分布式数据库中间件

用户指南

文档版本 01

发布日期 2021-10-13

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 产品介绍

1.1 产品概述

分布式数据库中间件（Distributed Database Middleware，简称DDM），是一款分布式关系型数据库。它兼容MySQL协议，采用存储计算分离架构的模式，使得存储层、计算层可以无限扩展，从而拥有了海量数据高并发访问能力。

实现原理

DDM在计算、数据存储两个方面的可扩展性，打破业务发展中因数据库扩展性及运维压力所产生的困扰。

产品优势

分布式数据库中间件具有简单易用、快速部署、低成本等优势。

- **高可扩展**

  DDM支持自动水平拆分。

  - 分钟级规格变更。
  - 节点扩容对应用透明。
  - 业务不中断平滑扩容。

- **简单易用**

  - 兼容MySQL协议、语法、客户端。
  - 轻松数据导入，数据库上云。
  - 业务零代码改动，实现读写分离。

- **快速部署**

  可在线快速部署实例，节省采购、部署、配置等自建数据库工作，缩短项目周期，帮助业务快速上线。

- **低成本**

  稳定的产品，完善的运维和技术支持，相比开源产品总体性价比更高；多种实例规格配置覆盖不同业务规模场景，按需申请。

1.2 常用概念

数据库实例

数据库实例是分布式数据库中间件服务的最小管理单元。一个实例代表了一个独立运行的数据库。您可以在一个DDM实例中通过创建多个逻辑库管理多个MySQL实例，并且可以独立访问MySQL实例。使用管理控制台或API可以方便的创建和修改DDM实例。

说明

本文中的MySQL实例包括RDS for MySQL实例和GaussDB（forMySQL）实例两种。

虚拟私有云

虚拟私有云（Virtual Private Cloud）是用户在云服务上申请的隔离的、私密的虚拟网络环境。用户可以自由配置VPC内的IP地址段、子网、安全组等子服务，也可以申请弹性带宽和弹性IP搭建业务系统。

子网

子网是虚拟私有云内的IP地址块。虚拟私有云中的所有云资源都必须部署在子网内。同一个虚拟私有云下，子网网段不可重复。子网创建成功后，网段无法修改。

安全组

安全组是一个逻辑上的分组，为具有相同安全保护需求并相互信任的云服务器提供访问策略。安全组创建后，用户可以在安全组中定义各种访问规则，当云服务器加入该安全组后，即受到这些访问规则的保护。

系统会为每个用户默认创建一个默认安全组，默认安全组的规则是在出方向上的数据报文全部放行，入方向访问受限，安全组内的云服务器无需添加规则即可互相访问。

参数模板

参数模板就像是参数配置值的容器，这些值可应用于一个或多个DDM实例。如果您想使用您自己的参数模板，只需创建一个新的数参数模板，创建实例的时候选择该参数模板。如果是在创建DDM实例后有这个需求，可以重新应用该参数模板。

弹性公网 IP

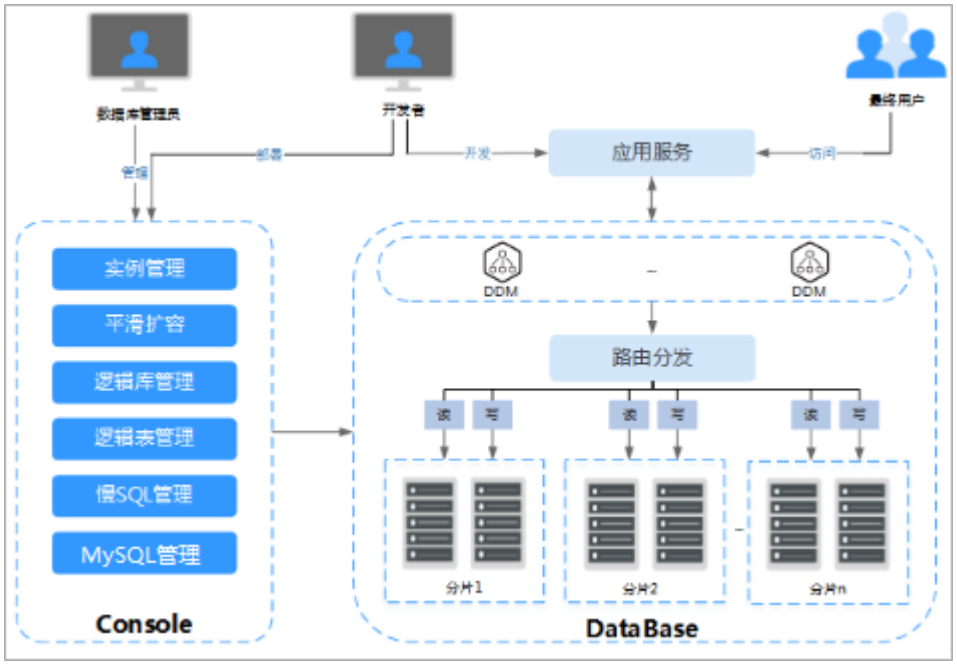
弹性公网IP（Elastic IP，简称EIP）提供独立的公网IP资源，包括公网IP地址与公网出口带宽服务。可以与DDM实例解绑和绑定。

1.3 产品架构

DDM作为数据库中间件，将底层数据库存储引擎以集群方式管理起来，用户使用非常方便。

应用程序不需要关心具体有多少分片。类似操作单机数据库，用户通过DDM管理控制台进行数据库运维，使用JDBC等驱动服务或MySQL客户端等方式连接数据库，进行数据读写。

图 1-1 DDM 业务架构



1.4 产品核心功能

分布式数据库中间件具备水平拆分、扩容、SQL语法、读写分离、全局序列等主要功能。

表 1-1 主要功能介绍

功能名称	说明
水平拆分	在创建分布式数据库时，只需选择拆分键，DDM就可以按照拆分键生成拆分规则，实现数据水平拆分。
扩容	DDM既支持计算层（DDM）扩容（增加节点数或提升节点规格），也支持存储层（MySQL）在线扩容，存储层扩容又分为平移扩容（总分片数不变）和翻倍扩容（分片数倍增）。计算层扩容对业务完全透明，存储层扩容对业务秒级影响。
分布式事务	DDM当前支持单机、FREE、XA三种事务模型。 • 单机：不允许跨分片事务。 • FREE：跨分片事务commit时部分失败无法回滚，导致数据不一致。 • XA分布式事务：两阶段提交，跨分片事务commit时部分失败会自动回滚，保证事务内数据一致性。

功能名称	说明
数据导入导出	支持外部数据导入，帮助用户实现数据库平滑上云。支持DDM实例数据按业务需求导出。
SQL语法	高度兼容MySQL协议和语法。
读写分离	DDM的读写分离功能对应用透明，无需修改任何业务代码，将只读实例添加到DDM即可。提升数据库处理能力，提高访问效率，轻松应对高并发的实时交易场景。
全局序列	DDM支持分布式全局唯一且有序递增的数字序列。满足业务在使用分布式数据库下对主键或者唯一键以及特定场景的需求。
Console运维管理界面	DDM提供Console界面，可在线对DDM实例、逻辑库等进行管理和维护。

DDM 与其他服务的关系

- 虚拟私有云（VPC）

  DDM运行于虚拟私有云，需要使用虚拟私有云创建的IP和带宽。通过虚拟私有云安全组的功能，可增强访问DDM服务的安全性。

- 弹性云服务器（ECS）

  成功申请DDM实例后，您需要通过弹性云服务器连接使用DDM实例。

- 云数据库服务（RDS）

  申请DDM实例后，需要关联同一虚拟私有云中的MySQL实例，实现分布式数据库计算与存储。

- 云审计服务（CTS）

  云审计服务（Cloud Trace Service），提供DDM服务资源的操作记录，记录内容包括您从云计算平台发起的云服务资源操作请求以及每次请求的结果，供您查询、审计和回溯使用。

- 弹性负载均衡（ELB）

  弹性负载均衡（ELB）将访问流量根据分配策略分发到后端多台服务器，通过流量分发扩展DDM对外的服务能力，同时通过消除单点故障提升DDM服务的可用性。

1.5 产品规格

- 通用增强型实例是新推出的一系列性能高、计算能力稳定的实例规格，搭载英特尔® 至强® 可扩展处理器，配套高性能网络，综合性能及稳定性全面提升，满足对业务稳定性及计算性能要求较高的企业级应用诉求。

表 1-2 产品规格

规格类型	架构	CPU（核）	内存（GB）
通用增强型	X86	8	16
		16	32

规格类型	架构	CPU（核）	内存（GB）
		32	64

1.6 使用限制

1.6.1 网络访问使用限制

在使用分布式数据库中间件（简称DDM）过程中，对于网络访问存在一些使用限制。

- 用户申请的MySQL实例、部署应用程序的ECS、DDM实例必须属于同一个VPC。
- 当用户需要在自己电脑访问DDM服务时，需要为DDM开通EIP服务，然后通过EIP访问DDM。

1.6.2 MySQL 实例使用限制

在使用分布式数据库中间件（简称DDM）过程中，对于MySQL实例存在一些使用限制。

- 目前支持5.7及8.0系列版本的MySQL实例。
- DDM暂不支持MySQL实例配置SSL连接。
- 对已经被DDM关联的MySQL实例进行修改配置等操作时，可能导致使用异常。修改后需要在DDM实例详情页面，单击“同步数据库信息”，将修改的配置进行同步，保证功能可用性。

1.6.3 SQL 语法使用限制

DDM高度兼容MySQL协议和语法，但由于分布式数据库和单机数据库在架构上存在较大的差异，对SQL使用存在一些限制。

由于分布式事务的特性，对于增、删、改数据表等批量SQL操作，需要设置手动提交事务，以确保数据记录改动的准确无误。

例如客户端在批量操作SQL语句时，可进行如下设置：

set autocommit=0;

{sql operations};

commit;

说明

当前版本仅支持在SQL语句中对当前连接库进行操作，不支持在SQL语句中对其他连接库进行操作，否则会导致异常结果。

表 1-3 DDL 的语法限制

DDL语法	限制条件
DDL语法	● ALTER命令不支持修改数据库和拆分键字段。 ● 不支持创建TEMPORARY类型的拆分表、广播表。 ● 不支持从另一张表创建新的拆分表、广播表。 ● 仅单表支持外键，拆分表和广播表不支持外键。 ● 支持从另一张单表或广播表创建新的单表，但不支持从另一张拆分表创建新的单表。 ● CREATE TABLE Syntax和DROP TABLE Syntax不支持注解，且不受参数sql_execute_timeout的限制。

表 1-4 DML 的语法限制

DML语法	限制条件
DELETE语句	● 不支持PARTITION子句。 ● 不支持子查询。
UPDATE语句	● 不支持PARTITION子句。 ● 不支持子查询。

表 1-5 数据库管理语句的限制

数据库管理语句	限制条件
数据库管理语句	● 不支持SET Syntax修改全局变量。 ● 不支持SHOW TRIGGERS语法。

表 1-6 SQL 高级功能的限制

SQL高级功能	限制条件
SQL高级功能	● 暂不支持用户自定义数据类型、自定义函数。 ● 暂不支持视图、存储过程、触发器、游标。 ● 暂不支持 BEGIN…END、LOOP…END LOOP、REPEAT…UNTIL…END REPEAT、WHILE…DO…END WHILE 等复合语句。 ● 暂不支类似 IF ， WHILE 等流程控制类语句。

1.7 区域和可用区

什么是区域、可用区?

我们用区域和可用区来描述数据中心的位置，您可以在特定的区域、可用区创建资源。

如何选择可用区?

是否将资源放在同一可用区内，主要取决于您对容灾能力和网络时延的要求。

- 如果您的应用需要较高的容灾能力，建议您将资源部署在同一区域的不同可用区内。
- 如果您的应用要求实例之间的网络延时较低，则建议您将资源创建在同一可用区内。

1.8 应用场景

分布式数据库中间件（简称DDM）是一个面向OLTP业务的分布式关系型数据库访问服务，适用于各行业数据库应用。

尤其适用于大规模的数据存储与高并发访问的行业应用，如互联网应用、物联网数据、高性价比数据库解决方案等应用场景。

- **互联网应用**

  电商、金融、O2O、零售、社交应用等行业，普遍存在用户基数大、营销活动频繁、核心交易系统数据库响应日益变慢的问题，制约业务发展。DDM提供线性水平扩展能力，能够实时提升数据库处理能力，提高访问效率，轻松应对高并发的实时交易场景。

- **物联网数据**

  在工业监控和远程控制、智慧城市的延展、智能家居、车联网等物联网场景下。传感监控设备多，采样频率高，数据规模大，会产生超过单机数据库存储能力极限的数据，造成数据库容量瓶颈。DDM提供的容量水平扩展能力，可以有效的帮助用户低成本的存储海量数据。

- **高性价比数据库解决方案**

  政务机构、大型企业、银行等行业为了支持大规模数据存储和高并发数据库访问，传统方案需要强依赖小型机和高端存储等高成本的商业解决方案。DDM利用普通服务器进行集群部署，提供与传统商业解决方案相同甚至更高的处理能力。

2 快速入门

2.1 概述

操作场景

本章以DDM实例连接RDS实例为例介绍如何使用DDM服务。

使用流程

步骤一：登录分布式数据库中间件服务

步骤二：申请数据库中间件实例

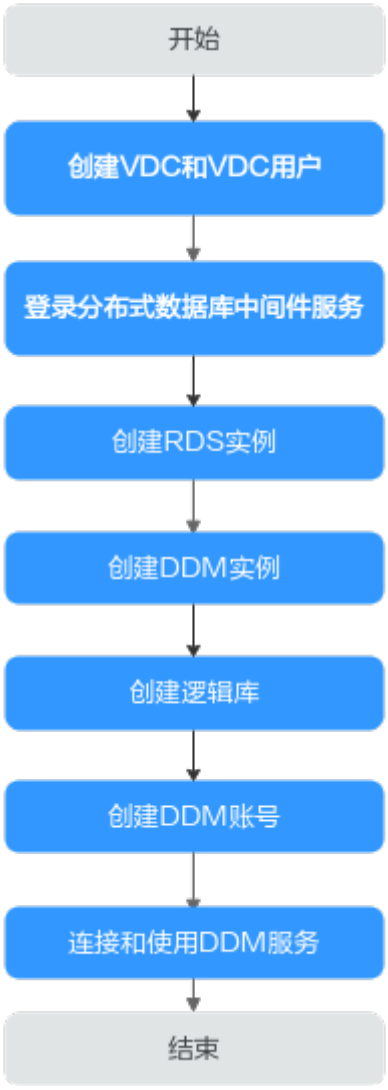
步骤三：申请RDS实例

步骤四：创建逻辑库

步骤五：创建DDM账号

步骤六：连接DDM实例或逻辑库

图 2-1 DDM 使用流程



2.2 步骤一：登录分布式数据库中间件服务

操作步骤

步骤1 在云控制台登录页，输入账号及密码，登录云控制台.

步骤2 单击管理左上角的 ，选择对应Region。

步骤3 在服务列表页面，选择“数据库 > 分布式数据库中间件 DDM ”，进入分布式数据库中间件信息页面。

----结束

2.3 步骤二：申请数据库中间件实例

操作步骤

步骤1 在RDS管理控制页面单击服务列表，选择“数据库 > 分布式数据库中间件 DDM ”，进入DDM管理控制台。

步骤2 在实例管理页面，单击页面右上方的申请数据库中间件实例。

步骤3 在申请数据库中间件实例页面，按需设置实例相关信息。

表 2-1 参数说明

参数	说明
可用区	可选择的可用区。 当前部分区域支持跨可用区部署，以增强DDM实例高可用性。 如果您的实例需要跨可用区部署，可以选择多个可用区，DDM实例的节点将部署在不同的可用区中。 **说明** 跨可用区部署会产生一定的网络时延，建议将应用程序和数据库服务（DDM、RDS）配置在相同可用区，减少网络时延。
实例名称	DDM实例的名称。 ● 名称不能为空。 ● 只能以英文字母开头。 ● 长度为4到64位的字符串。 ● 可包含英文字母、数字和中划线（-）。
性能规格	DDM实例的规格，“通用增强型”。 **说明** 为了使所申请的DDM实例能更好地满足应用需求，您需要先评估应用所需的计算能力和存储能力，结合业务类型及服务规模，选择合适的实例规格，主要包括：CPU、内存。
规格节点	实例所含有节点个数，最多支持32个节点数。
虚拟私有云	DDM实例所在的虚拟专用网络，可对不同业务进行网络隔离，方便地管理、配置内部网络，进行安全、快捷的网络变更。 单击“查看虚拟私有云”，系统跳转到虚拟私有云界面，可以查看相应的虚拟私有云，以及安全组的出方向规则和入方向规则。 **说明** 创建DDM实例选择的虚拟私有云要与RDS实例保持一致。 DDM实例必须与应用程序、RDS实例处于相同的VPC，以保证网络连通。
子网	包括子网名称与子网IP地址段。

参数	说明
安全组	已创建的安全组。 建议DDM实例与应用程序、RDS for MySQL实例选择相同的安全组，三者网络访问不受限制。如果选择了不同的安全组，请注意添加安全组访问规则，开通网络访问。
参数模板	您可以选择已有参数；同时支持单击查看参数模板，在参数模板页面，进行设置参数信息。

步骤4 实例信息设置完成后，单击页面下方“立即申请”。

步骤5 确认配置信息，单击“提交”。

----结束

申请

2.4 步骤三：申请 RDS 实例

操作步骤

步骤1 在DDM管理控制页面单击服务列表，选择云数据库RDS，进入RDS管理控制台。

步骤2 在RDS管理控制台右上角单击“申请数据库实例”，在详情页面选择相关信息，单击立即创建，进入实例预览页面。

步骤3 在申请RDS实例规格确认页面，确认信息。

步骤4 确认配置信息，单击“提交”。

步骤5 等待1-3分钟实例即可创建成功。

----结束

2.5 步骤四：创建逻辑库

创建DDM逻辑库有两个入口，以服务列表页为例。

图 2-2 实例详情-逻辑库列表页进入

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏单击“创建逻辑库”。

步骤2 在创建逻辑库基本信息页面选择拆分模式、填写逻辑库名称，选择单数据库分片数，设置好之后，单击页面下方的“下一步：选择RDS”。

图 2-3 创建逻辑库

说明

拆分模式：

- 拆分：一个逻辑库对应多个数据库实例。
- 非拆分：一个逻辑库仅对应一个数据库实例，在该数据库实例上仅创建1个分片。

逻辑库名称：长度为2-24个字符，必须以字母开头，可以包含字母、数字、下划线，不能包含其它特殊字符。

单数据库分片数：支持1-64分片数自选。

步骤3 在选择RDS实例列表页面，勾选RDS for MySQL实例，您可将该RDS for MySQL实例设置为“单表的储存节点”。单击页面下方“预览”。

步骤4 在预览页面，输入RDS for MySQL实例管理员密码，检查分片预览信息是否无误，若有误请重新选择，确认无误后单击“测试连接”，测试通过后，单击页面下方的“完成”。

图 2-4 测试连接

----结束

2.6 步骤五：创建 DDM 账号

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤3 在账号管理页面单击“创建DDM账号”，在弹窗中填选账号参数信息。

表 2-2 创建 DDM 账号配置参数

参数	说明
账号名称	DDM账号的名称，命名规则如下。 ● 必须以字母开头 ● 不区分大小写 ● 长度为1~32个字符 ● 可以包含字母，数字、下划线，不能包含其它特殊字符。
密码	DDM账号的密码，密码复杂度要求如下。 ● 长度为8~32个字符 ● 至少包含三种字符组合：大小写字母、数字、特殊字符`~!@#$%^&*()-_=+\\|[{}];:'\"<.>/?和空格 ● 不能简单、强度不够、容易被猜测的弱口令
确认密码	-
关联逻辑库	DDM账号与逻辑库关联绑定，下拉列表中显示可关联的逻辑库。 DDM账号只对已关联的逻辑库有访问权限。
账号权限	权限分为基础权限和扩展权限。 ● 基础权限：CREATE、DROP、ALTER、INDEX、INSERT、DELETE、UPDATE、SELECT ● 扩展权限：全表查询、全表删除、全表更新。 ● 权限配置应该遵循如下原则： 基础权限，用户可根据需要自定义选择。 扩展权限对应的基础权限必须选择，对应关系如下： – “全表查询”对应“SELECT” – “全表删除”对应“DELETE” – “全表更新”对应“UPDATE”
描述	对DDM账号的详细描述信息，长度不能超过256个字符。

步骤4 信息填写完成，单击“确定”即可创建成功。

----结束

2.7 步骤六：连接 DDM 实例或逻辑库

用户申请DDM实例后，可以使用Navicat等客户端连接DDM实例，也可以使用CLI或JDBC驱动连接DDM实例中的目标逻辑库。

本章节指导用户如何连接DDM实例或逻辑库。

准备工作

连接DDM实例或逻辑库前，需要获取DDM实例或逻辑库的连接地址。

获取 DDM 实例连接地址

步骤1 登录DDM管理控制台。

步骤2 在左侧选择“数据库> 分布式数据库中间件”，进入实例管理页面。

步骤3 单击左侧菜单栏的“实例管理”。在实例管理界面单击DDM实例名称。

步骤4 在实例信息页的“网络信息”模块，找到“连接地址”。

说明

连接DDM时，如果5次输入错误密码，系统将会自动锁定20分钟。

----结束

获取 DDM 逻辑库连接地址

步骤1 登录DDM管理控制台。

步骤2 在左侧选择“数据库> 分布式数据库中间件”，进入实例管理页面。

步骤3 单击左侧菜单栏的“实例管理”，在实例管理界面单击DDM实例名称，进入实例基本信息页。

步骤4 在左侧导航栏选择“逻辑库列表”。

步骤5 单击逻辑库名称进入逻辑库基本信息页面。

步骤6 在“连接地址栏”获取“命令行连接地址”和“jdbc连接地址”。

说明

连接DDM时，如果5次输入错误密码，系统将会自动锁定20分钟。

----结束

连接方法概述

方法一：**Navicat客户端连接DDM实例**。

方法二：**MySQL命令行连接DDM逻辑库**。

方法三：**JDBC驱动连接DDM逻辑库**。

方法四：**Console界面连接DDM实例**。

说明

1. 为保证系统安全，请使用与DDM实例处于同一VPC的弹性云服务器。
2. 弹性云服务器已安装MySQL客户端或已配置MySQL连接驱动。
3. 连接DDM实例前，需要先在工具中配置DDM实例信息。

Navicat 客户端连接 DDM 实例

步骤1 登录分布式数据库中间件服务，单击需要连接的DDM实例名称，进入实例基本信息页面。

步骤2 在“实例信息”模块的弹性公网IP单击“绑定”。绑定已申请的公网IP。

步骤3 在DDM管理控制台左侧选择虚拟私有云图标。单击“访问控制>安全组”

步骤4 在安全组界面，单击操作列的“配置规则”，进入安全组详情界面。在安全组详情界面，单击“添加规则”，弹出添加规则窗口。根据界面提示配置安全组规则，设置完成后单击“确定”即可。

说明

绑定弹性公网IP后，建议您在内网安全组中设置严格的出入规则，以加强数据库安全性。

步骤5 打开Navicat客户端，单击“连接”。在新建连接窗口中填写主机IP地址（弹性公网IP地址）、用户名和密码（DDM账号、密码）。

说明

Navicat客户端推荐使用版本为Navicat12。

步骤6 单击“连接测试”，若显示连接成功，单击“确定”，等待1-2分钟即可连接成功。连接失败会直接弹出失败原因，请修改后重试。

----结束

说明

通过其他可视化的MySQL工具（例如 Workbench）连接DDM实例的操作与此章基本一致，不做详细描述。

MySQL 命令行连接 DDM 逻辑库

步骤1 登录弹性云服务器，打开命令行工具，输入以下命令。

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

表 2-3 参数说明

参数示例	参数填写说明	参数举例
DDM_SERVER_ADDRESS	DDM实例所在IP地址。	192.168.0.200
DDM_SERVER_PORT	DDM实例连接端口。	5066

参数示例	参数填写说明	参数举例
DDM_USER	DDM实例账号。	dbuser01
DDM_DBNAME	DDM实例逻辑库名，选填。	-
default-character-set=utf8	指定字符编码为UTF-8，选填。 当MySQL连接编码和实际编码不一致，导致DDM解析出现乱码时请配置该参数。	-
default_auth=mysql_native_password	默认使用密码认证插件。	-

说明

假如您使用了 MySQL 8.0的客户端，需要增加 default_auth=mysql_native_password参数。

步骤2 下面为Windows服务器命令行窗口中使用表中举例参数MySQL命令连接服务器的回显情况。

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

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

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

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

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

mysql>
```

----结束

JDBC 驱动连接 DDM 逻辑库

步骤1 加载驱动程序。

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

说明

建议JDBC驱动版本为5.1.35~5.1.45。

步骤2 打开数据库连接。

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

步骤3 创建Statement对象。

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


步骤4 执行SQL语句。

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

步骤5 （可选）优化代码。

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

说明

- **loadBalanceAutoCommitStatementThreshold**和**retriesAllDown**参数必须按照以上样例进行配置，否则在连接切换时可能进入死循环，导致栈溢出。
- loadBalanceAutoCommitStatementThreshold：表示执行多少个语句后重新选择连接。
- loadBalanceHostRemovalGracePeriod：设置宽限期，以等待主机从负载均衡连接中移除，当主机当前是活动主机时释放主机。
- loadBalanceBlacklistTimeout：通过控制服务器在全局黑名单中的生存时间，检查不可用服务器之间的时间间隔（以毫秒为单位）。
- loadBalancePingTimeout：使用负载平衡连接时，等待每个负载平衡物理连接ping响应的时间（以毫秒为单位）。
- retriesAllDown：当所有的连接地址都无法连接时，轮询重试的最大次数。重试次数达到阈值仍然无法获取有效连接，将会抛出SQLException
- connectTimeout：套接字连接的超时（毫秒），其中0表示没有超时。只能在JDK-1.4或更高版本上工作。默认为“0”。

----结束

Console 界面连接 DDM 实例

步骤1 在浏览器上输入DDM的Console地址、账号和密码，登录分布式数据库中间件服务管理控制台。

步骤2 在侧边栏选择“实例管理”。

步骤3 在实例列表操作栏，单击“登录”。

系统自动跳转至数据管理服务管理控制台的实例登录页面。

步骤4 在实例登录窗口中输入DDM账号和密码。

步骤5 单击“测试连接”。

步骤6 （可选）选择“定时采集”和“SQL执行记录”。

步骤7 确认无误后，单击“登录”。

----结束

3 功能总览

分布式数据库中间件（Distributed Database Middleware，简称DDM），是一款分布式关系型数据库。它兼容MySQL协议，采用存储计算分离架构的模式，使得存储层、计算层可以无限扩展，从而拥有了海量数据高并发访问能力。

分布式数据库中间件支持的功能如**表3-1**所示。

表 3-1 分布式数据库中间件服务功能列表

功能分类	功能描述
实例管理	包括实例创建、实例规格变更、实例删除、实例重启、实例续费与退订等功能。具体使用方法请参考**实例管理**。
备份管理	包括数据的备份与恢复、数据的一致性备份与恢复等功能。具体使用方法请参考**备份管理**。
参数模板管理	包括参数模板的创建、编辑、比较、复制、应用等功能。具体使用方法请参考**参数模板管理**。
任务中心	该模块可以查看用户在控制台上提交的异步任务的执行进度和状态。具体使用方法请参考**任务中心**。
逻辑库管理	包括逻辑库的创建、导出、导入、删除等功能。具体使用方法请参考**逻辑库管理**。
逻辑库扩容管理	包括平移扩容和翻倍扩容两种，具体使用方法请参考**逻辑库扩容**。
账号管理	包括DDM账号的创建、修改、删除和重置密码等功能。具体使用方法请参考**账号管理**。
监控管理	包括监控指标一览表和如何查看监控指标，具体使用方法请参考**监控管理**。
SQL语法管理	包括DDL语法、DML语法、全局序列、实用SQL语句和多种拆分算法的使用等功能，具体使用方法请参考**SQL语法**。

4 实例管理

4.1 实例申请

前提条件

成功登录分布式数据库中间件服务控制台。

操作步骤

步骤1 在实例管理页面，单击页面右上方的申请数据库中间件实例。

步骤2 在申请数据库中间件实例页面，按需设置实例相关信息。

表 4-1 参数说明

参数	说明
可用区	可选择的可用区。 目前多数有多个可用区的区域支持跨可用区部署，以增强DDM实例高可用性。 如果您的实例需要跨可用区部署，可以选择多个可用区，DDM实例的节点将部署在不同的可用区中。 **说明** 跨可用区部署会产生一定的网络时延，建议将应用程序和数据库服务（DDM、RDS）配置在相同可用区，减少网络时延。
实例名称	DDM实例的名称。 ● 名称不能为空。 ● 只能以英文字母开头。 ● 长度为4到64位的字符串。 ● 可包含英文字母、数字和中划线（-）。

参数	说明
性能规格	DDM实例的规格，包括X86架构的“通用增强型”、ARM架构的“鲲鹏通用计算增强型”。 **说明** 为了使所申请的DDM实例能更好地满足应用需求，您需要先评估应用所需的计算能力和存储能力，结合业务类型及服务规模，选择合适的实例规格，主要包括：CPU、内存。
规格节点	实例所含有节点个数，单次最多可批量申请32个节点数。
虚拟私有云	DDM实例所在的虚拟专用网络，可对不同业务进行网络隔离，方便地管理、配置内部网络，进行安全、快捷的网络变更。 单击“查看虚拟私有云”，系统跳转到虚拟私有云界面，可以查看相应的虚拟私有云，以及安全组的出方向规则和入方向规则。 **说明** 创建DDM实例选择的虚拟私有云要与MySQL实例保持一致，以保证网络连通。
安全组	已创建的安全组。 建议DDM实例与应用程序、MySQL实例选择相同的安全组，三者网络访问不受限制。如果选择了不同的安全组，请注意添加安全组访问规则，开通网络访问。
参数模板	参数模板就像是参数配置值的容器，这些值可应用于一个或多个DDM实例。您可以使用参数模板中的参数来管理DDM实例的配置。

步骤3 实例信息设置完成后，单击页面下方“立即申请”。

步骤4 确认配置信息，单击“提交”。

----结束

4.2 组

4.2.1 组介绍

概述

DDM组是对DDM计算资源进行隔离的一种扩展能力。将DDM集群节点分成只读节点组和读写节点组，各自承担读流量和读写流量，缓解DDM集群读写负载压力。

只读组与读写组使用同一份数据，在高并发、大流量的场景下，只读组可直接在RDS主实例或者只读实例上进行复杂查询或离线抽取数据等需求，减少查询响应时间，抵御高并发访问压力。

操作便捷，无需构建复杂链路，也无需进行数据同步等其他操作。

说明

目前组仅支持白名单开放，如需使用，请工单联系服务人员。

4.2.2 管理组

操作步骤

步骤1 登录分布式数据库中间件DDM管理控制台，选择“实例管理”页签，在实例管理表中单击“实例名称”，默认进入“基本信息页”。

步骤2 在“基本信息页”，可以看到一个默认的“group-default”读写组，您之前申请的节点默认被分在“group-default”组。

须知

组特性需满足实例内核版本大于等于2.4.1.2。

步骤3 如果您需要创建新的读写组或者只读组，单击“创建组”。

说明

- 一个DDM实例下支持创建多个读写/只读组，组的总节点数不超过32个，建议每组至少2个节点。
- 节点不支持跨不同的组，也不支持更换组，同一个组内所有节点的规格保持一致。

步骤4 在创建组页面，根据需求选择读写组/只读组角色，选择合适的性能规格，同时为组添加节点，单击下一步。

说明

只读组只处理读流量。

步骤5 确认组创建信息，单击“提交”。

步骤6 创建完成，可在节点信息页面查看相关组信息。每个组创建后，会配置一个对应的连接地址

说明

包周期类型的DDM实例，暂不支持删除组。

如需删除组，点击“删除”即可，删除组对应连接失效，可能会影响您的业务，请谨慎操作。读写组至少保留一个。

----结束

4.3 规格变更

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 实例状态为“运行中”。

须知

实例规格变更期间服务会短暂中断，建议在业务空闲时变更。

操作步骤

须知

如果未使用只读组功能，规格变更的入口请参考**步骤1**。

如果使用了只读组功能，规格变更的入口请参考**步骤2**。

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“规格变更”。

步骤2 在分布式数据库中间件DDM管理控制台，选择“实例管理 > 实例名称 > 基本信息页 > 节点信息，性能规格中单击“规格变更”。

步骤3 在分布式数据库中间件变更规格页面，您可查看当前实例规格，并在页面下部选择需要变更成的实例规格。

步骤4 确认无误后，单击“下一步”即可修改成功。

步骤5 确认配置信息，单击“提交”。

步骤6 返回实例管理列表页面，查看当前实例状态为“规格变更中”。

说明

- 一旦执行变更操作后不可撤销。如果需要修改，需要在当前变更操作结束后重新提交变更操作。
- 规格变更支持升高规格和降低规格两种。

----结束

4.4 计算节点扩容

操作场景

随着业务数据的增加，为了提高实例业务稳定性，您可对DDM实例节点进行扩容。

说明

- 请在业务空闲时间段进行节点扩容操作。
- 请确保实例关联的MySQL实例状态正常并且没有进行其他操作。

操作步骤

须知

如果未使用只读组功能，节点扩容的入口请参考**步骤1**。

如果使用了只读组功能，节点扩容的入口请参考**步骤2**。

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“节点扩容”。

步骤2 在分布式数据库中间件DDM管理控制台，选择“实例管理 > 实例名称 > 基本信息页 > 节点信息”，规格节点中单击“节点扩容”。

步骤3 在分布式数据库中间件节点扩容页面，您可查看当前规格，在选择节点模块选择可用区，并添加节点。

说明

一个DDM实例最多支持32个节点。

步骤4 设置完节点数，单击页面下方的“下一步”。

步骤5 在规格确认页面，若您需要重新修改节点数，请单击“上一步”，再次确认所选规格无误后，单击页面下方的“提交”，提交节点扩容任务。

说明

节点扩任务提交后，实例此时状态为“节点扩容中”。

----结束

4.5 计算节点缩容

操作场景

随着业务数据的较少，为了降低成本，您可对实例节点进行缩容。

说明

- 请在业务空闲时间段进行节点缩容操作。
- 请确保实例关联的RDS实例或GaussDB(for MySQL)实例状态正常并且没有进行其他操作。

操作步骤

须知

如果未使用只读组功能，节点缩容的入口请参考**步骤1**。

如果使用了只读组功能，节点缩容的入口请参考**步骤2**。

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“节点缩容”。

步骤2 在分布式数据库中间件DDM管理控制台，选择“实例管理 > 实例名称 > 基本信息页 > 节点信息，规格节点中单击“节点缩容”。

步骤3 在分布式数据库中间件节点缩容页面，您可查看当前规格，并按需设置节点缩容数量。

说明

一个DDM实例最少支持1个节点。

步骤4 设置完节点数，单击页面下方的“下一步”。

步骤5 在规格确认页面，确认所选规格无误后，单击页面下方的“提交”，提交节点缩容任务。

说明

节点缩容任务提交后，实例此时状态为“节点缩容中”。

----结束

4.6 访问控制

操作场景

DDM开启了负载均衡之后，默认不限制访问的IP地址，需通过“访问控制”功能做访问的安全控制。直连DDM节点的访问，安全组依然有效。

操作步骤

步骤1 在分布式数据库中间件DDM管理控制台，选择“实例管理 > 实例名称 > 基本信息页 > 节点信息，组信息中单击打开“访问控制”的开关，打开后，出现设置按钮。

步骤2 单击“设置”，弹出“修改访问控制”弹框。

步骤3 在修改访问控制弹框中选择“访问控制方式”，输对应名单的IP地址，单击“确定”。

----结束

4.7 实例重启

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 当前实例状态为“运行中”。

须知

实例重启期间服务不可用且操作无法撤销，请谨慎操作。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“重启实例”。

步骤2 在弹出确认窗口中，单击“是”。

步骤3 在实例管理列表页面，等待实例重启成功。

----结束

4.8 实例删除

前提条件

成功登录分布式数据库中间件服务控制台。

须知

删除操作无法恢复，请谨慎操作。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“删除实例”。

步骤2 在弹出确认窗口中，单击“是”。

说明

- 如需删除挂载于DDM上的MySQL实例数据，请勾选“删除存储层实例数据”。

----结束

4.9 表数据重载

前提条件

成功登录分布式数据库中间件服务控制台。

已经通过DRS服务将数据迁移到新的DDM实例。

应用场景

在DDM跨区域容灾场景下，通过数据迁移服务（DRS）迁移业务数据，DDM无法感知逻辑表信息所在位置，所以迁移完成之后需要用户主动下发“表数据重载”重新加载信息。

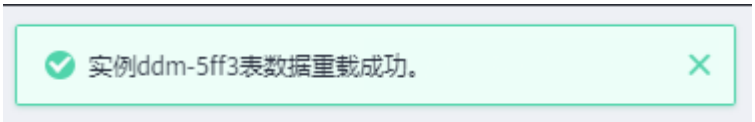
操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，选择目标实例。

步骤2 在操作栏，选择“更多”>“表数据重载”。

系统自动弹出实例XXX表数据重载成功。

图 4-1 表数据重载成功提示



----结束

4.10 参数管理

操作场景

为了确保DDM服务发挥出最优性能，您可以根据自己的业务情况对DDM实例的运行参数进行配置。

前提条件

已有DDM实例，且实例状态处于“运行中”。

操作步骤

步骤1 输入账户名和密码登录分布式数据库中间件服务控制台。

步骤2 单击左侧菜单栏的“实例管理”，进入实例管理页面。

步骤3 在实例管理页面单击实例名称，进入实例信息详情页。

步骤4 在该页面单击“参数管理”。您可以根据需要修改对应参数。

图 4-2 参数管理页面

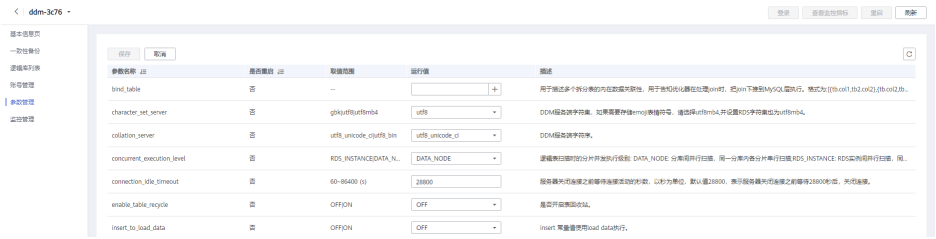


表 4-2 DDM 实例配置参数说明

参数名称	缺省值	取值范围	参数说明
bind_table	-	格式要求：[{tb.col1,tb2.col2},{tb.col2,tb3.col1},...]该参数要以“表名.列名”的形式出现且可以多对同时出现。 版本要求： DDM 2.3.2.7版本及以上。	用于描述多个拆分表的内在数据关联性，用于告知优化器在处理join时，把join下推到MySQL层执行。参数举例请详见表下的说明。
max_connections	20000	10~40000	允许同时连接的客户端总数。 此参数需要结合RDS for MySQL实例的规格及处理能力配置合适的值。连接数过多，可能造成连接等待，影响性能。DDM的连接数消耗与分片数量和SQL设计等因素相关。 例如：SQL带拆分键时，1个DDM连接同时消耗后面1个RDS for MySQL实例连接；SQL不带拆分键时，假设分片个数为N，那么会消耗N个RDS连接。 因此，SQL合理设计且DDM和RDS的处理能力不成为瓶颈的前提，DDM最大连接数可以配置成略小于“后端RDS的数量 * 单个RDS for MySQL实例支持的最大连接数”。 建议根据自己的业务进行实际压测，配置合理的值。
connection_idle_timeout	28800	60~86400 (s)	服务器关闭连接之前等待连接活动的秒数。
long_query_time	1	0.1~3600 (s)	记录慢查询的最小秒数。
sql_execute_timeout	28800	100~28800(s)	SQL执行超时秒数。
connection_failed_threshold	50	1~10000	客户端连接失败达到多少次后账号和IP地址被锁定。
connection_failed_delay	1200	1~86400 (s)	账号和IP地址被锁定后延迟多少秒解锁。

参数名称	缺省值	取值范围	参数说明
max_allowed_packet	16777216	1024~1073741824	包或任何生成的中间字符串的最大值。包缓冲区初始化为net_buffer_length字节，但需要时可以增长到max_allowed_packet字节。该值默认很小，以捕获大的（可能是错误的）数据包，该值必须设置为1024的倍数。
character_set_server	utf8	gbk、utf8、utf8mb4	DDM服务端字符集，如果需要存储emoji表情符号，请选择utf8mb4并设置RDS for MySQL实例字符集也为utf8mb4。 **说明** DDM字符集与RDS for MySQL字符集（包括character_set_client、character_set_connection、character_set_database、character_set_results、character_set_server）需要保持一致。
collation_server	将根据您设置的character_set_server项进行匹配。	将根据您设置的character_set_server项进行匹配。	DDM服务端字符序。 根据您所设置的character_set_server联动匹配collation_server选项： ● gbk -->gbk_chinese_ci、gbk_bin； ● utf8 -->utf8_unicode_ci、utf8_bin； ● utf8mb4 --> utf8mb4_unicode_ci、utf8mb4_bin。
sql_audit	OFF	OFF、ON	开启或关闭SQL审计。
ddm_instance_type	-	SINGLE,MASTER,SLAV	SINGLE:设置DDM实例为单实例。 SLAVE:设置DDM实例在灾备中作为备实例。 MASTER:设置DDM实例在灾备中作为主实例。
transaction_policy	XA	XA、FREE、NO_DTX	XA：XA 事务，保证原子性，保证可见性； FREE：允许多写，不保证原子性，无性能损耗； NO_DTX：单分片事务。

DDM默认支持修改以上实例参数，特殊场景（如数据迁移）下如需修改更多实例参数请联系技术支持人员协助处理。

参数举例：

图 4-3 未使用 bind_table 结果展示

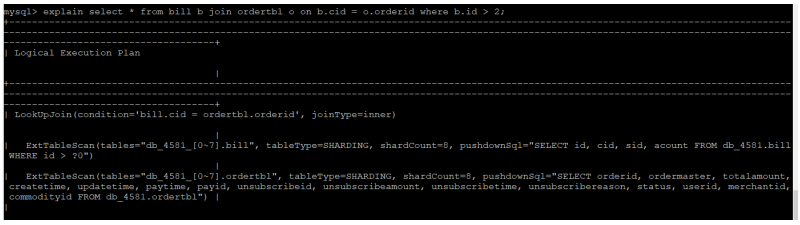
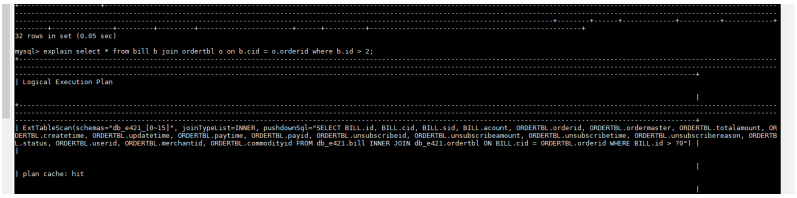


图 4-4 使用 bind_table 结果展示



步骤5 确认无误后，单击“保存”，并在弹框中单击“是”完成参数修改。

图 4-5 参数确认页面

说明

- 修改配置参数可能影响应用访问DDM实例，请谨慎操作。
- 修改参数命令下发成功后，预计需要20~60秒生效，请耐心等待。

----结束

4.11 读写分离

前提条件

- 已申请DDM实例和带只读实例的MySQL/GaussDB(for MySQL)实例。
- 已经创建好逻辑库。

操作步骤

步骤1 登录DDM控制台，选择目标DDM实例，进入实例基本信息页面。

步骤2 单击右下角“关联数据库信息”模块。

步骤3 单击“修改读策略”，修改主实例和只读实例的读写权重。

说明

- DDM读写分离功能可以将只读查询的流量按比例分摊至下挂存储节点的主实例和只读实例，从而减轻主实例的工作负担，保障读写事务的性能。一般来说该比例的设置需结合业务实际特点以及存储节点实际负载进行设置。
  - 如果只读查询对数据实时性要求不高（容忍亚秒级可见性延迟）且只读查询的开销较大并对业务核心读写事务有一定影响，可以考虑设置为（0:100），即所有只读查询均由只读实例承担，最大程度保证主实例性能。
  - 对于其他场景，建议结合实际情况酌情调整。
- 若select语句带有hint或者在事务中做了数据修改的select语句，读请求都会下发主实例执行。

----结束

4.12 设置参数模板

前提条件

成功登录分布式数据库中间件服务控制台。

选择参数模板

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“设置参数模板”。

系统自动弹出设置参数模板窗口。

步骤2 选择需要设置的参数模板后。单击“确定”。

图 4-6 设置参数模板

----结束

5 参数模板管理

5.1 创建参数模板

您可以使用参数模板中的参数来管理DDM实例中的参数配置。参数模板就像是DDM参数配置的容器，这些值可应用于一个或多个DDM实例。

如果您在创建DDM实例时未指定客户创建的参数模板，系统将会为您的实例适配默认的参数模板。该默认参数模板包含多个系统默认值，具体根据计算等级、实例的分配存储空间而定。您无法修改默认参数模板的参数设置，您必须创建自己的参数模板才能更改参数设置的默认值。

如果您想使用您自己的参数模板，只需创建一个新的参数模板，创建实例的时候选择该参数模板，如果是在创建实例后有这个需求，可以重新应用该参数模板，请参见**应用参数模板**。

若您已成功创建参数模板，并且想在新的参数模板中包含该组中的大部分自定义参数和值时，复制参数模板是一个方便的解决方案，请参见**复制参数模板**。

以下是您在使用参数模板中的参数时应了解的几个要点：

- 当您修改当前实例的参数模板并保存后，仅应用于当前实例，不会对其他实例造成影响。
- 当您更改参数并保存参数模板时，参数更改将在您应用到实例后，手动重启DDM实例后生效。
- 在参数模板内设置参数不恰当可能会产生意外的不利影响，包括性能降低和系统不稳定。修改参数时应始终保持谨慎，且修改参数模板前要备份数据。将参数模板更改应用于使用DDM实例前，您应当在测试实例上试用这些参数模板设置更改。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在页面左上角单击 ，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面，单击“创建参数模板”。

步骤5 输入参数模板名称并添加对该参数模板的描述，单击“确定”，创建参数模板。

- 参数模板名称长度在1~64个字符之间，区分大小写，可包含字母、数字、中划线、下划线或句点，不能包含其他特殊字符。
- 参数模板的描述长度不能超过256个字符，且不能包含回车和>!<"&'=特殊字符。

说明

每个用户最多可以创建100个DDM参数模板。

----结束

5.2 编辑参数模板

为确保分布式数据库中间件服务发挥出最优性能，用户可根据业务需求对用户创建的参数模板中的参数进行调整。

您可以修改用户创建的参数模板中的参数值，但不能更改默认参数模板中的参数值。

以下是您在使用参数模板中的参数时应了解的几个要点：

- 当您更改参数并保存参数模板时，参数更改将在您应用到实例后，手动重启与参数模板关联的实例后生效。应用参数模板到DDM实例，请参见**应用参数模板**。
- 如果您更改一个参数值，则所做更改的应用时间将由该参数的类型决定。

说明

系统提供的默认参数模板不允许修改，只可单击参数模板名进行查看。当用户参数设置不合理导致实例无法启动时，可参考默认参数模板重新配置。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在页面左上角单击 ，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面的“自定义”页签，选择目标参数模板，单击参数模板名称。

步骤5 默认在“参数”页签下，根据需要修改相关参数值。

可进行的操作如下：

- 单击“保存”，在弹出框中单击“是”，保存修改。
- 单击“取消”，放弃本次设置。

步骤6 参数修改完成后，您可以单击“模板历史记录”查看参数的修改详情。

须知

参数模板修改后，不会立即应用到当前使用的实例，您需要进行应用操作才可生效，具体操作请参见**应用参数模板**。

修改某些参数或字符集后需要手动重启，由于变更规格导致的强制重启，不会触发该参数生效。

----结束

5.3 比较参数模板

操作场景

您可以在同一个DDM实例上选择不同的参数模板，以了解参数对实例的影响。

您也可以在不同的DDM实例上选择同样的参数模板，了解参数对不同实例的影响。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面的“自定义”页签，选择一个用户创建的参数模板，单击“比较”。

步骤5 选择不同参数模板，单击“确定”，比较两个参数模板之间的配置参数差异项。

说明

您也可以在“参数模板”页面的“系统默认”页签，选择一个用户创建的参数模板进行比较。步骤与自定义页签下的一致。

- 有差异项，则会显示差异参数模板的如下信息：参数名称、两个参数模板的参数值。
- 无差异项，则不显示。

----结束

5.4 查看参数修改历史

操作场景

您可以查看当前实例所使用参数模板以及自定义参数模板的修改历史，以满足业务需要。

说明

用户创建或导出的新参数模板，在未进行参数修改前，无修改历史。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面的“自定义”页签，单击目标参数模板列表操作栏的“更多 > 历史修改记录”。

您可查看参数对应的参数名称、修改前参数值、修改后参数值、修改状态和修改时间。

----结束

5.5 复制参数模板

操作场景

您可以复制您创建的自定义数据库参数模板。当您已创建一个数据库参数模板，并且想在新的数据库参数模板中包含该组中的大部分自定义参数和值时，复制参数模板是一个方便的解决方案。

复制数据库参数模板之后，新参数模板可能不会立即显示，建议您等待5分钟再使用。

您无法复制默认参数模板。不过，您可以创建基于默认参数模板的新参数模板。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面的“自定义”页签，选择需要复制的参数模板，单击“复制”。

步骤5 在弹出框中，填写新参数模板名称和描述，单击“确定”。

- 参数模板名称长度在1~64个字符之间，区分大小写，可包含字母、数字、中划线、下划线或句点，不能包含其他特殊字符。
- 参数模板的描述长度不能超过256个字符，且不能包含回车和>!<"&'=特殊字符。

创建完成后，会生成一个新的参数模板，您可在参数模板列表中对其进行管理。

----结束

5.6 应用参数模板

操作场景

参数模板编辑修改后，您可以根据业务需要应用到DDM实例中。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面，根据参数模板类型不同进行如下操作。

- 若需要将默认参数模板应用到实例，在“系统默认”页签的目标参数模板单击“应用”。
- 若需要将用户自己创建的参数模板应用到实例，在“自定义”页签的目标参数模板单击“更多 > 应用”。

一个参数模板可被应用到一个或多个实例。

步骤5 在弹出框中，选择或输入所需应用的实例，单击“确定”。

参数模板应用成功后，您可**查看参数模板应用记录**。

----结束

5.7 查看参数模板应用记录

操作场景

参数模板编辑修改后，您可根据业务需要将其应用到DDM实例中。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 单击“参数模板”。

步骤5 在“系统默认”页签下，选择目标参数模板，单击“应用记录”；或在“自定义”页签下，选择目标参数模板，单击“更多 > 应用记录”，查看应用记录。

您可查看参数模板所应用到的实例名称/ID、应用状态、应用时间、失败原因。

----结束

5.8 修改参数模板描述

操作场景

参数模板创建成功后，用户可根据需要对自己创建的参数模板描述进行修改。

说明

默认参数模板的描述不可修改。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面的“自定义”页签，选择一个用户创建的参数模板，单击“描述”列。

步骤5 输入新的描述信息，单击，提交修改，单击，取消修改。

- 参数模板的描述长度不能超过256个字符，且不能包含>!<"&'=特殊字符。
- 修改成功后，可在参数模板列表的“描述”列查看改后的描述信息。

----结束

5.9 删除参数模板

操作场景

您可删除废弃的参数模板。

须知

- 参数模板删除后，不可恢复，请谨慎操作。
- 默认参数模板不可被删除。

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“参数模板”页面的“自定义”页签，选择需要删除的参数模板，单击“更多 > 删除”。

步骤5 单击“是”，删除参数模板。

----结束

6 任务中心

您可以通过“任务中心”查看用户在控制台上提交的异步任务的执行进度和状态。

说明

分布式数据库中间件服务支持查看以下任务：

- 创建实例
- 删除实例
- 规格变更
- 节点扩容
- 节点缩容
- 重启实例
- 绑定EIP
- 解绑EIP
- 恢复数据
- 导入逻辑库信息
- 扩容
- 一致性备份
- 删除一致性备份
- 一致性恢复

操作步骤

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 在页面左上角单击，选择“数据库 > 分布式数据库中间件 ”。进入分布式数据库中间件信息页面。

步骤4 在“任务中心”页面，选择目标任务，查看任务信息。

- 通过任务名称/订单ID、实例名称/ID确定目标任务，或通过右上角的搜索框输入任务名称来确定目标任务。
- 单击页面右上角的，查看某一时间段内的任务执行进度和状态，默认时长为一周。

任务保留时长最多为一个月。

- 系统支持查看以下状态的即时任务：
  - 执行中
  - 完成
  - 失败
- 查看任务创建时间和结束时间。

----结束

7 逻辑库管理

7.1 创建逻辑库

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 当前DDM实例状态为“运行中”。
- DDM服务支持RDS和GaussDB(for MySQL)两种实例类型，以RDS for MySQL实例为例。
- DDM在RDS上创建的内部账号（DDMRW*、DDMR*、DDMREP*）不要修改和删除，否则会影响业务。

  说明

  - 内部账号名称组成规则：固定前缀（DDMRW、DDMR、DDMREP）+RDS实例id取hash值。
  - 口令规则：口令随机生成，长度最小16，最长32。

创建DDM逻辑库有两个入口，以服务列表页为例。

图 7-1 服务列表页进入

图 7-2 实例详情-逻辑库列表页进入

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，在目标实例操作栏单击“创建逻辑库”。

步骤2 在创建逻辑库基本信息页面选择拆分模式、填写逻辑库名称，选择单数据库分片数，设置好之后，单击页面下方的“下一步：选择RDS”。

图 7-3 创建逻辑库

说明

拆分模式：

- 拆分：一个逻辑库对应多个数据库实例。
- 非拆分：一个逻辑库仅对应一个数据库实例，在该数据库实例上仅创建1个分片。

逻辑库名称：长度为2-48个字符，必须以字母开头，可以包含字母、数字、下划线，不能包含其它特殊字符。

单数据库分片数：支持1-64分片数自选。

步骤3 在选择RDS实例列表页面，勾选RDS for MySQL实例，您可将该RDS for MySQL实例设置为“单表的储存节点”。单击页面下方“预览”。

步骤4 在预览页面，输入RDS for MySQL实例管理员密码，检查分片预览信息是否无误，若有误请重新选择，确认无误后单击“测试连接”，测试通过后，单击页面下方的“完成”。

图 7-4 测试连接

----结束

7.2 导出逻辑库

操作场景

跨region容灾，导出主实例的逻辑库信息。

前提条件

DDM实例中已创建逻辑库。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，选择目标DDM实例，单击实例名称，进入实例基本信息页面。

步骤2 在实例基本信息页面左侧导航栏，选择“逻辑库列表”选项卡，查看DDM实例逻辑库。

图 7-5 逻辑库列表

步骤3 在逻辑库列表页面，单击“导出逻辑库信息”，系统会自动导出当前实例下的逻辑库的json文件。

----结束

7.3 导入逻辑库

操作场景

跨region容灾，创建备实例的逻辑库。

前提条件

DDM实例没有逻辑库。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，选择目标DDM实例，单击实例名称，进入实例基本信息页面。

步骤2 在实例基本信息页面左侧导航栏，选择“逻辑库列表”选项卡，查看DDM实例逻辑库。

图 7-6 逻辑库列表

步骤3 在逻辑库列表页面，单击“导入逻辑库信息”进入导入逻辑库信息页面。

说明

json可重复导入，但可重复导入的前提是： 目标DDM实例没有同名逻辑库。

步骤4 在“导入逻辑库信息”页面单击“添加文件”，在本地选择需要导入的.json文件（**导出逻辑库**导出的json文件），选择需要使用RDS实例或GaussDB(for MySQL)实例，输入正确的数据库密码，单击完成即可。

说明

选择的RDS的数量与导入DDM关联的RDS实例或GaussDB(for MySQL)实例数量一致。

----结束

7.4 删除逻辑库

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 已创建逻辑库。

须知

删除操作无法恢复，请谨慎操作。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“逻辑库列表”，进入逻辑库列表页面。

步骤3 在逻辑库列表页面，选择目标逻辑库，在其操作栏单击“删除”。

步骤4 在删除确认弹窗中，单击“确定”。

说明

- 不能直接在RDS实例列表删除和DDM逻辑库关联的实例，会直接导致逻辑库故障。
- 如需删除RDS for MySQL实例上的数据，请在弹窗中勾选“删除关联RDS实例的数据”。
- 若您想删除逻辑库，请首先确认RDS实例或GaussDB(for MySQL)实例是否存在。若实例已删除，请先单击“同步数据库信息”，再进行删除操作。
- 若您所连接的RDS实例有名称、引擎、引擎版本号、最大连接数max_connections、端口号、ip等信息的修改，不需要删除逻辑库，只需单击“同步数据库信息”同步最新配置。

----结束

7.5 配置 SQL 黑名单

概述

配置SQL黑名单即配置该逻辑库不允许执行的SQL语句。

前提条件

- 成功登录分布式数据库中间件服务控制台。
- DDM实例逻辑库且DDM实例运行正常。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，选择目标DDM实例，单击实例名称，进入实例基本信息页面。

步骤2 在实例基本信息页面左侧导航栏，选择“逻辑库列表”选项卡，查看DDM实例逻辑库。

步骤3 在逻辑库列表页面，单击右侧操作栏的“配置SQL黑名单”。

图 7-7 配置 SQL 黑名单

步骤4 在配置SQL黑名单弹窗中，单击“编辑”，按需输入前缀匹配、全量匹配、正则匹配的SQL信息，设置完成后单击“确定”即可。

说明

- 前缀匹配：禁止在对应逻辑库执行带有某些关键字的SQL语句，例如“DROP XXXX”,"DELETE XXX"。
- 全量匹配：禁止在对应逻辑库执行该SQL语句。
- 配置的黑名单SQL之间以英文分号隔开，前缀匹配与全量匹配中的SQL语句加起来大小不超过1kb。
- 若在配置SQL黑名单弹窗中清除之前编辑的前缀匹配与全量匹配中的SQL语句，并单击“确定”，则表示清空之前配置的SQL黑名单。

----结束

7.6 逻辑库扩容

随着业务增长，逻辑库存储空间不足，并发压力较大，此时可对DDM实例进行逻辑库扩容，增加实例，提高数据存储能力与并发支持能力。

本章节以RDS for MySQL实例为例说明逻辑库扩容使用方法。

7.6.1 平移扩容

概述

平移扩容：将逻辑库部分分片（分库）平移到新增的实例上，数据不会进行重分布。

前提条件

- DDM实例中已创建逻辑库。
- 已有MySQL实例且与DDM实例处于在相同的VPC，该MySQL实例没有被其它DDM实例使用。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，选择目标DDM实例，单击实例名称，进入实例基本信息页面。

步骤2 在实例基本信息页面左侧导航栏，选择“逻辑库列表”选项卡，查看DDM实例逻辑库。

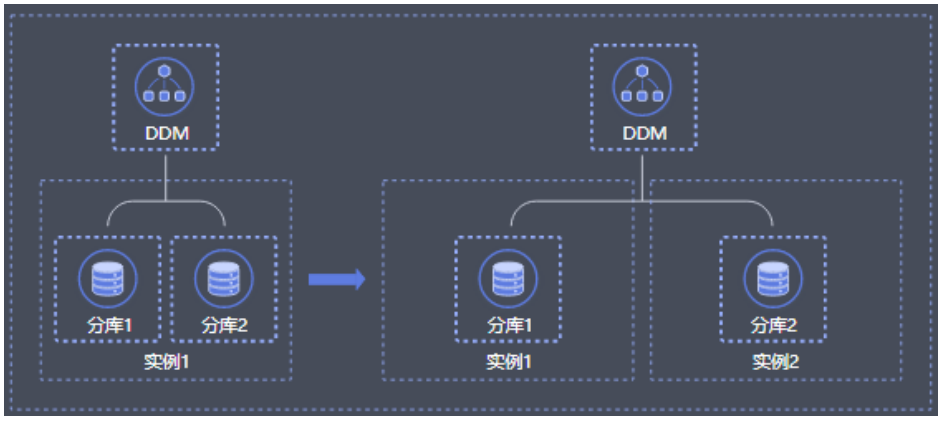
步骤3 在逻辑库列表页面，单击右侧操作栏的“扩容”进入扩容选择详情页面。

图 7-8 扩容

步骤4 在“扩容”页面选择“平移扩容”，根据需要选择“切换策略”、“数据清理”方式后，勾选目标实例，并输入数据库的密码，单击测试连接。连接通过后，单击“下一步：信息确认”。

图 7-9 平移扩容

图 7-10 原理图



步骤5 确认无误后，单击“下一步：预检查”。检查通过后，单击“开始扩容”即可。

在逻辑库列表页面，当“逻辑库状态”为“运行中”时，表示扩容成功，“已使用RDS”列将会呈现新扩容的RDS for MySQL实例。

说明

在进行扩容时，若DDM的实例版本低于2.4.1.3。在选择MySQL实例的参数sql_mode时，请不要选择ANSI_QUOTES。不能使用双引号来引用文字字符串，因为它们被解释为标识符。

例如：select * from test where tb = "logic"。

步骤6 在逻辑库列表页面，当“逻辑库状态”为“扩容失败”时，表示扩容失败。您可单击“重试”或者“回滚”进行操作。

说明

- 请在业务低峰时进行扩容。
- DDM扩容不支持无主键表。
- 只有“逻辑库状态”为“运行中”才能进行扩容。
- 一个DDM实例内，只允许同时对一个逻辑库进行扩容操作。
- 最多支持扩容256个RDS for MySQL实例。
- 可在“逻辑库列表”页查看扩容进度，扩容过程需要5分钟~30分钟，具体时长与实际需要迁移数据量相关。
- 扩容过程中，DDM会自动开启LOAD DATA功能，扩容结束后，您可根据需要自行在RDS for MySQL实例的参数模板设置里关闭LOAD DATA功能。

----结束

7.6.2 翻倍扩容

概述

翻倍扩容：逻辑库对应的物理库分片会扩为之前的两倍，会尽可能保证在每个数据库实例上的物理分片是均匀分布的， 数据一定会重新分布。

前提条件

- DDM实例中已创建逻辑库。
- 已有MySQL实例且与DDM实例处于在相同的VPC，该MySQL实例没有被其它DDM实例使用。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，选择目标DDM实例，单击实例名称，进入实例基本信息页面。

步骤2 在实例基本信息页面左侧导航栏，选择“逻辑库列表”选项卡，查看DDM实例逻辑库。

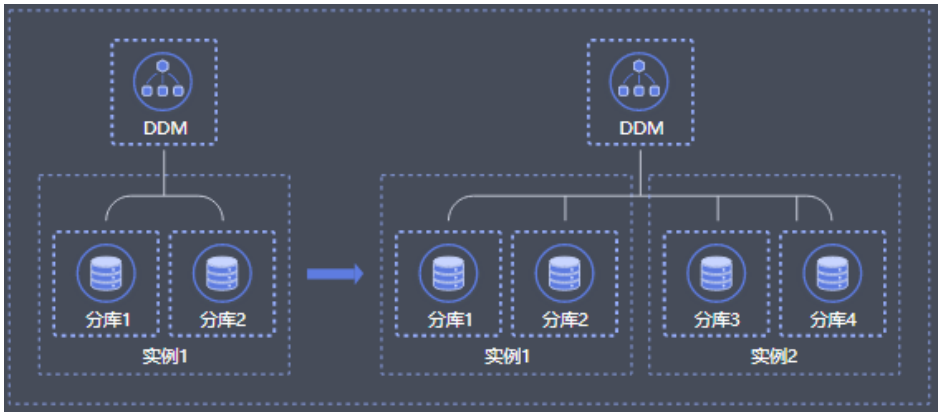
步骤3 在逻辑库列表页面，单击右侧操作栏的“扩容”进入扩容选择详情页面。

图 7-11 扩容

步骤4 在“扩容”页面选择“翻倍扩容”，根据需要选择“切换策略”、“数据清理”方式后，勾选目标实例，并输入数据库的密码，单击“测试连接”。连接通过后，单击“下一步：信息确认”。

图 7-12 翻倍扩容

图 7-13 翻倍扩容



步骤5 确认无误后，单击“下一步：预检查”。检查通过后，单击“开始扩容”即可。

在逻辑库列表页面，当“逻辑库状态”为“运行中”时，表示扩容成功，“已使用RDS”列将会呈现新扩容的RDS for MySQL实例。

说明

在进行扩容时，若DDM的实例版本低于2.4.1.3。在选择MySQL实例的参数sql_mode时，请不要选择ANSI_QUOTES。不能使用双引号来引用文字字符串，因为它们被解释为标识符。

例如：select * from test where tb = "logic"。

步骤6 在逻辑库列表页面，当“逻辑库状态”为“扩容失败”时，表示扩容失败。您可单击“重试”或者“回滚”进行操作。

说明

- 请在业务低峰时进行扩容。
- DDM扩容不支持无主键表。
- 只有“逻辑库状态”为“运行中”才能进行扩容。
- 一个DDM实例内，只允许同时对一个逻辑库进行扩容操作。
- 最多支持扩容256个RDS for MySQL实例。
- 可在“逻辑库列表”页查看扩容进度，扩容过程需要5分钟~30分钟，具体时长与实际需要迁移数据量相关。
- 扩容过程中，DDM会自动开启LOAD DATA功能，扩容结束后，您可根据需要自行在RDS for MySQL实例的参数模板设置里关闭LOAD DATA功能。

----结束

8 账号管理

8.1 创建账号

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 当前DDM实例有逻辑库。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“账号管理”，进入账号管理页面。

图 8-1 账号管理

步骤3 在账号管理页面单击“创建DDM账号”，在弹窗中填选账号参数信息。

图 8-2 创建 DDM 账号

表 8-1 创建 DDM 账号配置参数

参数	说明
账号名称	DDM账号的名称，命名规则如下。 长度为1-32个字符，必须以字母开头，可以包含字母，数字、下划线，不能包含其它特殊字符。
密码	DDM账号的密码，密码复杂度要求如下。 • 长度为8~32个字符 • 至少包含三种字符组合：小写字母、大写字母、数字、特殊字符 ~ ! @ # % ^ * - _ = + ?
确认密码	-

参数	说明
账号权限	账号权限分为基础权限和扩展权限。 ● 基础权限：CREATE、DROP、ALTER、INDEX、INSERT、DELETE、UPDATE、SELECT ● 扩展权限：全表查询、全表删除、全表更新。 权限配置遵循如下原则： ● 基础权限用户可以按需选择。 ● 扩展权限对应的基础权限必须选择，对应关系如下： – “全表查询”对应“SELECT” – “全表删除”对应“DELETE” – “全表更新”对应“UPDATE”
关联逻辑库	DDM账号与逻辑库关联绑定，下拉列表中显示可关联的逻辑库。 DDM账号只对已关联的逻辑库有访问权限。
描述	对DDM账号的详细描述信息，长度不能超过256个字符。

步骤4 确认填写无误后，单击“确定”。

----结束

8.2 修改账号信息

前提条件

成功登录分布式数据库中间件服务控制台。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤3 账号管理页面选择目标账号，在其操作栏单击“修改”。

图 8-3 修改账号

步骤4 在修改账号弹窗中，您可以修改关联逻辑库、账号权限及描述信息。

图 8-4 修改 DDM 账号

步骤5 编辑完成账号信息后，单击“确定”，保存修改信息。

----结束

8.3 删除账号

前提条件

成功登录分布式数据库中间件服务控制台。

说明

删除操作无法恢复，请谨慎操作。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤3 账号管理页面选择目标账号，在其操作栏单击“更多 > 删除”。

图 8-5 删除 DDM 账号

步骤4 在删除账号确认弹窗中，单击“是”，删除账号信息。

----结束

8.4 重置密码

前提条件

成功登录分布式数据库中间件服务控制台。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤3 账号管理页面选择目标账号，在其操作栏单击“更多 > 重置密码”。

图 8-6 重置 DDM 账号密码

步骤4 在输入新密码并再次输入“确认密码”后，单击“确认”，进行密码重置。

----结束

9 备份管理

9.1 数据恢复

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 当前实例状态为“运行中”。
- 支持GaussDB(for MySQL)实例和RDS实例，以RDS实例为例说明。

操作步骤

步骤1 在当前DDM实例所在区域创建一个新的DDM实例或者寻找一个满足限制的实例。

步骤2 创建或寻找与当前DDM实例下相同数量的RDS实例。

说明

- 新建的DDM实例不能挂载RDS实例
- 寻找的实例不能存在逻辑库和DDM账号。
- 每个新创建的RDS实例引擎版本为MySQL5.7版，其小版本号不得低于源DDM下RDS版本号。
- 每个实例存储空间不得小于当前DDM实例下的RDS空间。

步骤3 在DDM实例列表页面单击**1**中的当前实例名称，进入实例基本信息页面。

步骤4 在基本信息页面右上角单击“恢复数据”。进入恢复数据页签。

步骤5 在数据恢复页签选择“可恢复时间段”和“可恢复时间点”，并选择**1**中创建的DDM实例作为目标实例。

步骤6 选择**2**中创建的RDS实例作为目标RDS实例，选中确认复选框。确认无误后，单击“确认”，等待1-3分钟数据恢复完成。

说明

- 恢复到目标DDM实例会导致实例数据被覆盖，且恢复过程中目标实例数据库不可用。
- 用户选择的可恢复时间点到当前时间之间需要保证没有做过创建逻辑库和删除逻辑库操作。
- 只可选择与原实例相同引擎、相同版本或高版本的实例，且存储空间大于等于原实例的已有实例进行恢复。

----结束

9.2 一致性备份

9.2.1 备份

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 当前实例状态为“运行中”。
- DDM实例已经创建了逻辑库。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“一致性备份”，进入备份页面。

步骤3 在一致性备份页面单击“创建备份”，进入创建备份页面。

步骤4 在创建备份页面输入备份名称，简要描述备份的原因，确认无误后，单击确认即可。

说明

备份时间的长度取决于实例内部数据量的大小，请耐心等待。

----结束

9.2.2 恢复

前提条件

- 成功登录分布式数据库中间件服务控制台。
- 当前实例状态为“运行中”。
- DDM实例已经创建了逻辑库。
- DDM实例已全量备份完成。

操作步骤

步骤1 在分布式数据库中间件服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤2 在左侧导航栏选择“一致性备份”，进入备份页面。

步骤3 在一致性备份页面单击“恢复”，进入一致性备份恢复页面。

步骤4 在一致性备份恢复页面选择目标DDM、目标RDS实例。确认无误后单击“确定”即可。

说明

- 备份恢复时间的长度取决于实例内部数据量的大小，请耐心等待。
- 一致性备份恢复的目标DDM实例不可以有DDM账号。

----结束

10 监控管理

10.1 监控指标

功能说明

本节定义了分布式数据库中间件服务上报云监控的监控指标的命名空间，监控指标列表和维度定义，您可以通过云监控提供的API接口来检索DDM产生的监控指标信息。

监控指标

指标名称	含义	取值范围	测量对象和监控对象	监控周期（原始指标）
CPU使用率	该指标用于统计当前DDM节点的CPU利用率。	0~100%	DDM节点	1分钟
内存使用率	该指标用于统计当前DDM节点的内存使用率。	0~100%	DDM节点	1分钟
网络输入吞吐量	该指标用于统计当前DDM节点平均每秒的输入流量。	≥ 0 bytes/s	DDM节点	1分钟
网络输出吞吐量	该指标用于统计当前DDM节点平均每秒的输出流量。	≥ 0 bytes/s	DDM节点	1分钟
QPS	该指标用于统计当前DDM节点的每秒请求数。	≥ 0 counts	DDM节点	1分钟
读写比例	该指标用于统计当前DDM节点在每个采集周期内的读写比例。	0-100%	DDM节点	1分钟

指标名称	含义	取值范围	测量对象和监控对象	监控周期（原始指标）
读次数	该指标用于统计当前DDM节点在每个采集周期内新增的读次数。	≥ 0 counts/s	DDM节点	1分钟
写次数	该指标用于统计当前DDM节点在每个采集周期内新增的写次数。	≥ 0 counts/s	DDM节点	1分钟
慢SQL	该指标用于统计当前DDM节点在每个采集周期内新增的慢SQL条数。	≥ 0 counts	DDM节点	1分钟
平均响应时延	该指标用于统计当前DDM节点的SQL平均响应时延。	≥ 0 ms	DDM节点	1分钟
连接数	该指标用于统计当前DDM节点的连接数。	≥ 0 counts	DDM节点	1分钟
后端连接池水位	该指标用于统计当前DDM节点后端活跃连接数与后端最大连接数的比例。	0-100%	DDM节点	1分钟

维度

Key	Value
ddm_node_id	DDM节点。

10.2 查看监控指标

操作场景

监控管理控制台提供了对DDM实例的监控管理，包括读写比例监控与慢SQL监控，您可以根据监控反馈结果，对数据库进行调优。

- 慢SQL是指在数据库上执行时间比较耗时的慢SQL语句。通过对SQL统计分析有助于发现数据库服务的性能瓶颈。
- 读写比例是指在一张逻辑表上的读写比例，即读写比例=读请求的个数/读写总请求的个数。

前提条件

- 成功登录分布式数据库中间件服务控制台。

- 已创建DDM实例、逻辑库。

操作步骤

步骤1 在分布式数据库中间件服务实例管理列表页面，单击进入目标实例。

步骤2 单击左侧菜单栏的“监控管理”页签。

图 10-1 监控管理

步骤3 选择需要查看监控的类型：“慢日志监控”、“读写比例监控”等监控详情。

- 慢日志监控

  您可以通过时间范围进行筛选。在慢SQL监控列表中查看DDM账号、逻辑库、SQL、开始执行时间、执行时长、分片、影响行等信息。

图 10-2 慢日志监控

表 10-1 慢日志监控

监控参数名称	监控参数说明
DDM账号	执行慢SQL的DDM账号。
逻辑库名称	实例逻辑库名称。
SQL	DDM实例上执行时长比较耗时的慢SQL语句。
开始执行时间	选择的时间范围内第一条业务慢SQL语句开始执行的时间。
执行时长	单位：ms。
分片	SQL执行涉及的所有分片。
影响行	SQL执行涉及的行数。

- 读写比例监控

您可以通过时间范围进行筛选读写比例监控信息，获取读次数、写次数以及最后执行时间信息。

图 10-3 读写比例监控

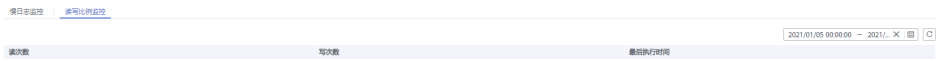


表 10-2 读写比例监控

监控参数名称	监控参数说明
读次数	对该逻辑表的读操作次数。您可在界面上选择指定时间段进行查询和统计。
写次数	对该逻辑表的写操作次数。您可在界面上选择指定时间段进行查询和统计。
最后执行时间	选择的时间范围内最后一条业务SQL语句的执行时间。

步骤4 单击“查看监控指标”。进入“云监控服”务页面。

图 10-4 查看监控指标

步骤5 您可在云监控服务页面查看指标。

1. 在>“云服务监控”导航栏，获取云服务监控列表信息。
2. 选择您所需查看的实例名称，单击 可展开当前实例下节点信息列表，并在目标实例节点右侧操作栏单击“查看监控指标”，进入该节点监控指标详情页面。
3. 您可在监控指标信息页面，根据时间范围查看各项指标详细信息。

----结束

11 审计

11.1 支持审计的关键操作列表

通过，您可以记录与分布式数据库中间件实例相关的操作事件，便于日后的查询、审计和回溯。

表 11-1 支持的 DDM 操作列表

操作名称	资源类型	事件名称
参数模板应用	parameterGroup	applyParameterGroup
清理逻辑库扩容后的元数据	logicDB	cleanMigrateLogicDB
清理用户资源	all	cleanupUserAllResources
比较参数模板	parameterGroup	compareParameterGroup
复制参数模板	parameterGroup	copyParameterGroup
创建实例	instance	createInstance
创建逻辑库	instance	createLogicDB
创建参数模板	parameterGroup	createParameterGroup
创建账号	instance	createUser
企业项目迁入/迁出	all	dealResourceTag
删除实例	instance	deleteInstance
删除逻辑库	logicDB	deleteLogicDB
删除参数模板	parameterGroup	deleteParameterGroup

操作名称	资源类型	事件名称
删除账号	instance	deleteUser
节点扩容	instance	enlargeNode
重启实例	instance	instanceRestart
导入逻辑库信息	logicDB	loadMetadata
扩容路由切换	logicDB	manualSwitchRoute
逻辑库扩容	logicDB	migrateLogicDB
修改参数模板	parameterGroup	modifyParameterGroupTempate
修改路由切换时间	logicDB	modifyRouteSwitchTime
修改账号信息	instance	modifyUser
节点缩容	instance	reduceNode
重置参数模板	parameterGroup	resetParameterGroup
重置账号密码	instance	resetUserPassword
规则变更	instance	resizeFlavor
恢复实例	instance	restoreInstance
重试逻辑库扩容	logicDB	retryMigrateLogicDB
回滚DDM实例版本	instance	rollback
回滚逻辑库扩容	logicDB	rollbackMigrateLogicDB
保存参数模板	parameterGroup	saveParameterGroup
访问控制	instance	switchIpGroup
同步数据库信息	instance	synRdsinfo
升级DDM实例版本	instance	upgrade

11.2 查看追踪事件

操作场景

在您开通了云审计服务后，系统开始记录云服务资源的操作。云审计服务管理控制台保存最近7天的操作记录。

本节介绍如何在管理控制台查看最近7天的操作记录。

操作步骤

步骤1 登录管理控制台。

步骤2 选择“管理与部署 > 云审计服务”，进入云审计服务信息页面。

步骤3 单击左侧导航树的“事件列表”，进入事件列表信息页面。

步骤4 事件列表支持通过筛选来查询对应的操作事件。详细信息如下：

- 事件来源、资源类型和筛选类型：在下拉框中选择查询条件。
  其中筛选类型选择资源ID时，还需选择或者手动输入某个具体的资源ID。
- 操作用户：在下拉框中选择某一具体的操作用户。
- 事件级别：可选项为“所有事件级别”、“normal”、“warning”、“incident”，只可选择其中一项。
- 时间范围：可通过选择时间段查询操作事件。

步骤5 选择查询条件后，单击“查询”。

步骤6 在需要查看的记录左侧，单击展开该记录的详细信息。

步骤7 在需要查看的记录右侧，单击“查看事件”，在弹出框中显示该操作事件结构的详细信息。

步骤8 单击右侧的“导出”，将查询结果以CSV格式的文件导出，该CSV文件包含了云审计服务记录的七天以内的操作事件的所有信息。

关于事件结构的关键字段详解，请参见《云审计服务用户指南》的“事件结构”和“事件样例”章节。

----结束

12 SQL 语法

12.1 简介

DDM兼容MySQL协议及其语法，但因分布式数据库与单机数据库之间存在一定的差异性，导致SQL使用存在些限制。

在评估DDM方案之前，请先完成当前应用中的SQL语法及与DDM支持语法的兼容性评估。

MySQL EXPLAIN

当您在需要执行的SQL语句前加上EXPLAIN，然后执行SQL时，您将会看到其具体的执行计划，以此分析耗时，进而修改SQL，达到优化的效果。

表 12-1 EXPLAIN 列的解释

列名称	描述
table	显示该行数据所归属的表。
type	显示连接使用了何种类型。从最好到最差的连接类型为const、eq_reg、ref、range、index和ALL。
possible_keys	显示可能应用在该表中的索引。
key	实际使用的索引。如果为NULL，则没有使用索引。个别情况下，MySQL会选择优化不足的索引。在SELECT语句中使用USE INDEX（indexname）来强制使用一个索引或者用IGNORE INDEX（indexname）来强制MySQL忽略索引。
key_len	使用的索引的长度。在不损失精确性的情况下，长度越短越好。
ref	显示索引被使用的列，通常为一个常数。
rows	MySQL用来返回请求数据的行数。
Extra	关于MySQL如何解析查询的额外信息。

SQL 大类限制

- 不支持临时表；
- 不支持外键、视图、游标、触发器及存储过程；
- 不支持用户自定义数据类型及自定义函数；
- 不支持“IF”，“WHILE” 等流程控制类语句；
- 不支持 BEGIN…END、LOOP…END LOOP、REPEAT…UNTIL…END REPEAT、WHILE…DO…END WHILE 等复合语句。

DDL 语法

- 拆分表和广播表不支持外键；
- 不支持修改分片字段（拆分键）；
- 不支持ALTER DATABASE Syntax；
- 不支持从另一张表创建新的拆分表、广播表；
- create table不支持generated column语法；
- 不支持ALTER命令修改拆分键和全局序列字段；
- 不支持创建TEMPORARY类型的拆分表、广播表；
- 逻辑表名只允许字母(不区分大小写)数字以及下划线
- CREATE TABLE tbl_name LIKE old_tbl_name 不支持；
- CREATE TABLE tbl_name SELECT statement 不支持；
- 不支持insert into on duplicate key update 更新拆分键值；
- 暂不支持跨Schema的DDL, 例如， CREATE TABLE db_name.tbl_name (…)；
- 使用MySQL关键字或保留字做表名、列名、索引名等标识符时，需要使用反引号扩起来；

DML 语法

- 不支持PARTITION子句；
- 不支持 UPDATE 使用子查询；
- 不支持 INSERT DELAYED Syntax；
- 不支持 STRAIGHT_JOIN 和 NATURAL JOIN；
- 受限支持 跨分片 UPDATE 多表需要join update的表需要有PK；
- 受限支持 跨分片 DELETE 多表中的数据，需要Join delete的表需要有PK；
- 不支持 SQL 中对于变量的引用和操作，比如 SET @c=1, @d=@c+1; SELECT @c, @d。

函数

- 不支持 XML 函数；
- 不支持 GTID 函数；
- 不支持全文检索函数；
- 不支持企业加密函数；
- 不支持row_count()函数。

子查询

不支持 HAVING 子句中的子查询，JOIN ON 条件中的子查询；

数据类型

不支持空间数据类型。

12.2 DDL

DDM支持通用的DDL操作：建库，建表，修改表结构等，但实现方式与普通的MySQL数据库有所区别。

在 MySQL 客户端执行 DDL 操作

- TRUNCATE Syntax；
  举例：
  TRUNCATE TABLE t1
  表示清空表格t1。
  TRUNCATE会将表完全清空，它需要DROP权限。在逻辑上类似于删除所有行的DELETE语句。
- ALTER TABLE Syntax；
  举例：
  ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
  表示更改表t2的结构:删除c列和d列。
  ALTER可以添加或删除列、创建或销毁索引、更改现有列的类型或重命名列或表本身。还可以更改特性，如用于表或表注释的存储引擎。
- DROP INDEX Syntax；
  举例：
  DROP INDEX 'PRIMARY' ON t;
  表示删除表t中的主键。
  DROP INDEX即从表tbl_name中删除名为index_name的索引。
- CREATE INDEX Syntax；
  举例：
  CREATE INDEX part_of_name ON customer (name(10));
  表示使用name列的前10个字符创建索引（假设name具有非二进制字符串类型）。
  CREATE INDEX用于向现有表添加索引。
- CREATE/DROP DATABASE。
  举例：
  CREATE DATABASE t shard 100;
  表示创建个具有100个分片数的数据库t。
  CREATE DATABASE即使用给定名称创建一个数据库。

12.2.1 创建表

分库分表

假设使用HASH的拆分库算法，拆分表算法为MOD_HASH，样例如下：

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

分库不分表

假设使用HASH的拆分库算法，样例如下：

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

广播表

如下为创建广播表的样例：

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

单表

创建单表样例如下，不做任何拆分：

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

12.2.2 拆分算法概述

支持的拆分算法概览

DDM是一个支持既分库又分表的数据库服务，目前DDM分库函数与分表函数的支持情况如下：

表 12-2 拆分算法概览表

拆分函数	描述	能否用于分库	能否用于分表
MOD_HASH	简单取模	是	是
UNI_HASH	简单取模	是	是
RIGHT_SHIFT	数值向右移	是	是
YYYYMM	按年月哈希	是	是
YYYYDD	按年日哈希	是	是
YYYYWEEK	按年周哈希	是	是
HASH	简单取模	是	是
RANGE	按范围	是	否
MM	按月份哈希	否	是

拆分函数	描述	能否用于分库	能否用于分表
DD	按日期哈希	否	是
MMDD	按月日哈希	否	是
WEEK	按星期哈希	否	是

说明

- 分库的拆分键及分表的拆分键，均不支持为空。
- 在DDM中，一张逻辑表的拆分方式是由拆分函数（包括分片数目与路由算法）与拆分键（包括拆分键的MySQL数据类型）共同定义。
- 当一张逻辑表的分库拆分方式与分表拆分方式不一致时，若SQL查询没有同时带上分库条件与分表条件，则DDM在查询过程会产生全分库扫描或全分表扫描的操作。

DDL 拆分函数的数据类型

DDM 的拆分函数对各数据类型对支持情况有所不同，下表显示了 DDM 拆分函数对各种数据类型的支持情况。

表 12-3 DDM 拆分函数对各种数据类型的支持情况

拆分算法	TINYINT	SMALLINT	MEDIUMINT	INT	BIGINT	CHAR	VARCHAR	DATE	DATETIME	TIMESTAMP	OTHERS
HASH	√	√	√	√	√	√	√	√	√	√	×
UNI_HASH	√	√	√	√	√	√	√	×	×	×	×
RIGHT_SHIFT	√	√	√	√	√	×	×	×	×	×	×
YYYYDD	×	×	×	×	×	×	×	√	√	√	×
YYYYWEEK	×	×	×	×	×	×	×	√	√	√	×
YYYYMM	×	×	×	×	×	×	×	√	√	√	×

DDM 的拆分函数创表语法

DDM兼容MySQL的创表语法，并新增加了partition_options的分库分表关键字。

```
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]
     ]
```

12.2.3 拆分算法使用说明

12.2.3.1 MOD_HASH 算法

使用说明

拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL, NUMERIC）或字符串类型。

路由方式

根据分库键的键值按分库数取余，使用HASH分库但不使用同一个拆分键进行HASH分表时，分库键的键值按分库数取余。如果键值是字符串，则字符串会被计算成哈希值再对分库/表数取余。

例如，MOD_HASH('6')等价于6 % D（D是分库数目）。

分库和分表使用同一个拆分键进行HASH时，拆分键的键值按总的分表数取余。

例如，有2个分库，每个分库4张分表，那么0库上保存分表0~3，1库上保存分表4~7。某个键值为15，15 % (2 * 4) =7，所以15被分到1库的表7上。

算法计算方式

方式一：拆分键是整型

表 12-4 拆分键是整型时的计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = 分库拆分键值 % 分库数 分表路由结果 = 分表拆分键值 % 分表数	分库： 16 % 8 = 0 分表：16 % 3 = 1
分库拆分键 = 分表拆分键	分表路由结果 = 拆分键值 % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	分表：16 % (8 * 3) = 16 分库： 16 / 3 = 5

方式二：拆分键是字符类型

表 12-5 拆分键是字符型时的计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = hash(分库拆分键值) % 分库数 分表路由结果 = hash(分表拆分键值) % 分表数	hash（‘abc’)=‘abc’.hashCode()=96354 分库 :96354 % 8 = 2; 分表 :96354 % 3 = 0;
分库拆分键 = 分表拆分键	分表路由结果 = hash(拆分键值) % (分库数 * 分表数) 分库路由结果 =分表路由结果 / 分表数	hash（‘abc’)=‘abc’.hashCode()=96354 分表 :96354% (8 * 3) = 18 分库 :18 / 3=6

建表语法

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

注意事项

- 拆分键和键值皆不能修改。
- MOD_HASH算法是简单取模，要求拆分列的值自身分布均衡才能保证哈希均衡。

12.2.3.2 MOD_HASH_CI 算法

须知

Mod_hash_ci算法除了不区分大小写外，其他与mod_hash算法完全一致。

比如：拆分键值’abcd’与拆分键值’ABCD’的路由结果在mod_hash_ci算法中是一致的。

使用说明

拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL, NUMERIC）或字符串类型。

路由方式

根据分库键的键值按分库数取余，使用HASH分库但不使用同一个拆分键进行HASH分表时，分库键的键值按分库数取余。如果键值是字符串，则字符串会被计算成哈希值再对分库/表数取余。

例如，MOD_HASH('6')等价于6 % D（D是分库数目）。

分库和分表使用同一个拆分键进行HASH时，拆分键的键值按总的分表数取余。

例如，有2个分库，每个分库4张分表，那么0库上保存分表0~3，1库上保存分表4~7。某个键值为15，15 % (2 * 4) =7，所以15被分到1库的表7上。

算法计算方式

方式一：拆分键是整型

表 12-6 拆分键是整型时的计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = 分库拆分键值 % 分库数 分表路由结果 = 分表拆分键值 % 分表数	分库： 16 % 8 = 0 分表：16 % 3 = 1
分库拆分键 = 分表拆分键	分表路由结果 = 拆分键值 % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	分表：16 % (8 * 3) = 16 分库： 16 / 3 = 5

方式二：拆分键是字符类型

表 12-7 拆分键是字符型时的计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = hash(分库拆分键值) % 分库数 分表路由结果 = hash(分表拆分键值) % 分表数	hash（‘abc’)=‘abc’.hashCode()=96354 分库 :96354 % 8 = 2; 分表 :96354 % 3 = 0;
分库拆分键 = 分表拆分键	分表路由结果 = hash(拆分键值) % (分库数 * 分表数) 分库路由结果 =分表路由结果 / 分表数	hash（‘abc’)=‘abc’.hashCode()=96354 分表 :96354% (8 * 3) = 18 分库 :18 / 3=6

建表语法

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

注意事项

- 拆分键和键值皆不能修改。
- MOD_HASH_CI算法是简单取模，要求拆分列的值自身分布均衡才能保证哈希均衡。

12.2.3.3 UNI_HASH 算法

适用场景

适用于需要按用户ID或订单ID进行分库的场景。

使用说明

拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL, NUMERIC）或字符串类型。

路由方式

根据分库键的键值按分库/表数取余。如果键值是字符串，则字符串会被计算成哈希值再对分库/表数取余。

例如，UNI_HASH('6')等价于6 % D（D是分库数目）。

算法计算方式

方式一：拆分键是整型

表 12-8 拆分键是整型时的计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = 分库拆分键值 % 分库数 分表路由结果 = 分表拆分键值 % 分表数	分库： 16 % 8 = 0 分表：16 % 3 = 1
分库拆分键 = 分表拆分键	分库路由结果 = 拆分键值 % 分库数 分表路由结果 = (拆分键值%分库数)*分表数+(拆分键值 /分库数)%分表数	分库： 16 % 8 = 0 分表：(16 % 8) *3 + (16/8)%3 = 2

方式二：拆分键是字符类型

表 12-9 拆分键是字符型时的计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = hash(分库拆分键值) % 分库数 分表路由结果 = hash(分表拆分键值) % 分表数	hash（‘abc’)=‘abc’.hashCode()=96354 分库 :96354 % 8 = 2; 分表 :96354 % 3 = 0;
分库拆分键 = 分表拆分键	分库路由结果 = hash(拆分键值) % 分库数 分表路由结果 = (hash(拆分键值) %分库数)*分表数+(hash(拆分键值) /分库数)%分表数	分库 :96354% 8 = 2 分表 :(96354 % 8) *3 + (96354/8)%3 = 8

建表语法

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

注意事项

- 拆分键和键值皆不能修改。
- UNI_HASH算法是简单取模，要求拆分列的值自身分布均衡才能保证哈希均衡。

与 MOD_HASH 的比较

- 在使用 UNI_HASH 分库但不分表时，UNI_HASH 和 MOD_HASH 的路由方式一样，都是根据分库键的键值按分库数取余。
- 分库和分表都使用同一个拆分键进行 MOD_HASH 时，随着分表数的变化，同一个键值分到的分库不是固定的。
- 分库和分表都使用同一个拆分键进行 UNI_HASH 时，无论分表数是多少，同一个键值总是分到相同的分库。
- 如果两张逻辑表需要根据同一个拆分键进行分库分表，但分表数不同，那么当两张表按该拆分键进行 JOIN 时，如果使用 MOD_HASH 会出现跨库 JOIN，而 UNI_HASH 不会有跨库 JOIN。

12.2.3.4 RIGHT_SHIFT 算法

适用场景

当拆分键大部分键值的低位部位区分度比较低而高位部分区分度比较高时，则适用于通过此拆分算法提高散列结果的均匀度。

使用说明

拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL, NUMERIC）。

路由方式

根据拆分键的键值（键值必须是整数）有符号地向右移二进制移指定的位数（位数由用户通过 DDL 指定），然后将得到的整数值按分库/分表数目取余。

算法计算方式

表 12-10 计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = 分库拆分键值 % 分库数 分表路由结果 = 分表拆分键值 % 分表数	分库：(123456 >> 4) % 8 = 4 分表：(123456 >> 4) % 3 = 0
分库拆分键 = 分表拆分键(拆分键)	分库路由结果 = 拆分键值 % 分库数 分表路由结果 = (拆分键值%分库数)*分表数+(拆分键值 /分库数)%分表数	分库：(123456 >> 4) % 8 = 4 分表：((123456 >> 4) % 8) *3 + ((123456 >> 4)/8)%3= 13

建表语法

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

注意事项

- 拆分键和键值皆不能修改。
- 移位的数目不能超过整数类型所占有的位数目，路由到0分片。

12.2.3.5 YYYYMM 算法

适用场景

适用于需要按年份与月份进行分库的场景。

使用说明

拆分键的数据类型必须是DATE / DATETIME / TIMESTAMP其中之一。

路由方式

根据拆分键的时间值的年份与月份计算哈希值，然后再按分库数取余。

例如，YYYYMM(‘2018-12-31 12:12:12’) 等价于 (2018 * 12 + 12) % D（D是分库数目）。

算法计算方式

表 12-11 计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	拆分键：yyyy-MM-dd 分库路由结果 = (yyyy * 12 + MM) % 分库数 分表路由结果 = (yyyy * 12 + MM) % 分表数	拆分键：2012-11-20 分库 ： (2012 * 12 + 11) % 8 = 3 分表：(2012 * 12 + 11) % 3 = 2
分库拆分键 = 分表拆分键(拆分键)	拆分键：yyyy-MM-dd 分表路由结果 = (yyyy*12+MM) % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	拆分键：2012-11-20 分表：(2012 * 12 + 11) % (8 * 3) = 11 分库 ： 11 % 3 = 2

建表语法

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

注意事项

- 拆分键和键值皆不能修改。
- YYYYMM算法不支持对于每一个年月都独立对应一张分库。
- 当月份经历一个轮回（如 2013-03 是 2012-03 的一个轮回）后，同一个月份有可能被路由到同一个分库，请以实际的分库数目而定。

12.2.3.6 YYYYDD 算法

适用场景

适用于需要按年份与一年的天数进行分库的场景。

使用说明

拆分键的数据类型必须是DATE / DATETIME / TIMESTAMP其中之一。

路由方式

根据拆分键的时间值的年份与一年的天数计算哈希值，然后再按分库/表数取余。

例如，YYYYDD(‘2018-12-31 12:12:12’) 等价于 (2018 * 366 + 365) % D（D是分库数目）。

说明

计算得出“2018-12-31”是 2018 年第 365 天。计算公式中“2018*366”的“366”为固定取值。

算法计算方式

表 12-12 计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	拆分键：yyyy-MM-dd 分库路由结果 = (yyyy * 366 + 一年第几天) % 分库数 分表路由结果 = (yyyy * 366 + 一年第几天) % 分表数	拆分键：2012-12-31 分库： (2012 * 366 + 366) % 8 = 6 分表：(2012 * 366 + 366) % 3 = 0
分库拆分键 = 分表拆分键(拆分键)	拆分键：yyyy-MM-dd 分表路由结果 = (yyyy * 366 + 一年第几天) % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	拆分键：2012-12-31 分库： (2012 * 366 + 366) % (8*3) = 6 分库： 6 / 3 = 2

建表语法

```
create table yyyydd_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
update_time datetime DEFAULT NULL,
    primary key(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by YYYYDD(create_time)
tbpartition by YYYYDD(update_time) tbpartitions 30;
```

注意事项

- 拆分键和键值皆不能修改。
- YYYYDD算法不支持对于每一个年天都独立对应一张分库。
- 当日期经历一个轮回（如 2013-03 是 2012-03 的一个轮回）后，同一个日期有可能被路由到同一个分库，请以实际的分库数目而定。

12.2.3.7 YYYYWEEK 算法

适用场景

适用于需要按年份与一年的周数进行分库的场景。

使用说明

拆分键的数据类型必须是DATE / DATETIME / TIMESTAMP其中之一。

路由方式

根据拆分键的时间值的年份与一年的周数计算哈希值，然后再按分库/表数取余。

例如，YYYYWEEK(‘2018-12-31 12:12:12’) 等价于 (2019 * 54 + 1) % D（D是分库数目）。

说明

此处“2018-12-31”是2019年第一周。计算公式中“2019*54”的“54”为固定取值。

算法计算方式

表 12-13 计算方式

条件	计算方式	举例
分库拆分键 ≠ 分表拆分键	拆分键：yyyy-MM-dd 分库路由结果 = (yyyy * 54 + 一年第几周) % 分库数 分表路由结果 = (yyyy * 54 + 一年第几周) % 分表数	拆分键：2012-12-31 分库：(2013 * 54 + 1) % 8 = 7 分表：(2013 * 54 + 1) % 3 = 1
分库拆分键 = 分表拆分键(拆分键)	拆分键：yyyy-MM-dd 分表路由结果 = (yyyy * 54 + 一年第几周) % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	拆分键：2012-12-31 分库：(2013 * 54 + 1) % (8*3) = 7 分库：7 / 3 = 2

建表语法

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

注意事项

- 拆分键和键值皆不能修改。
- YYYYWEEK算法不支持对于每一个年周都独立对应一张分库。
- 当周数经历一个轮回（如2013年第一周是2012年第一周的一个轮回）后，相同周数有可能被路由到同一个分库，请以实际的分库数目而定。

12.2.3.8 HASH 算法

适用场景

适用于需要将数据均匀分布的场景或需要按时间（年、月、日、周及其组合）对数据进行拆分的场景，在SQL查询条件中，使用“=”、“IN”之类运算符相对较多。

使用说明

拆分键为表字段或表字段+日期函数。若拆分键为表字段+日期函数，其中的数据表字段必须是日期类型（DATE / DATETIME / TIMESTAMP），日期函数适用于需要按时间（年、月、日、周及其组合）对数据进行拆分的场景。

路由方式

首先102400对分库数/分表数进行分范围。

假如逻辑库分8个分片，那么102400/8=12800，则每一个分片对应的范围是：0=0-12799，1=12800-25599，2=25600-38399，3=38400-51199，4=51200-63999，5=64000-76799，6=76800-89599，7=89600-102399。

当计算路由结果时，计算拆分键值的CRC32值然后对102400取余，根据计算结果落到那个范围进行路由。

算法计算方式

方式一：拆分键非日期类型

表 12-14 拆分键非日期类型

条件	计算方式	举例
拆分键非日期类型	分库路由结果 = crc32(分库拆分键值) % 102400 分表路由结果 = crc32(分表拆分键值) % 102400	分库/分表：crc32(16) % 102400 = 49364; 49364属于3=38400-51199,则路由到分片3

方式二：拆分键是日期类型

表 12-15 支持的日期函数

日期函数	计算方式	举例
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)=该日期是今年的第几周	weekofyear (‘2019-10-11’)=41
day()	day(yyyy-MM-dd)=该日期是月份的第几天	day (‘2019-10-11’)=11

表 12-16 拆分键是日期类型

条件	计算方式	举例
拆分键是日期类型	分库路由结果 = crc32(日期函数(分库拆分键值)) % 102400 分表路由结果 = crc32(日期函数(分库拆分键值)) % 102400	分库/分表： crc32(year（‘2019-10-11’)) % 102400 = 5404; 5404属于0=0-12799,则路由到分片0

建表语法

假设用户需要对ID列按HASH函数进行分库不分表:

```
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);
```

假设用户需要对ID列按HASH函数既分库又分表:

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

注意事项

拆分键和键值皆不能修改。

12.2.3.9 Range 算法

适用场景

适用于范围类操作较多的场景。在SQL查询条件中，使用“>”、“<”、“BETWEEN ... AND ...”之类运算符相对较多。

使用说明

拆分键的类型只支持整型类型、日期类型和日期函数结合，若使用日期函数，拆分键的数据类型必须是date、datetime、timestamp其一。

拆分键为表字段或表字段+日期函数。若拆分键为表字段+日期函数，其中数据表字段必须是日期类型（DATE / DATETIME / TIMESTAMP），日期函数适用于需要按时间（年、月、日、周及其组合）对数据进行拆分的场景。

路由方式

根据拆分键，按照算法元数据的规则将数据行存储到相应的分片上。

算法计算方式

方式一：拆分键是整型

表 12-17 拆分键是整型时的计算方式

条件	计算方式	举例
拆分键是整型	分库路由结果 = 根据分库拆分键值在设定的元数据的范围进行路由 分表路由结果 = 根据分表拆分键值在设定的元数据的范围进行路由	分库： 拆分值为3属于3-4=1，则路由到1分片 分表：拆分值为5属于5-6=2，则路由到2分片

方式二：拆分键是日期类型

表 12-18 支持的日期函数

日期函数	计算方式	举例
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)=该日期是今年的第几周	weekofyear(‘2019-10-11’)=41
day()	day(yyyy-MM-dd)=该日期是月份的第几天	day (‘2019-10-11’)=11

表 12-19 拆分键是日期类型时的计算方式

条件	算法	举例
拆分键是日期类型	分库路由结果 =根据日期函数(分库拆分键值)在设定的元数据的范围进行路由 分表路由结果 =根据日期函数(分表拆分键值)在设定的元数据的范围进行路由	分库/分表： month(2019-10-11)=10属于9-10=4，则路由到4分片

建表语法

```
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
};
```

注意事项

- 拆分键和键值皆不能修改。
- range可以作为分库算法，不能作为分表算法。

12.2.3.10 MM 算法

适用场景

MM 适用于按月份数进行分表，分表的表名就是月份数。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值的月份数进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库键确定分库后，确定分表的计算方式是：月份mod分表数，即：1 mod 12 = 1。

算法计算方式

表 12-20 算法举例

条件	计算方式	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值：2019-1-15 分表：1 % 12 = 1

建表语法

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

注意事项

按 MM 进行分表，由于一年的月份只有 12 个月，所以各分库的分表数不能超过 12 张分表。

12.2.3.11 DD 算法

适用场景

DD 适用于按日期的天数进行分表，分表的表数就是日期的天数。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值的日期的天数进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库建确定分库后，确定分表的计算方式是：一个月的第几天mod分表数，即：15 mod 31 = 15。

算法计算方式

表 12-21 算法举例

条件	计算方式	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值：2019-1-15 分表：15 % 31 = 15

建表语法

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

注意事项

按 DD 进行分表，由于一个月的中日期（DATE_OF_MONTH）的取值范围是1~31，所以各分库的分表数不能超过 31 张分表。

12.2.3.12 MMDD 算法

适用场景

MMDD 适用于按一年的天数（即一年中日期）进行分表，分表的表名的下标就是一年中的第几天，一年最多 366 天。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值所对应的日期在一年中对应的天数，然后进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库键确定分库后，确定分表的计算方式是：一年的第几天mod分表数，即：15 mod 366 = 15; 2019-1-15是一年的第15天。

算法计算方式

表 12-22 算法举例

条件	计算方式	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值：2019-1-15 分表：15 % 366= 15

建表语法

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

注意事项

由于一年最多只有 366 天，当按 MMDD 进行分表时，所以各个分库的分表数目不能超过 366 张分表。

12.2.3.13 WEEK 算法

适用场景

WEEK 适用于按周数的日期目进行分表，分表的表名的下标分别对应一周中的各个日期（星期一到星期天）。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值所对应的一周之中的日期进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库建确定分库后，确定分表的计算方式是：一周的第几天mod分表数，即：3 mod 7 = 3; 2019-1-15是一周的第3天。

算法计算方式

表 12-23 算法举例

条件	计算方式	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值：2019-1-15 分表：3 % 7= 3

建表语法

```
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;
```

注意事项

由于一周共有 7 天，当按 WEEK 进行分表时，所以各分库的分表数不能超过 7 张。

12.3 DML

不支持的 DML 语法

表 12-24 DML 的语法限制

DML语法	限制条件
DELETE语句	不支持PARTITION子句。
UPDATE语句	• 不支持跨分片子查询。 • 不支持拆分键的更新。
SELECT语句	不支持类似ORDER BY FIELD(id,1,2,3)这种自定义排序。

DML语法	限制条件
系统库查询	支持以下系统库查询： SELECT version() • information_schema.SCHEMA_PRIVILEGES • information_schema.TABLE_PRIVILEGES • information_schema.USER_PRIVILEGES • information_schema.SCHEMATA • information_schema.tables • information_schema.columns SHOW KEYS FROM `table ` FROM `database **说明** • 仅支持=、in、like三种操作符，和and条件关联。 • 不支持子查询、关联查询、排序、聚合查询、LIMIT等等复杂查询。 • information_schema.tables和information_schema.columns支持<>操作符。

12.3.1 INSERT

INSERT是将数据插入到数据库对象中的指令。

常用语法

```
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] ... ]
```

语法限制

- 不支持INSERT DELAYED...；
- 不支持不包含拆分字段的INSERT；
- 暂不支持PARTITION 语法，建议不要使用partition表；

12.3.2 REPLACE

REPLACE用于往表中插入行或替换表中的行。

常用语法

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

语法限制

- 暂不支持PARTITION 语法；
- 当自增表格ID不存在时，使用REPLACE将会插入一条指定ID的数据，但不会自动生成ID。

12.3.3 DELETE

DELETE指令为用于删除表中符合条件的行。

常用语法

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

语法限制

- WHERE 条件中不支持子查询（相关子查询和非相关子查询）；
- 不支持在多表删除中删除广播表中的数据（目标表列表中不可包含广播表）。

12.3.4 UPDATE

常用语法

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

语法限制

- 不支持使用子查询（相关子查询和非相关子查询）；
- UPDATE语句中的where_condition不支持计算表达式及其子查询；
- 不支持在多表更新中修改广播表（广播表中的列不可出现在 SET 中赋值语句的左侧）；
- 不支持更新逻辑表的拆分键字段，更新拆分键字段可能导致数据重新分布，DDM暂不支持。

12.3.5 SELECT

SELECT通常用于查询一个或多个表中的数据。

常用语法

```
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}]
```


表 12-25 说明信息

语法	说明
select_expr	每个select_expr都指示一个您想要查询的列。
FROM table_references	指您将从哪个或哪些表中查询。
WHERE	关键词WHERE其后跟一个表达式，用于表示被选择的行所须满足的条件。
GROUP BY	语法中被使用的子句将按一定的顺序排列，GROUP BY表示语句间关系，支持列名。如一个HAVING子句必须位于GROUP BY子句之后，并在ORDER BY子句之前。
ORDER BY	语法顺序排列的一种方式，表示语句间关系，支持列名和指定的排序方式（如ASC、 DESC）。
LIMIT/OFFSET	对输出结果集的偏移量及大小给予约束，如：LIMIT接受一个或者两个数字参数。

语法说明

- 暂不支持以空字符串作为别名;
- 不支持select ... group by ... with rollup;
- 暂不支持 STRAIGHT_JOIN 和 NATURAL JOIN;
- select for update 仅支持简单查询，不支持 join、group by、order by、limit;
- 对于 UNION 中的每个 SELECT, DDM 暂不支持使用多个同名的列，如下:

  # 如下 SQL 的 SELECT 中存在重复的列名，暂不支持SELECT id, id, name FROM t1 UNION SELECT pk, pk, name FROM t2。

12.3.6 SELECT JOIN Syntax

常用语法

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)
```

语法限制

不支持SELECT STRAIGHT_JOIN 和 NATURAL JOIN。

示例

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

12.3.7 SELECT UNION Syntax

常用语法

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

示例

```
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;
```

语法限制

对于UNION中的每个SELECT，不支持使用多个同名的列。

12.3.8 SELECT Subquery Syntax

The Subquery as Scalar Operand

示例

```
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

语法

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

示例

```
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

语法

```
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)
```

示例

```
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)

语法

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

示例

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

语法限制

- Derived Tables 必须拥有一个别名。
- Derived Tables 不可以成为 Correlated Subqueries，即不能包含子查询外部表的引用。
- 标量子查询在一些场景下当前不能得到正确结果，建议改写为join，同时可提高性能。
- 不支持 HAVING 子句中的子查询，JOIN ON 条件中的子查询。
- 不支持Row Subqueries。

12.4 函数

支持的函数

表 12-26 操作符函数

函数表达式	示例
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

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

函数表达式	示例
IFNULL	SELECT IFNULL(product_id, 0) FROM Products; **说明** • 3月20日之前已经创建的实例，拆分表不支持IFNULL与聚合函数的嵌套函数调用，如：select IFNULL(sum(yan),0) from shenhai，结果会和预期不一样。 • 3月20日之后的实例，拆分表只支持IFNULL与聚合函数的一层嵌套函数调用。

表 12-27 时间日期函数

函数表达式	示例
DAY()	SELECT * FROM TAB_DT WHERE DAY(dt)=21 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')
MONTH()	SELECT * FROM TAB_DT WHERE MONTH(dt)=12 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')
YEAR()	SELECT * FROM TAB_DT WHERE YEAR(dt)=2018 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')
DAYOFYEAR()	SELECT * FROM TAB_DT WHERE DAYOFYEAR(dt)=365 SELECT * FROM TAB_DT WHERE dt='2018-12-31' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')
DAYOFWEEK()	SELECT * FROM TAB_DT WHERE DAYOFWEEK(dt)=6 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')
WEEKOFYEAR()	SELECT * FROM TAB_DT WHERE WEEKOFYEAR(dt)=51 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')

表 12-28 数学函数

函数表达式	示例
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

函数表达式	示例
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

不支持的函数

表 12-29 函数的限制

函数	限制条件
函数	暂不支持row_count()函数。

12.5 其他不支持语句

- 不支持触发器；
- 不支持临时表；
- 不支持DO语句；
- 不支持外键关联；
- 不支持RESET语句；
- 不支持FLUSH语句；
- 不支持BINLOG语句；
- 不支持HANDLER语句；
- 不支持show warnings；
- 不支持 ‘:=’ 赋值运算符；
- 暂不支持<=>运算符；
- 暂不支持'IS UNKNOWN'表达式；
- 不支持用户管理及权限管理语句；
- 不支持INSTALL/UNINSTALL PLUGIN语句；
- 不支持分布式级别的存储过程及自定义函数；
- 库名、表名不可修改，拆分字段的名称和类型都不可以变更；
- 不支持SHOW PROFILES、SHOW ERRORS等多数运维SHOW语句；
- 不支持表维护语句，包括ANALYZE/CHECK/CHECKSUM/OPTIMIZE/REPAIR TABLE；

- 不支持session变量赋值与查询，如set @rowid=0;select @rowid:=@rowid+1,id from user。
- 不支持SQL语句中包含单行注释 ' --' 或者多行（块）注释 ' /*...*/'。

权限级别支持情况：

- 全局层级（暂不支持）
- 数据库层级（支持）
- 表层级（支持）
- 列层级（暂不支持）
- 子程序层级（暂不支持）

12.6 实用 SQL 语句

12.6.1 CHECK TABLE

12.6.1.1 检查当前逻辑库下所有逻辑表各分表的 DDL 一致性

用途：用于对某逻辑库所有的逻辑表的一致性情况进行全局概览。

命令格式：

```
check table
```

命令输出：

如果全部逻辑表都一致, 输出结果为：

```
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)
```

如果存在不一致的逻辑表, 输出结果为：

```
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)
```

输出详解：

每一行表示一个逻辑表的检查结果概况。

- DATABASE_NAME：逻辑库名称。
- TABLE_NAME：逻辑表名称。
- TABLE_TYPE：逻辑表类型。

  - SINGLE：单表。
  - BROADCAST：广播表。
  - SHARDING：拆分表。
- DDL_CONSISTENCY：该逻辑表对应所有物理表DDL是否一致。
- TOTAL_COUNT：该逻辑表有几个物理表。
- INCONSISTENT_COUNT：该逻辑表有几个物理表的DDL不一致。
- DETAILS：DDL不一致的物理表名。

12.6.1.2 检查某一张逻辑表各分表的 DDL 一致性

用途：对特定一张逻辑表进行详细检查。

命令格式：

```
check table <table_name>
```

命令输出：

若返回结果集为空, 表示该逻辑表各物理分表DDL都是一致的：

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

若返回结果集不为空, 表示各个不一致的物理表：

```
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)
```

输出详解：

每一行表示一个不一致的物理拆分表的详细检查结果。

- DATABASE_NAME：物理表所在的物理分库。
- TABLE_NAME：物理表表的表名。
- TABLE_TYPE：物理表所属逻辑表类型。
- EXTRA_COLUMNS：该物理表多出来的列。
- MISSING_COLUMNS：表示该物理表缺少的列。
- DIFFERENT_COLUMNS: 表示该物理表属性不一致的列(包括名称, 类型)。
- KEY_DIFF：表示该物理表不一致的索引。
- ENGINE_DIFF：表示该物理表不一致的引擎。
- CHARSET_DIFF：表示该物理表不一致的字符集。
- COLLATE_DIFF：表示该物理表不一致的排序规则。
- EXTRA_PARTITIONS：(分区表专用) 表示该物理表多出来的分区。
- MISSING_PARTITIONS：(分区表专用) 表示该物理表缺少的分区。

- DIFFERENT_PARTITIONS：(分区表专用) 表示该物理表属性不一致的分区。
- EXTRA_INFO：其他信息, 如物理表缺失, 将在这里显示。

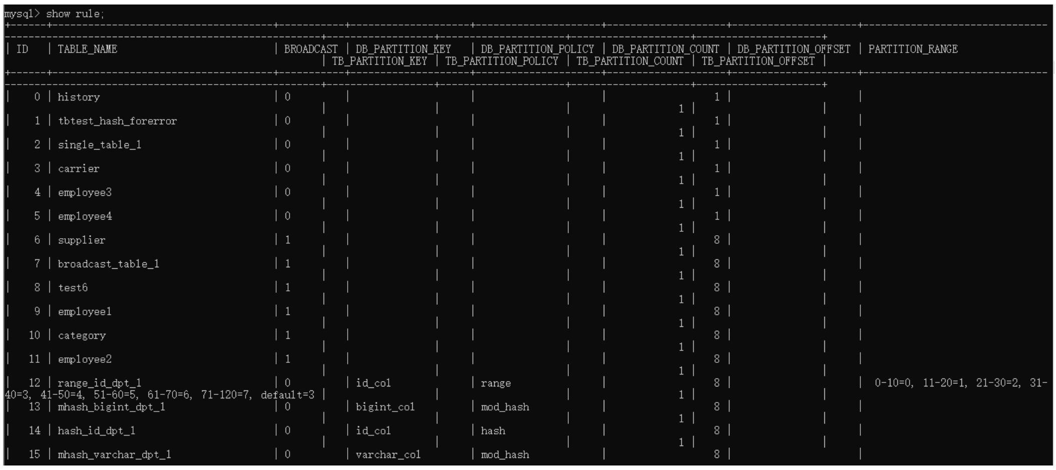
12.6.2 SHOW RULE

命令格式:

```
show rule：查看数据库下每一个逻辑表的拆分情况。
show rule from <table_name>：查看数据库下指定逻辑表的拆分情况。
```

命令输出:

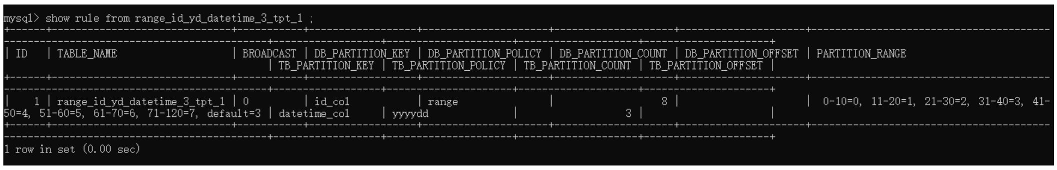
show rule 如下截图:



```
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			8					1	
7	broadcast_table_1	1			8					1	
8	test6	1			8					1	
9	employee1	1			8					1	
10	category	1			8					1	
11	employee2	1			8					1	
12	range_id_dpt_1	0	id_col	range	8		0-10=0, 11-20=1, 21-30=2, 31-40=3, 41-50=4, 51-60=5, 61-70=6, 71-120=7, default=3			1	
13	mhash_bigint_dpt_1	0	bigint_col	mod_hash	8					1	
14	hash_id_dpt_1	0	id_col	hash	8					1	
15	mhash_varchar_dpt_1	0	varchar_col	mod_hash	8						

show rule from *<table_name>* 如下截图:



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

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

```
1 row in set (0.00 sec)
```

输出详解:

TABLE_NAME ： 表名。

BROADCAST：是否为广播表（0：否，1：是）。

DB_PARTITION_KEY：分库的拆分键，没有分库的话，值为空。

DB_PARTITION_POLICY：分库的拆分策略，取值包括哈希或YYYYMM、YYYYDD、YYYYWEEK 等日期策略。

DB_PARTITION_COUNT：分库数。

DB_PARTITION_OFFSET ： 分库偏移量。

PARTITION_RANGE：分库拆分算法为range时的拆分范围设置。

TB_PARTITION_KEY：分表的拆分键，没有分表的话，值为空。

TB_PARTITION_POLICY：分表的拆分策略，取值包括哈希或MM、DD、MMDD、WEEK等日期策略。

TB_PARTITION_COUNT：分表数。

TB_PARTITION_OFFSET：分表偏移量。

12.6.3 SHOW TOPOLOGY

命令格式：

show topology from xxx<table_name>：查看数据库下指定逻辑表的物理分布情况。

输出详解：

Rds_instance_id：rds的实例id。

HOST： ip 。

PORT： 端口 。

DATABASE： 物理库 。

TABLE： 物理表 。

ROW_COUNT： 表的数据量(大致的值，在information_schema.TABLES取值)。

12.6.4 SHOW DATA NODE

命令格式：

show data node：查看物理分片的数据

输出详解：

Rds_instance_id：rds的实例id。

PHYSICAL_NODE：物理节点。

HOST：主机号。

PORT：端口号。

12.6.5 TRUNCATE TABLE

hint对于所有的单表以及全局表失效，只对各种分表起作用。

12.6.5.1 HINT-DB

命令格式：

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

描述：

删除对应的*<physical_db_name>*物理库下对应的所有的*<table_name>*的分表数据，其余分库的表不受影响。

12.6.5.2 HINT-TABLE

命令格式：

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

描述：

删除当前库下表名*<physical_table_name>*的所有物理表的数据，其余分表不受影响。

12.6.5.3 HINT-DB/TABLE

命令格式：

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

描述：

删除库名为<physical_db_name>下表名为<physical_table_name>的物理分表数据，其余分库下其他分表不受影响。

12.6.6 HINT- ALLOW_ALTER_RERUN

命令格式：

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

描述：

使用该hint可支持命令重复执行不报错，共支持八种alter table的命令形式： add column、modify column、drop column、add index、drop index、change column、add partition和drop partition。

12.6.7 LOAD DATA

标准示例

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

说明

如果数据中可能包含一些特殊字符，比如分割符转义符等，建议用引号扩起来，通过OPTIONALLY ENCLOSED BY '"'指定。

如果上述方法不起作用，可以把字段值中的引号替换成\"。

- 如果指定**LOCAL**关键词，则表明从客户端主机读文件。如果local没指定，出于安全考虑不支持此功能。
- 可以通过**FIELDS TERMINATED BY**指定字符之间的分割符号，默认值为\t 。
- 通过**OPTIONALLY ENCLOSED BY**忽略数据源字段中的符号。
- 通过**LINES TERMINATED BY**可以指定行之间的换行符，默认为\n。

  说明

  有些windows上的文本文件的换行符可能为\r\n，由于是不可见字符，所以请小心检查。
- 通过**CHARACTER SET**指定文件的编码，建议跟RDS for MySQL实例上物理库（分片）的编码一致，否则可能乱码。其中字符集编码必须用引号扩起来，否则会解析出错。
- 通过**IGNORE**或者**REPLACE**指定遇到重复记录是替换还是忽略。
- 目前列名必须指定，且必须包括分片字段，否则没办法确定路由。

- 其他参数参考MySQL的**load data infile官方文档**说明。其他参数的先后顺序不能乱，顺序参考**官方说明**。

须知

1. 数据导入阶段会在一定程度上影响DDM以及RDS for MySQL实例性能，请选择在业务空闲时间导入。
2. 建议不要同时发起多个LOAD DATA请求。多个LOAD DATA同时进行，数据高并发写入，表锁竞争以及系统IO抢占会影响总体效率，可能会出现SQL事务超时现象，导致LOAD DATA全部失败。
3. 由于分布式事务的特性，使用LOAD DATA导入数据时，需要设置手动提交事务，以确保数据记录改动的准确无误。

   例如客户端可进行如下设置：

   mysql> set autocommit=0;

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

   mysql> commit;

限制

不支持：

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

12.6.8 SHOW PHYSICAL PROCESSLIST

命令格式1：

show physical processlist：展示后端rds的processlist。

命令格式2：

show physical processlist with info：在命令1的结果集中过滤掉info列为空的数据，只展示info列不为空的结果。

命令执行效果如下：

图 12-1 命令执行效果

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

命令输出详解：

ip:后端rds的ip地址。

port:后端rds的端口号。

instance id：后端rds的实例id。

type:master是读写rds,readreplica是只读rds。

后面的列是对应的后端rds的processlist信息，与在后端rds直接执行show processlist获取的信息一致。

命令格式3:

kill physical physical_thread_id@rds_ip:rds_port：kill掉对应rds上的执行线程

12.6.9 自定义 Hint 读写分离

DDM提供HINT来指定SQL语句是在主实例上执行还是在只读实例上执行。

HINT的格式为：/*mycat:db_type=**host***/。

其中**host**可以是master或者slave，master代表主实例，slave代表只读实例。

目前只支持SELECT语句。

说明

通常情况下，实现了读写分离之后，主实例上只能进行写操作，只读实例上只进行读操作。但是在某些特殊情况，需要在主实例上读取数据时，可以用自定义Hint的方式指定在主实例上进行读操作。

12.6.10 自定义 Hint 跳过执行计划缓存

DDM 提供HINT来控制SELECT语句是否跳过缓存的执行计划。

HINT的格式为：/*!GAUSS:skip_plancache=**flag***/。

其中**flag**可以是true或者false，true代表跳过执行计划缓存，false代表不跳过。

目前只支持SELECT语句。

12.7 全局序列

全局序列主要指基于DB的全局序列。

说明

- 支持修改自增序列初始值。
- 全局序列主要保证ID全局唯一，并不能保证一定是连续递增的。

表 12-30 全局序列支持的表类型

表类型	拆分表	广播表	单表
基于DB的全局序列	支持	支持	不支持

创建自增序列

步骤1 使用客户端连接DDM实例。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入命令创建自增序列。

create sequence *xxxxx*;

说明

- xxxxx代表序列名。

----结束

删除自增序列

步骤1 使用客户端连接DDM实例。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入命令“show sequences”查看所有序列。

步骤4 输入命令删除序列。

drop sequence *xxxxx*;

drop sequence *DB.xxx*;

说明

- 对大小写不敏感。
- 如果序列属于某张表格（即创建这张表时有一列是自增列），不允许删除。

----结束

修改自增序列初始值

步骤1 使用客户端连接DDM实例。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入“show sequences”查看所有序列。

步骤4 输入命令修改序列起始值。

alter sequence *xxxxx* **START WITH** *yyyyy;*

说明

- xxxxx代表序列名。
- yyyyy代表目标序列起始值。

----结束

查询自增序列

步骤1 使用客户端连接DDM实例。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入命令“show sequences”查看所有序列。

show sequences;

```
mysql> show sequences;
+----------------+-----------+-----------+
| NAME           | START_WITH | INCREMENT |
+----------------+-----------+-----------+
| WORLD.WORLDYML | 1          | 1000      |
| WORLD.YML      | 1          | 1000      |
+----------------+-----------+-----------+
```

----结束

修改自增序列 Cache

须知

该功能仅支持内核3.0.3以上的版本。

步骤1 使用客户端连接DDM实例。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入命令“alter sequence test cache 5000”，修改表test的全局序列的cache值。

步骤4 输入命令“show sequences”，查看test序列的INCREMENT值即是cache值。

----结束

12.7.1 nextval、currval 在全局序列的使用

- nextval返回下一个序列值，currval返回当前序列值。其中nextval可以通过nextval(n)返回n个唯一序列值。
- nextval(n)只能单独用在select sequence.nextval(n)场景下并且不支持跨库操作。
- currval不支持currval(n)的用法。

操作步骤

步骤1 使用客户端连接DDM实例。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入命令创建全局序列。

create sequence seq_test;

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

步骤4 输入命令，返回下一个序列值。

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)
```

步骤5 输入命令，获取当前序列值。

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)
```


步骤6 输入命令，批量获取序列值。

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)
```

说明

- 批量获取序列值场景不支持跨库操作。
- 未使用过全局序列时，currval的返回值是0。

----结束

12.7.2 全局序列在 INSERT 或 REPLACE 语句中的使用

要想在同一个实例下实现跨逻辑库序列的全局唯一，可以在insert语句或者replace语句中结合全局序列一起使用。Insert语句和replace语句支持nextval和currval两个方式序列的获取。其中，nextval表示返回下一个序列值，currval表示返回当前序列值。

可以通过schema.seq.nextval、schema.seq.currval使用，如果不指定schema，默认是当前连接schema下的全局序列。

支持并发获取全局序列，在多session下并发通过schema.seq.nextval获取全局序列能够产生唯一值。

用法举例

两个逻辑库dml_test_1、dml_test_2，里面都有表test_seq。

表定义

表定义方法：create table test_seq(col1 bigint,col2 bigint) dbpartition by hash(col1)。

操作步骤

步骤1 使用客户端连接DDM实例。

步骤2 连接成功后，打开目标逻辑库。

步骤3 在库级别下，输入命令创建全局序列。

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)
```

步骤4 使用以下语句，实现全局序列在insert语句或者replace语句的使用。

- **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)
```

由于全局序列是创建在逻辑库dml_test_1下的，在逻辑库dml_test_2下使用全局序列需要显式指定逻辑库dml_test_1.seq_test.nextval、dml_test_1.seq_test.currval。

说明

- 全局序列结合insert和replace的使用只支持拆分表，不支持广播表和单表。
- nextval和currval在insert和replace语句中是从左到右执行的，如果一条语句使用同一个全局序列nextval多次，每出现一次就递增一次。
- 全局序列是属于逻辑库的，删除逻辑库，所在删除逻辑库的全局序列也会被删除。

----结束

12.8 数据库管理语法

支持的数据库管理语法

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

支持的数据库工具命令

- DESC Syntax
- USE Syntax
- EXPLAIN Syntax

  与MySQL内部的explain有所区别，DDM的explain显示的结果是当前语句路由到的节点描述。

不支持的数据库管理语法

表 12-31 数据库管理语句的限制

数据库管理语句	限制条件
数据库管理语句	• 不支持SET Syntax修改全局变量。 • 不支持SHOW TRIGGERS语法。 下列的SHOW指令会随机发到某个物理分片，每个物理分片如果在不同的RDS for MySQL实例上，查得的变量或者表信息可能不同： • SHOW TABLE STATUS • SHOW VARIABLES Syntax • check table不支持hash和key分区表 • SHOW WARNINGS Syntax 不支持 LIMIT/COUNT 的组合 • SHOW ERRORS Syntax 不支持 LIMIT/COUNT 的组合

12.9 SQL 高级功能

表 12-32 SQL 高级功能的限制

SQL高级功能	限制条件
SQL高级功能	● 暂不支持Prepare\EXECUTE语法。 ● 暂不支持用户自定义数据类型、自定义函数。 ● 暂不支持视图、存储过程、触发器、游标。 ● 暂不支持 BEGIN…END、LOOP…END LOOP、REPEAT… UNTIL…END REPEAT、WHILE…DO…END WHILE 等复合语句。 ● 暂不支类似 IF ，WHILE 等流程控制类语句。 ● 暂不支持的预处理类型： **PREPARE** Syntax **EXECUTE** Syntax ● 不支持在建表语句中，对索引增加COMMENT形式的注释。 ● 不支持进行用户权限相关的设置。 例如grant all on *.* to 'test'@'%' identified by 'test123'等用户权限设置均不支持，请在前台console设置。

13 常见问题

13.1 DDM 通用类

13.1.1 DDM 提供哪些高可靠保障

数据完整性

DDM实例故障不会影响数据的完整性。

- 业务数据存储在RDS for MySQL实例上分片中，DDM不存储业务数据。
- 逻辑库与逻辑表等配置信息存储在DDM数据库中，DDM数据库主备高可用。

容错机制

DDM采用的是RDS for MySQL自身的容错机制，数据是否写入DB由RDS for MySQL保证。RDS for MySQL实例返回sql执行成功，DDM就认为成功。

13.1.2 DDM 自身是否会存储业务数据

DDM实例自身不存储客户业务相关数据，客户业务相关数据都存储在RDS for MySQL实例上分片中。

DDM节点目前硬盘主要用来存储日志和一些临时文件，日志和临时文件会做定期清理，空间足够使用。

13.1.3 如何选择和配置安全组

DDM实例采用了VPC和安全组等网络安全保护措施，以下内容帮助您正确配置安全组。

通过 VPC 内网访问 DDM 实例

DDM实例的访问和使用，包括客户端所在ECS访问DDM实例，以及DDM实例访问其关联的RDS for MySQL实例。

除了ECS、DDM实例、RDS for MySQL实例必须处于相同VPC之外，还需要他们的安全组分别配置了正确的规则，允许网络访问。

1. 建议ECS、DDM、RDS配置相同的安全组。安全组创建后，默认包含同一安全组内网络访问不受限制的规则。
2. 如果配置了不同安全组，可参考如下配置方式：

说明

- 假设ECS、DDM、RDS分别配置了安全组：sg-ECS、sg-DDM、sg-RDS。
- 假设DDM实例服务端口为5066，RDS for MySQL实例服务端口为3306。
- 以下规则，远端可使用安全组，也可以使用具体的IP地址。

ECS所在安全组需要增加图13-1中的规则，以保证客户端能正常访问DDM实例：

图 13-1 ECS 安全组策略

DDM所在安全组需要增加图13-2和图13-3中的规则，以保证能访问RDS for MySQL实例，且被客户端访问。

图 13-2 DDM 安全组入方向配置

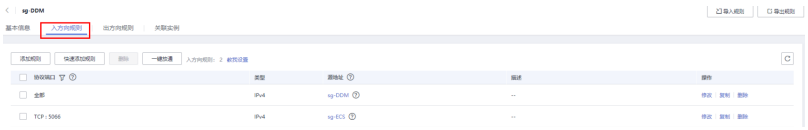
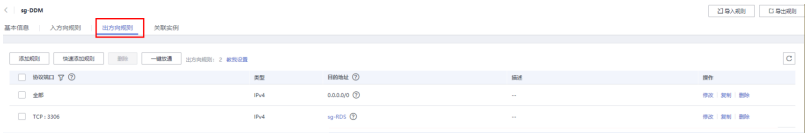
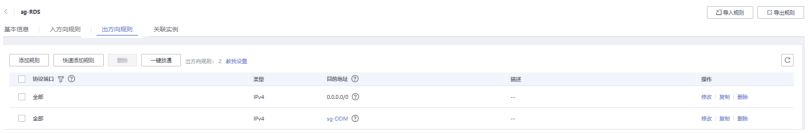


图 13-3 DDM 安全组出方向配置



RDS for MySQL实例所在安全组需要增加图13-4中的规则，以保证能被DDM访问。

图 13-4 RDS 安全组配置



13.2 DDM 使用类

13.2.1 如何解决 JDBC 驱动方式连接 DDM 异常问题

MySQL驱动（ JDBC ）通过Loadbalance方式连接DDM，在某些场景下连接切换时会陷入死循环，最终导致栈溢出。

问题定位

1. 查看APP日志，定位异常原因。

   例如，从以下日志中分析出异常最终原因为栈溢出。

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

2. 分析溢出源。

   例如，从以下日志可以分析出，溢出原因为驱动内部陷入死循环。

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

3. 查看使用的MySQL版本，为5.1.44。

   查看该版本源代码，发现获取连接时，**LoadBalance**会根据负载均衡策略更新连接，并将老连接的配置复制给新连接，在新连接**AutoCommit**为**true**，新连接部分参数和老连接不一致，**loadBalanceAutoCommitStatementThreshold**参数没有配置的场景下，会陷入死循环，更新连接函数调用同步参数函数，同步参数又调用更新连接，最终导致栈溢出。

解决方法

在连接DDM的URL添加**loadBalanceAutoCommitStatementThreshold=5&retriesAllDown=10**参数。

```
//使用负载均衡的连接示例
//jdbc:mysql:loadbalance://ip1:port1,ip2:port2..ipN:portN/{db_name}
String url = "jdbc:mysql:loadbalance://192.168.0.200:5066,192.168.0.201:5066/db_5133?
loadBalanceAutoCommitStatementThreshold=5&retriesAllDown=10";
```

- loadBalanceAutoCommitStatementThreshold：表示连接上执行多少个语句后会重新选择连接。

  假设loadBalanceAutoCommitStatementThreshold设为5，则当执行5个sql后（ Queries或者updates等 ），将会重新选择连接。若为0表示“粘性连接，不重新

选择连接”。关闭自动提交时（autocommit=false）会等待事务完成再考虑是否重新选择连接。

13.2.2 如何选择 JDBC 驱动方式的版本和参数

DDM暂不支持使用5.1.46版本的JDBC驱动连接DDM，建议您使用以下版本的JDBC驱动：5.1.35-5.1.45。

JDBC驱动下载地址：**https://dev.mysql.com/doc/index-connectors.html**。

JDBC URL中推荐参数如**表13-1**所示。

表 13-1 参数

参数名称	参数说明	推荐取值
ip:port	连接地址和端口，用于连接DDM。	在DDM管理控制台DDM实例管理中查看连接地址。
db_name	连接逻辑库名称。	在DDM管理控制台，DDM实例管理 > 逻辑库管理下查看逻辑库名称。
loadBalanceAutoCommitStatementThreshold	表示连接上执行多少个语句后会重新选择连接。 • 若取值为5，则当执行5个sql后（Queries或者updates等），将会重新选择连接。 • 若取值为0，则表示“粘性连接，不重新选择连接”。 关闭自动提交时（autocommit=false）会等待事务完成再考虑是否重新选择连接。	5
loadBalanceHostRemovalGracePeriod	设置主机从负载均衡连接中移除的宽限时间。	15000
loadBalanceBlacklistTimeout	设置服务器在全局黑名单中存留的时间。	60000
loadBalancePingTimeout	使用负载均衡连接时，等待每个负载均衡连接ping响应的毫秒数。	5000
retriesAllDown	当所有的连接地址都无法连接时，轮询重试的最大次数。 重试次数达到阈值仍然无法获取有效连接，将会抛出SQLException。	10

参数名称	参数说明	推荐取值
connectTimeout	和数据库服务器建立socket连接时的超时。 单位：毫秒，0表示永不超时，适用于JDK 1.4及更高版本。	10000
socketTimeout	socket操作（读写）超时。 单位：毫秒， 0表示永不超时	根据业务实际情况合理配置。

13.2.3 使用 mysqldump 从 MySQL 导出数据非常缓慢的原因

mysqldump客户端的版本和DDM所支持的MySQL版本不一致，可能会导致从MySQL导出数据非常缓慢。

建版本保持一致。

13.2.4 导入数据到 DDM 后出现主键重复

创表时设置自增起始值，并确保起始值大于导入数据自增键的最大值。

13.2.5 如何处理数据迁移过程中自增列后报错：主键重复

重新设置自增主键的初始值为大于当前已有数据的最大值，执行如下语句：ALTER SEQUENCE 库名.SEQ名 START WITH 新初始值;

13.2.6 如何处理配置参数未超时却报错

建议可将参数SocketTimeOut值调整或者去掉，默认为0则不断开连接。

13.2.7 如何处理 DDM 逻辑库与 RDS 实例的先后关系

不支持先删除逻辑库关联的RDS，再删除逻辑库。因为先删除关联的RDS，再删除逻辑库会失败，需要通过删除DDM实例才能删除逻辑库。

13.2.8 DDM 逻辑库删除后，RDS 实例里面残留着部分预留的 DDM 数据库和一些 DDM 的账户，这些是否需要手动删除

如果不需要了，可以直接手动删除，释放空间。

13.3 SQL 语法类

13.3.1 DDM 是否支持 SQL 跨库访问

不支持SQL通过带数据库名称的方式跨逻辑库访问。

DDM会自动去除SQL中的库名，比如select * from dn1.item,会自动改成select * from item。

13.3.2 DDM 是否支持分布式 JOIN

DDM支持分布式JOIN。

- 表设计时，增加字段冗余
- 支持跨分片的JOIN，主要实现的方式有三种：广播表，ER分片和ShareJoin。
- DDM目前禁止多个表的跨库update和delete。

13.3.3 如何进行 SQL 优化

- 尽量避免使用LEFT JOIN或RIGHT JOIN，建议使用INNER。
- 在使用LEFT或RIGHT JOIN时，ON会优先执行，WHERE条件在最后执行，所以在使用过程中，条件尽可能在ON语句中判断，减少WHERE的执行。
- 尽量少用子查询，改用JOIN，避免大表全表扫描。

13.3.4 DDM 是否支持数据类型强制转换

数据类型转换属于高级用法，DDM对SQL的兼容性会逐步完善，如有需要请提工单处理。

13.3.5 如何处理 INSERT 语句批量插入多条数据时报错

解决方案

建议拆分为多条INSERT语句插入，或在控制台配置参数管理页面，修改max_allowed_packet参数值，并重启实例使之生效。

13.4 RDS 相关类

13.4.1 数据库表名是否区分大小写

DDM和RDS的MySQL版本默认对数据表名和序列名称不区分大小写。

13.4.2 RDS 哪些高危操作会影响 DDM

RDS for MySQL相关高危操作如表13-2所示。

表 13-2 RDS for MySQL 高危操作

操作类别	操作	操作影响
RDS控制台操作类	删除RDS实例	RDS for MySQL实例删除后，DDM关联该RDS for MySQL实例的逻辑库、逻辑表都无法使用。

操作类别	操作	操作影响
	切换RDS for MySQL主备实例	切换主备实例可能造成短时间内的RDS服务闪断，并有可能在主备同步时延过大的情况下，导致少量数据丢失。 ● RDS for MySQL实例主备切换过程中，DDM将无法进行创建逻辑库、创建表、平滑扩容等操作。 ● RDS for MySQL实例主备切换后，DDM中RDS for MySQL实例ID不变。
	重启实例	重启过程中，RDS for MySQL实例将不可用，DDM业务将会受影响。
	重置密码	RDS重置密码后，DDM这边创建逻辑库时输入重置后的密码即可。
	修改参数模板	其中如下参数为固定值，如果修改，将会影响DDM正常运行。 ● 数据表名和序列名称不区分大小写，“lower_case_table_names”固定为“1”。 ● 扩容场景，必须将“local_infile”配置为“ON”。
	修改安全组	将导致DDM服务无法连接RDS for MySQL实例。
	修改VPC	DDM实例与RDS for MySQL实例不在同一VPC中将导致无法互通。
	恢复	恢复数据可能会破坏数据完整性。
RDS for MySQL客户端类	删除DDM创建的物理库	删除物理库后，原数据将会丢失，新数据将无法写入。
	删除DDM创建的物理账号	删除物理账号后将无法在DDM上创建逻辑表。
	删除DDM创建的物理表	删除物理表后，将导致DDM数据丢失，DDM后续无法正常使用该逻辑表。
	修改DDM创建的物理表名	将导致DDM无法获取该逻辑表的数据，且后续无法正常使用。
	修改记录	如修改全局表记录，将会影响各分片数据一致性。
	修改白名单	需要确保DDM服务在RDS for MySQL实例的白名单内，否则DDM服务将无法访问RDS for MySQL实例。

13.4.3 如何处理表中存在主键重复的数据

场景一

DDM实例的逻辑表中已存在主键数据类型边界值的记录，如果插入的数据超过主键数据类型的范围，表中会出现主键重复的数据。

场景一处理方法

步骤1 登录云服务管理控制台。

步骤2 在RDS的“实例管理”页面，查找DDM实例对应的RDS for MySQL实例，单击目标RDS实例名称，进入实例的“基本信息”页面。

步骤3 在基本信息页面的左侧导航栏中选择“参数修改”。

步骤4 在“参数”页签搜索“sql_mode”，单击“值”列中的下拉框，勾选“STRICT_ALL_TABLES”或“STRICT_TRANS_TABLES”方式，单击“保存”。

说明

“STRICT_ALL_TABLES”和“STRICT_TRANS_TABLES”方式属于严格模式。严格模式控制MySQL如何处理非法或丢失的输入值。

- 非法：数据类型错误或超出范围。
- 丢失：如果某列定义为非空列且没有DEFAULT值，当新插入的行不包含该列时，该行记录丢失。
- 在进行扩容时，若DDM的实例版本低于2.4.1.3。在选择MySQL实例的参数sql_mode时，请不要选择ANSI_QUOTES。不能使用双引号来引用文字字符串，因为它们被解释为标识符。

  例如：select * from test where tb = "logic"。

关于“sql_mode”更多信息，请参考**Server SQL Modes**。

步骤5 在“DDM实例管理”页面，重启DDM实例。

----结束

场景二

DDM实例的拆分表（hash\range\mod）联合主键，如果拆分键数据中有插入0和1的话，一定会出现重复主键。

场景二处理方法

步骤1 登录云服务管理控制台。

步骤2 在RDS的“实例管理”页面，查找DDM实例对应的RDS for MySQL实例，单击目标RDS实例名称，进入实例的“基本信息”页面。

步骤3 在基本信息页面的左侧导航栏中选择“参数修改”。

步骤4 在“参数”页签搜索“sql_mode”，单击“值”列中的下拉框，勾选“NO_AUTO_VALUE_ON_ZERO”方式，单击“保存”。

步骤5 在“DDM实例管理”页面，重启DDM实例。

----结束

13.4.4 如何通过 show full innodb status 指令查询 RDS 相关信息

通过MySQL客户端连接DDM实例后，可直接输入show full innodb status指令查询该DDM实例所关联的RDS for MySQL实例信息。可查询信息如：

- 当前的时间及自上次输出以来经过的时长。
- 可以使用命令show full innodb status来查看master thread的状态信息。
- 如果有高并发的工作负载，您需关注SEMAPHORES信号量，它包含了两种数据：事件计数器以及可选的当前等待线程的列表，如果有性能上的瓶颈，可使用这些信息来找出瓶颈。

13.5 连接管理类

13.5.1 本地环境是否可以连接 DDM 实例

可以连接，首先绑定弹性公网IP到DDM实例，就可以在本地环境使用连接工具（例如：Navicat）通过EIP访问DDM实例。

Navicat 客户端连接 DDM 实例

步骤1 登录分布式数据库中间件服务，单击需要连接的DDM实例名称，进入实例基本信息页面。

步骤2 在“实例信息”模块的弹性公网IP单击“绑定”。绑定已申请分配的公网IP。

步骤3 在DDM管理控制台左侧选择虚拟私有云图标。单击“访问控制>安全组”

步骤4 在安全组界面，单击操作列的“配置规则”，进入安全组详情界面。在安全组详情界面，单击“添加规则”，弹出添加规则窗口。根据界面提示配置安全组规则，设置完成后单击“确定”即可。

说明

绑定弹性公网IP后，建议您在内网安全组中设置严格的出入规则，以加强数据库安全性。

步骤5 打开Navicat客户端，单击“连接”。在新建连接窗口中填写主机IP地址（弹性公网IP地址）、用户名和密码（DDM账号、密码）。

步骤6 单击“连接测试”，若显示连接成功，单击“确定”，等待1-2分钟即可连接成功。连接失败会直接弹出失败原因，请修改后重试。

----结束

说明

通过其他可视化的MySQL工具（例如 Workbench）连接DDM实例的操作与此章基本一致，不做详细描述。

13.5.2 MySQL 连接 DDM 时出现乱码如何解决

MySQL连接的编码和实际的编码不一致，可能导致DDM解析时出现乱码。

通过“default-character-set=utf8”指定客户端连接的编码即可。

如下所示：


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

