*)**

```
CREATE USER Jenny IDENTIFIED BY 'xxxxxx';
```

NOTE

Each username and password must meet the corresponding requirements.

DROP USER

Syntax:

```
DROP USER username
```

Example: **Removing user Jenny**

```
DROP USER Jenny;
```


SET PASSWORD

Syntax:

```
SET PASSWORD FOR 'username'@'%' = 'auth_string'
```

NOTE

To be compatible with the MySQL syntax, the username must be in the format of 'username'@' %'.

Example: **Changing the password of Jenny to *xxxxxx***

```
SET PASSWORD FOR 'Jenny'@'%' = 'xxxxxx'
```

GRANT

Syntax:

```
GRANT
  priv_type[, priv_type] ...
  ON priv_level
  TO user [auth_option]
  priv_level: {
      | *.*
      | db_name.*
      | db_name.tbl_name
      | tbl_name}
  auth_option: {
      IDENTIFIED BY 'auth#string'
}
```

NOTE

If a GRANT statement provides no accounts and does not specify **IDENTIFIED BY**, a message **No account found** will be returned. If **IDENTIFIED BY** is specified, an account will be created accordingly and permissions will be granted to it.

GRANT ALL [PRIVILEGES] can be used to assign table-, user-, and database-level permissions.

Example 1: Create a **user-level** account with all permissions. The username is **Mike**.

Method 1: Create an account and then grant permissions to it.

```
CREATE USER Mike IDENTIFIED BY 'password';
GRANT SELECT, INSERT ON *.* to Mike;
```

Method 2: Execute one SQL statement to create an account and grant it permissions.

```
GRANT SELECT, INSERT ON *.* to Mike IDENTIFIED BY 'password';
```

Example 2: Create a **database-level** account with all permissions. Create account **david** in database **testdb** and grant the SELECT permissions of database **testdb** to the account.

Method 1: Create an account and then grant permissions to it.

```
CREATE USER david IDENTIFIED BY 'password';
GRANT SELECT ON testdb.* to david;
```

Method 2: Execute one SQL statement to create an account and grant it permissions.

```
GRANT SELECT ON testdb.* to david IDENTIFIED BY 'password';
```


Example 3: Create a **table-level** account with all permissions. Create account **hanson** in database **testdb** and grant all permissions of table **testdb.employees** to the account.

```
GRANT  ALL PRIVILEGES ON testdb.employees to hanson IDENTIFIED BY 'password';
```

REVOKE

Syntax:

```
REVOKE
    priv_type [, priv_type] ...
    ON priv_level FROM user;
```

Example: Deleting CREATE, DROP, and INDEX permissions of user **hanson** on table **testdb.emp**.

```
REVOKE CREATE,DROP,INDEX ON testdb.emp FROM hanson;
```

NOTE

REVOKE can delete actions at each permission level of an account. The permission level is specified by **priv_level**.

SHOW GRANTS

Syntax:

```
SHOW GRANTS FOR user;
```

Example 1: Querying user permissions with any of the following statements:

```
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER;
SHOW GRANTS FOR CURRENT_USER();
```

Example 2: Querying other permissions. This operation can be performed only when the current user can grant user-level permissions.

```
mysql> show grants for david;
+-----------------------------+
|Grants for david             |
+-----------------------------+
|GRANT USAGE ON *.* TO david |
+-----------------------------+
1 row in set (0.00 sec)
```


12 Backups and Restorations

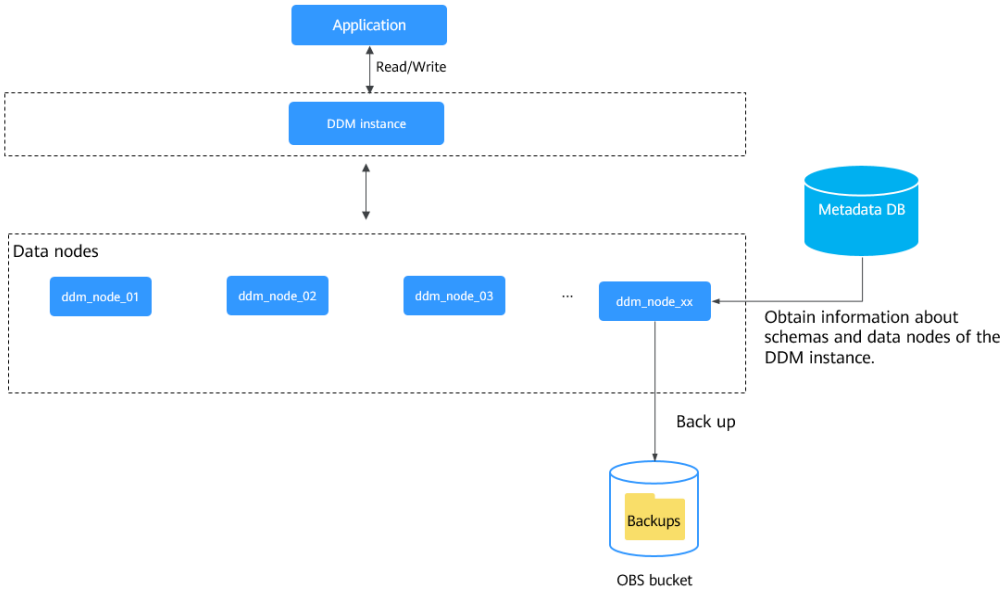
12.1 Backup Principles

DDM instances cannot be backed up manually. The system automatically backs up them from 02:00 to 03:00 GMT+08:00 every day. Metadata backup can also be triggered by any of the key operations such as deleting a schema, clearing source data after shard configuration, and deleting a DDM instance.

DDM instances cannot be backed up manually. The system automatically backs them up at 02:00 UTC+00:00 every day. Metadata backup can also be triggered by any of the key operations such as deleting a schema, clearing source data after shard configuration, and deleting a DDM instance.

Figure 12-1 illustrates the backup process.

Figure 12-1 Backup principles



NOTE

The metadata database stores information of your DDM instances and their associated data nodes. All of your DDM instances in different regions share the same metadata database.

12.2 Consistent Backups

DDM replaces its consistent backup function with a new feature called "metadata restore" under Data Restoration.

After the change, existing consistency backups cannot be used for restoration, and operations related to consistency backups are hidden

You can restore data by referring to section "Restoring Metadata."

12.3 Restoring Data to a New Instance

DDM allows you to restore data from the current instance to any point in time using an existing backup. This is a good choice for routine service backup and restoration.

This section uses an RDS for MySQL instance as an example to describe how to restore data to a new DDM instance.

Precautions

- Restoring data to a new instance restores your DDM instance and its data nodes (RDS for MySQL instances). Before the restoration, you need to prepare a new DDM instance and as many new RDS for MySQL instances as there are data nodes.
- Restoring data to a new DDM instance will overwrite data on it and cause the instance to be unavailable during restoration.
- The new RDS for MySQL instances must have the same or later versions and as much as or more storage space than the source instances.
- Restoration is not supported if the destination DDM instance is in the primary network and its associated RDS for MySQL instance is in the extended network.
- The source DDM instance must be of the version 2.3.2.11 or later, and the destination DDM instance must be of the version 3.0.8 or later.
- Time points that data can be restored to depend on the backup policy set on original data nodes.

Procedure

Step 1 Log in to the DDM console.

Step 2 Create a DDM instance or select an existing DDM instance that meets the requirements in the region where the source DDM instance is located.

NOTE

Ensure that the new DDM instance or the selected existing DDM instance is not associated with any RDS for MySQL instance and has no schemas or accounts.

Step 3 On the RDS console, create as many RDS for MySQL instances as there are in the source DDM instance.

NOTE

- Ensure that the new RDS for MySQL instances have the same or later versions than RDS for MySQL instances associated with the source DDM instance.
- Ensure that each new RDS for MySQL instance has the same or larger storage space than RDS for MySQL instances associated with the source DDM instance.

Step 4 Switch back to the DDM console, in the instance list, locate the DDM instance whose data you want to restore, and click its name.

Step 5 In the navigation pane on the left, choose **Backups & Restorations**.

Step 6 Click **Restore to New Instance**.

Step 7 On the displayed **Restore to New Instance** page, specify a time range and a time point, and select destination DDM instance and associated data nodes.

Table 12-1 Parameter description

Parameter	Description
Time Range	Select a time range.
Time Point	Select a time point DDM checks whether the associated data nodes have available backups at the selected time point.
Destination DDM Instance	Select the DDM instance created in **Step 2** as the destination instance.
Associated Data Nodes	Select the RDS for MySQL instances created in **Step 3** as the destination data nodes.

Step 8 Confirm the information and click **OK**. The restoration duration depends on the data volume of the instance. The average restoration speed is 100 Mbit/s. If there are multiple DNs, data can be restored to the DNs concurrently.

----End

12.4 Restoring Metadata

The metadata backup policy requires DDM to automatically back up DDM instance metadata from 02:00 to 03:00 every day and retain the backup data for up to 30 days. Metadata backup is also triggered by key operations, such as deleting a schema, deleting source data after shard configuration, and deleting instances.

The metadata backup policy requires DDM to automatically back up DDM instance metadata at 02:00 UTC+00:00 every day and retain the backup data for

30 days. Metadata backup is also triggered by key operations that affect metadata, such as deleting a schema, deleting source data after shard configuration, and deleting instances.

When you delete a schema by mistake or your RDS for MySQL instance becomes abnormal, metadata restoration allows you to restore your DDM instance metadata and match the metadata with the RDS for MySQL instance that has PITR completed to re-establish the relationship between your DDM instance and RDS for MySQL instance. The metadata restoration supports only RDS for MySQL.

To restore metadata of a DDM instance, you can specify a point in time by referring to **Restoring Metadata to a Point in Time**, or using an available backup by referring to **Restoring Metadata Using Backups**.

Precautions

- Metadata restoration mainly restores the metadata of your DDM instance to a new DDM instance. It starts after a point-in-time recovery (PITR) for the associated data nodes is complete.

  NOTE

  PITR indicates that a data node has been restored to a specified point in time. For details about how to perform PITR on RDS for MySQL, see section "Data Restorations" > "Restoring Data to RDS for MySQL" > "PITR: Restoring a DB Instance to a Point in Time" in *Working with RDS for MySQL*.
- The destination DDM instance is not associated with any RDS for MySQL instance and has no schemas and accounts.
- Restoration is not supported if the destination DDM instance is in the primary network and its associated RDS for MySQL instance is in the extended network.
- The source DDM instance must be of the version 2.3.2.11 or later, and the destination DDM instance must be of the version 3.0.8 or later.
- Time points that data can be restored to depend on the backup policy set on original data nodes.

Prerequisites

- There is a source DDM instance available and associated with an RDS for MySQL instance. This RDS for MySQL instance is the source data node.
- The source data node has been restored to a specified time point.
- The destination RDS for MySQL instance and the destination DDM instance must be deployed in the same VPC and subnet and have the same security group rules.

Restoring Metadata to a Point in Time

Step 1 Log in to the DDM console.

Step 2 Create a new DDM instance and use it as the destination DDM instance. For details, see **Creating a DDM instance**.

Step 3 In the DDM instance list, locate the source instance and click its name.

Step 4 In the navigation pane on the left, choose **Backups & Restorations**.

Step 5 Click **Restore Metadata**.

Step 6 On the displayed page, specify a time point. DDM will select an appropriate DDM metadata backup closest to the time point.

Table 12-2 Parameter description

Parameter	Description
Restore To	Select a time point to restore metadata. After you specify a point in time, DDM selects the most recent backup and uses it to restore metadata.
Destination DDM Instance	Select the DDM instance created in **Step 2**.
Destination Data Nodes	Select the RDS for MySQL instance that has PITR completed. You can search for the instance by name or ID. DDM will match the selected data nodes with shard information in the selected metadata backup to restore metadata.

Step 7 Click **OK**. If a message is displayed indicating that the metadata is restored successfully, the restoration is complete.

----End

Restoring Metadata Using Backups

Step 1 Log in to the DDM console.

Step 2 Create a new DDM instance and use it as the destination DDM instance. For details, see **Creating a DDM instance**.

Step 3 In the navigation pane on the left, choose **Backups**. Alternatively, go to the instance details page and choose **Backups** from the navigation pane.

Step 4 Locate the required backup based on the instance name and backup time and click **Restore** in the **Operation** column.

Step 5 On the displayed page, configure required parameters.

Table 12-3 Parameter description

Parameter	Description
Backup Name	Name of the backup to be restored.
Destination DDM Instance	Select the DDM instance created in **Step 2**.

Parameter	Description
Destination Data Nodes	Select the RDS for MySQL instance that has PITR completed. DDM will match the selected data nodes with shard information in the selected metadata backup to restore metadata.

Step 6 Click **OK**. If a message is displayed indicating that the metadata is restored successfully, the restoration is complete.

----End

13 Data Migration

13.1 Overview

Data migration is a process of migrating data from databases to DDM or exporting data from DDM to other database systems. You can use the MySQL tool mysqldump to export data. If both full and incremental migrations are required, use Data Replication Service (DRS).

Precautions

- Services may be interrupted during migration. The duration of the interruption depends on the amount of data to be migrated and on network conditions.
- Data migration is complicated and is recommended during off-peak hours. This guide is for reference only. Design a proper migration solution based on your service scenario, data volume, and downtime requirements.
- To migrate a large amount of data or a large data table, perform a rehearsal before formal migration.
- DDM can be accessed only using an ECS. So you need to export databases as files, upload the files to the ECS, and import the file data from the ECS to DDM.

Migration Scenarios

Data migration is involved in the following scenarios:

1. **Scenario 1: Migrating Data from an On-Premises MySQL Instance to DDM**
2. **Scenario 2: Migrating Data from a Third-Party Cloud MySQL Instance to DDM**
3. **Scenario 3: Migrating Data from an ECS-hosted MySQL Instance to DDM**
4. **Scenario 4: Exporting Data from a DDM Instance**

13.2 Migration Evaluation

Evaluate and verify data migration before migrating your database to a cloud platform.

Based on the status of the data to be migrated and the future service scale, evaluate migration scenarios separately and make the required preparations.

Table 13-1 Evaluation and preparation before data migration

Item	Description
Amount of the data to be migrated and class of the required DDM instance and RDS instance	• Use vertical sharding and then horizontal sharding to shard the source RDS for MySQL instance. • Execute the following SQL statement to evaluate storage space occupied by the source RDS for MySQL instance: `select concat(round(sum(DATA_LENGTH/ 1024/1024),2),'MB')as data from information_schema.TABLES;` • Partition a table with more than 10 million rows (or expected to exceed 10 million rows). • Ensure that data stored on a single RDS instance does not exceed 500 GB.
Information about schemas and logical tables mapped to each table in the source database	• Specify the information about logical tables mapped to each source table, including the number of data records, logical table type, schema, DDM instance, and associated RDS instances. • If there is an RDS for MySQL instance available in the destination DDM instance and the target schema is unsharded, associate the RDS for MySQL instance with the schema with no need to migrate table structures and data.

Item	Description
Compatibility	• Check whether the source database uses the same MySQL version as the target MySQL instance associated with the destination DDM instance. • The class and storage space of the destination instance cannot be less than those of the source database. • Check whether the table structure and character set of the source database are the same as those of the destination instance. • For a logical table that uses hash sharding, ensure that the number of data records to be migrated each time does not exceed 10 million. For a logical table using range sharding, the number cannot exceed 5 million. • If a table contains a huge number of data records, import and export the data in batches. Specify the WHERE condition on mysqldump to limit the number of data records to be imported or exported at a time. • SQL text files imported into DDM can only contain standard DML INSERT statements. • Evaluate the compatibility of SQL statements in your application with DDM.

Before migrating data, collect the required information listed in **Table 13-2**.

Table 13-2 Information to be collected

Source/Destination	Information Item
Source RDS instance	Connection address
	Listening port
	Database account
	Database name
	Table name
Destination DDM instance	Connection address
	Listening port
	Username
	Name of a shard on the RDS instance associated with the DDM instance

Source/Destination	Information Item
	Connection address of the new RDS instance
	Listening port of the new RDS instance
	Administrator of the new RDS instance
	Database name
ECS	EIP
	Login credentials (username and password)

13.3 Scenario 1: Migrating Data from an On-Premises MySQL Instance to DDM

Scenarios

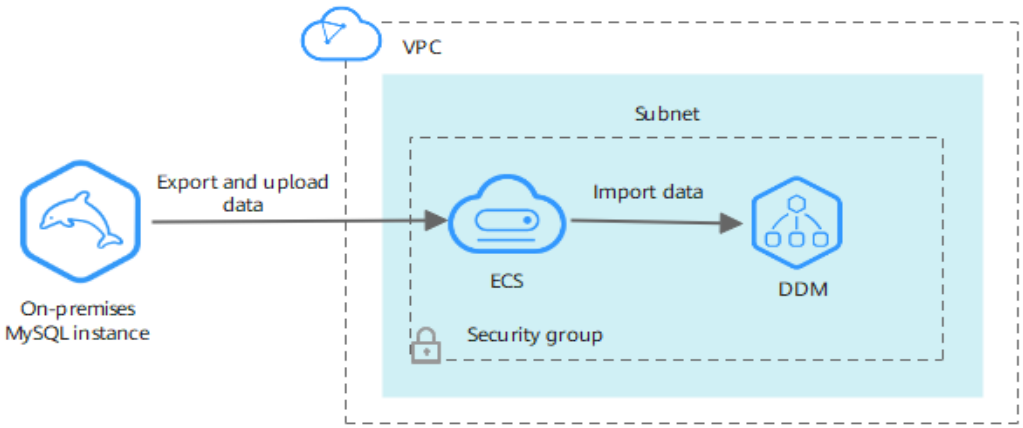
You are using an on-premises MySQL instance and want to use DDM to store data in a distributed manner.

NOTICE

Services may be interrupted during migration. The duration of the interruption depends on the amount of data to be migrated and on network conditions.

Migration Process

Figure 13-1 Migrating data from an on-premises MySQL instance to DDM



Constraints

- The destination DDM instance is associated with a prepared RDS for MySQL instance and can communicate with the ECS where the RDS for MySQL instance is deployed.
- Before data migration, you have to stop your workloads to ensure data integrity.
- Before migrating data, you have to prepare a new DDM instance and create schemas, sharded tables, or broadcast tables with the same names and structures as the source. Methods for creating different types of logical tables are described in **Table 13-4**.
- The prepared RDS for MySQL instance must be of the same MySQL version as the source MySQL instance.

Preparing for the Migration

- Prepare an ECS that can access the data center where the source MySQL instance is deployed.
  a. Ensure that the data center can communicate with the destination DDM instance, prepared RDS for MySQL instance, and prepared ECS.
  b. Install an official MySQL (5.6 or 5.7) client on the ECS and the following OSs by running the required commands:
     - Red Hat Linux: **yum install mysql mysql-devel**
     - Debian Linux: **apt install mysql-client-5.7 mysql-client-core-5.7**
  c. Ensure that there is enough disk space and memory on the ECS to store and compare dump files.
- Prepare a DDM instance and configure information about DDM accounts, schemas, and logical tables.
  a. Create a DDM instance and then create a DDM account and schema on the DDM console.
  b. Export table structures of the source MySQL instance to a SQL text file.
     - If the MySQL client version is 5.6 or 5.7, run the following command:
       mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --no-data --skip-add-locks --add-locks=false --skip-tz-utc **{DB_NAME}** **{TABLE_NAME}** > **{mysql_table_schema.sql}**
     - If the MySQL client version is 8.0, run the following command:
       mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --no-data --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc **{DB_NAME}** **{TABLE_NAME}** > **{mysql_table_schema.sql}**

     Table 13-3 describes the parameters in the command.

Table 13-3 Parameter description

Parameter	Description	Remarks
DB_ADDRESS	Indicates the connection address of the target database.	This parameter is mandatory.
DB_PORT	Indicates the listening port of the target database.	This parameter is mandatory.
DB_USER	Indicates the database account.	This parameter is mandatory.
--skip-lock-tables	Indicates that data is exported when tables are not locked.	The declaration for table locking is enabled for some parameters by default. Add this parameter to the end of each data export statement.
--add-locks=false	Indicates that no locks are added to tables.	-
--no-data	Indicates that only database table structures are exported.	This parameter is used to export table structures.
--column-statistics=0	Indicates column statistics is disabled when the MySQL client version is 8.0.	If the MySQL client version is 8.0, this parameter is mandatory.
DB_NAME	Indicates the database name.	This parameter is mandatory.
TABLE_NAME	Indicates the table name.	Separate table names of the same type using a space. Exporting only service-related table structures is recommended.

Parameter	Description	Remarks
mysql_table_schema.sql	Indicates the name of the generated table structure file.	The file varies depending on the table whose structure is exported. You can name the file in the format of *schema name_logical table name*_**schema** to prevent data from being overwritten, for example, **mysql_table_schema.sql**.

NOTE

The parameters listed above are generally used for data export. Not all mysqldump parameters are listed here. If some parameters are tuned, log in to the official MySQL website to view them.

c. Create a logical table.

Ensure that structures of the logical table are consistent with those exported in **b**. Map source tables to the logical table of the destination DDM instance, and specify migration policies for different table structures and data.

NOTE

Before creating a logical table, execute SHOW CREATE TABLE *{TABLE_NAME}* to query data table structures in the on-premises MySQL instance.

Table 13-4 Table migration policies

Schema Type	Logical Table Type	Table Structure Migration Policy	Table Data Migration Policy
Sharded	Sharded	Specify a sharding key in each SQL statement for creating original tables, connect to DDM, and execute the new SQL statement on the corresponding schema. For details about how to add a sharding key, see documents about CREATE TABLE statement.	Import source table data using DDM.

Schema Type	Logical Table Type	Table Structure Migration Policy	Table Data Migration Policy
Sharded	Broadcast	Specify a broadcast table in each SQL statement for creating original tables, connect to DDM, log in to the corresponding schema, and execute it. For details about how to add a broadcast table, see documents about CREATE TABLE statement.	
Sharded	Unsharded	Obtain the SQL statement for creating original tables, connect to DDM, log in to the corresponding schema and execute the statement.	
Unsharded	Unsharded		

d. Clear test data in the destination DDM instance to prevent conflicts with the data to be migrated.

- Prepare an RDS for MySQL instance.

Exporting Data

Connect to the on-premises MySQL instance using the IP address of the ECS and export data from the instance using mysqldump.

Before you export data, make sure that:

- The destination DDM instance is in the same subnet and VPC as the ECS on the client side.
- Security group rules allow DDM access.

Export table data from the source MySQL instance to a SQL text file, and upload the file to the prepared ECS.

Step 1 Stop the service system of the source MySQL instance to ensure that exported data is up to date.

Step 2 Open your MySQL client and run the following command to connect to the on-premises MySQL instance and export data as a SQL text file:

- If the MySQL client version is 5.6 or 5.7, run the following command:
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments


```
--skip-add-locks --add-locks=false --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME}
> {mysql_table_data.sql}
```

- If the MySQL client version is 8.0, run the following command:

```
mysqldump -h {DB_ADDRESS} -P {DB_PORT} -u {DB_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data.sql}
```

NOTE

If the source MySQL instance contains multiple schemas, export data from each schema separately.

Table 13-5 describes the parameters in the above command.

Table 13-5 Parameter description

Parameter	Description	Remarks
DB_ADDRESS	Indicates the connection address of the target database.	This parameter is mandatory.
DB_PORT	Indicates the listening port of the target database.	This parameter is mandatory.
DB_USER	Indicates the database account.	This parameter is mandatory.
--complete-insert	Uses a complete INSERT statement (including column names).	-
--single-transaction	After this parameter is configured, a BEGIN SQL statement is submitted before data is exported. The BEGIN SQL statement does not block any application and ensures data consistency when the data is exported. It applies only to the multi-version storage engine, InnoDB.	-
--quick	Directly exports data to standard output files without buffering queries.	This avoids sharply increasing memory usage if there is massive volumes of data.

Parameter	Description	Remarks
--hex-blob	Exports binary string fields in hexadecimal format. This parameter is mandatory if there is binary data to be exported.	-
--no-create-info	Exports data without adding any CREATE TABLE statements.	This parameter is used for data export.
--skip-comments	Disables additional comments.	-
--add-locks=false	Indicates that no locks are added to tables.	-
--set-gtid-purged=OFF	If the MySQL version is 8.0 or 5.7, configure this parameter.	If the MySQL version is 5.6 or earlier, do not configure this parameter.
--skip-add-locks	Controls lock operations during data export to avoid performance issues.	-
--where	Dumps only the records selected based on a specified WHERE condition.	If the condition contains command comment symbols such as special spaces or characters, put the condition in quotes.
DB_NAME	Indicates the database name.	This parameter is mandatory.
TABLE_NAME	Indicates the table name.	Separate every table names of the same type using a space. Exporting only service-related table structures is recommended.
mysql_table_data.sql	Indicates the name of the generated table data file.	The file name varies depending on the table whose data is exported. You can name the file in the format of *schema name_logical table name*_**data** to prevent data from being overwritten, for example, **mysql_table_data.sql**.

NOTE

- The parameters listed above are generally used for data export. Not all mysqldump parameters are listed here. If some parameters are tuned, log in to the official MySQL website to view them.
- Ensure that your MySQL client has the same MySQL version as the RDS instance associated with your destination DDM instance if you want to migrate data using mysqldump. If the versions are inconsistent, data export may be affected.

Step 3 Check the size of the SQL text file and check whether data is successfully exported.

- If the file size is not 0 bytes, data export is successful.
- If the file size is 0 bytes, data export fails. Contact DDM technical support personnel.

Step 4 Upload the SQL file to the prepared ECS.

----End

Importing Data

Step 1 Enable read-only access to DDM databases on your application.

Step 2 Clear test data in the destination DDM instance to prevent conflicts with the data to be migrated.

Step 3 If you want to import unsharded or common tables, use a MySQL client to connect to the destination DDM instance and run the following commands:

```
mysql -f -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_schema.sql}
Enter password: **********
mysql -f -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_data.sql}
Enter password: **********
```

- **DDM_ADDRESS** indicates the address of the destination DDM instance that data will be imported into.
- **DDM_PORT** indicates the port number of the destination DDM instance.
- **DDM_USER** indicates the username for logging in to the destination DDM instance.
- **DB_NAME** indicates the name of the DDM schema. If you want to import unsharded tables, set **DB_NAME** to the name of the first shard in the DDM instance.
- **mysql_table_schema.sql** indicates the name of the table structure file to be imported.
- **mysql_table_data.sql** indicates the name of the table data file to be imported.

NOTE

Before importing data of unsharded or common tables, delete the last line (for example, **Dump completed on 2018-06-28 19:53:03**) in the table structure file. Otherwise, data import may fail.

Step 4 If you want to import sharded tables or broadcast tables, use a MySQL client to connect to the destination DDM instance and run the following command:

```
mysql -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_data.sql}
Enter password: **********
```

- **DDM_ADDRESS** indicates the address of the destination DDM instance that data is imported into.
- **DDM_PORT** indicates the listening port of the destination DDM instance.
- **DDM_USER** indicates the username for logging in to the destination DDM instance.
- **DB_NAME** indicates the name of the DDM schema.
- **mysql_table_data.sql** indicates the name of the table data file to be imported.

NOTICE

- Import data during off-peak hours. Performance of the destination DDM instance and its associated RDS for MySQL instance may be affected during data import.
- If an interruption or exception occurs during data import, execute TRUNCATE TABLE *{TABLE_NAME}* to clear and import data again, to prevent primary key conflicts. Executing this statement will clear all table data. Exercise caution when executing it.
- Ensure that the number of data records to be imported into a broadcast table is less than 5 million.

----End

Verify Data

Step 1 Back up logical information of the destination DDM instance on the ECS.

- Export table structures:
  - If the MySQL client version is 5.6, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password  --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
    ```
  - If the MySQL client version is 8.0, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password --column-statistics=0 --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
    ```
- Export table data:
  - If the MySQL client version is 5.6, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data_new.sql}
    ```
  - If the MySQL client version is 8.0, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-
    ```



```
info --skip-comments --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME} {TABLE_NAME} > {mysql_table_data_new.sql}
```

Step 2 Check data consistency.

1. Execute the following SQL statement on the source MySQL instance and destination DDM instance to check whether there are the same number of records in each table. **{TABLE_NAME}** indicates the table name.
   ```
   select count(*) from {TABLE_NAME};
   ```
2. On the ECS, check whether table structures and data records are consistent before and after the export:
   ```
   diff -B -w -q -i {mysql_table_schema.sql} {mysql_table_schema_new.sql};echo $?
   diff -B -w -q -i {mysql_table_data.sql} {mysql_table_data_new.sql};echo $?
   ```

   NOTE

   - The command for exporting table structures is available only to unsharded and common tables.
   - Table structures and data records cannot be compared if they are exported in a sequence different from before.

   - If yes, the data is successfully migrated.
   - If the data is inconsistent, contact DDM technical support personnel.

Step 3 Verify in an E2E manner whether your application can read related tables of the destination DDM instance normally.

Step 4 Disable read-only access to the destination DDM instance.

----End

Verify Services

1. Switch the service data source to DDM.
2. Check whether you can read and write data from and to DDM normally.
   - If yes, data migration is completed.
   - If no, switch the service data source to an on-premises MySQL instance. Contact the DDM administrator.

13.4 Scenario 2: Migrating Data from a Third-Party Cloud MySQL Instance to DDM

Scenarios

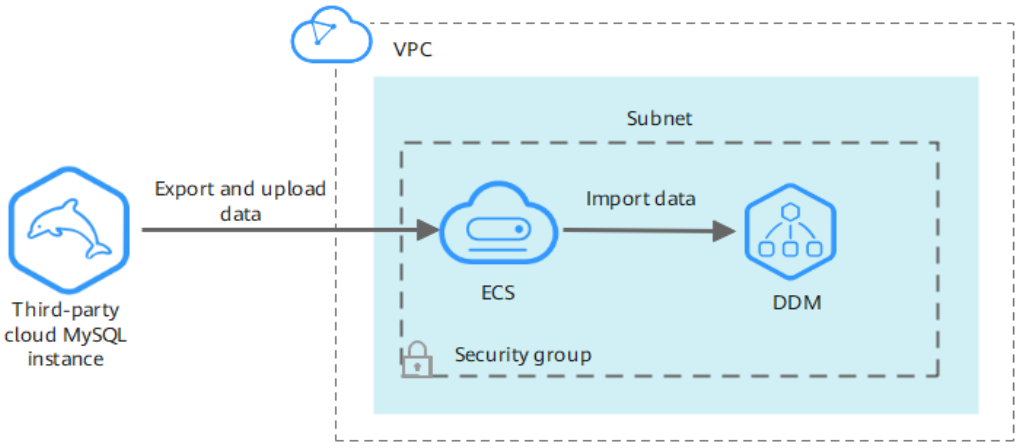
You are using a third-party cloud MySQL instance and want to use DDM to store data in a distributed manner.

NOTE

Services may be interrupted during migration. The duration of the interruption depends on the amount of data to be migrated and on network conditions.

Migration Process

Figure 13-2 Migrating data from a third-party cloud MySQL instance to DDM



Constraints

- The destination DDM instance is associated with a prepared RDS for MySQL instance and can communicate with the ECS where the RDS for MySQL instance is deployed.
- Before data migration, you have to stop your workloads to ensure data integrity.
- Before migrating data, you have to prepare a new DDM instance and create schemas, sharded tables, or broadcast tables with the same names and structures as the source. Methods for creating different types of logical tables are described in **Table 13-7**.
- The prepared RDS for MySQL instance must be of the same MySQL version as the third-party cloud MySQL instance.

Prepare for the Migration

- Prepare an ECS that can access the third-party cloud MySQL instance.
  a. Ensure that the third-party cloud MySQL instance can communicate with the destination DDM instance, prepared RDS for MySQL instance, and prepared ECS. If they cannot communicate with each other, export data as a file and upload the file to the ECS through a transit server.
  b. Install an official MySQL (5.6 or 5.7) client on the ECS and the following OSs by running the required commands:
     - Red Hat Linux: **yum install mysql mysql-devel**
     - Debian Linux: **apt install mysql-client-5.7 mysql-client-core-5.7**
  c. Ensure that there is enough disk space and memory on the ECS to store and compare dump files.
- Prepare a DDM instance and configure information about DDM accounts, schemas, and logical tables.

a. Create a DDM instance and then create a DDM account and schema on the DDM console.

b. Export table structures of the third-party cloud MySQL instance to a SQL text file.

- If the MySQL client version is 5.6 or 5.7, run the following command:
  ```
  mysqldump -h {DB_ADDRESS} -P {DB_PORT} -u {DB_USER} -p --no-data --skip-add-locks --add-locks=false --skip-tz-utc {DB_NAME} {TABLE_NAME} > {mysql_table_schema.sql}
  ```
- If the MySQL client version is 8.0, run the following command:
  ```
  mysqldump -h {DB_ADDRESS} -P {DB_PORT} -u {DB_USER} -p --no-data --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc {DB_NAME} {TABLE_NAME} > {mysql_table_schema.sql}
  ```

Table 13-6 describes the parameters in the command.

Table 13-6 Parameter description

Parameter	Description	Remarks
DB_ADDRESS	Indicates the connection address of the database whose data is to be exported.	This parameter is mandatory.
DB_PORT	Indicates the listening port of the target database.	This parameter is mandatory.
DB_USER	Indicates the database account.	This parameter is mandatory.
--skip-lock-tables	Indicates that data is exported when tables are not locked.	The declaration for table locking is enabled for some parameters by default. Add this parameter to the end of each data export statement.
--add-locks=false	Indicates that no locks are added to tables.	-
--no-data	Indicates that only database table structures are exported.	This parameter is used to export table structures.
--column-statistics=0	Indicates column statistics is disabled when the MySQL client version is 8.0.	If the MySQL client version is 8.0, this parameter is mandatory.

Parameter	Description	Remarks
DB_NAME	Indicates the database name.	This parameter is mandatory.
TABLE_NAME	Indicates the table name.	Separate table names of the same type using a space. Exporting only service-related table structures is recommended.
mysql_table_schema.sql	Indicates the name of the generated table structure file.	The file varies depending on the table whose structure is exported. You can name the file in the format of *schema name_logical table name*_**schema** to prevent data from being overwritten, for example, **mysql_table_schema.sql**.

NOTE

The parameters listed above are generally used for data export. Not all mysqldump parameters are listed here. If some parameters are tuned, log in to the official MySQL website to view them.

c. Create a logical table.

Ensure that structures of the logical table are consistent with those exported in **b**. Map source tables to the logical table of the destination DDM instance, and specify migration policies for different table structures and data.

NOTE

Before creating a logical table, execute SHOW CREATE TABLE *{TABLE_NAME}* to query table structures of the third-party cloud RDS for MySQL instance.

Table 13-7 Table migration policies

Schema Type	Logical Table Type	Table Structure Migration Policy	Table Data Migration Policy
Sharded	Sharded	Specify a sharding key in each SQL statement for creating original tables, connect to DDM, and execute the new SQL statement on the corresponding schema. For details about how to add a sharding key, see documents about CREATE TABLE statement.	Import source table data using DDM.
Sharded	Broadcast	Specify a broadcast table in each SQL statement for creating original tables, connect to DDM, log in to the corresponding schema, and execute it. For details about how to add a broadcast table, see documents about CREATE TABLE statement.	
Sharded	Unsharded	Obtain the SQL statement for creating original tables, connect to DDM, log in to the corresponding schema and execute the statement.	
Unsharded	Unsharded		

d. Clear test data in the destination DDM instance to prevent conflicts with the data to be migrated.

- Prepare an RDS for MySQL instance.

Export Data

Connect to the third-party cloud MySQL instance using the IP address of the ECS and export data from the instance using mysqldump.

Before you export data, make sure that:

- The destination DDM instance is in the same subnet and VPC as the ECS on the client side.
- Security group rules allow DDM access.

Export table data from the third-party cloud MySQL instance to a separate SQL text file, and upload the file to the prepared ECS.

Step 1 Stop your workloads that use the third-party cloud MySQL instance to ensure that exported data is up to date.

Step 2 Export table data from the third-party cloud MySQL instance to a SQL text file.

- If the MySQL client version is 5.6 or 5.7, run the following command:
  m*ysqldump -h* **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**
- If the MySQL client version is 8.0, run the following command:
  m*ysqldump -h* **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**

NOTE

- If there are multiple schemas in the third-party cloud MySQL instance, export data from each schema separately.
- If the MySQL instance does not allow you to export table data using mysqldump, in the upper right corner of the console, choose **Service Tickets** > **Create Service Ticket** to contact the administrator.

Table 13-8 describes the parameters in the above command.

Table 13-8 Parameter description

Parameter	Description	Remarks
DB_ADDRESS	Indicates the connection address of the database whose data is to be exported.	This parameter is mandatory.
DB_PORT	Indicates the listening port of the target database.	This parameter is mandatory.
DB_USER	Indicates the database account.	This parameter is mandatory.
DB_NAME	Indicates the database name.	This parameter is mandatory.

Parameter	Description	Remarks
TABLE_NAME	Indicates the table name.	Separate every table names of the same type using a space. Exporting only service-related table structures is recommended.
mysql_table_data.sql	Indicates the name of the generated table data file.	The file name varies depending on the table whose data is exported. You can name the file in the format of *schema name_logical table name*_**data** to prevent data from being overwritten, for example, **mysql_table_data.sql**.
--complete-insert	Uses a complete INSERT statement (including column names).	-
--single-transaction	After this parameter is configured, a BEGIN SQL statement is submitted before data is exported. The BEGIN SQL statement does not block any application and ensures data consistency when the data is exported. It applies only to the multi-version storage engine, InnoDB.	-
--quick	Directly exports data to standard output files without buffering queries.	This avoids sharply increasing memory usage if there is massive volumes of data.
--hex-blob	Exports binary string fields in hexadecimal format. This parameter is mandatory if there is binary data to be exported.	-

Parameter	Description	Remarks
--no-create-info	Exports data without adding any CREATE TABLE statements.	This parameter is used for data export.
--skip-comments	Disables additional comments.	-
--skip-lock-tables	Indicates that data is exported when tables are not locked.	The declaration for table locking is enabled for some parameters by default. Add this parameter to the end of each data export statement.
--add-locks=false	Indicates that no locks are added to tables.	-
--skip-add-locks	Controls lock operations during data export to avoid performance issues.	-
--set-gtid-purged=OFF	If the MySQL version is 8.0 or 5.7, configure this parameter.	If the MySQL version is 5.6 or earlier, do not configure this parameter.
--where	Dumps only the records selected based on a specified WHERE condition.	If the condition contains command comment symbols such as special spaces or characters, put the condition in quotes.

NOTE

The parameters listed above are generally used for data export. Not all mysqldump parameters are listed here. If some parameters are tuned, log in to the official MySQL website to view them.

Step 3 Check the size of the SQL text file and check whether data is successfully exported.

- If the file size is not 0 bytes, data export is successful.
- If the file size is 0 bytes, data export fails. Contact the DDM administrator.

Step 4 Upload the SQL file to the prepared ECS.

----End

Import Data

Step 1 Enable read-only access to DDM databases on your application.

Step 2 Clear test data in the destination DDM instance to prevent conflicts with the data to be migrated.

Step 3 If you want to import unsharded or common tables, use a MySQL client to connect to the destination DDM instance and run the following commands:

```
mysql -f -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_schema.sql}
Enter password: **********
mysql -f -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_data.sql}
Enter password: **********
```

- **DDM_ADDRESS** indicates the address of the destination DDM instance that data will be imported into.
- **DDM_PORT** indicates the port number of the destination DDM instance.
- **DDM_USER** indicates the username for logging in to the destination DDM instance.
- **DB_NAME** indicates the name of the DDM schema. If you want to import unsharded tables, set **DB_NAME** to the name of the first shard in the DDM instance.
- **mysql_table_schema.sql** indicates the name of the table structure file to be imported.
- **mysql_table_data.sql** indicates the name of the table data file to be imported.

  NOTE

  Before importing data of unsharded or common tables, delete the last line (for example, **Dump completed on 2018-06-28 19:53:03**) in the table structure file. Otherwise, data import may fail.

Step 4 If you want to import sharded tables or broadcast tables, use a MySQL client to connect to the destination DDM instance and run the following command:

```
mysql -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_data.sql}
Enter password: **********
```

- **DDM_ADDRESS** indicates the address of the destination DDM instance that data is imported into.
- **DDM_PORT** indicates the listening port of the destination DDM instance.
- **DDM_USER** indicates the username for logging in to the destination DDM instance.
- **DB_NAME** indicates the name of the DDM schema.
- **mysql_table_data.sql** indicates the name of the table data file to be imported.

NOTICE

- Import data during off-peak hours. Performance of the destination DDM instance and its associated RDS for MySQL instance may be affected during data import.
- If an interruption or exception occurs during data import, execute TRUNCATE TABLE *{TABLE_NAME}* to clear and import data again, to prevent primary key conflicts. Executing this statement will clear all table data. Exercise caution when executing it.
- Ensure that the number of data records to be imported into a broadcast table is less than 5 million.

----End

Verify Data

Step 1 Back up logical information of the destination DDM instance on the ECS.

- Export table structures:
  - If the MySQL client version is 5.6, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password  --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
    ```
  - If the MySQL client version is 8.0, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password --column-statistics=0 --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
    ```
- Export table data:
  - If the MySQL client version is 5.6, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data_new.sql}
    ```
  - If the MySQL client version is 8.0, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME} {TABLE_NAME} > {mysql_table_data_new.sql}
    ```

Step 2 Check data consistency.

1. Execute the following SQL statement on the source RDS for MySQL instance and destination DDM instance to check whether there are the same number of records in each table. **{TABLE_NAME}** indicates the table name.
   ```
   select count(*) from {TABLE_NAME};
   ```
2. On the ECS, check whether table structures and data records are consistent before and after the export:
   ```
   diff -B -w -q -i {mysql_table_schema.sql} {mysql_table_schema_new.sql};echo $?
   diff -B -w -q -i {mysql_table_data.sql} {mysql_table_data_new.sql};echo $?
   ```


NOTE

- The command for exporting table structures is available only to unsharded and common tables.
- Table structures and data records cannot be compared if they are exported in a sequence different from before.

– If yes, the data is successfully migrated.
– If the data is inconsistent, contact DDM technical support.

Step 3 Verify in an E2E manner whether your application can read related tables of the destination DDM instance normally.

Step 4 Disable read-only access to the destination DDM instance.

----End

Verify Services

1. Switch the service data source to DDM.
2. Check whether you can read and write data from and to DDM normally.
   – If yes, data migration is completed.
   – If no, switch the service data source to the third-party cloud MySQL instance. Contact the DDM administrator.

13.5 Scenario 3: Migrating Data from an ECS-hosted MySQL Instance to DDM

Scenarios

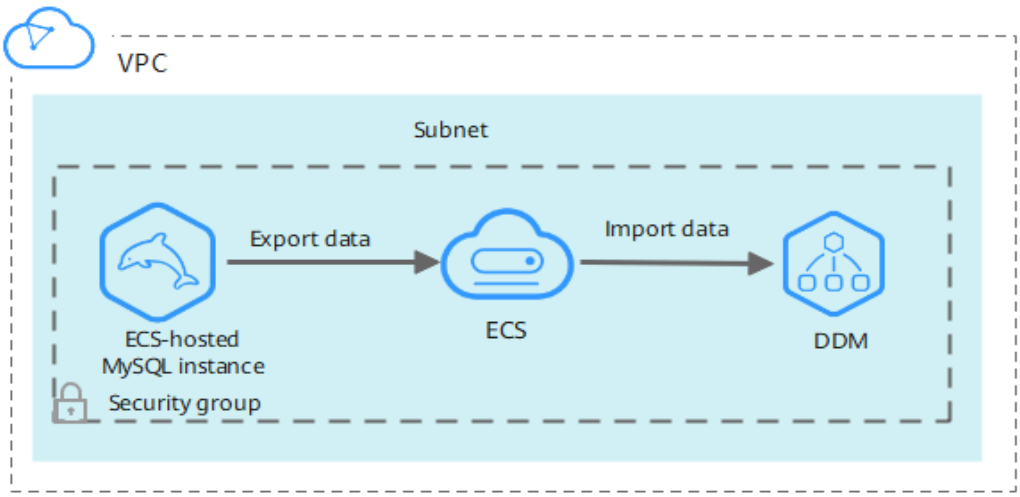
You have built an ECS-hosted MySQL instance and want to migrate your data from the instance to DDM for distributed data storage.

NOTE

Services may be interrupted during migration. The duration of the interruption depends on the amount of data to be migrated and on network conditions.

Migration Process

Figure 13-3 Migrating data from an ECS-hosted MySQL instance to DDM



NOTE

The ECS where the ECS-hosted MySQL instance is located, destination DDM instance, and prepared RDS for MySQL instance must be in the same VPC and have the same security group rules.

Constraints

- The ECS where the MySQL instance is hosted, destination DDM instance, and prepared RDS for MySQL instance must be in the same VPC and security group.
- Before data migration, you have to stop your workloads to ensure data integrity.
- Before migrating data, you have to prepare a new DDM instance and create schemas, sharded tables, or broadcast tables with the same names and structures as the source. **Table 13-10** describes the methods for creating various types of logical tables.
- The new RDS for MySQL instance must be of the same MySQL version as the source RDS for MySQL instance.

Prepare for the Migration

- Prepare an ECS that can access the source MySQL instance.
  a. Ensure that the ECS where the source MySQL instance is located, destination DDM instance, and prepared new RDS for MySQL instance are in the same VPC.
  b. Configure the same security group for the ECS, destination DDM instance, and prepared new RDS for MySQL instance. If they belong to different security groups, configure security group rules to allow them to access each other.

c. Install an official MySQL (5.6 or 5.7) client on the ECS and the following OSs by running the required commands:

- Red Hat Linux: **yum install mysql mysql-devel**
- Debian Linux: **apt install mysql-client-5.7 mysql-client-core-5.7**

d. Ensure that there is enough disk space and memory on the ECS to store and compare dump files.

- Prepare a DDM instance and configure information about DDM accounts, schemas, and logical tables.

a. Create a DDM instance and then create a DDM account and schema on the DDM console.

b. Export table structures of the ECS-hosted MySQL instance to a SQL text file.

- If the MySQL client version is 5.6 or 5.7, run the following command:

```
mysqldump -h {DB_ADDRESS} -P {DB_PORT} -u {DB_USER} -p --single-transaction --set-gtid-purged=OFF --no-data --skip-add-locks --add-locks=false --skip-tz-utc {DB_NAME} {TABLE_NAME} > {mysql_table_schema.sql}
```

- If the MySQL client version is 8.0, run the following command:

```
mysqldump -h {DB_ADDRESS} -P {DB_PORT} -u {DB_USER} -p --single-transaction --set-gtid-purged=OFF --no-data --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc {DB_NAME} {TABLE_NAME} > {mysql_table_schema.sql}
```

Table 13-9 describes the parameters in the command.

Table 13-9 Parameter description

Parameter	Description	Remarks
DB_ADDRESS	Indicates the connection address of the database whose data is to be exported.	This parameter is mandatory.
DB_PORT	Indicates the listening port of the database.	This parameter is mandatory.
DB_USER	Indicates the database account.	This parameter is mandatory.
--skip-lock-tables	Indicates that data is exported when tables are not locked.	Indicates that the declaration for table locking is enabled for some parameters by default. Add this parameter to the end of each data export statement.
--add-locks=false	Indicates that no locks are added to tables.	-

Parameter	Description	Remarks
--no-data	Indicates that only database table structures are exported.	This parameter is used to export table structures.
--column-statistics=0	Indicates column statistics is disabled when the MySQL client version is 8.0.	If the MySQL client version is 8.0, this parameter is mandatory.
DB_NAME	Indicates the database name.	This parameter is mandatory.
TABLE_NAME	Indicates the table name.	Separate every table names of the same type using a space. Exporting only service-related table structures is recommended.
mysql_table_schema.sql	Indicates the name of the generated table structure file.	The file varies depending on the table whose structure is exported. You can name the file in the format of *schema name_logical table name*_**schema** to prevent data from being overwritten, for example, **mysql_table_schema.sql**.

NOTE

The parameters listed above are commonly used for data export. Not all mysqldump parameters are listed here. To optimize certain parameters, log in to the official MySQL website.

c. Create a logical table.

Ensure that structures of the logical table are consistent with those exported in **b**. Map source tables to the logical table of the destination DDM instance, and specify migration policies for different table structures and data.

NOTE

Before creating a logical table, execute SHOW CREATE TABLE *TABLE_NAME* to query structures of data tables in the source MySQL instance.

Table 13-10 Table migration policies

Schema Type	Logical Table Type	Table Structure Migration Policy	Table Data Migration Policy
Sharded	Sharded	Specify a sharding key in each SQL statement for creating original tables, connect to DDM, and execute the new SQL statement on the corresponding schema. For details about how to add a sharding key, see documents about CREATE TABLE statement.	Import source table data using DDM.
Sharded	Broadcast	Specify a broadcast table in each SQL statement for creating original tables, connect to DDM, log in to the corresponding schema, and execute it. For details about how to add a broadcast table, see documents about CREATE TABLE statement.	
Sharded	Unsharded	Obtain the SQL statement for creating original tables, connect to DDM, log in to the corresponding schema, and execute the statement.	
Unsharded	Unsharded		

d. Clear test data in the destination DDM instance to prevent conflicts with the data to be migrated.

- Prepare a new RDS for MySQL instance.

Export Data

Export table data from the ECS-hosted MySQL instance to a separate SQL text file.

Step 1 Stop the service of the ECS-hosted MySQL instance to ensure that the exported data is up to date.

Step 2 Export table data from the ECS-hosted MySQL instance to a SQL text file.

- If the MySQL client version is 5.6 or 5.7, run the following command:
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**
- If the MySQL client version is 8.0, run the following command:
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false **--column-statistics=0** --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**

NOTE

If the ECS-hosted MySQL instance contains multiple schemas, export data from each schema separately.

Table 13-11 describes the parameters in the above command.

Table 13-11 Parameter description

Parameter	Description	Remarks
DB_ADDRESS	Indicates the connection address of the database whose data is to be exported.	This parameter is mandatory.
DB_PORT	Indicates the listening port of the database.	This parameter is mandatory.
DB_USER	Indicates the database account.	This parameter is mandatory.
--single-transaction	After this parameter is configured, a BEGIN SQL statement is submitted before data is exported. The BEGIN SQL statement does not block any application and ensures data consistency when the data is exported. It applies only to the multi-version storage engine, InnoDB.	-

Parameter	Description	Remarks
--hex-blob	Exports binary string fields in hexadecimal format. This parameter is mandatory if there is binary data to be exported.	-
--complete-insert	Uses a complete INSERT statement (including column names).	-
--set-gtid-purged=OFF	If the MySQL version is 8.0 or 5.7, configure this parameter.	If the MySQL version is 5.6 or earlier, do not configure this parameter.
--quick	Directly exports data to standard output files without buffering queries.	-
--no-create-info	Exports data without adding any CREATE TABLE statements.	-
--skip-comments	Disables additional comments.	-
--skip-add-locks	Controls lock operations during data export to avoid performance issues.	-
--add-locks=false	Indicates that no locks are added to tables.	-
--where	Dumps only the records selected based on a specified WHERE condition.	If the condition contains command comment symbols such as special spaces or characters, put the condition in quotes.
DB_NAME	Indicates the database name.	This parameter is mandatory.
TABLE_NAME	Indicates the table name.	Separate every table names of the same type using a space. Exporting only service-related table structures is recommended.

Parameter	Description	Remarks
mysql_table_data.sql	Indicates the name of the generated table data file.	The file name varies depending on the table whose data is exported. You can name the file in the format of *schema name_logical table name*_**data** to prevent data from being overwritten, for example, **mysql_table_data.sql**.

NOTE

- The parameters listed above are commonly used for data export. Not all mysqldump parameters are listed here. To optimize certain parameters, log in to the official MySQL website.
- Ensure that your MySQL client has the same MySQL version as the target RDS for MySQL instance if you want to migrate data using mysqldump. If the versions are inconsistent, data export may be affected.

Step 3 Check the size of the SQL text file and check whether data is successfully exported.

- If the file size is not 0 bytes, data export is successful.
- If the file size is 0 bytes, data export fails. Contact the DDM administrator.

----End

Import Data

Step 1 Enable read-only access to DDM databases on your application.

Step 2 Clear test data in the destination DDM instance to prevent conflicts with the data to be migrated.

Step 3 If you want to import unsharded or common tables, use a MySQL client to connect to the destination DDM instance and run the following commands:

```
mysql -f -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_schema.sql}
Enter password: **********
mysql -f -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_data.sql}
Enter password: **********
```

- **DDM_ADDRESS** indicates the address of the destination DDM instance that data will be imported into.
- **DDM_PORT** indicates the port number of the destination DDM instance.
- **DDM_USER** indicates the username for logging in to the destination DDM instance.
- **DB_NAME** indicates the name of the DDM schema. If you want to import unsharded tables, set **DB_NAME** to the name of the first shard in the DDM instance.

- **mysql_table_schema.sql** indicates the name of the table structure file to be imported.
- **mysql_table_data.sql** indicates the name of the table data file to be imported.

  NOTE

  Before importing data of unsharded or common tables, delete the last line (for example, **Dump completed on 2018-06-28 19:53:03**) in the table structure file. Otherwise, data import may fail.

Step 4 If you want to import sharded tables or broadcast tables, use a MySQL client to connect to the destination DDM instance and run the following command:

```
mysql -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p {DB_NAME} < {mysql_table_data.sql}
Enter password: **********
```

- **DDM_ADDRESS** indicates the address of the destination DDM instance that data is imported into.
- **DDM_PORT** indicates the listening port of the destination DDM instance.
- **DDM_USER** indicates the username for logging in to the destination DDM instance.
- **DB_NAME** indicates the name of the DDM schema.
- **mysql_table_data.sql** indicates the name of the table data file to be imported.

NOTICE

- Import data during off-peak hours. Performance of the destination DDM instance and its associated RDS for MySQL instance may be affected during data import.
- If an interruption or exception occurs during data import, execute TRUNCATE TABLE *{TABLE_NAME}* to clear and import data again. Executing this statement will clear all table data. Exercise caution when executing it.
- Ensure that the number of data records to be imported into a broadcast table is less than 5 million.

----End

Verify Data

Step 1 Back up logical information of the destination DDM instance on the ECS.

- Export table structures:
  - If the MySQL client version is 5.6, run the following command:

    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password  --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
    ```

  - If the MySQL client version is 8.0, run the following command:

    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password --column-statistics=0 --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
    ```


- Export table data:
  - If the MySQL client version is 5.6 or 5.7, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-add-locks --add-locks=false --skip-comments --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data_new.sql}
    ```
  - If the MySQL client version is 8.0, run the following command:
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-add-locks --add-locks=false --skip-comments --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data_new.sql}
    ```

Step 2 Check data consistency.

1. Execute the following SQL statement on the ECS-hosted MySQL instance and destination DDM instance to check whether there are the same number of records in each table. **{TABLE_NAME}** indicates the table name.
   ```
   select count(*) from {TABLE_NAME};
   ```
2. On the ECS, check whether table structures and data records are consistent before and after the export:
   ```
   diff -B -w -q -i {mysql_table_schema.sql} {mysql_table_schema_new.sql};echo $?
   diff -B -w -q -i {mysql_table_data.sql} {mysql_table_data_new.sql};echo $?
   ```

   NOTE

   - The command for exporting table structures is available to only unsharded and common tables.
   - Table structures and data records cannot be compared if they are exported in a sequence different from before.

   - If yes, the data is successfully migrated.
   - If the data is inconsistent, contact DDM technical support personnel.

Step 3 Verify in an E2E manner whether your application can read related tables of the destination DDM instance normally.

Step 4 Disable read-only access to DDM databases on your application.

----End

Verify Services

1. Switch the service data source to DDM.
2. Check whether you can read and write data from and to DDM normally.
   - If yes, data migration is completed.
   - If no, switch the service data source to the ECS-hosted MySQL instance. Contact the DDM technical support personnel.

13.6 Scenario 4: Exporting Data from a DDM Instance

Scenarios

Export data from a DDM instance to a SQL text file.

Precautions

- Export table data during off-peak hours. This is because performance of the DDM instance and its associated RDS for MySQL instances may be affected during the export.
- Use mysqldump to dump database data on a large disk to ensure that there is enough disk space.
- Run the following command in Linux to ensure that mysqldump still works when a session times out:

  nohup {mysqldump CLI} &

Export Table Structure

If the DDM version is 2.4.*X* or later, run the following command to export table structures:

- If the MySQL client version is 5.6 or 5.7, run the following command:

  ```
  mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --no-data --skip-lock-tables --default-auth=mysql_native_password --skip-tz-utc {DB_NAME} {TABLE_NAME} > {mysql_table_schema_ddm.sql}
  ```

- If the MySQL client version is 8.0, run the following command:

  ```
  mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --no-data --skip-lock-tables --default-auth=mysql_native_password --column-statistics=0 --skip-tz-utc {DB_NAME} {TABLE_NAME} > {mysql_table_schema_ddm.sql}
  ```

Export Table Data

If the DDM version is 2.4.*X* or later, run the following command to export table data:

```
mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --add-locks=false --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments [--where=""] --skip-tz-utc {DB_NAME} {TABLE_NAME} > {mysql_table_data_ddm.sql}
```

NOTE

- You can connect to the target RDS for MySQL instance to export data of unsharded tables to improve export efficiency.
- For details about mysqldump5.7, visit **https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html**.

14 Session Management

You can manage sessions in the following scenarios:

- If the number of sessions of an instance has reached the upper limit and the instance cannot be logged in to any longer, you can view and kill unnecessary sessions using the emergency channel function.
- You can view history logs to learn details of the kill operations that you performed using the emergency channel function.

Precautions

- DDM allows you to view CN sessions (connections between applications and DDM) and DN sessions (connections between DDM and data nodes).
- Only RDS for MySQL instances are supported.
- Instances in the creating or abnormal state are not supported.
- When the CPU is overloaded, requests to kill sessions may time out for about one minute. You can refresh the page to check whether the target sessions are available. If yes, kill the sessions again.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click its name.

Step 3 In the navigation pane, choose **Sessions** and go to the **CN Logical Sessions** page.

Step 4 On the displayed page, view sessions between applications and DDM.

- In the upper part of the page, select a filter and enter a keyword to query the session information.
- In the session list, select one or more sessions, click **Kill**. In the displayed dialog box, click **OK**.
- In the session list, click a session attribute to sort sessions in ascending or descending order.

Step 5 Click the **DN Physical Session** tab to view sessions between DDM and data nodes (RDS for MySQL instances).

- In the upper part, select one or more data nodes to view sessions between the DDM instance and its associated data nodes. If multiple data nodes are

selected, the number of pages and total sessions of only one data node are displayed.

- In the upper part, enter an SQL statement or keyword to filter session information.
- In the session list, select one or more sessions, click **Kill**. In the displayed dialog box, click **OK**.
- In the session list, click a session attribute to sort sessions in ascending or descending order.

Step 6 Click the **History Logs** tab to view session-killing operations.

You can also perform the following operations:

- Specify a time range to view the killing operations performed during that time period.
- Select the required DDM instance or data node (RDS for MySQL instance) from the drop-down list in the upper right corner to search for the specified CN or DN sessions.

NOTE

- An open API for killing sessions only transfers session IDs. If a session is killed using an open API, only the ID of the session is displayed on the **History Logs** tab page. Other information about the session is not displayed.

----End

15 Slow Queries

Scenarios

DDM provides a Slow Queries function that sorts out the same type of slow SQL statements within a specified period of time by SQL template. You can specify a time range, search for all types of slow SQL statements within the time range, and then optimize them.

Viewing Log Details

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click its name.

Step 3 In the navigation pane, choose **Slow Queries**.

- On the **Slow Queries** page, specify a time range and view SQL statements executed within this time range.
- In the upper right corner of the page, you can click to export slow query logs.

  NOTE

  The size of the exported slow query log file cannot exceed 8 MB. You can select different periods to export slow query log files in batches.

----End

Downloading a Slow Query Log

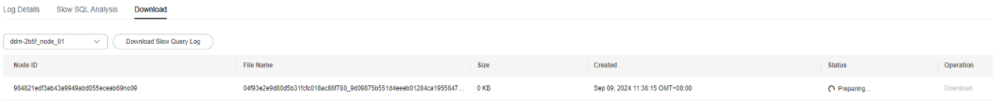
Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click its name.

Step 3 In the navigation pane, choose **Slow Queries**.

Step 4 On the **Download** tab, select a node and click **Download Slow Query Log**.

Figure 15-1 Downloading slow query logs



- Locate a log whose status is **Preparation completed** and click **Download** in the **Operation** column. The system automatically loads the downloading preparation tasks. The time required depends on the log file size and the network environment.
  - When the log is being prepared for download, the file status is **Preparing**.
  - Once the logs are ready for download, the file status changes to **Preparation completed**.
  - Once download preparations fails, the file status changes to **Preparation failed**.
  - Files in the **Preparing** or **Preparation failed** status cannot be downloaded.
- Only ZIP files can be downloaded from this page. The time range is calculated from the time you download the files back to the time when the accumulated file size reaches 40 MB.
- The download link is valid for 5 minutes. After the download link expires, a message is displayed indicating that the download link has expired. If you need to redownload the log, click **OK**.

----End

16 Monitoring

16.1 Supported Metrics

16.1.1 DDM Instance Metrics

Description

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

Namespace

SYS.DDMS

Metrics

Table 16-1 DDM metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
ddm_cpu_util	CPU Usage	CPU usage of the DDM instance node	0—100	DDM nodes	1 minute
ddm_mem_util	Memory Usage	Memory usage of the DDM instance node.	0—100	DDM nodes	1 minute
ddm_bytes_in	Network Input Throughput	Incoming traffic per second of the DDM instance node	≥ 0	DDM nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
ddm_bytes_out	Network Output Throughput	Outgoing traffic per second of the DDM instance node	≥ 0	DDM nodes	1 minute
ddm_qps	QPS	Requests per second of the DDM instance node	≥ 0	DDM nodes	1 minute
ddm_read_count	Reads	Read operations of the DDM instance node within each monitoring period	≥ 0	DDM nodes	1 minute
ddm_write_count	Writes	Write operations of the DDM instance node within a monitoring period	≥ 0	DDM nodes	1 minute
ddm_slow_log	Slow SQL Logs	Slow SQL logs of DDM-Core	≥ 0	DDM nodes	1 minute
ddm_rt_avg	Average Response Latency	Average response latency of DDM-Core	≥ 0	DDM nodes	1 minute
ddm_connections	Connections	Connections of DDM-Core	≥ 0	DDM nodes	1 minute
ddm_backend_connection_ratio	Percentage of Active Connections	Percentage of active connections (from a DDM node to the target RDS instance)	0—100	DDM nodes	1 minute
ddm_active_connections	Active connections	Active connections of each DDM instance node	≥ 0	DDM nodes	1 minute
ddm_connection_util	Connection Usage	Percentage of active connections to each DDM instance node	0—100	DDM nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
ddm_kernel_fgc_cnt	Kernel Full GC Times	Full GC times of each DDM node per minute	≥ 0 counts	DDM nodes	1 minute
ddm_node_status_alarm_code	DDM Node Connectivity	Whether each DDM instance node is unavailable. The value can be **0** and **1**. **0** indicates that the node is available, and **1** indicates that the node is unavailable.	0 or 1	DDM nodes	1 minute

Dimensions

Key	Value
node_id	DDM node ID

16.1.2 Network Metrics

If load balancing is enabled for your DDM instance, you can view network metrics in the following table. If load balancing is not enabled, you do not have the permissions to view them.

Table 16-2 Load balancing metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
m7_in_Bps	Inbound Rate	Traffic used for accessing the monitored object from the Internet per second Unit: byte/s	≥ 0	Dedicated load balancer	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
m8_out_Bps	Outbound Rate	Traffic used by the monitored object to access the Internet per second Unit: byte/s	≥ 0	Dedicated load balancer	1 minute
ma_normal_servers	Healthy Servers	Number of healthy backend servers associated with the monitored object Unit: count	≥ 0	Dedicated load balancer	1 minute
l4_in_bps_usage	Layer-4 Inbound Bandwidth Usage	Percentage of inbound TCP/UDP bandwidth from the monitored object to the client	0–100	Dedicated load balancer	1 minute
l4_out_bps_usage	Layer-4 Outbound Bandwidth Usage	Percentage of outbound TCP/UDP bandwidth from the monitored object to the client	0–100	Dedicated load balancer	1 minute

16.2 Viewing Metrics

16.2.1 Viewing Instance Metrics

Cloud Eye monitors the running status of DDM instances. You can view instance monitoring metrics on the DDM console.

Monitored data requires a period of time for transmission and display. The status of the monitored object displayed on the Cloud Eye page is the status obtained 5 to 10 minutes before. If you have created a DDM instance, wait for 5 to 10 minutes to view its monitored data on Cloud Eye.

Prerequisites

- The DDM instance is running normally.

  Monitored data of faulty or deleted DDM instances are not displayed on Cloud Eye.
- The DDM instance has been normally running for about 10 minutes.

It takes a while to view the monitoring data and graphics of a newly created DDM instance.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click **More** > **View Metric** in the **Operation** column.

Alternatively, click the instance name, on the displayed page, click **View Metric** in the upper right corner.

Step 3 In the instance list, click in the front of the target instance. Locate a node and click **View Metric** in the **Operation** column.

You can view instance metrics, including CPU usage, memory usage, network input throughput, network output throughput, QPS, and slow query logs. For details, see **DDM Instance Metrics**.

----End

16.2.2 Viewing Network Metrics

The DDM console supports monitoring and management of network metrics.

Prerequisites

If load balancing is enabled for your DDM instance, you can view network metrics. If load balancing is not enabled, you do not have the permissions to view them.

Procedure

Step 1 Log in to the DDM console.

Step 2 In the instance list, locate the required DDM instance and click its name.

Step 3 In the navigation pane on the left, choose **Monitoring**.

Step 4 Click **Network**.

You can select a time range and view metrics such as inbound rate, outbound rate, unhealthy servers, and healthy servers. For details, see **Network Metrics**.

----End

16.3 Creating Monitoring Alarm Rules

Scenarios

You can set alarm rules to customize the monitored objects and notification policies to stay aware of the DDM running status in a timely manner.

DDM alarm rules include alarm rule names, resource types, dimensions, monitored objects, metrics, alarm thresholds, monitoring intervals, and whether to send notifications.

Procedure

Step 1 Log in to the DDM console.

Step 2 Click **Service List**, choose **Management & Governance** > **Cloud Eye**.

Step 3 In the navigation pane on the left, choose **Cloud Service Monitoring** > **Distributed Database Middleware**.

Step 4 In the instance list, locate the required instance and click ⌄ on the left of the instance. Select a node and click **Create Alarm Rule** in the **Operation** column.

Step 5 On the displayed page, configure required parameters.

Table 16-3 Alarm rule parameters

Parameter	Description
Name	Specifies the name of the policy. The system generates a random name and you can modify it.
Description	(Optional) Provides supplementary information about the alarm rule.
Method	Select an associated template, use an existing template or create a custom template as required. If you select **Configure manually**, you can configure **Alarm Policy** and **Alarm Severity** as required. **NOTE** After an associated template is modified, the policies contained in this alarm rule to be created will be modified accordingly.

Table 16-4 Alarm notification parameters

Parameter	Description
Alarm Notification	Specifies whether to notify users when alarms are triggered. Notifications can be sent by email, text message, or HTTP/HTTPS message.

Parameter	Description
Notification Recipient	You can select a notification group or topic subscription as required. • Notification group Specifies the notification group that needs to send alarm notifications. • Topic subscription Specifies the object that receives alarm notifications. You can select the account contact or a topic. – Account contact is the mobile phone number and email address of the registered account. – A topic is used to publish messages and subscribe to notifications. If there are no topics that you want, create one first and add subscriptions to it.
Notification Window	Cloud Eye sends notifications only within the notification window specified in the alarm rule. If **Notification Window** is set to **08:00-20:00**, Cloud Eye sends notifications only within 08:00-20:00.
Trigger Condition	Specifies the condition for triggering an alarm notification. You can select **Generated alarm** (when an alarm is generated), **Cleared alarm** (when an alarm is cleared), or both.

NOTE

Alarm notifications sent by SMN will be billed.

Step 6 After the configuration is complete, click **Create**.

After the alarm rule is created, Cloud Eye immediately informs you that an exception has occurred when the metric data reaches the specified threshold.

----End

16.4 Event Monitoring

16.4.1 Overview

Event monitoring provides event reporting, querying, and alarming functions, which helps you collect key events or cloud resource operational events to Cloud Eye. When specific events occur, Cloud Eye generates alarms for you.

Event monitoring is enabled by default. You can view monitoring details of both system events and custom events. For details, see **Events Supported by Event Monitoring**.

16.4.2 Viewing Event Monitoring Data

Scenarios

Event monitoring provides event data reporting, query, and alarm reporting. You can collect key events or cloud resource operational events to Cloud Eye. When specific events occur, Cloud Eye generates alarms for you.

Event monitoring is enabled by default. You can view monitoring details about system events and custom events.

This section describes how to view the event monitoring data.

Procedure

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click **More** > **View Metric** in the **Operation** column.

Alternatively, click the instance name, on the displayed page, click **View Metric** in the upper right corner.

Step 3 In the navigation pane on the left, choose **Event Monitoring**.

On the displayed page, all system events generated in the last 24 hours are displayed by default.

You can also click **1h**, **3h**, **12h**, **1d**, **7d**, or **30d** to view events generated in different time periods.

Step 4 Locate an event and click **View Event** in the **Operation** column to view its details.

----End

16.4.3 Creating an Alarm Rule for Event Monitoring

Scenarios

This section describes how to create an alarm rule to monitor an event.

Procedure

Step 1 Log in to the DDM console.

Step 2 Click  in the upper left corner of the page, choose **Management & Governance** > **Cloud Eye**.

Step 3 On the displayed page, in the navigation pane on the left, choose **Event Monitoring**.

Step 4 On the event list page, click **Create Alarm Rule** in the upper right corner.

Step 5 On the **Create Alarm Rule** page, configure required parameters.

Table 16-5 Parameter description

Parameter	Description
Name	Specifies the name of the policy. The system generates a random name and you can modify it.
Description	(Optional) Provides supplementary information about the alarm rule.
Alarm Type	Select **Event**.
Event Type	Specifies the event type, which can be **System event** or **Custom event**.
Event Source	• System event: Specifies the cloud service the event is generated for. Select **Distributed Database Middleware**. • Custom event: The event source must be the same as that of the reported fields and written in the service.item format.
Method	The default value is **Configure manually**.
Alarm Policy	**Event Name** indicates the instantaneous operations users performed on system resources, such as login and logout. For details, see **Events Supported by Event Monitoring**. Specify **Alarm Policy** and **Alarm Severity** as required.

Click  to enable alarm notification. The validity period is 24 hours by default. If the topics you require are not displayed in the drop-down list, click **Create an SMN topic**.

Table 16-6 Alarm notification parameters

Parameter	Description
Alarm Notification	Specifies whether to send email, SMS, HTTP, or HTTPS notifications to users when an alarm is triggered. (You will be billed for the notifications. For details, see .)
Notification Recipient	You can select a notification group or topic subscription as required.
Notification Group	Specifies the notification group that needs to send alarm notifications. .

Parameter	Description
Notification Object	Specifies the object that receives alarm notifications. You can select the account contact or a topic. • **Account contact** is the mobile phone number and email address of the registered account. • **Topic** is used to publish messages and subscribe to notifications. If the required topic is unavailable, create one first and add subscriptions to it.
Notification Window	Cloud Eye sends notifications only within the validity period specified in the alarm rule. If **Notification Window** is set to **08:00-20:00**, Cloud Eye sends notifications only within 08:00-20:00.
Trigger Condition	Specifies the condition for triggering an alarm notification.

Step 6 After the configuration is complete, click **Create**.

----End

16.4.4 Events Supported by Event Monitoring

Table 16-7 Distributed Database Middleware

Event Source	Event Name	Event ID	Alarm Severity	Description	Solution	Impact
DDM			Major	The underlying resources are insufficient.	Release resources and create the instance again.	A DDM instance cannot be created.
	Failed to change node class of a DDM instance	resizeFlavorFailed	Major	The underlying resources are insufficient.	Coordinate resources and try again.	Services on some nodes are interrupted.

Event Source	Event Name	Event ID	Alarm Severity	Description	Solution	Impact
	Failed to scale out a DDM instance	enlargeNodeFailed	Major	The underlying resources are insufficient.	Coordinate resources, delete the node that fails to be added, and add a node again.	A DDM instance cannot be scaled out.
	Failed to scale in a DDM instance	reduceNodeFailed	Major	The underlying resources fail to be released.	Contact technical support.	A DDM instance cannot be scaled in.
	Failed to restart an instance	restartInstanceFailed	Major	The DB instances associated are abnormal.	Check whether DB instances associated are normal. If the instances are normal, contact technical support.	Services on some nodes are interrupted.

Event Source	Event Name	Event ID	Alarm Severity	Description	Solution	Impact
	Failed to create a schema	createLogicDbFailed	Major	The possible causes are as follows: 1. The username and password of the DB instance are incorrect. 2. The security group of the DDM instance and the associated DB instance are incorrectly configured. As a result, the DDM instance cannot communicate with the associated DB instance.	Check the following items: 1. Check whether the username and password of the DB instance are correct. 2. Check whether the security groups associated with the DDM instance and underlying database instance are correctly configured.	Services cannot run properly.
	Failed to bind an EIP	bindEipFailed	Major	The EIP is abnormal.	Try again later. In case of emergency, contact technical support to rectify the fault.	The DDM instance cannot be accessed from the Internet.
	Failed to scale out a schema	migrateLogicDbFailed	Major	The underlying resources fail to be processed.	Contact technical support.	The schema cannot be scaled out.
	Failed to re-scale out a schema	retryMigrateLogicDbFailed	Major	The underlying resources fail to be processed.	Contact technical support.	The schema cannot be scaled out.

Event Source	Event Name	Event ID	Alarm Severity	Description	Solution	Impact
DDM	Instance metadata backup failed	backupMetaDataFailed	Major	The failure is generally caused by an abnormal connection to OBS.	Contact technical support.	Metadata restoration fails because there is no available metadata backup.

17 Task Center

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

Tasks That Can Be Viewed

The following tasks can be viewed:

- Creating a DDM instance
- Deleting a DDM instance
- Changing node 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 DDM instance
- Importing schema information
- Configuring shards
- Retrying shard configuration
- Deleting a backup
- Restarting a node
- Upgrading the version of a DDM instance
- Downgrading the version of a DDM instance
- Rolling back the version

Procedure

Step 1 Log in to the DDM console.

Step 2 Choose **Task Center** in the left navigation pane, locate the required task, and view its details.

- You can locate a task by name, order ID, or DB instance name/ID, or search for the required task by entering an instance ID 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 for 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

18 Tags

Tag Management Service (TMS) enables you to use tags on the console to manage resources. TMS works with other cloud services to manage tags. TMS manages tags globally. Other cloud services manage only their own tags.

Precautions

- A tag consists of a key and value. You can add only one value for each key.
- Each instance can have up to 20 tags.

Adding a Tag

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click its name.

Step 3 In the navigation pane on the left, choose **Tags**. On the displayed page, click **Add Tag**.

Step 4 In the displayed dialog box, enter a tag key and value and click **OK**.

The tag key and value must comply with the following rules.

Table 18-1 Parameter description

Parameter	Description
Tag key	This parameter is mandatory and cannot be null. The key: • Must be unique for each instance. • Can include 1 to 36 characters. • Cannot be an empty string or start with _**sys**_, and cannot start or end with a space. • Cannot contain: Non-printable ASCII characters (0-31), "*", "<", ">", "\", ",", "|", "/"

Parameter	Description
Tag value	This parameter is mandatory. The key value: • Is an empty string by default. • Can contain 0 to 43 characters. • Cannot contain: Non-printable ASCII characters (0-31), "*", "<", ">", "\", ",", "|"

Step 5 View and manage the tag on the **Tags** page.

----End

Editing a Tag

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click its name.

Step 3 In the navigation pane on the left, choose **Tags**, and click **Edit** in the **Operation** column. In the displayed dialog box, change the tag value and click **OK**.

Only the tag value can be edited.

Step 4 View and manage tags on the **Tags** page.

----End

Deleting a Tag

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, locate the required instance and click its name.

Step 3 In the navigation pane on the left, choose **Tags**, and click **Delete** in the **Operation** column. In the displayed dialog box, click **OK**.

Step 4 Verify that the tag is no longer displayed on the **Tags** page.

----End

Searching for Instances by Tag

After tags are added, you can search for instances by tag to quickly find specific types of instances.

Step 1 Log in to the DDM console.

Step 2 On the **Instances** page, click **Search by Tag** in the upper right corner of the instance list.

Step 3 Enter a tag key and a tag value and click **Search**.

Step 4 View the instance found.

----End

19 Auditing

19.1 Key Operations Recorded by CTS

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

Table 19-1 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
Replicating a parameter template	parameterGroup	copyParameterGroup
Creating a DDM instance	instance	createInstance
Creating a schema	logicDB	createLogicDB
Creating a parameter template	parameterGroup	createParameterGroup
Creating an account	user	createUser
Deleting a DDM instance	instance	deleteInstance
Deleting a schema	logicDB	deleteLogicDB
Deleting a parameter template	parameterGroup	deleteParameterGroup
Deleting an account	user	deleteUser

Operation	Resource Type	Trace Name
Scaling out a DDM instance	instance	enlargeNode
Restarting a DDM instance	instance	instanceRestart
Importing schema information	instance	loadMetadata
Switching the route during scaling	logicDB	manualSwitchRoute
Scaling out a schema	logicDB	migrateLogicDB
Modifying a parameter template	parameterGroup	modifyParameterGroup
Changing the route switching time	logicDB	modifyRouteSwitch-Time
Modifying an account	user	modifyUser
Scaling in a DDM instance	instance	reduceNode
Resetting a parameter template	parameterGroup	resetParameterGroup
Resetting the password of an account	user	resetUserPassword
Change instance specifications	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 scale-out task	logicDB	rollbackMigrateLo-gicDB
Configuring access control	instance	switchIpGroup
Synchronizing data node information	instance	synRdsinfo
Upgrading the version of a DDM Instance	instance	upgrade
Adding a tag	instance	addTag
Deleting a tag	instance	deleteTag
Editing a tag	instance	modifyTag
Creating a node group	group	createGroup
Deleting a group	group	deleteGroup
Modifying the floating IP address	instance	modifyIp

Operation	Resource Type	Trace Name
Modifying the name of an instance	instance	modifyName
Restarting a node	node	nodeRestart
Exporting the instance list	instance	exportInstance
Reloading metadata	instance	reloadInstanceConfig
Changing a database port	instance	modifyInstancePort
Querying the flavor change list	instance	queryFlavor2Resize
Querying slow SQL statements	instance	listSlowLogs
Obtaining the instance version	instance	listDatabaseVersions
Querying slow log statistics	instance	querySlowLogAnalysis
Binding an EIP	instance	bindEIP
Unbinding an EIP	instance	unbindEIP
Obtaining information about the EIP bound to an instance	instance	queryEIP
Querying a CN session	instance	queryLogicalProcessList
Killing a CN session	instance	killLogicalProcess
Querying a DN session	instance	queryPhysicalProcessList
Killing a DN session	instance	killPhysicalProcess
Querying information about killed sessions	instance	queryKillProcessesAuditLog
Querying access control	instance	queryIpGroup
Obtaining the tag list	tag	listTags
Obtaining the instance tag list	instance	listInstanceTags
Enabling or disabling SSL for an instance kernel	instance	switchSsl
Obtaining the address for downloading the SSL certificate of an instance	instance	getSslCerts
Querying schemas	instance	listDatabases
Exporting schema information	instance	dumpMetadata
Querying the size of a schema	instance	queryLogicDbSize

Operation	Resource Type	Trace Name
Pre-verifying shard configuration	logicDB	preCheckMigrateLogicDb
Querying the shard configuration pre-verification results	instance	queryPreCheckMigrateLogicDb
Querying details about a shards configuration task	instance	queryMigrateTaskDetail
Updating the SQL blacklist of an instance	logicDB	configSqlBlackList
Obtaining the SQL blacklist	logicDB	querySqlBlacklist
Changing the read/write weight of an instance	instance	updateReadAndWriteStrategy
Changing the read/write weight of instances in batches	instance	batchModifyReadAndWriteStrategy
Querying whether to enable read/write splitting	instance	queryReadWriteSeparationSwitch
Enabling or disabling read/write splitting	instance	updateReadAndWriteStrategySwitch
Querying available data nodes of an instance	instance	ListAvailableRds
Querying data nodes associated with an instance	instance	queryRelatedRdsList
Querying available data nodes for shards configuration	logicDB	queryAvailableMigrateRdsList
Querying shards configuration actions for a schema	instance	queryMigrateAction
Querying kernel task execution logs	instance	listTaskLogs
Checking data node connectivity	instance	checkRdsConnection
Querying logical tables of a schema	logicDB	listLogicTables
Querying details of a logical table	table	showLogicTable
Querying a backup list	backup	listBackups
Deleting a backup	backup	deleteBackup
Querying backup details	backup	showBackup
Restoring metadata	instance	restoreMetaData

Operation	Resource Type	Trace Name
Querying the associated data node at a specified time point	instance	showBackupRelatedDn
Querying the restoration time	instance	queryRestoreTime
Querying RDS instances that can be restored	instance	queryRds4Restore
Querying instances that can be restored	instance	queryDDM4Restore
Checking whether the parameter group name exists	parameterGroup	checkConfiguration-Name
Querying parameter groups	parameterGroup	listConfigurations
Querying details about a parameter group	parameterGroup	showConfiguration
Querying instances that a parameter group of a template can be applied to	parameterGroup	queryApplicableIn-stances
Querying the parameter group change history	parameterGroup	queryModifyHistory
Querying application records of a parameter group of a template	parameterGroup	queryApplyHistory
Querying parameter groups of an instance	instance	showInstanceConfi-guration
Querying the size of a logical table	logicDB	queryLogicTableSize
Verifying weak passwords	project	isWeakPassword
Querying read-only data nodes	dn	queryReadOnlyDBIn-stance
Querying the instance list	instance	listInstances
Querying details of a specified instance	instance	showInstance
Querying details about a schema	logicDB	showDatabase
Querying nodes of an instance	instance	listNodes
Querying details of an instance node	node	showNode
Querying accounts	instance	listUsers

Operation	Resource Type	Trace Name
Changing the security group of an instance	instance	updateInstanceSecurityGroup
Creating a slow query log download task	instance	createSlowLogDownload
Resetting an administrator account	instance	resetAdministrator
Querying downloaded slow query log information	instance	querySlowLogDownLoad
Querying the result of an asynchronous task	job	showJobResult
Querying the latest shard configuration task for a schema	logicDB	queryLatestMigrateJob
Querying DB engines	project	listEngines
Querying enterprise projects	project	listEnterpriseProjects
Querying the node class list	project	listFlavors

19.2 Querying Traces

After CTS is enabled, the system starts recording operations on cloud resources. CTS stores operation records of the last seven days.

This section describes how to query operation records of the last seven days on the CTS console.

Procedure

Step 1 Log in to the management console.

Step 2 Under **Management & Governance**, 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 the 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

20 SQL Syntax

20.1 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 selecting a DDM solution, evaluate the SQL syntax compatibility between your application and DDM.

DDM EXPLAIN

If you add **EXPLAIN** before a SQL statement, you will see a specific execution plan when you execute the statement. You can analyze the time required based on the plan and modify the SQL statement for optimization.

Table 20-1 Common operators

Operator Name	Description
ExtTableScan	This operator is optimized from LogicalTableScan, primarily targeting data retrieval, but it is not limited to full table scans.
Union	This operator can be used to combine the result-set of two or more queries.
HashAggregate	This operator aggregates the input rows by evaluating the values of the designated grouping columns and applying specified aggregate functions, such as GROUP BY, DISTINCT, SUM, and COUNT, to generate the aggregated results. A large number of resources, such as CPU and memory resources, are required to use this operator.
Join	The NestedLoopJoin operator executes inequality joins. The LookUpJoin operator is optimized from NestedLoopJoin.
Sort	Common Sort operators are MergeSort, MemorySort, and TopN.

Operator Name	Description
DirectPushdownNode	This operator directly pushes SQL statements to DNs for execution.
Project	This operator is used for projection. It selects specific columns from the input data or executes calculation on specific columns using functions or expressions before outputting the results.

Table 20-2 Description of the **EXPLAIN** column

Column Name	Description
table	Table that the row of data belongs to
type	Type of the connection. Connection types in descending order of execution speed: **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 choose to optimize indexes, for example, force MySQL to use an index by adding **USE INDEX(indexname)** to a SELECT statement or to ignore an index by adding **IGNORE INDEX(indexname)**.
key_len	Length of the used index. The shorter the length is, the better the index is if accuracy is not affected.
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.
- A logical table name contains only letters, digits, and underscores (_).
- CREATE TABLE ... LIKE statement is not supported.
- CREATE TABLE ... 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.
- Variables cannot be referenced or operated in SQL statements.

  Example:

  ```
  SET @c=1, @d=@c+1;
  SELECT @c, @d;
  ```

- Inserting keyword DEFAULT or updating a sharding key value to DEFAULT is not supported.
- Repeatedly updating the same field in an UPDATE statement is not supported.
- Updating a sharding key using UPDATE JOIN syntax is not supported.
- UPDATE cannot be used to update self-joins.
- Referencing other object columns in assignment statements or expressions may cause unexpected update results.

  Example:

  ```
  update tbl_1 a,tbl_2 b set a.name=concat(b.name,'aaaa'),b.name=concat(a.name,'bbbb')
  on a.id=b.id;
  ```


- If a text protocol is used, BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB data must be converted into hexadecimal data.
- DDM processes invalid data based on **sql_mode** settings of associated MySQL instances.
- UPDATE JOIN supports only joins with WHERE conditions.
- The expression in a SQL statement has a maximum of 1000 factors.

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 types are not supported.

20.2 DDL

20.2.1 Overview

DDM supports common DDL operations, such as creating databases, creating tables, and modifying table structure, but uses different implementation methods from common MySQL databases.

DDL Statements that Can Be Executed on a MySQL Client

CAUTION

- TRUNCATE Syntax

  Example:

  ```
  TRUNCATE TABLE t1;
  ```

  Deletes all data from table **t1**.

  TRUNCATE TABLE has the DROP permission and can delete all data from a table. In logic, TRUNCATE TABLE is similar to the DELETE statement that can delete all rows from a table.
- ALTER TABLE Syntax

  Example:


```
ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
```

Changes the structure of table **t2** and deletes columns **c** and **d** from table **t2**.

ALTER can 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 for tables or table comments.

- DROP INDEX Syntax

  Example:

  ```
  DROP INDEX `PRIMARY` ON t;
  ```

  Deletes the primary key of table **t**.

  DROP INDEX can 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 can add an index to an existing table.

20.2.2 Creating a Table

NOTE

- Do not create tables whose names start with **_ddm**. DDM manages such tables as internal tables by default
- Sharded tables do not support globally unique indexes. If the unique key is different from the sharding key, data uniqueness cannot be ensured.
- The auto-increment key should be of the BIGINT data type. To avoid duplicate values, do not use TINYINT, SMALLINT, MEDIUMINT, INTEGER, or INT as the auto-increment key.
- You need to specify a character set and collation when creating a table to prevent inconsistent collation between tables caused by the difference between the collation of the schema and the default collation of the DN.

Database and Table Sharding

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

```
CREATE TABLE tbpartition_tbl (
id bigint 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) tbpartitions 8;
```

Database Sharding

The following is an example statement when HASH is used:

```
CREATE TABLE dbpartition_tbl (
id bigint 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 bigint 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

A global sequence can also be specified for an unsharded table, but this function is always ignored. An unsharded table provides auto-increment using auto-increment values of corresponding physical tables.

The following is an example statement:

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

20.2.3 Sharding Algorithm Overview

Supported Sharding Algorithms

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

Table 20-3 Sharding algorithms

Algorithm	Description	Database Sharding Supported	Table Sharding Supported
MOD_HASH	Performing a simple modulo operation	Yes	Yes
MOD_HASH_CI	Performing a simple modulo operation (case-insensitive)	Yes	Yes
HASH	Calculating the CRC32 value and performing the modulo operation	Yes	Yes

Algorithm	Description	Database Sharding Supported	Table Sharding Supported
RANGE	Performing a RANGE-based operation	Yes	No
RIGHT_SHIFT	Arithmetically right shifting a sharding key value and then performing a modulo operation	Yes	Yes
YYYYMM	Getting a hash code for a YearMonth object and then performing a modulo operation	Yes	Yes
YYYYDD	Getting a hash code for a YearDay object and then performing a modulo operation	Yes	Yes
YYYYWEEK	Getting a hash code for a YearWeek object and then performing a modulo operation	Yes	Yes
MM	Getting a hash code for a MONTH object and then performing a modulo operation	No	Yes
DD	Getting a hash code for a DAY object and then performing a modulo operation	No	Yes

Algorithm	Description	Database Sharding Supported	Table Sharding Supported
MMDD	Getting a hash code for a MonthDay object and then performing a modulo operation	No	Yes
WEEK	Getting a hash code for a WEEK object and then performing a modulo operation	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 (MySQL data type).
- If a logical table uses different database and table sharding algorithms, DDM will perform full-shard or full-table scanning when you do not specify database and table conditions in SQL queries.

Data Type of Sharding Algorithms

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

Table 20-4 Supported data types

Sharding Algorithm	TINYINT	SMALLINT	MEDIUMINT	INTEGER	INT	BIGINT	CHAR	VARCHAR	DATE	DATETIME	TIMESTAMP	OTHERS
MOD_HASH	√	√	√	√	√	√	√	√	×	×	×	×
MOD_HASH_CI	√	√	√	√	√	√	√	√	×	×	×	×
HASH	√	√	√	√	√	√	√	√	×	×	×	×

Sharding Algorithm	TINYINT	SMALLINT	MEDIUMINT	INTEGER	INT	BIGINT	CHAR	VARCHAR	DATE	DATETIME	TIMESTAMP	OTHERS
RANGE	√	√	√	√	√	√	×	×	×	×	×	×
RIGHT_SHIFT	√	√	√	√	√	√	×	×	×	×	×	×
YYYYMM	×	×	×	×	×	×	×	×	√	√	√	×
YYYYDD	×	×	×	×	×	×	×	×	√	√	√	×
YYYYWEEK	×	×	×	×	×	×	×	×	√	√	√	×
MM	×	×	×	×	×	×	×	×	√	√	√	×
DD	×	×	×	×	×	×	×	×	√	√	√	×
MMDD	×	×	×	×	×	×	×	×	√	√	√	×
WEEK	×	×	×	×	×	×	×	×	√	√	√	×

NOTE

√: Supported. ×: Not supported.

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])}
     [TBPARTITION BY
{{HASH|MOD_HASH|UNI_HASH|RIGHT_SHIFT|YYYYMM|YYYYWEEK|YYYYDD}(column)}
          [TBPARTITIONS num]
     ]
```

20.2.4 Sharding Algorithms

20.2.4.1 MOD_HASH

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).
- If you use a sharding key of the INTEGER type, do not convert the data type of its value in SQL statements. Changing the data type may cause a failure of routing calculation, and data will be routed to the default shard and cannot be found when you query it.

Data Routing

The data route depends on the remainder of the sharding key value divided by database or table shards. If the value is a string, convert the string into a hashed value and calculate the data route based on the value (case sensitive).

For example, MOD_HASH('8') is equivalent to 8% D. D is the number of database or table shards.

Calculation Method

Method 1: Use an Integer as the Sharding Key

Table 20-5 Required calculation methods when the sharding key is the integer data type

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
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 **NOTE** The database routing result is rounded off to the nearest integer.	Table shard: 16 % (8 x 3) = 16 Database shard: 16 / 3 = 5

Method 2: Use a String as the Sharding Key

Table 20-6 Required calculation methods when the sharding key is the string data type

Condition	Calculation Method	Example
Database sharding key ≠ 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'.hashCode()=96354 Database shard: 96354 % 8 = 2; Table shard: 96354 % 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 **NOTE** The database routing result is rounded off to the nearest integer.	hash('abc') = 'abc'.hashCode()=96354 Table shard: 96354 % (8 x 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:

```
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);
```

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

```
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) tbpartitions 4;
```

Precautions

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.

20.2.4.2 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).
- If you use a sharding key of the string type, do not convert the data type of its value in SQL statements. Changing the data type may cause a failure of routing calculation, and data will be routed to the default shard and cannot be found when you query it.

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 20-7 Required calculation methods when the sharding key is the integer data type

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
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 **NOTE** The database routing result is rounded off to the nearest integer.	Table shard: 16 % (8 x 3) = 16 Database shard: 16 / 3 = 5

Method 2: Use a String as the Sharding Key

Table 20-8 Required calculation methods when the sharding key is the string data type

Condition	Calculation Method	Example
Database sharding key ≠ 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 **NOTE** The database routing result is rounded off to the nearest integer.	hash('abc') = 'abc'.toUpperCase().hashCode()=64578 Table shard: 64578% (8 x 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) tbpartitions 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.

20.2.4.3 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 value is an integer.

Data Routing

The data route depends on the remainder of the new sharding key value divided by the number of database or table shards. 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 20-9 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: (123456 >> 4) % 8 = 4 Table shard: (123456 >> 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 >> 4) % 8 = 4 Table table: ((123456 >> 4) % 8) x 3 + ((123456 >> 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)
tbpartition by RIGHT_SHIFT(id, 4) tbpartitions 2;
```

Precautions

The number of shifts cannot exceed the number of bits occupied by the integer type.

20.2.4.4 MM

Application Scenarios

This algorithm applies if you want to shard data by month. One table shard for one month is recommended, and its name is the month number.

Instructions

- The sharding key must be DATE, DATETIME, or TIMESTAMP.
- This algorithm can be used only for table sharding. It cannot be used for 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 mod 12 = 1.

Calculation Method

Table 20-10 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 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) tbpartitions 12;
```

Precautions

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

20.2.4.5 DD

Application Scenarios

This algorithm applies if you want to shard data by date. One table shard for one day is recommended, and its name is the day number.

Instructions

- The sharding key must be DATE, DATETIME, or TIMESTAMP.
- This algorithm can be used only for table sharding. It cannot be used for 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 mod 31 = 15.

Calculation Method

Table 20-11 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 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) tbpartitions 31;
```

Precautions

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

20.2.4.6 WEEK

Application Scenarios

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

Instructions

- The sharding key must be DATE, DATETIME, or TIMESTAMP.
- This algorithm can be used only for table sharding. It cannot be used for 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.

NOTE

You can run the following SQL statement to query the workday index of a specific date (0 = Monday, 1 = Tuesday, ..., 6 = Sunday):

```
mysql> SELECT WEEKDAY('2019-01-15');
        -> 1
```

If the value returned from the above SQL statement is **1**, the weekday for date 2019-01-15 is Tuesday. Sunday is the first day of the week, so Tuesday is the third day of the week.

Calculation Method

Table 20-12 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) tbpartitions 7;
```


Precautions

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

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

Instructions

- The sharding key must be DATE, DATETIME, or TIMESTAMP.
- This algorithm can be used only for table sharding. It cannot be used for 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 mod 366 = 15.

Calculation Method

Table 20-13 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) tbpartitions 366;
```


Precautions

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

20.2.4.8 YYYYMM

Application Scenarios

This algorithm applies when data is routed to shards by year and month. Recommend you to use this algorithm together with tbpartition YYYYMM(ShardKey).

Instructions

The sharding key must be DATE, DATETIME, or TIMESTAMP.

Data Routing

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

For example, YYYYMM ('2012-12-31 12:12:12') is equivalent to (2012 x 12 + 12) % D. D is the number of database or table shards.

Calculation Method

Table 20-14 Required calculation methods

Condition	Calculation Method	Example
Database sharding key ≠ Table sharding key	Sharding key: yyyy-MM-dd Database routing result = (yyyy x 12 + MM) % Database shards Table routing result = (yyyy x 12 + MM) % Table shards	Sharding key: 2012-11-20 Database shard: (2012 x 12 + 11) % 8 = 3 Table shard: (2012 x 12 + 11) % 3 = 2

Condition	Calculation Method	Example
Database sharding key = Table sharding key	Sharding key: yyyy-MM-dd Table routing result = (yyyy x 12 + MM) % (Database shards x Table shards) Database routing result = Table routing result / Table shards **NOTE** The database routing result is rounded off to the nearest integer.	Sharding key: 2012-11-20 Table shard: (2012 x 12 + 11) % (8 x 3) = 11 Database shard: 11 % 3 = 3

Syntax for Creating Tables

Assume that there are already 8 physical databases in your database instance. Now you want to shard data by year and month and require that data of the same month be stored in one table and each month within two years should correspond to an independent table, so that you can query data from a physical table in a physical database by the sharding key.

In this scenario, you can select the YYYYMM algorithm. Then create 24 physical tables for 24 months of two years, each month corresponding to one table. Since you already have 8 shards, three physical tables should be created in each of them. The following is an example SQL statement for creating a table:

```
create table test_yyyymm_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYMM(create_time)
tbpartition by YYYYMM(create_time) tbpartitions 3;
```

Syntax for creating tables when only database sharding is required:

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

Precautions

- This YYYYMM algorithm does not apply if each month of a year corresponds to one database shard. The number of tables must be fixed if database and table sharding is both required.
- Data of the same month in different years may be routed to the same database or table. The result depends on the number of tables.

20.2.4.9 YYYYDD

Application Scenarios

This algorithm applies when data is routed to shards by year and day. Recommend you to use this algorithm together with tbpartition YYYYDD(ShardKey).

Instructions

The sharding key must be DATE, DATETIME, or TIMESTAMP.

Data Routing

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. The data route depends on the remainder of the hash value divided by the number of database or table shards.

For example, YYYYDD('2012-12-31 12:12:12') is equivalent to (2012 x 366 + 366) % D. D is the number of database or table shards.

NOTE

2012-12-31 is the 366th day of 2012, so the routing result is 2012 x 366 + 366.

Calculation Method

Table 20-15 Required calculation methods

Condition	Calculation Method	Example
Database sharding key ≠ Table sharding key	Sharding key: yyyy-MM-dd Database routing result = (yyyy x 366 + Day of the current year) % Database shards Table routing result = (yyyy x 366 + Day of the current year) % Table shards	Sharding key: 2012-12-31 Database shard: (2012 x 366 + 366) % 8 = 6 Table shard: (2012 x 366 + 366) % 3 = 0
Database sharding key = Table sharding key	Sharding key: yyyy-MM-dd Table routing result = (yyyy x 366 + Day of the current year) % (Database shards x Table shards) Database routing result = Table routing result / Table shards **NOTE** The database routing result is rounded off to the nearest integer.	Sharding key: 2012-12-31 Database shard: (2012 x 366 + 366) % (8 x 3) = 6 Database shard: 6 / 3 = 2

Syntax for Creating Tables

Assume that there are already 8 physical databases in your database instance. Now you want to shard data by year and day and require that data of the same day be stored in one table and each day within two years should correspond to an independent table, so that you can query data from a physical table in a physical database by the sharding key.

In this scenario, you can select the YYYYDD algorithm. Then create at least 732 physical tables for 732 days of the two years (366 days for one year), each day corresponding to one table. Since you already have 8 shards, 92 (732 / 8 = 91.5, rounded up to 92) physical tables should be created in each of them. The number of tables should be an integral multiple of databases. The following is an example SQL statement for creating a table:

```
create table test_yyyydd_tb (
        id int,
        name varchar(30) DEFAULT NULL,
        create_time datetime DEFAULT NULL,
        primary key(id)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    dbpartition by YYYYDD(create_time)
    tbpartition by YYYYDD(create_time) tbpartitions 92;
```

Syntax for creating tables when only database sharding is required:

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

Precautions

- This YYYYDD algorithm does not apply if each day of a year corresponds to one database shard. The number of tables must be fixed if database and table sharding is both required.
- Data of the same day in different years may be routed to the same shard. The result depends on the number of tables.

20.2.4.10 YYYYWEEK

Application Scenarios

This algorithm applies when data is routed to shards by week. Recommend you to use this algorithm together with tbpartition YYYYWEEK(ShardKey).

Instructions

The sharding key must be DATE, DATETIME, or TIMESTAMP.

Data Routing

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. The data route depends on the remainder of the hash value divided by the number of database or table shards.

For example, YYYYWEEK('2012-12-31 12:12:12') is equivalent to (2013 x 54 + 1) % D. D is the number of database or table shards.

NOTE

- 2012-12-31 is the first week of 2013, so the routing result is 2013 x 54 + 1.
- For details on how to use YYYYWEEK, see **YEARWEEK Function**.

Calculation Method

Table 20-16 Required calculation methods

Condition	Calculation Method	Example
Database sharding key ≠ Table sharding key	Sharding key: yyyy-MM-dd Database routing result = (yyyy x 54 + Week of the current year) % Database shards Table routing result = (yyyy x 54 + Week of the current year) % Table shards	Sharding key: 2012-12-31 Database shard: (2013 x 54 + 1) % 8 = 7 Table shard: (2013 x 54 + 1) % 3 = 1
Database sharding key = Table sharding key	Sharding key: yyyy-MM-dd Table routing result = (yyyy x54 + Week of the current year) % (Database shards x Table shards) Database routing result = Table routing result / Table shards **NOTE** The database routing result is rounded off to the nearest integer.	Sharding key: 2012-12-31 Database shard: (2013 x 54 + 1) % (8 x 3) = 7 Database shard: 7 / 3 = 2

Syntax for Creating Tables

Assume that there are already 8 physical databases in your database instance. Now you want to shard data by week and require that data of the same week be stored in one table and each week within two years should correspond to an independent table, so that you can query data from a physical table in a physical database by the sharding key.

In this scenario, you can select the YYYYWEEK algorithm. Then create at least 106 physical tables for 53 (rounded off) weeks of the two years, each week corresponding to one table. Since you already have 8 shards, 14 (14 x 8 = 112 > 106) physical tables should be created in each of them. The number of tables should be an integral multiple of databases. The following is an example SQL statement for creating a table:


```
create table test_yyyymm_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYWEEK(create_time)
tbpartition by YYYYWEEK(create_time) tbpartitions 14;
```

Syntax for creating tables when only database sharding is required:

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

Precautions

- This YYYYWEEK algorithm does not apply if each week of a year corresponds to one database shard. The number of tables must be fixed if database and table sharding is both required.
- Data of the same week in different years may be routed to the same shard.

20.2.4.11 HASH

Application Scenarios

This algorithm features even distribution of data and sharding tables. Arithmetic operators such as equal sign (=) and IN are often used in SQL queries.

Instructions

- The sharding key must be CHAR, VARCHAR, INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, or DECIMAL (the precision can be 0). The sharding key must be DATE, DATETIME, or TIMESTAMP if you use HASH together with date functions.
- If you use a sharding key of the string type, do not convert the data type of its value in SQL statements. Changing the data type may cause a failure of routing calculation, and data will be routed to the default shard and cannot be found when you query it.

Data Routing

Determine the range of each database or table shard using 102400.

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, calculate CRC32 value based on the sharding key value (case sensitive) and divide the CRC value by 102400. Then check which range the remainder belongs to.

Calculation Method

Method 1: Use a Non-date Sharding Key

Table 20-17 Required calculation methods when the sharding key is not the DATE type

Condition	Calculation Method	Example
Non-date sharding key	Database routing result = crc32(Database sharding key) % 102400 Table routing result = crc32(Table sharding key) % 102400	Database/Table shard: 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 20-18 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 number of the current year **NOTE** For the definition of the week number in a year, see **WEEKOFYEAR(*date*)**.	weekofyear('2019-10-11')=41
day()	day(yyyy-MM-dd)=Day number of the current month	day('2019-10-11')=11

Table 20-19 Required calculation methods when the sharding key is the DATE type

Condition	Calculation Method	Example
Date sharding key	Database routing result = crc32(Date function(Database sharding key)) % 102400 Table routing result = crc32(Date function(Table 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)
tbpartition by hash (ID) tbpartitions 4;
```

Precautions

None

20.2.4.12 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

The 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 DATE, DATETIME, or TIMESTAMP.

Data Routing

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

Metadata needs to be set when a table is created. For example, if there are eight shards in one schema, the metadata range can be 1-2=0, 3-4=1, 5-6=2, 7-8=3, 9-10=4, 11-12=5, 13-14=6, and default=7. Data is routed to shards by the sharding key value based on the range.

Calculation Method

Method 1: Use an Integer as the Sharding Key

Table 20-20 Required calculation methods when the sharding key is the integer data type

Condition	Calculation Method	Example
Integer sharding keys	Database routing result: Data is routed to different shards based on the sharding key and the preset metadata range.	Data is routed to shard1 if the sharding key value is 3 and the preset metadata range is 3–4.

Method 2: Use a Date as the Sharding Key

Table 20-21 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 number of the current year **NOTE** For the definition of the week number in a year, see **WEEKOFYEAR(*date*)**.	weekofyear('2019-10-11')=41
day()	day(yyyy-MM-dd)=Day number of the current month	day('2019-10-11')=11

Table 20-22 Calculation methods

Condition	Calculation Method	Example
Date sharding key	Database routing: Data is routed to different database shards based on the date function (database sharding key value) and the preset metadata range.	Data is routed to shard 4 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

None

20.3 DML

20.3.1 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 Constraints

- INSERT DELAYED is not supported.
- Only INSERT statements that contain sharding fields are supported.
- PARTITION syntax is not supported, so partitioned tables are not recommended.
- Setting **YYYY** of **datetime** (in the format of **YYYY-MM-DD HH:MM:SS**) to **1582** or any value smaller in INSERT statements is not supported.
- INSERT cannot be used to insert sharding key value **DEFAULT**.
- If you specify an auto-increment key value in an INSERT statement and execute it on a sharded table, the auto-increment key value of the inserted data entry changes. Auto-increment key values of data entries inserted subsequently will increase based on the first inserted data entry unless you specify a new auto-increment key value.
- Referencing a table column in function REPEAT of the VALUES statement is not supported.

  Example:

  ```
  INSERT INTO T(NAME) VALUES(REPEAT(ID,3));
  ```


- Only constants are supported when you use the INSERT DUPLICATE statement to update the sharding key. Functions and expressions, such as VALUES and LAST_INSERT_ID, are not supported.
- Sharded tables containing GSI cannot be updated using the INSERT DUPLICATE statement.

Use Constraints

- If the sharding key value in the INSERT statement is invalid, data is routed to database shard 0 or table shard 0 by default.
- Do not use functions VERSION, DATABASE, or USER in the INSERT statement. When you execute such functions, you may not obtain the expected results because their results depend on whether the statement is pushed to data nodes for execution.
- Do not use the INSERT DUPLICATE statement to update the sharding key. You may not obtain the expected results because the statement is pushed down to data nodes for execution.

20.3.2 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 Constraints

- 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.

Use Constraints

- If the sharding key value in the REPLACE statement is invalid, data is routed to database shard 0 or table shard 0 by default.
- Do not use functions VERSION, DATABASE, or USER in the REPLACE statement. When you execute such functions, you may not obtain the expected results because their results depend on whether the statement is pushed to data nodes for execution.

20.3.3 DELETE

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

Common Syntax

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

Syntax Constraints

- The WHERE clause does not support subqueries, including correlated and non-correlated subqueries.

- Data in reference tables cannot be deleted when multiple tables are deleted at a time.

20.3.4 UPDATE

Common Syntax

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

Syntax Constraints

- Subqueries are not supported, including correlated and non-correlated subqueries.
- The WHERE condition in the UPDATE statement does not support arithmetic expressions and their subqueries.
- Modifying broadcast tables is not supported during an update of multiple tables. Do not specify a column for a broadcast table in the left part of a SET statement.
- Updating the sharding key field of a logical table is not supported because this operation may cause data redistribution.
- Setting **YYYY** of **datetime** (in the format of **YYYY-MM-DD HH:MM:SS**) to **1582** or any value smaller in UPDATE statements is not supported.
- UPDATE cannot be used to update sharding key value **DEFAULT**.
- Repeatedly updating the same field in an UPDATE statement is not supported.
- Updating a sharding key using UPDATE JOIN syntax is not supported.
- Updating sharding keys in subqueries is not allowed for secondary sharded tables that contain JSON fields.
- UPDATE cannot be used to update self-joins.
- Referencing other target columns in assignment statements or expressions may cause unexpected update results.

  Example:

  ```
  update tbl_1 a,tbl_2 b set a.name=concat(b.name,'aaaa'),b.name=concat(a.name,'bbbb')
  on a.id=b.id
  ```

- UPDATE JOIN supports only joins with WHERE conditions.

20.3.5 SELECT

SELECT is generally 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 20-23 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 after the GROUP BY clause and before the ORDER BY clause.
ORDER BY	Indicates relationships between statements. Sorting by column name or by a specified order such as ASC and DESC is supported.
LIMIT/OFFSET	Restrains the offset and size of output result sets, for example, one or two values can be input after LIMIT.

Syntax Description

- An empty string cannot be used as an alias.
- The SELECT... GROUP BY... WITH ROLLUP query statement is not supported. (When the queried table is a sharded table, you cannot obtain the expected result)
- 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.
- The field sequence after the SELECT statement must be the same as that in the table column. All JOIN fields can be added to the field list after SELECT to improve the SELECT query efficiency.
- DDM does not support multiple columns with the same name for each SELECT statement in UNION. For example, duplicate column names are used in the following SELECT statement:
  ```
  SELECT id, id, name FROM t1 UNION SELECT pk, pk, name FROM t2;
  ```

20.3.6 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 is not supported.

NATURAL JOIN is 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;
```

20.3.7 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 duplicate column names.

20.3.8 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 Constraints

- 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.

20.3.9 Unsupported DML Statements

Unsupported DML Statements

Table 20-24 Syntax restrictions on DML

DML Syntax	Constraint
DELETE statement	PARTITION clauses are not supported.
UPDATE statement	Cross-shard subquery is not supported.
SELECT statement	ORDER BY statement. User-defined sequencing similar to **ORDER BY FIELD(id,1,2,3)** is not supported. **NOTE** When the queried table is a sharded table, you cannot obtain the expected result

20.3.10 Supported System Schema Queries

Table 20-25 Supported System Schema Queries

DML Syntax	Constraint
System schema queries	The following system schema queries are supported: Version query: **SELECT version()** • information_schema.SCHEMA_PRIVILEGES • information_schema.TABLE_PRIVILEGES • information_schema.USER_PRIVILEGES • information_schema.SCHEMATA • information_schema.tables • information_schema.columns Index query: **SHOW KEYS FROM <table> FROM <database>** **NOTE** • 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 **>**.

20.4 Online DDL

DDM supports online DDL operations, including adding, deleting, or modifying fields, setting default values, and modifying character sets and table names.

Online DDL extended syntax provides an explicit statement about what algorithm and lock will be used and can transparently transmit the statement to data nodes. This function is available only to DDM 3.1.0 or later.

If your DDM instance is associated with a MySQL 5.7 instance, Online DDL supports the following syntax:

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...]

alter_option: {
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | ADD [COLUMN] (col_name column_definition,...)
  | DROP [COLUMN] col_name
  | ALTER [COLUMN] col_name {
        SET DEFAULT {literal | (expr)}
      | DROP DEFAULT
    }
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST | AFTER col_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | RENAME [TO | AS] new_tbl_name

  | ALGORITHM [=] {DEFAULT | INPLACE | COPY}
  | LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
}
```

If your DDM instance is associated with a MySQL 8.0 instance, Online DDL supports the following syntax:

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...]

alter_option: {
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | ADD [COLUMN] (col_name column_definition,...)
  | DROP [COLUMN] col_name
  | ALTER [COLUMN] col_name {
        SET DEFAULT {literal | (expr)}
      | DROP DEFAULT
    }
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST | AFTER col_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | RENAME COLUMN old_col_name TO new_col_name
  | RENAME [TO | AS] new_tbl_name

  | ALGORITHM [=] {DEFAULT | INSTANT | INPLACE | COPY}
  | LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
}
```


> **CAUTION**
>
> If you use Online DDL syntax, the selected algorithm and lock will take effect on your data node. If there are multiple data nodes, DDM may not handle concurrent requests as specified by parameters.

Syntax Examples

Adding a column

```
# Add column x of the INT type to table t2 using the in-place algorithm and lock NONE.
ALTER TABLE t2 ADD COLUMN x INT, ALGORITHM=INPLACE, LOCK=NONE;
```

Modifying a column

```
# Modify the data type of column x to VARCHAR(255) for table t2 using the copy algorithm and a shared lock.
ALTER TABLE t2 MODIFY x VARCHAR(255), ALGORITHM=COPY, LOCK=SHARED;
```

Modifying a character set

```
# Change the character set and collation of table t2 to utf8 and utf8_bin using the copy algorithm and a shared lock.
ALTER TABLE t2 CHARACTER SET utf8 COLLATE utf8_bin, ALGORITHM=COPY, LOCK=SHARED;
```

20.5 Functions

Supported Functions

Table 20-26 Operator functions

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

Function Expression	Example
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('test') ORDER BY vendor_name;
IFNULL	SELECT IFNULL(product_id, 0) FROM Products;

Table 20-27 Time and date functions

Function Expression	Example	Application Scope
DAY()	SELECT * FROM TAB_DATE WHERE DAY(date)=21; SELECT * FROM TAB_DATE WHERE date='2018-12-21'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22');	-
MONTH()	SELECT * FROM TAB_DATE WHERE MONTH(date)=12; SELECT * FROM TAB_DATE WHERE date='2018-12-21'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22');	-
YEAR()	SELECT * FROM TAB_DATE WHERE YEAR(date)=2018; SELECT * FROM TAB_DATE WHERE date='2018-12-21'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22');	-
TIME()	SELECT * FROM TAB_DATE WHERE TIME(date)='01:02:03'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid time or datetime expression or character string.
TIME_TO_SEC()	SELECT * FROM TAB_DATE WHERE TIME_TO_SEC(date)=3603; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:00:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:00:03');	The parameter must be a valid time or datetime expression or character string.
SEC_TO_TIME()	SELECT * FROM TAB_DATE WHERE SEC_TO_TIME(date)='00:01:00'; SELECT * FROM TAB_DATE WHERE date=60; INSERT INTO TAB_DATE(id,date) VALUES(1,60);	The parameter must be a numerical value or a character string that can be converted into a numerical value.

Function Expression	Example	Application Scope
SECOND()	SELECT * FROM TAB_DATE WHERE SECOND(date)=3; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid time or datetime expression or character string.
MINUTE()	SELECT * FROM TAB_DATE WHERE MINUTE(date)=2; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid time or datetime expression or character string.
HOUR()	SELECT * FROM TAB_DATE WHERE HOUR(date)=1; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid time or datetime expression or character string.
DAYNAME()	SELECT * FROM TAB_DATE WHERE DAYNAME(date)='Friday'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid date or datetime expression or character string.
MONTHNAME()	SELECT * FROM TAB_DATE WHERE MONTHNAME(date)='January'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid date or datetime expression or character string.
LAST_DAY()	SELECT * FROM TAB_DATE WHERE LAST_DAY(date)='2021-01-31'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid date or datetime expression or character string.

Function Expression	Example	Application Scope
DAYOFWEEK()	SELECT * FROM TAB_DATE WHERE DAYOFWEEK(date)=6; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid date or datetime expression or character string.
DAYOFMONTH()	SELECT * FROM TAB_DATE WHERE DAYOFWEEK(date)=6; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	The parameter must be a valid date or datetime expression or character string.
DAYOFYEAR()	SELECT * FROM TAB_DATE WHERE DAYOFYEAR(date)=365; SELECT * FROM TAB_DATE WHERE date='2021-12-31 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-12-31 01:02:03');	The parameter must be a valid date or datetime expression or character string.
WEEKOFYEAR()	SELECT * FROM TAB_DATE WHERE WEEKOFYEAR(date)=53; SELECT * FROM TAB_DATE WHERE date='2021-12-31 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-12-31 01:02:03');	The parameter must be a valid date or datetime expression or character string.

Function Expression	Example	Application Scope
DATE_ADD(date, INTERVAL expr unit)	SELECT * FROM TAB_DATE WHERE DATE_ADD(date,INTERVAL 1 YEAR)='2022-01-01 01:02:03'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	• Parameter **date** must be a valid time, date or datetime expression or character string. • **expr** indicates the date interval. This parameter must be an integer or a character string that can be converted into an integer. • **unit** indicates the time unit. The value can be **SECOND**, **MINUTE**, **HOUR**, **DAY**, **WEEK**, **MONTH**, **QUARTER**, or **YEAR**. • If **date** indicates a date, its lower boundary value is **1000-01-01**. If the value is out of the range, an error may be reported. • The **date** parameter can be precise to milliseconds.

Function Expression	Example	Application Scope
DATE_SUB(date, INTERVAL expr unit)	SELECT * FROM TAB_DATE WHERE DATE_SUB(date,INTERVAL -1 DAY)='2021-01-02 01:02:03'; SELECT * FROM TAB_DATE WHERE date='2021-01-02 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-02 01:02:03');	• Parameter **date** must be a valid time, date or datetime expression or character string. • **expr** indicates the date interval. This parameter must be an integer or a character string that can be converted into an integer. • **unit** indicates the time unit. The value can be **SECOND**, **MINUTE**, **HOUR**, **DAY**, **WEEK**, **MONTH**, **QUARTER**, or **YEAR**. • If **date** indicates a date, its lower boundary value is **1000-01-01**. If the value is out of the range, an error may be reported. • The **date** parameter can be precise to milliseconds.

CAUTION

When the time zones of a DDM instance and a data node are different, if a time function is used in SQL statements and is pushed down, the time of the time zone where the data node is located is returned. If the time function is calculated on DDM, the time zone of the DDM instance is returned.

Table 20-28 Mathematical functions

Function Expression	Example	Application Scope
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;	-
ROUND()	SELECT ROUND(product_price) AS round_product FROM Products;	The parameter must be a numerical value or a character string that can be converted into a numerical value.
SIN()	SELECT SIN(x) AS sin_x FROM math_tbl;	The parameter must be a numerical value or a character string that can be converted into a numerical value.
COS()	SELECT COS(x) AS cos_x FROM math_tbl;	The parameter must be a numerical value or a character string that can be converted into a numerical value.
TAN()	SELECT TAN(x) AS tan_x FROM math_tbl;	The parameter must be a numerical value or a character string that can be converted into a numerical value.

Function Expression	Example	Application Scope
COT()	SELECT COT(x) AS cot_x FROM math_tbl;	The parameter must be a numerical value or a character string that can be converted into a numerical value.
FLOOR()	SELECT FLOOR(product_price) AS floor_product FROM Products;	The parameter must be a numerical value or a character string that can be converted into a numerical value.
CEILING()	SELECT CEILING(product_price) AS ceiling_product FROM Products;	The parameter must be a numerical value or a character string that can be converted into a numerical value.
ABS()	SELECT ABS(x) AS abs_x FROM math_tbl;	The value ranges from **-9223372036854775807** to **9223372036854775807**. If the value is out of the range, an error is reported.
LOG()	SELECT LOG(x) AS log_x FROM math_tbl;	The parameter must be a numerical value greater than 0 or a character string that can be converted into a numerical value.
LN()	SELECT LN(x) AS ln_x FROM math_tbl;	The parameter must be a numerical value greater than 0 or a character string that can be converted into a numerical value.
EXP()	SELECT EXP(x) AS exp_x FROM math_tbl;	The value ranges from **-∞** to **709**. If the value is out of the range, an error is reported.

Table 20-29 Character string functions

Function Expression	Example	Application Scope
TRIM()	SELECT TRIM(' hello, world ') AS trim_character FROM Character;	The parameter must be a character string or of a similar data type.

NOTE

You are advised to use a supported function. Before using a function, check whether it is supported. If you use an unsupported function, the returned result may be different from that in MySQL.

Unsupported Functions

Table 20-30 Function restrictions

Item	Restriction
ROW_COUNT()	Function **ROW_COUNT()** is not supported.
COMPRESS()	Function **COMPRESS()** is not supported.
SHA()	Function **SHA()** is not supported.
SHA1()	Function **SHA1()** is not supported.
MD5()	Function **MD5()** is not supported.
AES_ENCRYPT()	Function **AES_ENCRYPT()** is not supported.
AES_DECRYPT()	Function **AES_DECRYPT()** is not supported.
YEARWEEK()	Function **YEARWEEK()** is not supported.
TIME_FORMAT()	Function **TIME_FORMAT()** is not supported. If you are not sure whether the function can be pushed down to RDS, use function **DATE_FORMAT()**.

20.6 Unsupported Objects and Use Constraints

- 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 (<), equal sign (=), and greater than (>)
- Expression IS UNKNOWN
- INSTALL and UNINSTALL PLUGIN statements
- Cross-shard stored procedures and custom functions
- Modifying database names and sharding field names and types is not allowed.
- Most of SHOW statements such as SHOW PROFILES and SHOW ERRORS
- Table maintenance statements, including CHECK, CHECKSUM, OPTIMIZE, and REPAIR TABLE
- Statements for assigning a value to or querying variable **session**

  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
- The result of the REPEAT function contains a maximum of 1,000,000 characters (in version 3.0.9 or later).

Permission Levels

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

20.7 Supported SQL Statements

20.7.1 CHECK TABLE

20.7.1.1 Checking DDL Consistency of Physical Tables in All Logical Tables

Purpose: To check 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 contains 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 physical tables corresponding to the logical table are consistent.
- **TOTAL_COUNT**: indicates the number of physical tables in the logical table.
- **INCONSISTENT_COUNT**: indicates the number of physical tables with inconsistent DDL results.
- **DETAILS**: indicates names of the physical tables with inconsistent DDL check results.

20.7.1.2 Checking DDL Consistency of All Physical Tables Corresponding to One Logical Table

Purpose: To check DDL consistency of all physical tables corresponding to a specific logical table

Command Format:

check table *<table_name>*

Command Output:

If the returned result set is empty, DDL results of all physical tables corresponding to this logical table are consistent.


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

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

- **DATABASE_NAME**: indicates the database shard containing the physical table.
- **TABLE_NAME**: indicates the name of the physical table.
- **TABLE_TYPE**: indicates the type of the logical table that the physical table belongs to.
- **EXTRA_COLUMNS**: indicates extra columns in the physical table.
- **MISSING_COLUMNS**: indicates missing columns in the physical table.
- **DIFFERENT_COLUMNS**: indicates name and type columns whose attributes are inconsistent in the physical table.

- **KEY_DIFF**: indicates inconsistent indexes in the physical table.
- **ENGINE_DIFF**: indicates inconsistent engines in the physical table.
- **CHARSET_DIFF**: indicates inconsistent character sets in the physical table.
- **COLLATE_DIFF**: indicates inconsistent collations in the physical table.
- **EXTRA_PARTITIONS**: indicates extra partitions in the physical table. This field is only available to partitioned tables.
- **MISSING_PARTITIONS**: indicates missing partitions in the physical table. This field is only available to partitioned tables.
- **DIFFERENT_PARTITIONS**: indicates partitions with inconsistent attributes in the physical table. This field is only available to partitioned tables.
- **EXTRA_INFO**: indicates other information such as missing physical tables.

20.7.2 SHOW RULE

Command Format:

- It is used to view the sharding rule of each logical table in a certain schema.

  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 | test6                     | 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-
40=3, 41-50=4, 51-60=5, 61-70=6, 71-120=7, default=3 |
|   13 | mhash_bigint_dpt_1        | 0         | bigint_col       | mod_hash            |                  1 |                   8 |                     |
|   14 | hash_id_dpt_1             | 0         | id_col           | hash                |                  1 |                   8 |                     |
|   15 | mhash_varchar_dpt_1       | 0         | varchar_col      | mod_hash            |                    |                   8 |                     |
```

- It is used to view the sharding rule of a specific logical table in a certain schema.

  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               |                  8 |                     |  0-10=0, 11-20=1, 21-30=2, 31-40=3, 41-
50=4, 51-60=5, 61-70=6, 71-120=7, default=3 | datetime_col     | yyyydd              |                  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 shard 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 in each database shard.

TB_PARTITION_OFFSET: indicates where a new physical table starts from.

20.7.3 SHOW TOPOLOGY

Command Format:

It is used to view physical tables corresponding to a specified logical table.

show topology from *<table_name>*

Output Details:

Rds_instance_id: indicates the ID of the RDS instance.

HOST: indicates the IP address of the RDS instance.

PORT: indicates the port number of the RDS instance.

DATABASE: indicates the physical database in the RDS instance.

TABLE: indicates the physical table.

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

20.7.4 SHOW DATA NODE

Command Format:

show data node

It is used to view data about database shards in the RDS instance.

Output Details:

RDS_INSTANCE_ID: indicates the ID of the RDS instance.

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

HOST: indicates the IP address of the RDS instance.

PORT: indicates the port number of the RDS instance.

20.7.5 TRUNCATE TABLE

20.7.5.1 HINT-DB

Command Format:

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

Description:

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

20.7.5.2 HINT-TABLE

Command Format:

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

Description:

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

Example output before the table is deleted:

```
mysql> show topology from user_tb;
+-----------------+-----------+-------+-----------+-----------+-----------+
| Rds_instance_id | Host      | Port  | Database  | Table     | Row_count |
+-----------------+-----------+-------+-----------+-----------+-----------+
| shard1          | localhost | 33061 | test_0000 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_0 | 2         |
| shard1          | localhost | 33061 | test_0001 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_1 | 3         |
| shard1          | localhost | 33061 | test_0002 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_1 | 5         |
| shard1          | localhost | 33061 | test_0003 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_2 | 3         |
| shard3          | localhost | 33063 | test_0005 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_3 | 2         |
| shard3          | localhost | 33063 | test_0007 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_3 | 0         |
+-----------------+-----------+-------+-----------+-----------+-----------+
32 rows in set (0.22 sec)

mysql> /*+table = user_tb_3*/truncate table user_tb;
Query OK, 0 rows affected (5.18 sec)
```

Example output after the table is deleted:


```
mysql> show topology from user_tb;
+-----------------+-----------+-------+-----------+-----------+-----------+
| Rds_instance_id | Host      | Port  | Database  | Table     | Row_count |
+-----------------+-----------+-------+-----------+-----------+-----------+
| shard1          | localhost | 33061 | test_0000 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_0 | 2         |
| shard1          | localhost | 33061 | test_0001 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_1 | 3         |
| shard1          | localhost | 33061 | test_0002 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_1 | 5         |
| shard1          | localhost | 33061 | test_0003 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_2 | 3         |
| shard3          | localhost | 33063 | test_0005 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_3 | 0         |
+-----------------+-----------+-------+-----------+-----------+-----------+
32 rows in set (0.16 sec)
```

20.7.5.3 HINT-DB/TABLE

Command Format:

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

Description:

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

20.7.6 HINT- ALLOW_ALTER_RERUN

Command Format:

/*+ allow_alter_rerun=true*/*<ALTER TABLE>***

Description:

Using this hint ensures that commands can be repeatedly executed, and no errors are 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.

Example:

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

20.7.7 LOAD DATA

Standard Example

LOAD DATA LOCAL INFILE '/data/data.txt' IGNORE INTO TABLE test CHARACTER SET 'utf8' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n' (id, sid, asf);

NOTE

If a data field contains special characters like separators and escapes, execute OPTIONALLY ENCLOSED BY '"' to enclose the field with double quotation marks ("").

Example:

The following data field contains separators (,) and is enclosed with quotation marks:

"aab,,,bba,ddd"

If the preceding method does not work, add a backslash (\) before each quotation mark (") in the field. For example, **"aab,,,bba,ddd\"ddd\"bb,ae"**

- 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 that should be the same as the code used by physical databases in the target RDS for MySQL 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 for MySQL 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/data.txt' IGNORE INTO TABLE test CHARACTER SET 'utf8' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n' (id, sid, asf);
   mysql> commit;
   ```

Use Constraints

- LOW_PRIORITY is not supported.
- CONCURRENT is not supported.
- PARTITION (partition_name [, partition_name] ...) is not supported.
- LINES STARTING BY 'string' is not supported.
- User-defined variables are not supported.
- ESCAPED BY supports only '\\'.
- If you have not specified a value for your auto-increment key when you insert a data record, DDM will not fill a value for the key. The auto-increment keys of data nodes of a DDM instance all take effect, so the auto-increment key values may be duplicate.
- If the primary key or unique index is not routed to the same physical table, REPLACE does not take effect.
- If the primary key or unique index is not routed to the same physical table, IGNORE does not take effect.
- Executing LOAD DATA statements is not allowed on the tables that contain global secondary indexes.

20.7.8 SHOW PHYSICAL PROCESSLIST

- Command 1: returns all processes that run on the associated RDS instance.

  show physical processlist

- Command 2: filters out the data records whose **info** is empty from result sets of command 1 and returns only the data records whose **info** is not empty.

  show physical processlist with info

  Command Output:

Figure 20-1 Command execution effect

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

Output Details:

IP: indicates the IP address of the RDS instance.

Port: indicates the port number of the RDS instance.

Instance_id: indicates the ID of the RDS instance.

Type:master: indicates that the RDS instance is a primary instance, and **readreplica** indicates that the RDS instance is a read replica.

Columns after column **Type** provide information about processes running on the associated RDS instance. Such information is the same as the output of command **show processlist** when you execute it on the associated RDS instance.

- Command 3: kills the execution thread on the associated RDS instance:

  kill physical *<physical_thread_id>@<rds_ip>:<rds_port>*

  <physical_thread_id>: indicates the ID of the execution thread on the associated RDS instance. You can obtain it from result sets of command 2.

  <rds_ip>: indicates the IP address of the associated RDS instance. You can obtain it from result sets of command 2.

  <rds_port>: indicates the port number of the associated RDS instance. You can obtain it from result sets of command 2.

NOTICE

- This feature is available only in kernel 3.0.1 or later.
- You need to log in to the DDM instance and run the preceding commands on it.

20.7.9 Customized Hints for Read/Write Splitting

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

The following hint formats are supported:

Format 1:

/*!mycat:db_type=*host* ***/**

Format 2:

/*+ db_type=*host* ***/**

host can be **master** or **slave**. **master** indicates a primary instance, and **slave** indicates a read replica.

Currently, this function only applies to SELECT statements.

NOTE

After read/write splitting is enabled, write operations are performed only on the primary DB instance, and read operations are performed only on its read replicas. To read from the primary instance, you can customize a hint to forcibly perform read operations on the primary instance. This method is only suitable for queries.

20.7.10 Setting a Hint to Skip the Cached Execution Plan

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

The hint is in the following format:

/*!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.

20.7.11 Specifying a Shard Using a Hint When You Execute a SQL Statement

DDM allows you to customize a hint to specify whether SQL statements are executed on one shard or multiple shards.

The following hint formats are supported:

- SQL statement executed on one shard: **/*+db=***<physical_db_name>****/** *<your query>***;**
- SQL statement executed on multiple shards: **/*+db=***{<physical_db_name1>, <physical_db_name2>,<physical_db_name3>.....}****/** *<your query>***;**

Example:

- SQL statement executed on one shard: **/*+db=test_0000*/ select * from t1;**
- SQL statement executed on multiple shards: **/*+db={test_0001, test_0002, test_0003}*/ select * from t2;**

Constraints:

- If the SQL statement is executed on multiple shards, the value of **physical_db_name** must be unique.
- The hint is valid only for SELECT, DML, and TRUNCATE statements.
- The hint works only under the text protocol, rather than the Prepare protocol.

20.8 Global Sequence

20.8.1 Overview

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.
- If the auto-increment sequence is used, its value must be **null**. If you do not specify the value or set it to **null**, DDM will assign a value by default. The user-defined value may conflict with the assigned auto-increment key value.

Table 20-31 Table types supported by global sequence

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

Creating an Auto-Increment Sequence

Step 1 Connect to a DDM instance.

Step 2 Open the required schema.

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

create sequence *<sequence name >*

NOTE

- The auto-increment key should be a BIGINT value. To avoid duplicate values, do not use TINYINT, SMALLINT, MEDIUMINT, INTEGER, or INT as the auto-increment key.
- Run **show sequences** to view the usage of the auto-increment sequence. If the usage reaches or approaches 100%, do not insert data. Contact DDM technical support personnel.

----End

Dropping an Auto-Increment Sequence

Step 1 Connect to a DDM instance.

Step 2 Open the required schema.

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

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

drop sequence *<sequence name >*

drop sequence *DB.****;

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 Sequence

Step 1 Connect to a DDM instance.

Step 2 Open the required schema.

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

Step 4 Run the command to change the start value:

alter sequence *<sequence name >* **START WITH** *<start value of the target sequence >*

----End

Querying an Auto-Increment Sequence

Step 1 Connect to a DDM instance.

Step 2 Log in to the target schema.

Step 3 Run the following command to view all sequences:

show sequences;

```
mysql> show sequences;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    82944
Current database: test

+-----------------------------------------+------------+-----------+---------------------+-------------------+
| NAME                                    | START_WITH | INCREMENT | MAX_VALUE           | USAGE_PERCENT(%)  |
+-----------------------------------------+------------+-----------+---------------------+-------------------+
| TEST.AATP_ALLOC_PATH_PRODUCT_T          | 871001     | 1000      | 9223372036854775807 | 0.00              |
| TEST.BLOB_TEST                          | 1          | 1000      | 2147483647          | 0.00              |
| TEST.BROADCAST_TABLE_1                  | 1001       | 1000      | 2147483647          | 0.00              |
| TEST.COMMODITY                          | 1          | 1000      | 2147483647          | 0.00              |
| TEST.DCR_CSS_DC_SKU_T                   | 1          | 10000     | 9223372036854775807 | 0.00              |
| TEST.DCR_CSS_DC_SKU_TI                  | 14810001   | 10000     | 9223372036854775807 | 0.00              |
| TEST.DCR_DELIVER_PLAN_REVIEW_DTL_T      | 247804001  | 1000      | 9223372036854775807 | 0.00              |
| TEST.DCR_DELIVER_PLAN_REVIEW_T          | 973001     | 1000      | 9223372036854775807 | 0.00              |
| TEST.DCR_PRODUCT_CONFIG_T               | 29371156   | 1000      | 9223372036854775807 | 0.00              |
| TEST.DCR_STORE_CONFIG_T                 | 617600     | 1000      | 9223372036854775807 | 0.00              |
| TEST.DF_ENTITY_PRODUCT_CONFIG_T         | 844001     | 1000      | 9223372036854775807 | 0.00              |
```

----End

Modifying the Auto-Increment Cache Value

NOTICE

This feature is only available in kernel 3.0.3 and later versions.

Step 1 Connect to a DDM instance.

Step 2 Log in to the required schema.

Step 3 Run the following command to change cached values of the global sequence of table **test**:

alter sequence test cache 5000

Step 4 Run the following command to view cached values (or **INCREMENT** values) of the global sequence of table **test**:

show sequences

----End

Updating Auto-Increment Sequences of All Tables

NOTICE

This feature is available only in kernel 3.0.4.1 or later.

Step 1 Connect to a DDM instance.

Step 2 Run the following command to change sequences of all schemas:

fresh all sequence start value

----End

20.8.2 Using NEXTVAL or 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 a DDM instance.

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

20.8.3 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.

Prerequisites

- There are two schemas **dml_test_1** and **dml_test_2**.
- Both of them have table **test_seq**.

  Run the following command to create a table:

  create table test_seq(col1 bigint,col2 bigint) dbpartition by hash(col1);

Procedure

Step 1 Connect to a DDM instance.

Step 2 Log in to schema **dml_test_1**.

use dml_test_1;

Step 3 Run the following command to create a global sequence:

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(col1 bigint,col2 bigint) dbpartition by hash(col1);
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:

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

```
mysql> insert into test_seq(col1,col2)values(seq_test.nextval,seq_test.currval);
Query OK, 1 row affected (0.31 sec)

mysql> select * from test_seq;
+------+------+
| col1 | col2 |
+------+------+
|    1 |    1 |
+------+------+
1 row in set (0.05 sec)
```

Step 5 Log in to schema **dml_test_2**.

use dml_test_2;

Step 6 Run the following statement to use the global sequence in an INSERT or REPLACE statement:

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(col1 bigint,col2 bigint) dbpartition by hash(col1);
Query OK, 0 rows affected (0.52 sec)

mysql> insert into test_seq(col1,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;
+------+------+
| col1 | 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 a schema name, for example, **dml_test_1.seq_test.nextval** or **dml_test_1.seq_test.currval**.

NOTE

- Using global sequences in INSERT and REPLACE statements is supported only in sharded tables, but not in 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

20.9 Database Management Syntax

Supported Database Management Syntax

- SHOW COLUMNS
- SHOW CREATE TABLE
- SHOW TABLE STATUS
- SHOW TABLES
- SHOW DATABASES

  NOTE

  If the required database is not found, check fine-grained permissions of your account.
- SHOW INDEX FROM
- SHOW VARIABLES

Supported Database Tool Commands

- DESC
- USE
- EXPLAIN

  NOTE

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

Unsupported Database Management Syntax

- Executing SET to modify global variables
- SHOW TRIGGERS statements are not supported.
- CHECK TABLE does not support sharding tables by hash or sharding key.
- SHOW WARNINGS and SHOW ERRORS do not support the combination of LIMIT and COUNT. The SHOW statements are randomly sent to a database shard. If database shards are on different RDS for MySQL instances, the returned variables or table information may be different.

20.10 Advanced SQL Functions

- 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

  EXECUTE
- Comments for indexes are not supported in table creation statements.

21 FAQs

21.1 General Questions

21.1.1 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 data nodes, but not on DDM.
- Configuration information of schemas and logical tables is stored in DDM databases. Primary and standby DDM databases are highly available.

High Availability

DDM is deployed using multiple stateless nodes in cluster mode and provides services through the IP address bound to your load balancer.

- If one DDM node becomes faulty, an error is returned for connections established on the node, without affecting the DDM cluster. The faulty node is generally deleted from the cluster within 5 seconds.
- If a data node becomes faulty, services can be restored within 30 seconds after the data node is recovered.

21.1.2 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 data nodes.

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

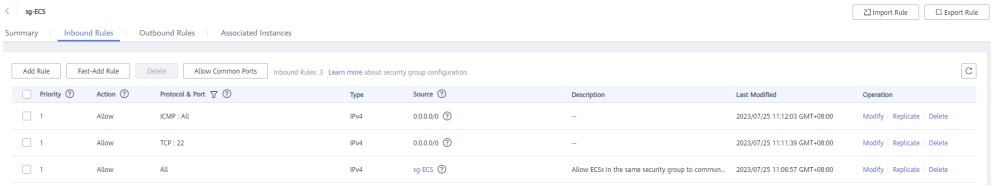
1. Using the same security group is recommended for the ECS, DDM instance, and data nodes. After a security group is created, network access in the group is not restricted by default.
2. If different security groups are configured, you may need to refer to the following configurations:

   NOTE

   - Assume that the ECS, DDM instance, and RDS for MySQL 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 for MySQL instance is **3306**.
   - The remote end should be a security group or an IP address.

   Add the rules described in the following figure to the security group of the ECS to ensure that your client can access the DDM instance.

Figure 21-1 ECS security group rules



Add the rules shown in the following figure to the security group of the ECS where your DDM instance is located so that your DDM instance can access associated data nodes and can be accessed by your client.

Figure 21-2 Configuring security group inbound rules for your DDM instance

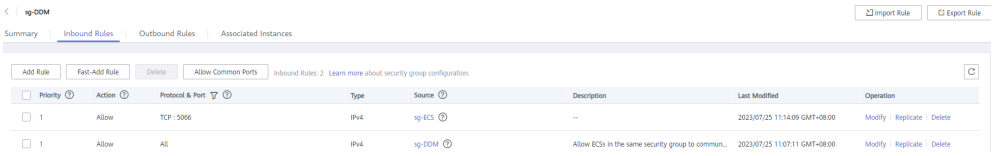
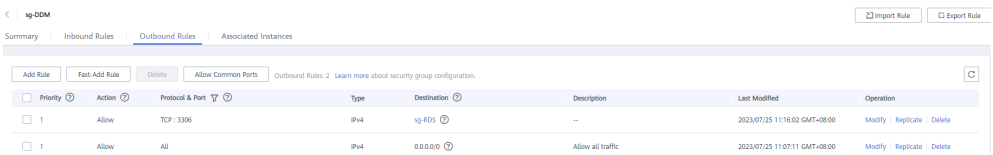


Figure 21-3 Configuring security group outbound rules for your DDM instance



Add the rules shown in the following figure to the security group of the ECS where the data node is located so that your DDM instance can access the node.

Figure 21-4 Configuring security group inbound rules for the associated RDS instance

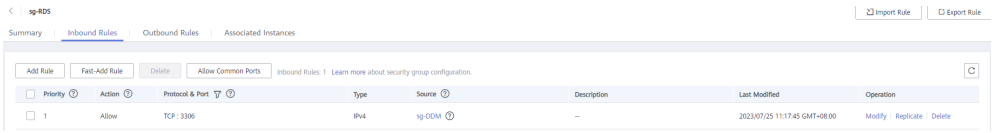
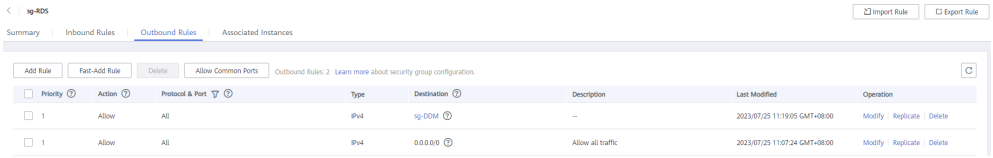


Figure 21-5 Configuring security group outbound rules for the associated RDS instance



21.1.3 Can Data Nodes Associated with a DDM Instance Share Data?

No. Different data nodes associated with a DDM instance are independent of each other and cannot share data.

21.1.4 What Data Nodes Can Be Associated with a DDM Instance?

Any data nodes can be associated with a DDM instance as long as they are:

- Running normally
- In the same VPC as the DDM instance
- Not in use by other DDM instances

21.2 DDM Usage

21.2.1 How Does DDM Perform Sharding?

Distributed databases use shard-based storage, which removes capacity bottlenecks of single-node databases caused by a large amount of data in one unsharded table. Therefore, when creating a schema and logical tables, you need to consider your actual conditions and determine whether to create sharded tables and which sharding rule should be used.

NOTE

Avoid cross-shard JOIN operations on the data that is stored in different shards, to ensure optimal performance and resource availability.

- Whether logical tables are sharded

  DDM supports three types of logical tables: broadcast, sharded, and unsharded. You can select the most appropriate type based on your specific needs.

  – Unsharded: One physical table is created and stores data only in the first shard.

  - Broadcast: The same physical table is created in each shard and stores the same data.
  - Sharded: The same physical table is created in each shard, and data is distributed to all shards based on a certain sharding rule.
- Sharding rule of logical tables

  The selection of a sharding key is important for each logical table. Selecting the sharding key based on your needs is recommended. If an entity relationship exists between different logical tables, select the same field as the sharding key to avoid cross-shard JOIN.

Pay attention to the following suggestions before determining whether to use sharding:

- Do not shard tables that each have less than 10 million data records.
- Shard the tables that each have more than 10 million data records. Storing data in different sharded tables removes performance bottlenecks caused by a large amount of data in one table, while also improving concurrency capability. Select an appropriate sharding key in advance.
- Avoid across-table JOIN operations during service reading and cross-shard operations for any individual transaction.
- Include the sharding key into query conditions to avoid scanning table shards in all sharded tables.

21.2.2 What Do I Do If I Fail to Connect to a DDM Instance Using the JDBC Driver?

When you access a DDM instance 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.writeStringNoNull(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 the DDM instance.

```
//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.

21.2.3 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, so exporting data from MySQL is slow.

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

21.2.4 What Do I Do If a Duplicate Primary Key Error Occurs When Data Is Imported into DDM?

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

21.2.5 What Should I Do If an Error Message Is Returned When I Specify an Auto-Increment Primary Key During Migration?

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 Database name.SEQ name START WITH New start value
```

21.2.6 What Do I Do If an Error Is 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.

21.2.7 Which Should I Delete First, a Schema or its Associated RDS Instances?

After an RDS instance is associated with your schema, you cannot delete the instance directly. To delete it, you have to delete the schema first and then delete the instance.

21.2.8 Can I Manually Delete Databases and Accounts Remained in Data Nodes After a Schema Is Deleted?

You can delete DDM instances or accounts if they are no longer needed.

21.3 SQL Syntax

21.3.1 Does DDM Support Distributed JOINs?

Yes. DDM supports distributed JOINs.

- 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-schema update or deletion of multiple tables.

21.3.2 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.

21.3.3 Does DDM Support Forced Conversion of Data Types?

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

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

Solution

Split an INSERT statement into multiple small statements.

21.4 RDS-related Questions

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

DDM is case-insensitive to database names, table names, column names, and column values by default.

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

Table 21-1 lists risky operations on RDS for MySQL.

Table 21-1 Risky operations on RDS for MySQL

Operation Type	Operation	Impact of the Operation
Operations on the RDS for MySQL console	Deleting an RDS for MySQL instance	After an RDS for MySQL instance is deleted, all schemas and logical tables of the DDM instance associated with the RDS instance become unavailable.
	Performing the primary/standby switchover of an RDS for MySQL instance	RDS for MySQL 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. • Creating schemas or logical tables is not allowed on DDM during the primary/standby switchover of the RDS for MySQL instance. • After a primary/standby switchover of an RDS for MySQL instance, the RDS instance ID remains unchanged in DDM.

Operation Type	Operation	Impact of the Operation
	Restarting an RDS for MySQL instance	The restart of an RDS for MySQL instance makes itself unavailable and will also affect the associated DDM instance.
	Resetting a password	After the RDS for MySQL instance password is reset, enter the new password on DDM when you create a schema.
	Modifying a parameter template	The following parameters are set to fixed values. If their values are modified, DDM will not function properly. • **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	Your DDM instance cannot connect to associated RDS for MySQL instances.
	Modifying a VPC	The DDM instance and RDS for MySQL instance cannot communicate with each other if they are in different VPCs.
	Restoring data	Restoring data may damage data integrity.
Operations through an RDS for 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 an account created on DDM	After an account is deleted, logical tables cannot be created on DDM.
	Deleting a physical table created on DDM	After a physical table is deleted, data stored on DDM will be lost. The corresponding logical table becomes unavailable on DDM.

Operation Type	Operation	Impact of the Operation
	Changing the name of a physical table created on DDM	DDM cannot obtain data of the corresponding logical table, and the logical table becomes unavailable on DDM.
	Changing a record	Changing a record in a broadcast table will affect the data consistency of shards.
	Modifying a whitelist	A DDM instance cannot access the RDS for MySQL instance if it is not in the RDS instance whitelist.

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

Scenario

If there is already a primary key whose data type is a boundary value, in your DDM instance, duplicate primary keys will be reported when you insert a data record that is beyond the data range of the primary key.

Procedure

Step 1 Log in to the management console.

Step 2 On the **Instances** page, locate the RDS for MySQL instance associated with your DDM instance and click the name of the RDS for MySQL instance.

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 statement 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

21.4.4 How Can I Query RDS for MySQL Information by Running Command show full innodb status?

After you connect to a DDM instance through the MySQL client, you can run command **show full innodb status** to query information about the associated RDS for MySQL 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.

21.4.5 What Should I Pay Attention to When Selecting RDS for MySQL Instance Specifications?

Specifications of associated RDS for MySQL instances should be greater than that of the DDM instance. Otherwise the performance will be affected.

Example:

Question: If a DDM instance has 8 vCPUs and 16 GB memory, which specification should I select for its associated RDS for MySQL instances? Should I use a primary/standby or single-node architecture?

Answer: According to best practices, the configuration ratio of DDM instances to RDS for MySQL instances should remain 1:2. If a DDM instance has 8 vCPUs and 16 GB memory, select at least two RDS for MySQL instances with 8 vCPUs and 16 GB memory. The number of required RDS for MySQL instances depends on your service requirements.

To run production workloads, select primary/standby RDS for MySQL instances to ensure high availability.

21.5 Connection Management

21.5.1 How Can I Handle Garbled Characters Generated When I Connect a MySQL Instance to a DDM Instance?

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 -h 127.0.0.1 -P 5066 -D database --default-character-set=utf8 -u ddmuser –p password
```

