分布式数据库中间件

用户指南

文档版本	01
发布日期	2024-07-30

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://e.huawei.com

安全声明

产品生命周期政策

华为公司对产品生命周期的规定以“产品生命周期终止政策”为准，该政策的详细内容请参见如下网址：
https://support.huawei.com/ecolumnsweb/zh/warranty-policy

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

华为初始证书权责说明

华为公司对随设备出厂的初始数字证书，发布了“华为设备初始数字证书权责说明”，该说明的详细内容请参见如下网址：
https://support.huawei.com/enterprise/zh/bulletins-service/ENEWS2000015766

华为企业业务最终用户许可协议(EULA)

本最终用户许可协议是最终用户（个人、公司或其他任何实体）与华为公司就华为软件的使用所缔结的协议。最终用户对华为软件的使用受本协议约束，该协议的详细内容请参见如下网址：
https://e.huawei.com/cn/about/eula

产品资料生命周期策略

华为公司针对随产品版本发布的售后客户资料（产品资料），发布了“产品资料生命周期策略”，该策略的详细内容请参见如下网址：
https://support.huawei.com/enterprise/zh/bulletins-website/ENEWS2000017760

目 录

1 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中间件。兼容MySQL协议，专注于解决数据库分布式扩展问题，突破传统数据库的容量和性能瓶颈，实现海量数据高并发访问。

DDM支持的功能如**表1-1**所示。

表 1-1 DDM 服务功能列表

功能分类	功能描述
权限管理	包括创建用户并授权使用DDM和DDM服务的自定义策略。具体使用方法请参考**权限管理**。
实例管理	包括实例创建、实例规格变更、实例删除、实例重启等功能。具体使用方法请参考**实例管理**。
备份管理	包括数据的恢复到新实例、Metadata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**。
分片变更	可以通过增加分片或者数据节点进行存储层分片变更，具体使用方法请参考**分片变更**。
DN管理	该功能管理DDM实例关联的RDS for MySQL实例，可以快速执行读权重、同步DN信息、开启读写分离等操作，具体使用方法请参考**DN管理**。
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**。
数据迁移管理	包括从华为云迁移到DDM、其他云迁移到DDM和将数据从DDM导出等功能。具体使用方法请参考**数据迁移**。

功能分类	功能描述
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**。
标签管理	该功能用于云平台，通过统一的标签管理各种资源，具体请参考**标签**。
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**。

2 内核版本说明

本章节介绍内核版本更新说明。

表 2-1 内核版本更新说明

版本	说明
3.1.2.0	新功能 ● 连接池增强异常处理，增加DN节点的fail-fast特性（默认关闭）。 ● 强化分片变更数据校验性能。 ● 优化内存设置。 ● 支持ANALYZE TABLE语句。 ● 全局二级索引重建（白名单特性）。
3.1.1	新功能 ● 支持全局二级索引（白名单特性）。 ● 支持SSL加密连接。
3.1.0	新功能 ● 新增Show Processlist和Kill Sessionid支持增加过滤条件。 ● 新增支持MySQL 8.0 Online DDL相关关键字。 修复问题 ● 修复执行使用DATE_SUB函数进行update语句后出现毫秒值丢失。 ● 优化部分复杂查询下的字段别名显示。 ● 修复Rename操作混在其他ddl语句后面操作成功后，新表无法使用。 ● 修复Sequence并发插入报错问题。

版本	说明
3.0.9	新功能 ● 支持对Sequence自增进度查询能力。 ● 支持单条大小超过16M的记录查询。 ● 支持MariaDB Connector/J驱动。 ● 优化读写分离中的事务拆分功能，事务中连接未发生事务更新时可根据读写分离权重决定路由到主实例或只读实例。
3.0.8	新增功能 ● 新增支持ROW表达式的路由计算。 ● 新增支持表级回收站能力。 ● 新增支持METADATA备份恢复能力。 修复问题 ● 优化DDM MetaDb链接池。 ● 优化DDM进程监控机制。 ● 优化Show Processlist显示。 ● 优化Reload获取元数据的流程。 ● 优化增量回放如果遇到XA ROLLBACK时报错流程。
3.0.6	新增功能 ● 支持分片变更内核讲METADATA链接统一使用链接池。 修复问题 ● 修复对执行中的DDL命令做Ctrl+C操作, 无法记录结束状态问题。 ● 优化客户端大数据查询下，进程Kill场景下CPU使用率高。
3.0.4	修复问题 ● 修复部分异常场景下的读写分离报错问题。 ● 优化group_concat函数在数据量下的执行。

3 权限管理

3.1 创建用户并授权使用 DDM

如果云服务平台账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用DDM服务的其它功能。

如果您需要对所拥有的DDM进行精细的权限管理，您可以参考**统一身份认证服务**（Identity and Access Management，简称IAM）。

通过IAM，您可以：

- 根据企业的业务组织，在您的账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用DDM资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将DDM资源委托给更专业、高效的其他账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

本章节为您介绍对用户授权的方法。

前提条件

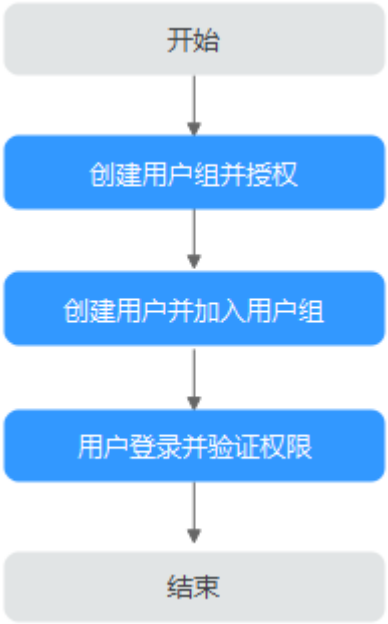
给用户组授权之前，请您了解用户组可以添加的DDM权限，并结合实际需求进行选择。

DDM支持的系统权限，请参见**权限策略**。

如果您需要对除了DDM之外的服务授权，请参见**权限策略**。

示例流程

图 3-1 授权 DDM 权限流程



1. **创建用户组并授权**

   在IAM控制台创建用户组，并授予分布式数据库中间件权限“DDM ReadOnlyAccess”。

2. **创建用户并加入用户组**

   在IAM控制台创建用户，并将其加入**1**中创建的用户组。

3. **用户登录**并验证权限

   新创建的用户登录控制台，切换至授权区域，验证权限：

   - 在“服务列表”中选择分布式数据库中间件服务，进入DDM主界面，单击右上角“购买数据库中间件实例”，尝试购买数据库中间件实例，如果无法购买数据库中间件实例（假设当前权限仅包含DDM ReadOnlyAccess），表示“DDM ReadOnlyAccess”已生效。
   - 在“服务列表”中选择除分布式数据库中间件服务外（假设当前策略仅包含DDM ReadOnlyAccess）的任一服务，如果提示权限不足，表示“DDM ReadOnlyAccess”已生效。

3.2 DDM 自定义策略

如果系统预置的DDM权限，不满足您的授权要求，可以创建自定义策略。

目前云服务平台支持以下两种方式创建自定义策略：

- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。
- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

具体创建步骤请参见：**创建自定义策略**。

本章为您介绍常用的DDM自定义策略样例。

策略样例

- **示例：拒绝用户删除DDM实例**

  拒绝策略需要同时配合其他策略使用，否则没有实际作用。用户被授予的策略中，一个授权项的作用如果同时存在Allow和Deny，则遵循Deny优先。拒绝策略示例如下：

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

  Allow和Deny同时存在的拒绝策略示例如下：

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

3.3 数据库账号权限说明

创建逻辑库、导入逻辑库、分片变更操作时，数据库账号需具备以下权限，建议您创建具备以下权限的账号或直接使用管理员账号进行相关操作。

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

4 实例管理

4.1 实例状态

实例状态是DDM实例的运行情况。用户可以使用管理控制台查看DDM实例状态。

表 4-1 实例状态

分类	状态	说明
正常	运行中	DDM实例正常和可用。
异常	创建失败	DDM实例创建失败。
	异常	DDM实例不可用。
	部分节点异常	DDM实例部分节点不可用。
	备份失败	备份实例失败。
	冻结	账户余额小于或等于0美元，系统对该用户下的实例进行冻结。
动作执行中	创建中	正在创建DDM实例。
	备份中	正在备份实例。
	恢复中	正在恢复备份到实例中。
	升级中	正在进行实例内核版本升级。
	回滚中	正在进行实例内核版本回滚。
	切换SSL中	正在进行实例SSL切换。
	端口修改中	正在修改DDM实例的服务端口。
	删除中	正在删除DDM实例。
	重启中	正在重启DDM实例。

分类	状态	说明
	节点扩容中	正在扩容该实例下的节点个数。
	节点缩容中	正在缩容该实例下的节点个数。
	规格变更中	正在变更实例的CPU和内存规格。
	转包周期中	按需付费实例正在转为包周期实例。
	正在创建组	正在为节点创建分组。
	正在删除组	正在删除节点分组。

4.2 购买实例

本章节主要介绍在DDM控制台上创建DDM实例的方法。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理页面，单击页面右上方的“购买数据库中间件实例”。

步骤3 在购买实例页面，设置实例相关信息。

表 4-2 参数说明

参数	说明
计费模式	支持“包年/包月”和“按需付费”两种模式，购买后同时支持计费模式互转。 ● 包年/包月：用户选购完服务配置后，可以根据需要设置购买时长，系统会一次性按照购买价格对账户余额进行扣费。 ● 按需计费：用户选购完服务配置后，无需设置购买时长，系统会根据消费时长对账户余额进行扣费。
区域	DDM实例所在区域，可根据需要直接切换区域。
项目	DDM实例所在的项目。
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、下划线（_）和中划线（-）。
时区	由于世界各国家与地区经度不同，地方时也有所不同，因此会划分为不同的时区。

参数	说明
节点个数	实例所含有节点个数，最多支持32个节点数。 **说明** ● 单节点存在高可用风险，建议至少创建2节点。 ● DDM实例单个节点的磁盘使用情况：系统盘40G + 数据盘100G。
可用区	可选择的可用区。 为了避免单个物理机故障影响多台云主机的情况发生，相同可用区内，DDM支持不同虚拟机反亲和性部署，即集群里不同虚拟机部署在不同物理主机上。 DDM支持跨可用区部署，以增强DDM实例高可用性，达到跨可用区的容灾。 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。
节点规格	DDM实例的规格，支持“通用增强型”和“鲲鹏通用计算增强型”。 **说明** 为了使所购买的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。
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 for MySQL实例保持一致。 ● DDM实例必须与应用程序、RDS for MySQL实例处于相同的VPC，以保证网络连通。 ● 目前DDM实例创建后不支持切换虚拟私有云，请谨慎选择。
子网	通过子网提供与其他网络隔离的、可以独享的网络资源，来提高网络安全性。创建DDM实例时会自动为您配置内网地址，实例创建成功后该内网地址可修改。
内网安全组	已创建的内网安全组。 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。
企业项目	企业项目管理提供了一种按企业项目管理云资源的方式，帮助您实现以企业项目为基本单元的资源及人员的统一管理。
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。

参数	说明
标签	可选配置。使用标签可以方便识别和管理您拥有的分布式数据库中间件服务资源。 您可以为DDM实例添加标签，每个DDM实例最多支持添加10个标签。 ● 新建标签 您可以在DDM控制台新建标签。新建标签时，需要设置相应的标签“键”和“值”。 **键：该项为必选参数，不能为空。** – 对于每个实例，每个标签的键唯一。 – 长度为1~36个字符。 – 不能为空字符串，不能以“_sys_”开头和以空格开头、结尾。 – 不能包含下列字符： 非打印字符ASCII(0-31)，“*”，“<”，“>”，“\”，“,”，“|”，“/”。 **值：该项为必选参数。** – 可以不填值，此时默认为空字符串。 – 长度为0~43个字符。 – 不能包含下列字符： 非打印字符ASCII(0-31)，“*”，“<”，“>”，“\”，“,”，“|”。 ● 添加预定义标签 预定义标签可以实现通过同一个标签来标识多种云资源。 您需要先在标签管理服务创建预定义标签，为云资源添加标签时，在标签输入框的下拉列表中可直接选择已创建的预定义标签，无需输入标签的“键”与“值”。 例如，已创建预定义标签，其键为“Usage”，值为“Project1”，后续为DDM设置“键”与“值”时，页面会出现已创建的预定义标签。 实例创建成功后，您可以单击实例名称，在“标签”页签下查看对应标签，也可以修改或删除标签。同时，您还可以通过标签快速筛选指定实例。 如果您在申请实例时未添加标签，可以待实例创建成功后，再为实例添加标签。
购买时长	购买DDM实例的时长，该参数仅当“计费模式”为“包年/包月”时才显示。 您可选择1个月、2个月、3个月、4个月、5个月、6个月、7个月、8个月、9个月和1年。 勾选“自动续费”后，实例自动续费周期与原订单周期一致。

步骤4 实例信息设置完成后，单击页面下方“立即购买”。

步骤5 确认配置信息，根据所选实例的计费模式进行后续操作。

- 选择“按需计费”模式，单击“提交”。
- 选择“包年/包月”模式，单击“去支付”。

步骤6 实例创建成功后，用户可以在“实例管理”页面对其进行查看和管理。

DDM服务端口默认为5066，实例创建成功后可修改。

具体请参见**修改DDM服务端口**。

----结束

4.3 只读业务隔离

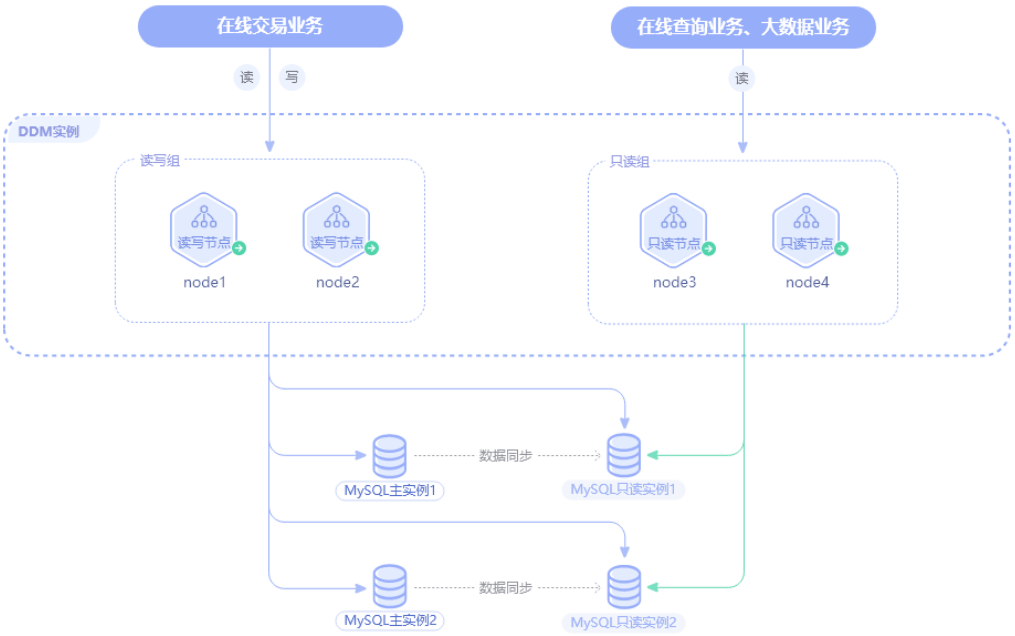
4.3.1 什么是只读业务隔离

只读业务隔离是DDM服务通过“组”的形式，实现对计算资源和存储层资源进行物理隔离的一种扩展能力，避免在线读写业务与只读业务互相干扰。

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，只读组默认将读流量下发到存储层只读实例上，缓解DDM集群主业务读负载压力。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，只读组可直接在数据节点的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。

原理示意图

图 4-1 原理示意图



4.3.2 如何进行只读业务隔离

本章节主要介绍进行只读业务隔离的步骤。

使用须知

- 如需使用，内核版本请升级至2.4.1.2及以上版本。
- 通过只读组进行SQL查询时，请先确保当前关联的数据节点已经挂载了只读实例且正常运行。以下报错情况可能是因为没有挂载只读实例或者只读实例运行异常导致的：
  - backend database connection error；
  - query has been canceled
  - execute error: No read-only node

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表中单击“实例名称”，默认进入“基本信息页”。

步骤3 在“节点信息”模块，单击“创建组”。

步骤4 在创建组页面，设置组信息。

图 4-2 创建组

表 4-3 参数说明

参数名称	说明
组名	必须以字母开头，可以包含字母、数字、中划线，不能包含其他的特殊字符。
组角色	组分为读写组和只读组，是DDM实例下承担了读写或者只读角色的节点的分组，用于计算资源隔离，每个组都配置对应的连接地址。
虚拟私有云	虚拟私有云默认和实例所在的虚拟私有云相同，且不能修改。 但是您可以设置组所在的子网。
节点规格	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需要保持一致。
可用区	选择可用区。

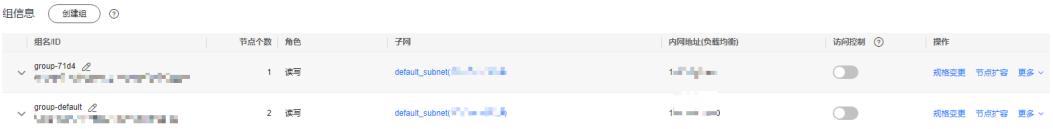
参数名称	说明
节点个数	一个DDM实例下支持创建多个只读组，实例的总节点数不超过32个，建议每组至少2个节点。您可以单击“添加”按钮来新增节点个数。

步骤5 单击“下一步”。

步骤6 确认组信息无误后，单击“提交”。

步骤7 组创建完成，节点信息变成组信息，在组信息模块对组进行管理。

图 4-3 组信息



说明

- 创建组之后，节点的管理、规格变更、节点扩/缩容、访问控制等操作会调整到组列表处进行管理。
- 创建组之后，已有的节点会被分成一个默认的读写组，主要用于主业务的读写。
- 创建只读组后，通过只读组的读流量会默认访问DN的只读实例，由于只读实例的数据是从DN主实例异步复制而来，可能存在可见性延迟，如果延迟超过阈值会出现访问报错。
- 包周期类型的DDM实例，创建组后暂不支持删除。
- 按需类型的DDM实例如果需删除组，单击“操作”列的“更多 > 删除组”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

4.4 变更规格

CPU/内存规格可根据业务需要进行变更，本章节主要介绍变更规格的操作。

使用须知

- 节点规格变更期间服务会短暂中断，建议在业务低峰时变更。
- 如果开启了只读业务隔离特性，即创建了只读组，规格变更功能入口将移动到组列表的操作列。
- 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。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 单击“规格变更”。如果您已经完成创建组操作，请在“组信息”模块中单击“操作”列的“规格变更”。

步骤4 在变更规格页面，选择实例规格。

步骤5 单击“下一步”。

步骤6 确认变更信息，根据所选实例的计费模式进行后续操作。

- “按需计费”模式，单击“提交”。
- “包年/包月”模式，单击“去支付”。

步骤7 返回实例管理列表页面，查看当前实例状态为“规格变更中”，也可在任务中心查看变更任务。

----结束

4.5 计算节点扩容

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。

使用须知

- 计算节点扩容期间，服务不中断，不影响业务的正常运行。
- 请在业务低峰时间段进行节点扩容操作。
- 请确保实例关联的数据节点状态正常并且没有进行其他操作。
- 一个DDM实例最多支持32个节点。
- 如果开启了只读业务隔离特性，即创建了只读组，节点扩容功能入口将移动到组列表的操作列。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 单击“节点扩容”。如果您已经完成创建组操作，请在“组信息”模块中单击“操作”列的“节点扩容”。

步骤4 在节点扩容页面，选择可用区和节点个数。

您可以单击操作列的“添加”，添加多个节点。一个DDM实例最多支持32个节点。

图 4-4 节点扩容选择

步骤5 设置完节点数，单击页面下方的“下一步”。

步骤6 在规格确认页面，如果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。

----结束

4.6 计算节点缩容

随着业务数据的减少，为了降低成本，您可对实例节点进行缩容。

使用须知

- 计算节点缩容期间，服务不中断，不影响业务的正常运行。
- 请在业务低峰时间段进行节点缩容操作，对于“按需计费”类型的实例在请求提交后立即执行。
- 请确保实例关联的数据节点状态正常并且没有进行其他操作。
- 一个DDM实例最少保留1个节点。
- 如果开启了只读业务隔离特性，即创建了只读组，节点缩容功能入口将移动到组列表的操作列。

操作步骤

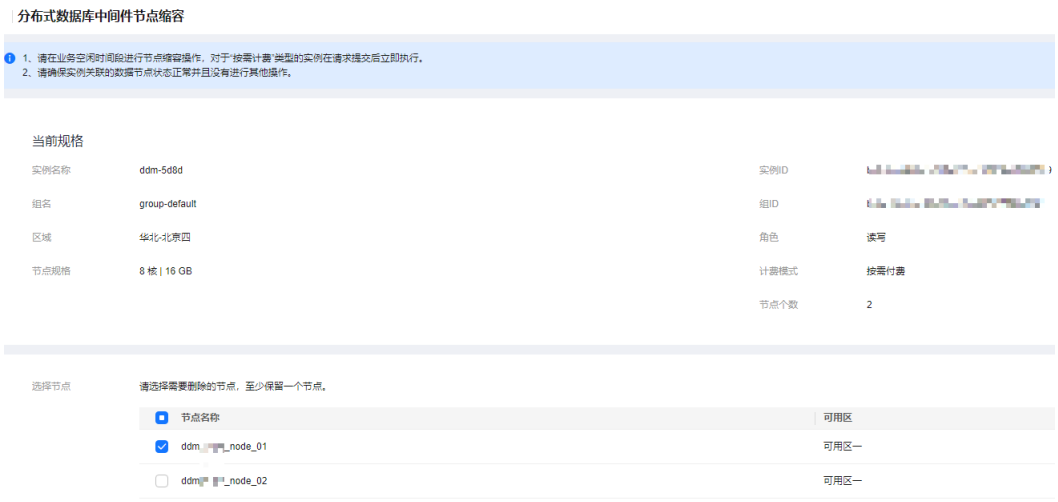
步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 单击“节点缩容”。如果您已经完成创建组操作，请在“组信息”模块中单击“操作”列的“更多 > 节点缩容”。

步骤4 在节点缩容页面，您可查看当前规格，并设置节点缩容数量。

图 4-5 节点缩容规格选择



步骤5 设置完节点数，单击页面下方的“下一步”。

步骤6 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。

----结束

4.7 重启实例或节点

通常出于维护目的，您可能需要重启数据库实例。您可以通过控制台对整个实例或者单个节点执行重启操作。

使用须知

实例重启期间服务不可用且操作无法撤销，请谨慎操作。

重启整个实例

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，在目标实例操作栏，选择“更多 > 重启实例”。

您也可以在实例管理列表页面，单击目标实例名称，进入“基本信息页”，在页面右上角单击“重启”。

步骤3 在弹出确认窗口中，单击“是”。

步骤4 在实例管理列表页面，等待实例重启成功。

----结束

重启单个节点

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入基本信息页。

步骤3 在“节点信息”模块中选择目标节点，单击操作列的“重启节点”。

如果开启了只读业务隔离特性，即创建了只读组，您需要在“组信息”模块中单击组前面的后单击操作列的“重启节点”。

步骤4 在弹出确认窗口中，单击“是”。

步骤5 等待重启节点成功。

----结束

4.8 删除按需实例

对于“按需计费”的DDM实例，您可根据业务需要，在“实例管理”页面手动删除实例来释放资源。

对于“包年包月”的DDM实例，您可以执行退订操作，系统将根据资源是否使用代金券和折扣券等条件返还一定金额到您的帐户。详细的退订规则请参见**云服务退订**。

使用须知

- 正在执行操作的实例不能手动删除，只有在实例操作完成后，才可删除实例。
- 删除操作无法恢复，请谨慎操作。
- 如果DDM上存在已关联的RDS for MySQL，删除DDM时系统会提醒您关联的实例信息，包括实例名称、实例状态和数据库类型。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，在目标实例操作栏，选择“更多 > 删除实例”。

说明

- 如需删除挂载于DDM上的数据节点的数据，请勾选“删除数据节点上的数据”。
- 如果DDM上存在已关联的RDS for MySQL实例，删除DDM时系统会提醒您关联的实例信息，包括实例名称、实例状态和数据库类型。
- 包年/包月DDM实例不能直接删除，如需删除请通过“费用中心 > 订单管理 > 退订与退换货 > 云服务管理”执行资源退订操作。

步骤3 单击“是”完成删除实例操作。

----结束

4.9 表数据重载

DDM跨region容灾场景下，通过数据复制服务（DRS）进行存储层数据迁移，迁移完成之后，目标DDM无法感知逻辑表信息所在位置，所以需要在目标DDM主动下发“表数据重载”，重新加载信息，跟分片建立联系。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择目标实例。

步骤3 在操作栏，选择“更多 > 表数据重载”。

----结束

4.10 设置参数模板

您可以通过实例管理页面“设置参数模板”的功能为DDM实例关联参数模板。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择目标实例。

步骤3 在操作栏，选择“更多 > 设置参数模板”。

步骤4 选择目标参数模板，单击“确定”。

----结束

4.11 修改 DDM 实例参数

为了确保DDM服务的性能更好的体现，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。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理页面单击实例名称，进入实例信息详情页。

步骤3 在左侧导航栏中单击“参数管理”。

您可以根据需要修改对应参数。

图 4-6 参数管理

DDM默认支持修改的实例参数请参见**实例参数说明**。

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 4-7 未使用 bind_table 结果展示

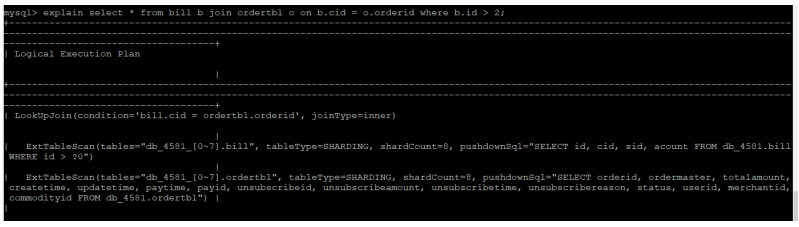
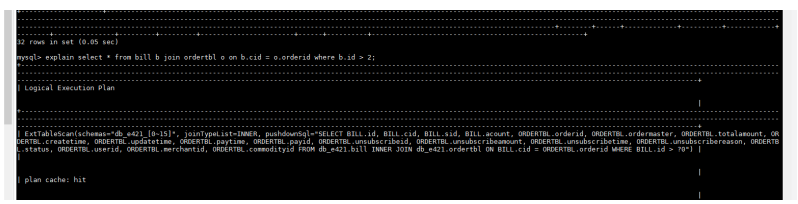


图 4-8 使用 bind_table 结果展示



步骤4 确认无误后，单击“保存”，并在弹框中单击“是”完成参数修改。

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

4.12 版本回滚

操作场景

DDM实例升级到新版本后，支持将内核版本回滚至最近一次升级前版本。

注意事项

- 回滚数据库内核版本会重启DDM实例，服务可能会出现闪断，请您尽量在业务低峰期执行该操作，或确保您的应用有自动重连机制。
- 版本回滚只支持回滚至最近一次升级前版本。
- 实例版本升级后如果进行过节点扩容，需将新扩节点做缩节点处理，再进行版本回滚。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的目标实例，单击实例名称。

步骤3 在实例基本信息页面，单击“实例信息”模块的“版本回滚”。

步骤4 在版本回滚弹窗中选单击“立即回滚”。

步骤5 确认无误后单击“是”进行版本回滚。

步骤6 版本回滚时，实例状态将变为“回滚中”。

步骤7 版本回滚完成后，实例状态由“回滚中”变为“运行中”，版本将显示回滚后的版本号。

----结束

4.13 版本升级

什么是 DDM 系列优选版本?

DDM内核版本通常由4位数字组成（如3.0.8.x），取前三位数字作为大版本号（如3.0.8）。每个大版本会在版本迭代过程中发布一系列小版本，本系列优选版本是当前大版本下的推荐版本，通常是最新且最稳定的小版本。对于同一大版本下的DDM实例，将内核版本升级至本系列优选版本属于小版本升级，通常涉及问题修复和优化，语法兼容风险较小，推荐客户将实例升级至本系列优选版本。

什么是 DDM 最新版本?

当前DDM最新大版本的优选版本。

操作场景

- DDM支持手动升级内核版本，可选择当前系列优选版本和最新版本升级。
  - 系列优选版本：相同大版本下的推荐版本。改动较小，兼容风险较小。
  - 最新版本：最新大版本下的推荐版本。改动涉及新特性、性能优化、问题修复，属于大版本升级，存在兼容性风险，建议升级前做充分的业务测试。
- 新创建的DDM实例默认为最新版本。如果华为云有新的内核版本发布时，您可以在“实例管理”页面的“版本”列看到内核版本升级提示，单击“版本升级”弹出升级版本弹窗。

注意事项

- 升级数据库内核版本会重启DDM实例，服务可能会出现闪断，请您尽量在业务低峰期执行该操作，或确保您的应用有自动重连机制。
- 如果实例已经为本系列优选版本，则只可升级至最新版本。
- 如果当前版本与升级目标版本跨度较大，请务必在测试实例上做好充分的业务兼容性测试后，再进行生产实例的版本升级，确保生产业务稳定不受影响。
- 版本升级后如有业务不兼容问题，可及时将版本回滚至升级前版本，详细内容请参考**版本回滚**。
- 内核版本说明详情请参见**内核版本说明**。
- 如果DDM实例的VPC已开启IPv6子网，版本升级的目标版本必须小于等于3.0.9版本或者大于等于3.1.3版本。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的目标实例，单击实例名称。

步骤3 在实例基本信息页面，单击“实例信息”模块的“版本升级”。

您也可以在“实例管理”页面的“版本”处单击“版本升级”弹出升级版本弹窗。

步骤4 在升级版本弹窗中选择目标版本，单击“立即升级”。

- 系列优选版本：相同大版本下的推荐版本。
- 最新版本：最新大版本下的推荐版本。

图 4-9 选择目标版本

步骤5 确认无误后单击“是”进行版本升级。

步骤6 版本升级时，实例状态将变为“升级中”。

步骤7 版本升级完成后，实例状态由“升级中”变为“运行中”，版本将显示升级后的版本号。

----结束

4.14 DDM 引擎及操作系统更新

当前DDM引擎及OS暂不支持租户侧维护窗口自助升级，如果需要升级，您可以联系华为云客服，由华为云工程师在给出升级分析评估后进行升级。

华为云仍然会通过热补丁方式及时修复对引擎及操作系统影响重大的漏洞。

5 连接管理

5.1 访问控制

操作场景

当前创建DDM实例时，DDM默认支持负载均衡（个别没有负载均衡能力的局点除外）。当业务通过控制台提供的内网地址连接DDM时，默认不限制访问的IP地址，即此时安全组是失效的，需要通过“访问控制”功能来做访问的安全控制。如果直连DDM节点，安全组依然有效。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理页面，单击目标实例名称，进入基本信息页面。

步骤3 打开访问控制开关。

- 实例只有一个组时，在“网络信息”区域，单击“访问控制”右侧的，打开访问控制开关。

  图 5-1 单个组开启访问控制

- 实例有多个组时，访问控制开关将移动到“组信息”模块中。您可以在实例基本信息页面下方的“组信息”列表中，单击“访问控制”列的打开访问控制开关。

  图 5-2 多个组开启访问控制

步骤4 单击“设置”，弹出“修改访问控制”弹框，输入对应名单的IP地址，单击“是”。

图 5-3 修改访问控制

----结束

5.2 修改实例和组内网地址

操作场景

DDM开启了负载均衡之后，支持修改实例以及组的内网地址。

修改实例的内网地址

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称，进入实例基本信息页面。

步骤3 在“网络信息”模块“内网地址”处，单击“修改”。

步骤4 在弹框中，输入与当前内网地址在同一VPC子网下的其他未被使用的地址，单击“确定”。

图 5-4 修改内网地址

----结束

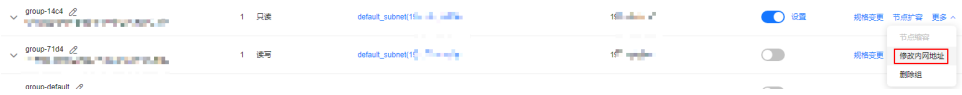
修改组的内网地址

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称，进入实例基本信息页面。

步骤3 在“组信息”模块，在目标组“操作”列选择“更多 > 修改内网地址”。

图 5-5 组信息



步骤4 在弹框中，输入与当前内网地址在同一VPC子网下的其他未被使用的地址，单击“确定”。

图 5-6 修改内网地址

----结束

5.3 绑定和解绑弹性公网 IP

DDM实例创建成功后，默认未开启公网访问功能（即未绑定弹性公网IP）。DDM支持用户绑定实例的弹性公网IP，通过公共网络访问数据库实例，绑定后也可根据需要解绑。

使用须知

- DDM实例开启负载均衡之后，如果用户绑定了弹性公网IP，系统将会在DDM实例下的多个节点中随机选择一个读写节点实施弹性公网IP绑定。此操作将会无法实现负载均衡，因此该场景下不建议客户绑定弹性公网IP。

  如需为DDM实例开通负载均衡，您可以在管理控制台右上角，选择“工单 > 新建工单”，提交开通负载均衡的申请。
- 对于已绑定弹性公网IP的实例，需解绑后，才可重新绑定其他弹性公网IP。
- 对于包含DDM只读组的实例，仅支持绑定到所属的DDM读写组的某个节点。

前提条件

已在VPC申请弹性公网IP。

绑定实例的弹性公网 IP

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称，进入实例基本信息页面。

步骤3 在“实例信息”模块“弹性公网IP”处，单击“绑定”。

步骤4 在弹出框的地址列表中，显示“未绑定”状态的弹性公网IP，选择所需绑定的弹性公网IP，单击“确认”，提交绑定任务。

如果没有可用的弹性公网IP，单击“查看弹性公网IP”，获取弹性公网IP。

图 5-7 绑定弹性公网 IP 弹窗

步骤5 在实例基本信息页面，查看绑定成功的弹性公网IP。

----结束

解绑实例的弹性公网 IP

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定实例，单击实例名称，进入实例基本信息页面。

步骤3 在“实例信息”模块“弹性公网IP”处，单击“解绑”。

步骤4 在弹出框中单击“是”，解绑弹性公网IP。

步骤5 在实例基本信息页面，查看解绑成功。

----结束

5.4 修改 DDM 服务端口

DDM服务支持修改实例的端口，修改后会重启DDM实例。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称。

步骤3 在基本信息页面，在“网络信息”模块的“DDM服务端口”处，单击，修改服务端口。

此端口为DDM实例对外提供服务的端口，默认为5066，设置范围为1025~65534，其中1033、7009、8888、12017被DDM系统占用，不可设置。

步骤4 单击，提交修改。

- 在弹出框中，单击“是”，提交修改。修改实例的服务端口，需要重启实例。
- 在弹出框中，单击“否”，取消本次修改。

步骤5 在实例的基本信息页面，查看修改结果。

----结束

5.5 修改实例安全组

DDM服务支持修改数据库实例的安全组。

更多配置安全组操作，请参考**如何选择和配置安全组**。

使用须知

修改安全组后可能导致当前DDM实例与关联的数据节点网络不通，请谨慎选择。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称。

步骤3 在基本信息页面，在“网络信息”模块的“安全组”处，单击，选择对应的安全组。

步骤4 单击，提交修改。

步骤5 在实例的基本信息页面，查看修改结果。

----结束

6 逻辑库管理

6.1 创建逻辑库

本章节主要介绍在DDM控制台创建逻辑库的方法。

使用须知

- 创建逻辑库时，仅支持关联和DDM实例处于相同VPC的数据节点，且数据节点没有被其他DDM实例使用。DDM将在关联的数据节点上新建数据库，不会影响已有的库表。
- 创建逻辑库时，同一个逻辑库，MySQL大版本需要相同，不可以混用。
- 数据节点的规格建议不小于DDM的规格，否则会影响性能。
- 创建逻辑库时，同一个DDM实例可以创建多个逻辑库。多个逻辑库可关联同一个数据节点。
- 一个数据节点无法被不同的DDM实例关联。
- 创建逻辑库时选多个分片的场合，分片名遵循“逻辑库名_xxxx”的命名规则，其中xxxx为从“0000”开始递增的数字。如逻辑库名为“db_cbb5”，总分片数为2，则分片名为“db_cbb5_0000”和“db_cbb5_0001”。
- 创建逻辑库时，关联的数据节点不能处于只读状态。
- DDM在数据节点上创建的内部账号（DDMRW*、DDMR*、DDMREP*）请勿修改和删除，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）+数据节点ID取HASH值。
  - 口令规则：口令随机生成，长度最小16，最长32。

前提条件

- 已**购买实例**，并且实例状态为“运行中”。
- 已**创建账号**。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例操作栏“创建逻辑库”。

图 6-1 创建逻辑库-入口 1

您也可以在实例管理列表页面，单击目标实例名称，进入基本信息页面。在左侧导航栏选择逻辑库管理页签，在页面右侧单击创建逻辑库。

图 6-2 创建逻辑库-入口 2

步骤3 在创建逻辑库页面，填选逻辑库模式、逻辑库名称、需要关联的DDM账号、数据节点和逻辑库总分片数。

图 6-3 创建逻辑库

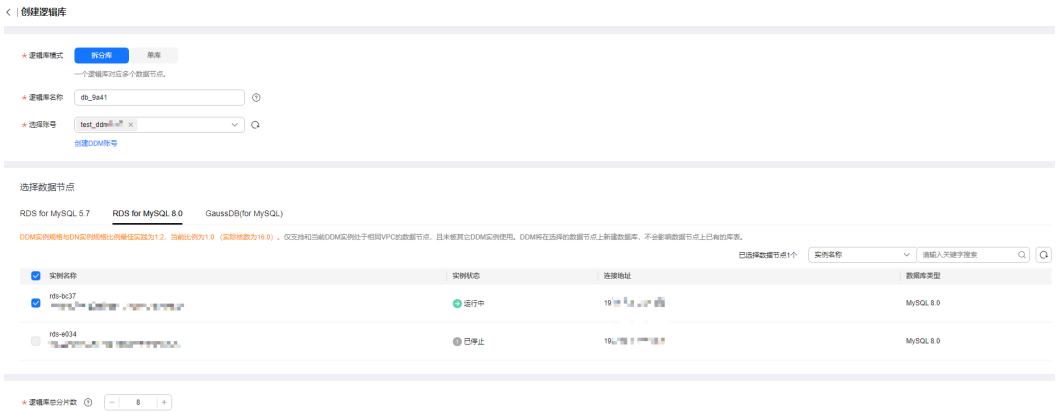


表 6-1 参数说明

参数名称	说明
逻辑库模式	• 拆分库：一个逻辑库可以关联多个数据节点，分片数均匀的分布在这些数据节点上。 • 单库：一个逻辑库仅关联一个数据节点，在该数据节点上仅创建一个分片。

参数名称	说明
逻辑库名称	长度为2-48个字符，以小写字母开头且仅支持小写，可以包含小写字母、数字、下划线。
选择账号	需要关联的DDM账号。
选择数据节点	仅支持和当前DDM实例处于相同VPC、且未被其他DDM实例使用的数据节点。DDM将在选择的数据节点上新建数据库，不会影响数据节点上已有的库表。
逻辑库总分片数	逻辑库总分片数是所选数据节点分片数的总和，为了确保每个数据节点上都能均匀分配到分片，逻辑库总分片数不能小于选择的数据节点数。考虑到业务会持续增长，建议每个数据节点上最小8分片，最大不超过64分片。

步骤4 单击“下一步”。

步骤5 在数据节点可用性检测页面，输入关联数据节点的账号及密码，单击“测试”。

说明

数据节点账号所需权限: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER WITH GRANT OPTION。

建议您提前在数据节点上创建具有上述权限的账号。

图 6-4 数据节点可用性检测

步骤6 测试通过后，单击页面下方的“完成”。

完成创建后，可观察“逻辑库状态”如下：

- 创建中
- 创建失败
- 运行中

----结束

6.2 导出逻辑库

用于跨region容灾或者数据迁移场景。在源实例进行导出，导出的逻辑库信息主要包含逻辑库的基本信息和分片信息，不包含业务数据和索引数据。

前提条件

DDM实例中已创建逻辑库。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，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3 在左侧导航栏，选择“逻辑库管理”，查看对应实例逻辑库列表。

图 6-5 逻辑库列表

步骤4 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

6.3 导入逻辑库

用于跨Region容灾或者数据迁移场景。在目标实例进行导入，导入的逻辑库信息主要包含逻辑库的基本信息和分片信息，不包含业务数据和索引数据。

前提条件

DDM实例没有逻辑库。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，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3 在左侧导航栏，选择“逻辑库管理”，查看对应实例逻辑库列表。

步骤4 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5 在“导入逻辑库信息”页面单击“上传文件”，在本地选择需要导入的.json文件（**导出逻辑库**导出的JSON文件），选择需要使用的数据节点，输入具有相关权限的数据库账号及密码，单击“完成”即可。

说明

- 选择的数据节点的数量与导入DDM关联的数据节点数量一致。
- 数据库账号所需权限：SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER WITH GRANT OPTION。

----结束

6.4 删除逻辑库

您可以删除不再需要的逻辑库。

使用须知

删除操作无法恢复，请谨慎操作。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 在左侧导航栏选择“逻辑库管理”，查看对应实例逻辑库列表。

步骤4 在逻辑库列表页面，选择目标逻辑库，操作列单击“删除”。

步骤5 在删除确认弹窗中，单击“是”。

说明

- 请勿直接在数据节点列表删除和DDM逻辑库关联的实例，会直接导致逻辑库故障。
- 如需删除数据节点上的数据，请在删除逻辑库的弹窗中勾选“删除数据节点上的数据”。
- 如果您想删除逻辑库，请首先确认数据节点是否存在。如果实例已删除，请先单击“同步DN信息”，再进行删除操作。
- 如果您所连接的数据节点有名称、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DN信息”同步最新配置。

----结束

6.5 配置 SQL 黑名单

配置SQL黑名单即配置该逻辑库不允许执行的SQL语句。

该功能主要用于解决用户某类突发的并发过高的SQL导致DDM实例不稳定的场景。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，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3 在左侧导航栏，选择“逻辑库管理”，查看DDM实例逻辑库。

步骤4 在逻辑库列表页面，单击右侧操作栏的“配置SQL黑名单”。

图 6-6 配置 SQL 黑名单

步骤5 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，例如带有DROP或者DELETE的SQL语句。
- 全量匹配：禁止在对应逻辑库执行该SQL语句，SQL语句中如果有多个空格或者换行，将不会被替换为单个空格或截断为单个空格来匹配。
- 正则匹配：禁止在对应逻辑库执行含有该正则表达式的SQL语句。
- 配置的黑名单SQL之间以英文分号隔开，前缀匹配、全量匹配、正则匹配中的SQL语句大小分别不超过1KB。
- 如果在配置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

6.6 查看逻辑库列表和逻辑表信息

您可以通过DDM控制台查看DDM实例的逻辑库列表和逻辑表信息，无需使用代码查询。

使用须知

- 创建失败的逻辑库，无法查看该逻辑库的表结构。

- 查看逻辑表信息时，逻辑库的状态为“运行中”或“分片变更中”。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，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3 在实例基本信息页面左侧导航栏，选择“逻辑库管理”选项卡，查看DDM实例逻辑库。

图 6-7 逻辑库列表

说明

如果您需要查看DDM实例逻辑库的磁盘大小，请联系客服提交申请。

- 仅支持查看状态为“运行中”的逻辑库。
- 页面显示的磁盘大小为预估值，存在误差。
- 查询逻辑库的磁盘大小会影响DDM实例的性能。

步骤4 在逻辑库列表页面，单击目标逻辑库名称或者单击操作列的“管理”，进入逻辑库基本信息页面。

- 在“已关联实例”页签下，查看当前逻辑库中的实例信息。
- 在“连接地址栏”页签下，获取DDM实例的“命令行连接地址”和“jdbc连接地址”。

  连接逻辑库的方法请参见**连接DDM逻辑库**。

----结束

7 分片变更

7.1 特性和应用场景介绍

分片变更是DDM的一项核心功能，通过增加数据节点数或者增加分片数，提高数据存储能力和并发支持能力。可解决随着业务增长，逻辑库对应的物理存储空间不足问题。分片变更过程对业务影响相对较小，可在不影响您业务使用的情况下快速解决业务在快速发展的过程中针对数据库扩展性产生的后顾之忧与运维压力。

应用场景

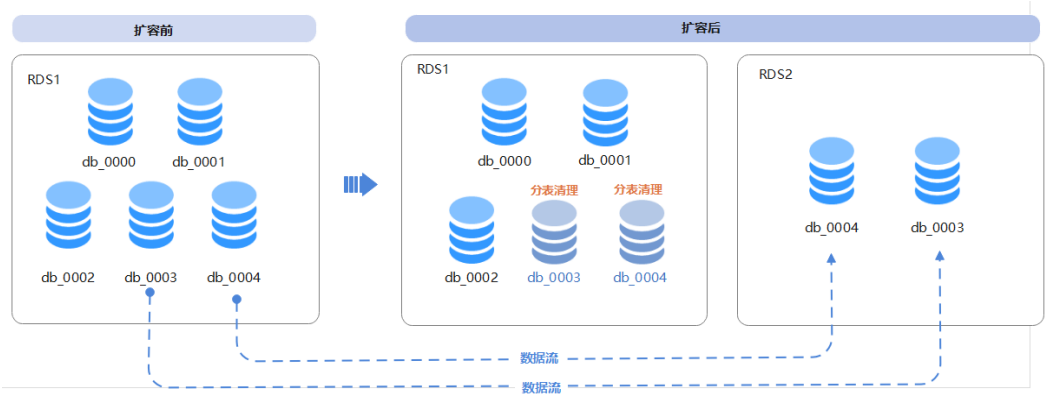
DDM支持以下三种不同的分片变更方式，以满足不同的应用场景。

方式一：分片数不变，增加数据节点数量

该变更方式不改变当前分片数，只增加数据节点数量。将原数据节点的部分分片平移到新增数据节点上，分片数据进行平移，数据相对位置不需要重新分布，所以变更速度最快，推荐您优先使用此方式进行分片变更。

该方式适用于水平拆分业务后业务规模快速增长的场景，可在业务初期减少成本。也适用于RDS for MySQL实例无法满足存储空间，读写性能的场景。

图 7-1 分片数不变，增加 RDS for MySQL 实例数量

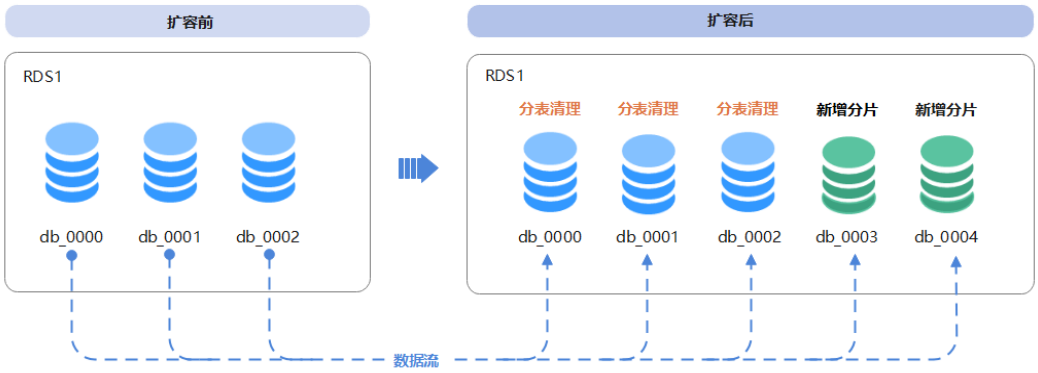


方式二：增加分片数，不增加数据节点数量

该变更方式增加分片数，不增加数据节点数量。此种情况分片总数、分表总数、分表规则都会发生变化，数据将重新分布到不同的分片中，原来分片上的表将被清理，广播表分片数量增加。

该方式适用于单个物理表数据量过大，查询性能受到限制，但是整体RDS for MySQL实例可用空间充足的场景。

图 7-2 增加分片数，不增加 RDS for MySQL 实例数

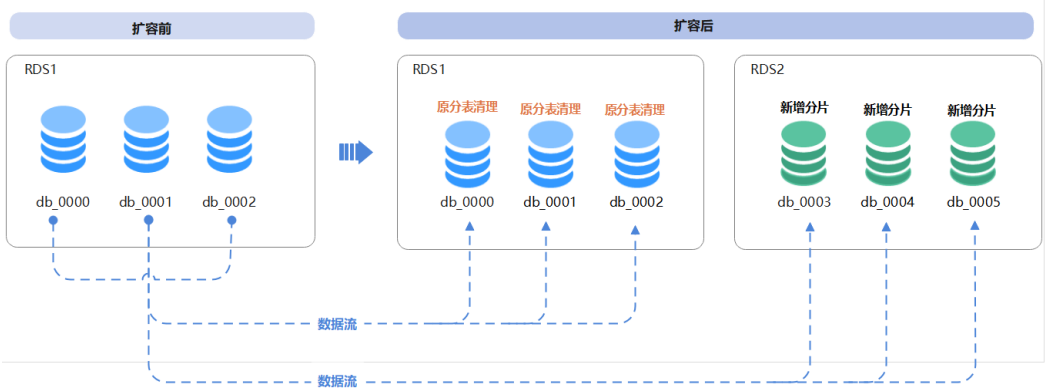


方式三：增加分片数，也增加数据节点数量

该变更方式既增加分片数，也增加数据节点数量。此种情况分片总数、分表总数、分表规则都会发生变化，数据将重新分布到不同的分片中，原来分片上的表将被清理，广播表分片数量增加。

该方式适用于RDS for MySQL实例无法满足存储空间，读写性能，且单个物理表数据量过大，查询性能受到限制的场景。

图 7-3 既增加分片数，也增加 RDS for MySQL 实例数



7.2 变更评估

分片变更前，可根据以下几点对分片变更做一个初步的评估，根据评估结果选择适当的新分片数, DDM实例规格和DN实例规格, 以及DN节点个数。

- 数据量：可用***show db status***命令来获得目前现网的数据量。
- DDM规格：当前DDM实例的CPU个数、内存大小、DDM节点数。

- DN规格：当前DN节点的vCPU个数和内存大小以及DN节点数。
- 业务情况：目前业务规模以及对日后增长趋势的预估。分片变更是重要的数据变更动作，如果DN节点的存储空间足够，建议您推迟执行分片变更操作。
- 是否增加分片：增加分片意味着拆分规则发生变化，当前逻辑库中所有的数据都需要按照新拆分规则重新计算并移动位置，相比不增加分片的变更需要更大的资源占用，速度也相对较慢。
- 分片变更过程中是否要执行DDL：当前分片变更过程中，客户读写业务不受影响，但为了保证数据一致性，不允许在分片变更过程中执行DDL，请您合理规划。

客户案例：

某客户当前有DDM实例共4个节点，规格均为8U16GB，关联了6个数据节点（DN实例），数据量约12TB，1000亿条数据，7.3w张物理分表，业务量较大。

由于分片数变化就一定会引起数据重分布，需要迁移逻辑库的全部数据，而且每一条都需要经过重新路由，计算速度上会明显慢于分片数不变的变更。综合考虑客户业务，建议客户先将DDM实例规格升为32U64GB（DDM支持弹性扩缩容, 可以在变更结束后还原为之前的规格），再增加数据节点至12个并升级DDM引擎内核版本到最新版本。由于分片数没有发生变化，仅需要将一半的物理分片从原DN移动到新的DN节点上，且不涉及路由重计算。除非是单个物理表存放的数据量达到上限, 一般建议使用分片数不变, 增加数据节点的平移变更方式。

7.3 预检查

为了避免分片变更失败，请您最晚在变更前一天完成以下内容的检查。

预检查内容

表 7-1 预检查内容

检查内容	检查目的	检查未通过解决方案
表名长度检查	分片变更需要数据重分布时（例如：增加分片），会创建临时表，临时表的表名长度会略长于原表名，需确保临时表的表名长度不超过MySQL限制。	请修改过长的表名。
DN实例binlog全量备份时间检查	客户全量备份是否保留足够长时间。	进入DN console，确保全量备份保留时间大于等于30天。
DN节点binlog必须开启	Binlog必须开启以支持在线变更。	如果您的DN节点是RDS实例，确保Binlog开启。
DN节点binlog本地保留时间检查	Binlog在DN节点上的保留时间必须足够长。	如果您的DN节点是RDS实例，无需解决。
广播表数据一致性检查	保证广播表数据一致后再执行分片变更。	请联系DDM运维人员。

检查内容	检查目的	检查未通过解决方案
源物理分片的字符集和排序规则检查	保证分片变更后字符的展示和排序一致。	请联系DDM运维人员。
物理表建表语句检查	保证各物理分片上的表结构保持一致。	请先使用check table 命令查询表结构不一致详情，之后再使用alter 语法对表进行修正。
主键检查	要求源库所有表都具有主键，且拆分键是主键一部分以保证分片变更后数据一致性。	如果表不存在主键，请使用alter语句增加主键。
数据库实例链接检查	DN节点是否可连接。	检查安全组等配置。
数据库实例参数检查	源DN节点和目标DN节点数据库关键参数配置需要相同。	请进入DN console对参数配置进行修改。
数据库实例磁盘空间检查	防止分片变更过程中，DN节点磁盘不足。	对DN节点进行磁盘扩容。 **注意** 本项检查根据估计值进行判断，极端情况下与实际值有一定差别。
数据库实例时区检查	源DN节点和目标DN节点时区要求相同。	请进入DN console参数配置对时区进行修改。
物理分表个数限制检查	在增加分片场景中，源表每一条数据都需要重新计算路由并分发到新的物理分表中。如果分片变更后物理分表数过多，分片变更耗时过长，需检查单个DN节点的物理分表个数是否超过限制。	请联系客服处理。

常见问题和解决方案

- 物理表结构不一致导致扩容失败。

  解决方案：请先使用check table语句对表结构做一致性检查，并配合alter table等语句对表结构做修正。如果无法进行DDL修正（如主键、唯一键因为数据原因无法修改），请联系运维人员处理。

- 没有主键的表不支持迁移。 如果没有主键，就无法精确定位记录，在分片变更过程中如果发生重试，可能导致数据增多。

  解决方案：增加主键。

- 分片键不是主键的一部分可能导致逻辑表存在主键重复的数据(因为位于不同的物理分表内)。当数据需要重分布时，这类数据如果路由到同一物理表，由于主键相同，将只会保留一条，必定会导致迁移后的数据量和原来不一致，而导致分片变更失败。

说明

主键是全局唯一序列和分片数不变化的情况下不会发生此种错误。

解决方案：订正数据，再重新校验。

7.4 分片变更操作指导

本章节以RDS for MySQL实例为例说明分片变更的使用方法。

前提条件

- DDM实例中已创建逻辑库。
- 已有RDS for MySQL实例与DDM实例处于相同的VPC，该RDS for MySQL实例没有被其它DDM实例使用。如果需增加DN实例，则DN实例与DDM实例需要处于相同的VPC。
- 本特性需满足DDM内核版本大于等于3.0.8.3版本，建议您使用最新的内核版本来进行分片变更。
- DDM实例关联的数据节点不能处于只读状态。
- 拆分模式为“非拆分”的逻辑库暂不支持分片变更功能。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，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3 在实例基本信息页面左侧导航栏，选择“逻辑库管理”选项卡，查看DDM实例逻辑库。

步骤4 在逻辑库列表页面，单击“操作”列“分片变更”。

图 7-4 选择分片变更操作

步骤5 在“分片变更”页面，填选对应参数，单击“测试”。

图 7-5 分片变更

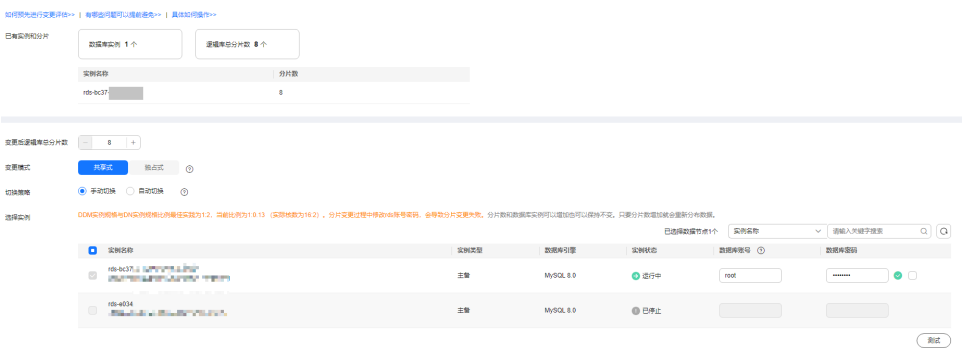


表 7-2 参数说明

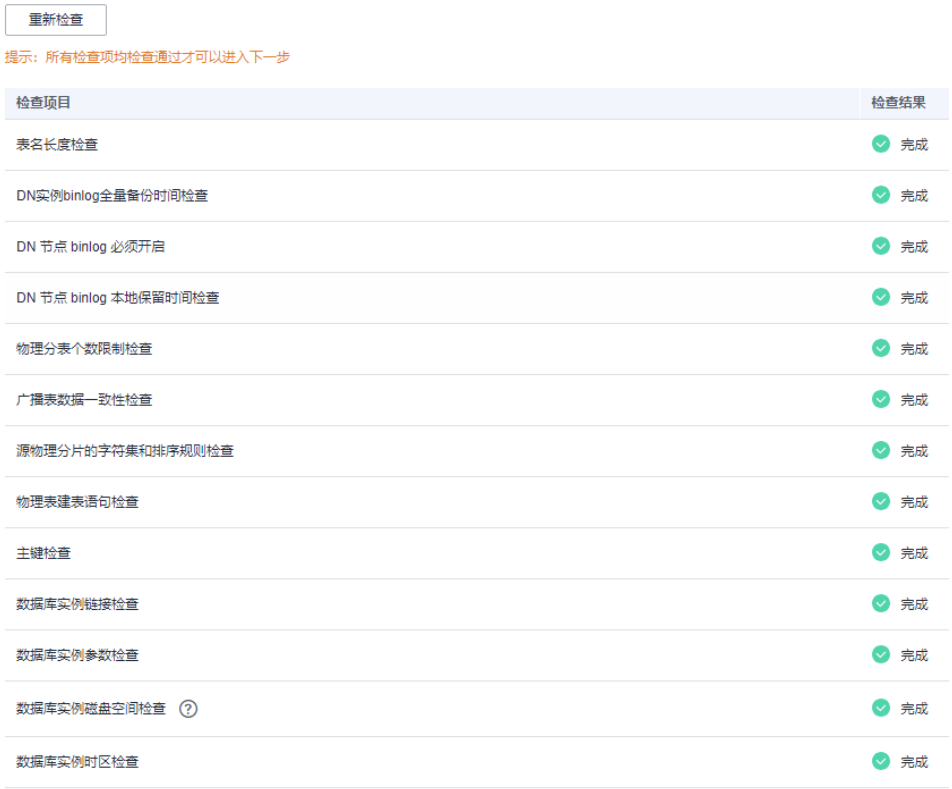
参数名称	说明
已有实例和分片	展示已有实例和分片数的信息。
变更后逻辑库总分片数	默认显示的是当前已有的总分片数，如果需要增加分片，请填写增加后的总分片数，DDM会尽量均匀分配到数据节点上。
变更模式	默认为共享式变更模式，您可以业务需要选择变更模式。 ● 共享式：节点资源同时用于业务与分片变更，分片变更性能将受到较大影响。 ● 独占式：独占式将从默认读写组中随机选择正常节点来变更，选择的节点资源全部用于分片变更，可以提高分片变更的性能。如果您担心节点独占影响您的业务，可以先进行计算节点扩容。 选择独占式变更模式时，默认组要开启负载均衡，且节点数大于1个。
切换策略	支持手动切换和自动切换两种方式。 切换任务会将读写流量切换到新增的实例上，在切换过程中，会有一到两次闪断，服务不受影响。建议在业务低峰期执行切换。
选择实例	默认选中的是已有的实例，您可以根据业务需要选择是否增加数据库实例。 数据库实例需要填写账号密码进行连接测试，测试连接成功后才可进行下一步操作。

说明

- DDM分片变更不支持无主键表。
- 逻辑库在单数据节点上的物理分片数不超过64。如果因业务需要分片数超过64，请联系DDM客服人员。
- 数据库账号所需权限：SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER WITH GRANT OPTION。

步骤6 测试通过后，单击“下一步”，进入预检查页面。

图 7-6 预检查



说明

- 预检查时，实际还未开始真正的变更，只有单击“确定”下发任务后才开始。
- 检查项中如果出现风险项，请确保该风险项不会影响业务后，再单击“忽略此风险”，建议先解决风险项，再进行变更。

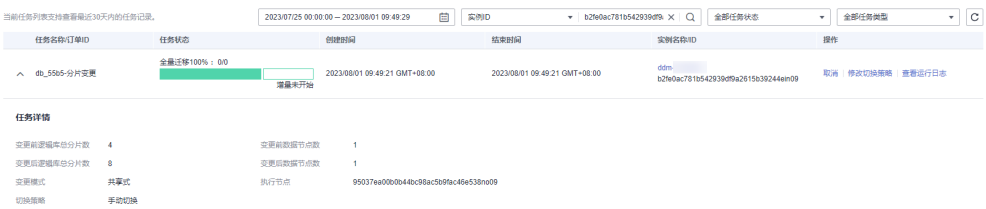
步骤7 检查完成后，单击“开始分片变更”。

- 分片变更任务进行中，数据迁移分为全量迁移和增量待处理两个阶段。

  您可以通过“任务中心”的进度条查看迁移进度。

  单击任务前的 可查看本次分片变更的任务详情，包含变更前后逻辑库总分片数、变更前后数据节点数、切换策略等详细内容。

  图 7-7 查看分片变更任务进度条



  也可以通过在SQL客户端执行***“show migrate status”***命令来查看分片变更的操作进度。

图 7-8 执行命令查看进度

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

说明

一个源RDS实例查询出一条记录，存在N个源RDS实例会查询出N条记录。

SOURCE_RDS：源RDS实例。

MIGRATE_ID：扩容ID。

SUCCEED_TABLE_STRUCTURE：已经迁移成功的表结构数。

TOTAL_TABLE_STRUCTURE：需要迁移的表结构总数。

SUCCEED_TABLE_DATA：已经迁移成功的表数据的数量，以表为计量单位。

TOTAL_TABLE_DATA：需要迁移表数据的总数量，以表为计量单位。

SUCCEED_INDEX_DATA：已经迁移成功的索引的数量，以表为计量单位。

TOTAL_INDEX_DATA：需要迁移索引的总数量，以表为计量单位。

FULL_SUCCEED_COUNT：当前扩容子任务的已完成全量迁移的对象总量。

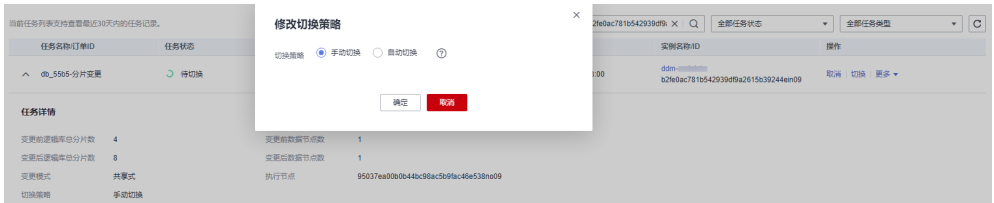
FULL_TOTAL_COUNT：当前扩容子任务的全部全量迁移对象总量。

FULL_PERCENTAGE：当前扩容子任务的全量迁移完成百分比。

将各个扩容子任务的全部全量对象总量和已完成全量对象总量进行汇聚，得到当前扩容任务的全量迁移总量和已完成量，即展示在“任务中心”的进度条数据

- 在“任务中心”单击操作栏的“更多 > 修改切换策略”可以对分片变更的切换策略进行修改。

图 7-9 修改切换策略



- 在“任务中心”单击操作栏的“更多 > 查看运行日志”可以查看任务的日志记录。

图 7-10 查看运行日志

- 数据迁移完成，如果切换策略选择了手动切换，需要在“任务中心”操作栏单击“切换”将路由切换到新的分片上或者数据节点上。如果切换策略选择了自动切换，任务将在设置的切换时间内，自动进行切换。

图 7-11 手动切换

说明

- 切换是本次变更的核心操作，未切换前本次分片变更还没有对原数据库中数据产生实质影响，可以通过取消任务来结束本次分片变更任务。
- 切换过程中，如果分片数未变只增加了RDS for MySQL实例进行分片平移，会禁写，如果分片数有变化，则会禁写禁读。
- 为了确保数据一致性，切换过程中DDM服务会进行数据完整性校验，导致切换时间变长，具体时间由数据量大小决定，建议在业务低峰期操作。

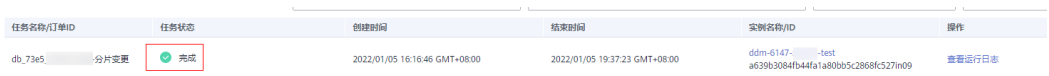
步骤8 分片变更结束后数据将会重新分布，确认完数据无误后可单击“清理”来清除原RDS for MySQL数据库实例的数据。

图 7-12 清理数据

步骤9 请仔细阅读弹窗内容，确认任务没有问题后单击“是”进行清理。

步骤10 清理完成。

图 7-13 清理完成



说明

清理操作使用drop语句对残留的原数据进行清理，由于MySQL的特性，清理完成后磁盘空间可能无法立刻完成释放。

步骤11 分片变更结束后，可使用以下命令进行检查。

show data node：查询新的数据节点和物理分片的对应关系。

show db status：查询逻辑库磁盘预估占用。

----结束

8 DN 管理

8.1 DN 管理介绍

DN管理提供数据节点管理服务，管理DDM实例关联的RDS for MySQL实例，展现实例的状态、存储、规格、读权重等信息，提供设置读权重、同步DN信息、开启读写分离的快捷操作。

表 8-1 功能介绍

功能	使用场景
同步DN信息	主要用于数据节点相关信息变化时（如增删只读实例，变更连接地址/端口号，规格变更，删除数据节点等操作），将数据节点变化的信息同步到DDM实例上。
开启/关闭读写分离	主要应用于DDM内核版本大于等于3.1.0版本的场景。 DDM内核版本大于等于3.1.0版本时，需要手动开启读写分离，然后调整只读实例和主实例的读写权重来实现读写分离操作。您也可以根据业务场景手动关闭读写分离功能。 如果DDM内核版本小于3.1.0版本，系统默认开启读写分离，只需要调整只读实例和主实例的读写权重即可以实现读写分离操作。此场景下不能关闭读写分离功能。
设置读权重	主要用于调整主实例和只读实例的读写权重，实现读写分离操作。 • 支持批量设置多个实例的读写权重。 • 如果数据节点未挂载只读实例，该主实例无法设置权重。

8.2 同步 DN 信息

同步DN信息主要用于数据节点相关信息变化时（如增删只读实例，变更连接地址/端口号，规格变更，删除数据节点等操作），将数据节点变化的信息同步到DDM实例上。

在数据节点相关信息变化时，需要用户主动通过同步DN信息将数据节点变化的信息同步到DDM，才能正常使用。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，选择目标实例。

步骤3 单击实例名称，进入基本信息页面。

步骤4 在左侧导航栏，选择“DN管理”页签，单击“同步DN信息”。

步骤5 系统自动触发同步DN信息命令，此过程大概需要1-3分钟。

----结束

8.3 开启读写分离

DDM对读写分离功能进行了优化，由之前的添加只读实例后自动开启读写分离变更为添加只读实例后，手动开启读写分离然后设置主实例和只读实例读的权重。

您可以根据自身业务需求开启或关闭读写分离。

使用须知

DDM内核版本需大于等于3.1.0版本。

开启读写分离

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，选择目标实例。

步骤3 单击实例名称，进入基本信息页面。

步骤4 在左侧导航栏，选择“DN管理”页签，单击“开启读写分离”。

步骤5 在弹窗中单击“确定”，开启读写分离。

说明

- 对于RDS for MySQL实例来说，开启读写分离后，会将主实例的读权重设置为100，您可以根据业务需求自行调整主实例和只读实例的读权重。
- 由于只读实例的数据是从主实例异步复制过来的，可能存在可见性延迟。
- 开启读写分离后，读查询会根据配置的**读权重**按比例分配至主或只读节点，分配至只读节点的查询可能存在一定的复制延迟。对于不在同一事务中，但是对前一个事务写入的数据存在逻辑依赖的查询语句，建议在查询语句中添加/*+ db_type=master*/，此hint可以指定本次查询强制走主节点，确保查询到前一个事务最新写入的数据。

----结束

关闭读写分离

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，选择目标实例。

步骤3 单击实例名称，进入基本信息页面。

步骤4 在左侧导航栏，选择“DN管理”页签。

步骤5 单击“关闭读写分离”。

说明

关闭读写分离之后，您的读写业务会在主实例上进行读写，已经设置的权重将不再生效，可能会对主实例所在业务有部分影响，请谨慎操作。

步骤6 在弹窗中单击“确定”，关闭读写分离。

----结束

8.4 设置读权重

主要用于调整主实例和只读实例的读写权重。对于拥有较多数据节点的DDM实例来说，可以批量配置数据节点读权重。

使用须知

如果数据节点未挂载只读实例，该主实例无法设置权重。

批量设置读权重

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，选择目标实例。

步骤3 单击实例名称，进入基本信息页面。

步骤4 在左侧导航栏，选择“DN管理”页签，单击上方的“设置读权重”。

图 8-1 DN 管理

步骤5 设置实例的读权重。

在批量设置的弹窗中，“同步”功能可以用来将第一个实例的读权重设置同步到其他的实例上。此操作需满足所有实例的只读实例数量一致才可以实行。

如果有实例的只读实例数量与其他实例不一致，则无法使用“同步”功能，需手动设置各个实例的读权重。

- 读权重可支持的设置范围为0~100。
- 只读实例挂载后默认承载全部可分离的只读请求，如果需要重新分配读写请求，可通过设置读权重来实现。
- 设置了实例的读权重后，主实例和只读实例将按照以下公式处理读请求。
  - 主实例处理读请求：主实例读权重/主实例和只读实例读权重总数
  - 只读实例处理读请求：只读实例读权重/主实例和只读实例读权重总数

例如：RDS for MySQL实例有1个主实例和1个只读实例，主实例和只读实例的读权重配置为20、80，则主实例和只读实例将按照1：4比例处理读请求。即主实例处理1/4的读请求，只读实例处理3/4的读请求，写请求自动发往主实例。

步骤6 读权重设置成功后，在DN管理列表页面将显示最新设置的权重数。

----结束

设置单个实例的读权重

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，选择目标实例。

步骤3 单击实例名称，进入基本信息页面。

步骤4 在左侧导航栏，选择“DN管理”页签。

步骤5 单击目标实例操作列“设置读权重”。

图 8-2 单个设置读权重

- 读权重可支持的设置范围为0~100。
- 只读实例挂载后默认承载全部可分离的只读请求，如果需要重新分配读写请求，可通过设置读权重来实现。
- 设置了实例的读权重后，主实例和只读实例将按照以下公式处理读请求。
  - 主实例处理读请求：主实例读权重/主实例和只读实例读权重总数
  - 只读实例处理读请求：只读实例读权重/主实例和只读实例读权重总数

  例如：RDS for MySQL实例有1个主实例和1个只读实例，主实例和只读实例的读权重配置为20、80，则主实例和只读实例将按照1：4比例处理读请求。即主实例处理1/4的读请求，只读实例处理3/4的读请求，写请求自动发往主实例。

步骤6 读权重设置成功后，在DN管理列表页面将显示最新设置的权重数。

----结束

8.5 读写分离操作指导

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数据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。此功能对应用透明，业务代码无需改造，只需要在控制台上设置主实例和只读实例的读权重，即可实现将读流量按照权重分流到主实例和只读实例上，写流量不受影响，默认会分流到主实例上。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。

只读实例上的数据是从主实例上异步复制而来，所以存在毫秒级的延迟。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，设置主实例和只读实例的权重为0:100，即所有只读查询均由只读实例承担，更好的保证主实例的性能。对于其他场景，建议结合实际情况酌情调整。

使用须知

- DDM内核版本大于等于3.1.0版本时，需要手动开启读写分离，然后调整只读实例和主实例的读权重来实现读写分离操作。
- 如果DDM内核版本小于3.1.0版本，系统默认开启读写分离，只需要调整只读实例和主实例的读权重即可以实现读写分离操作。
- 如果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。
- 如果存储节点主实例故障，此时只读实例上Seconds_Behind_Master=NULL，只读查询仍会下发到主实例执行，需要尽快恢复主实例。

前提条件

- 已购买DDM实例和带只读实例的数据节点。
- 已创建逻辑库。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 单击目标实例名称，进入实例基本信息页面。

步骤3 在左侧导航栏，单击“DN管理”页签。

步骤4 开启读写分离开关只影响读写组的节点。

- 内核版本大于等于3.1.0时，需手动开启读写分离，请参考**开启读写分离**，然后继续进行**步骤5**。
- 内核版本小于3.1.0时，系统默认开启读写分离，忽略此步骤，请继续进行**步骤5**。

说明

您也可以通过创建只读组的方式，无需开启读写分离开关，使用只读组的内网连接地址进行读流量，具体请参见**创建只读组**。

步骤5 设置读权重。

具体操作请参考**设置读权重**。

----结束

9 参数模板管理

9.1 实例参数说明

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

表 9-1 参数说明

参数名称	参数说明	取值范围	缺省值
bind_table	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。	• 格式要求：[{<table1>.<column1>,<table2>.<column2>}, {<table1>.<column2>,<table3>.<column1>},...] 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。	-
character_set_server	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字符集也为utf8mb4。 3.0.9版本之后，连接DDM实例执行show variables like '%char%'查询命令，查询结果中character_set_client/character_set_results/character_set_connection固定显示为utf8mb4。	gbk、utf8、utf8mb4	utf8mb4

参数名称	参数说明	取值范围	缺省值
collation_server	DDM服务端字符序。	utf8mb4_unicode_ci、utf8mb4_bin、utf8mb4_general_ci	utf8mb4_unicode_ci
concurrent_execution_level	逻辑表扫描时的分片并发执行级别: ● DATA_NODE：分库间并行扫描，同一分库内各分片串行扫描。 ● RDS_INSTANCE：RDS实例间并行扫描，同一RDS实例内各分片串行扫描。 ● PHY_TABLE：各物理分片全部并行扫描。	RDS_INSTANCE、DATA_NODE、PHY_TABLE	DATA_NODE
connection_idle_timeout	服务器关闭连接之前等待连接活动的秒数，以秒为单位，默认值28800，表示服务器关闭连接之前等待28800秒后，关闭连接。	60~86400	28800
contains_shard_key	是否强制SELECT，UPDATE，DELETE语句中过滤条件中包含拆分字段。	OFF、ON	OFF
ddl_precheck_mdl_threshold_time	DDL预检查中MDL锁持有时长阈值， 以秒为单位，默认值120。	1~3600	120
enable_table_recycle	● ON：开启表回收站。 ● OFF：关闭表回收站。 开启表回收站之后，表删除会进入回收站，7天内可以通过RESTORE命令进行恢复。	OFF、ON	OFF
long_query_time	记录慢查询的最小秒数，以秒为单位，默认值为1，表示如果SQL执行大于等于1秒就定义为慢SQL。	0.01~10	1
max_allowed_packet	服务端和客户端在一次传送数据包的过程中最大允许的数据包大小。该值必须设置为1024的倍数。	1024~1073741824	1073741824

参数名称	参数说明	取值范围	缺省值
max_backend_connections	允许每个DDM节点同时连接RDS的最大客户端总数。0为默认值标识符,实际值等于(RDS的最大连接数-20)/DDM节点数。此参数生效需要RDS上对最大连接数也进行相应的设置。	0~10000000	0
max_connections	允许每个实例节点同时连接的客户端总数。 此参数需要结合数据节点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个数据节点连接；SQL不带拆分键时，假设分片个数为N，那么会消耗N个数据节点连接。 因此，SQL合理设计且DDM和数据节点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“后端数据节点的数量 * 单个数据节点支持的最大连接数”。 建议根据自己的业务进行实际压测，配置合理的值。	10~40000	20000
min_backend_connections	允许每个DDM节点同时连接RDS的最小客户端总数。默认值为10。	0~10000000	10
seconds_behind_master	主从RDS节点延迟时间阈值，以秒为单位，默认值为30，表示主RDS与从RDS之间的数据同步时间值不能超过30秒，如果超过30s，读数据指令就不走当前读节点。	0~7200	30
sql_execute_timeout	SQL执行超时秒数，以秒为单位，默认值28800，表示SQL执行大于等于28800秒超时。 建议将DN实例中的net_write_timeout参数值设置为大于sql_execute_timeout的数值。	100~28800	28800

参数名称	参数说明	取值范围	缺省值
temp_table_size_limit	临时表大小。	500000~2000000000	1000000
transfer_hash_to_mod_hash	创表时是否将hash算法转为mod_hash算法。	OFF、ON	OFF
ultimate_optimize	是否需要根据参数值来优化SQL执行计划。	OFF、ON	ON
force_read_master_in_transaction	事务中涉及的SQL语句是否强制从主节点读取。 **注意** 3.0.9版本开始支持此参数。如果此开关被打开后，版本降级为3.0.9以下，之后再升级为3.0.9或以上时，此开关配置保持打开状态。	OFF、ON	OFF
ddl_flowcontrol_threshold	允许单位时间（1小时）DDL执行次数(包括执行正常和失败)。	1000~100000	10000

9.2 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**。

如果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。
- 每个用户最多可以创建100个DDM参数模板。

- 同一项目下的所有DDM实例共享参数模板配额。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“参数模板”页面，单击“创建参数模板”。

步骤3 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。

图 9-1 创建参数模板

----结束

9.3 修改自定义参数模板

在使用DDM实例过程中您可以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。
- 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 登录分布式数据库中间件控制台。

步骤2 单击左侧导航栏“参数模板”，在“自定义”页签下单击参数模板名称。

步骤3 在“参数详情”页签下，根据需要修改相关参数值，相关参数说明请参见**实例参数说明**。

图 9-2 编辑参数模板

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。
- 单击“预览”，可对比参数修改前和修改后的值。

步骤4 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。

- 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。
- 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。
- 参数模板修改后，某些参数会立即应用到当前使用实例中，请谨慎操作。

----结束

9.4 比较参数模板

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。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 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”。

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3 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。

图 9-3 选择并比较参数模板

- 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。

图 9-4 对比参数模板

----结束

9.5 查看参数修改历史

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。

使用须知

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 登录分布式数据库中间件控制台。

步骤2 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”。

您可查看一定时间范围内（小于等于2年）的模板历史记录，默认查询7天内的模板修改历史。

图 9-5 模板修改历史

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

9.6 复制参数模板

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 登录分布式数据库中间件控制台。

步骤2 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3 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

9.7 应用参数模板

参数模板创建成功或者根据业务需求调整完参数之后，需要将参数模板应用到DDM实例中。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“参数模板”页面，根据参数模板类型不同进行如下操作。

- 如果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”。
- 如果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3 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

9.8 查看参数模板应用记录

操作场景

参数模板应用到DDM实例之后，您可以查看参数模板的应用记录。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 单击“参数模板”。

步骤3 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。

图 9-6 查看应用记录

----结束

9.9 修改参数模板描述

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。

使用须知

默认参数模板的描述不可修改。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 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3 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

9.10 删除参数模板

您可删除不再需要的参数模板。

使用须知

- 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 登录分布式数据库中间件控制台。

步骤2 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3 单击“是”，删除参数模板。

----结束

10 账号管理

10.1 管理员账户

DDM提供管理员账户功能，此账户拥有最高的superuser权限，可对“账号管理”里的DDM账号进行权限修改，且默认具备所有库表(包括还未创建的逻辑库)的读写权限，一经创建无法删除。

您可以在创建实例时设置管理员账户，未设置管理员账户的实例可以在实例详情中通过新增管理员账户的方式设置管理员账户。

操作场景

- 在使用DDM过程中，如果忘记管理员账号的密码，可以重新设置密码，请参考**重新设置密码操作步骤**。
- 创建DDM实例时，管理员账户的设置选择了“创建后设置”，可通过新增管理员账号来设置，请参考**新增管理员账号操作步骤**。

前提条件

DDM内核版本大于等于3.0.9版本。

注意事项

- 管理员账户创建成功后，不支持修改账户名。
- 管理员账户与“账号管理”里的DDM账号不能重复。如果重复将无法创建。
- 管控修改后，会把客户之前的权限全部清除， 按照管控侧提供的权限重新赋权。

重新设置密码操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 在“实例信息”模块的“管理员账户”处，单击“重置密码”。

步骤4 在“重置密码”弹框，输入新管理员密码及确认密码。

步骤5 单击“确定”，新增管理员账户命令下发成功提示。

图 10-1 重置密码弹框

----结束

新增管理员账号操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 在“实例信息”模块的“管理员账户”处，单击“新增账号”。

步骤4 在“新增账号”弹框，输入管理员账户、密码及确认密码。

步骤5 单击“是”，新增管理员账户命令下发成功提示。

----结束

10.2 创建账号

当同一数据库实例或同一数据库需要不同权限的用户访问时，可创建多个用户。您可以在分布式数据库中间件控制台创建所需的用户账号。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤4 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 10-2 创建 DDM 账号

表 10-1 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，不区分大小写，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。 • 大小写敏感。 • 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 ~ ! @ # % ^ * - _ + ? • 不能使用简单、强度不够、容易被猜测的密码。 • 不能与账号名称或者倒序的账号名称相同。
确认密码	确认密码必须和输入的密码保持一致。

参数	说明
密码有效期	设置密码有效期，取值范围为0-65535的整数，单位为天。如果DDM的账号状态为“已过期”，账号将无法登录，已有连接会断开，需要重置密码后重新登录。 ● 0表示密码永不过期。 ● 如果不设置密码有效期则默认密码永不过期。 **说明** 内核版本需大于或者等于3.0.4。
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	选择需要的基础权限，包括CREATE、DROP、ALTER、INDEX、INSERT、DELETE、UPDATE、SELECT。
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5 确认填写无误后，单击“确定”。

----结束

10.3 修改账号信息

您可以在DDM控制台修改账号信息。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤4 账号管理页面选择目标账号，在其操作栏单击“修改”。

步骤5 在修改账号弹窗中，您可以修改密码有效期、关联逻辑库、账号权限及描述信息。

密码有效期，取值范围为0-65535的整数，单位为天。0表示密码永不过期。

图 10-3 修改 DDM 账号

步骤6 编辑完成账号信息后，单击“确定”，保存修改信息。

----结束

10.4 删除账号

您可以删除不再需要的账号。

使用须知

删除操作无法恢复，请谨慎操作。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤4 账号管理页面选择目标账号，在其操作栏单击“更多 > 删除”。

步骤5 在删除账号确认弹窗中，单击“是”，删除账号信息。

----结束

10.5 重置密码

您可以在DDM控制台重置账号密码。

使用须知

- 重置DDM账号密码属于高风险操作，需确保账号具备IAM中“修改DDM账号”的权限。
- DDM的账号状态为“已过期”时，账号将无法登录，已有连接会断开，需要重置密码后重新登录。

图 10-4 账号已过期

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤3 在左侧导航栏选择“账号管理”，进入账号管理页面。

步骤4 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 10-5 重置 DDM 账号密码

步骤5 在输入新密码并再次输入“确认密码”后，单击“确定”，进行密码重置。

----结束

10.6 账号权限

10.6.1 账号权限介绍

DDM的权限管理系统参考MySQL的权限管理进行实现，DDM支持大部分的MySQL的语法和权限类型。MySQL账号和权限系统的详细信息请参见**MySQL官方文档**。

本文档主要介绍DDM账号规则、权限级别、权限项、以及权限操作。

说明

DDM里通过CREATE USER或GRANT语句创建出来的账号属于DDM的账号体系，与DDM关联的RDS没有任何关系，也不会同步到RDS中去。

10.6.2 账号规则

账号

与MySQL不同，DDM只通过用户名确定一个账号，而不是通过'username'@'host'方式来确定一个账号。

用户名规则

- 大小写敏感。
- 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。

密码规则

- 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~!@#%^*-_+?
- 不能使用简单、强度不够、容易被猜测的弱口令。

10.6.3 权限管理

权限级别支持情况

- 用户层级（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）
- 全局层级（暂不支持）

权限项

DDM通过GRANT语句授权所支持的权限项如下：

权限类型	权限简述
ALL	所有权限
Drop	删除table
Index	创建/删除index
Alter	执行ALTER语句
Create	创建table
Select	读取表内容
Insert	插入数据到表
Update	更新表中数据
Grant	授予用户权限
Revoke	删除用户权限
Set	SET用户密码权限
File	从文件加载数据库权限
Create User	创建用户

注意事项

- DDM账户的基础权限base_authority只能通过控制台界面来修改。
- DDM账户拥有逻辑库的任意表级权限或者库级权限，控制台界面就会展示已关联该逻辑库。
- Create user只支持用户级权限。
- 删除逻辑库，删除表不会影响DDM账户的grant info权限信息。
- 用grant语法授权给DDM账户，命令如下所示：
  ```
  grant grant option on {用户级、库级别、表级别} to DDM账户
  ```
- 创建的DDM账户必须关联逻辑库才会给该账户赋权。

权限操作

须知

除show grants命令需要3.0.2及以上版本外，其他功能需要2.4.1.4及以上版本。

创建账号（CREATE USER）语句

语法规则：

```
CREATE USER username IDENTIFIED BY 'auth#string'
```

示例：**创建一个名为Jenny，密码为xxxxxx。**

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

说明

用户名和密码需要满足对应规则。

删除账号（DROP USER）语句

语法规则：

```
DROP USER username
```

示例：**移除账号**Jenny**。**

```
DROP USER Jenny;
```

修改账号密码（SET PASSWORD）语句

语法规则：

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

说明

为了兼容MySQL语法用户需要统一写成'username'@'%'格式。

示例：**修改账号**Jenny**的密码为xxxxxx**

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

授权权限（GRANT）语句

语法规则：

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

说明

GRANT 语句里面的账号如果不存在，同时又没有提供 IDENTIFIED BY 信息，则报账号不存在异常；如果提供了 IDENTIFIED BY 信息，则会创建该账号同时授权。

当前支持使用GRANT ALL [PRIVILEGES]语法授权表级、用户级以及库级所有权限。

示例1：创建一个**用户级**所有权限的账号。用户名为Mike。

方法1：先创建账号，再授权。

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

方法2：一条语句完成创建账号和授权两个操作。

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

示例2：创建一个**数据库级**所有权限的账号。在数据库 testdb 下面，创建一个用户名为 david，具有 testdb 数据库SELECT权限的账号。

方法1：先创建账号，再授权。


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

方法2：一条语句完成创建账号和授权两个操作。

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

示例3：创建一个**表级别**所有权限的账号。在数据库 testdb 下面，创建一个用户名为 hanson，具有 testdb.employees 表所有权限的账号。

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

回收权限（REVOKE）语句

语法规则：

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

示例：删除 hanson在 testdb.emp 表的 CREATE、DROP、INDEX 权限。

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

说明

删除账号在某个权限级别下的权限项，具体权限级别由 priv_level 指定。

查看授权（SHOW GRANTS）语句

语法规则：

```
SHOW GRANTS FOR user;
```

示例1：查看当前用户权限可以使用以下三种方式之一：

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

示例2：查看其他用户权限，只有当前用户具有**用户级GRANT权限**时才可使用此操作。

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

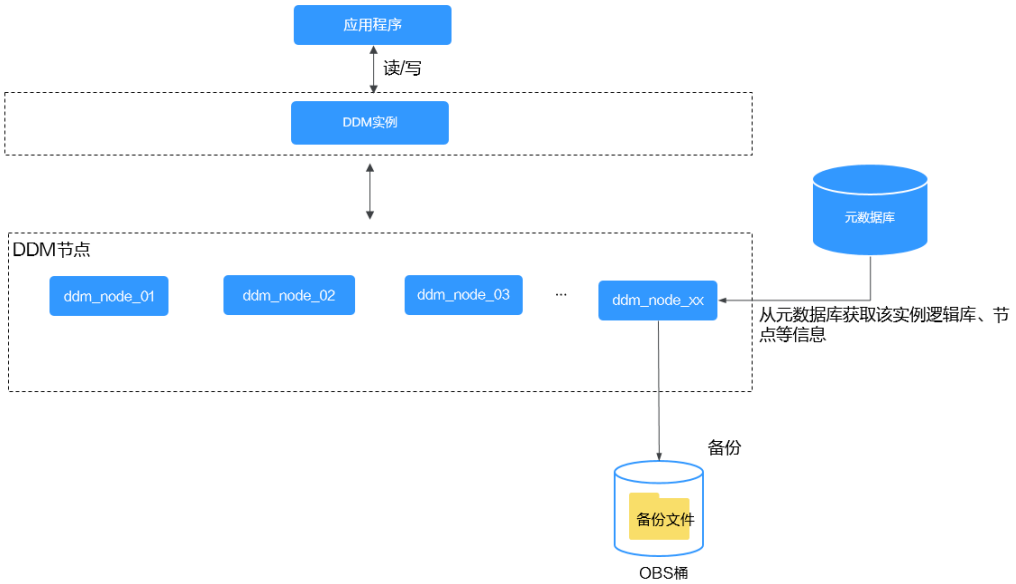
11 备份恢复

11.1 备份原理

DDM实例暂不支持客户手动备份，实例将在每日凌晨2点至3点自动备份，删除逻辑库、逻辑库分片变更后清理数据、删除实例等影响Metadata的重要操作也会触发元数据备份。

备份原理如图11-1所示。

图 11-1 备份原理



说明

元数据库是用来存放DDM实例信息以及下挂的数据节点信息，各区域的所有DDM实例共用一个元数据库。

11.2 一致性备份说明

DDM已在2022年2月底对一致性备份进行功能优化调整，将该功能调整成恢复数据中的“Metadata恢复”。在各个区域陆续进行，调整后一致性备份页签将会隐藏，已经创建的一致性备份将不可再用于恢复。

11.3 恢复到新实例

DDM支持基于已有备份集将实例恢复至任意时间点，适用于日常业务常规备份恢复场景。

本章节以RDS for MySQL实例举例说明。

使用须知

- 恢复到新实例主要是指DDM实例和数据节点的整体恢复，包含了数据节点（RDS for MySQL）的恢复流程，需要预置新的DDM和RDS for MySQL实例。
- 恢复到目标DDM实例会导致实例数据被覆盖，恢复过程中目标实例数据库不可用。
- 恢复的目标RDS for MySQL实例版本需要大于等于源实例版本，且存储空间大于等于原实例。
- 此恢复功能暂不支持备份恢复到RDS for MySQL本地SSD盘实例。
- 暂不支持目标DDM实例在主网段、RDS for MySQL实例在扩展网段的场景。
- 源DDM实例版本需大于等于2.3.2.11版本，目标DDM实例版本需大于等于3.0.8版本。
- 所能够恢复到的时间点依赖您在原DN实例上设置的备份策略。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在源DDM实例所在区域创建一个新的DDM实例或者寻找一个满足使用条件的DDM实例。

说明

新建的DDM实例或者满足使用条件的已有实例均不能挂载RDS for MySQL实例，不能创建逻辑库和账号。

步骤3 在云数据库 RDS控制台，创建与源DDM实例下相同数量的RDS for MySQL实例。

说明

- 新创建的RDS for MySQL实例版本不得低于源DDM下RDS for MySQL实例版本号。
- 每个实例存储空间不得小于源DDM实例下的RDS for MySQL的存储空间。

步骤4 返回分布式数据库中间件控制台，在DDM实例列表页面单击需要恢复的实例名称，进入实例基本信息页面。

步骤5 在左侧导航栏选择“备份恢复”，进入恢复数据页面。

步骤6 单击“恢复新实例”按钮。

图 11-2 基本信息页

步骤7 在恢复新实例页签中，设置可恢复时间段、可恢复时间点、目标DDM和目标DN实例。

图 11-3 设置恢复信息

表 11-1 参数说明

参数名称	说明
可恢复时间段	选择恢复时间段。
可恢复时间点	选择恢复时间点。 DDM会检测与DDM实例关联的DN实例在选中的可恢复时间点上是否存在备份。
目标DDM	选择**步骤2**中创建的DDM实例作为目标实例。
DN实例映射	选择**步骤3**中创建的RDS for MySQL实例作为目标DN实例。

步骤8 勾选确认信息复选框，单击页面右下角“确定”，等待1-3分钟数据恢复完成。

----结束

11.4 Metadata 恢复

关于Metadata的备份策略，用户无需手动创建备份，DDM将在每日凌晨2点至3点对实例的Metadata数据进行备份，备份保留时长30天。删除逻辑库、逻辑库分片变更后清理数据、删除实例等影响Metadata的重要操作也会触发元数据备份。

元数据恢复特性用于误删库或者RDS for MySQL本身出现异常等业务场景，可根据过去的某个时间点，将Metadata数据与已经PITR恢复完成的RDS for MySQL实例进行匹

配，重建DDM和RDS for MySQL的关联关系，恢复DDM。当前该特性仅支持RDS for MySQL引擎。

Metadata恢复支持客户自己选择时间点进行恢复，详细请参考**根据自定义时间点进行Metadata恢复**。也支持客户根据已经完成的备份进行恢复，详细请参考**根据备份文件进行Metadata恢复**。

使用须知

- Metadata恢复主要侧重于元数据恢复，是基于已经PITR恢复完成的数据节点（RDS for MySQL）进行的DDM恢复，只需预置新的DDM实例。

  说明

  PITR是指已经将数据节点恢复到指定时间点。

- 目标DDM实例不能关联RDS for MySQL实例，不能创建逻辑库和账号。
- 暂不支持目标DDM实例在主网段、RDS for MySQL实例在扩展网段的场景。
- 源DDM实例版本需大于等于2.3.2.11版本，目标DDM实例版本需大于等于3.0.8版本。
- 所能够恢复到的时间点依赖您在源DDM实例的数据节点上设置的备份策略。

前提条件

- 已购买源DDM实例，并已关联RDS for MySQL实例，该RDS for MySQL实例为源数据节点。
- 源数据节点已经按指定时间点恢复到目标数据节点。

根据自定义时间点进行 Metadata 恢复

步骤1 登录分布式数据库中间件控制台。

步骤2 购买新的DDM实例，将新购买的DDM实例作为目标DDM实例。DDM购买请参考**实例购买**。

步骤3 在DDM实例列表页面单击源DDM实例名称，进入实例基本信息页面。

步骤4 在左侧导航栏选择“备份恢复”，进入恢复数据页面。

步骤5 单击“Metadata恢复”按钮。

步骤6 在“Metadata恢复”页面，设置选择恢复到的时间点，源DDM将在该时间点就近选择合适的DDM元数据备份集。

图 11-4 恢复信息

表 11-2 参数说明

参数名称	说明
恢复时间点	选择恢复时间点，系统将根据恢复时间点就近筛选Metadata备份。
目标DDM	选择**步骤2**中新创建的DDM实例。
目标数据节点	选择已经完成PITR的RDS for MySQL实例。系统将根据您选择的数据节点与筛选的Metadata备份上的分片信息进行匹配，如果匹配成功将进行Metadata数据重建。

步骤7 单击“确定”，等待出现Metadata恢复成功提示，即表示恢复完成。

----结束

根据备份文件进行 Metadata 恢复

步骤1 登录分布式数据库中间件控制台。

步骤2 购买新的DDM实例，将新购买的DDM实例作为目标DDM实例。DDM购买请参考**实例购买**。

步骤3 在左侧导航栏选择“备份管理”，进入备份管理页面。

步骤4 根据实例名称、备份开始/结束时间选择需要恢复的备份，单击操作栏的“恢复”。

图 11-5 Metadata 恢复

步骤5 在Metadata恢复页面，设置备份恢复信息。

图 11-6 恢复信息

表 11-3 参数说明

参数名称	说明
备份名称	需要恢复的备份名称。
目标DDM	选择**步骤2**中新创建的DDM实例。
目标数据节点	选择已经完成PITR的RDS for MySQL实例。系统将根据您选择的数据节点与筛选的Metadata备份上的分片信息进行匹配，如果匹配成功将进行Metadata数据重建。

步骤6 单击“确定”，等待出现Metadata恢复成功提示，即表示恢复完成。

----结束

12 数据迁移

12.1 迁移介绍

数据迁移指将原有数据库中的数据迁移到DDM服务中，或因为业务使用需要，将DDM服务的数据导出到其他数据库系统中使用。您可以使用MySQL官方工具mysqldump进行数据全量导出，当您需要全量迁移+增量迁移时，建议您使用数据复制服务（Data Replication Service，简称DRS）服务。

迁移须知

- 迁移过程中可能会出现业务中断情况，中断时长与迁移数据量大小、网络情况相关。
- 数据迁移是一项比较复杂的操作，建议在业务量较低时进行。本指南仅供参考，您需要根据自己业务场景、数据量、停机时间要求等情况，设计合适的迁移方案。
- 对于数据表和数据量较大的场景，建议您在管理控制台右上角，提交工单或售后服务联系DDM客服进行支撑，在正式数据迁移前进行充分的迁移演练测试。
- 由于DDM仅支持通过弹性云服务器（ECS）访问，因此需要先将数据库导出为文件并上传到ECS，然后从ECS将文件中的数据导入到DDM。

迁移方式

DDM支持如下两种方式进行数据迁移。

- 基于MySQL官方客户端工具，此部分以RDS for MySQL 为例重点介绍该迁移方式。
- 基于数据复制服务DRS。

迁移场景

数据迁移主要有以下场景：

1. **场景一：数据中心自建MySQL迁移到DDM**
2. **场景二：其他云MySQL迁移到DDM**
3. **场景三：华为云上自建MySQL迁移到DDM**

4. **场景四：从DDM实例导出数据**
5. **场景五：其他异构数据库迁移到DDM**
6. **场景六：从华为云RDS for MySQL迁移到DDM**

12.2 迁移评估

数据库迁移上云前需要进行充分地评估和验证。

根据待迁移数据的现状和未来业务规模，分类进行评估和准备，详细信息如**表12-1**所示。

表 12-1 迁移前评估和准备

评估项	说明
迁移数据量与DDM、RDS的实例规格	• 关于源数据库数据拆分，建议采用先垂直拆分，后水平拆分的方式。 • 对于源MySQL实例数据库表占用存储空间，可执行如下SQL语句评估。 `select concat(round(sum(DATA_LENGTH/ 1024/1024),2),'MB')as data from information_schema.TABLES;` • 建议对数据行数超过1000万行（或者预计超过1000万行）的表进行分片，设计为拆分表。 • 单RDS实例存储不超过500GB。
源数据库每张表对应逻辑库和逻辑表信息	• 以表为单位，细化每张源表映射对应逻辑表信息，如记录数、逻辑表类型、所属逻辑库、所属DDM实例和关联RDS实例等。 • 如果已经有RDS（MySQL）且DDM逻辑库选择非拆分库，则只需要在创建逻辑库时关联该RDS（MySQL）实例即可，不涉及表结构和表数据的迁移。
兼容性	• 检查源数据库和目标实例的MySQL版本号一致。 • 目标实例的实例规格和存储空间等原则上不低于源数据库。 • 检查源数据库和目标实例的表结构和字符集等一致。 • “拆分算法”为“hash”的逻辑表单次迁移记录数不超过1000万条；“拆分算法”为“range”的逻辑表单次迁移记录数不超过500万条。 • 对于数量过大的表可以采用分批导出导入，通过mysqldump指定参数where条件来限定每批的记录数。 • 导入DDM只支持导入的SQL文本文件含有标准的DML插入语句。 • 评估应用程序SQL语句在DDM中的兼容性。

迁移前需要收集相关信息，帮助您更好的完成迁移，详细信息如**表12-2**所示。

表 12-2 迁移前信息收集

迁移源/目标	信息项
源RDS实例	RDS实例连接地址
	RDS实例侦听端口
	数据库用户
	数据库名称
	数据库表名
目标DDM实例	DDM实例连接地址
	DDM实例侦听端口
	DDM实例用户名
	DDM关联RDS实例上创建的数据库名称
	新建RDS实例连接地址
	新建RDS实例侦听端口
	RDS实例用户名
	数据库名称
弹性云服务器	弹性IP地址
	系统登录用户/密码

12.3 场景一：数据中心自建 MySQL 迁移到 DDM

场景介绍

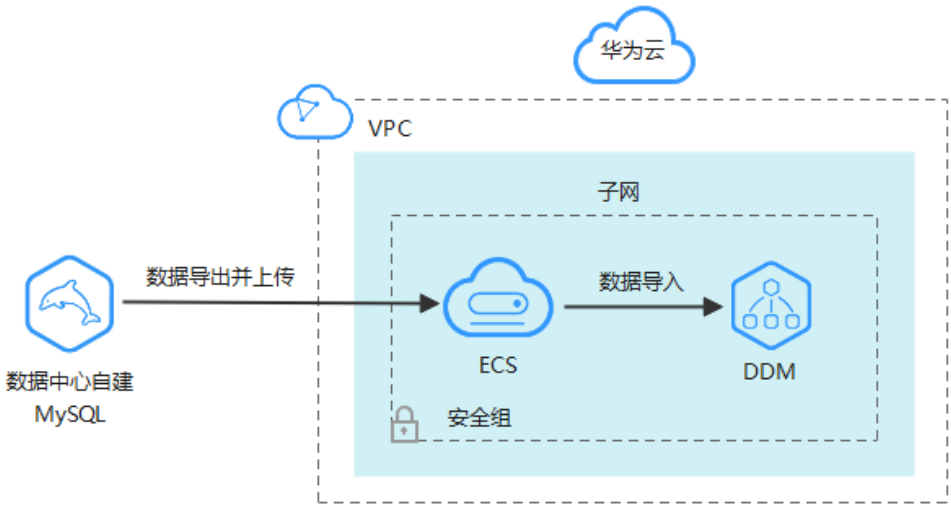
企业当前使用数据中心自建MySQL，希望能使用DDM将数据进行分布式存储。

须知

迁移过程中可能会出现业务中断情况，中断时长与迁移数据量大小、网络情况相关。

迁移示意图

图 12-1 数据中心自建 MySQL 迁移到 DDM 示意图



约束限制

- 目标DDM实例、RDS for MySQL实例所在ECS必须保证网络互通。
- 为了保持数据完整性，需要先停止业务后再进行数据迁移。
- DDM不支持以自动新建库或者新建拆分表、广播表的方式导入数据。因此导入数据前需要先创建好相同名称的逻辑库，相同拆分表、广播表结构的逻辑表，然后再进行数据导入。各类逻辑表创建方式请参见**表12-4**。
- 目标DDM使用的RDS for MySQL实例与自建MySQL的MySQL版本需要保持一致。

迁移前准备

- 准备可以访问自建MySQL所在数据中心的ECS。
  a. 确保自建MySQL所在数据中心和目标DDM实例、RDS for MySQL实例都与ECS网络互通。
  b. ECS已安装MySQL官方客户端，MySQL客户端版本建议为5.6或5.7。
     - Redhat系列Linux安装命令：**yum install mysql mysql-devel**
     - Debian系列Linux安装命令：**apt install mysql-client-5.7 mysql-client-core-5.7**
  c. ECS磁盘空间足够存放临时转储文件；ECS内存空间足够，可以用来比较转储文件。
- 准备DDM实例，并配置DDM账号、DDM逻辑库、DDM逻辑表等相关信息。
  a. 申请DDM实例，并在DDM控制台创建DDM账号、创建逻辑库。
  b. 导出自建MySQL数据表结构至SQL文本文件。
     - MySQL客户端版本为5.6和5.7时请执行以下命令：

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

- MySQL客户端版本为8.0时请执行以下命令：

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

相关参数解释如表12-3所示。

表 12-3 参数解释

参数	说明	备注
DB_ADDRESS	待导出数据的数据库连接地址。	必填
DB_PORT	数据库侦听端口	必填
DB_USER	数据库用户	必填
--skip-lock-tables	在不锁表的情况下导出数据。	某些参数会默认开启加锁声明，因此建议在数据导出语句末尾增加此参数。
--add-locks=false	导出的数据文件中不加锁表的声明。	-
--no-data	不导出任何数据，只导出数据库表结构。	导出表结构时使用。
--column-statistics=0	如果使用的MySQL客户端版本为8.0，则必须关闭该特性。	MySQL客户端版本为8.0时必填。
DB_NAME	数据库名称	必填
TABLE_NAME	表名	可以多个同类型的表，用空格隔开。建议只导出与业务相关的表结构
mysql_table_schema.sql	生成的表结构文件名。	每次导出表结构时文件名不同。 建议以“逻辑库名”+“_”+“逻辑表名”+“_”+“schema”格式命名，以免数据被覆盖。如mysql_table_schema.sql。

说明

此处举例的参数为数据导出中常用的参数，由于mysqldump参数无法逐一列举，如果存在个别参数调优等特殊情况，请在MySQL官网查询。

c. 创建逻辑表。

创建逻辑表结构请与**b**中导出的表结构保持一致，把源表映射到目标DDM实例逻辑表，明确对应表结构和表数据的迁移策略，如**表12-4**所示。

说明

创建前可先通过SQL语句：show create table **{TABLE_NAME}**查看自建MySQL中数据表结构。

表 12-4 表迁移策略

逻辑库类型	逻辑表类型	表结构迁移策略	表数据迁移策略
拆分	拆分表	在原表结构基础上增加拆分键声明（详见建表语句说明文档），形成适用于DDM的建表语句，连接DDM并登录对应逻辑库后执行。	通过DDM导入原表数据。
拆分	广播表	在原表结构基础上增加广播声明（详见建表语句说明文档），形成适用于DDM的建表语句，连接DDM并登录对应逻辑库后执行。	
拆分	单表	获取原表建表语句，连接DDM并登录对应逻辑库后执行。	
非拆分	单表		

d. 清理目标DDM实例的测试数据，防止和待迁移数据冲突。

- 准备RDS for MySQL实例。

导出数据

此处以本地IP连接的方式介绍，通过使用mysqldump工具来导出数据。

在进行数据导出操作之前，您需要做到以下两点：

- 保障DDM实例的子网及虚拟私有云与客户端ECS实例保持一致。
- 放通安全组入方向规则。

从数据中心自建MySQL中将表数据导出到单独的SQL文本文件中，然后上传至ECS。

步骤1 停止自建MySQL的业务系统，否则可能会导致导出数据不是最新的。

步骤2 打开MySQL客户端，输入如下命令，连接自建MySQL，并导出自建MySQL表数据至SQL文本文件。

- MySQL客户端版本为5.6和5.7时请执行以下命令：
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**
- MySQL客户端版本为8.0时请执行以下命令：
  mysqldump -h {DB_ADDRESS} -P {DB_PORT} -u {DB_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data.sql}

说明

如果自建MySQL中有多个逻辑库，建议分开多次执行该命令导出表数据。

相关参数解释如表12-5所示。

表 12-5 参数解释

参数	说明	备注
DB_ADDRESS	待导出数据的数据库连接地址。	必填
DB_PORT	数据库侦听端口	必填
DB_USER	数据库用户	必填
--complete-insert	使用完整的insert语句(包含列名称)。	-
--single-transaction	该选项在导出数据之前提交一个BEGIN SQL语句，BEGIN 不会阻塞任何应用程序且能保证导出时数据库的一致性状态。它只适用于多版本存储引擎，仅InnoDB。	-
--quick	不缓冲查询，直接导出到标准输出。	避免大数据情况内存爆涨。
--hex-blob	使用十六进制格式导出二进制字符串字段。如果有二进制数据就必须使用该选项。	-
--no-create-info	只导出数据，而不添加CREATE TABLE 语句。	导出数据时使用。
--skip-comments	关闭附加注释信息。	-
--add-locks=false	导出的数据文件中不加锁表的声明。	-
--set-gtid-purged=OFF	如果使用的MySQL版本为8.0或5.7，则需要配置该参数。	如果使用的MySQL版本为5.6或低于5.6，则不需要配置该参数。

参数	说明	备注
--skip-add-locks	在导出数据时，控制加锁动作，以避免因耗能引起的性能问题。	-
--where	只转储给定的WHERE条件选择的记录。	如果条件包含命令解释符专用空格或字符，一定要将条件引用起来。
DB_NAME	数据库名称	必填
TABLE_NAME	表名	可以多个同类型的表，用空格隔开。 建议只导出与业务相关的表结构。
mysql_table_data.sql	生成的表数据文件名。	每次导出不同表时文件名不同。 建议以“逻辑库名”+“_”+“逻辑表名”+“_”+“data”格式命名，以免数据被覆盖。如mysql_table_data.sql。

说明

- 此处举例的参数为数据导出中常用的参数，由于mysqldump参数无法逐一列举，如果存在个别参数调优等特殊情况，请在MySQL官网查询。
- 使用mysqldump工具进行转移MySQL数据时，请保持MySQL客户端版本和DDM所支持的MySQL版本一致。如果版本不一致，可能会影响数据导出性能。

步骤3 查看导出SQL文本文件的大小，验证导出数据是否成功。

- 如大小不为0字节，说明导出成功。
- 如大小为0字节，说明导出失败，请联系DDM客服人员。

步骤4 将导出的SQL数据文件上传至已准备的ECS。

----结束

导入数据

步骤1 开启应用程序访问DDM数据库只读开关。

步骤2 清理目标DDM实例的测试数据，防止和待迁移数据冲突。

步骤3 如果是单表或普通表，采用MySQL客户端直连DDM实例，直接执行以下命令导入表结构文本文件和数据文件。

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



```
{mysql_table_data.sql}
Enter password: **********
```

- DDM_ADDRESS为待导入数据的DDM实例的地址。
- DDM_PORT为DDM实例的端口。
- DDM_USER为DDM实例的用户名。
- DB_NAME为DDM逻辑库名称，如果导入的是单表，DB_NAME为DDM实例第一个分片的物理数据库。
- mysql_table_schema.sql为待导入的表结构文件名。
- mysql_table_data.sql为待导入的表数据文件名。

  说明

  单表或普通表导入前，需要编辑表结构文本文件，将最后一行信息删除（ Dump completed on 2018-06-28 19:53:03 ），否则可能导致无法导入。

步骤4 如果是拆分表或广播表，采用MySQL客户端连接DDM将数据文件导入。

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

- DDM_ADDRESS为待导入数据的DDM的地址。
- DDM_PORT为DDM侦听端口。
- DDM_USER为DDM用户。
- DB_NAME为DDM逻辑库名称。
- mysql_table_data.sql为待导入的表数据文件名。

须知

- 数据导入阶段会在一定程度上影响DDM实例以及RDS for MySQL实例性能，请选择在业务低峰时间导入。
- 如果导入过程中出现中断或异常，为防止表数据主键冲突可以用SQL语句 **truncate table *{TABLE_NAME}*** 清空再重新导入。该命令属于高危操作，执行后会清空表中所有数据，请谨慎使用。
- 请勿把数据量大（超过500万）的数据导入到广播表。

----结束

数据验证

步骤1 在ECS上对DDM实例进行逻辑备份。

- 导出表结构：
  - MySQL客户端版本为5.6时请执行以下命令：

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

  - MySQL客户端版本为8.0时请执行以下命令：

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


- 导出表数据：
  - MySQL客户端版本为5.6时请执行以下命令：
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysq_table_data_new.sql}
    ```
  - MySQL客户端版本为8.0时请执行以下命令：
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME} {TABLE_NAME} > {mysq_table_data_new.sql}
    ```

步骤2 检查数据一致性。

1. 在自建MySQL和DDM实例执行如下SQL语句检查每张表的记录数是否相等。其中TABLE_NAME是表名。
   ```
   select count(*) from {TABLE_NAME};
   ```
2. 在ECS上对导出前后的表结构和表数据进行比较。
   ```
   diff -B -w -q -i {mysql_table_schema.sql} {mysql_table_schema_new.sql};echo $?
   diff -B -w -q -i {mysql_table_data.sql} {mysql_table_data_new.sql};echo $?
   ```

   说明

   - 表结构的导出命令只适用于单表和普通表。
   - 如果导出数据的顺序不一致，无法比较。
   - 如果导入前后相同，则表示数据迁移成功。
   - 如果数据存在差异，建议联系DDM客服人员进行定位。

步骤3 端到端验证应用程序通过DDM实例访问相关表只读功能是否正常。

步骤4 关闭应用程序访问DDM数据库只读开关。

----结束

业务验证

1. 切换业务数据源至DDM。
2. 验证是否能正常读取、写入数据。
   - 正常：完成迁移。
   - 异常：切换业务数据源至自建MySQL，联系DDM管理人员进行定位。

12.4 场景二：其他云 MySQL 迁移到 DDM

场景介绍

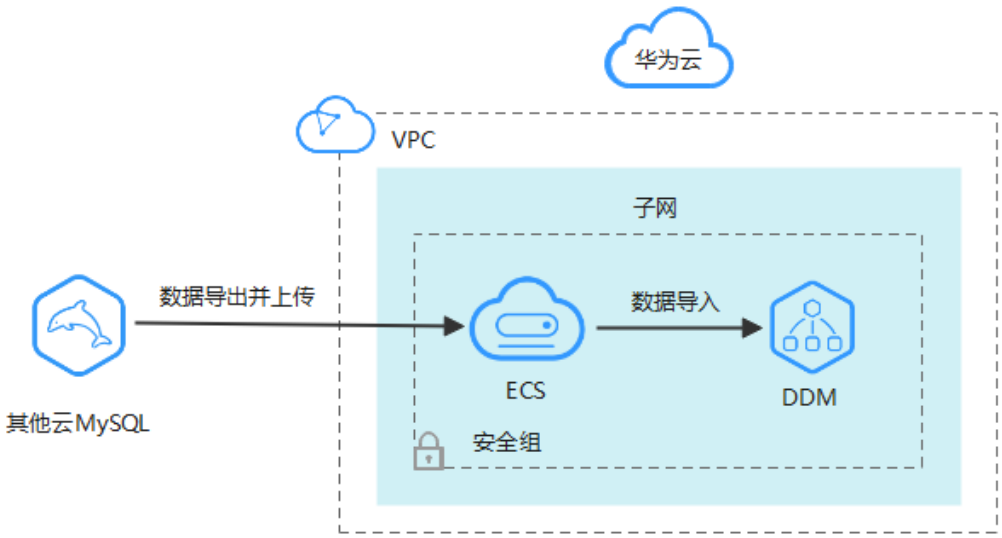
企业当前使用其他云MySQL实例，希望能使用华为云DDM将数据进行分布式存储。

说明

迁移过程中可能会出现业务中断情况，中断时长与迁移数据量大小、网络情况相关。

迁移示意

图 12-2 其他云 MySQL 迁移到 DDM 示意图



约束限制

- 目标DDM实例、RDS for MySQL实例所在ECS必须保证网络互通。
- 为了保持数据完整性，需要先停止业务后再进行数据迁移。
- DDM不支持以自动新建库或者新建拆分表、广播表的方式导入数据。因此导入数据前需要先创建好相同名称的逻辑库，相同拆分表、广播表结构的逻辑表，然后再进行数据导入。各类逻辑表创建方式请参见**表12-7**。
- 目标DDM使用的RDS for MySQL与其他云的MySQL版本需要保持一致。

迁移前准备

- 准备可以访问其他云MySQL实例的ECS。
  a. 确保其他云MySQL实例和目标DDM实例、RDS for MySQL实例都与ECS网络互通。如果网络不通，数据导出后，通过其他中转服务器，将数据文件上传到华为云ECS。
  b. ECS已安装MySQL官方客户端，MySQL客户端版本建议为5.6或5.7。
     - Redhat系列Linux安装命令：**yum install mysql mysql-devel**
     - Debian系列Linux安装命令：**apt install mysql-client-5.7 mysql-client-core-5.7**
  c. ECS磁盘空间足够存放临时转储文件；ECS内存空间足够，可以用来比较转储文件。
- 准备DDM实例，并配置DDM账号、DDM逻辑库、DDM逻辑表等相关信息。
  a. 申请DDM实例，并在DDM控制台创建DDM账号、创建逻辑库。
  b. 导出其他云MySQL实例数据表结构至SQL文本文件。

- MySQL客户端版本为5.6和5.7时请执行以下命令：
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --no-data --skip-add-locks --add-locks=false --skip-tz-utc **{DB_NAME}** **{TABLE_NAME}** > **{mysql_table_schema.sql}**
- MySQL客户端版本为8.0时请执行以下命令：
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --no-data --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc **{DB_NAME}** **{TABLE_NAME}** > **{mysql_table_schema.sql}**

相关参数解释如表12-6所示。

表 12-6 参数解释

参数	说明	备注
DB_ADDRESS	待导出数据的数据库连接地址。	必填
DB_PORT	数据库侦听端口	必填
DB_USER	数据库用户	必填
--skip-lock-tables	在不锁表的情况下导出数据。	某些参数会默认开启加锁声明，因此建议在数据导出语句末尾增加此参数。
--add-locks=false	导出的数据文件中不加锁表的声明。	-
--no-data	不导出任何数据，只导出数据库表结构。	导出表结构时使用。
--column-statistics=0	如果使用的MySQL客户端版本为8.0，则必须关闭该特性。	MySQL客户端版本为8.0时必填。
DB_NAME	数据库名称	必填
TABLE_NAME	表名	可以多个同类型的表，用空格隔开。建议只导出与业务相关的表结构
mysql_table_schema.sql	生成的表结构文件名。	每次导出表结构时文件名不同。 建议以“逻辑库名”+“_”+“逻辑表名”+“_”+“schema”格式命名，以免数据被覆盖。如mysql_table_schema.sql。

说明

此处举例的参数为数据导出中常用的参数，由于mysqldump参数无法逐一列举，如果存在个别参数调优等特殊情况，请在MySQL官网查询。

c. 创建逻辑表。

创建逻辑表结构请与**b**中导出的表结构保持一致，把源表映射到目标DDM实例逻辑表，明确对应表结构和表数据的迁移策略，如**表12-7**所示。

说明

创建前可先通过SQL语句：show create table **{TABLE_NAME}**查看其他云RDS for MySQL实例中数据表结构。

表 12-7 表迁移策略

逻辑库类型	逻辑表类型	表结构迁移策略	表数据迁移策略
拆分	拆分表	在原表结构基础上增加拆分键声明（详见建表语句说明文档），形成适用于DDM的建表语句，连接DDM并登录对应逻辑库后执行。	通过DDM导入原表数据。
拆分	广播表	在原表结构基础上增加广播声明（详见建表语句说明文档），形成适用于DDM的建表语句，连接DDM并登录对应逻辑库后执行。	
拆分	单表	获取原表建表语句，连接DDM并登录对应逻辑库后执行。	
非拆分	单表		

d. 清理目标DDM实例的测试数据，防止和待迁移数据冲突。

- 准备RDS for MySQL实例。

导出数据

此处以本地IP连接的方式介绍，通过使用mysqldump工具来导出数据。

在进行数据导出操作之前，您需要做到以下两点：

- 保障DDM实例的子网及虚拟私有云与客户端ECS实例保持一致。
- 放通安全组入方向规则。

从其他云MySQL实例中将表数据导出到单独的SQL文本文件中，然后上传至ECS。

步骤1 停止自其他云MySQL实例的业务系统，否则可能会导致导出数据不是最新的。

步骤2 导出其他云MySQL实例表数据至SQL文本文件。

- MySQL客户端版本为5.6和5.7时请执行以下命令：
  m*ysqldump -h* **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**
- MySQL客户端版本为8.0时请执行以下命令：
  m*ysqldump -h* **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --column-statistics=0 --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**

说明

- 如果其他云MySQL实例中有多个逻辑库，建议分开多次执行该命令导出表数据。
- 如果其他云MySQL实例不支持mysqldump导出表数据方式，请联系管理员协助处理。

相关参数解释如表12-8所示。

表 12-8 参数解释

参数	说明	备注
DB_ADDRESS	待导出数据的数据库连接地址。	必填
DB_PORT	数据库侦听端口	必填
DB_USER	数据库用户	必填
DB_NAME	数据库名称	必填
TABLE_NAME	表名	可以多个同类型的表，用空格隔开。 建议只导出与业务相关的表结构。
mysql_table_data.sql	生成的表数据文件名。	每次导出不同表时文件名不同。 建议以“逻辑库名”+“_”+“逻辑表名”+“_”+“data”格式命名，以免数据被覆盖。如mysql_table_data.sql。
--complete-insert	使用完整的insert语句(包含列名称)。	-
--single-transaction	该选项在导出数据之前提交一个BEGIN SQL语句，BEGIN 不会阻塞任何应用程序且能保证导出时数据库的一致性状态。它只适用于多版本存储引擎，仅InnoDB。	-

参数	说明	备注
--quick	不缓冲查询，直接导出到标准输出。	避免大数据情况内存爆涨。
--hex-blob	使用十六进制格式导出二进制字符串字段。如果有二进制数据就必须使用该选项。	-
--no-create-info	只导出数据，而不添加CREATE TABLE 语句。	导出数据时使用。
--skip-comments	关闭附加注释信息。	-
--skip-lock-tables	在不锁表的情况下导出数据。	某些参数会默认开启加锁声明，因此建议在数据导出语句末尾增加此参数。
--add-locks=false	导出的数据文件中不加锁表的声明。	-
--skip-add-locks	在导出数据时，控制加锁动作，以避免因耗能引起的性能问题。	-
--set-gtid-purged=OFF	如果使用的MySQL版本为8.0或5.7，则需要配置该参数。	如果使用的MySQL版本为5.6或低于5.6，则不需要配置该参数。
--where	只转储给定的WHERE条件选择的记录。	如果条件包含命令解释符专用空格或字符，一定要将条件引用起来。

说明

此处举例的参数为数据导出中常用的参数，由于mysqldump参数无法逐一列举，如果存在个别参数调优等特殊情况，请在MySQL官网查询。

步骤3 查看导出SQL文本文件的大小，验证导出数据是否成功。

- 如大小不为0字节，说明导出成功。
- 如大小为0字节，说明导出失败，请联系DDM管理人员。

步骤4 将导出的SQL数据文件上传至已准备的ECS。

----结束

导入数据

步骤1 开启应用程序访问DDM数据库只读开关。

步骤2 清理目标DDM实例的测试数据，防止和待迁移数据冲突。

步骤3 如果是单表或普通表，采用MySQL客户端直连DDM实例，直接执行以下命令导入表结构文本文件和数据文件。


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

- DDM_ADDRESS为待导入数据的DDM实例的地址。
- DDM_PORT为DDM实例的端口。
- DDM_USER为DDM实例的用户名。
- DB_NAME为DDM逻辑库名称，如果导入的是单表，DB_NAME为DDM第一个分片的物理数据库。
- mysql_table_schema.sql为待导入的表结构文件名
- mysql_table_data.sql为待导入的表数据文件名

  说明

  单表或普通表导入前，需要编辑表结构文本文件，将最后一行信息删除（如：Dump completed on 2018-06-28 19:53:03），否则可能导致无法导入。

步骤4 如果是拆分表或广播表，采用MySQL客户端连接DDM将数据文件导入。

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

- DDM_ADDRESS为待导入数据的DDM的地址。
- DDM_PORT为DDM侦听端口。
- DDM_USER为DDM用户。
- DB_NAME为DDM逻辑库名称。
- mysql_table_data.sql为待导入的表数据文件名。

须知

- 数据导入阶段会在一定程度上影响DDM实例以及RDS for MySQL实例性能，请选择在业务低峰时间导入。
- 如果导入过程中出现中断或异常，为防止表数据主键冲突可以用SQL语句**truncate table *{TABLE_NAME}***清空再重新导入。该命令属于高危操作，执行后会清空表中所有数据，请谨慎使用。
- 请勿把数据量大（超过500万）的数据导入到广播表。

----结束

数据验证

步骤1 在ECS上对DDM实例进行逻辑备份。

- 导出表结构：
  - MySQL客户端版本为5.6时请执行以下命令：

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

  - MySQL客户端版本为8.0时请执行以下命令：
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password --column-statistics=0 --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
    ```

- 导出表数据：
  - MySQL客户端版本为5.6时请执行以下命令：
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysq_table_data_new.sql}
    ```
  - MySQL客户端版本为8.0时请执行以下命令：
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-comments --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME} {TABLE_NAME} > {mysq_table_data_new.sql}
    ```

步骤2 检查数据一致性。

1. 在其他云RDS for MySQL和DDM实例执行如下SQL语句检查每张表的记录数是否相等。其中TABLE_NAME是表名。
   ```
   select count(*) from {TABLE_NAME};
   ```
2. 在ECS上对导出前后的表结构和表数据进行比较。
   ```
   diff -B -w -q -i {mysql_table_schema.sql} {mysql_table_schema_new.sql};echo $?
   diff -B -w -q -i {mysql_table_data.sql} {mysql_table_data_new.sql};echo $?
   ```

   说明

   - 表结构的导出命令只适用于单表和普通表。
   - 如果导出数据的顺序不一致，无法比较。
   - 如果导入前后相同，则表示数据迁移成功。
   - 如果数据存在差异，建议联系DDM客服人员进行定位。

步骤3 端到端验证应用程序通过DDM实例访问相关表只读功能是否正常。

步骤4 关闭应用程序访问DDM数据库只读开关。

----结束

业务验证

1. 切换业务数据源至DDM。
2. 验证是否能正常读取、写入数据。
   - 正常：完成迁移。
   - 异常：切换业务数据源至其他云MySQL实例，联系DDM管理员进行定位。

12.5 场景三：华为云上自建 MySQL 迁移到 DDM

场景介绍

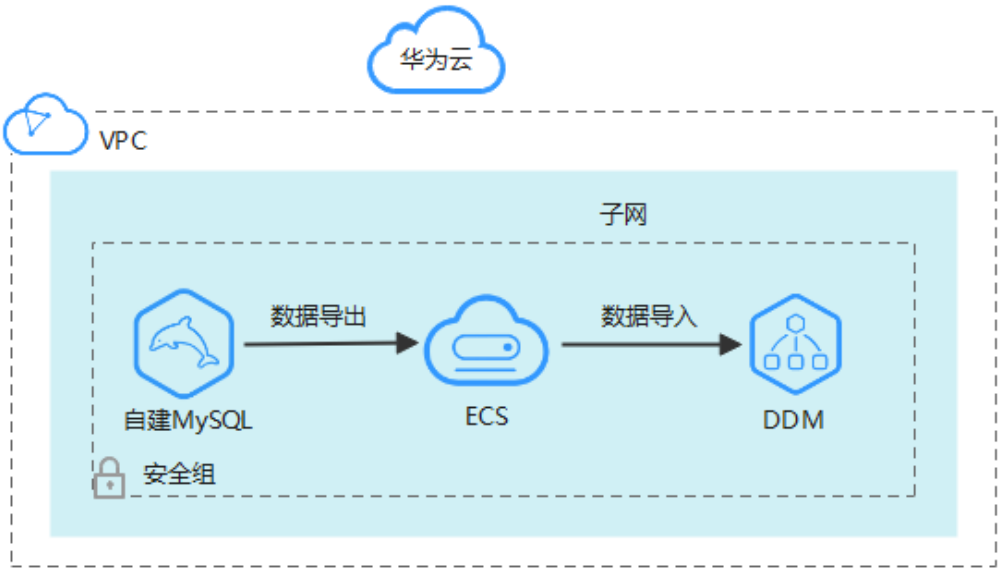
企业在华为云上自建MySQL数据库，希望能使用DDM将数据进行分布式存储。

说明

迁移过程中可能会出现业务中断情况，中断时长与迁移数据量大小、网络情况相关。

迁移示意

图 12-3 华为云上自建 MySQL 迁移到 DDM 示意图



说明

自建MySQL所在服务器与目标DDM实例、RDS for MySQL实例必须处于相同VPC，且安全组配置相同。

约束限制

- 华为云上自建MySQL所在ECS、新购DDM和RDS for MySQL实例建议配置相同VPC和安全组。
- 为了保持数据完整性，需要先停止业务后再进行数据迁移。
- DDM不支持以自动新建库或者新建拆分表、广播表的方式导入数据。因此导入数据前需要先创建好相同名称的逻辑库，相同拆分表、广播表结构的逻辑表，然后再进行数据导入。各类逻辑表创建方式请参见**表12-10**。
- 新增RDS for MySQL实例与源RDS实例的MySQL版本需要保持一致。

迁移前准备

- 准备可以访问源华为云上自建MySQL所在ECS的ECS。
  a. 确保源华为云上自建MySQL、目标DDM实例，RDS for MySQL实例都在同一个VPC下，保证网络互通。
  b. 源华为云上自建MySQL所在ECS、目标DDM实例、RDS for MySQL实例的安全组建议配置相同，如果不同则需要放开对应端口访问。
  c. ECS已安装MySQL官方客户端，MySQL客户端版本建议为5.6或5.7。

- Redhat系列Linux安装命令：**yum install mysql mysql-devel**
- Debian系列Linux安装命令：**apt install mysql-client-5.7 mysql-client-core-5.7**

d. ECS磁盘空间足够存放临时转储文件；ECS内存空间足够，可以用来比较转储文件。

- 准备DDM实例，并配置DDM账号、DDM逻辑库、DDM逻辑表等相关信息。

a. 申请DDM实例，并在DDM控制台创建DDM账号、创建逻辑库。

b. 导出源华为云上自建MySQL的数据表结构至SQL文本文件。

- MySQL客户端版本为5.6和5.7时请执行以下命令：

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

- MySQL客户端版本为8.0时请执行以下命令：

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

相关参数解释如表12-9所示。

表 12-9 参数解释

参数	说明	备注
DB_ADDRESS	待导出数据的数据库连接地址。	必填
DB_PORT	数据库侦听端口	必填
DB_USER	数据库用户	必填
--skip-lock-tables	在不锁表的情况下导出数据。	某些参数会默认开启加锁声明，因此建议在数据导出语句末尾增加此参数。
--add-locks=false	导出的数据文件中不加锁表的声明。	-
--no-data	不导出任何数据，只导出数据库表结构。	导出表结构时使用。
--column-statistics=0	如果使用的MySQL客户端版本为8.0，则必须关闭该特性。	MySQL客户端版本为8.0时必填。
DB_NAME	数据库名称	必填
TABLE_NAME	表名	可以多个同类型的表，用空格隔开。 建议只导出与业务相关的表结构。

参数	说明	备注
mysql_table_schema.sql	生成的表结构文件名。	每次导出表结构时文件名不同。 建议以“逻辑库名”+“_”+“逻辑表名”+“_”+“schema”格式命名，以免数据被覆盖。如mysql_table_schema.sql。

说明

此处举例的参数为数据导出中常用的参数，由于mysqldump参数无法逐一列举，如果存在个别参数调优等特殊情况，请在MySQL官网查询。

c. 创建逻辑表。

创建逻辑表结构请与b中导出的表结构保持一致，把源表映射到目标DDM实例逻辑表，明确对应表结构和表数据的迁移策略，如表12-10所示。

说明

创建前可先通过SQL语句：SHOW CREATE TABLE **{TABLE_NAME}**查看源华为云上自建MySQL中数据表结构。

表 12-10 表迁移策略

逻辑库类型	逻辑表类型	表结构迁移策略	表数据迁移策略
拆分	拆分表	在原表结构基础上增加拆分键声明（详见建表语句说明文档），形成适用于DDM的建表语句，连接DDM并登录对应逻辑库后执行。	通过DDM导入源表数据。
拆分	广播表	在原表结构基础上增加广播声明（详见建表语句说明文档），形成适用于DDM的建表语句，连接DDM并登录对应逻辑库后执行。	
拆分	单表	获取原表建表语句，连接DDM并登录对应逻辑库后执行。	
非拆分	单表		

d. 清理目标DDM实例的测试数据，防止和待迁移数据冲突。

- 准备新RDS for MySQL实例。

导出数据

从源华为云上自建MySQL中将表数据导出到单独的SQL文本文件中。

步骤1 停止华为云上自建MySQL的业务系统，否则可能会导致导出数据不是最新的。

步骤2 导出华为云上自建MySQL的表数据至SQL文本文件。

- MySQL客户端版本为5.6和5.7时请执行以下命令：
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**
- MySQL客户端版本为8.0时请执行以下命令：
  mysqldump -h **{DB_ADDRESS}** -P **{DB_PORT}** -u **{DB_USER}** -p --single-transaction --hex-blob --complete-insert **--set-gtid-purged=OFF** --quick --no-create-info --skip-comments --skip-add-locks --add-locks=false **--column-statistics=0** --skip-tz-utc [--where=""] **{DB_NAME}{TABLE_NAME}** > **{mysql_table_data.sql}**

说明

如果源华为云上自建MySQL中有多个逻辑库，建议分开多次执行该命令导出表数据。

相关参数解释如表12-11所示。

表 12-11 参数解释

参数	说明	备注
DB_ADDRESS	待导出数据的数据库连接地址。	必填
DB_PORT	数据库侦听端口	必填
DB_USER	数据库用户	必填
--single-transaction	该选项在导出数据之前提交一个BEGIN SQL语句，BEGIN 不会阻塞任何应用程序且能保证导出时数据库的一致性状态。它只适用于多版本存储引擎，仅InnoDB。	-
--hex-blob	使用十六进制格式导出二进制字符串字段。如果有二进制数据就必须使用该选项。	-
--complete-insert	使用完整的insert语句(包含列名称)。	-
--set-gtid-purged=OFF	如果使用的MySQL版本为8.0或5.7，则需要配置该参数。	如果使用的MySQL版本为5.6或低于5.6，则不需要配置该参数。

参数	说明	备注
--quick	不缓冲查询，直接导出到标准输出。	-
--no-create-info	只导出数据，而不添加CREATE TABLE 语句。	-
--skip-comments	关闭附加注释信息。	-
--skip-add-locks	在导出数据时，控制加锁动作，以避免因耗能引起的性能问题。	-
--add-locks=false	导出的数据文件中不加锁表的声明。	-
--where	只转储给定的WHERE条件选择的记录。	如果条件包含命令解释符专用空格或字符，一定要将条件引用起来。
DB_NAME	数据库名称	必填
TABLE_NAME	表名	可以多个同类型的表，用空格隔开。 建议只导出与业务相关的表结构。
mysql_table_data.sql	生成的表数据文件名。	每次导出不同表时文件名不同。 建议以“逻辑库名”+“_”+“逻辑表名”+“_”+“data”格式命名，以免数据被覆盖。如mysql_table_data.sql。

说明

- 此处举例的参数为数据导出中常用的参数，由于mysqldump参数无法逐一列举，如果存在个别参数调优等特殊情况，请在MySQL官网查询。
- 使用mysqldump工具进行转移mysql数据时，请保持MySQL客户端版本和DDM所支持的MySQL版本一致。如果版本不一致，可能会影响数据导出性能。

步骤3 查看导出SQL文本文件的大小，验证导出数据是否成功。

- 如大小不为0字节，说明导出成功。
- 如大小为0字节，说明导出失败，请联系DDM管理人员。

----结束

导入数据

步骤1 开启应用程序访问DDM数据库只读开关。

步骤2 清理目标DDM实例的测试数据，防止和待迁移数据冲突。

步骤3 如果是单表或普通表，采用MySQL客户端直连DDM实例，直接执行以下命令导入表结构文本文件和数据文件。

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

- DDM_ADDRESS为待导入数据的DDM实例的地址。
- DDM_PORT为DDM实例的端口。
- DDM_USER为DDM实例的用户名。
- DB_NAME为DDM逻辑库名称，如果导入的是单表，DB_NAME为DDM第一个分片的物理数据库。
- mysql_table_schema.sql为待导入的表结构文件名
- mysql_table_data.sql为待导入的表数据文件名

说明

单表或普通表导入前，需要编辑表结构文本文件，将最后一行信息删除（如：Dump completed on 2018-06-28 19:53:03），否则可能导致无法导入。

步骤4 如果是拆分表或广播表，采用MySQL客户端连接DDM将数据文件导入。

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

- DDM_ADDRESS为待导入数据的DDM的地址。
- DDM_PORT为DDM侦听端口。
- DDM_USER为DDM用户。
- DB_NAME为DDM逻辑库名称。
- mysql_table_data.sql为待导入的表数据文件名。

须知

- 数据导入阶段会在一定程度上影响DDM实例以及RDS for MySQL实例性能，请选择在业务低峰时间导入。
- 如果导入过程中出现中断或异常，为防止表数据主键冲突可以用SQL语句**truncate table {TABLE_NAME}**清空再重新导入。该命令属于高危操作，执行后会清空表中所有数据，请谨慎使用。
- 请勿把数据量大（超过500万）的数据导入到广播表。

----结束

数据验证

步骤1 在ECS上对DDM实例进行逻辑备份。

- 导出表结构：

- MySQL客户端版本为5.6时请执行以下命令：
  ```
  mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password  --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
  ```
- MySQL客户端版本为8.0时请执行以下命令：
  ```
  mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --skip-lock-tables --default-auth=mysql_native_password --column-statistics=0 --set-gtid-purged=OFF --skip-tz-utc --no-data {DB_NAME} {TABLE_NAME} > {mysql_table_schema_new.sql}
  ```

- 导出表数据：
  - MySQL客户端版本为5.6和5.7时请执行以下命令：
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-add-locks --add-locks=false --skip-comments --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data_new.sql}
    ```
  - MySQL客户端版本为8.0时请执行以下命令：
    ```
    mysqldump -h {DDM_ADDRESS} -P {DDM_PORT} -u {DDM_USER} -p --single-transaction --hex-blob --complete-insert --set-gtid-purged=OFF --quick --no-create-info --skip-add-locks --add-locks=false --skip-comments --column-statistics=0 --skip-tz-utc [--where=""] {DB_NAME}{TABLE_NAME} > {mysql_table_data_new.sql}
    ```

步骤2 检查数据一致性。

1. 在源云上自建MySQL和DDM实例执行如下SQL语句检查每张表的记录数是否相等。其中TABLE_NAME是表名。
   ```
   select count(*) from {TABLE_NAME};
   ```
2. 在ECS上对导出前后的表结构和表数据进行比较。
   ```
   diff -B -w -q -i {mysql_table_schema.sql} {mysql_table_schema_new.sql};echo $?
   diff -B -w -q -i {mysql_table_data.sql} {mysql_table_data_new.sql};echo $?
   ```

   说明

   - 表结构的导出命令只适用于单表和普通表。
   - 如果导出数据的顺序不一致，无法比较。

   - 如果导入前后相同，则表示数据迁移成功。
   - 如果数据存在差异，建议联系DDM客服人员进行定位。

步骤3 端到端验证应用程序通过DDM实例访问相关表只读功能是否正常。

步骤4 关闭应用程序访问DDM数据库只读开关。

----结束

业务验证

1. 切换业务数据源至DDM。
2. 验证是否能正常读取、写入数据。
   - 正常：完成迁移。
   - 异常：切换业务数据源至源华为云上自建MySQL，联系DDM客服人员进行定位。

12.6 场景四：从 DDM 实例导出数据

场景介绍

因为业务使用需要，将DDM实例的数据导出成SQL文本文件。

使用须知

- 本阶段会在一定程度上影响DDM实例以及RDS for MySQL实例性能，请选择在业务低峰时间导出。
- 请选择在大磁盘上进行mysqldump操作，保证磁盘空间充足。
- 在Linux系统中，为防止会话超时导致mysqldump提前退出生成不完整的数据文件，建议在系统后台执行。

  执行语句：nohup {mysqldump 命令行} &

导出表结构

DDM为2.4.X及以上版本执行以下命令导出表结构。

- MySQL客户端版本为5.6和5.7时请执行以下命令：

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

- MySQL客户端版本为8.0时请执行以下命令：

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

导出表数据

DDM为2.4.X及以上版本执行以下命令导出表数据。

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

说明

- 为了提高导出数据的效率，对于非拆分库的表可以直连RDS for MySQL实例进行导出。
- mysqldump5.7官网链接：**https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html**。

12.7 场景五：其他异构数据库迁移到 DDM

其他异构数据库，如Oracle、PostgreSQL、SQL Server等数据迁移，可以参考华为云数据迁移（CDM）服务用户指南或者联系DDM客服人员进行迁移。

12.8 场景六：从华为云 RDS for MySQL 迁移到 DDM

通过数据复制服务（DRS）将RDS for MySQL数据迁移到DDM。

13 会话管理

您可以在以下场景使用会话管理功能：

- 紧急救助通道：在实例的连接数达到上限，无法正常登录时，该功能提供了一个特殊连接通道，可以查看会话或者执行Kill操作。
- 历史急救日志：查看您在急救通道执行过的Kill操作记录。

使用须知

- DDM支持查看CN会话（应用与DDM之间的连接）和DN会话（DDM与数据节点之间的连接）。
- 仅支持数据节点为RDS for MySQL。
- 不支持创建中、冻结、异常实例。
- CPU负载高的情况下，Kill会话请求有一定概率会超时（大约1分钟），可以稍后刷新会话列表查看目标会话是否还存在，如果还存在则需要重新执行Kill会话操作。

操作步骤

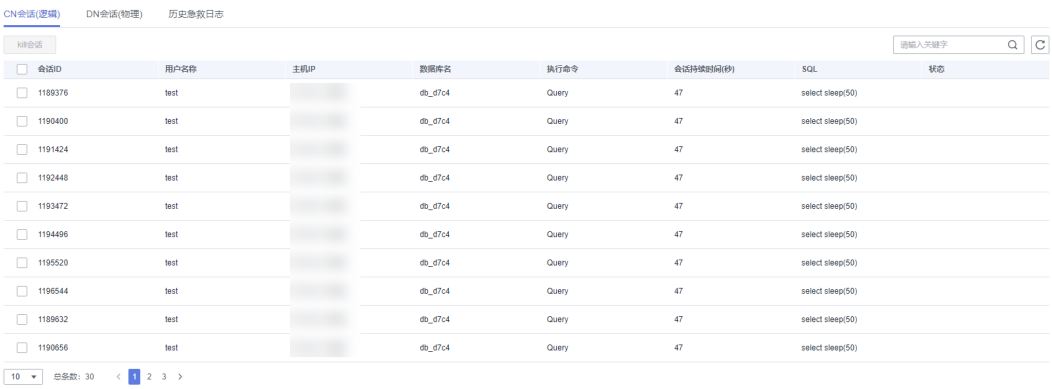
步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击进入目标实例。

步骤3 单击左侧菜单栏的“会话管理”页签，进入“CN会话(逻辑)”页面。

步骤4 在“CN会话(逻辑)”页面，可以查看到应用与DDM之间的会话信息列表。

图 13-1 CN 会话列表



- 在右上角通过输入关键字查询对应的会话信息。
- 在会话列表中，勾选一条或多条会话，单击“kill会话”并“确定”，完成结束会话操作。

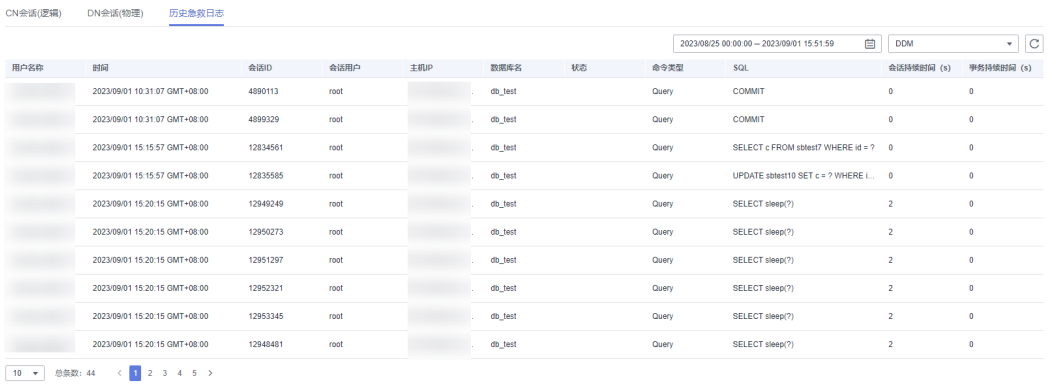
步骤5 单击“DN会话(物理)”页签，可以查看到DDM与数据节点（RDS for MySQL）之间的会话信息列表。

图 13-2 DN 会话列表

- 在右上角选择一个或多个数据节点，查看DDM与数据节点之间的会话信息。如果您选择多个数据节点，当前分页数量及总条数仅针对单个数据节点查询生效。
- 在右上角通过输入SQL语句查询对应的会话信息。
- 在会话列表中，勾选一条或多条会话，单击“kill会话”并“确定”，完成结束会话操作。

步骤6 单击“历史急救日志”页签，您可以获取执行过的Kill操作信息。

图 13-3 查看历史急救日志



您还可以进行如下操作：

- 通过设置开始时间和结束时间来查询所需时间段内执行过的Kill操作信息。
- 在列表右上角的下拉框中选择目标DDM实例或数据节点（RDS for MySQL实例），可筛选出对应的CN会话或DN会话信息。

----结束

14 慢查询

操作场景

DDM提供“慢查询”功能，将指定时间内的慢SQL语句进行统计分类，把结构相同的慢SQL语句整理成SQL模板，您可以查看指定区间内的所有慢SQL类型，针对这些类型进行优化处理。

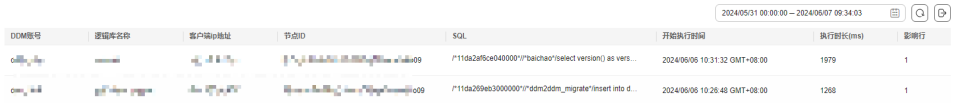
操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在实例管理列表页面，单击进入目标实例。

步骤3 单击左侧菜单栏的“慢查询”页签，进入“慢查询”页面。

图 14-1 慢查询服务页面



- 在“慢查询”页面可以查看超过指定时间的SQL语句。
- 在“慢查询”页面右上角单击 ，导出慢日志文件。

  说明

  导出慢日志的文件每次最大不超过8MB。您可以通过选择不同时间段分批导出慢日志文件。

----结束

15 监控与告警

15.1 支持的监控指标

15.1.1 实例监控指标

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。

命名空间

SYS.DDMS

说明

SYS.DDM是DDM1.0版本的命名空间。

SYS.DDMS是DDM2.0版本的命名空间。

目前DDM服务已经全面升级为2.0版本，少数1.0存量用户的命名空间仍然为SYS.DDM。

监控指标

表 15-1 DDM 支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
ddm_cpu_util	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
ddm_mem_util	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

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
ddm_bytes_in	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
ddm_bytes_out	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
ddm_qps	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
ddm_read_count	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
ddm_write_count	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
ddm_slow_log	慢SQL数	该指标用于统计数据面服务Core的慢SQL条数。	≥ 0 counts	DDM节点	1分钟
ddm_rt_avg	平均响应时延	该指标用于统计数据面服务Core的SQL平均响应时延。	≥ 0 ms	DDM节点	1分钟
ddm_connections	连接数	该指标用于统计数据面服务Core的连接数。	≥ 0 counts	DDM节点	1分钟
ddm_backend_connection_ratio	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
active_connections	活跃连接数	该指标用于统计每个DDM节点后台正在执行的连接数目。	≥ 0	DDM节点	1分钟
ddm_connection_util	连接数使用率	该指标用于统计每个DDM节点已用的连接数占总连接数的百分比。	0~100 %	DDM节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
ddm_node_status_alarm_code	DDM节点连通性检测	该指标用于检测每个DDM节点是否在集群中不可用。其中，0表示可用，1表示不可用。	0或1	DDM节点	1分钟
ddm_global_sequence_threshold_exceeded_count	超过使用率阈值的全局二级序列个数	此指标统计使用率超过75%的全局序列个数。使用率 = 当前值 / 最大值，默认使用率阈值为75%。全局序列的最大值取决于全局序列类型（例如：BIGINT类型，最大值：2^63-1）。	≥ 0 counts	DDM实例	10分钟

维度

Key	Value
instance_id	DDM实例ID。
node_id	DDM节点ID。

15.1.2 网络监控指标

如果DDM实例开通了负载均衡，则可以查看以下网络监控指标。如果没有开通，则无权查看。

表 15-2 负载均衡支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
m7_in_Bps	网络流入速率	从外部访问测量对象所消耗的流量。 单位：字节/秒	≥ 0 bytes/s	独享型负载均衡	1分钟
m8_out_Bps	网络流出速率	测量对象访问外部所消耗的流量。 单位：字节/秒	≥ 0 bytes/s	独享型负载均衡	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
ma_normal_servers	正常主机数	健康检查统计监控对象后端正常的主机个数。 单位：个	≥ 0 个	独享型负载均衡	1分钟
l4_in_bps_usage	4层入带宽使用率	统计客户端到测量对象当前使用TCP/UDP协议的入网带宽与设定的入网带宽的最大值之间的比。	0~100 %	独享型负载均衡	1分钟
l4_out_bps_usage	4层出带宽使用率	统计测量对象到客户端当前使用TCP/UDP协议的出网带宽与设定的出网带宽的最大值之间的比。	0~100 %	独享型负载均衡	1分钟

15.2 查看监控指标

15.2.1 查看实例监控指标

云监控服务可以对DDM实例的运行状态进行日常监控。您可以通过管理控制台，直观地查看DDM的各项监控指标。

由于监控数据的获取与传输会花费一定时间，因此，云监控显示的是当前时间5～10分钟前的DDM状态。如果您的DDM刚创建完成，请等待5～10分钟后查看监控数据。

前提条件

- DDM实例正常运行。
  故障、删除状态的DDM实例无法在云监控中查看其监控指标。
- DDM实例已正常运行一段时间（约10分钟）。
  对于新创建的DDM实例，需要等待一段时间，才能查看上报的监控数据和监控视图。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择目标实例，单击操作列中的“更多 > 查看监控指标”，进入云监控服务页面。

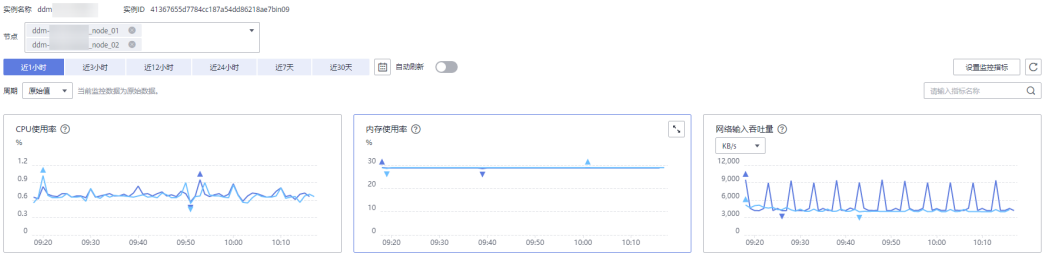
您也可以在“实例管理”页面，单击目标实例名称，在页面右上角，单击“查看监控指标”，进入云监控服务页面。

步骤3 选择目标实例，单击实例名称左侧的，单击操作列的“查看监控指标”。

图 15-1 查看指标

您可以查看实例CPU使用率、内存使用率、网络输入吞吐量、网络输出吞吐量、QPS、慢SQL数等监控指标详情，关于分布式数据库中间件支持的监控指标请参见**实例监控指标**。

图 15-2 指标详情



----结束

15.2.2 查看网络监控指标

DDM控制台提供了监控管理功能，旨在对网络监控指标进行监控管理。

前提条件

如果DDM实例开通了负载均衡，则可以查看网络监控指标。如果没有开通，则无权查看。

操作步骤

步骤1 登录分布式数据库中间件控制台。

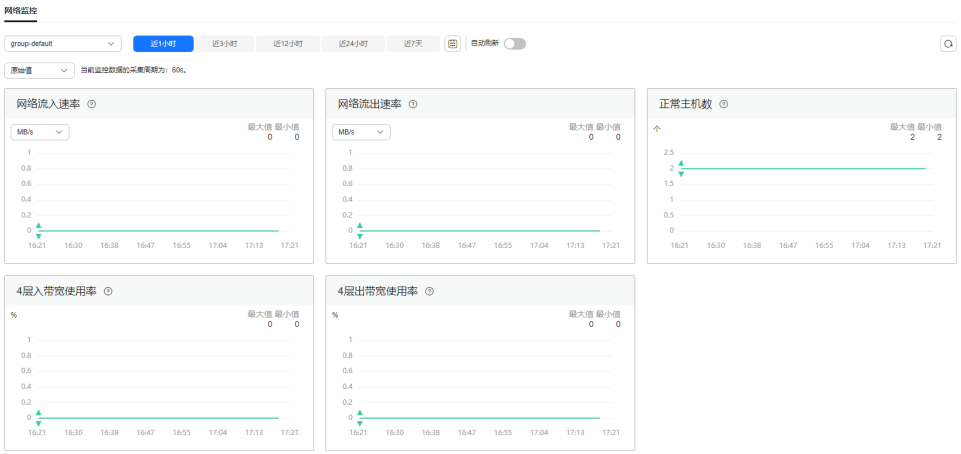
步骤2 在实例管理列表页面，单击进入目标实例，进入基本信息页面。

步骤3 在左侧导航栏中，单击“监控管理”页签。

步骤4 查看网络监控。

您可以通过时间范围进行筛选，在实时监控列表中查看网络流入速率、网络流出速率、正常主机数等监控信息。更多支持的网络监控指标请参见**网络监控指标**。

图 15-3 网络监控



----结束

15.3 设置监控指标告警规则

操作场景

云监控界面支持设置告警规则，用户可以自定义监控目标与通知策略，及时了解DDM的运行状况，从而起到预警作用。

设置DDM的告警规则包括设置告警规则名称、资源类型、维度、监控对象、监控指标、告警阈值、监控周期和是否发送通知等参数。

操作步骤

步骤1 登录管理控制台。

步骤2 在“服务列表”中选择“管理与监管 > 云监控服务”，进入“云监控”服务信息页面。

步骤3 在左侧导航栏选择“云服务监控 > 分布式数据库中间件”。

步骤4 选择目标实例，单击实例名称左侧的，选择操作列的“创建告警规则”。

图 15-4 创建告警规则

步骤5 在“创建告警规则”页面，填选相关信息。

图 15-5 告警规则

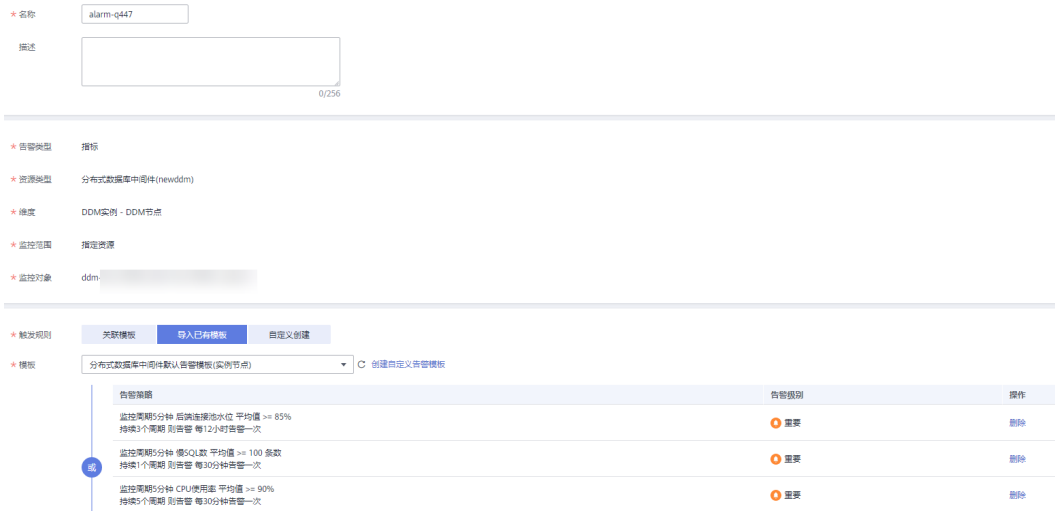


表 15-3 告警规则参数

参数名称	说明
名称	系统会随机产生一个名称，用户也可以进行修改。
描述	告警规则描述（此参数非必填项）。
触发规则	根据需要可选择关联模板、导入已有模板或自定义创建。 如果您选择了“自定义创建”，可根据实际需求配置告警策略及告警级别。 **说明** 选择关联模板后，所关联模板内容修改后，该告警规则中所包含策略也会跟随修改。

图 15-6 告警通知

表 15-4 告警通知参数

参数名称	说明
发送通知	配置是否发送邮件、短信、HTTP和HTTPS通知用户。

参数名称	说明
通知方式	根据需要可选择通知组或主题订阅两种方式。 • 通知组 需要发送告警通知的通知组。 • 主题订阅 需要发送告警通知的对象，可选择云账号联系人或主题名称。 – 云账号联系人为注册时的手机和邮箱。 – 主题是消息发布或客户端订阅通知的特定事件类型，如果此处没有需要的主题则需先创建主题并添加订阅，创建主题并添加订阅请参见**创建主题**、**添加订阅**。
生效时间	该告警仅在生效时间段发送通知消息，非生效时段则在隔日生效时段发送通知消息。 如生效时间为08:00-20:00，则该告警规则仅在08:00-20:00发送通知消息。
触发条件	可以选择“出现告警”、“恢复正常”两种状态，作为触发告警通知的条件。

说明

“告警通知”功能触发产生的告警消息由消息通知服务SMN发送，可能产生少量费用。

图 15-7 高级选项

表 15-5 高级选项参数

参数名称	说明
归属企业项目	告警规则所属的企业项目。只有拥有该企业项目权限的用户才可以查看和管理该告警规则。

步骤6 配置完成后，单击“立即创建”，完成告警规则的创建。

告警规则添加完成后，当监控指标触发设定的阈值时，云监控服务会在第一时间通过消息通知服务实时告知您云上资源异常，以免因此造成业务损失。

----结束

15.4 事件监控

15.4.1 事件监控简介

事件监控提供了事件类型数据上报、查询和告警的功能，方便您将业务中的各类重要事件或对云资源的操作事件收集到云监控服务，并在事件发生时进行告警。

事件监控默认开通，您可以在事件监控中查看系统事件和自定义事件的监控详情，DDM目前支持的系统事件请参见**事件监控支持的事件说明**。

15.4.2 查看事件监控数据

操作场景

事件监控提供了事件类型数据上报、查询和告警的功能。方便您将业务中的各类重要事件或对云资源的操作事件收集到云监控服务，并在事件发生时进行告警。

事件监控默认开通，您可以在事件监控中查看系统事件和自定义事件的监控详情。

本章节指导用户查看事件监控的监控数据。

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择目标实例，单击操作列中的“更多 > 查看监控指标”，跳转到云监控服务页面。

您也可以在“实例管理”页面，单击目标实例名称，在页面右上角，单击“查看监控指标”，进入云监控服务页面。

步骤3 单击左侧导航栏中的“事件监控”。

进入“事件监控”页面。在“事件监控”页面，默认展示近24小时的所有系统事件。

您也可以根据需要选择“近1小时”“近3小时”“近12小时”“近24小时”“近7天”“近30天”，分别查看不同时段的事件。

步骤4 展开对应的事件类型，单击具体事件右侧的操作列的“查看事件”，可查看具体事件的内容。

----结束

15.4.3 创建事件监控的告警通知

操作场景

本章节指导用户创建事件监控的告警通知。

操作步骤

步骤1 登录管理控制台。

步骤2 在页面左上角单击，选择“管理与监管 > 云监控服务”，进入“云监控服务”信息页面。

步骤3 在左侧导航栏选择“事件监控”，进入“事件监控”页面。

步骤4 在事件列表页面，单击页面右上角的“创建告警规则”。

步骤5 在“创建告警规则”界面，配置参数。

表 15-6 告警内容参数说明

参数	参数说明
名称	系统会随机产生一个名称，用户也可以进行修改。
描述	告警规则描述（此参数非必填项）。
告警类型	选择事件。
事件类型	用于指定事件类型，可选择系统事件或自定义事件。
事件来源	• 对于系统事件 需要选择事件来源的云服务名称，此处选择分布式数据库中间件。 • 对于自定义事件 事件来源需要与上报的字段一致，格式需要为service.item形式。
选择类型	默认为自定义创建。
告警策略	事件名称：用户操作系统资源的动作，如用户登录，用户登出，为一个瞬间的操作动作。 事件监控支持的操作事件请参见**事件监控支持的事件说明**。 用户根据需要选择触发方式、告警级别。

单击开启“发送通知”，生效时间默认为全天，如果没有您想要选择的主题，可以单击下一行的“创建主题”进行添加。

表 15-7 发送通知

参数	参数说明
发送通知	配置是否发送邮件、短信、HTTP和HTTPS通知用户。
通知方式	根据需要可选择通知组或主题订阅两种方式。
通知组	需要发送告警通知的通知组。

参数	参数说明
通知对象	需要发送告警通知的对象，可选择“云账号联系人”或主题。 • 云账号联系人：注册账号时的手机和邮箱。 • 主题：消息发布或客户端订阅通知的特定事件类型，如果此处没有需要的主题，需先创建主题并订阅该主题。 详细操作请参见**创建主题**和**添加订阅**。
生效时间	该告警规则仅在生效时间内发送通知消息。 如生效时间为08:00-20:00，则该告警规则仅在08:00-20:00发送通知消息。
触发条件	出现告警。

步骤6 配置完成后，单击“立即创建”，完成告警规则的创建。

----结束

15.4.4 事件监控支持的事件说明

表 15-8 分布式数据库中间件

事件来源	事件名称	事件ID	事件级别	事件说明	处理建议	事件影响
DDM	创建实例失败	createDdmInstanceFailed	重要	一般是由于底层资源不足等原因导致。	释放资源后重新创建。	无法创建DDM实例。
	变更规格失败	resizeFlavorFailed	重要	一般是由于底层资源不足等原因导致。	请工单联系运维在后台协调资源再重试规格变更操作。	部分节点业务中断。
	节点扩容失败	enlargeNodeFailed	重要	一般是由于底层资源不足等原因导致。	请工单联系运维后台协调资源，删除添加失败的节点，重新尝试添加节点。	节点扩容失败。
	节点缩容失败	reduceNodeFailed	重要	一般是由于底层释放资源失败等原因导致。	请工单联系运维后台处理资源。	节点缩容失败。
	重启实例失败	restartInstanceFailed	重要	一般是由于底层关联数据库实例异常等原因导致。	建议先排查底层数据库实例是否异常，如果无异常请工单联系运维进行排查。	部分节点业务中断。

事件来源	事件名称	事件ID	事件级别	事件说明	处理建议	事件影响
	创建逻辑库失败	createLogicDbFailed	重要	一般是由于以下几种原因导致的： 1. 数据库实例账号密码错误。 2. DDM实例与底层数据库实例安全组设置错误，导致无法通信。	请排查： 1. 数据库实例账号密码是否错误。 2. DDM实例与底层数据库实例安全组是否设置正确等问题。	业务无法正常运行。
	绑定弹性公网IP失败	bindEipFailed	重要	一般是由于EIP服务繁忙。	稍后重试，紧急情况下请联系运维排查问题。	无法通过公网访问服务。
	逻辑库扩容失败	migrateLogicDbFailed	重要	一般是由于底层处理失败。	请工单联系运维处理。	无法实现逻辑库扩容。
	逻辑库扩容重试失败	retryMigrateLogicDbFailed	重要	一般是由于底层处理失败。	请工单联系运维处理。	无法实现逻辑库扩容。

16 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。

支持查看的任务说明

DDM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
- 分片变更
- 分片变更重试
- 删除备份
- 创建组
- 删除组
- 重启节点
- 版本升级
- 版本降级
- 版本回滚

操作步骤

步骤1 登录分布式数据库中间件控制台。

步骤2 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实例ID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

17 标签

标签管理服务（Tag Management Service，TMS）用于用户在云平台，通过统一的标签管理各种资源。TMS服务与各服务共同实现标签管理能力，TMS提供全局标签管理能力，各服务维护自身标签管理。

使用须知

- 标签由“键”和“值”组成，每个标签中的一个“键”只能对应一个“值”。
- 每个实例最多支持10个标签配额。

添加标签

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称，进入实例的“基本信息”页签。

步骤3 在左侧导航栏选择“标签”，单击“添加标签”。

步骤4 在弹出框中，输入标签的键和值，然后单击“确定”。

图 17-1 设置标签

标签的键值需要满足如下规则：

表 17-1 键值参数说明

参数	说明
键	该项为必选参数，不能为空。 ● 对于每个实例，每个标签的键唯一。 ● 长度为1~36个字符。 ● 不能为空字符串，不能以“_sys_”开头和以空格开头、结尾。 ● 不能包含下列字符： 非打印字符ASCII(0-31)，“*”，“<”，“>”，“\”，“,”，“|”，“/”。
值	该项为必选参数。 ● 可以不填值，此时默认为空字符串。 ● 长度为0~43个字符。 ● 不能包含下列字符： 非打印字符ASCII(0-31)，“*”，“<”，“>”，“\”，“,”，“|”。

步骤5 添加成功后，您可在当前实例的所有关联的标签集合中，查询并管理自己的标签。

----结束

编辑标签

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称。

步骤3 在左侧导航栏选择“标签”，单击操作列的“编辑”，在弹出框中修改标签值，单击“确定”。

编辑标签时，不能修改标签的键，只能修改标签的值。

步骤4 编辑成功后，您可在当前实例的所有关联的标签集合中，查询并管理自己的标签。

----结束

删除标签

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，选择指定的实例，单击实例名称。

步骤3 在左侧导航栏选择“标签”，单击操作列的“删除”，在弹出框中单击“是”。

步骤4 删除成功后，该标签将不再显示在实例的所有关联的标签集合中。

----结束

根据标签筛选实例

标签添加成功后，您可以通过标签来筛选实例，快速查找指定分类的实例以进行管理。

步骤1 登录分布式数据库中间件控制台。

步骤2 在“实例管理”页面，单击实例列表右上方的“标签搜索”。

图 17-2 标签搜索

步骤3 输入标签键和值，单击“搜索”。

步骤4 查看搜索到的实例信息。

----结束

18 审计

18.1 支持审计的关键操作列表

通过云审计服务，您可以记录与华为云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 18-1 云审计服务支持的 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
租户进行包周期云服务续费、包周期转按需、按需转包周期	all	bssArrearage
更新云服务metadata信息	all	bssUpdateMetadata
清理逻辑库扩容后的元数据	logicDB	cleanMigrateLogicDB
清理用户资源	all	cleanupUserAllResources
复制参数模板	parameterGroup	copyParameterGroup
创建实例	instance	createInstance
创建逻辑库	logicDB	createLogicDB
创建参数模板	parameterGroup	createParameterGroup
创建账号	user	createUser
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	user	deleteUser
节点扩容	instance	enlargeNode
重启实例	instance	instanceRestart
导入逻辑库信息	instance	loadMetadata
扩容路由切换	logicDB	manualSwitchRoute
逻辑库扩容	logicDB	migrateLogicDB
修改参数模板	parameterGroup	modifyParameterGroup
修改路由切换时间	logicDB	modifyRouteSwitchTime
修改账号信息	user	modifyUser
节点缩容	instance	reduceNode
重置参数模板	parameterGroup	resetParameterGroup
重置账号密码	user	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
访问控制	instance	switchIpGroup
同步DN信息	instance	synRdsinfo
升级DDM实例版本	instance	upgrade
添加标签	instance	addTag
创建组	group	createGroup
修改内网地址	instance	modifyIp
修改实例名称	instance	modifyName

18.2 查看追踪事件

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。

说明

使用云审计服务前需要先开通云审计服务，请参见**开通云审计服务**。

操作步骤

步骤1 登录管理控制台。

步骤2 在“服务列表”中，选择“管理与监管 > 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：

- 事件类型、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

19 SQL 语法

19.1 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 19-1 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。连接类型按照执行速度从快到慢排序为：const、eq_reg、ref、range、index、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。
- 不支持CREATE TABLE tbl_name LIKE old_tbl_name。
- 不支持CREATE TABLE tbl_name SELECT statement。
- 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。

  例如：

  ```
  SET @c=1, @d=@c+1;
  SELECT @c, @d;
  ```

- INSERT或者UPDATE时，不支持插入或者更新拆分键值为DEFAULT关键字。
- UPDATE不支持在一个语句中对同一字段重复更新。
- UPDATE不支持关联更新拆分键。
- UPDATE不支持自关联更新。
- 关联更新中，不支持在目标列的赋值语句或表达式中引用其它目标列，将造成更新结果不符合预期。

例如：

```
update tbl_1 a,tbl_2 b set a.name=concat(b.name,'aaaa'),b.name=concat(a.name,'bbbb')
on a.id=b.id;
```

- 当使用文本协议时，BINARY、VARBINARY、TINYBLOB、BLOB、MEDIUMBLOB、LONGBLOB数据必须转换成16进制数据。
- DDM对非法数据的处理与后端MySQL的sql_mode有关。
- 关联更新不支持不带关联条件的Join。
- SQL语句中表达式的因子数量请勿超过1000个。

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。

注释说明

- 单行注释
  - 使用井号（#）开头，后面直接写注释内容。#后面的所有内容直到行尾都被视为注释。

    示例如下：

    ```
    SELECT * FROM customers; #注释内容
    ```
  - 使用两个连续的短划线（--），但这种注释要求第二个短划线后面至少有一个空格符，否则可能不会被正确解析。

    示例如下：

    ```
    SELECT * FROM Product; -- 注释内容
    ```
- 多行注释

  以/*开始并且以*/结束，可以包含多行文本。这种方式允许注释跨越多行。

  示例如下：

  ```
  /*
  注释内容
  */
  SELECT DISTINCT product_id, purchase_price FROM Product;
  ```

19.2 DDL

19.2.1 DDL 概述

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

⚠ 注意

- rename table name不支持与其他DDL语句在同一条SQL执行。
- 对拆分表执行修改字段名操作的同时执行类似SELECT * [DDL相关表]等包含全字段的查询语句时，有可能会上报列名不存在的异常，建议在业务低峰期执行此类修改操作，且等修改操作执行完成后再进行相关查询操作，以降低报错概率。
- 在DDM节点或后端RDS节点压力极大的情况下，对拆分表执行删除字段名操作的同时执行SELECT * [DDL相关表]等包含全字段的查询语句时，有可能会上报列名不存在的异常，建议在业务低峰期执行删除类操作，且等删除类操作执行完成后再进行相关查询操作，以降低报错概率。

- TRUNCATE Syntax

  示例：

  ```
  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

  示例：

  ```
  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

  示例：

  ```
  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

  示例：

  ```
  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。

19.2.2 创建表

说明

- 禁止创建表名以"_ddm"为前缀的表，系统默认认定此类表为系统内部表。
- 拆分表不支持全局唯一索引，当唯一键和拆分键不一致时，不能保证数据的唯一性。
- 建议使用bigint型作为自增键的数据类型。tinyint、smallint、mediumint、integer、int数据类型不建议作为自增键的类型，容易越界造成值重复。

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 bigint 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 bigint 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 bigint 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

单表也可以指定全局序列，但是会忽略这个功能。自增功能使用底层存储的自增值进行自增。

创建单表样例如下，不做任何拆分：

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

19.2.3 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 19-2 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
MOD_HASH_CI	简单取模（大小写不敏感）	是	是
HASH	计算CRC32值，再简单取模	是	是
RANGE	按范围	是	否
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
MM	按月份哈希	否	是
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，如果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 19-3 DDM 拆分函数对各种数据类型的支持情况

拆分算法	TINYINT	SMALLINT	MEDIUMINT	INTEGER	INT	BIGINT	CHAR	VARCHAR	DATE	DATETIME	TIMESTAMP	OTHERS
MOD_HASH	√	√	√	√	√	√	√	√	×	×	×	×
MOD_HASH_CI	√	√	√	√	√	√	√	√	×	×	×	×
HASH	√	√	√	√	√	√	√	√	×	×	×	×
RANGE	√	√	√	√	√	√	×	×	×	×	×	×
RIGHT_SHIFT	√	√	√	√	√	√	×	×	×	×	×	×
YYYYMM	×	×	×	×	×	×	×	×	√	√	√	×
YYYYDD	×	×	×	×	×	×	×	×	√	√	√	×
YYYYWEEK	×	×	×	×	×	×	×	×	√	√	√	×
MM	×	×	×	×	×	×	×	×	√	√	√	×
DD	×	×	×	×	×	×	×	×	√	√	√	×
MMDD	×	×	×	×	×	×	×	×	√	√	√	×
WEEK	×	×	×	×	×	×	×	×	√	√	√	×

说明

√表示支持，×表示不支持。

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]
    ]
```

19.2.4 拆分算法使用说明

19.2.4.1 MOD_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(支持精度为0的情况) ）或字符串类型(CHAR,VARCHAR)。
- 在SQL语句中对整数类型拆分键设置值时不要进行类型转换，类型转换可能造成路由计算失败后路由至默认分片，造成目标数据查询不到。

路由方式

根据拆分键的键值直接按分库数/分表数取余。如果键值是字符串，则字符串会被计算成哈希值再进行计算，完成路由计算。

例如：MOD_HASH('8')等价于8%D（ D是分库数目/分表数 ）。

算法计算方式

方式一：拆分键是整型

表 19-4 拆分键是整型时的计算方式

条件	算法	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(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 **说明** 分库路由结果四舍五入到最接近的整数。	分表：16 % (8 * 3) = 16 分库： 16 / 3 = 5

方式二：拆分键是字符类型

表 19-5 拆分键是字符型时的计算方式

条件	算法	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(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 **说明** 分库路由结果四舍五入到最接近的整数。	hash（‘abc’)= ‘abc’.hashCode()=96354 分表 :96354% (8 * 3) = 18 分库 :18 / 3=6

建表语法

- 如果用户需要对ID列按MOD_HASH函数进行分库不分表：

```
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 mod_hash(ID);
```

- 如果用户需要对ID列按MOD_HASH函数进行既分库又分表：

```
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 mod_hash(ID) tbpartition by mod_hash(ID) tbpartitions 4;
```

注意事项

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。

19.2.4.2 MOD_HASH_CI 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(支持精度为0的情况)）或字符串类型(CHAR,VARCHAR)。
- 在SQL语句中对数字类型拆分键设置值时不要进行类型转换，类型转换可能造成路由计算失败后路由至默认分片，造成目标数据查询不到。

路由方式

实现原理同MOD_HASH算法一致，区别在于MOD_HASH_CI算法对大小写不敏感，而MOD_HASH算法对大小写敏感。

算法计算方式

方式一：拆分键是整型

表 19-6 拆分键是整型时的计算方式

条件	算法	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(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 **说明** 分库路由结果四舍五入到最接近的整数。	分表：16 % (8 * 3) = 16 分库： 16 / 3 = 5

方式二：拆分键是字符类型

表 19-7 拆分键是字符型时的计算方式

条件	算法	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’.toUpperCase().hashCode()=64578 分库 :64578 % 8 = 2; 分表 :64578 % 3 = 0;
分库拆分键 = 分表拆分键(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 **说明** 分库路由结果四舍五入到最接近的整数。	hash（‘abc’)=‘abc’.toUpperCase().hashCode()=64578 分表 :64578% (8 * 3) = 18 分库 :18 / 3=6

建表语法

- 如果用户需要对ID列按MOD_HASH_CI函数进行分库不分表：

```
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);
```

- 如果用户需要对ID列按MOD_HASH_CI函数进行既分库又分表：

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

注意事项

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。

19.2.4.3 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。

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 19-8 计算方式

条件	算法	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

移位的数目不能超过整数类型所占有的位数目。

19.2.4.4 MM 按月份哈希

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 19-9 算法举例

条件	算法	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。

19.2.4.5 DD 按日期哈希

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 19-10 算法举例

条件	算法	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。

19.2.4.6 WEEK 按星期哈希

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。

说明

您可以使用如下SQL查询指定日期的工作日索引（0 = 星期一，1 = 星期二，... 6 = 星期日）。

```
mysql> SELECT WEEKDAY('2019-01-15');
        -> 1
```

上述SQL返回值为1，表示2019-01-15是周二，因为周日为一周中的第一天，所以周二则为一周中的第三天。

算法计算方式

表 19-11 算法举例

条件	算法	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。

19.2.4.7 MMDD 按月日哈希

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 19-12 算法举例

条件	算法	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。

19.2.4.8 YYYYMM 按年月哈希

适用场景

适用于需要按年份与月份进行分库的场景，建议该函数与 tbpartition YYYYMM(ShardKey) 联合使用。

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(‘2012-12-31 12:12:12’) 等价于 (2012 * 12 + 12) % D（D是分库数目/分表数）。

算法计算方式

表 19-13 算法计算方式

条件	算法	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 **说明** 分库路由结果四舍五入到最接近的整数。	拆分键：2012-11-20 分表：(2012 * 12 + 11) % (8 * 3) = 11 分库 ： 11 % 3 = 3

建表语法

假设用户的实例里已经分了8个物理库，现有一个业务想按年月进行分库。要求同一个月的数据能落在同一张分表内，并且两年以内的每个月都单独对应一张分表，查询时带上分库分表键后能直接将查询落在某个物理分库的某个物理分表。

用户这时就可以使用YYYYMM分库函数来解决：业务要求两年以内的每个月都对应一张分表（即一个月一张表），由于一年有 12 个月，所以至少需要创建 24 个物理分表才能满足用户的场景，而用户的DDM有8个分库，所以每个分库应该建3张物理分表。建表语法如下所示：

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

只分库场景的建表语法：

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


注意事项

- YYYYMM算法不支持对于每一个年月都独立对应一张分库，分库分表时必须固定分表数目。
- 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分表，请以实际的分表数目而定。

19.2.4.9 YYYYDD 按年日哈希

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，建议该函数与tbpartition YYYYDD(ShardKey) 联合使用。

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(‘2012-12-31 12:12:12’) 等价于 (2012 * 366 + 366) % D（D是分库数目/分表数）。

说明

"2012-12-31"是2012年第366天，所以为"2012 * 366+366"。

算法计算方式

表 19-14 算法计算方式

条件	算法	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 **说明** 分库路由结果四舍五入到最接近的整数。	拆分键：2012-12-31 分库：(2012 * 366 + 366) % (8*3) = 6 分库：6 / 3 = 2

建表语法

假设用户的实例里已经分了8个物理库，现有一个业务想按年日进行分库。要求同一天的数据都能落在同一张分表，并且两年以内的每一天都能单独对应一张分表，查询时带上分库分表键后能直接将查询落在某个物理分库的某个物理分表。

用户这时就可以使用YYYYDD分库函数来解决：业务要求两年以内的每天都对应一张分表（即一天一张表），由于一年最多有366天，所以两年至少需要创建732个物理分表才能满足用户的场景。用户的DDM有8个分库，所以每个分库应该建92张物理分表（732 / 8 = 91.5，取整为 92，分表数建议是分库数的整数倍）。建表语法如下所示：

```
create table test_yyyydd_tb (
        id int,
        name varchar(30) DEFAULT NULL,
        create_time datetime DEFAULT NULL,
        primary key(id)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    dbpartition by YYYYDD(create_time)
    tbpartition by YYYYDD(create_time) tbpartitions 92;
```

只分库的建表语法：

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

注意事项

- YYYYDD算法不支持对于每一个年日都独立对应一张分库，分库分表时必须固定分表数目。
- 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。

19.2.4.10 YYYYWEEK 按年周哈希

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，建议该函数与tbpartition YYYYWEEK(ShardKey) 联合使用。

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(‘2012-12-31 12:12:12’) 等价于 (2013 * 54 + 1) % D（D是分库数目/分表数）。

说明

- 此处“2012-12-31”是2013年第一周，所以为"2013* 54+ 1"。
- YYYYWEEK的具体用法请参见**YEARWEEK函数**。

算法计算方式

表 19-15 算法计算方式

条件	算法	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 **说明** 分库路由结果四舍五入到最接近的整数。	拆分键：2012-12-31 分库 ： (2013 * 54 + 1) % (8*3) = 7 分库 ： 7 / 3 = 2

建表语法

假设用户的实例里已经分了8个物理库, 现有一个业务想按年周进行分库。要求同一周的数据都能落在同一张分表，并且两年以内的每个周都单独对应一张分表，查询时带上分库分表键后能直接将查询落在某个物理分库的某个物理分表。

用户这时就可以使用YYYYWEEK分库函数来解决：业务要求两年以内的每个周都对应一张分表（就是一个周一张表），由于一年有近 53 个周(四舍五入)，所以两年至少需要创建 106个物理分表才能满足用户的场景。而用户的DDM有8个分库，所以每个分库应该建14张物理分表（14 * 8 = 112 > 106，分表数建议是分库数的整数倍）。建表语法如下所示：

```
create table test_yyyymm_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYWEEK(create_time)
tbpartition by YYYYWEEK(create_time) tbpartitions 14;
```

只分库的建表语法：

```
create table YYYYWEEK(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
```



```
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYWEEK(create_time);
```

注意事项

- YYYYWEEK算法不支持对于每一个年周都独立对应一张分库，分库分表时必须固定分表数目。
- 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。

19.2.4.11 HASH 算法

适用场景

适用于需要将数据均匀分布的场景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

- 拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL(支持精度为0的情况)）或字符串类型(CHAR,VARCHAR)。如果使用日期函数结合hash算法，拆分键的数据类型必须是DATE / DATETIME / TIMESTAMP其中之一。
- 在SQL语句中对数字类型拆分键设置值时不要进行类型转换，类型转换可能造成路由计算失败后路由至默认分片，造成目标数据查询不到。

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，根据计算结果落到某个范围进行路由。

算法计算方式

方式一：拆分键非日期类型

表 19-16 拆分键非日期类型

条件	算法	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 19-17 支持的日期函数

日期函数	算法	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 **说明** 关于一年中的第几周的定义请参见 **WEEKOFYEAR(*date*)**。	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 19-18 拆分键是日期类型

条件	算法	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(分表拆分键值)) % 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

无。

19.2.4.12 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，如果使用日期函数，拆分键的数据类型必须是date、datetime、timestamp其一。

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。

建表时要设定元数据，假如逻辑库分8个分片，则元数据可以设定的范围为：1-2=0，3-4=1，5-6=2，7-8=3，9-10=4，11-12=5，13-14=6，default=7。根据拆分键的值在某个范围路由到对应的分片上。

算法计算方式

方式一：拆分键是整型

表 19-19 拆分键是整型时的计算方式

条件	算法	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	分库： 拆分值为3属于3-4=1，则路由到1分片

方式二：拆分键是日期类型

表 19-20 支持的日期函数

日期函数	算法	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 **说明** 关于一年中的第几周的定义请参见 **WEEKOFYEAR(*date*)**。	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 19-21 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	分库： 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

无。

19.3 DML

19.3.1 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。
- INSERT操作不支持datetime（YYYY-MM-DD HH:MM:SS）中“YYYY”取值1582年及之前年份。
- INSERT操作不支持插入拆分键值为DEFAULT关键字。
- 拆分表执行INSERT操作时如果指定了自增值，只影响该插入数据的自增值。后续数据插入时如果不指定自增值，仍以原自增值为基础进行自增。
- 不支持在VALUES中调用REPEAT函数时引用表中的列作为参数。

  例如：

  ```
  INSERT INTO T(NAME) VALUES(REPEAT(ID,3));
  ```

- 使用INSERT DUPLICATE...更新拆分键时仅支持常量，不支持VALUES、LAST_INSERT_ID等函数或运算表达式。
- 不支持INSERT DUPLICATE语句更新含GSI的拆分表。

使用限制

- 如果INSERT语句中拆分键的值是非法的，数据默认会路由到0号分库或者0号分表。
- 不建议在INSERT语句中使用VERSION、DATABASE、USER函数，此类函数结果受语句是否下推至DN节点执行影响，执行结果可能不符合预期。
- 不建议INSERT DUPLICATE语句更新拆分键，此类方法直接下推DN，执行结果可能不符合预期。

19.3.2 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。

使用限制

- 如果REPLACE语句中拆分键的值是非法的，数据默认会路由到0号分库或者0号分表。
- 不建议在REPLACE语句中使用VERSION、DATABASE、USER函数，此类函数结果受语句是否下推至DN节点执行影响，执行结果可能不符合预期。

19.3.3 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）。

19.3.4 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。
- UPDATE操作不支持datetime（YYYY-MM-DD HH:MM:SS）中“YYYY”取值1582年及之前年份。
- UPDATE操作不支持更新拆分键值为DEFAULT关键字。
- UPDATE不支持在一个语句中对同一字段重复更新。
- UPDATE不支持关联更新拆分键。
- 不支持含有JSON类型字段的二级拆分表进行带子查询的拆分键更新。
- UPDATE不支持自关联更新。
- 关联更新中，不支持在目标列的赋值语句或表达式中引用其它目标列，将造成更新结果不符合预期。

  例如：

  ```
  update tbl_1 a,tbl_2 b set a.name=concat(b.name,'aaaa'),b.name=concat(a.name,'bbbb') on a.id=b.id。
  ```

- 关联更新不支持不带关联条件的Join。

19.3.5 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 19-22 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	指您将从某一个或多个表中查询。
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查询（当查询的表为分片表时，无法得到预期的结果）。
- 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。
- select后输出字段要和表字段顺序保持一致，如果JOIN查询，将所有参与JOIN的字段都添加到select后的查询字段列表中，这样能提升SELECT查询效率。
- 对于UNION中的每个SELECT, DDM 暂不支持使用多个同名的列。

  例如：

  如下SQL的SELECT中存在重复的列名。

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

19.3.6 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;
```

19.3.7 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。

19.3.8 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。

19.3.9 不支持的 DML 语法列举

不支持的 DML 语法

表 19-23 DML 的语法限制

DML语法	使用限制
DELETE语句	不支持PARTITION子句。

DML语法	使用限制
UPDATE语句	不支持跨分片子查询。
SELECT语句	支持ORDER BY语句，不支持类似ORDER BY FIELD(id,1,2,3)这种自定义排序。 **说明** 自定义排序语句中，如果查询的表为分片表，那么无法得到预期的结果。

19.3.10 支持的系统库查询

表 19-24 系统库查询

DML语法	使用限制
系统库查询	支持以下系统库查询： 版本查询：**SELECT version()** ● information_schema.SCHEMA_PRIVILEGES ● information_schema.TABLE_PRIVILEGES ● information_schema.USER_PRIVILEGES ● information_schema.SCHEMATA ● information_schema.tables ● information_schema.columns 索引查询：**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。

19.4 Online DDL

DDM支持通用的Online DDL操作：增加字段、删除字段、修改字段、设置默认值、修改编码、修改表名等。

Online DDL主要功能为在对应DDL上，提供ALGORITHM、LOCK显示声明的支持，并提供透传至后端数据库节点能力（此功能需满足DDM内核版本大于等于3.1.0版本）。

当DDM实例关联的DN实例为MySQL5.7版本时，Online DDL操作支持以下语法：

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...]

alter_option: {
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
```



```
  | ADD [COLUMN] (col_name column_definition,...)
  | DROP [COLUMN] col_name
  | ALTER [COLUMN] col_name {
        SET DEFAULT {literal | (expr)}
      | DROP DEFAULT
    }
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST | AFTER col_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | RENAME [TO | AS] new_tbl_name

  | ALGORITHM [=] {DEFAULT | INPLACE | COPY}
  | LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
}
```

当DDM实例关联的DN实例为MySQL8.0版本时，Online DDL操作支持以下语法：

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...]

alter_option: {
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | ADD [COLUMN] (col_name column_definition,...)
  | DROP [COLUMN] col_name
  | ALTER [COLUMN] col_name {
        SET DEFAULT {literal | (expr)}
      | DROP DEFAULT
    }
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST | AFTER col_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | RENAME COLUMN old_col_name TO new_col_name
  | RENAME [TO | AS] new_tbl_name

  | ALGORITHM [=] {DEFAULT | INSTANT | INPLACE | COPY}
  | LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
}
```

⚠ 注意

在使用Online DDL语法时，最终ALGORITHM、LOCK选项生效位置为后端数据库，当后端有多个数据库时，DDM表现出的并发性可能与参数值不一致。

使用示例

增加字段

```
# 向表t2中增加字段x，类型为int，Online DDL 算法为inplace，锁为NONE
ALTER TABLE t2 ADD COLUMN x INT, ALGORITHM=INPLACE, LOCK=NONE;
```

修改字段

```
# 修改表t2的字段x，修改字段类型为varchar(255)，Online DDL 算法为copy，锁为shared
ALTER TABLE t2 MODIFY x VARCHAR(255), ALGORITHM=COPY, LOCK=SHARED;
```

修改编码


```
# 修改表t2编码为utf8，排序算法为utf8_bin，Online DDL 算法为copy，锁为shared
ALTER TABLE t2 CHARACTER SET utf8 COLLATE utf8_bin, ALGORITHM=COPY, LOCK=SHARED;
```

19.5 函数

支持的函数

表 19-25 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;
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;

函数表达式	示例
*	SELECT order_num, product_id, quantity, item_price, quantity*item_price AS expanded_price FROM OrderItems WHERE order_num BETWEEN 000009 AND 000028 ORDER BY order_num;
/	SELECT 3 * 2+5-100/50;
UPPER	SELECT id, product_id, UPPER(product_name) FROM Products WHERE id > 10 ORDER BY product_id;
LOWER	SELECT id, product_id, LOWER(product_name) FROM Products WHERE id <= 10 ORDER BY product_id;
SOUNDEX	SELECT * FROM Vendors WHERE SOUNDEX(vendor_name) = SOUNDEX('test') ORDER BY vendor_name;
IFNULL	SELECT IFNULL(product_id, 0) FROM Products;

表 19-26 时间日期函数

函数表达式	示例	支持范围
DAY()	SELECT * FROM TAB_DATE WHERE DAY(date)=21; SELECT * FROM TAB_DATE WHERE date='2018-12-21'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22');	-
MONTH()	SELECT * FROM TAB_DATE WHERE MONTH(date)=12; SELECT * FROM TAB_DATE WHERE date='2018-12-21'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22');	-
YEAR()	SELECT * FROM TAB_DATE WHERE YEAR(date)=2018; SELECT * FROM TAB_DATE WHERE date='2018-12-21'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2018-05-22');	-
TIME()	SELECT * FROM TAB_DATE WHERE TIME(date)='01:02:03'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的时间、日期时间表达式或字符串。

函数表达式	示例	支持范围
TIME_TO_SEC()	SELECT * FROM TAB_DATE WHERE TIME_TO_SEC(date)=3603; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:00:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:00:03');	参数需为合法的时间、日期时间表达式或字符串。
SEC_TO_TIME()	SELECT * FROM TAB_DATE WHERE SEC_TO_TIME(date)='00:01:00'; SELECT * FROM TAB_DATE WHERE date=60; INSERT INTO TAB_DATE(id,date) VALUES(1,60);	参数需为数值或可转化为数值的字符串。
SECOND()	SELECT * FROM TAB_DATE WHERE SECOND(date)=3; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的时间、日期时间表达式或字符串。
MINUTE()	SELECT * FROM TAB_DATE WHERE MINUTE(date)=2; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的时间、日期时间表达式或字符串。
HOUR()	SELECT * FROM TAB_DATE WHERE HOUR(date)=1; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的时间、日期时间表达式或字符串。
DAYNAME()	SELECT * FROM TAB_DATE WHERE DAYNAME(date)='Friday'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的日期、日期时间表达式或字符串。
MONTHNAME()	SELECT * FROM TAB_DATE WHERE MONTHNAME(date)='January'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的日期、日期时间表达式或字符串。

函数表达式	示例	支持范围
LAST_DAY()	SELECT * FROM TAB_DATE WHERE LAST_DAY(date)='2021-01-31'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的日期、日期时间表达式或字符串。
DAYOFWEEK()	SELECT * FROM TAB_DATE WHERE DAYOFWEEK(date)=6; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的日期、日期时间表达式或字符串。
DAYOFMONTH()	SELECT * FROM TAB_DATE WHERE DAYOFWEEK(date)=6; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	参数需为合法的日期、日期时间表达式或字符串。
DAYOFYEAR()	SELECT * FROM TAB_DATE WHERE DAYOFYEAR(date)=365; SELECT * FROM TAB_DATE WHERE date='2021-12-31 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-12-31 01:02:03');	参数需为合法的日期、日期时间表达式或字符串。
WEEKOFYEAR()	SELECT * FROM TAB_DATE WHERE WEEKOFYEAR(date)=53; SELECT * FROM TAB_DATE WHERE date='2021-12-31 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-12-31 01:02:03');	参数需为合法的日期、日期时间表达式或字符串。

函数表达式	示例	支持范围
DATE_ADD(date, INTERVAL expr unit)	SELECT * FROM TAB_DATE WHERE DATE_ADD(date，INTERVAL 1 YEAR)='2022-01-01 01:02:03'; SELECT * FROM TAB_DATE WHERE date='2021-01-01 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-01 01:02:03');	• date参数需为合法的时间、日期、日期时间表达式或字符串。 • expr为从date开始加减运算的间隔值，要求为整数或可转化为整数的字符串。 • unit为单位，暂时可选SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR。 • 当date为日期类型时，"1000-01-01"是下边界。在运算时如果存在越界行为，运算可能出错。 • date参数最高支持毫秒精度。
DATE_SUB(date, INTERVAL expr unit)	SELECT * FROM TAB_DATE WHERE DATE_SUB(date，INTERVAL -1 DAY)='2021-01-02 01:02:03'; SELECT * FROM TAB_DATE WHERE date='2021-01-02 01:02:03'; INSERT INTO TAB_DATE(id,date) VALUES(1,'2021-01-02 01:02:03');	• date参数需为合法的时间、日期、日期时间表达式或字符串。 • expr为从date开始加减运算的间隔值，要求为整数或可转化为整数的字符串。 • unit为单位，暂时可选SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR。 • 当date为日期类型时，"1000-01-01"是下边界。在运算时如果存在越界行为，运算可能出错。 • date参数最高支持毫秒精度。

表 19-27 数学函数

函数表达式	示例	支持范围
SQRT()	SELECT id, product_price, SQRT(product_price) AS price_sqrt FROM Products WHERE product_price < 4000 ORDER BY product_price;	-
AVG()	SELECT AVG(product_price) AS avg_product FROM Products;	-
COUNT()	SELECT COUNT(*) AS num_product FROM Products;	-
MAX()	SELECT id, product_id, product_name, MAX(product_price) AS max_price FROM Products ORDER BY id;	-
MIN()	SELECT id, product_id, product_name, MIN(product_price) AS min_price FROM Products ORDER BY id;	-
SUM()	SELECT SUM(product_price) AS sum_product FROM Products;	-
ROUND()	SELECT ROUND(product_price) AS round_product FROM Products;	参数需为数值或可转化为数值的字符串。
SIN()	SELECT SIN(x) AS sin_x FROM math_tbl;	参数需为数值或可转化为数值的字符串。
COS()	SELECT COS(x) AS cos_x FROM math_tbl;	参数需为数值或可转化为数值的字符串。
TAN()	SELECT TAN(x) AS tan_x FROM math_tbl;	参数需为数值或可转化为数值的字符串。
COT()	SELECT COT(x) AS cot_x FROM math_tbl;	参数需为数值或可转化为数值的字符串。
FLOOR()	SELECT FLOOR(product_price) AS floor_product FROM Products;	参数需为数值或可转化为数值的字符串。
CEILING()	SELECT CEILING(product_price) AS ceiling_product FROM Products;	参数需为数值或可转化为数值的字符串。
ABS()	SELECT ABS(x) AS abs_x FROM math_tbl;	参数的范围为-9223372036854775807到9223372036854775807，超过此范围会报错。
LOG()	SELECT LOG(x) AS log_x FROM math_tbl;	参数需为大于0的数值或可转化为相应数值的字符串。

函数表达式	示例	支持范围
LN()	SELECT LN(x) AS ln_x FROM math_tbl;	参数需为大于0的数值或可转化为相应数值的字符串。
EXP()	SELECT EXP(x) AS exp_x FROM math_tbl;	参数的范围为-∞到709，超过此范围会报错。

表 19-28 字符串函数

函数表达式	示例	支持范围
TRIM()	SELECT TRIM(' hello, world ') AS trim_character FROM Character;	参数需为字符串相关类型。

说明

使用前，请先查看该函数是否在支持范围，建议使用支持范围内的函数。如果在范围外使用，返回结果与MySQL中的返回结果可能不一致。

不支持的函数

表 19-29 函数的限制

函数	限制条件
ROW_COUNT()	DDM 暂不支持ROW_COUNT()函数。
COMPRESS()	DDM 暂不支持COMPRESS()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数。
SHA()	DDM 暂不支持SHA()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数。
SHA1()	DDM 暂不支持SHA1()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数。
MD5()	DDM 暂不支持MD5()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数。
AES_ENCRYPT()	DDM 暂不支持AES_ENCRYPT()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数。
AES_DECRYPT()	DDM 暂不支持AES_DECRYPT()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数。

函数	限制条件
YEARWEEK()	DDM 暂不支持YEARWEEK()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数。
TIME_FORMAT()	DDM暂不支持TIME_FORMAT()函数。如果无法确认函数是否能下推到RDS，请不要使用该函数，建议使用DATE_FORMAT()函数。

19.6 使用限制

- 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CHECK/CHECKSUM/OPTIMIZE/REPAIR TABLE。
- 不支持session变量赋值与查询。

  例如：

  ```
  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 ' /*...*/'。
- REPEAT函数结果长度最大限制为1000000（适用于3.0.9版本及以上）。

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）

19.7 实用 SQL 语句

19.7.1 CHECK TABLE

19.7.1.1 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。

19.7.1.2 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：

如果返回结果集为空, 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)
```

如果返回结果集不为空, 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。

19.7.2 SHOW RULE

命令格式：

- 查看数据库下每一个逻辑表的拆分情况。

  show rule

```
mysql> show rule;
| ID   | TABLE_NAME                      | BROADCAST | DB_PARTITION_KEY   | DB_PARTITION_POLICY | DB_PARTITION_COUNT | DB_PARTITION_OFFSET | PARTITION_RANGE
      | TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT | TB_PARTITION_OFFSET |
|    0 | history                   | 0      |        |        |        1 |        |
      |        |        |        1 |        |
|    1 | tbtest_hash_forerror      | 0      |        |        |        1 |        |
      |        |        |        1 |        |
|    2 | single_table_1            | 0      |        |        |        1 |        |
      |        |        |        1 |        |
|    3 | carrier                   | 0      |        |        |        1 |        |
      |        |        |        1 |        |
|    4 | employee3                 | 0      |        |        |        1 |        |
      |        |        |        1 |        |
|    5 | employee4                 | 0      |        |        |        1 |        |
      |        |        |        1 |        |
|    6 | supplier                  | 1      |        |        |        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 |        |
```

- 查看数据库下指定逻辑表的拆分情况。

  show rule from *<table_name>*

```
mysql> show rule from range_id_yd_datetime_3_tpt_1 ;
| ID   | TABLE_NAME                   | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT | DB_PARTITION_OFFSET | PARTITION_RANGE
      | TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT | TB_PARTITION_OFFSET |
|    1 | range_id_yd_datetime_3_tpt_1 | 0         | id_col           | range               |                  8 |                     |  0-10=0, 11-20=1, 21-30=2, 31-40=3, 41-
50=4, 51-60=5, 61-70=6, 71-120=7, default=3 | datetime_col     | yyyydd              |                  3 |                     |
1 row in set (0.00 sec)
```

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。

19.7.3 SHOW TOPOLOGY

命令格式：

查看数据库下指定逻辑表的物理分布情况。

show topology from *<table_name>*

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 ）。

19.7.4 SHOW DATA NODE

命令格式：

show data node

描述：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。

19.7.5 TRUNCATE TABLE

19.7.5.1 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。

19.7.5.2 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。

删除前示例：

```
mysql> show topology from user_tb;
+-----------------+-----------+-------+-----------+-----------+-----------+
| Rds_instance_id | Host      | Port  | Database  | Table     | Row_count |
+-----------------+-----------+-------+-----------+-----------+-----------+
| shard1          | localhost | 33061 | test_0000 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_0 | 2         |
| shard1          | localhost | 33061 | test_0001 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_1 | 3         |
| shard1          | localhost | 33061 | test_0002 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_1 | 5         |
| shard1          | localhost | 33061 | test_0003 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_2 | 3         |
| shard3          | localhost | 33063 | test_0005 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_3 | 2         |
| shard3          | localhost | 33063 | test_0007 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_3 | 0         |
+-----------------+-----------+-------+-----------+-----------+-----------+
32 rows in set (0.22 sec)

mysql> /*+table = user_tb_3*/truncate table user_tb;
Query OK, 0 rows affected (5.18 sec)
```

删除后示例：

```
mysql> show topology from user_tb;
+-----------------+-----------+-------+-----------+-----------+-----------+
| Rds_instance_id | Host      | Port  | Database  | Table     | Row_count |
+-----------------+-----------+-------+-----------+-----------+-----------+
| shard1          | localhost | 33061 | test_0000 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_0 | 2         |
| shard1          | localhost | 33061 | test_0001 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_1 | 3         |
| shard1          | localhost | 33061 | test_0002 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_1 | 5         |
| shard1          | localhost | 33061 | test_0003 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_2 | 3         |
| shard3          | localhost | 33063 | test_0005 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_3 | 0         |
+-----------------+-----------+-------+-----------+-----------+-----------+
32 rows in set (0.16 sec)
```

19.7.5.3 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。

19.7.5.4 补充说明

HINT对于所有的单表以及全局表失效，只对各种分表起作用。

19.7.6 HINT- ALLOW_ALTER_RERUN

命令格式：

/*+ allow_alter_rerun=true*/*<ALTER TABLE的命令>*

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。

示例：

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

19.7.7 LOAD DATA

标准示例

LOAD DATA LOCAL INFILE '/data/data.txt' IGNORE INTO TABLE test CHARACTER SET 'utf8' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n' (id, sid, asf);

说明

如果数据中包含一些特殊字符，比如分隔符、转义符等，建议用引号扩起来，通过OPTIONALLY ENCLOSED BY '"'指定。

示例：

下面的一段数据中包含了分隔符（,）则需要用引号括起来。

"aab,,,bba,ddd"

如果数据中包含了引号字符，则上述方法可能不会生效，此时可以把数据中引号字符替换成\"。例如：**"aab,,,bba,ddd\"ddd\"bb,ae"**。

- 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，请选择在业务低峰时间导入。
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/data.txt' IGNORE INTO TABLE test CHARACTER SET 'utf8' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n' (id, sid, asf);
mysql> commit;
```

使用限制

- 不支持LOW_PRIORITY。
- 不支持CONCURRENT。
- 不支持PARTITION (partition_name [, partition_name] ...)。
- 不支持LINES STARTING BY 'string'。
- 不支持用户变量。
- ESCAPED BY 只支持'\\'。
- 如果导入数据时没有指定自增键的值，DDM不会填充自增值，自增能力使用的是底层DN的自增能力，因此自增值会重复。
- 如果主键或者唯一索引值经过路由后不在同一张物理表，REPLACE不生效。
- 如果主键或者唯一索引值经过路由后不在同一张物理表，IGNORE不生效。
- 不支持对含有全局二级索引的表执行LOAD DATA的操作。

19.7.8 SHOW PHYSICAL PROCESSLIST

- 命令格式1：展示后端rds的processlist。

  show physical processlist

- 命令格式2：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。

  show physical processlist with info

  命令执行效果如下：

图 19-1 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：RDS的IP地址。

Port：RDS的端口号。

Instance_id：RDS的实例ID。

Type：master指的是读写RDS，readreplica指的是只读RDS。

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

- 命令格式3：结束对应RDS上的执行线程。

  kill physical *<physical_thread_id>@<rds_ip>:<rds_port>*

  <physical_thread_id>：RDS上的执行线程ID，可从命令2的结果集中获取。

  <rds_ip>：RDS的IP地址，可从命令2的结果集中获取。

  <rds_port>：RDS的端口号，可从命令2的结果集中获取。

须知

- 该功能仅支持内核3.0.1及以上的版本。
- 上述的命令需要登录DDM后，在DDM上执行。

19.7.9 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支持以下两种格式：

格式一：

/*!mycat:db_type=*host* ***/**

格式二：

/*+ 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。此方式仅适用于查询功能。

19.7.10 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。

19.7.11 通过 HINT 指定分片直接执行 SQL

DDM提供HINT在一个或多个分片上执行SQL语句。

HINT支持以下两种格式：

- 一个分片上执行SQL：**/*+db=***<physical_db_name>****/** *<your query>***;**
- 多个分片上执行SQL：**/*+db=***{<physical_db_name1>, <physical_db_name2>,<physical_db_name3>.....}****/** *<your query>***;**

示例：

- 一个分片上执行SQL：**/*+db=test_0000*/ select * from t1;**
- 多个分片上执行SQL：**/*+db={test_0001, test_0002, test_0003}*/ select * from t2;**

使用限制：

- 指定多个分片时，physical_db_name不能重复。
- 此HINT只对SELECT/DML/TRUNCATE语句起作用。
- 此HINT仅在文本协议下工作，Prepare协议下无法使用。

19.8 全局序列

19.8.1 全局序列概述

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。
- 对使用DDM自增序列，不允许用户传null值以外的值，当用户不传或传null值时，DDM会默认分配，如果用户手工赋值会有和DDM分配自增键值冲突的风险。

表 19-30 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 连接DDM实例。

连接方法具体请参考**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 *<序列名>*;

说明

- 建议使用bigint型作为自增键的数据类型。tinyint、smallint、mediumint、integer、int数据类型不建议作为自增键的类型，容易越界造成值重复。
- 通过“show sequences”命令可查看自增序列的使用率。如果使用率已达到或接近100%，请不要再插入数据，联系DDM客服人员进行处理。

----结束

删除自增序列

步骤1 连接DDM实例。

连接方法具体请参考**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 *<序列名>*;

drop sequence *DB.****;

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 连接DDM实例。

连接方法具体请参考**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 *<序列名>* **START WITH** *<目标序列起始值>;*

----结束

查询自增序列

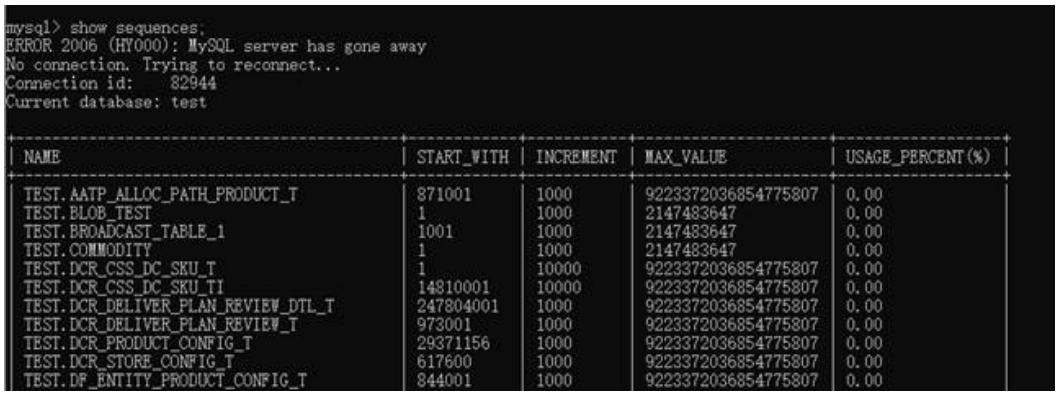
步骤1 连接DDM实例。

连接方法具体请参考**连接DDM实例**。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入如下命令，查看所有序列。

show sequences;



```
mysql> show sequences;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    82944
Current database: test

+-----------------------------------------+-----------+-----------+---------------------+-----------------+
| NAME                                    | START_WITH | INCREMENT | MAX_VALUE           | USAGE_PERCENT(%) |
+-----------------------------------------+-----------+-----------+---------------------+-----------------+
| TEST.AATP_ALLOC_PATH_PRODUCT_T          | 871001    | 1000      | 9223372036854775807 | 0.00            |
| TEST.BLOB_TEST                          | 1         | 1000      | 2147483647          | 0.00            |
| TEST.BROADCAST_TABLE_1                  | 1001      | 1000      | 2147483647          | 0.00            |
| TEST.COMMODITY                          | 1         | 1000      | 2147483647          | 0.00            |
| TEST.DCR_CSS_DC_SKU_T                   | 1         | 10000     | 9223372036854775807 | 0.00            |
| TEST.DCR_CSS_DC_SKU_TI                  | 14810001  | 10000     | 9223372036854775807 | 0.00            |
| TEST.DCR_DELIVER_PLAN_REVIEW_DTL_T      | 247804001 | 1000      | 9223372036854775807 | 0.00            |
| TEST.DCR_DELIVER_PLAN_REVIEW_T          | 973001    | 1000      | 9223372036854775807 | 0.00            |
| TEST.DCR_PRODUCT_CONFIG_T               | 29371156  | 1000      | 9223372036854775807 | 0.00            |
| TEST.DCR_STORE_CONFIG_T                 | 617600    | 1000      | 9223372036854775807 | 0.00            |
| TEST.DF_ENTITY_PRODUCT_CONFIG_T         | 844001    | 1000      | 9223372036854775807 | 0.00            |
```

----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 连接DDM实例。

连接方法具体请参考**连接DDM实例**。

步骤2 连接成功后，打开目标逻辑库。

步骤3 输入如下命令，修改表test的全局序列的cache值。

alter sequence test cache 5000

步骤4 输入如下命令，查看test序列的INCREMENT值即是cache值。

show sequences

----结束

刷新实例所有表自增序列

须知

该功能仅支持内核3.0.4.1以上的版本。

步骤1 连接DDM实例。

连接方法具体请参考**连接DDM实例**。

步骤2 输入如下命令，更改所有逻辑库的所有Sequence。

fresh all sequence start value

----结束

19.8.2 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 连接DDM实例。

连接方法具体请参考**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

19.8.3 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。

前提条件

- 已有两个逻辑库dml_test_1和dml_test_2。
- 逻辑库dml_test_1和dml_test_2中都有表test_seq。

  创建表的方法：

  create table test_seq(col1 bigint,col2 bigint) dbpartition by hash(col1);

操作步骤

步骤1 连接DDM实例。

连接方法具体请参考**连接DDM实例**。

步骤2 打开dml_test_1逻辑库。

use dml_test_1;

步骤3 输入命令创建全局序列。

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。

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)
```

步骤5 打开dml_test_2逻辑库。

use dml_test_2;

步骤6 使用以下语句，实现全局序列在insert语句或者replace语句的使用。

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

19.9 数据库管理语法

支持的数据库管理语法

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

  说明

  如果未搜索到对应的库，请先检查账号的细粒度权限。
- SHOW INDEX FROM
- SHOW VARIABLES

支持的数据库工具命令

- DESC
- USE
- EXPLAIN

说明

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

- 不支持SET修改全局变量。
- 不支持SHOW TRIGGERS语法。
- CHECK TABLE不支持hash和key分区表。
- SHOW WARNINGS、SHOW ERRORS语句不支持LIMIT/COUNT的组合。SHOW语句会随机下发到某个物理分片，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。

19.10 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

  EXECUTE
- 不支持在建表语句中，对索引增加COMMENT形式的注释。

20 配额

操作场景

为防止资源滥用，平台限定了各服务资源的配额，对用户的资源数量和容量做了限制。

如果当前资源配额限制无法满足使用需要，您可以申请扩大配额。

查看配额

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 在页面右上角，选择“资源 > 我的配额”，进入“服务配额”页面。

图 20-1 我的配额

步骤4 您可以在“服务配额”页面，查看分布式数据库中间件各项资源的总配额，以及使用情况。

----结束

申请扩大配额

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 在页面右上角，选择“资源 > 我的配额”，进入“服务配额”页面。

步骤4 单击“申请扩大配额”。

步骤5 在“新建工单”页面，根据您的需求，填写相关参数。

其中，“问题描述”项请填写需要调整的内容和申请原因。

步骤6 填写完毕后，勾选协议并单击“提交”。

----结束

21 常见问题

21.1 DDM 通用类

21.1.1 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。

- 业务数据存储在数据节点分片中，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。

高可用机制

DDM采用多个无状态节点集群式部署模式，通过弹性负载均衡地址提供服务。

- DDM节点自身宕机类故障，对于已建立在故障节点上的连接会断连报错，DDM集群整体服务不受影响，通常情况下可在5秒内将故障节点从集群中剔除。
- 下挂数据节点故障，通常情况下可以在下挂数据节点恢复后30秒内完全恢复正常服务能力。

21.1.2 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实例访问其关联的数据节点。

除了ECS、DDM实例、数据节点必须处于相同VPC之外，还需要安全组分别配置了正确的规则，允许网络访问。

1. 建议ECS、DDM、数据节点配置相同的安全组。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

- 例如ECS、DDM、RDS分别配置了安全组：sg-ECS、sg-DDM、sg-RDS。
- 例如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下图中的入方向规则，以保证客户端能正常访问DDM实例：

图 21-1 ECS 安全组策略

DDM所在安全组需要增加下图中的入方向和出方向规则，以保证能访问数据节点，且被客户端访问。

图 21-2 DDM 安全组入方向配置

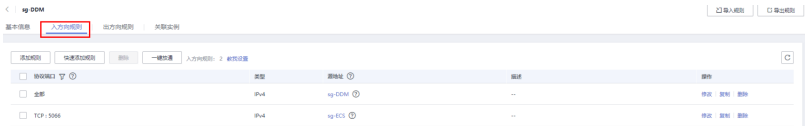
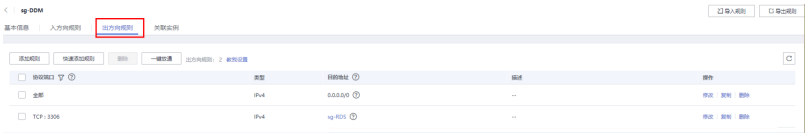


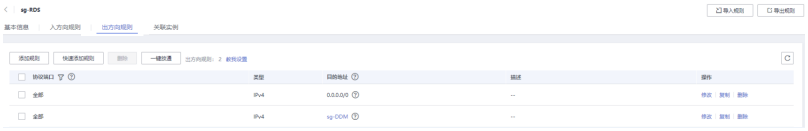
图 21-3 DDM 安全组出方向配置



数据节点所在安全组需要增加下图中的入方向和出方向规则，以保证能被DDM访问。

图 21-4 RDS 安全组入方向配置

图 21-5 RDS 安全组出方向配置



21.1.3 数据库时间与北京时间相差 13 或 14 小时该如何解决

问题现象

数据库时区设置为北京时间时，通过JDBC连接DDM，查询到的时间与北京时间相差13或14小时。

原因分析

JDBC驱动连接DDM时会向DDM查询数据库时区设置，DDM返回时区为CST(中国标准时间)。

CST有4种含义：

- 美国中部时间 Central Standard Time (USA) UTC-06:00
- 澳大利亚中部时间 Central Standard Time (Australia) UTC+09:30
- 中国标准时 China Standard Time UTC+08:00
- 古巴标准时 Cuba Standard Time UTC-04:00

在JDBC驱动中，会将CST时间解析为美国中部时间，与北京时间相差了13或14个小时。

解决方法

在JDBC连接数据库的字符串中添加时区配置项：

jdbc:mysql://*xxx***:3306/***database_name***?serverTimezone=Asia/Shanghai&useUnicode=true&characterEncoding=utf8**

21.1.4 一个 DDM 实例关联的不同数据节点之间是否可以共享数据

一个DDM实例关联的不同数据节点之间数据是互相独立的，无法共享。

21.1.5 DDM 实例关联的数据节点需要满足什么条件

DDM实例关联的数据节点需满足以下条件：

- 数据节点的状态为正常运行中。
- 数据节点和DDM实例处于相同VPC。
- 数据节点没有被其它DDM实例关联。

21.2 DDM 使用类

21.2.1 DDM 如何进行分片

在分布式数据库中，可以通过分片存储方式，轻松解决大数据量单表容量达到单机数据库存储上限的瓶颈，因此创建逻辑库和逻辑表时，需要根据实际情况确定逻辑表是否进行分片以及逻辑表的分片规则。

说明

分片存储后，需要尽量避免跨库JOIN操作带来的性能与资源消耗问题。

- 逻辑表是否分片

  DDM逻辑表支持全局表、拆分表、单表三种类型。用户可以按照数据表的实际使用需求，选择最合适的逻辑表类型创建，实际操作请参考**创建表**。

  - 单表只在第一个分片创建表以及存储数据。
  - 全局表在每一个分片创建表并且存储全量数据。
  - 拆分表在每一个分片创建表，数据按照拆分规则分散存储在分片中。

- 逻辑表的分片规则

  逻辑表的拆分键选择非常重要。建议按实际业务场景选择拆分键，不同逻辑表，如果具有E-R关系，建议选择相同字段做拆分键，避免跨库JOIN操作。

在实际使用中，请结合以下建议评估是否进行分片：

- 数据量在1000万条以下的表，不建议分片。
- 数据量在1000万条以上的表，建议分片。将数据分片存储后，既能解决单张表容量过大带来的性能瓶颈，同时提高并发支持。注意要选择合适的拆分键，提前做好规划。
- 业务读取尽量少用多表JOIN，同一个事务避免跨分片。
- 查询条件尽量带上拆分键，避免全拆分表扫描。

21.2.2 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。如果为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。

21.2.3 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。

建议版本保持一致。

21.2.4 导入数据到 DDM 过程中出现主键重复

在DDM中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。

21.2.5 如何处理数据迁移过程中自增列报错：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
```

21.2.6 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。

21.2.7 如何处理 DDM 逻辑库与 RDS 实例的先后关系

DDM逻辑库与关联的RDS强相关，不允许直接删除关联的RDS，这会导致业务不可用且逻辑库也会删除失败。如果需要删除，先删除逻辑库再删除RDS。

21.2.8 DDM 逻辑库删除后，数据节点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

如果某些DDM实例和账户不需要了，可以进行手动删除。

具体请参考**删除实例**和**删除账号**。

21.3 SQL 语法类

21.3.1 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。

21.3.2 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。

21.3.3 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。

21.3.4 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。

21.4 RDS 相关类

21.4.1 数据库表名是否区分大小写

DDM默认对databaseName、tableName、columnName、columnValue不区分大小写。

21.4.2 RDS for MySQL 哪些高危操作会影响 DDM

RDS for MySQL相关高危操作如**表21-1**所示。

表 21-1 RDS for MySQL 高危操作

操作类别	操作	操作影响
RDS for MySQL控制台操作类	删除RDS for MySQL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。
	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 for MySQL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 for MySQL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。

操作类别	操作	操作影响
	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。

21.4.3 如何处理表中存在主键重复的数据

场景

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。

处理方法

步骤1 登录云服务管理控制台。

步骤2 在RDS for MySQL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 for MySQL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

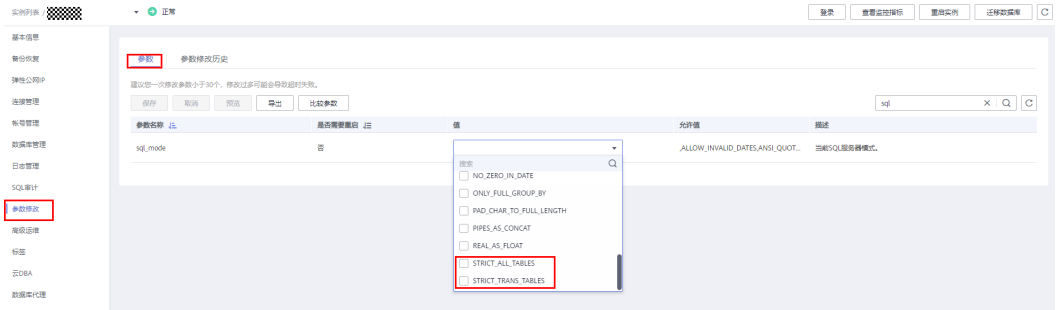
“STRICT_ALL_TABLES”和“STRICT_TRANS_TABLES”方式属于严格模式。严格模式控制MySQL如何处理非法或丢失的输入值。

- 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，如果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**。

图 21-6 修改实例参数



步骤5 在“DDM实例管理”页面，重启DDM实例。

----结束

21.4.4 如何通过 show full innodb status 指令查询 RDS for MySQL 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。

21.4.5 如何选择数据节点 RDS for MySQL 的规格

数据节点RDS for MySQL的规格建议不小于DDM的规格，否则会产生木桶效应，影响性能。

示例：

Q：例如DDM是8核16GB，那么RDS for MySQL的规格应该选多少？应该选择主备还是单机架构?

A：根据最佳实践结果，一般建议DDM和RDS for MySQL的资源比保持在1:2。如果DDM是8核16GB，需要选择2个或以上8核16GB的RDS for MySQL实例，RDS for MySQL的具体数量还需参考业务整体数据量综合考虑。

对于生产业务使用的实例，RDS for MySQL需选择主备架构来保证高可用。

21.5 连接管理类

21.5.1 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 -h 127.0.0.1 -P 5066 -D database --default-character-set=utf8 -u ddmuser –p password
```

21.6 资源冻结/释放/删除/退订

DDM 资源为什么被释放了?

客户在华为云购买产品后，如果没有及时的进行续费或充值，将进入宽限期。如宽限期满仍未续费或充值，将进入保留期。在保留期内资源将停止服务。保留期满仍未续费或充值，存储在云服务中的数据将被删除、云服务资源将被释放。请参见**资源停止服务或逾期释放说明**。

DDM 资源为什么被冻结了?

资源冻结的类型有多种，最常见类型为欠费冻结。

实例被冻结了，还可以备份数据吗?

不支持，如果是欠费冻结，需要您先续费解冻DDM实例后才能备份数据。

怎样将资源解冻?

欠费冻结：用户可通过续费或充值来解冻资源，恢复DDM正常使用。欠费冻结的DDM允许续费、释放 或删除；已经到期的包周期DDM不能发起退订，未到期的包周期DDM可以退订。

冻结、解冻、释放资源时对业务的影响

- 资源冻结时：
  - 资源将被限制访问和使用，会导致您的业务中断。例如DDM被冻结时，会使得用户无法再连接至数据库。
  - 包周期资源被冻结后，将被限制进行变更操作。
  - 资源被冻结后，可以手动进行退订/删除。
- DDM底层的DN节点被冻结会导致DDM与DN节点无法正常通信，进而使DDM功能不可用。
- 资源解冻时：资源将被解除限制，用户可以连接至数据库。
- 资源释放时：资源将被释放，实例将被删除。

怎样续费?

包年/包月方式购买的DDM到期后，请在管理控制台**续费管理**页面进行续费操作。详细操作请参考**续费管理**。

资源被释放了能否恢复？/退订错了可以找回吗?

实例被删除，无法找回。

退订资源前请一定要仔细确认资源信息。如果退订错了建议重新购买使用。

怎样删除 DDM 实例?

- 按需实例，请参见**删除按需实例**。