

# Distributed Database Middleware

## User Guide

**Issue**            01  
**Date**             2022-09-30



**Copyright © Huawei Technologies Co., Ltd. 2022. All rights reserved.**

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

## **Trademarks and Permissions**



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

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

## **Notice**

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

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

---

# Contents

---

<b>1 Function Overview.....</b>	<b>1</b>
<b>2 Permissions Management.....</b>	<b>3</b>
2.1 Creating a User and Granting Permissions.....	3
2.2 Creating a Custom Policy.....	4
<b>3 Instance Management.....</b>	<b>6</b>
3.1 Buying a DDM Instance.....	6
3.2 Splitting Read-only and Read-Write Services.....	8
3.2.1 What Is Read-only Service Isolation?.....	8
3.2.2 How Are Read-only Services Split from Read-Write Services.....	9
3.3 Changing Class of a DDM Node.....	10
3.4 Scaling Out a DDM Instance.....	11
3.5 Scaling In a DDM Instance.....	12
3.6 Changing Billing Mode of a DDM Instance.....	13
3.7 Renewing a DDM Instance.....	13
3.8 Restarting a DDM Instance.....	14
3.8.1 Restarting a DDM Instance.....	14
3.8.2 Restarting a Node.....	14
3.9 Unsubscribing from a DDM Instance.....	14
3.10 Deleting a DDM Instance.....	15
3.11 Modifying Parameters of a DDM Instance.....	16
3.12 Splitting Read and Write Requests.....	21
3.13 Configuring a Parameter Template.....	23
<b>4 Connection Management.....</b>	<b>25</b>
4.1 Configuring Access Control.....	25
4.2 Modifying the Floating IP Address of a DDM Instance and a Group.....	26
4.3 Binding and Unbinding an EIP.....	26
4.4 Changing a Database Port.....	28
4.5 Changing the Security Group of a DDM Instance.....	28
<b>5 Parameter Template Management.....</b>	<b>30</b>
5.1 Creating a Parameter Template.....	30
5.2 Editing a Parameter Template.....	31
5.3 Comparing Two Parameter Templates.....	32

5.4 Viewing Parameter Change History.....	33
5.5 Replicating a Parameter Template.....	34
5.6 Applying a Parameter Template.....	35
5.7 Viewing Application Records of a Parameter Template.....	36
5.8 Modifying the Description of a Parameter Template.....	36
5.9 Deleting a Parameter Template.....	37
<b>6 Task Center.....</b>	<b>38</b>
<b>7 Schema Management.....</b>	<b>40</b>
7.1 Creating a Schema.....	40
7.2 Exporting Schema Information.....	41
7.3 Importing Schema Information.....	42
7.4 Deleting a Schema.....	42
7.5 Configuring the SQL Blacklist.....	43
<b>8 Shard Configuration.....</b>	<b>45</b>
8.1 Overview and Application Scenarios.....	45
8.2 Assessment.....	46
8.3 Pre-check.....	47
8.4 Operation Guide.....	49
<b>9 Data Node Management.....</b>	<b>53</b>
9.1 Overview.....	53
9.2 Configuring Read Weights.....	53
9.3 Synchronizing Data Node Information.....	54
9.4 Reloading Table Data.....	54
<b>10 Account Management.....</b>	<b>56</b>
10.1 Creating an Account.....	56
10.2 Modifying an Account.....	58
10.3 Deleting an Account.....	58
10.4 Resetting the Password of an Account.....	59
10.5 Managing Permissions.....	60
10.5.1 Account Permissions.....	60
10.5.2 Account Requirements.....	60
10.5.3 Managing Permissions.....	61
<b>11 Backups and Restorations.....</b>	<b>65</b>
11.1 Overview.....	65
11.2 Consistent Backups.....	66
11.3 Restoring Data to a New Instance.....	66
11.4 Restoring Metadata.....	67
<b>12 Data Migration.....</b>	<b>69</b>
12.1 Overview.....	69
12.2 Migration Evaluation.....	70

12.3 Scenario 1: Migrating Data from Huawei Cloud RDS to DDM.....	72
12.4 Scenario 2: Migrating Data from an On-Premises RDS Instance for MySQL to DDM.....	72
12.5 Scenario 3: Migrating Data from a Third-Party RDS for MySQL Instance to DDM.....	82
12.6 Scenario 4: Migrating Data from a Self-Built MySQL Instance to DDM.....	91
12.7 Scenario 5: Migrating Data from Heterogeneous Databases to DDM.....	100
12.8 Scenario 6: Exporting Data from a DDM Instance.....	100
<b>13 Slow Queries.....</b>	<b>102</b>
<b>14 Monitoring Management.....</b>	<b>103</b>
14.1 Metrics.....	103
14.2 Viewing Metrics.....	105
14.3 Network Metrics.....	107
<b>15 Auditing.....</b>	<b>109</b>
15.1 Key Operations Recorded by CTS.....	109
15.2 Querying Traces.....	111
<b>16 SQL Syntax.....</b>	<b>113</b>
16.1 Introduction.....	113
16.2 DDL.....	115
16.2.1 Overview.....	116
16.2.2 Creating a Table.....	116
16.2.3 Sharding Algorithm Overview.....	117
16.2.4 Sharding Algorithms.....	120
16.2.4.1 MOD_HASH.....	120
16.2.4.2 MOD_HASH_CI.....	122
16.2.4.3 RIGHT_SHIFT.....	124
16.2.4.4 MM.....	125
16.2.4.5 DD.....	126
16.2.4.6 WEEK.....	127
16.2.4.7 MMDD.....	128
16.2.4.8 YYYYMM.....	129
16.2.4.9 YYYYDD.....	131
16.2.4.10 YYYYWEEK.....	133
16.2.4.11 HASH.....	134
16.2.4.12 Range.....	136
16.3 DML.....	138
16.3.1 INSERT.....	138
16.3.2 REPLACE.....	139
16.3.3 DELETE.....	139
16.3.4 UPDATE.....	140
16.3.5 SELECT.....	140
16.3.6 SELECT JOIN Syntax.....	141
16.3.7 SELECT UNION Syntax.....	142

16.3.8 SELECT Subquery Syntax.....	142
16.3.9 Unsupported DML Statements.....	143
16.3.10 Supported System Schema Queries.....	144
16.4 Functions.....	144
16.5 Use Constraints.....	147
16.6 Supported SQL Statements.....	148
16.6.1 CHECK TABLE.....	148
16.6.1.1 Checking DDL Consistency of Physical Tables in All Logical Tables.....	148
16.6.1.2 Checking DDL Consistency of Physical Tables in One Logical Table.....	149
16.6.2 SHOW RULE.....	151
16.6.3 SHOW TOPOLOGY.....	152
16.6.4 SHOW DATA NODE.....	152
16.6.5 TRUNCATE TABLE.....	152
16.6.5.1 HINT-DB.....	152
16.6.5.2 HINT-TABLE.....	153
16.6.5.3 HINT-DB/TABLE.....	154
16.6.5.4 Additional Information.....	154
16.6.6 HINT- ALLOW_ALTER_RERUN.....	154
16.6.7 LOAD DATA.....	155
16.6.8 SHOW PHYSICAL PROCESSLIST.....	156
16.6.9 Customized Hints for Read/Write Splitting.....	157
16.6.10 Setting a Hint to Skip the Cached Execution Plan.....	158
16.6.11 Specifying a Shard Using a Hint When Executing a SQL Statement.....	158
16.7 Global Sequence.....	158
16.7.1 Overview.....	158
16.7.2 Using NEXTVAL or CURRVAL to Query Global Sequence Numbers.....	161
16.7.3 Using Global Sequences in INSERT or REPLACE Statements.....	163
16.8 Database Management Syntax.....	165
16.9 Advanced SQL Functions.....	166
<b>17 Quotas.....</b>	<b>167</b>
<b>A Change History.....</b>	<b>169</b>

# 1 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 MySQL databases, helping handle highly concurrent access to massive volumes of data.

**Table 1-1** lists the functions supported by DDM.

**Table 1-1** DDM functions

Category	Function
Permissions	Creating a user and granting the user permissions to use DDM and DDM custom policies. For details, see <a href="#">Permissions Management</a> .
Instances	Creating, deleting, renewing, unsubscribing from, and restarting a DDM instance, and changing class of a DDM instance. For details, see <a href="#">Instance Management</a> .
Backups	Restoring data to a new DDM instance and restoring metadata. For details, see <a href="#">Backups and Restorations</a> .
Parameter templates	Creating, editing, replicating, and applying a parameter template, and comparing two parameter templates. For details, see <a href="#">Parameter Template Management</a> .
Task center	Enabling you to view progress and statuses of asynchronous tasks submitted on the console. For details, see <a href="#">Task Center</a> .
Schemas	Creating, exporting, importing, and deleting schemas. For details, see <a href="#">Schema Management</a> .
Flexible shards configuration for a schema	You can increase shards or data nodes to scale out storage. For details, see <a href="#">Shard Configuration</a> .
Accounts	Creating, modifying, and deleting a DDM account, and resetting its password. For details, see <a href="#">Account Management</a> .

Category	Function
Data migration	Migrating data from Huawei Cloud and a third-party cloud to DDM or exporting data from DDM. For details, see <a href="#">Data Migration</a> .
Monitoring	Providing metrics and methods of viewing metrics. For details, see <a href="#">Monitoring Management</a> .
SQL syntax	Describing DDL, DML, global sequence, SQL statements, and sharding algorithms. For details, see <a href="#">SQL Syntax</a> .



# 2 Permissions Management

---

## 2.1 Creating a User and Granting Permissions

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

This chapter describes fine-grained permissions management on DDM. For details, see **Service Overview > What Is IAM** in the *Identity and Access Management User Guide*.

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

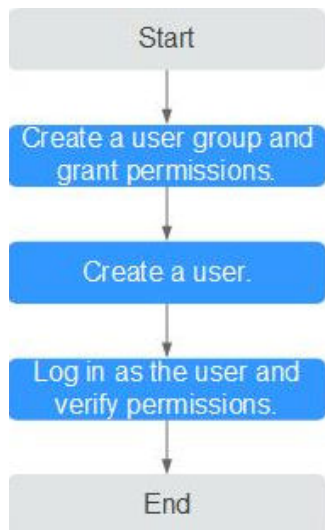
This section describes the procedure for granting permissions.

### Prerequisites

Before assigning permissions to user groups, you should learn about DDM system policies and select the policies based on service requirements. For details about the system permissions supported by DDM, see **Service Overview > Permissions Management** in the *Distributed Database Middleware User Guide*. For the system policies of other services, see **System Permissions > System Permissions**.

## Process Flow

**Figure 2-1** Flowchart for granting DDM permissions



1. Create a user group and assign permissions to it.  
Create a user group on the IAM console, and assign the **DDM ReadOnlyAccess** policy to the group.
2. Create an IAM user and add it to a user group.  
Create a user on the IAM console and add the user to the group created in step 1.
3. Log in and verify permissions.  
Log in to the DDM console using the created user, and verify that the user only has read permissions for DDM.
  - 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** (assuming that 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.

## 2.2 Creating a Custom Policy

Custom policies can be created as a supplement to system-defined policies of DDM.

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.

For details, see **Permissions > Policies > Custom Policies > Creating a Custom Policy** in the *Identity and Access Management User Guide*. The following section contains 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"
    }
  ]
}
```

# 3 Instance Management

## 3.1 Buying a DDM Instance

### Prerequisites

You have logged in to the DDM console.

### Procedure

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

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

**Table 3-1** Parameter description

Parameter	Description
Billing Mode	DDM instance billing mode, which can be Yearly/Monthly or Pay-per-use. <ul style="list-style-type: none"><li>• <b>Yearly/Monthly:</b> Specify a required duration. The system will deduct the fees incurred from your account based on the service price.</li><li>• <b>Pay-per-use:</b> Do not specify any required duration because the system deducts the fees incurred from your account every hour based on how much the service is used.</li></ul>
Region	Region where the DDM instance is located. Select the required region.

Parameter	Description
AZ	<p>Availability zone where the DDM instance is deployed.</p> <p>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.</p> <p>A DDM instance can be deployed across AZs to provide cross-AZ DR.</p> <p>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.</p> <p><b>NOTE</b> Deploy your application, DDM instance, and required RDS instances in the same AZ to reduce network latency. Cross-AZ deployment may increase network latency.</p>
Instance Name	<p>Name of the DDM instance, which:</p> <ul style="list-style-type: none"> <li>• Cannot be left blank.</li> <li>• Must start with a letter.</li> <li>• Must be 4 to 64 characters long.</li> <li>• Can contain only letters, digits, and hyphens (-).</li> <li>• Cannot contain other special characters.</li> </ul>
Node Class	<p>Class of the DDM instance. You can select general-enhanced (x86) or Kunpeng general computing-plus (Arm).</p> <p><b>NOTE</b> Estimate compute and storage requirements of your applications based on your service type and scale before you buy a DDM instance, and then select an appropriate class so that the CPU and memory specifications of your DDM instance can better meet your needs.</p>
Instance Nodes	<p>Number of nodes for deploying the DDM instance. You can apply for up to 32 nodes at a time.</p>
VPC	<p>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.</p> <p>Click <b>View VPC</b> to show more details and security group rules.</p> <p><b>NOTE</b> The DDM instance should be in the same VPC as the required data nodes. After a DDM instance is created, the VPC cannot be changed.</p>
Security Group	<p>Select an existing security group.</p> <p>The same security group is recommended for your DDM instance, application, and required data nodes so that network access is not restricted. If different security groups are selected, add security group access rules to enable such network access.</p>
Parameter Template	<p>A parameter template acts as a container for engine configuration values that can be applied to one or more DDM instances. You can modify parameters in a parameter template to manage configurations of the DDM instance that the template applies to.</p>

Parameter	Description
Enterprise Project	EPS provides a unified method to manage cloud resources and personnel by enterprise project.
Tags	Tags can be added to instances to help you manage instances and collect expense data.
Required Duration	Duration of the DDM instance. This parameter is available only if <b>Billing Mode</b> is set to <b>Yearly/Monthly</b> . You can select 1 month, 2 months, 3 months, 4 months, 5 months, 6 months, 7 months, 8 months, 9 months, 1 year, 2 years, or 3 years. If you select <b>Auto-renew</b> , the renew cycle is the same as the selected duration.

**Step 3** After the configuration is complete, click **Next** at the bottom of the page.

**Step 4** Perform subsequent operations based on the billing mode you select:

- If you select pay-per-use, click **Submit**.
- If you select yearly/monthly, click **Pay Now**.

----End

## 3.2 Splitting Read-only and Read-Write Services

### 3.2.1 What Is Read-only Service Isolation?

#### Overview

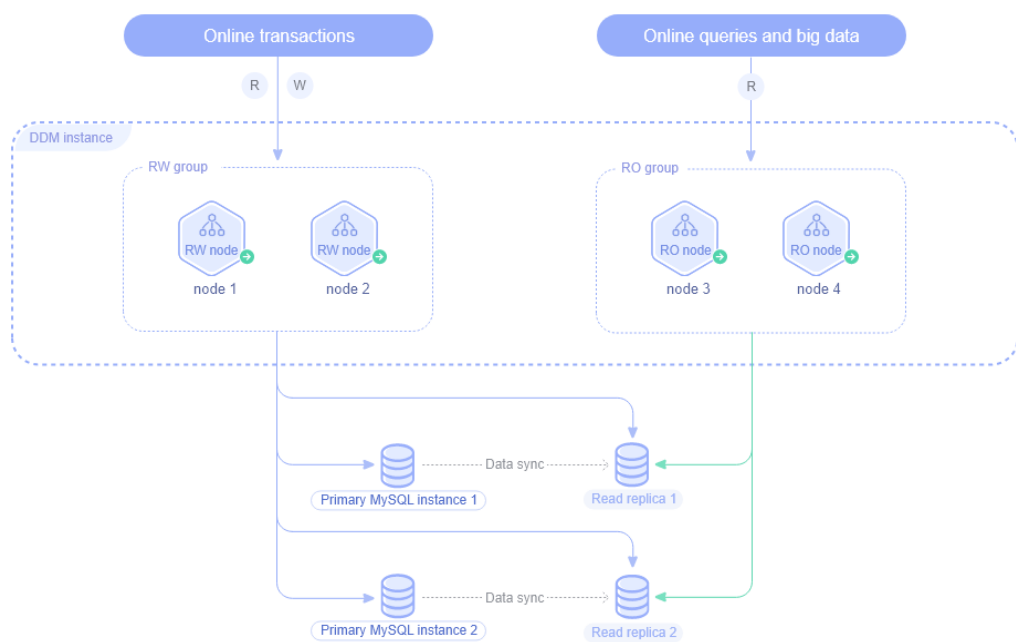
DDM provides read-only service isolation by grouping nodes of a DDM instance to provide physically separated compute and storage resources.

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.

**NOTE**

- The node group function is only available for whitelisted users. To enable this function, contact technical support.
- The kernel version must be 2.4.1.2 or later.
- If you want read-only groups to handle SQL queries, make sure that the associated data node 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

**Figure 3-1** Node groups



### 3.2.2 How Are Read-only Services Split from Read-Write Services

#### Procedure

- Step 1** Log in to the DDM console and choose **Instances** in the navigation pane. In the instance list, locate the required instance and click its name.
- Step 2** Choose **Basic Information** in the navigation pane to view node information.
- Step 3** In the **Node Information** area, click **Create Group**. After a group is created, existing nodes are included into another read/write group by default, responsible for handling read/write requests to core services.

**NOTE**

- One DDM instance supports multiple read-only groups. Each group contains at least 2 nodes, and each instance contains up to 32 nodes.
- One node belongs to only one group, and its group cannot be changed once determined. Nodes in the same group must be of the same class.

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

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

**Step 6** 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.

**Figure 3-2** Node groups

Group Name	Instance	Role	Floating IP Address	Access Control	Operation
group-f392	1	Read-only	[IP Address]	[Toggle]	Change Node Class   Scale Out   More
group-default	3	Read/Write	[IP Address]	[Toggle]	Change Node Class   Scale Out   More

**Figure 3-3** Group and node information

Group Name	Instance	Role	Floating IP Address	Access Control	Operation
group-f392	1	Read-only	[IP Address]	[Toggle]	Change Node Class   Scale Out   More

Name/ID	Status	AZ	Operation
[Node Name]	Running	az3	Restart

**NOTE**

- Before you create a read-only group, enable read/write splitting for your DDM instance.
- After a group is created, you can change its class, add new nodes to or remove nodes from it, or configure access control for it.
- Deleting groups is not supported for yearly/monthly DDM instances.

To delete a group of a pay-per-use DDM instance, locate the group that you want to delete and click **Delete**. The corresponding floating IP address becomes invalid once the group is deleted. This may affect your services. Retain at least one read/write group.

----End

## 3.3 Changing Class of a DDM Node

### Prerequisites

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



---

**NOTICE**

Change node class during off-peak hours because services will be interrupted for a while during class changing.

---

## Procedure

---

**NOTICE**

After a read-only group is created, the entry for changing node class will be moved to the operation column of the group.

---

- Step 1** In the instance list, locate the DDM instance whose node class you want to change and click its name. Click **Change**.
- Step 2** On the displayed page, select the required class.
- Step 3** Perform subsequent operations based on the instance's billing mode:
  - If the billing mode is Pay-per-use, click **Submit**.
  - If the billing mode is Yearly/Monthly, click **Pay Now**.
- Step 4** Confirm the configurations and click **Submit**.
- Step 5** Switch back to the instance list and check whether the status of the instance changes to **Changing class**. You can also view the change task at Task Center.

 **NOTE**

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

----End

## 3.4 Scaling Out a DDM Instance

### Scenarios

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

 **NOTE**

- Scale out your DDM instance during off-peak hours.
- Make sure that the associated 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** In the instance list, locate the DDM instance that you want to scale out and click its name. Click **Scale Out**.
- Step 2** On the displayed page, view the current instance configuration, select the required AZ, and specify the number of new nodes.
- Step 3** Click **Next**.
- Step 4** On the displayed page, click **Submit** if all configurations are correct.

**Figure 3-4** Confirming the instance information

Resource	Configuration	Billing Mode	Price
	Instance Name ddm-cb55-xuexin-test		
	Instance ID [REDACTED]		
DDM Instance	[REDACTED]	Pay-per-use	¥3.84/hour
	Nodes to be added 2 az3		

----End

## 3.5 Scaling In a DDM Instance

### Scenarios

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

#### NOTE

- Scaling in yearly/monthly DDM instances is not supported.
- Scale in your DDM instance during off-peak hours. For pay-per-use instances, nodes are removed after the scale-in request is submitted.
- 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** In the instance list, locate the DDM instance that you want to scale in and click its name. Click **Scale In**.
- Step 2** On the displayed page, view the current instance configuration and specify the number of nodes to be removed.
- Step 3** Click **Next**.
- Step 4** On the displayed page, click **Submit** if all configurations are correct.

----End

## 3.6 Changing Billing Mode of a DDM Instance

### Prerequisites

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

### Changing to Yearly/Monthly

**Step 1** In the instance list, locate a pay-per-use instance whose billing mode you want to change and choose **More > Change to Yearly/Monthly** in the **Operation** column.

**Step 2** Select a renewal duration and decide if you want to enable **Auto-renew** and click **Pay**.

----End

### Changing to Pay-Per-Use

**Step 1** In the instance list, locate a yearly/monthly instance whose billing mode you want to change and choose **More > Change to Pay-per-use** in the **Operation** column.

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

**Step 3** On the displayed page, click **Submit**.

#### NOTE

Once a request to change to pay-per-use is issued, the pay-per-use billing mode takes effect after the yearly/monthly subscription expires. You will be billed on an hourly basis for what you use.

----End

## 3.7 Renewing a DDM Instance

### Prerequisites

- You have logged in to the DDM console.
- The billing mode of the target instance is yearly/monthly.

### Procedure

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

**Step 2** On the **Renew** page, set the renewal duration.

**Step 3** Confirm the information and click **Pay**.

----End

## 3.8 Restarting a DDM Instance

### 3.8.1 Restarting a DDM Instance

#### Prerequisites

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

---

**NOTICE**

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

---

#### Procedure

- Step 1** In the instance list, locate the DDM instance that you want to restart and choose **More > Restart** in the **Operation** column.
  - Step 2** In the displayed dialog box, click **Yes**.
  - Step 3** Wait until the instance is restarted.
- End

### 3.8.2 Restarting a Node

You can restart a single node of your DDM instance.

- Step 1** In the instance list, locate the DDM instance whose node you want to restart and click its name.
  - Step 2** In the **Node Information** area, locate the target node and click **Restart** in the **Operation** column.
  - Step 3** In the displayed dialog box, click **Yes**.
  - Step 4** Wait until the node is restarted.
- End

## 3.9 Unsubscribing from a DDM Instance

#### Prerequisites

- You have logged in to the DDM console.
- There are DDM instances whose billing mode is yearly/monthly.

## Procedure

- Step 1** In the instance list, locate the instance that you want to unsubscribe from and choose **More > Unsubscribe** in the **Operation** column.
- Step 2** In the displayed dialog box, click **Yes**.  
The **Billing Center - Unsubscriptions** page is displayed.
- Step 3** On the **Cloud Service Unsubscriptions** page, click **Unsubscribe from In-Use Resources**.
- Step 4** Locate the DDM instance that you want to unsubscribe from and click **Unsubscribe from Resource** in the **Operation** column.
- Step 5** Confirm the instance information and select your reason.
- Step 6** View the unsubscription information, select **I have confirmed that a handling fee will be charged for this unsubscription**, and click **Unsubscribe**.

 **NOTE**

After a resource is unsubscribed from, its data will be deleted immediately and cannot be restored. Ensure that you have backed up the data or the data will not be required any longer.

- Step 7** In the displayed dialog box, click **Yes**.

----End

## 3.10 Deleting a DDM Instance

### Prerequisites

You have logged in to the DDM console.

---

**NOTICE**

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

---

### Procedure

- Step 1** In the instance list, locate the DDM instance that you want to delete and choose **More > Delete** in the **Operation** column.
- Step 2** In the displayed dialog box, click **Yes**.

 **NOTE**

- To delete data stored on data nodes, select **Delete data on data nodes**.
- A monthly/yearly instance cannot be directly deleted. If you no longer need the instance, switch to **Billing Center > Orders Unsubscriptions and Returns/Exchanges > Cloud Service Unsubscriptions** and unsubscribe the instance.

----End

## 3.11 Modifying Parameters of a DDM Instance

### Scenarios

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

### Prerequisites

There is a DDM instance available and running normally.

### Procedure

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

**Table 3-2** Parameters of a DDM instance

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

Parameter	Default Value	Value Range	Description
character_set_server	utf8mb4	gbk, utf8, utf8mb4	DDM server's character set. To store emoticons, set both this parameter and the character set on RDS to <b>utf8mb4</b> . For a DDM instance 3.0.9 or later, you can execute <b>show variables like '%char%'</b> to query its character set. You will find that <b>character_set_client</b> , <b>character_set_results</b> , and <b>character_set_connection</b> in the command output all have a fixed value, <b>utf8mb4</b> .
collation_server	utf8mb4_unicode_ci	utf8mb4_unicode_ci, utf8mb4_bin, utf8mb4_general_ci	Collation on the DDM server.
concurrent_execution_level	DATA_NODE	RDS_INSTANCE, DATA_NODE, PHY_TABLE	Concurrency level of scanning table shards in a logical table. <b>DATA_NODE</b> : indicates that database shards are scanned in parallel and table shards in each database shard are scanned in serial. <b>RDS_INSTANCE</b> : indicates that RDS instances are scanned in parallel and shards in each DB instance are scanned in serial. <b>PHY_TABLE</b> : indicates that all table shards are scanned in parallel.
connection_idle_timeout	28800	60—86400	Number of seconds the server waits for activity on a connection before closing it. The default value is <b>28800</b> , indicating that the server waits for 28800 seconds before closing a connection.
contains_shard_key	OFF	OFF or ON	Whether the SELECT, UPDATE, and DELETE statements must contain sharding keys in filter conditions.
ddl_precheck_mdل_threshold_time	120	0—3600	Threshold of the MDL duration in DDL pre-check. The unit is second. The default value is <b>120</b> .

Parameter	Default Value	Value Range	Description
enable_table_recycle	OFF	OFF or ON	<p><b>ON:</b> indicates that the table recycle bin is enabled.</p> <p><b>OFF:</b> indicates that the table recycle bin is disabled.</p> <p>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.</p>
long_query_time	1	0.01-10	Minimum duration of a query to be logged as slow, in seconds. The default value is <b>1</b> , indicating that the query is considered as a slow query if its execution duration is greater than or equal to 1 second.
max_allowed_packet	1073741824	1024-1073741824	Maximum size of one packet or any generated intermediate string. The packet message buffer is initialized to <b>net_buffer_length</b> bytes, but can grow up to <b>max_allowed_packet</b> bytes when needed. This value is small by default, to catch large (and possibly incorrect) packets. The value must be a multiple of <b>1024</b> .
max_background_connections	0	0—10000000	Maximum of concurrent client connections allowed per DDM instance. When this parameter is set to <b>0</b> (default), the maximum concurrent connections from a DDM node to an RDS instance is: (RDS instance's maximum connections - 20)/DDM nodes.



Parameter	Default Value	Value Range	Description
max_connections	20000	10–40000	<p>Maximum number of concurrent client connections allowed per DDM instance.</p> <p>This value depends on specifications and processing capabilities of the target 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.</p> <p>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.</p> <p>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.</p> <p>Carry out pressure tests on your services and then select a proper value.</p>
min_backend_connections	10	0—10000000	<p>Minimum concurrent connections from a DDM node to an RDS instance. The default value is <b>10</b>.</p>
seconds_behind_master	30	0—7200	<p>Threshold in seconds of the replication lag between a primary RDS instance to its read replica. The default value is <b>30</b>, 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.</p>
sql_execute_timeout	28800	100—28800	<p>Number of seconds to wait for a SQL statement to execute before it times out. The default value is <b>28800</b>, indicating that the SQL statement times out if its execution time is greater than or equal to 28800 seconds.</p>



**Figure 3-6** Result if `bind_table` is used

```

mysql> explain select * from bill b join ordertbl o on b.cid = o.orderid where b.id > 2;
+-----+
| Logical Execution Plan |
+-----+
|
|  EXPLAIN_SELECT--@_#421 [0-10] [joinType=left-outer, pushDown=]--SELECT BILL.id, BILL.cid, BILL.suf, BILL.account, ORDERTB.orderid, ORDERTB.ordermaster, ORDERTB.totalamount, OR
| ORDERTB.createTime, ORDERTB.updateTime, ORDERTB.payTime, ORDERTB.payid, ORDERTB.unsubscribeid, ORDERTB.unsubscribeamount, ORDERTB.unsubscribeTime, ORDERTB.unsubscribeReason, ORDERTB
| status, ORDERTB.version, ORDERTB.merchantid, ORDERTB.credential FROM @_#421 BILL INNER JOIN @_#421 ORDERTB ON BILL.cid = ORDERTB.orderid WHERE BILL.id > 2 |
|
+-----+
| plan cache: hit |
+-----+

```

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

**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

## 3.12 Splitting Read and Write Requests

### Overview

Read/write splitting offloads read requests from primary DB instances to read replicas on a data node 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.

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

If the associated primary DB 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 DB instance. Recover the faulty instance as soon as possible.

### Prerequisites

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

### Procedure

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

**Step 2** Choose **Data Nodes**.

**Step 3** Set weights of the associated instance.

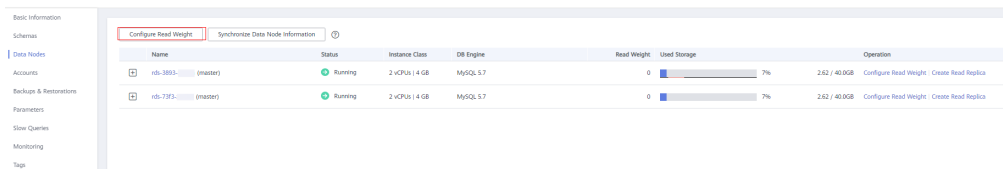
**NOTE**

If the associated instance is a read replica, it handles all separated read requests by default. To configure read/write weights of the read replica, perform the following operations:

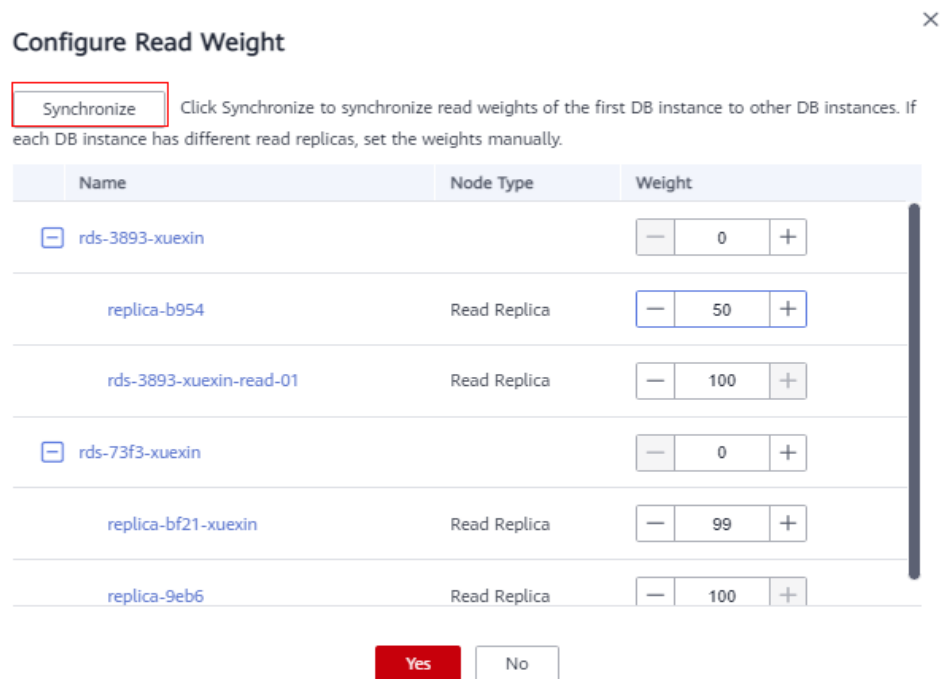
- If you want to set read weights of multiple instances, click **Configure Read Weight**. **Figure 3-7** shows an example.

In the displayed dialog box, you can click **Synchronize** to apply the read weight of the first instance to other instances, as shown in **Figure 3-8**. This operation requires that all involved instances should have the same number of read replicas. Otherwise, you need to manually configure the read weight for each instance.

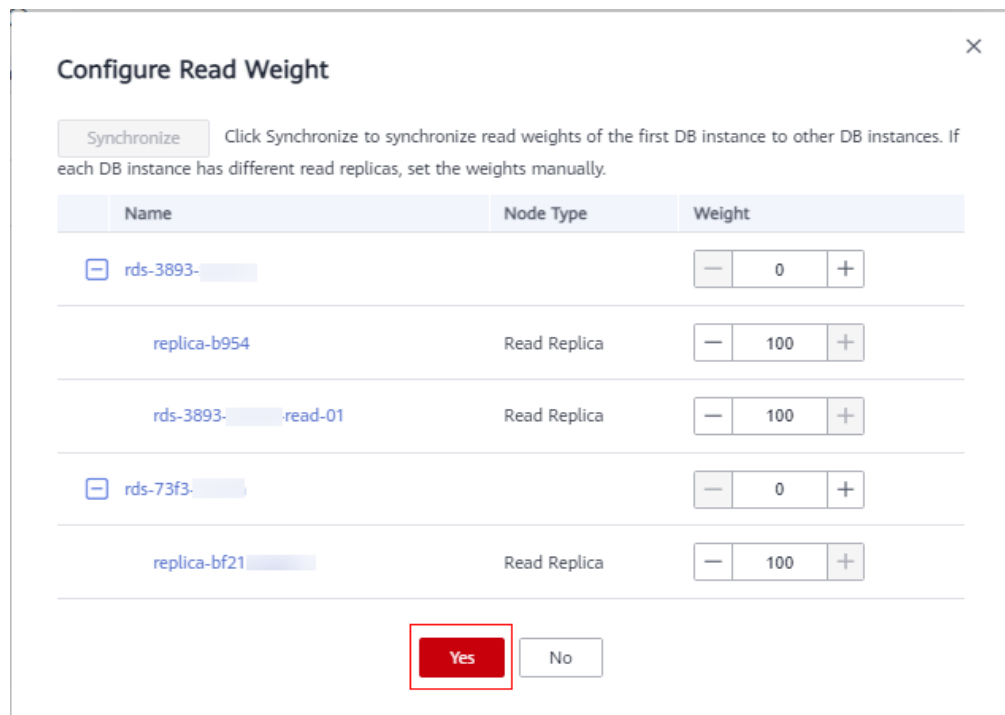
**Figure 3-7** Configuring read weights for multiple instances



**Figure 3-8** Synchronizing the read weight of the first instance to other instances

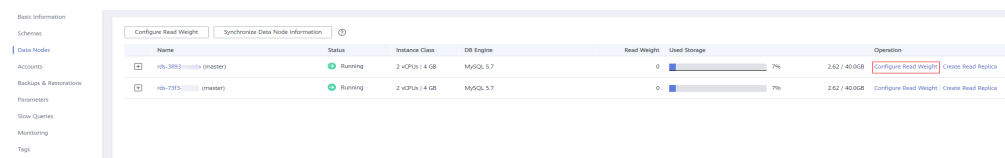


**Figure 3-9** Manually configuring the read weight for each instance



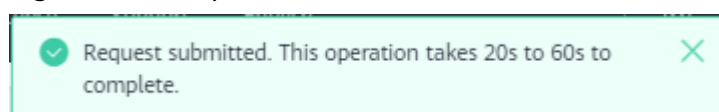
- If you want to set the read weight of an instance, locate the target instance and click **Configure Read Weight** in the **Operation** column.

**Figure 3-10** Configuring the read weight for a single instance



**Step 4** Wait the request is submitted.

**Figure 3-11** Request submitted



----End

## 3.13 Configuring a Parameter Template

### Prerequisites

You have logged in to the DDM console.

## Procedure

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

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

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

----End

# 4 Connection Management

---

## 4.1 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

---

**NOTICE**

After a read-only group is created, the entry for configuring access control will be moved to the operation column of the group.

---

- Step 1** On the DDM console, choose **Instances** in the navigation pane. In the instance list, locate the DDM instance that you want to scale out, and click its name. On the displayed **Basic Information** page, enable **Access Control** in the **Network Information** area.
- Step 2** Click **Configure**. In the **Configure Access Control** dialog box, specify **Access Policy**, enter the required IP addresses, and click **OK**.

 **NOTE**

If read/write splitting is enabled, access control only takes effect for groups.

----End

## 4.2 Modifying the Floating IP Address of a DDM Instance and a Group

### Scenarios

After load balancing is enabled for DDM, you can modify the floating IP addresses of your DDM instance and groups.

### Modifying the Floating IP Address of a DDM Instance

- Step 1** On the **Instances** page, select the instance whose floating IP address you want to modify and click its name.
- Step 2** In the **Network Information** area, click **Modify** beside the **Floating IP Address** field.
- Step 3** In the displayed dialog box, enter an available IP address in the same VPC and subnet as the current floating IP address and click **OK**.

----End

### Modifying the Floating IP Address of a Group

- Step 1** On the **Instances** page, select the instance whose floating IP address you want to modify and click its name.
- Step 2** In the **Group Information** area, locate the group whose floating IP address you want to modify and choose **More > Modify Floating IP Address**.
- Step 3** In the displayed dialog box, enter an available IP address in the same VPC and subnet as the current floating IP address and click **OK**.

----End

## 4.3 Binding and Unbinding an EIP

### Scenarios

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

### Constraints

After you bind an EIP to a DDM instance with load balancing enabled, the system will random select a node of the instance and bind the EIP to the node. In this case, load balancing may take effect. So, do not bind any EIP for a DDM instance with load balancing enabled.

In the upper right corner of the DDM console, you can choose **Service Tickets > Create Service Ticket** to enable load balancing for your DDM instance.



## Prerequisites

- You have applied for an EIP on the VPC console.
- 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 only to a node of the read/write groups.

## Binding an EIP

**Step 1** On the **Instances** page, select the instance that you want to bind an EIP for and click its name.

**Step 2** In the **Instance Information** area, click **Bind** beside **EIP**.

**Step 3** 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 4** On the **Basic Information** page, view the EIP that has been bound to the instance.

To unbind the EIP from the instance, see [Unbinding an EIP](#).

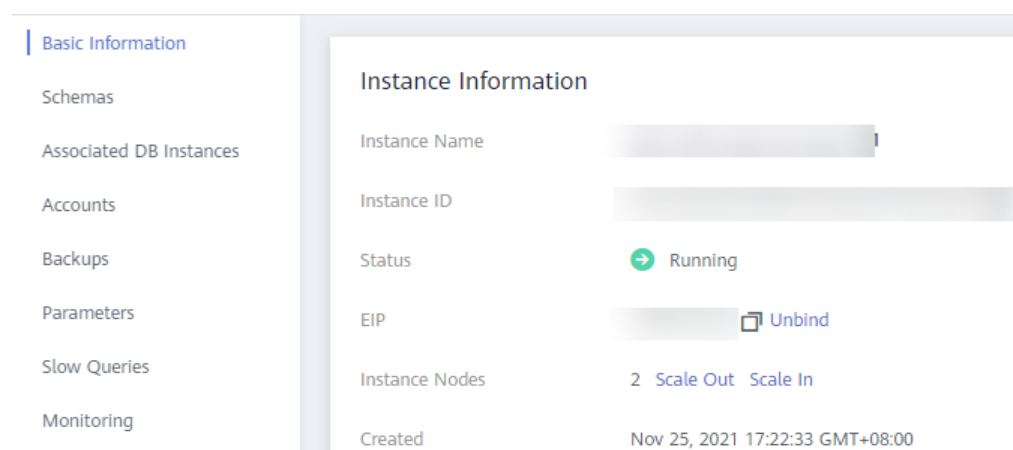
----End

## Unbinding an EIP

**Step 1** On the **Instances** page, select the DDM instance from which you want to unbind an EIP and click its name.

**Step 2** In the **Instance Information** area, click **Unbind**.

**Figure 4-1** Unbinding an EIP



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

**Step 4** On the **Basic Information** page, check whether the EIP is unbound.

----End


## 4.4 Changing a Database Port

### Scenarios

DDM allows you to change the database port of a DDM instance. After the port is changed, the instance will restart.

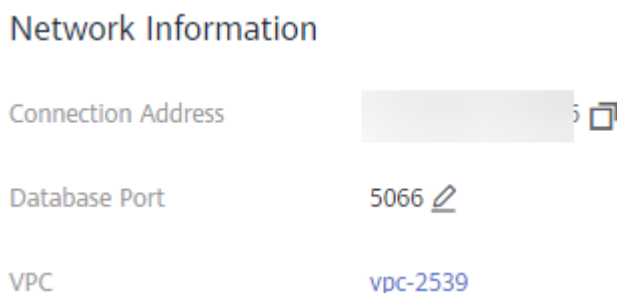
### Procedure



**Step 1** Log in to the DDM console, choose **Instances** in the navigation pane, locate the instance whose database port you want to change, and click its name.

**Step 2** In the **Connection Information** area on the **Basic Information** page, click  besides **Database Port**.

For DDM instances, the database 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**.

**Figure 4-2** Database port



- Specify a new port number and click .
  - In the dialog box, click **Yes**.  
Changing the database port requires a restart of the DDM instance.
  - In the dialog box, click **No**.
- To cancel the change, click .

**Step 3** View the results on the **Basic Information** page.

----End

## 4.5 Changing the Security Group of a DDM Instance

### Scenarios




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

For more information about security group configurations, see [FAQs > General Questions > How Do I Select and Configure a Security Group?](#)

 **NOTE**

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

## Procedure

- Step 1** Log in to the DDM console, choose **Instances** in the navigation pane, locate the instance whose security group you want to change, and click its name.
- Step 2** In the **Network Information** area on the **Basic Information** page, click  beside field **Security Group**.
- Specify a new security group and click .
  - To cancel the change, click .
- Step 3** View the results on the **Basic Information** page.

----End

# 5 Parameter Template Management

---

## 5.1 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](#).

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

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

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

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

## Procedure

**Step 1** Log in to the management console.

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

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

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

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

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

### NOTE

Each user can create up to 100 parameter templates.

----End

## 5.2 Editing a Parameter Template

To improve performance of a DDM instance, you can modify parameters in custom parameter templates based on service requirements.

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.

### NOTE

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

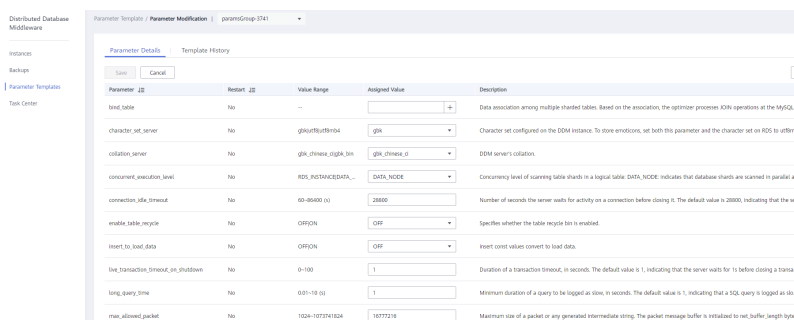
## Procedure

**Step 1** Log in to the management console.

- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Databases > Distributed Database Middleware**.
- Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the required parameter template, and click its name.
- Step 5** On the **Parameter Details** page, modify parameters as needed.

Available operations are as follows:

**Figure 5-1** Modifying parameters in a parameter template



- To save the modifications, click **Save**.
- To cancel the modifications, click **Cancel**.

**Step 6** After the parameter values are modified, click **Template History** to view details.

### NOTICE

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

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

----End



## 5.3 Comparing Two Parameter Templates

### Scenarios

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

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

## Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Databases > Distributed Database Middleware**.
- Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the required parameter template, and click **Compare** in the **Operation** column.
- Step 5** In the displayed dialog box, select a parameter template and click **OK**.

### NOTE

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

**Figure 5-2** Selecting a parameter template to be compared

## Compare Parameter Templates

Template 1 Default-DDM

Template 2

paramsGroup-3274

OK

Cancel

- 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.

**Figure 5-3** Comparing parameter templates

Parameter	paramsGroup-3741	Default-DDM
character_set_server	gbk	utf8

----End

## 5.4 Viewing Parameter Change History



### Scenarios

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

 NOTE

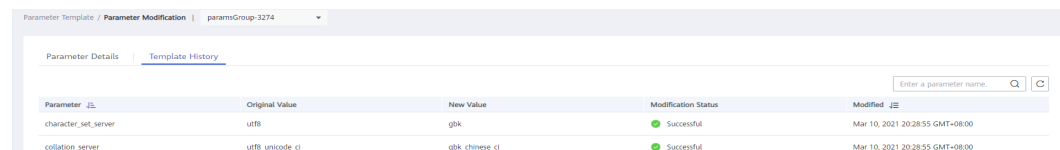
An exported or custom parameter template has initially a blank change history.

## Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.
- Step 3** Click  in the upper left corner of the page and choose **Databases > Distributed Database Middleware**.
- Step 4** Choose **Parameter Templates**, click the **Customer Templates** tab, locate the required parameter template, and choose **More > View Change History**.

You can view the template change history within a specified period (no more than two years). By default, the template change history of the last seven days is queried.

**Figure 5-4** Template History



Parameter	Original Value	New Value	Modification Status	Modified
character_set_server	utf8	gbk	Successful	Mar 10, 2021 20:28:55 GMT+08:00
collation_server	utf8_unicode_ci	gbk_chinese_ci	Successful	Mar 10, 2021 20:28:55 GMT+08:00

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

----End

## 5.5 Replicating a Parameter Template


### Scenarios

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


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

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

### Procedure

- Step 1** Log in to the management console.
- Step 2** Click  in the upper left corner and select a region and a project.



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

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



----End

## 5.6 Applying a Parameter Template

### Scenarios

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

### Procedure

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

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

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

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



----End

## 5.7 Viewing Application Records of a Parameter Template

### Scenarios

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

### Procedure

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

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

----End

## 5.8 Modifying the Description of a Parameter Template



### Scenarios


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



#### NOTE

You cannot modify the description of any default parameter template.

### Procedure

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

**Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the parameter template whose description you want to modify, and click  in the **Description** column.

**Step 5** Enter a new description. You can click  to submit or  to cancel the modification.

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

----End

## 5.9 Deleting a Parameter Template

### Scenarios

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

---

#### NOTICE

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

### Procedure

**Step 1** Log in to the management console.

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

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

**Step 4** Choose **Parameter Templates**, click the **Custom Templates** tab, locate the template that you want to delete, and click **Delete** in the **Operation** column.

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

----End

# 6 Task Center

---

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


 **NOTE**

The following tasks can be viewed:

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


## Procedure

**Step 1** Log in to the management console.

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

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

**Step 4** 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 a task name in the search box in the upper right corner.
- You can click  in the upper right corner to search for tasks executed within a specific period. The default time range is seven days.  
All tasks in the list can be retained 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**

# 7 Schema Management

## 7.1 Creating a Schema

### Prerequisites

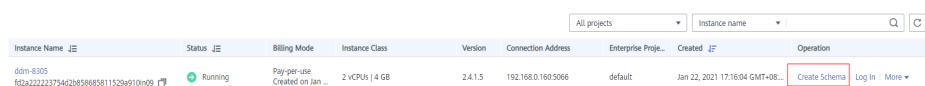
- You have logged in to the DDM console.
- The DDM instance is in the **Running** state.
- Do not modify or delete the internal accounts (DDMRW\*, DDMR\*, and DDMREP\*) created on data nodes. Otherwise, services will be affected.

#### 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 contains 16 to 32 characters.
- All DB instances associated with one schema must have the same major MySQL version.
- Multiple schemas can be created in a DDM instance and associated with the same data node. One DDM instance can be associated with either RDS for MySQL or GaussDB(for MySQL) instances, but not both.
- One data node cannot be associated with schemas in different DDM instances.
- If you select **Sharded** for **Sharding** when you create a schema, the shard name follows the rule: schema+xxxx. xxxx indicates the digit increased from 0000. For example, if a schema name is **db\_cbb5** and the total number of shards is 2, the shard names are **db\_cbb5\_0000** and **db\_cbb5\_0001**.

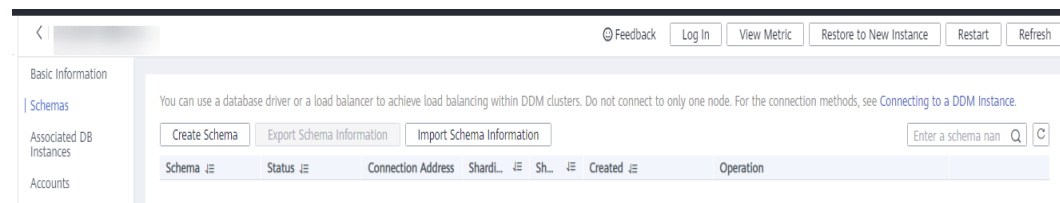
You can create a schema in two ways: on the **Instances** page or on the **Schemas** page. This section uses the **Instances** page as an example to describe how to create a schema.

**Figure 7-1** Instances page



Instance Name	Status	Billing Mode	Instance Class	Version	Connection Address	Enterprise Proj...	Created	Operation
ddm-8305 f02a222237540285586858115296910m09	Running	Pay-per-use Created on Jan...	2 vCPUs   4 GB	2.4.1.5	192.168.0.160:5066	default	Jan 22, 2021 17:16:04 GMT+08:00	Create Schema Log In More

Figure 7-2 Schemas page



## Procedure

- Step 1** In the navigation pane, choose **Instances**. In the instance list, locate the DDM instance that you want to create a schema for and click **Create Schema** in the **Operation** column.
- Step 2** On the displayed page, specify a sharding mode, enter a schema name, set the number of shards, select the required DDM accounts, and click **Next**.
- Step 3** Enter the database password and click **Test Connection**.
- Step 4** After the test becomes successful, click **Finish**.

----End

## 7.2 Exporting Schema Information

### Scenarios

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; in the instance list, locate the required DDM instance and click its name.
- Step 2** On the displayed page, in the navigation pane, choose **Schemas**.
- Step 3** On the displayed page, click **Export Schema Information**. All schema information of the current DDM instance is exported as a JSON file.

----End

## 7.3 Importing Schema Information

### Scenarios

When you deploy DR or migrate data across regions, you can import schema information in 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.

### Procedure

**Step 1** Log in to the DDM console; in the instance list, locate the DDM instance that you want to import schema information into and click its name.

**Step 2** On the displayed page, in the navigation pane, choose **Schemas**.

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

 **NOTE**

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

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

 **NOTE**

The number of selected data nodes is the same as the number of data nodes imported into the DDM instance.

----End

## 7.4 Deleting a Schema

### Prerequisites

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

---

**NOTICE**

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

---



## Procedure

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

### NOTE

- Your schema will become faulty if 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 DB instances have been deleted, click **Synchronize DB Instance Data** 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 DB Instance Data** and delete the schema.

----End

## 7.5 Configuring the SQL Blacklist

### Overview

Configure a blacklist and add those statements to it to prevent the system executing some SQL statements.

### Prerequisites

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

### Procedure

- Step 1** In the instance list, locate the instance that contains schemas you require and click the instance name.
- Step 2** On the displayed page, choose **Schemas**.
- Step 3** In the schema list, locate the schema that you want to configure a blacklist for and click **Configure SQL Blacklist** in the **Operation** column.
- Step 4** In the displayed dialog box, click **Edit**, enter the required SQL statements or regular expressions in prefix match, full-text, and regular expression match boxes, and click **OK**.

 **NOTE**

- **Prefix Match:** Enter SQL statements that contain keywords such as DROP XXXX or DELETE XXX and are not allowed by the current schema.
- **Full-text Match:** Enter full-text SQL statements that are not allowed by the current schema.
- **Regular Expression Match:** Enter regular expressions that are not allowed by the current schema.
- Separate SQL statements in the blacklist with commas (,). 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

# 8 Shard Configuration

## 8.1 Overview and Application Scenarios

### Overview

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 are burst.

### Application Scenarios

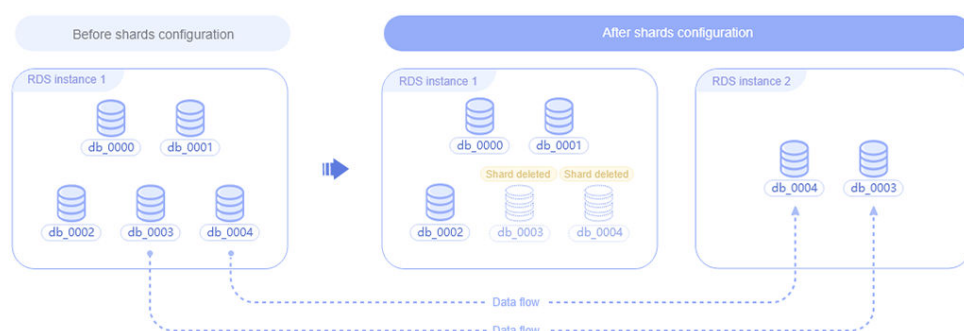
1. Keep shards unchanged and increase data nodes.

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

This method meets rapid service growth after horizontal sharding and can reduce costs at the beginning 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

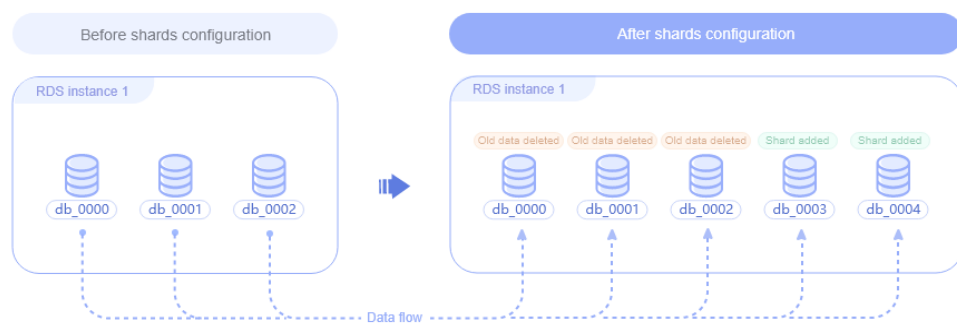


2. Increase shards with data nodes unchanged.

This method increases the shards, but not data nodes. It changes total shards, total table shards, and table sharding rules. Data is redistributed to different shards, and broadcast tables are increased.

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

**Figure 8-2** Increasing shards with RDS for MySQL instances unchanged

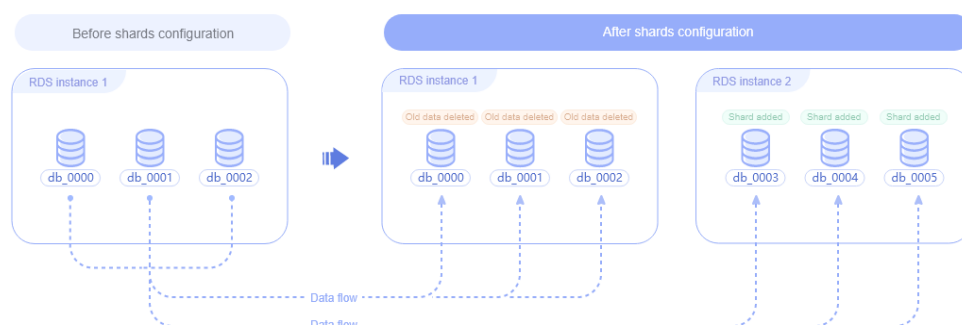


3. Increase both shards and data nodes.

This method increases both shards and data nodes. It changes total shards, total table shards, and table sharding rules. Data is redistributed to different shards, and broadcast tables are increased.

This method is suitable if 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** Increasing 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.

Customer stories:

A customer has a four-node DDM instance. Each node has 8 vCPUs and 16 GB memory. The instance is associated with 6 data nodes, containing 73,000 physical tables in total where 100 billion data records are stored, with the data volume up to about 12 TB.

Changing shards will definitely cause the migration of all schema data. Each data record must be rerouted, so the computing speed is obviously slower than the computing speed when shards are unchanged. Considering service requirements, change the DDM node class to 32 vCPUs | 64 GB, increase data nodes to 12, and upgrade the DDM kernel version to the latest. Then you can restore the node class to the original one as required. The shards are not changed, so only half of the shards are migrated from original data nodes to new data nodes, with no route redistribution involved. Keep shards unchanged and increase data nodes unless there is a single physical table whose storage has reached the upper limit.

## 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
Binlog backup time of the DB instance	Whether your full backups are retained for a time period long enough	Increase the retention period for full backups on the data node console.
Binlog enabled on data nodes	Whether binlog is enabled to support online shard configuration	If your data node is an RDS instance, no further action is required. If your data node is a GaussDB(for MySQL) instance, set <b>log_bin</b> to <b>true</b> on the GaussDB(for MySQL) console.

Item	Purpose	Solution to Check Failure
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. If your data node is a GaussDB(for MySQL) instance, set <b>binlog_expire_logs_seconds</b> to <b>604800</b> or a larger value.
Broadcast table consistency	Ensure broadcast table consistency before performing a shard configuration task.	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.	Contact DDM O&M personnel.
SQL statements for creating physical tables.	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.
Access to DB instances	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.
DB instance storage space	The disk space of data nodes is sufficient during shard configuration.	Scale up storage space of data nodes. <b>CAUTION</b> This check item is based on the estimated value that may be different from the actual value.

Item	Purpose	Solution to Check Failure
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 <b>Parameters</b> page of the data node console.

## 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 to rectify the inconsistencies. 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 keys to the tables.
- 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.

## Procedure

- Step 1** Log in to the DDM console. In the instance list, locate the instance that you want to configure shards for and click its name.
- Step 2** On the displayed page, choose **Schemas** to view schemas of the DDM instance.
- Step 3** In the schema list, locate the schema that you want to configure shards for and click **Configure Shards** in the **Operation** column.
- Step 4** On the **Configure Shards** page, configure the required parameters and click **Test Availability**.

### NOTE

- Tables without primary keys do not support shard configuration.
- **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.
- You can increase data nodes or shards. Data will be redistributed across all shards if one or more shards are added.
- Existing instances are selected by default in the data node list, but you still need to input the required password for testing connections.
- The number of physical shards per data node in the schema cannot exceed 64. If more than 64 shards are required, contact DDM technical support.

- Step 5** After the test is successful, click **Next** to go to the **Precheck** page.

**Figure 8-4** Pre-check

Recheck

All of the checks shown here must be complete before you continue.

Item	Result
Binlog retention period of associated DB instances	✔ Completed
Binlog of associated DN instances must open	✔ Completed
Expire time of binlog in associated DN instances	✔ Completed
Consistency of data in broadcast tables	✔ Completed
Character sets and collations of source shards	✔ Completed
Physical table creation statement	✔ Completed
Primary key	✔ Completed
DB instance link	✔ Completed
DB instance parameter	✔ Completed
DB instance disk space <span style="font-size: 0.8em;">?</span>	✔ Completed
DB instance time zone	✔ Completed

### 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 6** After all check items are complete, click **Configure shards**.



- Step 7** View progress at the Task Center or run command **show migrate status** on your SQL client to view progress. A shard configuration task consists of two phases: full migration and incremental migration.

**Figure 8-5** Run the required command to view task progress

```
mysql> show migrate status\G
***** 1. row *****
          ID: 1
      SOURCE_RDS: localhost:33061
      MIGRATE_ID: 97e763c12cd4596a68f1430def172d1
SUCCEED_TABLE_STRUCTURE: 0
  TOTAL_TABLE_STRUCTURE: 48
    SUCCEED_TABLE_DATA: 0
      TOTAL_TABLE_DATA: 4
    SUCCEED_INDEX_DATA: 0
      TOTAL_INDEX_DATA: 12
    FULL_SUCCEED_COUNT: 12
      FULL_TOTAL_COUNT: 76
      FULL_PERCENTAGE: 15.79
      LEFT_INCREMENT: -1
1 row in set (0.01 sec)
```

**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 scale-out task ID.

**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 the 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.

- Step 8** At the Task Center, click **View Log** to view task logs.

**Step 9** 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 new RDS for MySQL instances are 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 10** Click **Clear** in the **Operation** column to delete the data migrated from original RDS for MySQL instances.

**Step 11** Carefully read information in the dialog box, confirm that the task is correct, and click **Yes**.

**Step 12** Wait till the source data is cleared.

**Step 13** 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 Node Management

---

## 9.1 Overview

Managing data nodes is managing RDS for MySQL or GaussDB(for MySQL) instances that are associated with your DDM instance. You can view the instance status, storage, class, and read weight, configure read weights, and create read replicas on the data node management page.

You can set read weights for multiple data nodes in the list at the same time. If a data node has no read replicas, you cannot set read weights for its primary RDS instance.

If your DDM instance is associated with multiple data nodes, you can set a read weight for the first data node and then click **Synchronize** to synchronize the settings of the first node to other nodes with the same read replicas. If each data node has different read replicas, set the weights manually.

## 9.2 Configuring Read Weights

### Prerequisites

You have logged in to the DDM console.

### Scenarios

If one DDM instance is associated with multiple data nodes, you can synchronize read weight settings of the first data node to other data nodes.

### Procedure

- Step 1** In the instance list, locate the DDM instance whose data nodes you want to configure read weights for, and click its name.
- Step 2** Choose **Data Nodes** in the left navigation pane and click **Configure Read Weight**.
- Step 3** In the displayed **Configure Read Weight** dialog box, set the required parameters and click **OK**.

- Step 4** Wait the request to configure read weights is submitted.
  - Step 5** Check whether read weights of read replicas are updated.
- End

## 9.3 Synchronizing Data Node Information

### Prerequisites

You have logged in to the DDM console.

### Scenarios

Synchronize data node changes to DDM after you add or delete a read replica, change the connection address, port number, security group, or instance class, delete a DB instance, or migrate a database from one enterprise project to another.

### Procedure

- Step 1** In the instance list, locate the DDM instance whose data node changes you want to synchronize.
  - Step 2** Choose **Data Nodes** in the left navigation pane and click **Synchronize Data Node Information**.
  - Step 3** Wait till a message is returned, indicating that the request to synchronize data node information is submitted.
- End

## 9.4 Reloading Table Data

### Prerequisites

You have logged in to the DDM console.

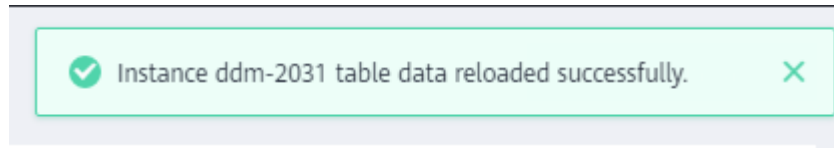
### Scenarios

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

### Procedure

- Step 1** Choose **Instances** on the left navigation pane, in the instance list, locate the instance whose information is changed and click the instance name.
- Step 2** Choose **More > Reload Table Data** in the **Operation** column.  
A message is returned, indicating that table data of instance *XXX* has been reloaded.

**Figure 9-1** Table data reloaded successfully



----End

# 10 Account Management

---

## 10.1 Creating an Account

### Prerequisites

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

### Procedure

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

**Figure 10-1** Creating a DDM account

**Create Account**

\* Username  ?

\* Password

\* Confirm Password

Schema

\* Permissions  All  
 CREATE  DROP  ALTER  INDEX  INSERT  DELETE  
 UPDATE  SELECT

Description  0/256

**Table 10-1** Required parameters

Parameter	Description
Username	Username of the account. The username can consist of 1 to 32 characters and must start with a letter. Only letters, digits, and underscores ( _ ) are allowed.
Password	Password of the account. The password: <ul style="list-style-type: none"> <li>• Can include 8 to 32 characters.</li> <li>• The password must contain at least three types of lowercase letters, uppercase letters, digits, and special characters ~! @#%^*_-=+?</li> <li>• Do not use weak or easy-to-guess passwords.</li> </ul>
Confirm Password	None
Schema	Schema to be associated with the account. You can select an existing schema from the drop-down list. The account can be used to access only the associated schemas.

Parameter	Description
Permissions	Options: <b>CREATE</b> , <b>DROP</b> , <b>ALTER</b> , <b>INDEX</b> , <b>INSERT</b> , <b>DELETE</b> , <b>UPDATE</b> , and <b>SELECT</b> . You can select any or a combination of them.
Description	Description of the account, which cannot exceed 256 characters.

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

----End

## 10.2 Modifying an Account

### Prerequisites

You have logged in to the DDM console.

### Procedure

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

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

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

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

**Step 5** Click **OK**.

----End

## 10.3 Deleting an Account

### Prerequisites

You have logged in to the DDM console.

#### NOTE

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

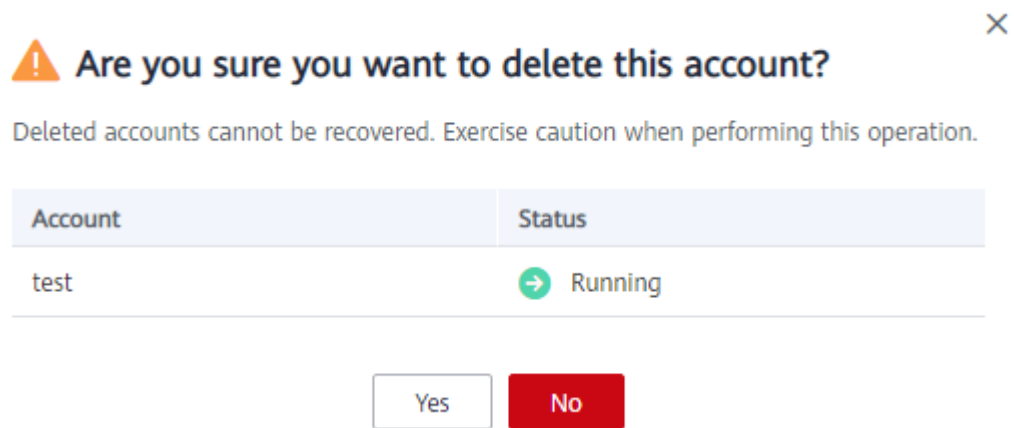
### Procedure

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



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

**Figure 10-2** Deleting an account



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

## 10.4 Resetting the Password of an Account

### Prerequisites

You have logged in to the DDM console.

### Procedure

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

**Figure 10-3** Resetting the password of an account

**Reset Password**

Account Name

\* Password

\* Confirm Password

**OK** Cancel

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

----End

## 10.5 Managing Permissions

### 10.5.1 Account Permissions

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 in associated RDS instances, and will not be synchronized to RDS instances either.

### 10.5.2 Account Requirements

#### Account

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

#### Username

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

## Password

- Contains 8 to 32 characters.
- Must contain at least three of the following character types: letters, digits, and special characters ~!@#%^\*\_-=+?
- Cannot be a weak password that is easily guessed.

## 10.5.3 Managing Permissions

### Permission Levels

- User level (supported)
- Database level (supported)
- Table level (supported)
- Column level (not supported)
- Subprogram 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
FILE	Uploading database permissions from a file
CREATE USER	Creating a user

## 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 'auth#string'
```

Example: **Creating an account (username: Jenny; password: Abc\_123456)**

```
CREATE USER Jenny IDENTIFIED BY 'Abc_123456';
```

### 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 Abc\_1234567**

```
SET PASSWORD FOR 'Jenny'@'%' = 'Abc_1234567'
```

### 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 only table-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 'your#password';  
GRANT SELECT, INSERT ON *.* to Mike;
```

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

```
GRANT SELECT, INSERT ON *.* to Mike IDENTIFIED BY 'your#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 'your#password';  
GRANT SELECT ON testdb.* to david;
```

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

```
GRANT SELECT ON testdb.* to david IDENTIFIED BY 'your#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 'your#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.

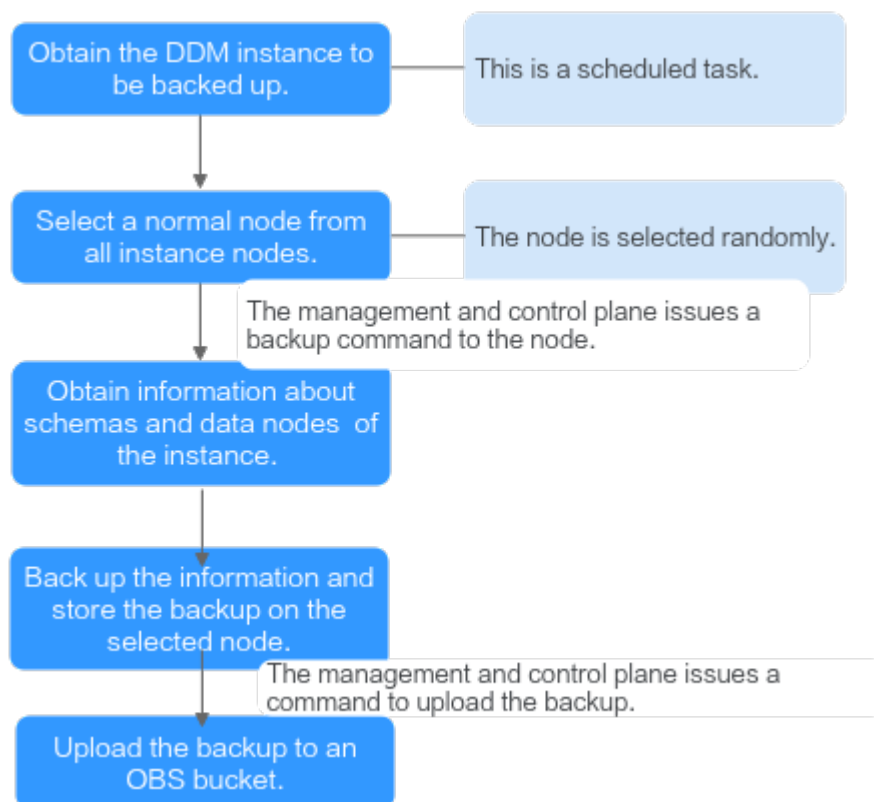
```
mysql> show grants for david;  
+-----+  
|Grants for david      |  
+-----+  
|GRANT USAGE ON *.* TO david |  
+-----+  
1 row in set (0.00 sec)
```

# 11 Backups and Restorations

## 11.1 Overview

DDM instances cannot be backed up manually. The system 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.

Figure 11-1 Backup flowchart



## 11.2 Consistent Backups

DDM replaced the consistent backup with a new feature called "metadata restore" under Data Restoration at the end of February 2022. This new feature will be released on a different schedule in different regions. After the release, the **Consistent Backups** tab is hidden, and existing consistent backups are not available for restoration any longer. If you have any questions, contact DDM technical support.

## 11.3 Restoring Data to a New Instance

### Overview

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.

**Figure 11-2** Restore to New Instance



### NOTICE

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

### Restrictions

- Restoring data to a new DDM instance will overwrite data on it and cause the instance to be unavailable during restoration.
- Ensure that no schemas are created or deleted between the restoration time and the current time.
- The new RDS for MySQL DB instances must have the same or later versions than the original ones, and their storage space must be greater than or equal to that of the original ones.
- Data cannot be restored to a local RDS for MySQL instance.



- Data cannot be restored to an RDS for MySQL instance that uses SSDs for storage.
- Restoration is not supported if the DDM instance is in the primary network and the destination RDS for MySQL instance is in the extended network.

## Prerequisites

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

## Procedure

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

### 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 2** Create as many RDS for MySQL instances as there are in the source DDM instance.

### NOTE

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

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

**Step 4** In the navigation pane on the left, choose **Backups & Restorations**. Click **Restore to New Instance**.

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

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

----End

## 11.4 Restoring Metadata

### Overview

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 30 days.

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 instance that has PITR

completed to re-establish the relationship between your DDM instance and RDS instance. The metadata restoration supports only RDS for MySQL.

## Constraints

- The destination DDM instance is not associated with any RDS for MySQL instance and has no schemas and accounts.
- Ensure that the selected RDS for MySQL instance has PITR completed.
- Restoration is not supported if the DDM instance is in the primary network and the destination RDS for MySQL instance is in the extended network.

## Prerequisites

- You have logged in to the DDM console.
- The target DDM instance is available and running normally.

## Procedure

- Step 1** Buy a new DDM instance. For details, see [Buying a DDM Instance](#).
- Step 2** In the DDM instance list, locate the newly-created instance and click its name.
- Step 3** In the navigation pane on the left, choose **Backups & Restorations**. Click **Restore Metadata**.
- Step 4** Specify a time point. DDM will select an appropriate DDM metadata backup closest to the time point.
- Step 5** Select the destination instance created in [Step 1](#). If there are no instances available, click **Create DDM Instance** to create one.
- Step 6** Select the RDS for MySQL instance that has PITR completed and click **OK**.
- Step 7** Check for a message that indicates the request is submitted.

----End

# 12 Data Migration

---

## 12.1 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).

### 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.

DDM supports the following migration methods:

- Using official MySQL clients. The following passages will use an RDS MySQL DB instance as an example to describe this method.
- Using DRS

Data migration is involved in the following scenarios:

1. [Scenario 1: Migrating Data from HUAWEI CLOUD RDS to DDM](#)
2. [Scenario 2: Migrating Data from On-Premises RDS MySQL Databases to DDM](#)
3. [Scenario 3: Migrating Data from Third-Party RDS MySQL Databases to DDM](#)
4. [Scenario 4: Migrating Data from Self-Built RDS MySQL Databases to DDM](#)
5. [Scenario 5: Migrating Data from Heterogeneous Databases to DDM](#)
6. [Scenario 6: Exporting Data from a DDM Instance](#)

 **NOTE**

- Data migration is complicated and should be done 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, submit a service ticket or contact technical support to perform a rehearsal before formal migration.
- You can access DDM only using an ECS. Therefore, export databases as files, upload the files to the ECS, and import the data in the files to DDM from the ECS.

## 12.2 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 12-1** Evaluation and preparation before data migration

Item	Description
Amount of the data to be migrated and class of the required DDM instance and RDS MySQL DB instance	<ul style="list-style-type: none"> <li>• Use vertical sharding, then horizontal sharding to shard a source RDS MySQL DB instance.</li> <li>• Execute the following SQL statement to evaluate storage space occupied by the source RDS MySQL DB instance:  <pre>select concat(round(sum(DATA_LENGTH/1024/1024), 2),'MB')as data from information_schema.TABLES;</pre> </li> <li>• Partition a table with more than 10 million rows (or expected to exceed 10 million rows).</li> <li>• Ensure that data stored on a single RDS DB instance does not exceed 500 GB.</li> </ul>
Information about schemas and logical tables mapped to each table in the source database	<ul style="list-style-type: none"> <li>• 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 DB instances.</li> <li>• If there is an RDS MySQL DB instance available in the destination DDM instance and the target schema is unsharded, associate the DB instance with the schema without the need to migrate table structures and data.</li> </ul>

Item	Description
Compatibility	<ul style="list-style-type: none"> <li>● Check whether the source database uses the same MySQL version as the target MySQL DB instance associated with the destination DDM instance.</li> <li>● The class and storage space of the destination instance cannot be less than those of the source database.</li> <li>● Check whether the table structure and character set of the source database are the same as those of the destination instance.</li> <li>● 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.</li> <li>● If a table contains a huge number of data records, export the data in batches. Specify the WHERE condition on mysqldump to limit the number of data records to be exported at a time.</li> <li>● SQL text files imported into DDM can only contain standard DML INSERT statements.</li> <li>● Evaluate the compatibility of SQL statements in your application with DDM.</li> </ul>

Before migrating data, collect the required information listed in [Table 12-2](#).

**Table 12-2** Information to be collected

Source/Destination	Information Item
Source RDS MySQL DB 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 target DB instance
	Connection address of the target DB instance
	Listening port of the target RDS DB instance

Source/Destination	Information Item
	Administrator of the RDS MySQL DB instance
	Database name
ECS	EIP
	Login credentials (username and password)

## 12.3 Scenario 1: Migrating Data from Huawei Cloud RDS to DDM

[Migrating Data from RDS for MySQL to DDM Using DRS](#)

## 12.4 Scenario 2: Migrating Data from an On-Premises RDS Instance for MySQL to DDM

### Scenarios

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

---

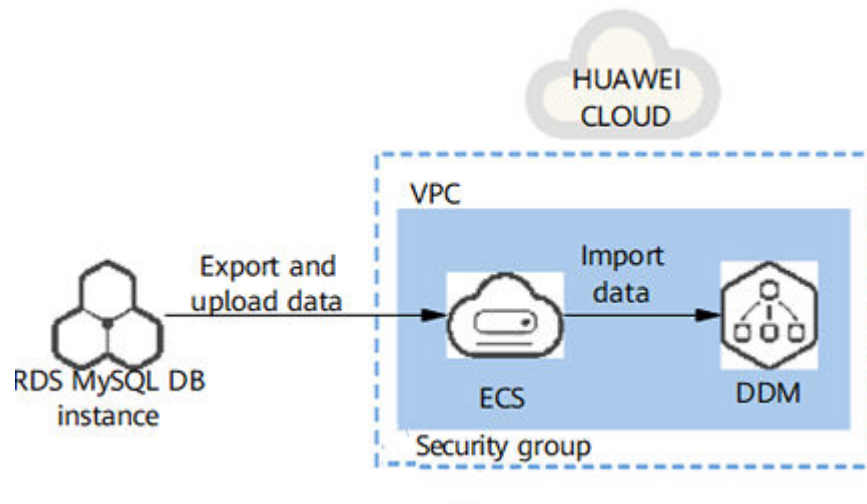
**NOTICE**

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 12-1 Migrating data from an on-premises MySQL instance to DDM



## Constraints

- The destination DDM instance can communicate with the ECS where the target 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 various types of logical tables are described in [Table 12-4](#).
- The target RDS for MySQL instance is of the same MySQL version as the source on-premises DB instance.

## Prepare for the Migration

- Prepare an ECS that can access the source DB instance.
  - a. Ensure that the source on-premises DB instance can communicate with the destination DDM instance and target DB instance.
  - 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 create the required DDM accounts, schemas, and logical tables.
  - a. Apply for a DDM instance and create a DDM account and schema on the DDM console.

- b. Export table structures of the source DB 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 12-3** describes the parameters in the command.

**Table 12-3** 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.



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_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 <i>schema name_logical table name_schema</i> to prevent data from being overwritten, for example, <b>mysql_table_schema.sql</b> .

 NOTE

The parameters listed above are generally used for data export. Not all mysqldump parameters are listed here. Log in to the official MySQL website or contact the DDM administrator to optimize certain parameters.

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 source DB database.

**Table 12-4** 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.
Sharded	Broadcast	Specify a broadcast table 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 broadcast table, see documents about CREATE TABLE statement.	
Sharded	Unsharded	Obtain the SQL statement for creating original tables, connect to DDM, and execute it on the corresponding schema.	
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 source DB instance using an IP address and export data from the instance using mysqldump.

- Prerequisites

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

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

**Step 1** Stop the service system of the source DB 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 RDS 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 DB instance contains multiple schemas, export data from each schema separately.

**Table 12-5** describes the parameters in the above command.

**Table 12-5** 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).	-

Parameter	Description	Remarks
--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.	-
--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.

Parameter	Description	Remarks
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 <i>schema name_logical table name_data</i> to prevent data from being overwritten, for example, <b>mysql_table_data.sql</b> .

 **NOTE**

- The parameters listed above are generally used for data export. Not all mysqldump parameters are listed here. Log in to the official MySQL website or contact the DDM administrator to optimize certain parameters.
- Ensure that your MySQL client has the same MySQL version as the target DB 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 failed. Contact DDM technical support.

**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** Use a MySQL client to connect to the target DDM instance and run the following commands to import structure and data files of unsharded or common tables:

```
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 DDM instance.
- **DDM\_PORT** indicates the port number of the DDM instance.
- **DDM\_USER** indicates the username for logging in to the DDM instance.
- **DB\_NAME** indicates the name of the schema. If data of unsharded tables is to be imported, **DB\_NAME** indicates 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** For sharded tables or broadcast tables, use a MySQL client to connect to DDM and import data files.

```
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 DDM instance into which data is imported.
- **DDM\_PORT** indicates the listening port.
- **DDM\_USER** indicates the DDM account.
- **DB\_NAME** indicates the name of the target 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 target DB 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 DB instance and destination DDM instance to check whether the number of records in each table is the same. **{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 no, contact DDM technical support.

**Step 3** Verify in an E2E manner whether your application's read-only access is normal.

**Step 4** Disable read-only access.

----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 source DB instance and contact DDM technical support.

## 12.5 Scenario 3: Migrating Data from a Third-Party RDS for MySQL Instance to DDM

### Scenarios

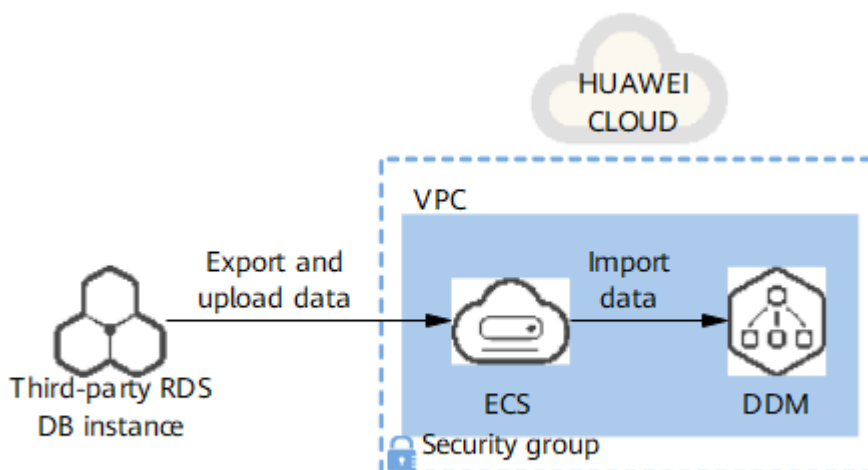
You are using a third-party RDS for MySQL instance and want to use DDM for distributed data storage.

#### 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 12-2** Migrating data from a third-party RDS for MySQL instance to DDM



### Constraints

- The destination DDM instance can communicate with the ECS where the target 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 various types of logical tables are described in [Table 12-7](#).
- The target DB instance is of the same version as the source DB instance.

### Prepare for the Migration

- Prepare an ECS that can access the source DB instance.



- a. Ensure that the ECS can access the source DB instance, destination DDM instance, and target DB instance. 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 create the required DDM accounts, schemas, and logical tables.
    - a. Apply for a DDM instance and create a DDM account and schema on the DDM console.
    - b. Export table structures of the source DB 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 12-6** describes the parameters in the command.

**Table 12-6** 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.

Parameter	Description	Remarks
--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 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 <i>schema name_logical table name_schema</i> to prevent data from being overwritten, for example, <b>mysql_table_schema.sql</b> .

 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 or contact DDM technical support.

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 structure of the source DB instance.

**Table 12-7** 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, and execute the new SQL statement on the corresponding schema. 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, and execute it on the corresponding schema.	
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 source DB instance using an IP address and export data from the instance using mysqldump.

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

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

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

**Step 2** Export table data from the source DB 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 source DB instance contains multiple schemas, export data from each schema separately.
- If the source DB instance does not support table data export using mysqldump, contact the administrator.

**Table 12-8** describes the parameters in the above command.

**Table 12-8** 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.
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 <i>schema name_logical table name_data</i> to prevent data from being overwritten, for example, <b>mysql_table_data.sql</b> .
--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 commonly used for data export. Not all mysqldump parameters are listed here. To optimize certain parameters, log in to the official MySQL website or contact DDM technical support.

- 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 failed. Contact 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** Use a MySQL client to connect to the target DDM instance and run the following commands to import structure and data files of unsharded or common tables:

```
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 DDM instance.
- **DDM\_PORT** indicates the port number of the DDM instance.
- **DDM\_USER** indicates the username for logging in to the DDM instance.
- **DB\_NAME** indicates the name of the schema. If data of unsharded tables is to be imported, **DB\_NAME** indicates 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** For sharded or broadcast tables, connect to DDM on your MySQL client and import the data file.

```
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 into which data is imported.
- **DDM\_PORT** indicates the listening port of the destination DDM instance.
- **DDM\_USER** indicates the DDM account.
- **DB\_NAME** indicates the name of the target 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 target DB 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 DB instance and destination DDM instance to check whether the number of records in each table is the same. `{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 no, contact DDM technical support.

**Step 3** Verify in an E2E manner whether your application's read-only access is normal.

**Step 4** Disable read-only access.

----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 services to the source DB instance and contact DDM technical support.

## 12.6 Scenario 4: Migrating Data from a Self-Built MySQL Instance to DDM

### Scenarios

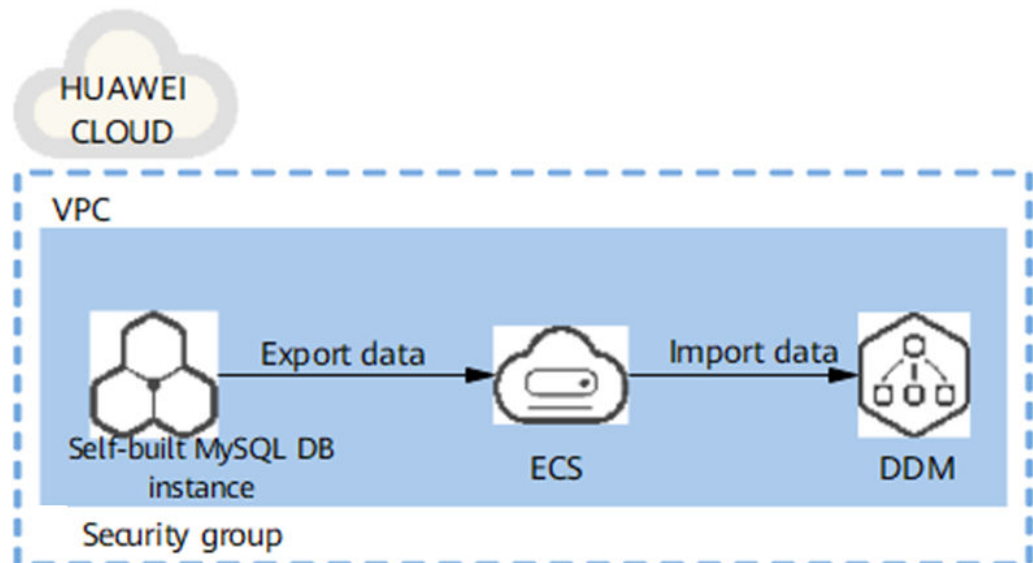
You have built a MySQL instance on an ECS and want to migrate your data from the instance to DDM for distributed data storage.

 **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 12-3 Migrating data from a self-built MySQL instance to DDM



### NOTE

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

## Constraints

- The ECS, destination DDM instance, and target DB instance are 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 12-10](#) describes the methods for creating various types of logical tables.
- The target DB instance must be of the same MySQL version as the source DB instance.

## Prepare for the Migration

- Prepare an ECS that can access the source DB instance.
  - a. Ensure that the ECS where the source DB instance is located, destination DDM instance, and target DB instance are in the same VPC.
  - b. Configure the same security group for the ECS, destination DDM instance, and target DB 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 create the required DDM accounts, schemas, and logical tables.
  - a. Apply for a DDM instance and create a DDM account and schema on the DDM console.
  - b. Export table structures of the source DB 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 12-9** describes the parameters in the command.

**Table 12-9** 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.

Parameter	Description	Remarks
--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 <i>schema name_logical table name_schema</i> to prevent data from being overwritten, for example, <b>mysql_table_schema.sql</b> .

 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 or contact DDM technical support.

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 DB instance.

**Table 12-10** 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, and execute the new SQL statement on the corresponding schema. 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, and execute it on the corresponding schema.	
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 source DB instance to a separate SQL text file.

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

**Step 2** Export table data from the source DB 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 source DB instance contains multiple schemas, export data from each schema separately.

**Table 12-11** describes the parameters in the above command.

**Table 12-11** 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.
--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.	-
--hex-blob	Exports binary string fields in hexadecimal format. This parameter is mandatory if there is binary data to be exported.	-

Parameter	Description	Remarks
--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.
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 <i>schema name_logical table name_data</i> to prevent data from being overwritten, for example, <b>mysql_table_data.sql</b> .

 NOTE

- The parameters listed above are generally used for data export. Not all mysqldump parameters are listed here. Log in to the official MySQL website or contact the DDM administrator to optimize certain parameters.
- Ensure that your MySQL client has the same MySQL version as the target DB 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 failed. 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** Use a MySQL client to connect to the target DDM instance and run the following commands to import structure and data files of unsharded or common tables:

```
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 DDM instance.
- **DDM\_PORT** indicates the port number of the DDM instance.
- **DDM\_USER** indicates the username for logging in to the DDM instance.
- **DB\_NAME** indicates the name of the schema. If data of unsharded tables is to be imported, **DB\_NAME** indicates 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** For sharded or broadcast tables, connect to DDM on your MySQL client and import the data file.

```
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 into which data is imported.



- **DDM\_PORT** indicates the listening port of the destination DDM instance.
- **DDM\_USER** indicates the DDM account.
- **DB\_NAME** indicates the name of the target 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 target DB 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 source DB instance and destination DDM instance to check whether the number of records in each table is the same. **{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 no, contact DDM technical support.

**Step 3** Verify in an E2E manner whether your application's read-only access is normal.

**Step 4** Disable read-only access.

----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 source DB instance and contact DDM technical support.

## 12.7 Scenario 5: Migrating Data from Heterogeneous Databases to DDM

For details about how to migrate data from a heterogeneous database such as Oracle, PostgreSQL, and SQL Server, see the *Cloud Data Migration User Guide* or contact DDM technical support.

## 12.8 Scenario 6: Exporting Data from a DDM Instance

---

 CAUTION

- Export table data during off-peak hours. This is because performance of the DDM instance and its associated RDS 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 **nohup {mysqldump CLI} &** in Linux to ensure that mysqldump still works when a session times out.
- 

## Scenarios

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

## Export Table Structure

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

- 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>.

# 13 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.

## Procedure

- Step 1** In the instance list, locate the DDM instance whose slow queries you want to view and click its name.
- Step 2** In the navigation pane, choose **Slow Queries**.
- Step 3** On the **Slow Queries** page, specify a time range and view SQL statements executed within this time range.

----End

# 14 Monitoring Management

## 14.1 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

#### NOTE

SYS.DDM is the namespace of DDM 1.0.

SYS.DDMS is the namespace of DDM 2.0.

DDM has been upgraded to version 2.0. The namespace is still SYS.DDM for existing users of DDM1.0.

### Metrics

**Table 14-1** 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_memory_util	Memory Usage	Memory usage of the DDM instance node.	0—100	DDM nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
ddm_bytes_in	Network Input Throughput	Incoming traffic per second of the DDM instance node	$\geq 0$	DDM nodes	1 minute
ddm_bytes_out	Network Output Throughput	Outgoing traffic per second of the DDM instance node	$\geq 0$	DDM nodes	1 minute
ddm_qps	QPS	Requests per second of the DDM instance node	$\geq 0$	DDM nodes	1 minute
ddm_read_count	Reads	Read operations of the DDM instance node within each monitoring period	$\geq 0$	DDM nodes	1 minute
ddm_write_count	Writes	Write operations of the DDM instance node within a monitoring period	$\geq 0$	DDM nodes	1 minute
ddm_slow_log	Slow SQL Logs	Slow SQL logs of DDM-Core	$\geq 0$	DDM nodes	1 minute
ddm_rt_avg	Average Response Latency	Average response latency of DDM-Core	$\geq 0$	DDM nodes	1 minute
ddm_connections	Connections	Connections of DDM-Core	$\geq 0$	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
active_connections	Active connections	Active connections of each DDM instance node	$\geq 0$	DDM nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Interval (Raw Data)
ddm_connection_util	Connection Usage	Percentage of active connections to each DDM instance node	0-100%	DDM nodes	1 minute

## Dimensions

Key	Value
node_id	DDM nodes

### NOTE

DDM supports dimension node\_id, but not dimension instance\_id. You can obtain the ID of a node by the corresponding instance ID.

## 14.2 Viewing Metrics

### Scenarios

The DDM console supports monitoring and management of DDM instances. You can tune databases based on real-time monitoring results.

### Prerequisites

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

### Procedure

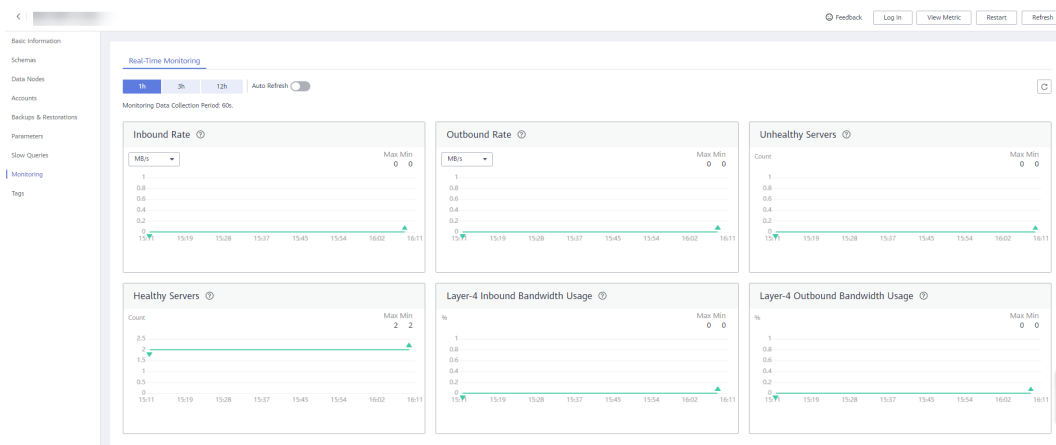
**Step 1** In the instance list, locate the DDM instance whose metrics you want to view and click its name.

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

**Step 3** Click **Real-Time Monitoring**.

If load balancing is enabled for your DDM instance, select a time range to view metrics Inbound Rate, Outbound Rate, Unhealthy Servers, Healthy Servers, Layer-4 Inbound Bandwidth Usage, and Layer-4 Outbound Bandwidth Usage in the monitoring list. If load balancing is not enabled, you have no permissions to view these metrics.

**Figure 14-1** Real-time monitoring




**Table 14-2** Real-time monitoring parameters

Parameter	Description
Inbound Rate	Traffic used for accessing the monitored object from the Internet per second
Outbound Rate	Traffic used by the monitored object to access the Internet per second
Unhealthy Servers	Number of unhealthy backend servers associated with the monitored object
Healthy Servers	Number of healthy backend servers associated with the monitored object
Layer-4 Inbound Bandwidth Usage	Percentage of inbound TCP/UDP bandwidth from the monitored object to the client
Layer-4 Outbound Bandwidth Usage	Percentage of outbound TCP/UDP bandwidth from the monitored object to the client

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

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

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

----End



## 14.3 Network Metrics

If load balancing is enabled for your DDM instance, you can view metrics in the following table. If load balancing is not enabled, you do not have the permissions to view them.

**Table 14-3** 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	$\geq 0$	Dedicated load balancer	1 minute
m8_out_Bps	Outbound Rate	Traffic used by the monitored object to access the Internet per second Unit: byte/s	$\geq 0$	Dedicated load balancer	1 minute
m9_abnormal_servers	Unhealthy Servers	Number of unhealthy backend servers associated with the monitored object Unit: count	$\geq 0$	Dedicated load balancer	1 minute
ma_normal_servers	Healthy Servers	Number of healthy backend servers associated with the monitored object Unit: count	$\geq 0$	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

<b>Metric ID</b>	<b>Metric Name</b>	<b>Description</b>	<b>Value Range</b>	<b>Monitored Object</b>	<b>Monitoring Interval (Raw Data)</b>
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

# 15 Auditing

## 15.1 Key Operations Recorded by CTS

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

**Table 15-1** DDM operations that can be recorded by CTS

Operation	Resource Type	Trace Name
Applying a parameter template	parameterGroup	applyParameterGroup
Renewing a yearly/monthly DDM instance to changing the billing mode of a DDM instance from yearly/monthly to pay-per-use or from pay-per-use to yearly/monthly	all	bssArrearage
Updating DDM metadata	all	bssUpdateMetadata
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

Operation	Resource Type	Trace Name
Deleting a DDM instance	instance	deleteInstance
Deleting a schema	logicDB	deleteLogicDB
Deleting a parameter template	parameterGroup	deleteParameterGroup
Deleting an account	user	deleteUser
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
Changing node class	instance	resizeFlavor
Restoring DB instance data	instance	restoreInstance
Retrying to scale out a schema	logicDB	retryMigrateLogicDB
Rolling back the version upgrade of a DDM instance	instance	rollback
Rolling back a schema scaling task	logicDB	rollbackMigrateLogicDB
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
Creating a node group	group	createGroup

Operation	Resource Type	Trace Name
Modifying the floating IP address	instance	modifyIp

## 15.2 Querying Traces

### Scenarios


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

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

#### NOTE

Before using CTS, you need to enable it.

### Procedure

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

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

----End

# 16 SQL Syntax

## 16.1 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.

### MySQL 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 16-1** Description of the **EXPLAIN** column

Column Name	Description
table	Table that the row of data belongs to
type	Type of the connection. Connection types from the best to the worst are <b>const</b> , <b>eq_reg</b> , <b>ref</b> , <b>range</b> , <b>index</b> , and <b>ALL</b> .
possible_keys	Index that may be applied to the table
key	Index that is actually used. If the value is <b>NULL</b> , no index is used. In some cases, MySQL may choose to optimize indexes, for example, force MySQL to use an index by adding <b>USE INDEX(indexname)</b> to a SELECT statement or to ignore an index by adding <b>IGNORE INDEX(indexname)</b> .
key_len	Length of the used index. The shorter the length is, the better the index is if accuracy is not affected.

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

## SQL Restrictions

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

## DDL Syntax

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

## DML Syntax

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



- STRAIGHT\_JOIN and NATURAL JOIN are not supported.
- Multiple-table UPDATE is supported if all tables joined across shards have primary keys.
- Multiple-table DELETE is supported if all tables joined across shards have primary keys.
- Using or manipulating variables in SQL statements is not supported, for 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.

## 16.2 DDL

## 16.2.1 Overview

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

### DDL Statements that Can Be Executed on a MySQL Client

- TRUNCATE

Example:

```
TRUNCATE TABLE t1
```

Deletes all data from table t1.

TRUNCATE TABLE is used to delete all data from a table and has the DROP permission. In logic, TRUNCATE TABLE is similar to the DELETE statement for deleting all data from a table.

- ALTER TABLE

Example:

```
ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
```

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 or comments of a table.

- DROP INDEX

Example:

```
DROP INDEX `PRIMARY` ON t;
```

Deletes primary key from table t.

DROP INDEX can delete index *index\_name* from table *tbl\_name*.

- CREATE INDEX

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.

## 16.2.2 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.

## 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 tpartition_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) tpartitions 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)
);
```

## 16.2.3 Sharding Algorithm Overview

### Supported Sharding Algorithms

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

**Table 16-2** 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	Performing a simple modulo operation	Yes	Yes

Algorithm	Description	Database Sharding Supported	Table Sharding Supported
RANGE	Performing a RANGE-based operation	Yes	No
RIGHT_SHIFT	Arithmetic right shifting of 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 16-3** 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	×	×	×	×	×	×	×	×	√	√	√	×
MMD	×	×	×	×	×	×	×	×	√	√	√	×
WEEK	×	×	×	×	×	×	×	×	√	√	√	×

## 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]
]
```

## 16.2.4 Sharding Algorithms

### 16.2.4.1 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).

## 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.

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 16-4** Required calculation methods when the sharding key is the integer data type

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

### Method 2: Use a String as the Sharding Key

**Table 16-5** Required calculation methods when the sharding key is the string data type

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

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

## Syntax for Creating Tables

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

```
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) tpartition by mod_hash(ID) tpartitions 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.

### 16.2.4.2 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).

## 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 16-6** Required calculation methods when the sharding key is the integer data type

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

### Method 2: Use a String as the Sharding Key

**Table 16-7** Required calculation methods when the sharding key is the string data type

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

## Syntax for Creating Tables

- Assume that you use field **ID** as the sharding key to shard databases based on MOD\_HASH\_CI:  

```
create table mod_hash_ci_tb(
  id int,
```

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

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

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

## Precautions

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

### 16.2.4.3 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 16-8** Required calculation methods

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

## Syntax for Creating Tables

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

## Precautions

- The number of shifts cannot exceed the number of bits occupied by the integer type.

### 16.2.4.4 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, instead of database sharding.

## Data Routing

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

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

## Calculation Method

**Table 16-9** 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: $1 \bmod 12 = 1$

## 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)
tbpartmention by MM(create_time) tpartitions 12;
```

## Precautions

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

### 16.2.4.5 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, instead of database sharding.

## Data Routing

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

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

## Calculation Method

**Table 16-10** 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: $15 \text{ mod } 31 = 15$

## Syntax for Creating Tables

```
create table test_dd_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH(id)
tbpartmention 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.

### 16.2.4.6 WEEK

## Application Scenarios

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

## Instructions

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

## Data Routing

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

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

## Calculation Method

**Table 16-11** 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: $3 \bmod 7 = 3$

## 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.

### 16.2.4.7 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.

## Instructions

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

## Data Routing

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

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

## Calculation Method

**Table 16-12** 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: $15 \% 366 = 15$

## 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)
tbpartmention by MMDD(create_time) tpartitions 366;
```

## Precautions

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

### 16.2.4.8 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  $(2012 \times 12 + 12) \% D$ . D is the number of database or table shards.

## Calculation Method

**Table 16-13** Required calculation methods

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

## 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 the two years, each month corresponding to one table. Since you already have 8 physical databases, 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) tpartitions 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.

### 16.2.4.9 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  $(2012 \times 366 + 366) \% D$ . D is the number of database or table shards.

### NOTE

"2012-12-31" is the 366th day of 2012, so the calculation is "2012 x 366".

## Calculation Method

**Table 16-14** Required calculation methods

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

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

## Syntax for Creating Tables

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 physical databases, 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) tbpertitions 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.

## 16.2.4.10 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  $(2013 \times 54 + 1) \% D$ . `D` is the number of database or table shards.

#### NOTE

2012-12-31 is the first week of 2013, so the calculation is  $2013 \times 54 + 1$ .

### Calculation Method

**Table 16-15** Required calculation methods

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

## Syntax for Creating Tables

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 physical databases, 14 ( $14 \times 8 = 112 > 106$ ) 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) tpartitions 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.

### 16.2.4.11 HASH

## Application Scenarios

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

## 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 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 16-16** Required calculation methods when the sharding key is not the DATE type

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

### Method 2: Use a Date Sharding Key

**Table 16-17** Supported date functions

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

**Table 16-18** 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(Database sharding key)) % 102400	Database/Table shard: crc32(year('2019-10-11')) % 102400 = 5404; 5404 belongs to range 0=0-12799, so data is routed to shard 0.

## Syntax for Creating Tables

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

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

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

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

## Precautions

None

### 16.2.4.12 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 16-19** 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 16-20** 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	weekofyear('2019-10-11')=41
day()	day(yyyy-MM-dd)=Day number of the current month	day('2019-10-11')=11

**Table 16-21** 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

## 16.3 DML

### 16.3.1 INSERT

INSERT is used to insert data into database objects.

#### Common Syntax

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

#### Syntax Restrictions

- INSERT DELAYED is not supported.
- Only INSERT statements that contain sharding fields are supported.
- PARTITION syntax is not supported. Partitioned tables are not recommended.
- Setting **datetime** 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, for example, INSERT INTO T(NAME) VALUES(REPEAT(ID,3)).

## 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 as functions, you may not obtain the expected results because its results depend on whether it pushed to data nodes for execution.

## 16.3.2 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 as functions, you may not obtain the expected results because its results depend on whether it pushed to data nodes for execution.

## 16.3.3 DELETE

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

### Common Syntax

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

### Syntax Restrictions

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

## 16.3.4 UPDATE

### Common Syntax

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

### Syntax Restrictions

- Subqueries are not supported, including correlated and non-correlated subqueries.
- The WHERE condition in the UPDATE statement does not support arithmetic expressions and their subqueries.
- Modifying reference tables is not supported during an update of multiple tables.
- Updating the sharding key field of a logical table is not supported because this operation may cause data redistribution.
- Setting **datetime** 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.
- 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
```
- UPDATE JOIN supports only joins with WHERE conditions.

## 16.3.5 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 16-22** Supported syntax

Syntax	Description
select_expr	Indicates a column that you want to query.
FROM table_references	Indicates the tables that you want to query.

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

## 16.3.6 SELECT JOIN Syntax

### Common Syntax

table\_references:

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

table\_reference:

```
table_factor | join_table
```

table\_factor:

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

join\_table:

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

join\_condition:

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

## Syntax Restrictions

SELECT STRAIGHT\_JOIN and NATURAL JOIN are not supported.

## Example

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

## 16.3.7 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.

## 16.3.8 SELECT Subquery Syntax

### Subquery as Scalar Operand

#### Example

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

## Comparisons Using Subqueries

### Syntax

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

### Example

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

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

### Syntax

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

### Example

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

## Derived Tables (Subqueries in the FROM Clause)

### Syntax

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

### Example

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

## Syntax Restrictions

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

## 16.3.9 Unsupported DML Statements

### Unsupported DML Statements

**Table 16-23** Syntax restrictions on DML

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

## 16.3.10 Supported System Schema Queries

**Table 16-24** Supported System Schema Queries

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

## 16.4 Functions

### Supported Functions

**Table 16-25** Operator functions

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

Expression	Example
IS NULL	SELECT product_name FROM Products WHERE product_price IS NULL
IS NOT NULL	SELECT id, product_name FROM Products WHERE product_price IS NOT NULL ORDER BY id
AND	SELECT * FROM Products WHERE vendor_id = 'V000001' AND product_price <= 4000 ORDER BY product_price
OR	SELECT * FROM Products WHERE vendor_id = 'V000001' OR vendor_id = 'V000009'
NOT	SELECT product_id, product_name FROM Products WHERE NOT vendor_id = 'V000002'
LIKE	SELECT * FROM Products WHERE product_name LIKE 'NAME %' ORDER BY product_name
NOT LIKE	SELECT * FROM Products WHERE product_name NOT LIKE 'NAME%' ORDER BY product_name
CONCAT	SELECT product_id, product_name, Concat( product_id , '(' , product_name , ')' ) AS product_test FROM Products ORDER BY product_id
+	SELECT 3 * 2+5-100/50
-	SELECT 3 * 2+5-100/50
*	SELECT order_num, product_id, quantity, item_price, quantity*item_price AS expanded_price FROM OrderItems WHERE order_num BETWEEN 000009 AND 000028 ORDER BY order_num
/	SELECT 3 * 2+5-100/50
UPPER	SELECT id, product_id, UPPER(product_name) FROM Products WHERE id > 10 ORDER BY product_id
LOWER	SELECT id, product_id, LOWER(product_name) FROM Products WHERE id <= 10 ORDER BY product_id
SOUNDEX	SELECT * FROM Vendors WHERE SOUNDEX(vendor_name) = SOUNDEX('test') ORDER BY vendor_name
IFNULL	<p>SELECT IFNULL(product_id, 0) FROM Products;</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>For DDM instances created before March 20, sharded tables do not support the calling of functions nested in the IFNULL and aggregation functions. For example, if you execute function <b>select IFNULL(sum(yan),0) from shenhai</b>, the result differs from the expected result.</li> <li>For DDM instances created after March 20, sharded tables support only the calling of functions nested in the IFNULL and aggregation functions.</li> </ul>

**Table 16-26** Time and date functions

Expression	Example	Application Scope
DAY()	<pre>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')</pre>	-
MONTH()	<pre>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')</pre>	-
YEAR()	<pre>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')</pre>	-
DAYOFYEAR()	<pre>SELECT * FROM TAB_DATE WHERE DAYOFYEAR(date)=365 SELECT * FROM TAB_DATE WHERE date='2018-12-31' INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22')</pre>	-
DAYOFWEEK()	<pre>SELECT * FROM TAB_DATE WHERE DAYOFWEEK(date)=6 SELECT * FROM TAB_DATE WHERE date='2018-12-21' INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22')</pre>	-
WEEKOFYEAR()	<pre>SELECT * FROM TAB_DATE WHERE WEEKOFYEAR(date)=51 SELECT * FROM TAB_DATE WHERE date='2018-12-21' INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22')</pre>	-



**Table 16-27** Mathematical functions

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	-

## Unsupported Functions

**Table 16-28** Function restrictions

Item	Restriction
ROW_COUNT()	Function <b>ROW_COUNT()</b> is not supported.

## 16.5 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 (<), assignment (=), and more than (>)
- Expression IS UNKNOWN
- INSTALL and UNINSTALL PLUGIN statements
- Cross-shard stored procedures and custom functions
- Statements for modifying database names, table names, and sharding field names and types
- Most of SHOW statements such as SHOW PROFILES and SHOW ERRORS
- Table maintenance statements, including ANALYZE, CHECK, CHECKSUM, OPTIMIZE, and REPAIR TABLE
- Statements for assigning a value to or querying variable **session**, for example, set @rowid=0;select @rowid:=@rowid+1,id from user
- SQL statements that use -- or /\*...\*/ to comment out a single line or multiple lines of code
- 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)

## 16.6 Supported SQL Statements

### 16.6.1 CHECK TABLE

#### 16.6.1.1 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.

## 16.6.1.2 Checking DDL Consistency of Physical Tables in One Logical Table

**Purpose:** To check DDL consistency of physical tables in a specific logical table

### Command Format:

```
check table <table_name>
```

### Command Output:

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

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

If the returned result set is not empty, there are 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.

## 16.6.2 SHOW RULE

### Command Format:

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

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

### Command Output:

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

```
mysql> show rule.
```

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

The following is an example output of **show rule from *table\_name***.

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

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

1 row in set (0.00 sec)

### Output Details:

**TABLE\_NAME:** indicates the name of the logical table.

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

**DB\_PARTITION\_KEY:** indicates the database sharding key. Leave this field blank if database sharding is not required.

**DB\_PARTITION\_POLICY:** indicates the database sharding algorithm. The value can be **HASH**, **YYYYMM**, **YYYYDD**, and **YYYYWEEK**.

**DB\_PARTITION\_COUNT:** indicates the number of database shards.

**DB\_PARTITION\_OFFSET:** indicates where a new database 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.

## 16.6.3 SHOW TOPOLOGY

### Command Format:

```
show topology from table_name; used to view physical tables corresponding to a specified logical table.
```

### 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`.

## 16.6.4 SHOW DATA NODE

### Command Format:

```
show data node; 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.

## 16.6.5 TRUNCATE TABLE

### 16.6.5.1 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.

**16.6.5.2 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)

### 16.6.5.3 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 other physical tables.

### 16.6.5.4 Additional Information

Hints are valid only for sharded tables.

### 16.6.6 HINT- ALLOW\_ALTER\_RERUN

**Command Format:**

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

**Description:**



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

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

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

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

#### NOTE

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

- You can use **CHARACTER SET** to specify a file code 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.

## 16.6.8 SHOW PHYSICAL PROCESSLIST

### Command Format 1:

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

### Command Format 2:

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

### Command Output

**Figure 16-1** 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:1world_0007	world_0007	Sleep	24373		
localhost	33062	shard2	master	5	DDMRV	localhost:1world_0007	world_0007	Sleep	24373		
localhost	33062	shard2	master	6	DDMRV	localhost:1world_0004	world_0004	Sleep	769		
localhost	33062	shard2	master	7	DDMRV	localhost:1world_0006	world_0006	Sleep	769		
localhost	33062	shard2	master	8	DDMRV	localhost:1world_0007	world_0007	Sleep	24373		
localhost	33062	shard2	master	9	DDMRV	localhost:1world_0007	world_0007	Sleep	24373		
localhost	33062	shard2	master	10	DDMRV	localhost:1world_0004	world_0004	Sleep	24373		
localhost	33062	shard2	master	11	DDMRV	localhost:1world_0005	world_0005	Sleep	769		
localhost	33062	shard2	master	12	DDMRV	localhost:1world_0007	world_0007	Query	0	starting	SHOW FULL
localhost	33062	shard2	master	13	DDMRV	localhost:1world_0004	world_0004	Sleep	24373		
localhost	33061	shard1	master	7	DDMRV	localhost:1world_0000	world_0000	Sleep	24372		
localhost	33061	shard1	master	8	DDMRV	localhost:1world_0000	world_0000	Sleep	19576		

**Output Details:**

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

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

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

**type:master:** indicates that the associated instance is a primary instance, and **readreplica** indicates that the associated instance is a read replica.

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

**Command Format 3:**

**kill physical physical\_thread\_id@rds\_ip:rds\_port:** kills the execution thread on the associated RDS instance.

**NOTICE**

This feature is available only in kernel 3.0.1 or later.

## 16.6.9 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:

- `/*!mycat:db_type=host*/`
- `/*+ 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.

## 16.6.10 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 format of `/*!GAUSS:skip_plancache=flag*/`.

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

Currently, this function only applies to SELECT statements.

## 16.6.11 Specifying a Shard Using a Hint When Executing a SQL Statement

### Command Format:

```
/*+db=<physical_db_name>*/ <your query>;
```

### Description:

Specify a shard by configuring `<physical_db_name>` and execute a SQL statement on the shard.

### Example:

```
/*+db=test_0000*/ select * from t1;
```

### Restrictions:

- The hint is valid only for SELECT, DML, and TRUNCATE statements.
- The hint works only under the text protocol, rather than the Prepare protocol.

## 16.7 Global Sequence

### 16.7.1 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.

**Table 16-29** Table types supported by global sequence

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

## Creating an Auto-Increment Sequence

**Step 1** Log in to the required DDM instance using a client.

**Step 2** Open the required schema.

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

```
create sequence xxxxx ;
```

### NOTE

- *xxxxx* indicates the sequence name.
- 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 100%, do not insert data any more and contact DDM technical support.

----End

## Dropping an Auto-Increment Sequence

**Step 1** Log in to the required DDM instance using a client.

**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 xxxxx ;
```

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

### NOTE

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

----End

## Modifying the Start Value of an Auto-Increment Sequence

**Step 1** Log in to the required DDM instance using a client.

**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 xxxxx START WITH yyyyy;
```

### NOTE

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

----End

## Querying an Auto-Increment Sequence

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

----End

## Updating Auto-Increment Sequences of All Tables

### NOTICE

This feature is available only in kernel 3.0.4.1 or later.

- Step 1** Log in to the required DDM instance using a client.

**Step 2** Run command **fresh all sequence start value** to change sequences of all schemas.

----End

## 16.7.2 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** Log in to the required DDM instance using a client.

**Step 2** Open the required schema.

**Step 3** Run the following command to create a global sequence:

```
create sequence seq_test;
```

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

**Step 4** Run the following command to obtain the next sequence number:

```
select seq_test.nextval;
```

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

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

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

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

```
select seq_test.currval;
```

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

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

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

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



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

**NOTE**

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

----End

## 16.7.3 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.

### Example

There are two schemas **dml\_test\_1** and **dml\_test\_2**, and both of them have table **test\_seq**.

### Table Definition

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

### Procedure

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

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

```
mysql> use dml_test_1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table test_seq(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:

- `use dml_test_1;`

`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)
```

- `use dml_test_2;`

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

```
mysql> use dml_test_2;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table test_seq(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

## 16.8 Database Management Syntax

### Supported Database Management Syntax

- SHOW Syntax
- SHOW COLUMNS Syntax
- SHOW CREATE TABLE Syntax
- SHOW TABLE STATUS Syntax
- SHOW TABLES Syntax
- SHOW DATABASES

If the required database is not found, check fine-grained permissions of your account.

- SHOW INDEX FROM
- SHOW VARIABLES Syntax

### Supported Database Tool Commands

- DESC Syntax
- USE Syntax
- EXPLAIN Syntax

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

### Unsupported Database Management Syntax

**Table 16-30** Restrictions on database management statements

Item	Restriction
Database management statements	<ul style="list-style-type: none"> <li>• Executing SET Syntax to modify global variables is not supported.</li> <li>• SHOW TRIGGERS is not supported.</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• SHOW TABLE STATUS</li> <li>• SHOW VARIABLES Syntax</li> <li>• CHECK TABLE does not support sharding tables by hash or sharding key.</li> <li>• SHOW WARNINGS Syntax does not support the combination of LIMIT and COUNT.</li> <li>• SHOW ERRORS Syntax does not support the combination of LIMIT and COUNT.</li> </ul>

## 16.9 Advanced SQL Functions

**Table 16-31** Restrictions on advanced SQL functions

Item	Restriction
SQL functions	<ul style="list-style-type: none"> <li>• PREPARE and EXECUTE syntax is not supported.</li> <li>• Customized data types and functions are not supported.</li> <li>• Views, stored procedures, triggers, and cursors are not supported.</li> <li>• Compound statements such as BEGIN...END, LOOP...END LOOP, REPEAT...UNTIL...END REPEAT, and WHILE...DO...END WHILE are not supported.</li> <li>• Process control statements such as IF and WHILE are not supported.</li> <li>• The following prepared statements are not supported: <b>PREPARE</b> Syntax <b>EXECUTE</b> Syntax</li> <li>• Comments for indexes are not supported in table creation statements.</li> </ul>

# 17 Quotas


## Scenarios

Quotas are enforced for service resources on the platform to prevent unforeseen spikes in resource usage. Quotas limit the number or amount of resources available to users.

If a quota cannot meet your needs, apply for a higher quota.

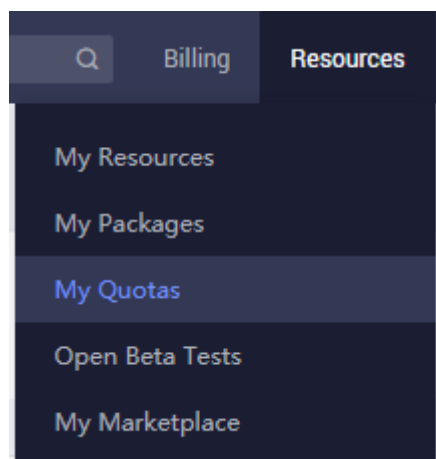
## Viewing Quotas

**Step 1** Log in to the management console.

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

**Step 3** In the upper right corner, choose **Resources** > **My Quotas**.

Figure 17-1 My Quotas




**Step 4** View the used and total quota of each type of DDM resource.

**Step 5** If the quotas cannot meet service requirements, click **Increase Quota** to adjust it.

----End

## Increasing Quotas

**Step 1** Log in to the management console.

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

**Step 3** In the upper right corner, choose **Resources > My Quotas**.

**Step 4** Click **Increase Quota**.

**Step 5** On the **Create Service Ticket** page, configure parameters as required.

In the **Problem Description** area, enter the required quota and reason for the adjustment.

**Step 6** After all necessary parameters are configured, select the agreement and click **Submit**.

----End

# A Change History

---

Released On	Description
2022-09-30	This is the first official release.