ALIBABA CLOUD

阿里云

云原生数据仓库AnalyticDB MySQL版
分析型数据库MySQL版2.0

文档版本：20240725

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
⚠ 警告	适用于可能会产生风险的场景。介绍用户在操作前就必须充分了解的信息、操作前必须要注意的事项或已具备的条件。	⚠ 警告 重启操作将导致业务短暂中断，建议您在业务低峰期执行重启操作，或确保已完成数据备份。如有必要，请联系阿里云技术支持提供协助。
ⓘ 重要	在操作前需要用户了解的提示信息、补充信息、注意事项、限制信息等。	ⓘ 重要 再次登录系统时，您需要修改登录账户的初始密码。
? 说明	用于额外的补充说明、最佳实践、窍门等，不是用户必须了解的信息。	? 说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.2.0版产品定价

1.1. 计费方式（2.0版）

云原生数据仓库 AnalyticDB MySQL 版2.0集群支持如下两种计费方式。

包月套餐	• 也称为预付费，即在新建数据库时支付费用。 • 适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。 • 包年包月数据库到期后自动释放，无需手动释放。若用户在合同期内退订包年包月的数据库，需遵循用户提前退订产品之退款规则。 • 包年包月无法变更为按量付费。
按量付费	• 也称为后付费，即每小时生成一个收费订单，并按照生成订单时的数据库规格从阿里云账号扣费。 • 适合短期需求，如测试阶段，用完可立即删除数据库，节省费用。待资源用量稳定后，可将按量付费变更为包月套餐。

价格

关于价格，请参见分析型数据库MySQL版价格详情。

1.2. 变更配置计费规则（2.0版）

变更数据库配置时的计费规则如下表所示。

包年包月（包月套餐）	• 合同期内变配 升级数据库配置所需费用 =（升级后数据库每天的价格 - 升级前数据库每天的价格）× 升级当天到服务到期日的剩余天数。 ◦ 若升级当天至服务到期日的剩余天数小于365天，则升级后数据库每天的价格为包月价格。 ◦ 若升级当天至服务到期日的剩余天数大于（含）365天，则升级后数据库每天的价格为包年价格。 • 到期时续费并变配 按照新数据库配置和包年或包月的时间进行计费。
按量付费	数据库变配后，计费规则不变，仍然是每小时扣费一次，按扣费时的数据库配置计费。

相关文档

变更配置

1.3. 到期或欠费的影响（2.0版）

数据库类型	数据库状态
包年包月数据库	• 数据库到期后第1天到第7天，数据库处于锁定状态，无法被访问。 • 数据库到期后第8天，数据库的计算资源被释放，数据不再保留。
按量付费数据库	• 阿里云账号欠费后，该账号下所有按量付费数据库将会变成欠费状态。 • 欠费后的第1天，数据库正常运行、正常收费。 • 欠费后的第2天，数据库处于锁定状态，无法被访问。 • 欠费后第9天，数据库的计算资源被释放，数据不再保留。

以上所有动作都有短信、邮件和站内信的方式提醒，如何配置提醒参见消息接收管理设置。

相关文档

- 自动续费
- 手动续费

1.4. 续费（2.0版）

续费是指延长包年包月集群的使用时间，续费的费用和新购集群费用相同。按量付费集群没有到期时间，无需续费，您只需保证阿里云账号的余额充足即可。

> **说明** 建议您开通自动续费，以免忘记续费而导致业务中断。

费用

续费集群的费用与新购集群相同。

如果包年包月的集群中包含按量付费的计费项，在给集群续费的同时，您也需要保证阿里云账号的余额充足。

续费方式

- 自动续费
- 手动续费

1.5. 查看消费明细（2.0版）

您可以通过阿里云管理控制台查看云原生数据仓库 AnalyticDB MySQL 版2.0集群的收费明细。

1. 登录阿里云管理控制台。
2. 在页面右上方，选择**费用** > **用户中心**。
3. 单击左侧导航栏**账单管理**。
4. 在**账单管理**页面，您可以通过以下形式查看消费详情。

- 月账单概览：按账期（即月份）查看自身的消费汇总信息。查看最近12个月的云产品消费概览，即每个产品一条消费汇总信息。
- 账单流水：按照订单号/账单号查看最近12个月的云产品消费流水信息，包括每一笔订单和每个计费周期的账单消费信息。
- 账单明细：查看最近12个月的云产品详细消费信息，例如商品的计费项、用量、价格以及资源包抵扣等。

2.2.0版快速入门

2.1. 快速入门综述

分析型数据库MySQL版（AnalyticDB for MySQL）是一种高并发低延时的PB级实时数据仓库，全面兼容MySQL协议以及SQL 2003语法标准，可以毫秒级针对万亿级数据进行即时的多维分析透视和业务探索。

前提条件

- 先注册阿里云账号。
- 如果以按量付费方式购买AnalyticDB for MySQL，请确保您的阿里云账户余额大于等于100元。

操作流程介绍

快速掌握AnalyticDB for MySQL的基本使用流程，通常需要如下步骤：

1. 购买AnalyticDB MySQL版2.0服务
2. 快速使用AnalyticDB MySQL版2.0服务
3. 关闭AnalyticDB MySQL版2.0服务

购买AnalyticDB MySQL版2.0服务

通过阿里云账号登录分析型数据管理控制台购买AnalyticDB for MySQL服务。

1. **选地域和可用区**

   您可以选择在任何一个AnalyticDB for MySQL开服的地域购买AnalyticDB for MySQL资源，建议选择距离您的业务最近的地域，从而提升访问速度。目前已开服的地域如下：

   中国内地与中国香港已经开服的地域有华北（北京、张家口）、华东（上海、杭州）、华南（深圳）、中国香港。

   国外已经开服的地域有新加坡、美国（硅谷）。

   可用区是指在同一地域中，例如华东1（杭州），电力、网络隔离的物理区域，可用区之间内网互通，可用区内网络延时更小。

2. **选ECU类型**

   弹性计算单元（Elastic compute units 简称ECU），是AnalyticDB for MySQL用来衡量数据库计算能力的元单位。ECU分为高性能和大存储两种类型。

   - **高性能**：以字母C或者H开头的ECU为高性能集群，数据全部存储在SSD磁盘中。 适用于对性能要求高、查询并发高的业务场景。
   - **大存储**：以字母S开头的ECU为大存储集群，采用SSD或HDD分层存储架构，热点数据存储在SSD磁盘中，冷数据存储在HDD磁盘中。 适用于并发稍低、性能要求不高（可接受数据查询响应时间受超过10秒以上）的业务场景。

   > **说明**
   > - AnalyticDB for MySQL仅支持相同类型的ECU之间自由变更ECU配置，即只能在C4和C8之间、或者S2N和S8N之间自由升配或降配、扩容或缩容。
   > - AnalyticDB for MySQL不支持在两种ECU类型之间相互转换，请根据业务需要慎重选择ECU类型。

3. **选ECU数量**

   一个AnalyticDB for MySQL集群最少需要购买2个ECU，且ECU的个数必须为偶数，请根据业务评估合理选择ECU个数。

   在后续使用过程中，AnalyticDB for MySQL支持在不影响业务正常运行的情况下在线扩容、缩容，即变更ECU个数。更多详情，请参见变更ECU配置。

4. **设置数据库名字**

   数据库名是AnalyticDB for MySQL的唯一标识，全局唯一，否则将提示您数据库已经存在，无法继续创建AnalyticDB for MySQL集群。

   数据库名的命名规则：以小写字符开头，可包含字母、数字以及下划线（_），但不能包含连续两个及以上的下划线（_），长度不超过64个字符。

5. **选择购买时长（包年包月集群）**

   以包年包月方式购买AnalyticDB for MySQL时，需要指定购买时长，时长越长，优惠越多。

   > **说明**
   > 建议您在购买时开通自动续费，以免忘记续费而导致业务中断。

快速使用AnalyticDB MySQL版2.0服务

AnalyticDB for MySQL全面兼容MySQL协议和SQL 2003，成功购买AnalyticDB for MySQL后，您就可以像使用MySQL数据库一样快速使用AnalyticDB for MySQL。

1. **创建表组**

   表组是一系列可发生关联的数据表的集合，AnalyticDB for MySQL为了管理相关联的数据表，引入了表组的概念。如何创建表组，请参见创建表组。

2. **创建表**

   AnalyticDB for MySQL的表分为维度表和普通表，表的详细介绍，请参见基本概念。如何创建表，请参见创建表。

3. **创建VPC**

   AnalyticDB for MySQL支持经典网络和专有网络（Virtual Private Cloud，简称VPC）两种网络类型。购买AnalyticDB for MySQL时默认只能创建经典网络，您可以在使用AnalyticDB for MySQL过程中创建VPC，详细操作步骤，请参见网络类型。

   VPC是逻辑隔离的私有网络，您可以自定义网络拓扑和IP地址，支持通过专线连接。相对经典网络而言，VPC具有更高的安全性和灵活性。

4. **连接**AnalyticDB for MySQL

   您可以通过分析型数据管理控制台连接AnalyticDB for MySQL，控制台内置DMS客户端，便于习惯通过图形化界面操作的用户使用。

   在应用开发中您还可以通过代码或其他客户端连接AnalyticDB for MySQL，详情请参见连接分析型数据库MySQL版。

5. **将数据导入**AnalyticDB for MySQL

   AnalyticDB for MySQL支持多种数据入库方式，您可以通过阿里云数据传输DTS（Data Transmission Service）将MySQL或PolarDB-X（原DRDS）中的数据导入AnalyticDB for MySQL，其中MySQL可以是RDS MySQL、其他云厂商或线上IDC的自建MySQL以及ECS自建MySQL。详情请参见使用DTS同步MySQL数据。

   更多数据导入方法，请参见同步数据。

6. **在**AnalyticDB for MySQL**中新增数据、删除数据、更新数据**

   - **新增数据**：实时表插入（INSERT）数据后一般需要5~10秒后才能查询结果。如果需要插入数据实时可见，请联系技术支持。
   - **删除数据**：AnalyticDB for MySQL支持通过删除（DELETE）普通表中的数据。

- **更新数据**：AnalyticDB for MySQL不支持UPDATE操作，支持通过主键覆盖INSERT实现数据更新。

关闭AnalyticDB MySQL版2.0服务

根据业务需求，您可以随时关闭AnalyticDB for MySQL服务，详情请参见关闭分析型数据库MySQL版服务。

2.2. 连接2.0集群

在应用开发中，您可以通过以下方式连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。

- 通过代码连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。
- 通过客户端连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。

应用开发中通过代码连接2.0集群

在应用开发中可以通过Java、PHP、Python、C#或者Druid连接2.0集群。

通过客户端连接2.0集群

支持使用客户端MySQLWorkbench、DBVisualizer或者Navicat for MySQL连接2.0集群。

连接说明

连接2.0集群时，需要获取以下连接信息。

- **连接地址和端口**

  登录云原生数据仓库AnalyticDB MySQL版管理控制台，**在连接信息处查看连接地址和端口。通过外网连接2.0集群时，使用经典网络地址连接串。通过内网连接2.0集群时，使用VPC地址连接串。**

- **用户名和密码**

  用户名和密码是指用户阿里云账号（或者RAM子账号）的AccessKey ID和Access Key Secret。获取AccessKey，请参见如何获取AccessKeyId和AccessKeySecret。

2.3. 创建表组

表组是一系列可发生关联的数据表的集合，分析型数据库MySQL版（AnalyticDB for MySQL）为管理相关联的数据表，引入表组的概念。

说明

创建AnalyticDB for MySQL数据库时，系统自动创建维度表组。

注意事项

- 表组名以字母开头，字母或数字结尾（不能以下划线结尾）；可包含字母、数字或下划线（_），长度不超过64个字符。
- 表组名中不能包含双下划线（__）。
- 同一个数据库中，表组名唯一。

通过DMS页面创建普通表组

1. 登录云原生数据仓库控制台，请参见AnalyticDB 控制台。
2. 在控制台左上方选择地域。
3. 单击目标数据库右侧的**登录数据库**，进入**DMS for AnalyticDB**页面，可以看到系统创建的维度表组，如下图所示。

4. 在顶部菜单栏单击**创建>表组**。
5. 在**新建表组**窗口，输入**表组名**。
6. 单击**提交**，即可创建一个普通表组。

通过DDL语句创建普通表组

CREATE TABLEGROUP（2.0版）

2.4. 创建表

AnalyticDB的表分为维度表和普通表，详细介绍请参见名词解释。您可以通过DMS页面或者DDL语句创建表。

注意事项

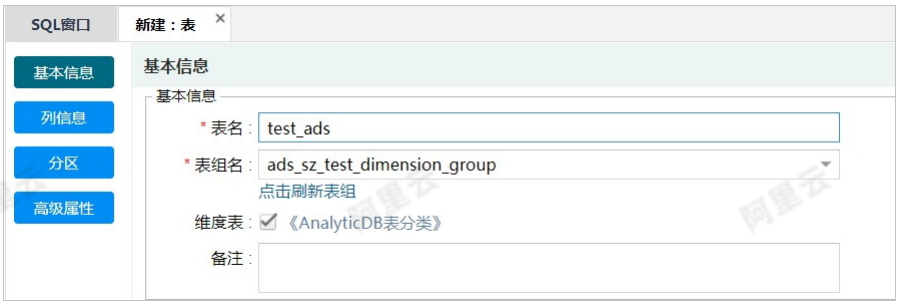
- 表名以字母开头，字母或数字结尾（不能以下划线结尾）；可包含字母、数字或下划线（_），长度不超过64个字符。
- 表名中不能包含双下划线（__）。

通过DMS页面创建维度表

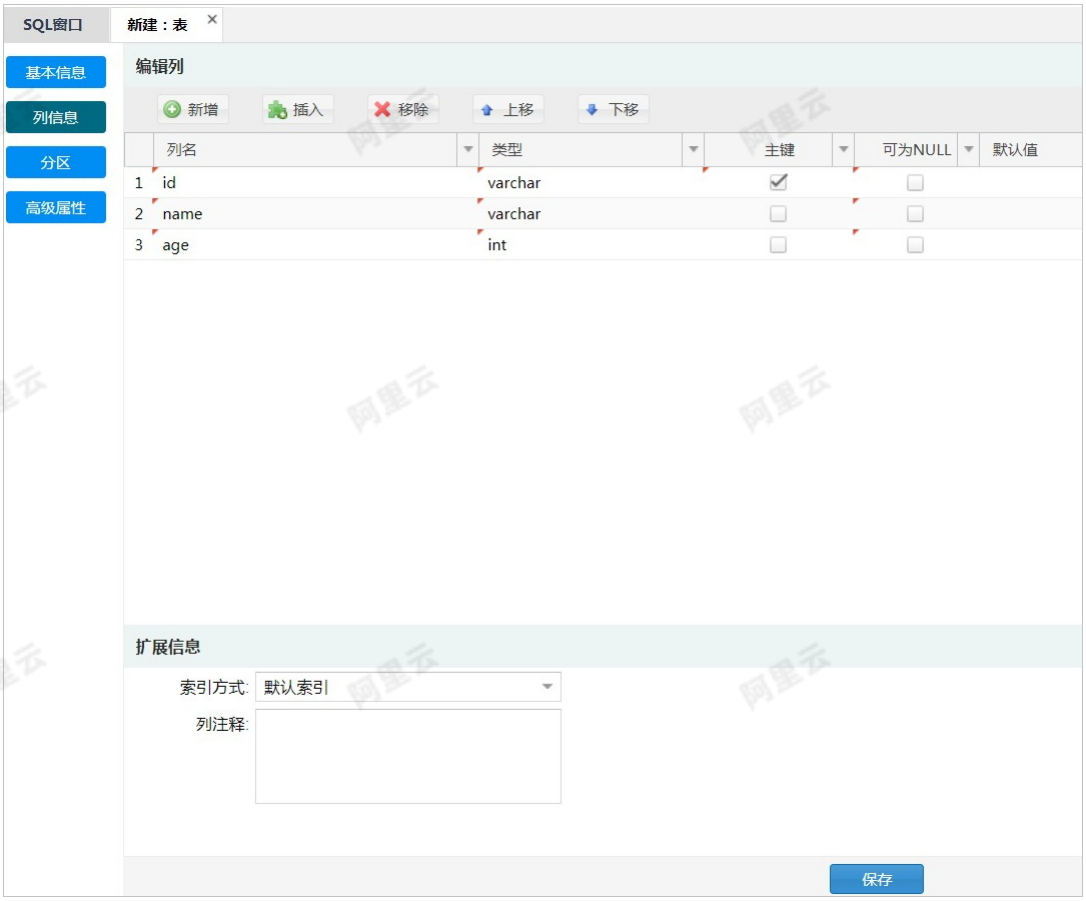
1. 登录分析型数据管理控制台。
2. 在控制台左上方选择数据库所属地域。
3. 单击目标数据库右侧的**登录数据库**，在**DMS for AnalyticDB**页面的顶部菜单栏单击**创建>表**。
4. 在**新建：表**页面，按照提示进行参数配置。

   在**新建：表**页面，按照提示进行参数配置。

   基本信息配置：



列配置：



分区配置：

高级属性：

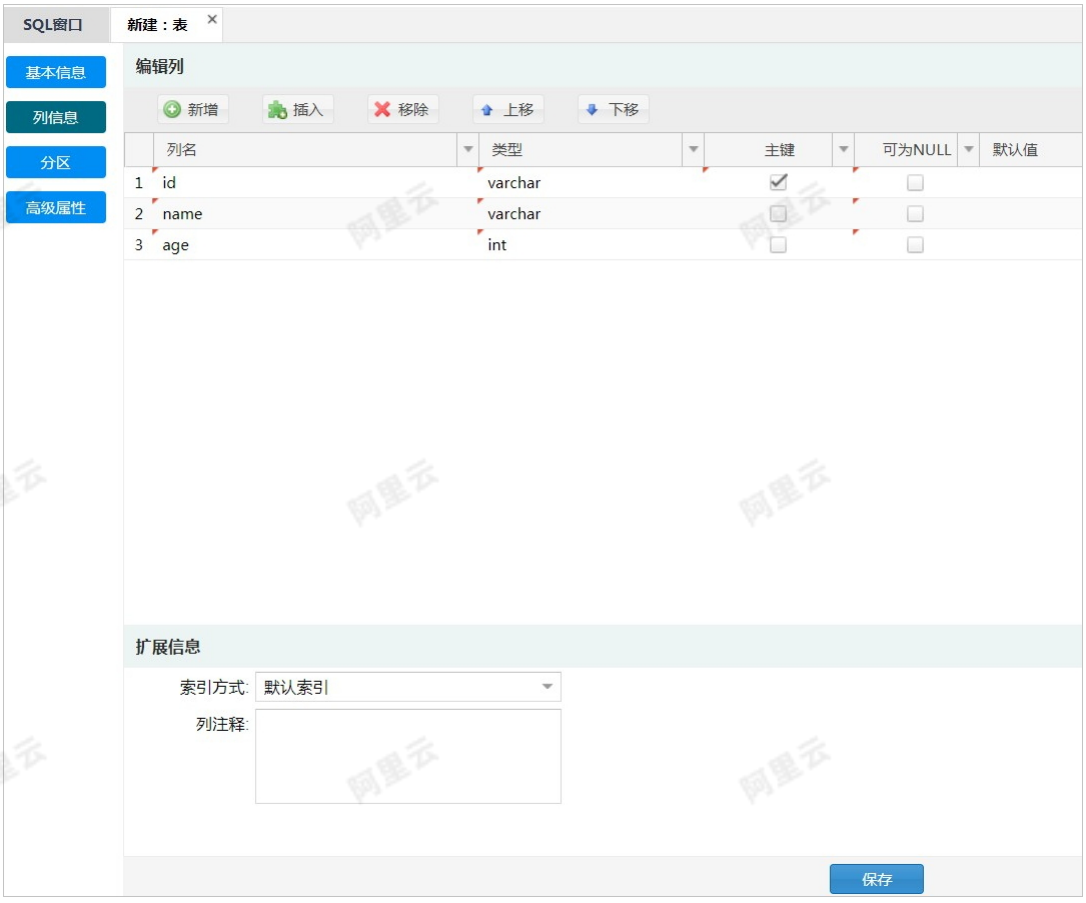
5. 配置好上述参数后，单击**保存**。

通过DMS页面创建普通表

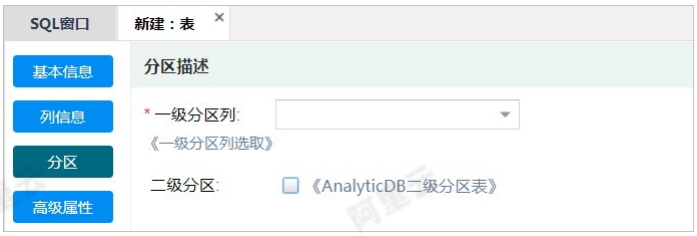
1. 登录分析型数据管理控制台。
2. 在控制台左上方选择数据库所属地域。
3. 单击目标数据库右侧的**登录数据库**，在**DMS for AnalyticDB**页面的顶部菜单栏单击**创建>表**。
4. 在**新建：表**页面，按照页面提示进行参数配置。

   基本信息配置：

列配置：



分区配置：



二级分区为非必选项，为高级功能，详细介绍参见二级分区表。

高级属性：

5. 配置好上述参数后，单击保存。

通过DDL语句创建表

详情，请参见CREATE TABLE

相关文档

- AnalyticDB系统保留字
- 列的最佳实践
- AnalyticDB与MySQL数据类型对比

2.5. 同步数据

本文介绍了多种同步数据的方式。

分析型数据库MySQL版支持多种数据加载方式，包括但不限于：

- 通过阿里云数据传输服务DTS将MySQL/DRDS中的数据导入分析型数据库MySQL版，其中MySQL可为RDS for MySQL、其他云厂商或线上IDC的自建MySQL以及ECS自建MySQL。详细操作步骤请参见使用DTS同步RDS for MySQL数据。
- 通过阿里云数据集成将ODPS/OSS/MySQL/Oracle/SqlServer中的数据导入分析型数据库MySQL版中。详细操作步骤请参见使用数据集成迁移数据至分析型数据库MySQL版。
- 通过Kettle将关系型数据库、Hbase等NoSQL数据源，以及Excel、Access中的数据导入分析型数据库MySQL版中，详细操作步骤请参见使用kettle将本地数据导入分析型数据库MySQL版。
- 通过Logstash插件将采集到的日志等数据等实时写入分析型数据库MySQL版。
- 通过分析型数据库MySQL版 client高效率的写入数据到数据库中。
- 通过ADB uploader将本地数据导入到数据库中。
- 通过DML（INSERT和DELETE）命令加载数据到分析型数据库MySQL版中。
- 如果数据在数据库其它表中已经存在，可以使用INSERT SELECT FROM加载数据。

2.6. 导出数据

分析型数据库MySQL版对海量数据分析计算后支持输出（DUMP）数据结果。

导出方式

目前支持两种DUMP方式：

- 导出数据到OSS
- 导出数据到MaxCompute

3.2.0版用户指南

3.1. 2.0版计费管理

3.1.1. 手动续费（2.0版）

包月套餐数据库有到期时间，如果到期未续费，将导致业务中断甚至数据丢失，详情请参见欠费或到期对数据库的影响。

在包月套餐数据库未到期或者到期后7天内，您可以手动给数据库续费，延长数据库的使用时间。

> *按量付费数据库没有到期时间，不涉及续费操作。*

AnalyticDB支持自动续费，开通自动续费可以免去定期手动续费的烦恼，且不会因忘记续费而导致业务中断。

1. 登录AnalyticDB 控制台。
2. 在控制台左上方选择AnalyticDB所属地域。
3. 在数据库列表中找到目标数据库，单击右侧的**续费**。
4. 在**续费**页面，设置**续费时长**。时间越长，折扣越多。

5. 勾选**《分析型数据库（包年包月）服务协议》**，单击**去支付**，完成支付即可。

批量续费多个数据库

1. 登录AnalyticDB 控制台。
2. 在控制台右上方，选择**费用>续费管理**。

3. 在**续费管理**页面左侧，选择**分析型数据库**。
4. 在**批量续费**页面，选择**按年**或者**按月**续费数据库。

5. 单击**去支付**，根据提示完成支付。

3.1.2. 自动续费（2.0版）

续费是指延长包年包月集群的使用时间，续费的费用和新购集群费用相同。按量付费集群没有到期时间，无需续费，您只需保证阿里云账号的余额充足即可。云原生数据仓库AnalyticDB MySQL 版支持自动续费和手动续费。本文介绍云原生数据仓库 AnalyticDB MySQL 版包年包月集群的续费操作。

注意事项

- 建议开通自动续费，以免忘记续费而导致业务中断。
- 如果没有开通自动续费，请在集群到期前，手动续费。到期未续费，将导致业务中断甚至数据丢失。

创建集群时开通自动续费

您可在创建包年包月集群时，选中**到期自动续费**。

续费控制台开通自动续费

如果您没有在创建集群时开通自动续费，可以通过续费控制台为已创建的包年包月集群开通自动续费。

> **重要**
> 已过期的包年包月集群不能在续费控制台开通自动续费。

1. 登录AnalyticDB MySQL控制台。
2. 在控制台上方，选择**费用>续费管理**。
3. 通过**到期时间**、**产品**、**地域**维度筛选出所需实例。
4. 选中所需的实例，在**操作**列单击**开通自动续费**。
5. 在**开通自动续费**弹窗中，选择**自动续费周期**，单击**开通自动续费**。

修改自动续费

1. 登录云原生数据仓库AnalyticDB MySQL控制台。
2. 在控制台右上方，选择**费用 > 续费管理**。
3. 在**续费管理**页面的左侧，选择**分析型数据库**。
4. 在**续费管理**页面，单击**自动续费**页签，找到目标数据库，单击右侧的**修改自动续费**。
5. 在**修改自动续费**页面，设置自动续费时长，然后单击**确定**。

3.1.3. 按量付费转包年包月（2.0版）

用户可以根据需求将按量付费类型的数据库转变为包年包月的计费方式。

1. 登录分析型数据库MySQL版控制台。
2. 在控制台左上方选择分析型数据库MySQL版所属地域。
3. 单击目标数据库右侧的**按量转包月**。

4. 在**确认订单**页面，设置**购买时长**，勾选**《分析型数据库MySQL版（包年包月）服务协议》**，单击**去开通**根据提示完成支付。

3.2. 2.0版数据库管理

3.2.1. ECU详解

弹性计算单元（Elastic compute units，简称ECU）是分析型数据库MySQL版中衡量实例计算能力的元单位。ECU由内存容量和磁盘容量组成。

分析型数据库MySQL版中有**高性能**和**大存储**两种类型的ECU。

- **高性能**：以字母C或者H开头的ECU为高性能实例，数据全部存储在SSD磁盘中。

适用于对性能要求高、查询并发高的业务场景 。

- **大存储**：以字母S开头的ECU为大存储实例，采用SSD/HDD分层存储架构，热点数据存储在SSD磁盘中，冷数据存储在HDD磁盘中。
  适用于并发稍低、性能要求不高（可接受数据查询响应时间受超过10秒以上）的业务场景 。

ECU规格

高性能	C4	30GB	SSD	180GB	-
	C8	45GB	SSD	480GB	-
	H8	160GB	SSD	2TB	-
大存储	S2N（S2）	45GB	SSD+HDD	480GB	2TB
	S8N	120GB	SSD+HDD	1TB	8TB

注意事项

- 分析型数据库MySQL版默认采用双副本存储形式，每个数据库实际可存储的数据量为：**ECU个数*ECU磁盘容量/2**。
- ECU购买个数限制：每种类型ECU的单次和总共购买次数的限制不同。
- 资源操作次数限制：每个数据库每天允许扩容、缩容、升配和降配总次数不能超过12次。

3.2.2. 变更ECU配置

如果当前ECU配置无法满足需求，您可以变更ECU配置。本章节为您介绍如何通过分析型数据库MySQL版控制台，变更ECU配置。

您的阿里云账号中没有未支付的续费订单。

计费规则

变更ECU配置时将引起费用变化，详情请参见变更ECU配置计费规则。

操作步骤

1. 登录分析型数据库MySQL版控制台。
2. 在控制台左上方选择分析型数据库MySQL版所属地域。
3. 单击目标数据库右侧的**更多>升配扩容**或者**降配缩容**。

4. 在**变配**页面，选择**ECU类型**和**ECU数量**，勾选**《分析型数据库MySQL版（按量付费）服务协议》**。
5. 单击**去开通**，根据提示完成支付。

> *注意：变配操作涉及到底层数据迁移，数据量不同变配时间不同，需要耐心等待。*

示例

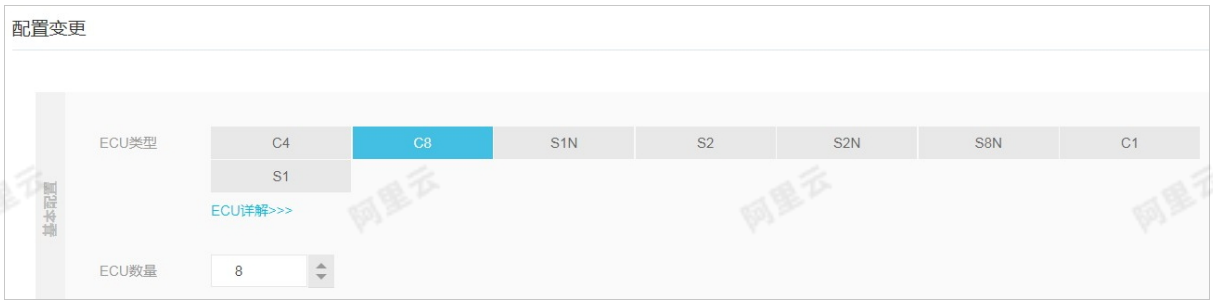
以下示例将分析型数据库MySQL版ECU配置从8个C4变配为4个C8。分为两个过程：

1. 8个C4升配为8个C8
2. 8个C8缩容为4个C8

详细操作步骤如下所示。

1. 登录分析型数据库MySQL版控制台。
2. 在控制台左上方选择分析型数据库MySQL版所属地域。
3. 单击目标数据库右侧的**更多>升配扩容**。

4. 在**变配**页面，**ECU类型**选择C8，**ECU数量**选择8，然后勾选**《分析型数据库MySQL版（按量付费）服务协议》**。



5. 单击**去开通**，根据提示完成支付。升配成功后ECU为8个C8。
6. 待目标分析型数据库MySQL版状态为**运行中**时，单击目标数据库右侧的**更多>降配缩容**。
7. 在**变配**页面，将**ECU数量**缩容为4，然后勾选**《分析型数据库MySQL版（按量付费）服务协议》**。

8. 单击**去开通**，阿里云将把缩容的费用退还到您的账户中。
   至此，完成了由8个C4变配为4个C8的操作。

3.2.3. 关闭分析型数据库MySQL版服务

根据业务需求，用户可以主动关闭分析型数据库MySQL版服务。

按量付费

1. 登录分析型数据库MySQL版控制台。
2. 在控制台左上方选择分析型数据库MySQL版所属地域。
3. 单击目标数据库右侧的**更多>删除**。

4. 在**删除数据库**提示框中，单击**确定**，大约5分钟后删除操作生效。

包月套餐

包月套餐数据库到期后会被自动释放，分析型数据库MySQL版支持提前退订。提前退订规则，请参见退订规则说明。

3.3. 网络

3.3.1. 网络类型

本文介绍了如何新建专有网络。

网络类型和访问方式

分析型数据库MySQL版支持的网络类型有两种：经典网络和专有网络（Virtual Private Cloud 以下简称VPC）。

- 经典网络：IP地址由阿里云统一分配，配置简便，使用方便，适用于对操作易用性要求比较高的场景。
  所有经典网络类型的数据库都建立在一个共用的基础网络上。数据库之间不通过网络进行隔离，只能依靠数据库自身的安全策略来阻挡非法访问。
- VPC：是逻辑隔离的私有网络，用户可以自定义网络拓扑和IP地址，支持通过专线连接。相对经典网络而言，VPC具有更高的安全性和灵活性。

新建专有网络

1. 登录分析型数据库MySQL版控制台。
2. 在控制台左上方选择分析型数据库MySQL版所属地域。
3. 单击**集群列表**，单击**2.0集群列表**。
4. 单击目标数据库右侧的**更多>创建专有网络**。

5. 在**创建专有网络**窗口选择**专有网络**和**专有网络交换机**，然后单击**确定**，需等待约几分钟专有网络即可生效。

注意事项

- 若当前数据库所属地域没有创建专有网络和交换机，请前往专有网络管理控制台创建专有网络和交换机。详情，请参见搭建IPv4专有网络中的步骤一创建专有网络和交换机。
- 新建VPC后，数据库访问方式属于混访模式，即经典网络和VPC同时存在。

3.4. 2.0版监控与报警

3.4.1. 设置报警规则（2.0版）

本文介绍如何通过云监控控制台为AnalyticDB for MySQL设置报警规则。

背景信息

监控报警是通过阿里云监控产品实现的。通过阿里云监控产品，您可以设置监控项，并在触发监控项的报警规则时，通知报警联系组中的所有联系人。您可以维护报警监控项对应的报警联系组，以便发生报警时，能及时通知到相关联系人。

操作步骤

1. 登录云监控控制台。
2. 单击左侧导航栏中的**事件监控**，进入**事件监控**页面。
3. 在**报警规则**页签下，单击右上角的**创建事件报警**，弹出**创建/修改事件报警**对话框，按照页面提示进行参数配置。

参数	说明
报警规则名称	设置报警规则的名称。 报警规则的名称由字母、数字、下划线（_）组成，长度不超过30个字符。
事件类型	选择**系统事件**。
产品类型	选择**分析型数据库**。
事件类型	选择**全部类型**。
事件等级	选择**全部等级**。
事件名称	选择**全部事件**。
资源范围	选择**全部资源**。 任何资源发生相关事件，均会按照配置发送通知给报警联系人。
-	勾选**报警通知**
联系人组	设置报警联系人组。 关于报警联系人/报警联系组，请参见创建报警联系人/报警联系组和报警联系人/报警联系组管理。
通知方式	设置报警的通知方式。

4. 完成上述参数配置后，单击**确定**创建报警规则。

3.4.2. 查看事件监控（2.0版）

本文介绍如何通过云监控控制台查看AnalyticDB for MySQL事件监控。通过事件监控，您可以明确知晓AnalyticDB for MySQL的使用状态。当您的业务出现问题时，也可以通过事件监控快速分析、定位问题。

系统事件

AnalyticDB for MySQL告警目前支持监控以下事件，监控事件的触发告警阈值由云监控后台统一设置，暂不支持用户自定义设置。

事件名称	事件含义	事件等级	触发条件
StorageUsage	磁盘使用率超过80%	CRITICAL	`磁盘使用率>80%` 时将触发报警。
InsertFailureRate	插入失败率10%	CRITICAL	5分钟内，AnalyticDB for MySQL的 `写入失败率>10%` ，且失败次数超过5次将触发报警。
SelectFailureRate	查询失败率10%	CRITICAL	5分钟之内，AnalyticDB for MySQL的 `查询失败率>10%` ，且失败次数超过5次将触发报警。

查询事件监控

1. 登录云监控控制台。
2. 单击左侧导航栏中的**事件监控**，进入**事件监控**页面。

3. 选择**系统事件**，在产品下拉框中选择**分析型数据库**，在事件下拉框中选择事件，然后选择时间，单击**搜索**即可查看指定时间内发生的事件。
4. 单击操作栏中的**查看详情**，查看相关事件的详细信息。

3.5. 2.0版SQL开发规范

云原生数据仓库 AnalyticDB MySQL 版2.0集群是一个分布式、列存数据库，在编写和优化SQL时，需要充分考虑其分布式特性。

在2.0集群中，编写和优化SQL的要求和经验总结如下：

- **SQL编写原则为追求简单**

  一般情况下，数据库性能会随SQL复杂度而下降。例如，单表查询（冗余设计）优于表关联查询。
- **SQL优化核心方法是减少I/O**

  尽可能少的进行列扫描，返回最小数据量，减少I/O同时也减少内存开销。
- **分布式计算，本地计算&并行计算**

  大数据计算情况下，本地计算时充分利用分布式多计算资源的能力，避免数据跨节点。
- **高QPS，分区裁剪**

  业务系统要求高QPS、毫秒级RT时，表和SQL必须设计为分区裁剪模式。

常见SQL优化细节

去掉不必要的列

云原生数据仓库 AnalyticDB MySQL 版2.0集群是列存数据库，返回的列的数量直接影响性能，在编写SQL时一定要确认业务需要返回的列，不要直接使用*。

典型错误SQL写法

```
select * from tab1 where c1>100 and c1<1000;
```

正确SQL写法

```
select col1, col2 from table_name where c1>100 and c1<1000;
```

索引&扫描

当SQL包含多个查询条件时，优先选择高筛选条件，其他条件可以通过扫描实现。

原理

云原生数据仓库 AnalyticDB MySQL 版2.0集群内部采用列存方式，通过单列高效过滤后，可直接通过内部记录指针扫描其他列值，减少其他列的索引查询开销。

示例

- **time条件通过内部扫描**

  以下SQL通过条件 `c1=3` 可快速查询到少量记录（假设10000），单独使用 `time>'2010-01-01 00:00:00'` 时返回的记录数又非常大。

  ```
  select c1,c2 from tab1 where c1=3 and time >='2010-01-01 00:00:00';
  ```

  此时可只通过c1进行索引，time通过内部扫描方式执行，查询更快，返回更多有效记录数。SQL示例如下：

  ```
  /*+ no-index=[tab1.time] */
  select c1,c2 from tab1
  where c1=3 and time>='2010-01-01 00:00:00';
  /*+ no-index=[tab1.time] */
  ```

  Hint表示强制 `time>='2010-01-01 00:00:00'` 条件走扫描。

  在上述SQL中，计算引擎首先检索列c1的索引，得出满足条件 `c1=3` 的行集合，然后读取每行所对应的time列数据。如果满足 `time>='2010-01-01 00:00:00'` ，则将该行数据加入返回结果。
- **不等于条件通过内部扫描**

  不等于条件查询，例如： `c2<>100` ，不通过索引扫描时， `c2<>100` 无法有效过滤掉无效记录。例如：

  ```
  select c1,c2 from tab1 where c1=3 and c2<>100;
  ```

  增加 `no-index` Hint，使不等于条件通过内部扫描执行，SQL示例如下：

  ```
  /*+ no-index=[tab1.c2] */
  select c1,c2 from tab1 where c1=3 and c2<>100;
  ```
- **like条件通过内部扫描**

  中缀或后缀查询，例如： `like '%abc'` 或 `like '%abc%'` 。

  增加 `no-index` Hint，使like条件通过内部扫描执行，更加快速地查询有效记录，SQL示例如下：

  ```
  /*+ no-index=[tab1.c3] */
  select c1,c2 from tab1 where c1=3 and c3 like '%abc%';
  ```

索引失效

索引失效时，SQL语句直接以扫描的方式进行查询，如果表记录数非常大，会导致查询缓慢。

以下情形容易引起索引失效：

- 函数转换（列）；
- 类型转换；
- like条件，例如： `like '%abc%'` 。

以下SQL中的函数转换导致索引失效。time为timestamp类型，存储时间2017-12-10 10:00:23。

```
select c1,c2 from tab1 where substr(cast(time as varchar),1,10)='2017-
10-12';
```

正确SQL：

```
select c1,c2 from tab1 where time>='2017-10-12 00:00:00' and time<='2017-10-12 23:59:59';
```

is not null

去掉不必要的 `is not null` 过滤条件。

示例

```
Select c1, c2 from tab1 where c1>100 and c1<1000 and c1 is not null;
```

示例SQL中的 `and c1 is not null` 为多余条件，优化后SQL如下：

```
Select c1,c2 from tab1 where c1>100 and c1<1000;
```

多表关联

- 普通表join普通表，尽量包含分区列join条件，如果不包含则，尽量通过where条件过滤掉多余的数据。
- 维度表join普通表，没有限制。

多表关联查询where条件中，需要明确写明每一个表的过滤条件。通常我们在传统数据库中，都是通过索引字段关联来快速检索数据。如下SQL：

```
Select count(*)
from customer_table C join
order_table O on C.customer_id= O.customer_id
where O.order_time between'2018-07-20 10:00:11'
and '2018-09-30 10:00:11'
and O.order_amount=100;
```

在明确t2与t1表都有同样的time和type过滤条件情况下，建议修改为如下SQL：

```
Select count(*)
from t1 join t2 on t1.id=t2.id
where t1.time between '2017-12-10 00:00:00' and '2017-12-10 23:59:59'
and t1.type= 100
and t2.time between '2017-12-10 00:00:00' and '2017-12-10 23:59:59'
and t2.type=100
```

3.6. 2.0版高级功能

3.6.1. 二级分区表（2.0版）

二级分区表是分析型数据库MySQL版向用户提供的高级功能，用于实现数据的增量同步。

语法

```
CREATE TABLE table_name (
column_name data_type [NOT NULL][DEFAULT 'default'][COMMENT 'comment'][, …],
    primary key (column_name[, … ])
)
PARTITION BY HASH KEY(column_name) [PARTITION NUM N]
SUBPARTITION BY LIST (subpart_col long)
SUBPARTITION OPTIONS (available_partition_num = NUM)
TABLEGROUP tablegroup_name
options (updateType='realtime');
```

参数

绝大多数参数和创建普通表语法中介绍一样，详情请参见CREATE TABLE，二级分区表多了两个参数：

- subpart_col：二级分区列，该列不在定义的列中需要重新定义，类型必须为long。
- available_partition_num：二级分区数，即为最大保留的二级分区数，当新的数据装载进来后，若线上存在的二级分区数大于这个值，分析型数据库MySQL版会根据二级分区的值进行排序，下线最小的若干分区的数据。

另外，primary key的定义也是要强调下，primary key中必须包含二级分区列。

示例

新建 `customer` 表，以 `search_time` 为二级分区列，二级分区保留个数为30。

```
CREATE TABLE customer (
  customer_id bigint NOT NULL COMMENT '顾客ID',
  customer_name varchar NOT NULL COMMENT '顾客姓名',
  phone_num bigint NOT NULL COMMENT '电话',
  city_name varchar NOT NULL COMMENT '所属城市',
  sex int NOT NULL COMMENT '性别',
  id_number varchar NOT NULL COMMENT '身份证号码',
  home_address varchar NOT NULL COMMENT '家庭住址',
  office_address varchar NOT NULL COMMENT '办公地址',
  age int NOT NULL COMMENT '年龄',
  login_time timestamp NOT NULL COMMENT '登录时间',
  search_time timestamp NOT NULL COMMENT '搜索时间',
  PRIMARY KEY (customer_id,phone_num,search_time)
)
PARTITION BY HASH KEY (customer_id)
SUBPARTITION BY LIST (search_time long)
SUBPARTITION OPTIONS (available_partition_num = 30)
TABLEGROUP tablegroup_name
OPTIONS (UPDATETYPE='realtime')
COMMENT '客户信息表';
```

二级分区使用场景

一般情况下，当一级分区数据量随时间增大到超过单个一级分区记录数最佳值（2000万~3000万）时，可以考虑设计二级分区。二级分区可以理解为按队列方式管理分区个数，当超过最大定义数，最小值分区自动删除，循环使用空间，所以二级分区是自动清除历史数据。

具体使用例子，请参见电子商务行业的实践。

最佳实践

如果二级分区过多，则会导致多次索引查询、性能下降。如果二级分区过少，会降低用户导入数据的频率，从而影响数据的实时性，所以需要合理选择二级分区数：

- 单表二级分区数小于等于90，每个一级分区下的二级分区包含的数据条数在300万到2000万之间。
- 如果单个分区每日增量数据超过300万，则推荐按天进行二级分区；如需要存储的时间范围更长，则可按周、月进行规划。

动态调整二级分区个数

分析型数据库MySQL版支持动态调整二级分区个数。当数据存储时间周期发生变化时，需要动态调整二级分区个数。具体语法参见修改二级分区数。

3.6.2. 聚集列（2.0版）

在分析型数据库MySQL版中，数据存储支持按一列或多列进行排序（先按第一列排序，第一列相同情况下使用第二列排序），以保证该列中值相同或相近的数据保存在磁盘同一位置，这样的列我们称之为聚集列。

当以聚集列为查询条件时，由于查询结果保存在磁盘同一位置，可以减少I/O（Input/Output）次数。分析型数据库MySQL版中主聚集列只有一列，因此需要选择最合适的列作为主聚集列。

语法

```
CREATE TABLE table_name (
    column_name data_type [NOT NULL][DEFAULT 'default'][COMMENT 'comment'][, …],
    primary key (column_name[, … ])
)
PARTITION BY HASH KEY(column_name) PARTITION NUM 128
TABLEGROUP table_group_name
[CLUSTERED BY (column_name1,column_name2)]
options (updateType='realtime')
```

参数

绝大多数参数和普通表中介绍一样，CLUSTERED BY用于指定聚集列，您可以把一列或者多列指定为聚集列。

聚集列选择依据

- 主要或大多数的查询条件中均包括某一列，且该查询条件具有较高的筛选率，则选择该列作为聚集列。
- Join子句中的等值条件列（通常是一级分区列）作为聚集列。

示例

现有一张数据表，其一级分区键和聚集列均为 `org_code` ，示例如下。

```
CREATE TABLE t_fact_mail_status (
mail_id varchar COMMENT '',
scan_timestamp timestamp COMMENT '',
biz_date bigint COMMENT '',
org_code varchar COMMENT '',
org_name varchar COMMENT '',
dlv_person_name varchar COMMENT '',
receiver_name varchar COMMENT '',
receiver_phone varchar COMMENT '',
receiver_addr varchar COMMENT '',
product_no varchar COMMENT '',
mag_no varchar COMMENT '',
PRIMARY KEY (mail_id,org_code,biz_date)
)
PARTITION BY HASH KEY (org_code) PARTITION NUM 128
SUBPARTITION BY LIST KEY (biz_date)
SUBPARTITION OPTIONS (available_partition_num = 30)
CLUSTERED BY (org_code)
TABLEGROUP ads
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

设置 `CLUSTERED BY(org_code)` 后，`org_code` 记录会按一级分区规则分布在某一个计算节点CN上，并尽可能的存储在同一个数据块上。当以聚集列为查询条件时，相比未设置聚集列的查询，如下SQL语句的访问I/O将减少数百倍。

```
select mail_id,biz_date,org_code,org_name
from t_fact_mail_status
where org_code='203202'and biz_date=20171221;
```

假设共有10万个不同的 `org_code` ，每个 `org_code` 每天大约有100条~5000条记录。如果 `org_code=203202` 的记录为1000条，则按以上方案设置一级分区和聚集列后，这1000条记录会存储在连续的几个数据块上。当您通过以上SQL语句查询数据时，SQL语句只需要扫描这几个数据块即可。

3.6.3. 分页（2.0版）

本文介绍如何实现查询分页。

语法

```
select ....
order by ....
limit m,n
```

注意事项

- 查询语句中必须带有ORDER BY子句。
- AnalyticDB for MySQL 2.0的分页兼容MySQL，即 `limit m,n m` 从0开始。

3.6.4. JSON索引（2.0版）

背景信息

大数据时代结构化数据检索已经逐渐有了多元化的、丰富的解决方案。但是，事实上大多数的大数据都是半结构化的，并且半结构化数据的数据量仍旧急剧增长。理解和分析半结构化数据的难度比结构化数据大很多，急需成熟的解决方案来处理半结构化数据。为了赋能用户、降低用户处理半结构化数据的难度，分析型数据库MySQL版提供了半结构化数据检索功能即JSON检索。

注意事项

分析型数据库MySQL版JSON索引有以下功能限制需要您注意。

- 不支持更改索引。表创建成功后不支持通过 `ALTER TABLE ADD` 增加索引，也不支持通过 `DROP JSON INDEX <idx_name> (col_name)` 删除索引列。
- 创建表时指定某一列类型为JSON之后，分析型数据库MySQL版自动构建JSON INDEX，系统不再支持普通的倒排索引操作。例如，`where json_column = '{"id":123}'` 类似的字符串等值、不等值、范围过滤以及 `LIKE` 操作等。
- JSON ARRAY查询对标于Elasticsearch的Object类型，而不是nested类型。

  例如，JSON ARRAY数据为 `{"addr":[{"city":"beijing", "no":11}, {"city":"shenzhen", "no":0}]}` ，如果检索条件是 `addr.city=beijing AND addr.no=0` ，虽然上述数组中没有元素为 `{"city":"beijing", "no":0}` ，但仍然可以成功执行SQL `select id, json_test from json_tbl where json_extract(json_test, '$.addr.city') = 'beijing' and json_extract(json_test, '$.addr.no') = 0;` 。

建表

在建表时，将某一列指定为JSON类型即可为该列自动构建JSON索引。例如，以下示例中的 `json_test` 字段类型为JSON类型，建表成功后分析型数据库MySQL版自动为 `json_test` 列构建JSON索引。

```
create tablegroup par2_group;
CREATE TABLE json_tbl (
  id bigint COMMENT '',
  sid bigint COMMENT '',
  json_test json COMMENT '类型为json，自动构建json index',
  PRIMARY KEY (id,sid)
)
PARTITION BY HASH KEY (sid) PARTITION NUM 8
CLUSTERED BY (sid)
TABLEGROUP par2_group
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

JSON格式要求

在写入数据时，分析型数据库MySQL版对JSON类型数据中的属性键 `key` 和属性值 `value` 有以下要求。

- **属性键** `key` ：
  - 必须使用双引号（ `""` ）将 `key` 引起来。
  - `key` 中不能包含点号（ `.` ），例如 `{"a.b":"value"}` 是不合法的JSON格式。
- **属性值** `value` ：
  - 如果 `value` 是字符串类型，必须使用双引号将 `value` 引起来。
  - 如果 `value` 是字符串类型，且 `value` 中包含双引号，需要做转义处理。

    例如， `value` 为 `{"addr":"xyz"ab"c"}` 时，如果没有对 `xyz"ab"c` 中的双引号做转义处理，则不符合JSON格式规范。正确的写法为 `{"addr":"xyz\"ab\"c"}` 。
  - 如果 `value` 是数值类型，直接写数据，无需使用双引号将 `value` 引起来。
  - 如果 `value` 是 `BOOLEAN` 类型，直接写 `TRUE` 或者 `FALSE` ，不能写成 `1` 或者 `0` 。
  - 如果 `value` 是 `NULL` ，直接写 `NULL` 。
  - 分析型数据库MySQL版采用隐式类型推断来判断各个 `value` 的类型。同一个属性键 `key` 对应的 `value` ，前后必须为同一种类型。

    例如，同一个JSON类型的字段先写入数据 `{"id":0}` ，分析型数据库MySQL版推断 `id` 为数值类型；随后又写入数据 `{"id":"1"}` ，分析型数据库MySQL版推断 `id` 为字符串类型，此时前后类型不一致，系统会提示类型不匹配错误。
- 分析型数据库MySQL版支持JSON数组写入，包括PLAIN ARRAY及嵌套ARRAY。

  例如， `{"hobby":["basketball", "football"]}` ， `{"addr":[{"city":"beijing", "no":0}, {"city":"shenzhen", "no":0}]}` 。
- JSON类型字段的长度限制与 `VARCHAR` 相同。

写入数据

向表中写入数据时，JSON类型字段的写入方式与VARCHAR类型字段的写入方式相同，在JSON串两端使用单引号引起来即可。

注意

分析型数据库MySQL版只支持标准JSON格式，写入的JSON串必须严格符合标准JSON格式规范。在JSON串 `{key:value, key:value}` 中，每个 `key:value` 为一个属性对，其中 `key` 为属性键， `value` 为属性值。

示例

以下SQL示例包含多种JSON数据格式，供您参考使用。

```
insert into json_tbl (id, sid, json_test) values(0, 0, '{"id":0, "name":"abc", "age":0}');
insert into json_tbl (id, sid, json_test) values(1, 1, '{"id":1, "name":"abc", "age":10, "gender":"femal"}');
insert into json_tbl (id, sid, json_test) values(2, 2, '{}');
insert into json_tbl (id, sid, json_test) values(3, 3, '{"id":3, "name":"xyz", "age":30, "company":{"name":"alibaba", "place":"hangzhou"}}');
insert into json_tbl (id, sid, json_test) values(4, 4, null);
insert into json_tbl (id, sid, json_test) values(5, 5, '{"id":5, "name":"abc", "age":50, "company":{"name":"alibaba", "place":"america"}}');
insert into json_tbl(id, sid, json_test) values (6, 6, '{"a":1, "b":"abc-char", "c":true, "d":null}');
insert into json_tbl(id, sid, json_test) values (7, 7, '{"uname":{"first":"lily", "last":"chen"}, "addr":[{"city":"beijing", "no":1},
{"city":"shenzhen", "no":0}], "age":10, "male":true, "like":"fish", "remark":null, "hobby":["basketball", "football"]}');
```

查询数据

从表中查询数据时，分析型数据库MySQL版支持使用函数 `json_extract` 进行查询。

- **语法：** `json_extract(json_col, 'json-path')`
- **参数：**
  - `json_col` ，JSON列的列名。
  - `json-path` ，通过点号（ `.` ）进行分割的JSON属性键 `key` 的路径，其中 `$` 表示最外层的路径。

除了基本查询以外，分析型数据库MySQL版支持对 `key` 按条件进行查询。

- **基本查询**

```
select * from json_tbl order by id limit 10;
```

- **等值查询**

```
select id, json_test from json_tbl where json_extract(json_test, '$.uname.first') = 'lily';
select id, json_test from json_tbl where json_extract(json_test, '$.age') = 10;
select id, json_test from json_tbl where json_extract(json_test, '$.addr.city') = 'shenzhen';
select id, json_test from json_tbl where json_extract(json_test, '$.age') != 10;
```

- **范围查询**

```
select id, json_test from json_tbl where json_extract(json_test, '$.age') > 0;
select id, json_test from json_tbl where json_extract(json_test, '$.age') < 100;
select id, json_test from json_tbl where json_extract(json_test, '$.name') > 'a' and json_extract(json_test, '$.name') < 'z';
```

- **BETWEEN AND查询**

```
select id, json_test from json_tbl where json_extract(json_test, '$.age') between 0 and 100;
```

- **IS NULL/IS NOT NULL查询**

```
select id, json_test from json_tbl where json_extract(json_test, '$.remark') is null;
select id, json_test from json_tbl where json_extract(json_test, '$.name') is null;
select id, json_test from json_tbl where json_extract(json_test, '$.name') is not null;
```

- **IN查询**

```
select id, json_test from json_tbl where json_extract(json_test, '$.hobby') in ('football');
```

- **LIKE查询**

```
select id, json_test from json_tbl where json_extract(json_test, '$.uname.first') like 'li%';
select id, json_test from json_tbl where json_extract(json_test, '$.uname.first') like '%il%';
select id, json_test from json_tbl where json_extract(json_test, '$.uname.first') like '%ly';
```

- **ARRAY查询**

```
select id, json_test from json_tbl where json_extract(json_test, '$.addr.city') = 'shenzhen' and json_extract(json_test, '$.addr.no') = 0;
select id, json_test from json_tbl where json_extract(json_test, '$.addr.city') = 'beijing' and json_extract(json_test, '$.addr.no') = 0;
```

3.6.5. 索引（2.0版）

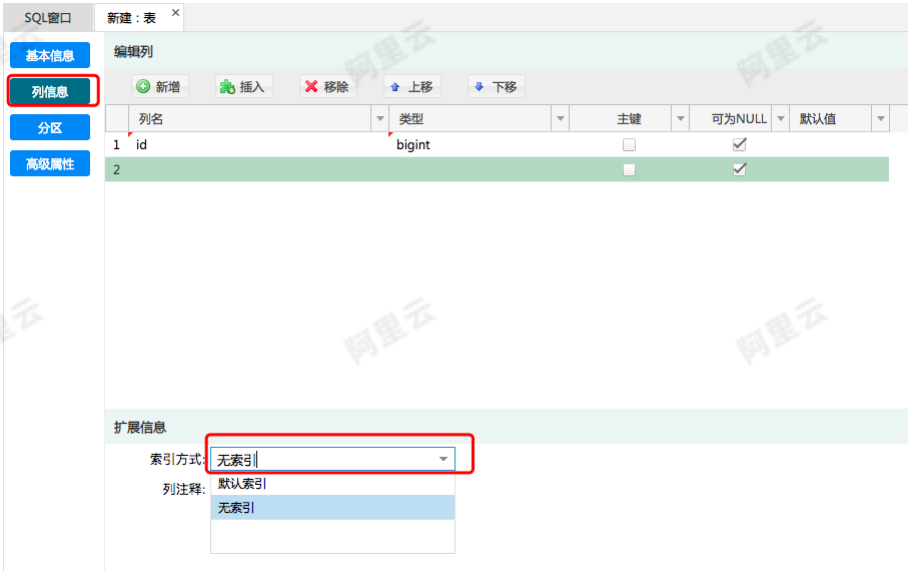
AnalyticDB for MySQL建表时默认是全索引，即为所有列创建索引。但您可以针对某一列不创建索引或者删除索引。没有创建索引的列，建议不要在查询中进行筛选和计算。

使用场景

列的类型为VARCHAR，写入长度超过16KB。

使用方法

- 通过DMS for AnalyticDB建表时，**列信息**中选择**无索引**。



- 通过CREATE TABLE（2.0版）建表时，列定义指定 `disableIndex true` ，例如以下TEST表的NAME字段定义为无索引。

```
CREATE TABLE
adb_test.test (
id bigint NOT NULL ,
name varchar  disableIndex true ,
primary key (id)
)
PARTITION BY HASH KEY(id) PARTITION NUM 128
TABLEGROUP table_group
OPTIONS(UPDATETYPE='realtime');
```

- 建表成功后，如果您需要删除某一列的索引，请联系技术支持。

4.2.0版权限与安全

4.1. 账号类型

云原生数据仓库MySQL版账号基于阿里云账号体系，阿里云账号是云原生数据仓库MySQL版资源使用的计量和计费主体。除了阿里云账号外，云原生数据仓库MySQL版同时也支持通过访问控制服务（RAM）子账号使用云原生数据仓库MySQL版。

账号格式

- 阿里云账号：`ALIYUN$account_name`（ALIYUN$为账号前缀，标识该账号类型为阿里云账号。）
- RAM子账号：`RAM$account_name:subaccount_name`（RAM$为账号前缀，标识该账号类型为RAM子账号，account_name为阿里云账号名，subaccount_name为RAM子账号名。）

用户名和密码

访问云原生数据仓库MySQL版方式有很多种：通过阿里云控制台、客户端或者JDBC。通过客户端或者JDBC访问时，需要数据库的用户名和密码，前面提到云原生数据仓库MySQL版账号是基于阿里云账号体系，所以数据库账号和密码为阿里云账号（或者RAM子账号）的访问密钥（AccessKey）。

- 数据库用户名为AccessKey ID
- 数据库密码为AccessKey Secret

用户类型

- 数据库拥有者：数据库的创建者。
- 用户：被授权的数据库用户，由数据库拥有者授权时自动添加。

4.2. 用户管理

本文介绍如何通过SQL命令查看、增加以及删除用户。

查看用户列表

注意：只有数据库创建者有权限查看用户列表。

语法

- 数据库创建者通过`LIST USERS`查看数据库中的所有用户（包含子账号）。

```
LIST USERS
```

- 数据库创建者也可以查询指定数据库中的所有用户（包括子账号）。

```
LIST USERS ON db_name.*
```

示例
数据库创建者通过`LIST USERS`查看用户列表。

```
LIST USERS
+-----------------------------+
| USERS                       |
+-----------------------------+
| ALIYUN$dt***_docs           |
| RAM$dt***_docs:adb_account  |
```

新增用户

注意：只有数据库创建者有权限新增用户。

语法

- 新增主账号

```
ADD USER 'account_name' ON db_name.*
```

- 新增子账号

```
ADD USER 'RAM$account_name:sub_account_name' ON db_name.*
```

`account_name`为阿里云账号名。

`subaccount_name`为RAM子账号名。

删除用户

注意：只有数据库创建者有权限删除用户。

语法

- 删除主账号

```
REMOVE USER 'ALIYUN$account_name' from db_name.*
```

- 删除子账号

```
REMOVE USER 'RAM$account_name:sub_account_name' from db_name.*
```

4.3. 权限管理

4.3.1. 使用阿里云访问控制进行权限管理

AnalyticDB for MySQL支持通过阿里云访问控制（RAM）创建的子账号登录数据库并管理子账号在不同条件下是否有使用数据库的权限。

子账号权限管理方式

管理方式	说明
（推荐）通过ACL对子账号进行管理	数据库创建者可以把子账号当作普通用户，通过授权ACL权限体系来访问数据库，详情请参见（推荐）通过ACL对子账号进行管理。
使用授权策略对子账号进行管理	在RAM的控制台中通过授予对应的授权策略，使子账号在一定条件下可以访问数据库，详情请参见使用授权策略对子账号进行管理。

（推荐）通过ACL对子账号进行管理

详情请参见使用DMS进行权限管理。

使用授权策略对子账号进行管理

RAM中支持的AnalyticDB for MySQL授权策略只有 `AliyunAnalyticDBFullAccess` ，当前没有集成为系统策略，需要用户添加为自定义策略。 `AliyunAnalyticDBFullAccess` 策略的定义格式如下：

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ads:*",
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

> **说明**
> - 一旦授予子账号 `AliyunAnalyticDBFullAccess` 访问策略，子账号将继承主账号在AnalyticDB for MySQL中全部权限，请慎重。
> - 暂不支持STS Token访问和角色。

4.3.2. ACL权限体系

AnalyticDB for MySQL 2.0支持基于数据库表的层级权限管理模型，提供类似MySQL的ACL授权模式。一个ACL授权由被授权的用户、授权对象和授予的对象权限组成。和MySQL不同的是，AnalyticDB for MySQL 2.0目前不支持针对用户在Host上授权。

以下列出AnalyticDB for MySQL 2.0的权限对象和各对象权限：

- Database（库），即 `db_name.*` 或 `*` （默认数据库），指定数据库或数据库上所有表/表组。
- TableGroup（表组），即 `db_name.table_group_name` 或 `table_group_name` ，特定表组。
- Table（表），即 `db_name.table_name` 或 `table_name` ，特定表。
- Column（列），语法上由 `column_list` 和Table组成，指定表的特定列。

权限模型说明

- 权限按数据库 > 表组 >表 > 列的顺序依次向下继承。

  > **重要** 如果某个账号有数据库级别的权限但回收了某个表的权限，则这个账号依然拥有该表的权限。

- 在数据库级别，某个权限实际可能包含多个权限，例如 `GRANT CREATE ON *.*` 同时包含创建数据表和创建表组的权限。

SELECT	✔	✔	✔	✔	查询数据。
LOAD DATA	✔	✔	✔	✖	导入表（分区）数据。
DUMP DATA	✔	✔	✔	✖	导出表（分区）数据。
DESCRIBE	✔	✔	✔	✖	查看数据库、表/表组信息（Global、Database）、查看表/表组信息（Table[Group]）。
SHOW	✔	✔	✔	✖	列出数据库、表/表组内部对象（Global、Database）、列出表内部对象（Table[Group]）。
ALTER	✔	✔	✔	✖	修改表/表组定义。
DROP	✔	✔	✔	✖	删除数据库、表/表组或分区（Global、Database）、删除表/表组或分区（Table[Group]）。
CREATE	✔	✔	✔	✖	创建表/表组。
INSERT	✔	✔	✔	✖	执行Insert的权限。
DELETE	✔	✔	✔	✖	执行Delete的权限。
ALL [PRIVILEGES]	✔	✔	✔	✔	以上所有权限。

例如，执行 `LOAD DATA` 命令可以向数据库或者表导入数据，但不支持仅向表中的某一列导入数据。

注意事项

- 权限聚合。按照Database > Table[Group] > Column由高到低的权限级别，高级别聚合低级别的所有权限。
- 使用阿里云账号在阿里云官网开通分析型数据库MySQL版服务后即可获取创建数据库的权限，无法通过授权方式获取创建数据库的权限。
- 部分查询操作不需要SELECT权限，例如 `SELECT now()` 。
- 导出数据需要 `DUMP DATA` 、 `SELECT` 权限以及导出目的地的数据写入相关权限。

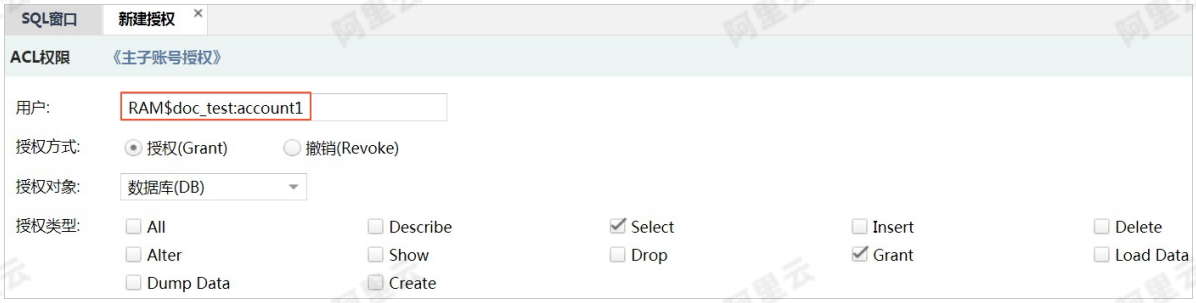
4.3.3. 使用DMS进行权限管理

AnalyticDB for MySQL 2.0中，数据库创建者可以通过DMS页面或者SQL语法为阿里云账号或者RAM子账号授予一定的数据库权限，然后您可以通过授权过的账号在权限范围内进行数据库操作。当您不再需要通过这些账号进行数据库操作时，数据库创建者也可以回收账号权限。

通过SQL语法授权和撤销权限，请参见GRANT和REVOKE（2.0版）。

授权阿里云账号/RAM子账号

1. 登录AnalyticDB 控制台。
2. 在页面左上角，选择集群所在地域。
3. 单击目标数据库右侧的**登录数据库**。
4. 在**DMS for AnalyticDB**页面的顶部菜单栏单击**主子账号授权**。
5. 在**新建授权**页面，选择**用户**，**授权方式**选择**授权（GRANT）**，选择**授权对象**和**授权类型**。
   - **用户为阿里云账号时**，账号格式为 `ALIYUN$account_name`，其中 `ALIYUN$` 为阿里云账号前缀，标识该账号为阿里云账号；`account_name` 为阿里云账号的账号名，例如 `ALIYUN$doc_test`。
   - **用户为RAM子账号时**，账号格式为 `RAM$account_name:subaccount_name`，其中 `RAM$` 为RAM子账号前缀，标识该账号为RAM子账号；`account_name` 为阿里云账号名；`subaccount_name` 为RAM子账号的账号名。例如 `RAM$doc_test:account1`。
   - **权限对象**：授权的对象层级，包括**数据库（DB）**、**表组（Table Group）**、**表（Table）**、**列（Column）**。
   - 权限类型，请参见ACL权限体系。



6. 单击**保存**和**确定**完成授权操作。

撤销阿里云账号/RAM子账号权限

1. 登录AnalyticDB 控制台。
2. 在页面左上角，选择集群所在地域。
3. 单击目标数据库右侧的**登录数据库**。
4. 在**DMS for AnalyticDB**页面的顶部菜单栏单击**主子账号授权**。
5. 在**新建授权**页面，选择**用户**，**授权方式**选择**撤销（Revoke）**，选择**授权对象**和**授权类型**。
6. 单击**保存**和**确定**完成撤销权限操作。

相关文档

- ACL权限体系
- 账号类型介绍
- 用户管理

4.4. 通过DataWorks跨账号迁移RDS for SQLServer数据到AnalyticDB

通过DataWorks可以实现多种数据源之间的迁移，本文档将通过infra-doc阿里云账号下的子账号123456xxx创建RDS for SQLServer实例，dtplus_doc阿里云账号创建Dataworks和AnalyticDB为例，详细为您介绍如何通过Dataworks实现不同账号下的RDS数据迁移至AnalyticDB。

背景信息

同一阿里云账号下的数据迁移，请参见通过Dataworks实现相同账号下的RDS for SQLServer数据迁移至AnalyticDB。

下面以RDS for SQLServer数据源为例，介绍如果如何通过数据集成把数据同步到AnalyticDB中，使用AnalyticDB进行数据分析。

- 配置SQLServer数据源
- 配置AnalyticDB数据源
- 配置同步任务中的数据来源和去向

4.5. DTS跨账号迁移

4.5.1. 通过DTS跨账号迁移RDS数据到AnalyticDB

数据传输服务（Data Transmission Service，简称DTS）可以将RDS for MySQL（以下简称RDS）中的数据实时同步至分析型数据库（AnalyticDB）。其中，RDS、DTS、AnalyticDB所属账号分为以下四种场景。

- **RDS使用A账号，DTS+AnalyticDB使用B账号：**
  i. A账号登录RDS，在RDS中通过RAM角色授权方式为账号B授权。
  ii. B账号登录DTS，在配置链路中数据源选择**其他阿里云账号下的RDS实例**。
  iii. 实施步骤。
- **用户有两个账号，RDS+DTS使用A账号，AnalyticDB使用B账号：**
  i. B账号登录AnalyticDB，为A账号和DTS公共云账号授权。
  ii. A账号登录DTS，手动在DTS中加白。
- **用户有两个账号，RDS+AnalyticDB使用A账号，DTS使用账号B：**
  i. A账号登录RDS，在RDS中通过RAM角色授权方式为B账号授权。
  ii. A账号登录AnalyticDB，为B账号和DTS公共云账号授权。
  iii. B账号登录DTS，手动在DTS中加白。
- **用户有一个账号，RDS+dts+AnalyticDB使用子账号购买**
  i. 主账号登录AnalyticDB，为DTS公共账号授权
  ii. 子账号登录DTS手动在DTS中加白

4.5.2. RDS使用A账号，DTS+AnalyticDB使用B账号

由于RDS使用A账号，DTS+AnalyticDB使用B账号，B账号无法读取RDS信息。因此，需要通过RDS所属账号登录RAM控制台，在RMA中通过角色的方式为DTS、AnalyticDB所属账号授权，使DTS可以读取到RDS信息。

步骤一： 新建RAM角色

1. 使用RDS实例所属账号登录RAM控制台。
2. 单击**RAM角色管理**>**新建RAM角色**。在**新建RAM角色**页面：
3. 在**新建RAM角色**页面，配置如下参数：

- **可信实体类型**选择**阿里云账号**。
- **受信云账号ID**选择**其他云账号**并填写DTS+AnalyticDB所属账号的ID。
- **填写RAM角色名称**，不超过64个字符，允许英文字母、数字，或”-“。

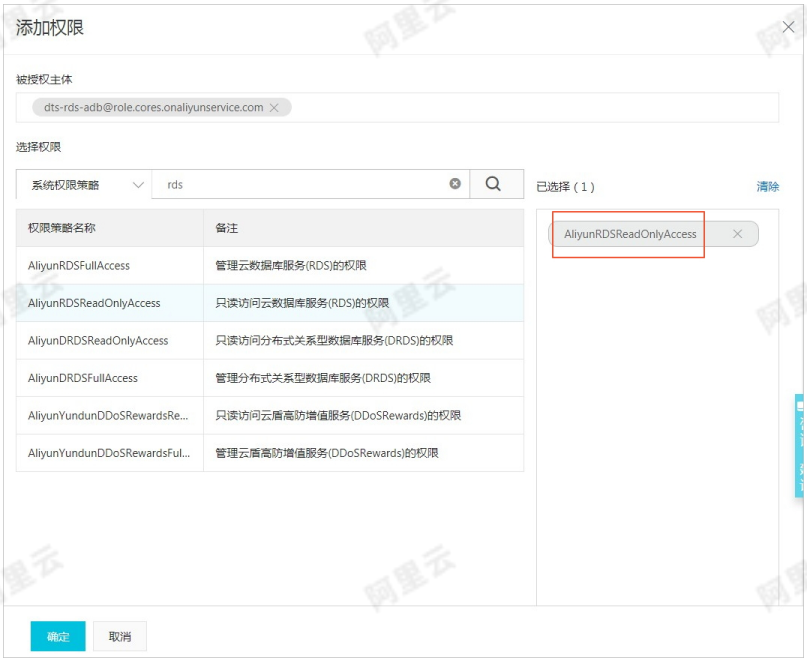
步骤二： 修改角色授信策略

在RAM角色列表中单击刚刚创建的角色，查看角色基本信息，单击**信任策略管理**>**修改信任策略**，用以下策略替换原始策略。

```
{
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "RAM": [
                    "acs:ram::1079926896999421:root"
                ],
                "Service": [
                    "1079926896999421@dts.aliyuncs.com"
                ]
            }
        }
    ],
    "Version": "1"
}
```

步骤三： 修改角色权限

在RAM角色列表中单击刚刚创建的角色，单击**添加权限**为角色添加如下两个**系统权限策略**：**AliyunRDSReadOnlyAccess**。



实施步骤

1. 步骤一：创建DTS同步作业
2. 步骤二：配置同步链路

步骤一：创建DTS同步作业

创建DTS同步作业需要用户支付一定的费用，DTS支持两种付费方式：包年包月（预付费）和按量付费。关于两种付费方式的价格详情，请参见DTS产品定价。

本例以按量付费为例，介绍创建同步作业的详细步骤。

1. 进入DTS产品详情页。
2. 单击**立即购买**。
3. 售卖页面上各项参数说明如下表所示，完成参数配置后，单击**立即购买**。

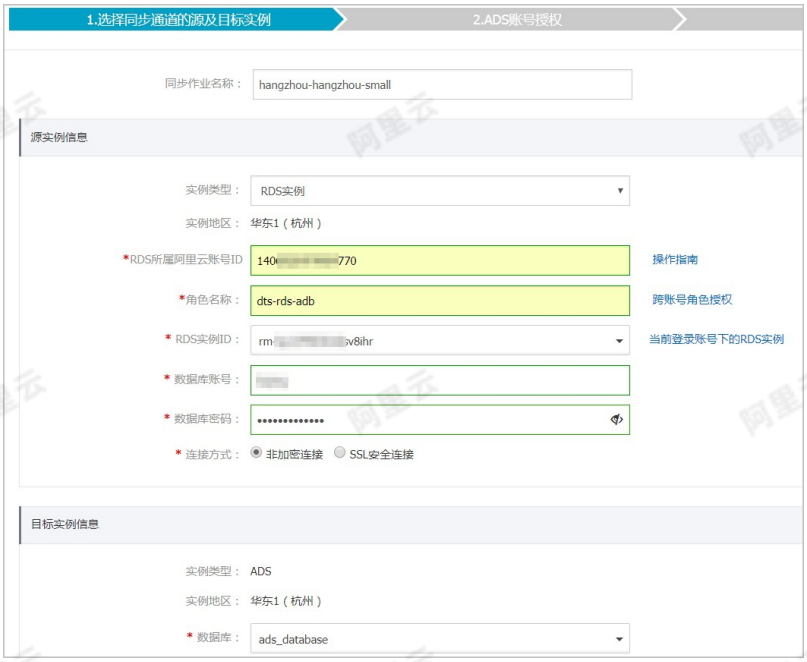
功能	数据同步。
源实例	MySQL。
源实例地域	本例选择华东1（杭州）。
目标实例	分析型数据库AnalyticDB。
目标实例地域	本例选择华东1（杭州）。
同步拓扑	单向同步。
网络类型	专线
同步链路规格	本例选择small

4. 在**确认订单**页面，勾选**《数据传输服务（按量付费）服务协议》**，根据提示完成支付流程。

步骤二：配置同步链路

1. 登录DTS控制台。

2. 在**数据传输**页面，单击左侧导航栏中的**数据同步**。
3. 选择**地域**。
4. 在**同步作业**列表中，单击目标实例右侧的**配置同步链路**，在**选择同步通道的源及目标实例**页面进行参数配置，详细的参数配置如下表所示。



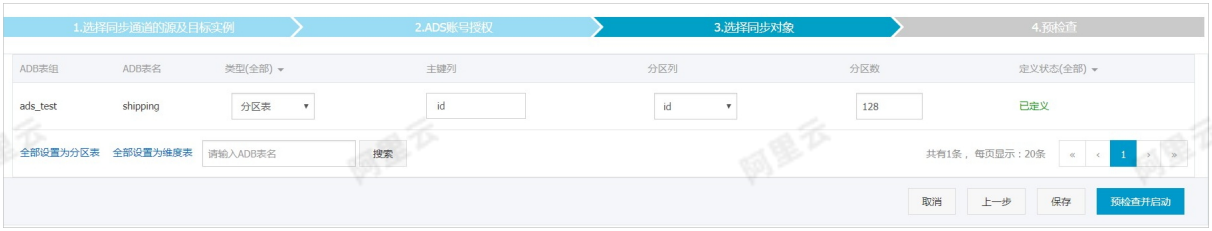
同步作业名称	可选项
实例类型	本例选择RDS。
源实例地区	本例为本例为华东1（杭州）。
实例ID	选择**其他阿里云账号下的RDS实例**
RDS所属阿里云账号ID	RDS实例所属阿里云账号ID
角色名称	在RAM中创建的角色
RDS实例ID	RDS实例的ID
数据库账号	RDS中数据库的账号
数据库密码	RDS中数据库的账号对应的密码。
连接方式	非加密连接
实例类型	AnalyticDB。
目标实例地区	本例为华东1（杭州）。
数据库	本例为ads_DataBase。

5. 完成上述参数配置后，单击**授权白名单并进入下一步**。
6. 在ADS账号授权中，将AnalyticDB数据库的相关权限授权给同步账号，然后单击**下一步**。
7. 在**选择同步对象**页面，完成下面两步骤配置后，单击**下一步**。

i. 勾选**结构初始化**和**全量数据初始化**。

ii. 在**源库对象** 中把同步的表添加到右侧的**已选择对象**中。

8. 进入配置表信息页面，完成参数配置。



类型	分区表
主键	支持复合主键，保证数据唯一。
聚集列	可选项，不填写。
分区列	选取参考一级分区。
分区数	建议128。

9. 完成上述参数配置后，单击**预检查并启动**，弹出**预检查**页面。

i. 如果预检查显示**失败**，可以根据提示DTS预检测进行排错处理。

ii. 预检查全部成功后，单击**关闭**。

查看同步情况

1. 返回DTS控制台，在**同步列表**中的**同步概况**中查看同步**延时**和**速度**。
2. 进入AnalyticDB控制台，在**ads_DataBase**数据库中可以查看到同步过来的数据表。

注意事项

- 如果目标表中列信息与源表不同，DTS支持字段映射功能。详细步骤参见库表列映射。
- 如果需要同步的表数量较少，且AnalyticDB表结构与源端表差异较大，可以在AnalyticDB中提前步骤二：配置同步链路**步骤7**中**结构初始化**的勾选项去掉即可。

4.5.3. RDS+DTS使用A账号，AnalyticDB使用B账号

创建DTS同步作业需要用户支付一定的费用，DTS支持两种付费方式：包年包月（预付费）和按量付费。关于两种付费方式的价格详情，请参见DTS产品定价。

步骤一：B账号登录AnalyticDB，为A账号和DTS公共云账号授权

- 为A账号授予AnalyticDB数据库的describe、select权限：

```
GRANT describe,select ON ads_database.* TO 'ALIYUN$infra-doc';
```

- 为DTS公共云账号授予AnalyticDB数据库的describe、select权限：

```
GRANT describe,select ON ads_database.* TO 'ALIYUN$dts_public_cloud@aliyun-inner.com';
```

步骤二：A账号登录DTS，手动在DTS中加白

1. 登录DTA控制台

执行结果

实施步骤

1. 创建DTS同步作业
2. 配置同步链路

步骤一：创建DTS同步作业

本例以按量付费为例，介绍创建同步作业的详细步骤。

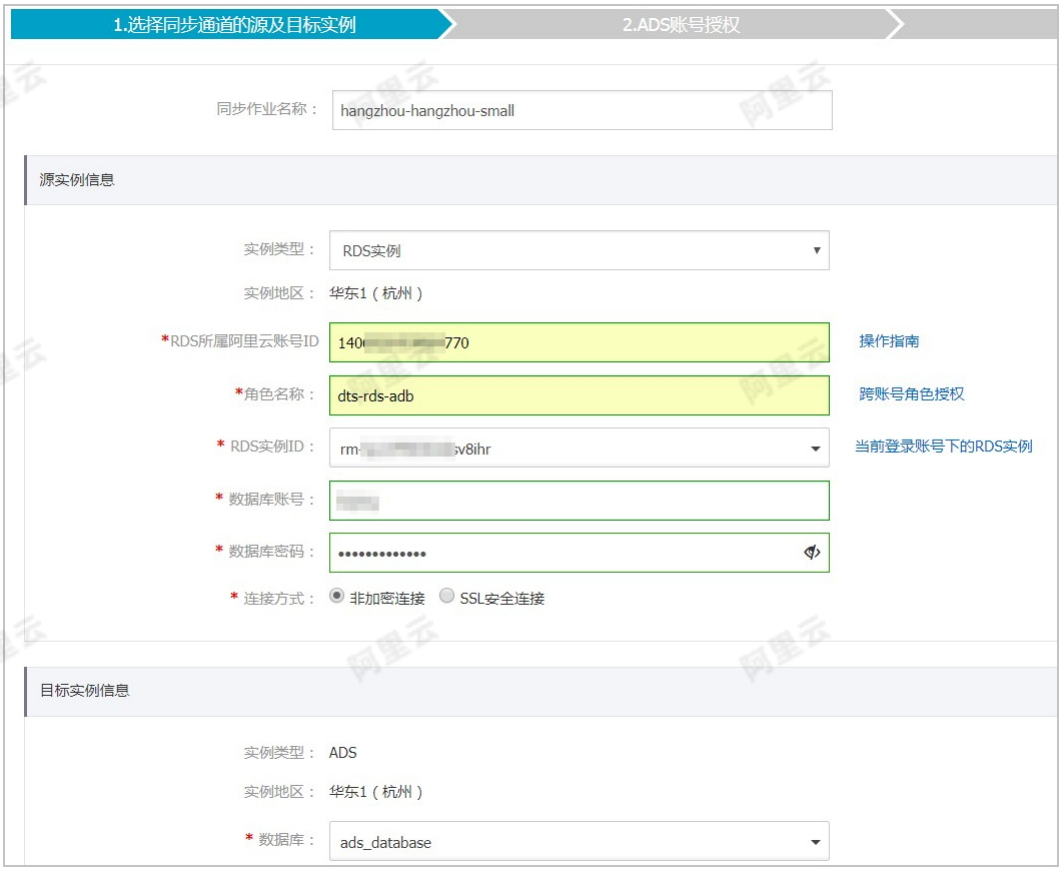
1. 进入DTS产品详情页，单击**立即购买**。
2. 售卖页面上各项参数说明如下表所示，完成参数配置后，单击**立即购买**。

功能	数据同步。
源实例	MySQL。
源实例地域	本例选择华东1（杭州）。
目标实例	分析型数据库AnalyticDB。
目标实例地域	本例选择华东1（杭州）。
同步拓扑	单向同步。
网络类型	专线
同步链路规格	本例选择small

3. 在**确认订单**页面，勾选**《数据传输服务（按量付费）服务协议》**，根据提示完成支付流程。

步骤二：配置同步链路

1. 登录DTS控制台。
2. 在**数据传输**页面，单击左侧导航栏中的**数据同步**。
3. 选择**地域**。
4. 在**同步作业**列表中，单击目标实例右侧的**配置同步链路**，在**选择同步通道的源及目标实例**页面进行参数配置，详细的参数配置如下表所示。

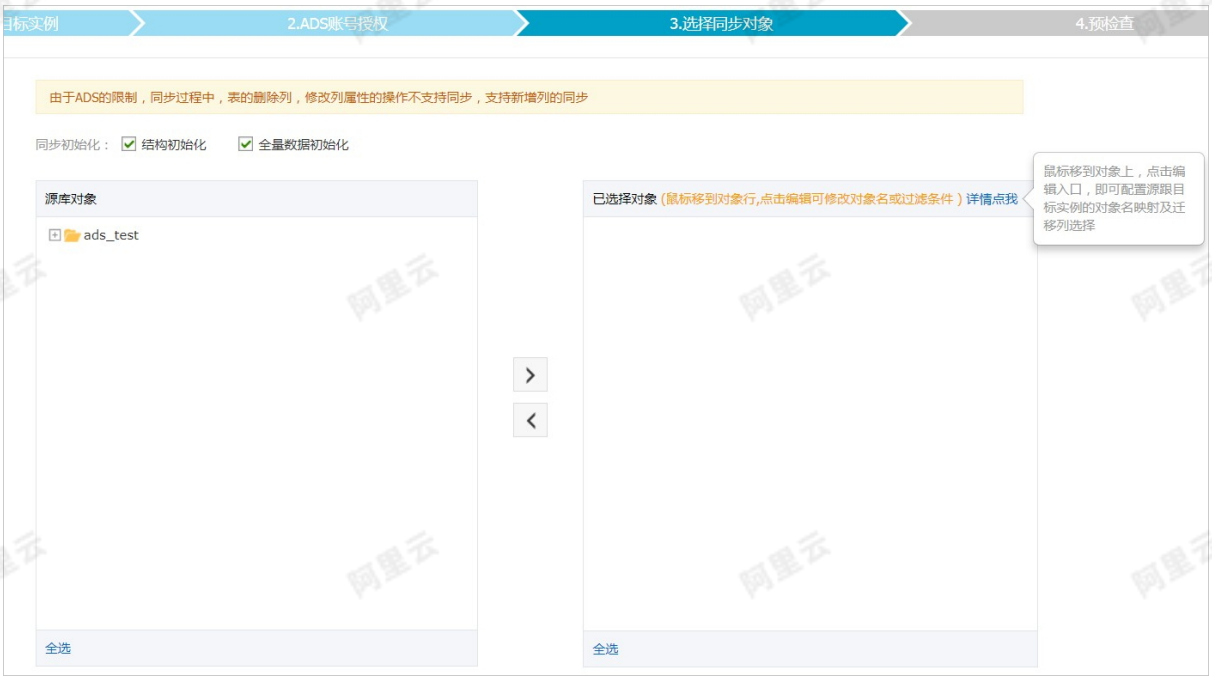


同步作业名称	可选项
实例类型	本例选择RDS。
源实例地区	本例为本例为华东1（杭州）。
实例ID	**选择其他阿里云账号下的RDS实例**
RDS所属阿里云账号ID	RDS实例所属阿里云账号ID
角色名称	在RAM中创建的角色
RDS实例ID	RDS实例的ID

数据库账号	RDS中数据库的账号
数据库密码	RDS中数据库的账号对应的密码。
连接方式	非加密连接
实例类型	AnalyticDB。
目标实例地区	本例为华东1（杭州）。
数据库	本例为ads_DataBase。

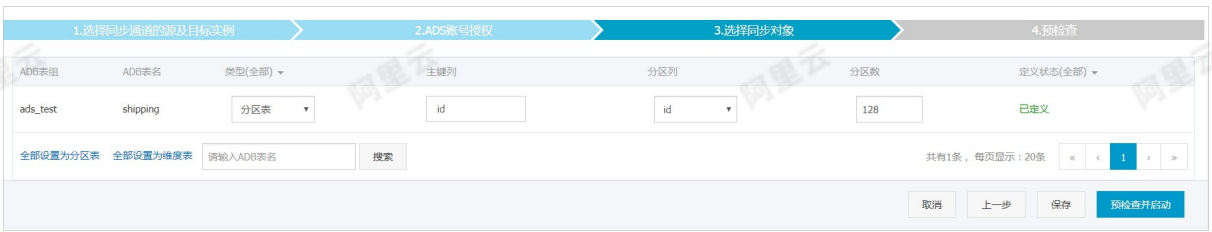
5. **完成上述参数配置后，单击授权白名单并进入下一步。**
6. **在ADS账号授权中，将AnalyticDB数据库的相关权限授权给同步账号，然后单击下一步。**

7. **在选择同步对象页面，完成下面两步骤配置后，单击下一步。**



i. **勾选结构初始化和全量数据初始化**
ii. **在源库对象 中把同步的表添加到右侧的已选择对象中。**

8. **进入配置表信息页面，详细参数说明如下**



类型	分区表

主键	支持复合主键，保证数据唯一
聚集列	可选项，不填写
分区列	选取参考一级分区列选择
分区数	建议128

9. 完成上述参数配置后，单击**预检查并启动**，弹出**预检查**页面。

i. 如果预检查显示**失败**，可以根据提示DTS预检查进行排错处理。

ii. 预检查全部成功后，单击**关闭**

查看同步情况

- 返回DTS控制台，在**同步列表**中的**同步概况**中查看同步**延时**和**速度**。
- 进入AnalyticDB控制台，在**ads_DataBase**数据库中可以查看到同步过来的数据表。

注意事项

- 如果目标表中列信息与源表不同，DTS支持字段映射功能。详细步骤参见库表列映射。
- 如果需要同步的表数量较少，且AnalyticDB表结构与源端表差异较大，可以在AnalyticDB中提前配置同步链路**步骤7**中**结构初始化**的勾选项去掉即可。

4.6. 管理RAM子账号的GRANT权限

由于之前分析型数据库MySQL版权限体系中无法为账号授予GRANT权限，只有数据库OWNER可以给其他账号（主账号或者子账号）授权，导致数据库授权用户不能使用DTS同步数据，每次都需要手动处理，严重影响链路体验。为此，分析型数据库MySQL版新增了GRANT权限授权功能。

GRANT权限授权规则

以下列出了分析型数据库MySQL版中GRANT权限授权规则，只有符合以下授权规则才能授权成功。

数据库增加GRANT OPTION权限。

GRANT OPTION权限只在数据库范围有效，除了数据库OWNER以外，只有数据库级别有ALL和GRANT OPTION权限的账号才可以给其他账号授予GRANT权限。

数据库OWNER拥有数据库最高权限（所有权限），并且可以为其他用户（主账号或者子账号）授予数据库级别的GRANT权限，无法为其他用户授予表级别的GRANT权限。

被授予GRANT权限的用户不能给其他用户（主账号或者子账号）授予GRANT权限，仅可以为其他用户授予自己所拥有的数据库级别或者表级别权限。

只有GRANT权限的授权人能执行REVOKE取消GRANT权限。

GRANT授权示例

主账号授予GRANT OPTION和 SELECT权限给子账号。

5.访问2.0集群

5.1. 业务系统连接2.0集群

5.1.1. Java访问（2.0版）

本文介绍Java中如何通过MySQL JDBC连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。

MySQL JDBC驱动版本

AnalyticDB for MySQL支持以下版本的MySQL JDBC驱动。

- 5.0版本系列：5.0.2，5.0.3，5.0.4，5.0.5，5.0.7，5.0.8。
- 5.1版本系列：
  5.1.1，5.1.2，5.1.3，5.1.4，5.1.5，5.1.6，5.1.7，5.1.8，5.1.11，5.1.12，5.1.13，5.1.14，5.1.15，5.1.16，5.1.17，5.1.18，5.1.19，5.1.20，5.1.21，5.1
  5.1.32, 5.1.33, 5.1.34。

注意事项

Java中创建MySQL JDBC连接依赖于MySQL-JDBC驱动包，您需要手动将MySQL-JDBC驱动包（mysql-connector-java-x.x.x.jar）加入到 `CLASSPATH` 中，否则无法创建MySQL JDBC连接。

不带重试的JDBC连接示例

您可以在业务系统的Java代码中添加以下代码，通过MySQL JDBC连接AnalyticDB for MySQL数据库。

```
Connection connection = null;
Statement statement = null;
ResultSet rs = null;
try {
    Class.forName("com.mysql.jdbc.Driver");
    //adb_url是AnalyticDB for MySQL实例的连接地址URL，可以在控制台的集群信息页面获取连接URL，3306是端口号。
    //db_name是AnalyticDB for MySQL实例的名称。
    String url = "jdbc:mysql:/adb_url:3306/db_name?useUnicode=true&characterEncoding=UTF-8";
    Properties connectionProps = new Properties();
    //my_access_key_id为阿里云账号或者RAM子账号的AccessKeyId。
    connectionProps.put("user", "my_access_key_id");
    //my_access_key_secret为阿里云账号或者RAM子账号的AccessKeySecret。
    connectionProps.put("password", "my_access_key_secret");
    connection = DriverManager.getConnection(url, connectionProps);
    statement = connection.createStatement();
    String query = "select count(*) from information_schema.tables";
    rs = statement.executeQuery(query);
    while (rs.next()) {
        System.out.println(rs.getObject(1));
    }
} catch (ClassNotFoundException e) {
    e.printStackTrace();
} catch (SQLException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (statement != null) {
        try {
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (connection != null) {
        try {
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

带重试的JDBC连接示例

在JDBC中通过配置参数可以实现连接重试机制。

```
public static final int MAX_QUERY_RETRY_TIMES = 3;
public static Connection conn = null;
public static Statement statement = null;
public static ResultSet rs = null;
public static void main(String[] args) throws ClassNotFoundException {
```

```
    //AnalyticDB for MySQL实例的名称。
    String yourDB = "user_db";
    //my_access_key_id为阿里云账号或者RAM子账号的AccessKeyId。
    String username = "my_access_key_id";
    //my_access_key_secret为阿里云账号或者RAM子账号的AccessKeySecret。
    String password = "my_access_key_secret";
    Class.forName("com.mysql.jdbc.Driver");
    //adb_url是AnalyticDB for MySQL实例的连接地址URL，可以在控制台的集群信息页面获取连接URL，3306是端口号。
    String url = "jdbc:mysql://adb_url:3306/" + yourDB + "?useUnicode=true&characterEncoding=UTF-8";
    Properties connectionProps = new Properties();
    connectionProps.put("user", username);
    connectionProps.put("password", password);
    String query = "select id from test4dmp.test limit 10";
    int retryTimes = 0;
      //通过循环自动重试。
    while (retryTimes < MAX_QUERY_RETRY_TIMES) {
        try {
            getConn(url, connectionProps);
            execQuery(query);//执行query。
            break; //query执行成功后，结束整个循环。
        } catch (SQLException e) {
            System.out.println("Met SQL exception: " + e.getMessage() + ", then go to retry task ...");
            try {
                if (conn == null || conn.isClosed()) {
                    retryTimes++;
                }
            } catch (SQLException e1) {
                if (conn != null) {
                    try {
                        conn.close();
                    } catch (SQLException e2) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
    // Clear connection resource.
    closeResource();
}
/**
 * Get connection.
 *
 * @param url
 * @param connectionProps
 * @throws SQLException
 */
public static void getConn(String url, Properties connectionProps) throws SQLException {
    conn = DriverManager.getConnection(url, connectionProps);
}
/**
 * Query task execution logic.
 *
 * @param sql
 * @throws SQLException
 */
public static void execQuery(String sql) throws SQLException {
    Statement statement = null;
    ResultSet rs = null;
    statement = conn.createStatement();
    for (int i = 0; i < 10; i++) {
        long startTs = System.currentTimeMillis();
        rs = statement.executeQuery(sql);
        int cnt = 0;
        while (rs.next()) {
            cnt++;
            System.out.println(rs.getObject(1) + " ");
        }
        long endTs = System.currentTimeMillis();
        System.out.println("Elapse Time: " + (endTs - startTs));
        System.out.println("Row count: " + cnt);
        try {
            Thread.sleep(160000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
/**
 * Close connection resource.
 */
public static void closeResource() {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }
    if (statement != null) {
        try {
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

5.1.2. Druid连接池配置（2.0版）

使用JDBC连接池连接云原生数据仓库 AnalyticDB MySQL 版2.0集群时，推荐使用Druid连接池。使用Druid连接池连接分析型数据库MySQL版建议配置 `keepAlive=true` ，并使用1.1.12之后的版本。

```
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${jdbc_url}" />
    <property name="username" value="${jdbc_user}" />
    <property name="password" value="${jdbc_password}" />

    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="5" />
    <property name="minIdle" value="10" />
    <property name="maxActive" value="20" />

    <!-- 配置获取连接等待超时的时间 -->
    <property name="maxWait" value="60000" />

    <!-- 配置间隔多久进行一次检测，检测需要关闭的空闲连接，单位毫秒 -->
    <property name="timeBetweenEvictionRunsMillis" value="2000" />

    <!-- 配置一个连接在连接池中的最小生存时间，单位毫秒 -->
    <property name="minEvictableIdleTimeMillis" value="600000" />
    <property name="maxEvictableIdleTimeMillis" value="900000" />

    <property name="validationQuery" value="select 1" />
    <property name="testWhileIdle" value="true" />
    <property name="testOnBorrow" value="false" />
    <property name="testOnReturn" value="false" />

    <property name="keepAlive" value="true" />
    <property name="phyMaxUseCount" value="100000" />

    <!-- 配置监控统计拦截的filters -->
    <property name="filters" value="stat" />
</bean>
```

5.1.3. Python访问（2.0版）

在python中可以通过MySQLdb的module来访问云原生数据仓库 AnalyticDB MySQL 版2.0集群。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import MySQLdb
# 打开数据库连接
# host为分析型数据库MySQL版实例的访问URL的域名或IP
# port为分析型数据库MySQL版实例的访问URL的端口
# user为分析型数据库MySQL版实例的用户账号（通常为阿里云AK的access id）
# passwd为分析型数据库MySQL版实例的用户密码（通常为阿里云AK的access key）
# db为分析型数据库MySQL版实例的database名称
db = MySQLdb.connect(host='test-db-89b1ac42.cn-xxx.ads.aliyuncs.com', port=9999, user='test_user', passwd='123456', db='test_db')
# 使用cursor()方法获取操作游标
cursor = db.cursor()
# 使用execute方法执行SQL语句
cursor.execute("SELECT VERSION()")
# 使用 fetchone() 方法获取一条数据
data = cursor.fetchone()
print "Database version : %s " % data
# 关闭数据库连接
db.close()
```

5.1.4. PHP访问（2.0版）

通过PHP访问云原生数据仓库 AnalyticDB MySQL 版2.0集群时，如果用户的操作系统是Linux，则需要安装php-mysql 5.1.x模块；如果操作系统是Windows，则需要安装php_MySQL.dll。

```
$strsql="SELECT user_id FROM my_ads_db.my_first_table limit 20;";
$result=mysqli_query($ads_conn, $strsql);
while($row = mysqli_fetch_array($result)) {
  echo $row["user_id"] ; //user_id为列名
}
```

使用mysqli连接云原生数据仓库 AnalyticDB MySQL 版2.0集群

```
//连接云原生数据仓库AnalyticDB MySQL版2.0集群的url，可从云原生数据仓库AnalyticDB MySQL版控制台获取url连接信息
$ads_server_name="mydbname-xxxx.ads-cn-hangzhou-1.aliyuncs.com";
$ads_username="my_access_key_id"; // 连接分析型数据库MySQL版的用户名
$ads_password="my_access_key_secret"; // 连接分析型数据库MySQL版的密码
$ads_database="my_ads_db"; // 分析型数据库MySQL版的名字
$ads_port=3003; //分析型数据库MySQL版端口号，可从分析型数据库MySQL版控制台获取端口号
// 连接云原生数据仓库AnalyticDB MySQL版2.0集群
$ads_conn=mysqli_connect($ads_server_name,$ads_username,$ads_password,$ads_database, $ads_port);
```

```
$strsql="SELECT user_id FROM my_ads_db.my_first_table limit 20;";
$result=mysqli_query($ads_conn, $strsql);
while($row = mysqli_fetch_array($result)) {
 echo $row["user_id"] ; //user_id为列名
}
```

使用PDO连接云原生数据仓库 AnalyticDB MySQL 版2.0集群

```
$ads_server_name = "xxxxx.cn-beijing-1.ads.aliyuncs.com";
$ads_username = "my_access_key_id"; // 连接云原生数据仓库AnalyticDB MySQL版的用户名
$ads_password = "my_access_key_secret"; // 连接云原生数据仓库AnalyticDB MySQL版的密码
$ads_database = 'my_ads_db'; // 云原生数据仓库AnalyticDB MySQL版的名字
$ads_port = 10024;
$dsn = "mysql:host={$ads_server_name};dbname={$ads_database};port={$ads_port}";
try {
    $dbh = new PDO($dsn, $ads_username, $ads_password);
    echo 'PDO Success !';
} catch (PDOException $e) {
    echo 'PDO Connection failed: ' . $e->getCode() ."\n" . $e->getMessage() ."\n". $e->getTraceAsString();
}
```

5.1.5. C#访问（2.0版）

本章节介绍使用MySQL的.NET connector访问云原生数据仓库 AnalyticDB MySQL 版2.0集群。

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
namespace adbdemo
{
    public class Tutorial2
    {
        public static void Main()
        {
            string connStr = "server=...;UID=...;database=...;port=...;password=...;SslMode=none;";

            MySqlConnection conn = new MySqlConnection(connStr);
            try
            {
                Console.WriteLine("Connecting to MySQL...");
                conn.Open();

                string sql = "select c_custkey, c_name from customer limit 1";
                MySqlCommand cmd = new MySqlCommand(sql, conn);
                MySqlDataReader rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    Console.WriteLine(rdr[0] + " --- " + rdr[]);
                }
                rdr.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
            }

            conn.Close();
            Console.WriteLine("Done.");
        }
    }
}
```

5.1.6. MySQL命令行连接（2.0版）

本文介绍如何通过MySQL命令行工具连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。

语法

```
mysql -h<host> -P<port> -u<user_name> -p<password> <db_name> -c -A
```

参数

- `host`：2.0集群的连接地址，通过控制台 > **集群信息** 页面中的连接信息版块获取连接地址。
- `port`：端口为3306。
- `user_name`：阿里云账号或者RAM子账号的AccessKeyId。
  关于AccessKeyId，请参见获取主账号的AccessKeyId和AccessKeySecret。
- `password`：阿里云账号或者RAM子账号的AccessKeySecret。
- `db_name`：2.0集群的名字。
- `-c`：通过MySQL命令行连接2.0集群时，为了识别查询中的HINT，需要指定 `-c` 参数。

示例

```
mysql -h127.0.0.1 -P9999 -uTestUser -pTestPassword test_db -c -A
```

5.1.7. Golang访问（2.0版）

本文介绍如何使用Golang连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。

前提条件

- 下载并安装Golang，请参见Golang。
- 安装Golang MySQL Driver。
  i. 下载Golang MySQL Driver，请参见 Golang MySQL Driver。
  ii. 使用Shell中的go工具将驱动包安装到 `$GOPATH` 中。

```
go get github.com/go-sql-driver/mysql
```

> **说明**
> 要求在您的主机和系统的PATH中安装了Git。

连接AnalyticDB MySQL 2.0

```
package main
import (
    "database/sql"
    "fmt"
    _ "github.com/go-sql-driver/mysql"
)
const (
    user = "xxx"
    password = "xxx"
    host = "127.0.0.1"
    port = 3306
    database = "xxx"
    // 数据库连接的超时时间
    connectTimeout = "10s"
)
func main() {
    // 打开数据库连接
    //user为AnalyticDB MySQL 2.0集群的用户账号，通常为阿里云AK的AccessKey ID。
    //password为AnalyticDB MySQL 2.0集群的用户密码，通常为阿里云AK的Access Key Secret。
    //host为AnalyticDB MySQL 2.0集群的访问URL的域名或IP。
    //port为AnalyticDB MySQL 2.0集群的访问URL端口。
    //database为AnalyticDB MySQL 2.0集群的名称。
    url := fmt.Sprintf("%s:%s@tcp(%s:%d)/%s?timeout=%s", user, password, host, port, database, connectTimeout)
    db, err := sql.Open("mysql", url)
    if err != nil {
        panic(err.Error())
    }
    // 设置最大打开的连接数，默认值为0，表示不限制。
    db.SetMaxOpenConns(2)
    // 设置最大闲置的连接数。
    db.SetMaxIdleConns(1)
    // 设置连接的最大生命周期，默认连接总是可重用。
    // 不能保证连接将在连接池中存在完整的1小时，很可能由于某种原因连接将变得不可用，并且在此之前自动关闭。
    // 这不是空闲超时。连接将在第1次创建后1小时后过期，而不是1小时后变成空闲。
    // 理论上，ConnMaxLifetime越短，从0开始创建连接的频率就越高。
    db.SetConnMaxLifetime(time.Hour)
    // defer the close till after the main function has finished
    // executing
    defer db.Close()
    rows, err := db.Query("show tables")
    if err != nil {
        panic(err.Error())
    }
    for rows.Next() {
        var tableName string
        err := rows.Scan(&tableName)
        if err != nil {
            panic(err.Error())
        }
        fmt.Println(tableName)
    }
}
```

开启客户端的PrepareStatement

目前AnalyticDB MySQL 2.0不支持服务端预编译，只支持在客户端使用Prepared Statement。

在Go MySQL driver中开启PrepareStatement时，需要配置 `interpolateParams=true` 参数，如下所示。

```
package main
import (
    "database/sql"
    "fmt"
    _ "github.com/go-sql-driver/mysql"
    "time"
)

const (
    user = "xxx"
    password = "xxx"
    host = "127.0.0.1"
    port = 3306
    database = "xxx"
    // 数据库连接的超时时间。
    connectTimeout = "10s"
)
func main() {
    // open the database connection
    url := fmt.Sprintf("%s:%s@tcp(%s:%d)/%s?timeout=%s&interpolateParams=true", user, password, host, port, database, connectTimeout)
    db, err := sql.Open("mysql", url)
    if err != nil {
        panic(err.Error())
    }
    // 设置最大打开的连接数，默认值为0，表示不限制。
    db.SetMaxOpenConns(2)
    // 设置最大闲置的连接数。
    db.SetMaxIdleConns(1)
    // 设置连接的最大生命周期，默认连接总是可重用。
    // 不能保证连接将在池中存在完整的1小时，很可能由于某种原因连接将变得不可用，并且在此之前自动关闭。
    // 这不是空闲超时。连接将在第1次创建后1小时后过期，而不是1小时后变成空闲。
    // 理论上，ConnMaxLifetime越短，从0开始创建连接的频率就越高。
    db.SetConnMaxLifetime(time.Hour)
    defer db.Close()
    rows, err := db.Query("select * from student where student_id = ?", 1)
    if err != nil {
        panic(err.Error())
    }
    defer rows.Close()
    for rows.Next() {
        var schoolId string
        var studentId string
        var studentName string
        err := rows.Scan(&schoolId, &studentId, &studentName)
        if err != nil {
            panic(err.Error())
        }
        fmt.Println(fmt.Sprintf("%s, %s, %s", schoolId, studentId, studentName))
    }
}
```

5.2. 客户端连接2.0集群

5.2.1. Navicat（2.0版）

本文介绍如何通过Navicat连接和管理云原生数据仓库 AnalyticDB MySQL 版2.0集群。

前提条件

开始使用Navicat for MySQL之前，需要先安装Navicat for MySQL。

操作步骤

1. 打开Navicat for MySQL，单击**文件>新建连接>MySQL**。在**新建连接**页面，进行参数配置，详细的参数配置如下所示。

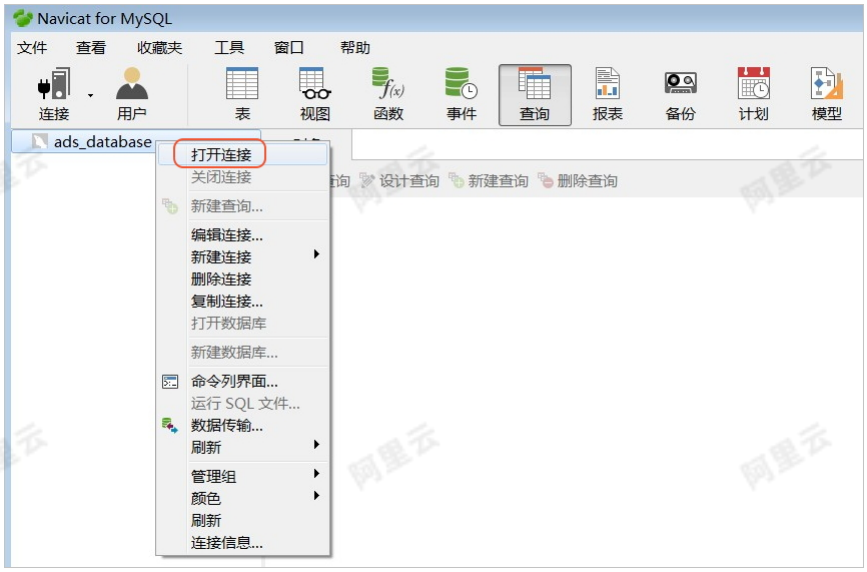
参数	说明
连接名	为数据库连接取一个名字，便于后续管理。
主机名或IP地址	AnalyticDB for MySQL 2.0数据库的连接地址。
端口	数据库对应的端口。
用户名	阿里云账号或者已授权RAM子账号的Access Key ID。
密码	阿里云账号或者已授权RAM子账号的Access Key Secret。

如果您的操作系统是macOS，配置完连接信息后，需要加上数据库名字，如下图所示：

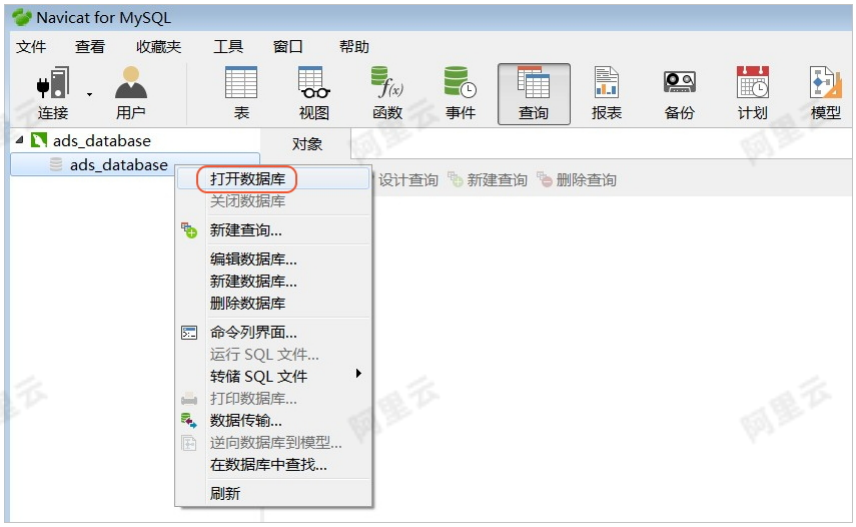
2. 单击**连接测试**测试连通性，测试成功后单击**确定**。

至此已成功建立数据库连接，但连接处于关闭状态需要手动打开连接。

3. 右键单击**连接名**>**打开连接**。



4. 右键单击数据库名**打开数据库**，打开数据库后， 您就可以利用Navicat for MySQL进行数据管理。



5.2.2. DBeaver（2.0版）

本文介绍如何通过DBeaver连接和管理云原生数据仓库 AnalyticDB MySQL 版2.0集群。

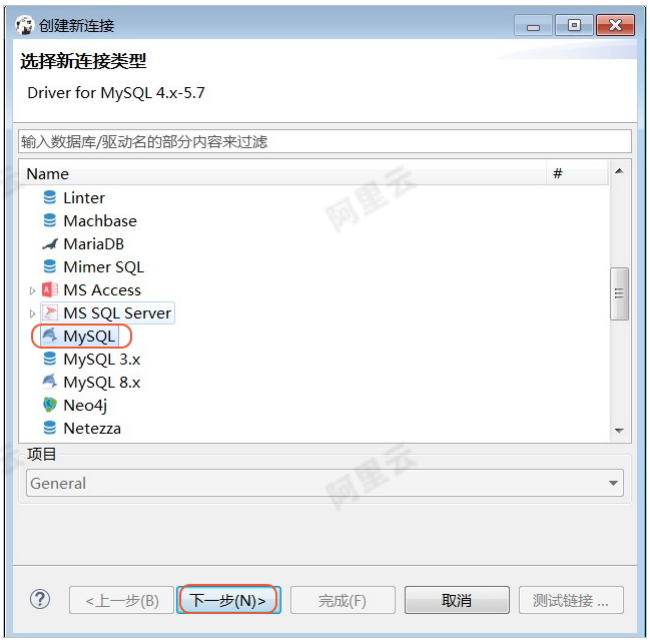
开始使用DBeaver之前，您需要先完成以下准备工作。

- 安装DBeaver。
- 安装MySQL JDBC驱动。

操作步骤

1. 打开**DBeaver**，单击**数据库**>**新建连接**。

2. 在**创建新连接**页面，连接类型选择**MySQL**，单击**下一步**。



3. 在**创建新连接**页面，按照页面提示进行参数配置。

服务器地址	分析型数据库MySQL版的连接地址。
端口	分析型数据库MySQL版对应的端口。
数据库	数据库的名字。
用户名	阿里云账号或者已授权RAM子账号的Access Key ID。
密码	阿里云账号或者已授权RAM子账号的Access Key Secret。

4. 完成上述参数配置后，单击**测试连接...**测试连通性，测试成功后单击**完成**连接AnalyticDB for MySQL 2.0数据库。
   成功连接至AnalyticDB for MySQL 2.0数据库后，您就可以通过DBeaver对AnalyticDB for MySQL 2.0数据库进行管理。

5.2.3. DBVisualizer（2.0版）

本文介绍如何通过DBVisualizer连接和管理云原生数据仓库 AnalyticDB MySQL 版2.0集群。

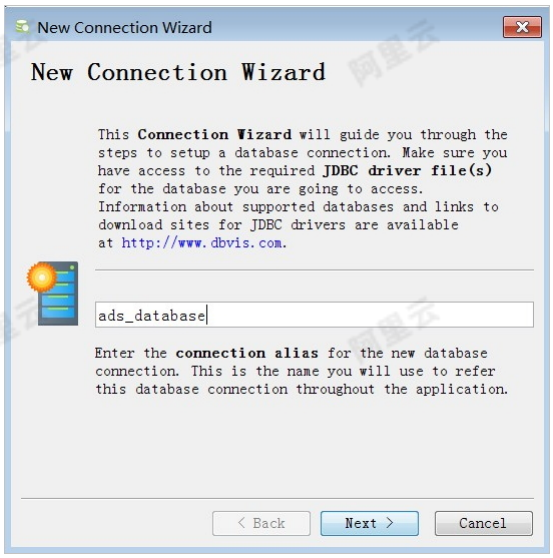
准备工作

开始使用DBVisualizer之前，需要先完成以下准备工作。

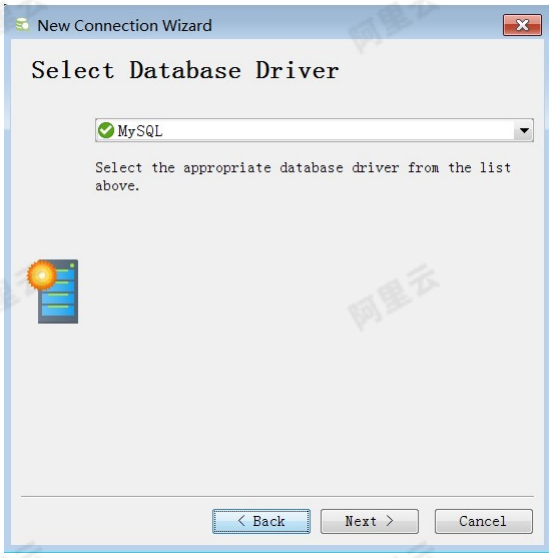
- 下载并安装MySQL JDBC驱动。
- 下载并安装DBVisualizer。

操作步骤

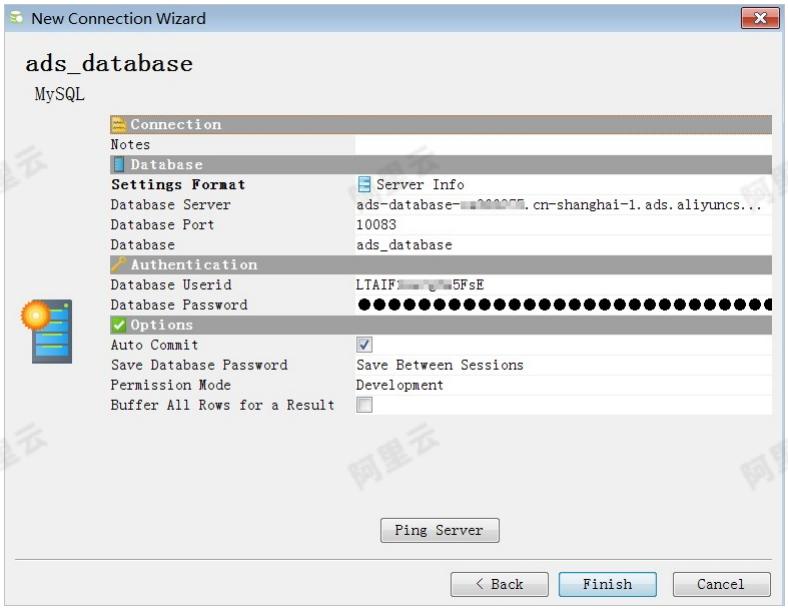
1. 打开DBVisualizer，DBVisualizer提示您为连接输入一个名字，便于后续管理。



2. 单击**Next**，选择**MySQL**作为Database Driver。



3. 单击**Next**，按照页面提示进行连接参数配置。



参数	说明
Notes	连接备注信息。
Database Server	AnalyticDB for MySQL 2.0数据库的连接地址。
Database Port	数据库对应的端口号。
Database	数据库的名字。
Database Userid	阿里云账号或者已授权RAM子账号的Access Key ID。
Database Password	阿里云账号或者已授权RAM子账号的Access Key Secret。

4. 完成上述参数配置后，单击**Ping Server**测试连通性，测试通过后，单击**Finish**。
   成功连接AnalyticDB for MySQL 2.0数据库后，您就可以通过DBVisualizer进行数据管理。

5.2.4. SQL WorkBench/J（2.0版）

本文介绍如何通过SQL WorkBench/J工具连接和管理云原生数据仓库 AnalyticDB MySQL 版2.0集群。

前提条件

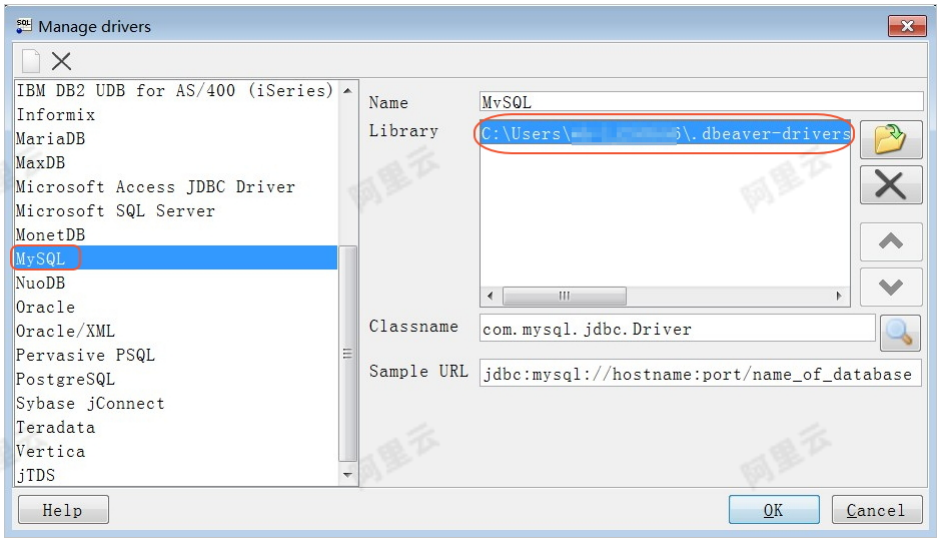
开始使用SQL WorkBench/J之前，您需要先完成以下准备工作。

- 下载并安装MySQL JDBC驱动。
- 下载并安装SQL WorkBench/J。

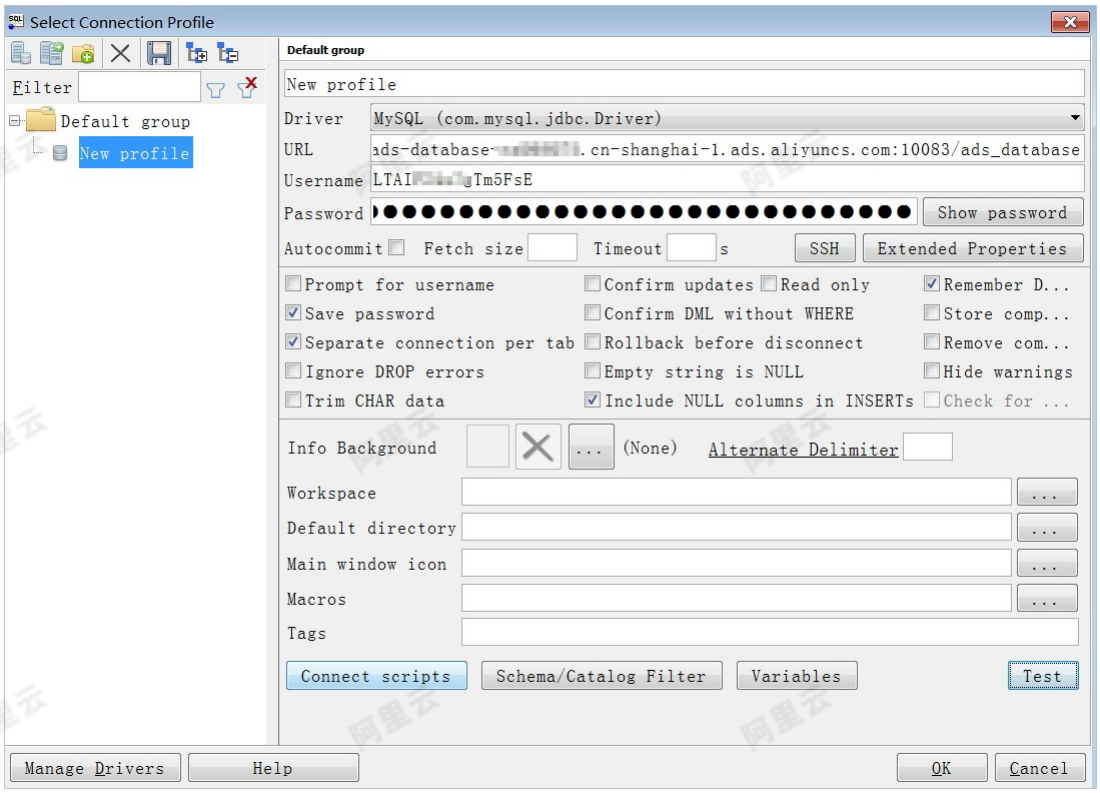
操作步骤

1. 打开SQL WorkBench/J，单击**File**>**Manage Drivers...**。

2. 在**Manage drivers**页面，驱动选择**MySQL**，并添加驱动jar包，单击**OK**。



3. 单击**File>Connect window**，在**Select Connection Profile**页面进行参数配置。



参数	说明
连接名称	连接名字。
Driver	选择**MySQL**。
URL	分析型数据库MySQL版的连接地址，格式为：`jdbc:mysql://hostname:port/name_of_database`。
Username	阿里云账号或者已授权RAM子账号的Access Key ID。
Password	阿里云账号或者已授权RAM子账号的Access Key Secret。

4. 完成上述参数配置后，单击**Test**测试连通性，测试通过后单击**OK**，连接至AnalyticDB for MySQL 2.0数据库。成功连接AnalyticDB for MySQL 2.0数据库后，您就可以对数据库进行管理。

5.3. BI工具连接2.0集群

5.3.1. QlikView（2.0版）

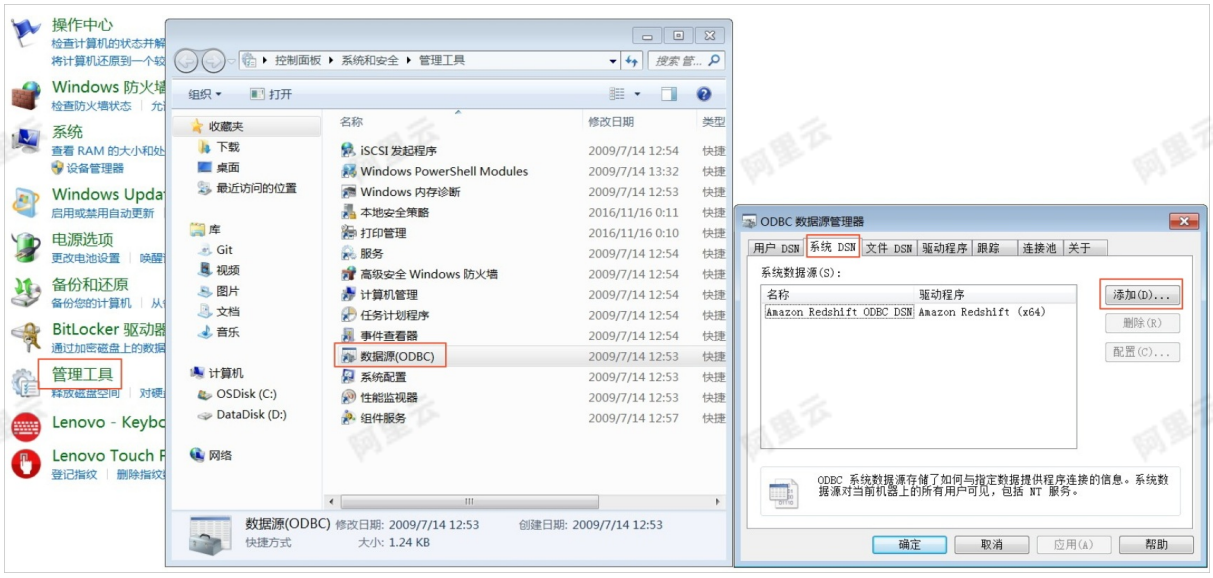
本章节介绍如何通过QlikView连接云原生数据仓库 AnalyticDB MySQL 版2.0集群，并通过QlikView构建BI系统。

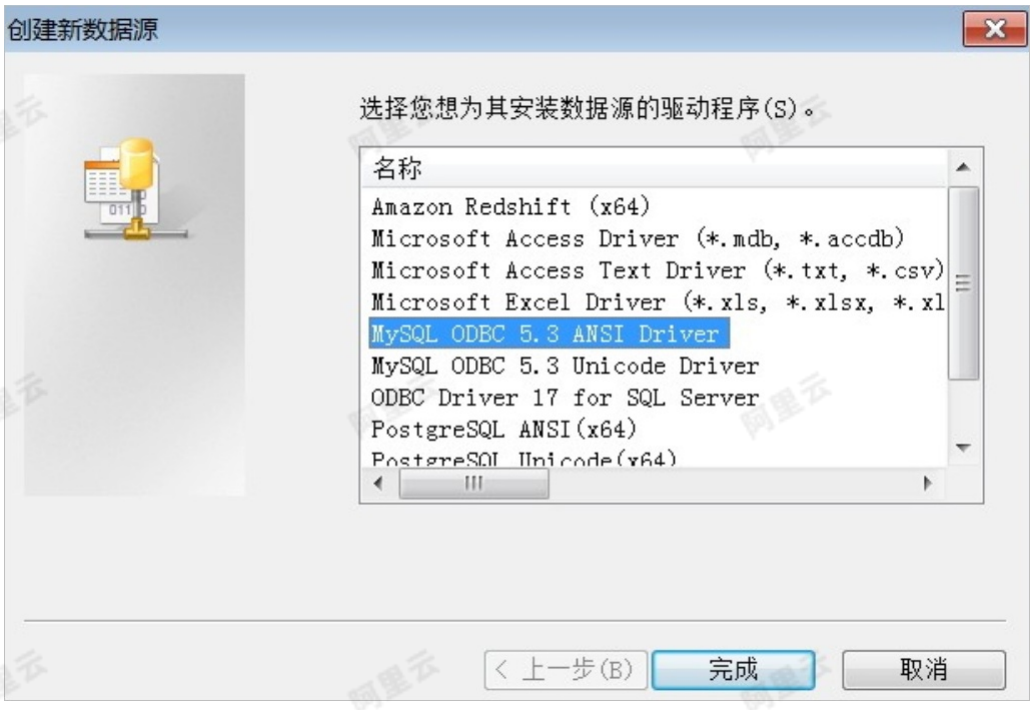
前提条件

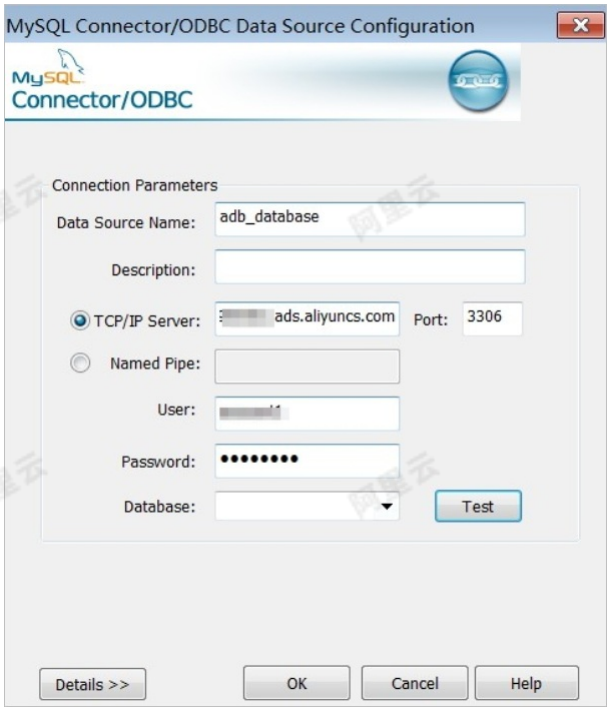
- 安装ODBC MySQL Driver，推荐使用MySQL Connector/ODBC 3.5.1或5.3 版本。
- 安装QlikView 11.20.x版本。

通过QlikView连接AnalyticDB for MySQL

1. 在安装QlikView的主机上，进入**控制面板>系统和安全>管理工具>数据源（ODBC）**（操作系统不同，此步骤可能不同），新建一个DSN，数据源选择**MySQL ODBC 5.xx Driver**。





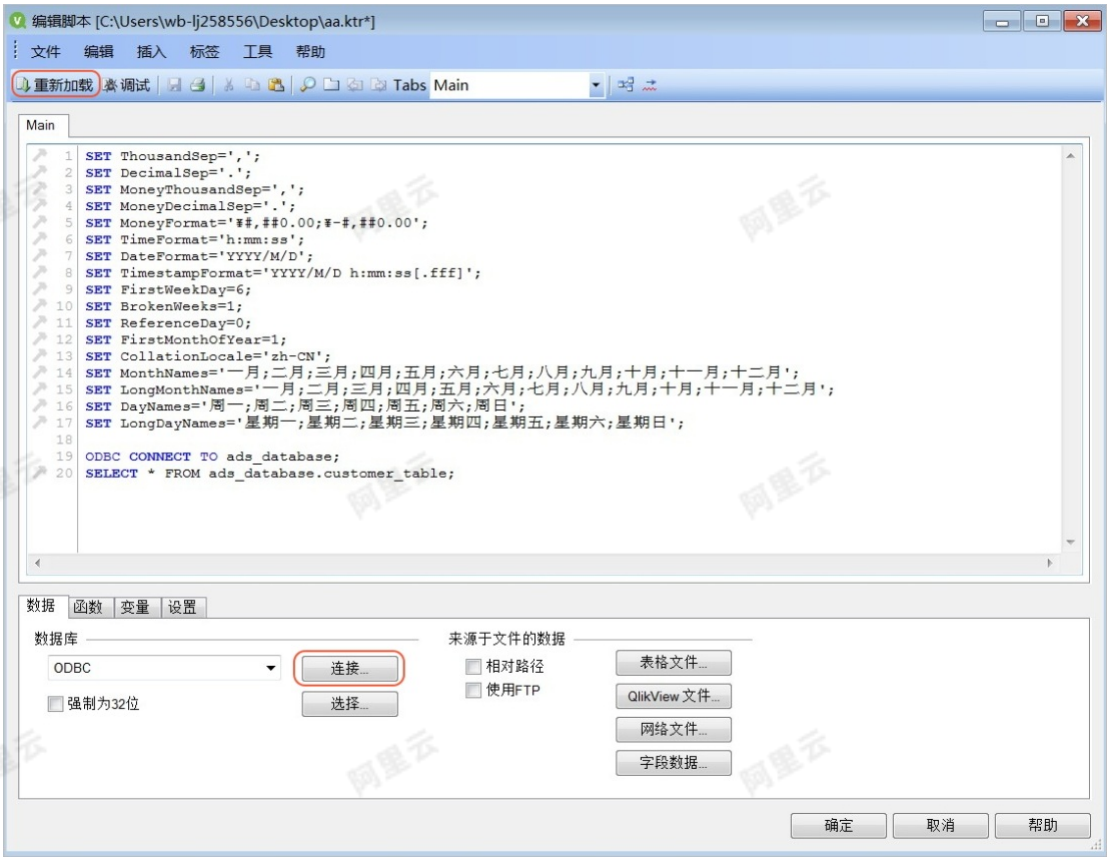


详细的参数配置如下表所示。

参数	说明
Data Source Name	分析型数据库MySQL版的名字。
TCP/IP Server	分析型数据库MySQL版的连接地址。
Port	连接地址对应的端口号。
User	AccessKey ID。
Password	Access Key Secret。
Database	分析型数据库MySQL版的名字。

2. 完成上述参数配置后，单击**Test**测试连通性，测试通过后，单击**OK**添加数据源。
3. 打开QlikView，单击**文件>编辑脚本**，输入以下SQL语句，单击**重新加载**执行SQL。也可以单击上图的**连接...**，测试是否正确连接分析型数据库MySQL版。

```
ODBC CONNECT TO DATABASE_NAME;
SELECT * FROM DATABASE_NAME.TABLE_NAME;
```



使用QlikView

在QlikView中读取到数据库数据后，用户可以利用QlikView进行更多数据操作。QlikView的使用请参见QlikView。

5.3.2. FineReport（2.0版）

本章节介绍如何通过FineReport连接云原生数据仓库 AnalyticDB MySQL 版2.0集群并进行报表管理。

准备工作

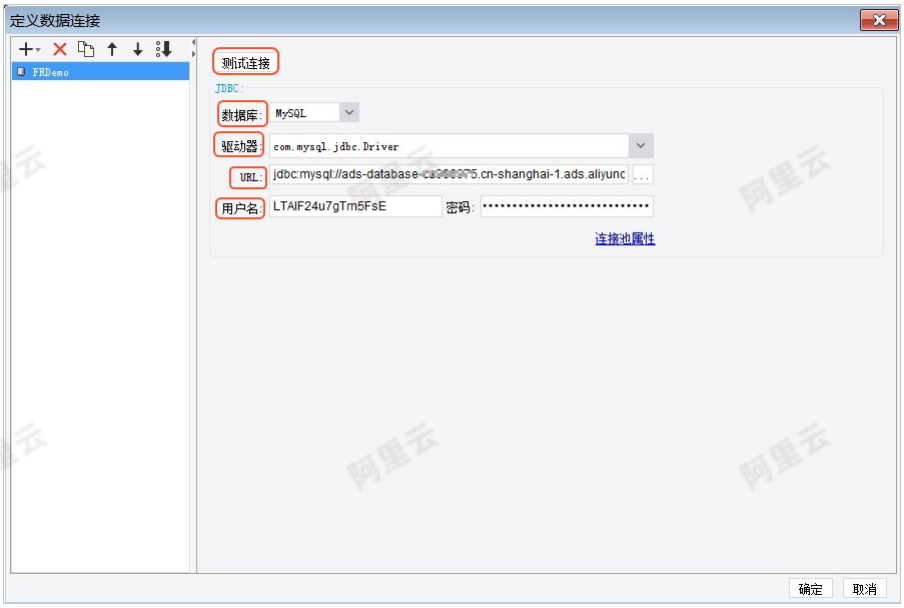
开始使用FineReport之前，用户需要先完成以下准备工作。

- 下载并安装MySQL JDBC驱动。
- 下载并安装FineReport。

操作步骤

1. 打开FineReport，单击**服务器>定义数据连接**。

2. 在**定义数据连接**页面进行以下参数配置，详细的参数说明如下表所示。



配置项	说明
数据库	选择**MySQL**。
驱动器	选择**MySQL jdbc驱动**。
URL	分析型数据库MySQL版的连接地址，格式为：`jdbc:mysql://hostname:port`。
用户名	AccessKey ID。
密码	Access Key Secret。

3. 完成上述参数配置后，单击**测试连接**测试连通性，测试通过后单击**确定**，连接至分析型数据库MySQL版。

使用FineReport

成功连接至分析型数据库MySQL版后，用户便可以在FineReport中利用分析型数据库MySQL版中的数据制作各类报表。FineReport的使用参见如何使用FineReport。

5.3.3. Tableau（2.0版）

Tableau是一款操作简单且功能强大的报表分析工具，支持连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。在Tableau中连接成功后，可以通过拖放或单击的方式快速创建智能视图和仪表板。

开始使用Tableau Desktop之前，您需要先完成以下准备工作：

- 安装Alibaba AnalyticDB for MySQL connector。
- 安装Tableau Desktop 9.0及以上版本。

操作步骤

1. 在Tableau Desktop右侧单击**MySQL**添加数据源，详细的参数配置如下表所示。

参数	说明
服务器	分析型数据库MySQL版的连接地址。
端口号	连接地址对应的端口号。
用户名	AccessKey ID。
密码	Access Key Secret。

2. 完成上述参数配置后，单击**确定**连接分析型数据库MySQL版。

使用Tableau

在Tableau Desktop上，用户可以看到拥有权限的数据库。选择数据库后，便可选取数据表以及预览数据进行可视化报表制作。如何使用Tableau请参见如何使用Tableau。

5.3.4. Quick BI（2.0版）

本章节介绍如何通过阿里云Quick BI连接云原生数据仓库 AnalyticDB MySQL 版2.0集群。

操作步骤

1. 登录Quick BI控制台。
2. 单击**工作空间>数据源**，进入**数据源**管理页面。
3. 单击**新建数据源>分析型数据库MySQL版**。

4. 在**AnalyticDB DB添加数据源**页面进行参数配置。

配置项	说明
显示名称	数据源名称。
数据库地址	分析型数据库MySQL版的连接地址。
端口	连接地址对应的端口号。
数据库	分析型数据库MySQL版的名字。
Access Id	AccessKey ID。
Access Key	Access Key Secret。

5. 完成上述参数配置后，单击**连接测试**测试连通性，测试通过后，单击**添加**添加数据源。

使用Quick BI

成功连接分析型数据库MySQL版后，用户可以参考以下步骤学习如何在Quick BI中完成报表分析等操作。

1. 创建数据集
2. 制作仪表板
3. 制作电子表格
4. 制作数据门户

关于Quick BI的更过功能，请参见Quick BI相关文档。

6.2.0版数据迁移

6.1. 通过DTS将PolarDB数据同步到AnalyticDB MySQL 2.0

6.1.1. 概述

背景信息

数据传输服务（Data Transmission Service，简称DTS）支持将云数据库PolarDB中的数据同步至分析型数据库MySQL版（AnalyticDB for MySQL）。通过DTS提供的数据同步功能，可以轻松实现数据的流转，将企业数据集中分析。

本文以PolarDB for MySQL数据源为例，详细为您介绍如何通过DTS将PolarDB数据同步至分析型数据库MySQL版。

前提条件

参照以下步骤，在PolarDB中准备好测试数据。

1. 创建PolarDB for MySQL数据库集群
2. 设置白名单
3. 创建数据库账号
4. 创建数据库
5. 连接数据库
6. 创建数据表
   本示例在PolarDB中创建 `test_ads` 数据库和 `login_info` 表，`login_info` 表存储5条数据。

```
CREATE TABLE `login_info` (
`id` int(11) NOT NULL,
`username` varchar(32) NOT NULL,
`password` varchar(32) NOT NULL,
`realname` varchar(32) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
insert `login_info`(`id`,`username`,`password`,`realname`) values(1,'Jams','1234','詹姆士');
insert `login_info`(`id`,`username`,`password`,`realname`) values(2,'Jacob','3462D','雅各伯');
insert `login_info`(`id`,`username`,`password`,`realname`) values(3,'Joshua','DS33242','约书亚');
insert `login_info`(`id`,`username`,`password`,`realname`) values(4,'Daniel','436783','丹尼尔');
insert `login_info`(`id`,`username`,`password`,`realname`) values(5,'Christopher','643689','克里斯托弗');
```

7. 开启PolarDB实例的Binlog功能

参照快速入门综述创建分析型数据库MySQL版。

下一步

实施步骤

6.1.2. 实施步骤

通过DTS实时同步PolarDB for MySQL数据到分析型数据库MySQL版需要以下几个步骤：

1. 步骤一：创建DTS同步作业
2. 步骤二：配置同步链路
3. 步骤三：查看同步数据

步骤一：创建DTS同步作业

创建DTS同步作业需要用户支付一定的费用，DTS支持两种付费方式：包年包月（预付费）和按量付费。关于两种付费方式的价格详情，请参见DTS产品定价。

本例以按量付费为例，介绍创建同步作业的详细步骤。

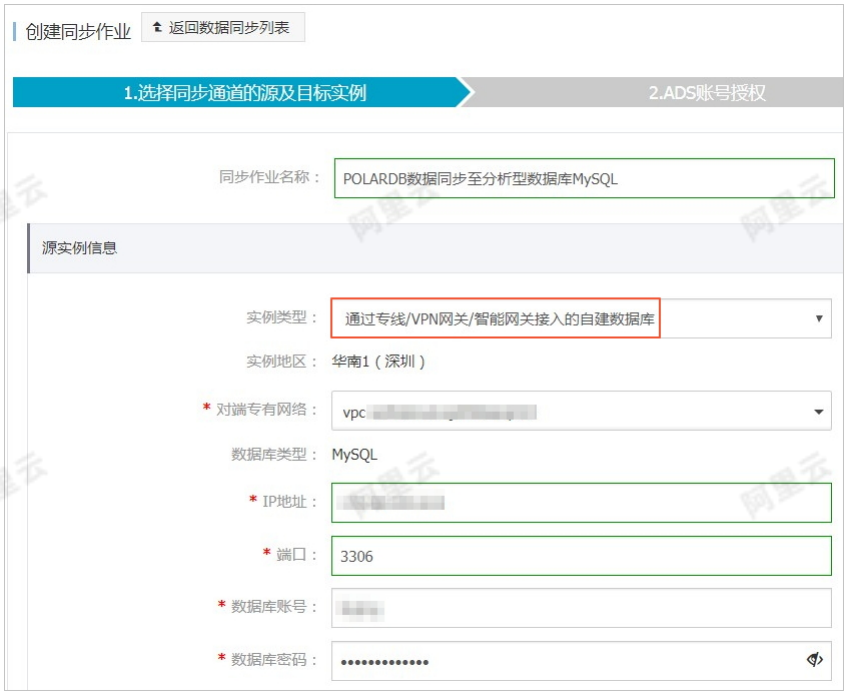
1. 进入DTS产品详情页，单击**立即购买**。
2. 售卖页面上各项参数说明如下表所示，完成参数配置后，单击**立即购买**。

配置项	说明
功能	数据同步。
源实例	MySQL。
源实例地域	本例选择华南1（深圳）。
目标实例	分析型数据库MySQL版。
目标实例地域	本例选择华南1（深圳）。
同步拓扑	单向同步。
网络类型	专线。
同步链路规格	本例选择small。

3. 在**确认订单**页面，勾选**《数据传输服务（按量付费）服务协议》**，根据提示完成支付流程。

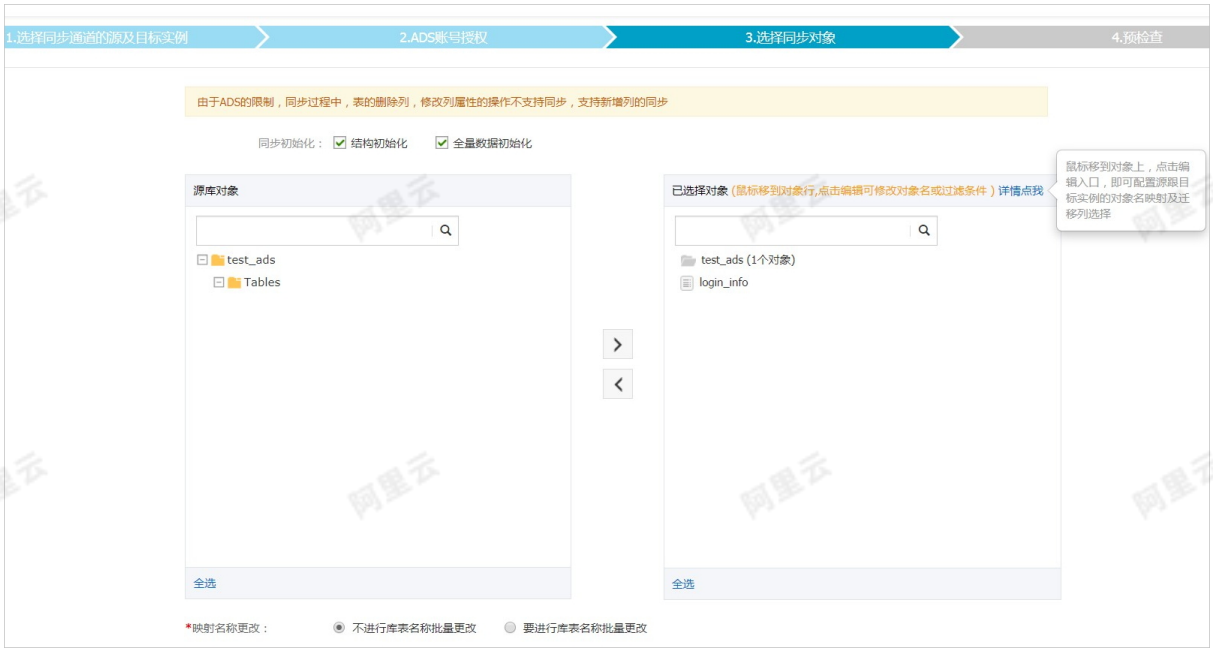
步骤二：配置同步链路

1. 登录DTS控制台。
2. 在**数据传输**页面，单击左侧导航栏中的**数据同步**。
3. 选择**地域**。
4. 在**同步作业**列表中，单击目标实例右侧的**配置同步链路**，在**选择同步通道的源及目标实例**页面进行参数配置，详细的参数配置如下表所示。



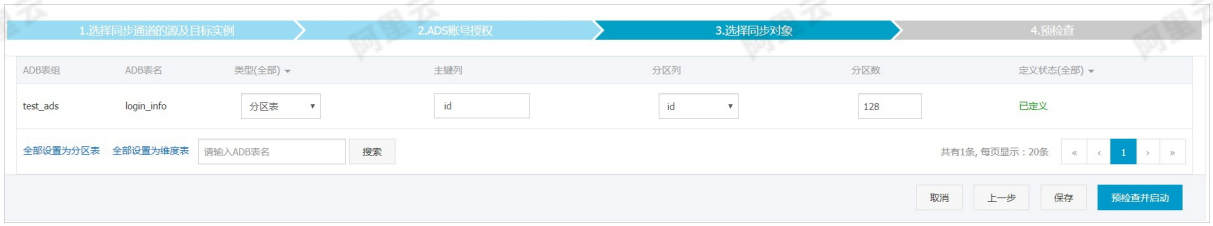
类别	配置项	说明
同步作业名称	-	可选项。
源实例信息	**实例类型**	**本例选择通过专线/VPN网关/智能网关接入的自建数据库** 。
	实例地区	本例为华南1（深圳）。
	对端专有网络	选择PolarDB实例所属的VPC ID。
	数据库类型	固定为MySQL，不可变更。
	IP地址	PolarDB实例的私网IP地址。 您可以在ECS或本地设备中，通过ping该PolarDB集群的主地址（私网）获取IP地址。 
	端口	PolarDB实例的监听端口，默认为3306。
	数据库账号	连接PolarDB数据库所使用的账号。
	数据库密码	连接PolarDB数据库所使用账号对应的密码。
目标实例信息	**实例类型**	ADS。
	实例地区	华南1（深圳）。
	数据库	分析型数据库MySQL版名称。

5. **完成上述参数配置后，单击授权白名单并进入下一步** 。
6. **在ADS账号授权** 页面，单击**下一步** 。
7. **进入选择同步对象** 页面，完成下面两个配置后，单击**下一步** 。



i. 勾选**结构初始化**和**全量数据初始化**。
ii. 在**源库对象**中把同步的表移动到右侧的**已选择对象**中。

8. 在**选择同步对象**页面，按照页面提示进行参数配置。



配置项	说明
类型	分区表或者维度表。
主键	支持复合主键，保证数据唯一。
分区列	请参见一级分区的规划和设计（2.0版）。
分区数	建议128。

9. 完成上述参数配置后，单击**预检查并启动**，进入**预检查**页面。
i. 如果预检查显示**失败**，需要根据提示并参见源库连接性检查进行排错处理。
ii. 预检查全部成功后，单击**关闭**同步任务正式开始。

步骤三：查看同步数据

1. 返回DTS控制台，在**同步列表**中的**同步概况**中查看同步**延时**和**速度**。
2. 进入分析型数据库MySQL版控制台，在**ads_sz_test**数据库中查看同步过来的数据表。

注意事项

- 配置同步链路过程中，如果目标表中列信息与源表不同，DTS支持字段映射功能，详细步骤请参见库表列映射。

- 如果需要同步的表数量较少且分析型数据库MySQL版表结构与源表差异较大，可以在分析型数据库MySQL版中提前创建表，配置同步链路时需要把步骤7中的中**结构初始化**选项去掉即可。

6.2. 数据集成（2.0版）

6.2.1. 使用数据集成迁移数据到AnalyticDB MySQL 2.0

数据集成是阿里集团对外提供的稳定高效、弹性伸缩的数据同步平台，致力于提供复杂网络环境下、丰富的异构数据源之间数据高速稳定的数据移动及同步能力。

支持的数据源类型

数据集成提供丰富的数据源支持，如下所示：

- 文本存储（FTP/SFTP/OSS/多媒体文件等）
- 数据库（RDS/DRDS/MySQL/PostgreSQL等）
- NoSQL（Memcache/Redis/MongoDB/HBase等）
- 大数据（MaxCompute/云原生数据仓库 AnalyticDB MySQL 版2.0/HDFS等）
- MPP数据库（HybridDB for MySQL等）

注意事项

通过数据集成导入数据到云原生数据仓库 AnalyticDB MySQL 版2.0集群中，推荐使用实时导入的方式。前提是在数据库中表一定要创建成实时表（普通表），通过实时导入效率高而且流程简单。

操作步骤

下面以RDS for SQLServer数据源为例，介绍如果如何通过数据集成把数据同步到云原生数据仓库 AnalyticDB MySQL 版2.0集群中，使用分析型数据库MySQL版进行数据分析。

1. 配置SQLServer数据源
2. 配置分析型数据库MySQL版数据源
3. 配置同步任务中的数据来源和去向

6.2.2. 配置SQLServer数据源

SQLServer数据源为您提供读取和写入SQLServer双向通道的功能，您可以通过向导模式和脚本模式配置同步任务。

前提条件

- 在配置SQLServer数据源之前，您需要在RDS for SQLServer端做好以下准备工作。
  i. 创建RDS for SQLServer实例，请参见快速创建RDS SQL Serer实例。
  ii. 在RDS for SQLServer中创建账号和数据库，并为需要同步数据的数据库所属账号分配相应的权限。详情，请参见创建数据库和账号。
  iii. 在创建好的SQLServer数据库中写入测试数据。
- 在DataWorks中创建了一个项目。详情，请参见创建一个项目。

背景信息

标准模式的工作空间支持数据源隔离功能，您可以分别添加并隔离开发环境和生产环境的数据源，以保护您的数据安全。详情请参见数据源开发和生产环境隔离。

操作步骤

1. 进入**数据源管理**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，鼠标悬浮至 图标，单击**工作空间配置**。
   iv. 在左侧导航栏，单击**数据源管理**，进入**数据源管理**页面。

   > **说明** 您也可以在数据集成页面进入**数据源管理**配置数据源，但此方式只支持生产环境的数据源。

2. 在**数据源管理**页面，单击右上角的**新增数据源**。
3. 在**新增数据源**对话框中，选择数据源类型为**SQLServer**。
4. 在**新增SQLServer数据源**对话框中，配置各项参数。
   i. 配置数据源的基本信息。
   SQLServer数据源包括**阿里云实例模式**和**连接串模式**两种类型。

- 以新增**阿里云实例模式**类型的数据源为例，配置数据源的基本信息。

参数	描述
数据源类型	当前选择的数据源类型为**阿里云实例模式**。
数据源名称	数据源名称必须以字母、数字、下划线（_）组合，且不能以数字和下划线（_）开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
适用环境	可以选择**开发**或**生产**环境。 **说明** 仅标准模式工作空间会显示该配置。
地域	选择购买RDS的地区。
RDS实例ID	您可以进入RDS控制台，查看RDS实例ID。
RDS实例主账号ID	实例购买者登录DataWorks控制台，鼠标悬停至右上角的用户头像，查看账号ID。
默认数据库名	此处配置的是该数据源对应的默认数据库名称。后续配置同步任务的说明如下： - 配置整库同步（包含实时和离线）或同步解决方案任务时，您可以选择相应RDS实例下所有具有权限的数据库。 - 配置离线同步任务，当您选择使用多个数据库时，则每个数据库均需要配置一个数据源。
用户名	登录数据库的用户名称。
密码	登录数据库的密码。密码中避免使用@符号。

说明
您需要先添加RDS白名单才能连接成功，详情请参见设置白名单。

- 以新增**连接串模式**类型的数据源为例，配置数据源的基本信息。

参数	描述
数据源类型	当前选择的数据源类型为**连接串模式**。
数据源名称	数据源名称必须以字母、数字、下划线（_）组合，且不能以数字和下划线（_）开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
适用环境	可以选择**开发**或**生产**环境。 **说明** 仅标准模式工作空间会显示该配置。
JDBC URL	JDBC连接信息，格式为 `jdbc:sqlserver://ServerIP:Port;DatabaseName=Database` 。此连接串中的Database为本数据源的默认数据库，但在配置同步任务时，您可以使用相应RDS实例下所有的数据库。
用户名	登录数据库的用户名。
密码	登录数据库的密码。

5. 选择**资源组连通性**类型为**数据集成**。
6. 在资源组列表，单击相应资源组后的**测试连通性**。
   数据同步时，一个任务只能使用一种资源组。您需要测试每个资源组的连通性，以保证同步任务使用的数据集成资源组能够与数据源连通，否则将无法正常执行数据同步任务。如果您需要同时测试多种资源组，请选中相应资源组后，单击**批量测试连通性**。详情请参见网络连通方案。

> **说明**
> - （推荐）资源组列表默认仅显示独享数据集成资源组，为确保数据同步的稳定性和性能要求，推荐使用独享数据集成资源组。
> - 如果您需要测试公共资源组或自定义资源组的连通性，请在资源组列表右下方，单击**更多选项**，在**警告**对话框单击**确定**，资源组列表会显示可供选择的公共资源组和自定义资源组。

7. 测试连通性通过后，单击**完成**。

6.2.3. 配置AnalyticDB MySQL 2.0数据源

准备工作

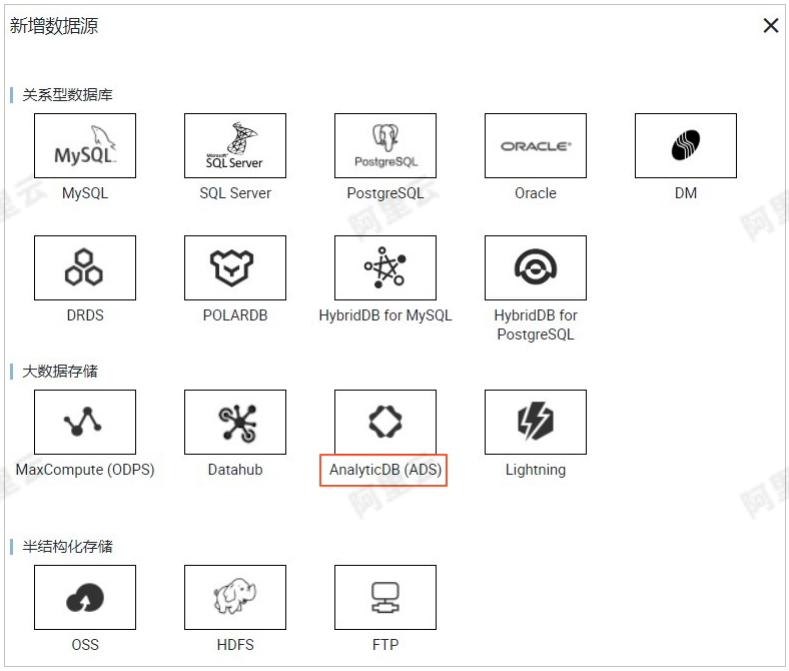
在分析型数据库MySQL版中创建数据库、表组和表。详情请参见快速入门-开通分析型数据库MySQL版服务、创建表组和创建表。

> **重要** 分析型数据库MySQL版中的表需要创建成实时表（普通表），通过实时导入效率高而且流程简单。

操作步骤

1. 登录DataWorks控制台，单击对应项目栏中的**进入数据集成**。
   本例项目名为SQLServer_DataWorks_ADB1。

2. 单击**新增数据源**，数据源选择**分析型数据库MySQL版**。



3. **在新增分析型数据库MySQL版数据源 页面，进行以下参数配置。**

详细的参数配置如下表所示。

配置项	说明
数据源名称	为数据源指定一个名字，便于后续管理。
数据源描述	添加数据源描述，该项为可选填项。
连接Url	分析型数据库MySQL版的连接地址和端口。
数据库	分析型数据库MySQL版的名字。
Access Id	分析型数据库MySQL版创建者的Access Id。
Access Key	Access Id对应的Access Key。

4. **完成上述参数配置后，单击测试连通性 进行连通性测试，测试通过后单击完成。**

6.2.4. 配置同步任务中的数据来源和去向

成功将数据导入分析型数据库MySQL版后，您就可以使用分析型数据库MySQL版进行数据分析。

1. 进入DataWorks控制台，单击对应项目操作栏中的**数据开发**。

2. 单击左侧菜单栏中的**数据开发**，右键单击**业务流程**新建一个流程。

3. 右键单击**步骤2**中新建的业务流程下的**数据集成**，选择**新建数据集成节点**>**数据同步**，输入同步**节点名称**。

4. 双击**步骤3**中创建的节点，配置数据同步任务的**数据来源**（Reader）、**数据去向**（Writer）、**字段映射**、**通道控制**信息。

数据来源（Reader）配置信息：

配置项	说明
数据源	选择**SQLServer**，系统将自动关联配置SQLServer数据源时设置的数据源名称。
表	选择SQLServer中的一张表进行数据同步。
数据过滤	同步数据的筛选条件，暂时不支持limit关键字过滤。 SQL语法随着所选择的数据源不同而不同。
切分键	选择SQLServer数据表中的主键作为切分键。

数据去向（Writer）配置信息：

配置项	说明
数据源	选择ADS，系统将自动关联配置AnalyticDB MySQL 2.0数据源时设置的数据源名称。

表	选择分析型数据库MySQL版中的一张表，将SQLServer中的数据同步至该表中。
导入模式	根据分析型数据库MySQL版中表的更新方式设置导入模式，本例为实时导入。

字段映射配置信息：

> 重要
> 注意列与列之间映射的字段类型是否有做数据兼容。

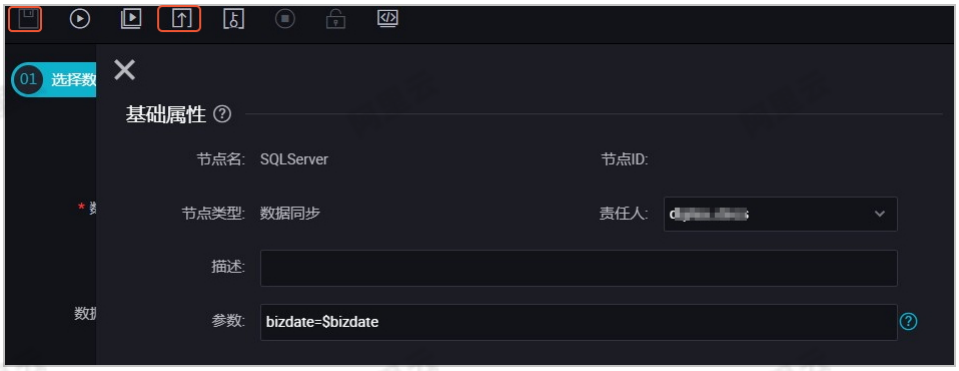
配置项	说明
同行映射	自动将同一行的数据设置映射关系。
自动排版	设置完映射关系后，字段排序展示。

通道控制配置信息：

配置项	说明
DMU	任务运行所需要的资源量。
作业并发数	配置的时候会结合读取端指定的切分建，将数据分成多个Task，多个Task同时运行，以达到提速的效果。
同步速率	设置同步速率可保护读取端数据库，以避免抽取速度过大，给读取端造成太大的压力。同步速率建议限流，结合源库的配置，请合理配置抽取速率。
错误记录数超过	NA

任务资源组	NA

5. 单击**保存**和**提交**，配置任务需要的其他信息。



6. 完成同步任务的配置后，先**保存**和**提交**节点，单击**运行**开始导入操作。

6.3. 使用kettle将本地数据导入AnalyticDB MySQL 2.0

本文以Excel为例，介绍如何通过kettle将本地Excel数据迁移到云原生数据仓库 AnalyticDB MySQL 版2.0集群。

背景信息

Kettle是一款非常受欢迎的开源ETL工具软件，主要用于数据整合、转换和迁移。Kettle除了支持各种关系型数据库，HBase MongoDB这样的NoSQL数据源外，它还支持Excel、Access这类小型的数据源。并且通过这些插件扩展，kettle可以支持各类数据源。

准备工作

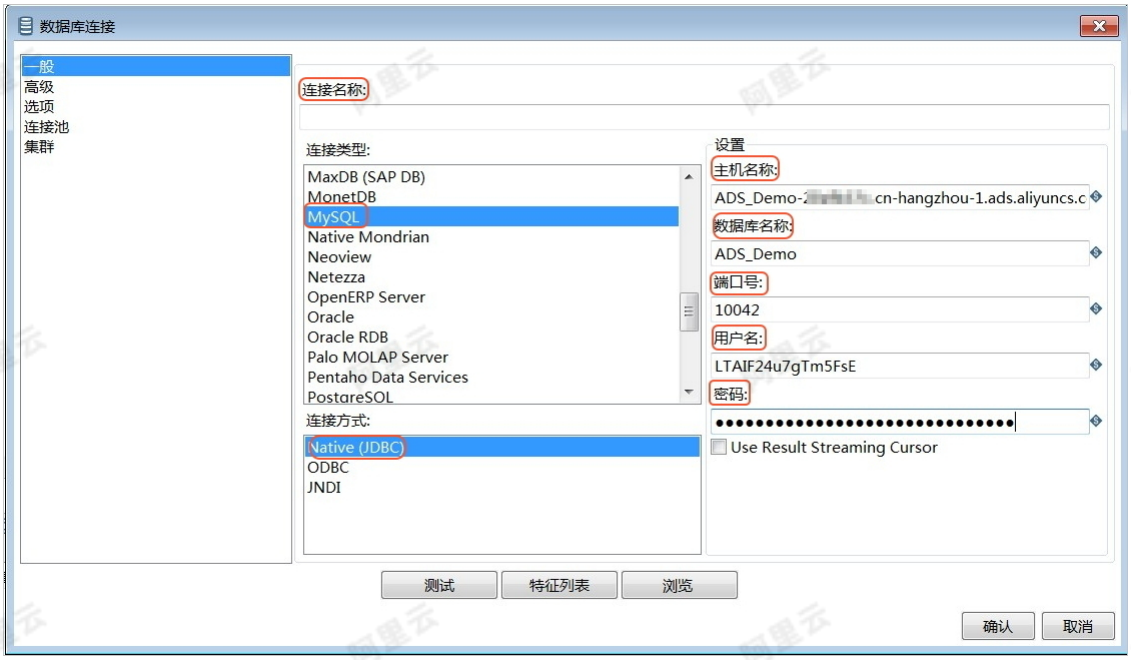
使用Kettle将本地Excel数据导入云原生数据仓库 AnalyticDB MySQL 版2.0集群之前，需要完成以下准备工作。

- 在本地主机中安装kettle。
- 在云原生数据仓库 AnalyticDB MySQL 版2.0集群中创建目标数据库、表组和表。关于如何创建云原生数据仓库 AnalyticDB MySQL 版2.0集群的数据库和表组，请参见快速入门-开通分析型数据库MySQL版服务、创建表组和创建表。

操作步骤

1. 在kettle中新建一个转换。
2. 在转换中新建一个数据库连接，详细的参数配置如下表所示。

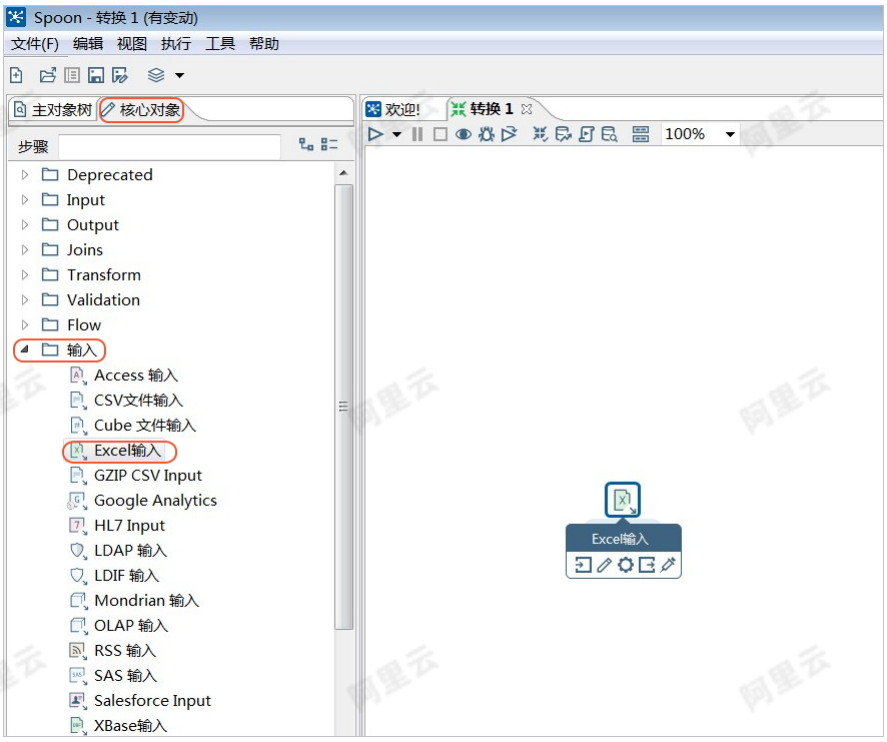
   配置参数时，不要勾选**Use Result Streaming Cursor**。



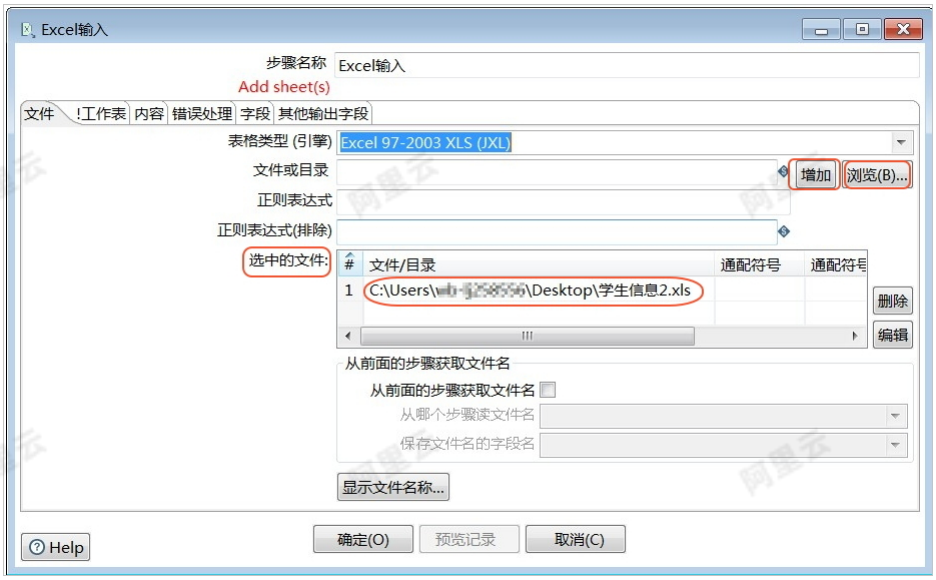
连接名称	数据连名
连接类型	选择MySQL
连接方式	选择Native（JDBC）

主机名	云原生数据仓库 AnalyticDB MySQL 版2.0集群的连接地址
数据库名称	云原生数据仓库 AnalyticDB MySQL 版2.0集群的名字
端口号	连接地址对应的端口号
用户名	AccessKey ID
密码	Access Key Secret

3. 完成上述参数配置后，单击**测试**测试连通性，测试通过后单击**确认**。
4. 在kettle左侧**核心对象**的**输入**中，找到**Excel输入**，并将其拖动入到工作区。

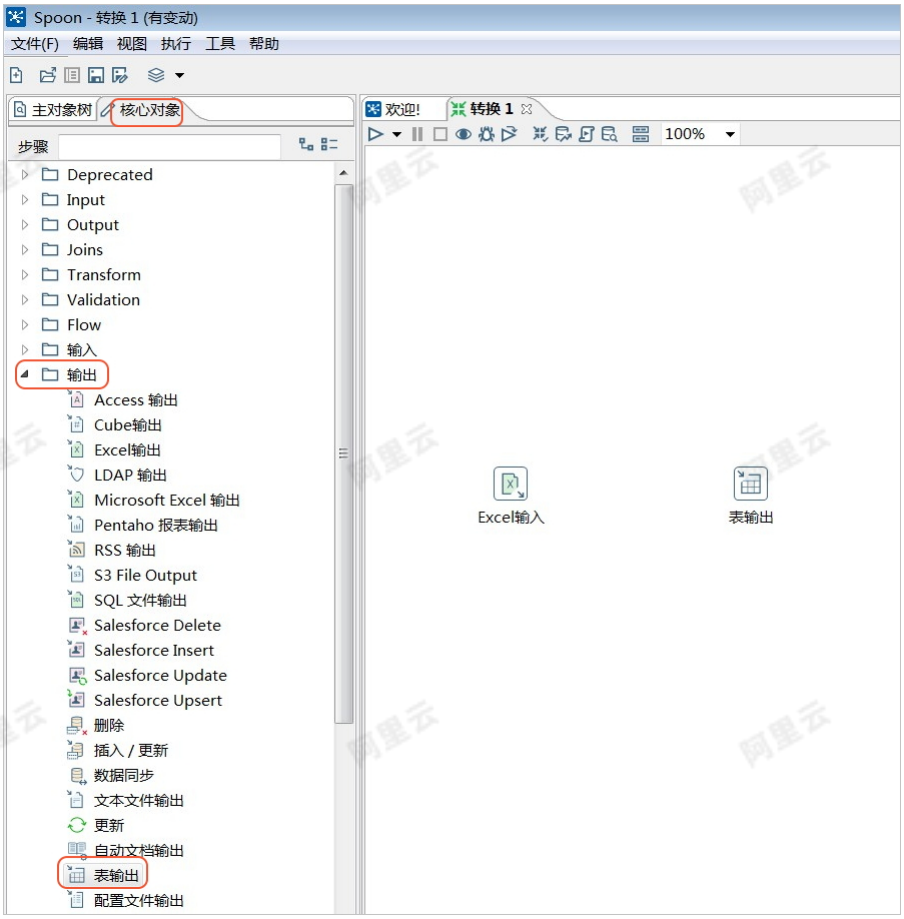


5. 工作区的**Excel输入**，在**Excel输入**对话框中，先单击**浏览**上传需要导入的Excel表格，再单击**增加**将其添加到**选中的文件**中。

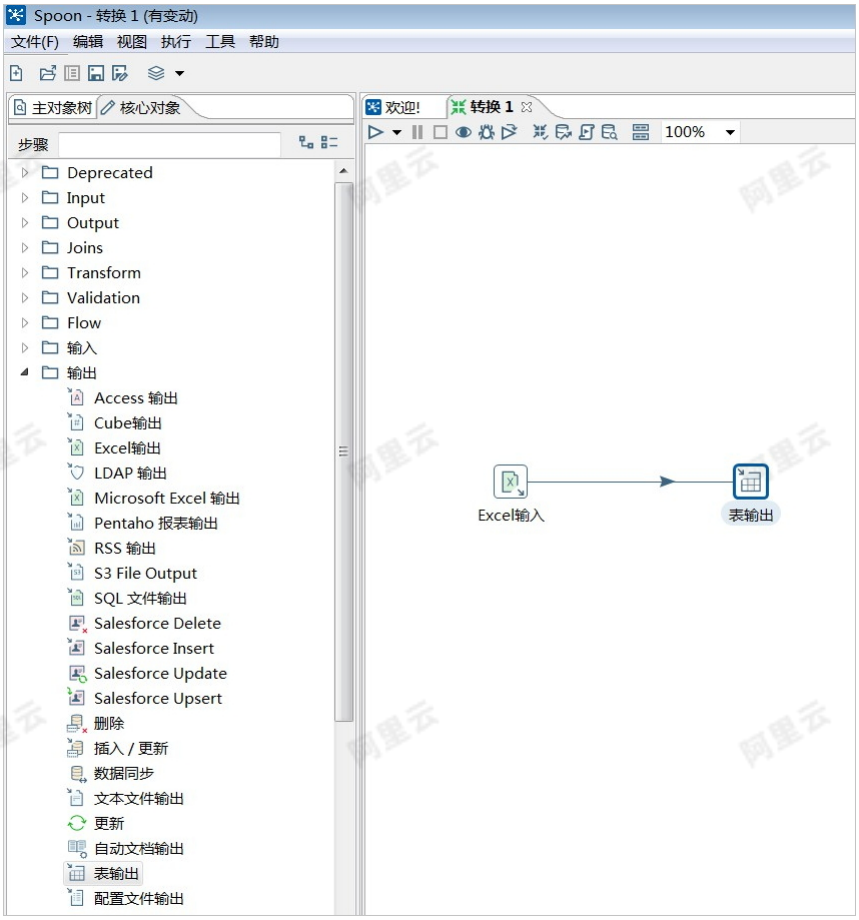


根据实际需要设置工作表、内容、字段等选项卡，单击预览记录查看输入的数据是否符合要求。

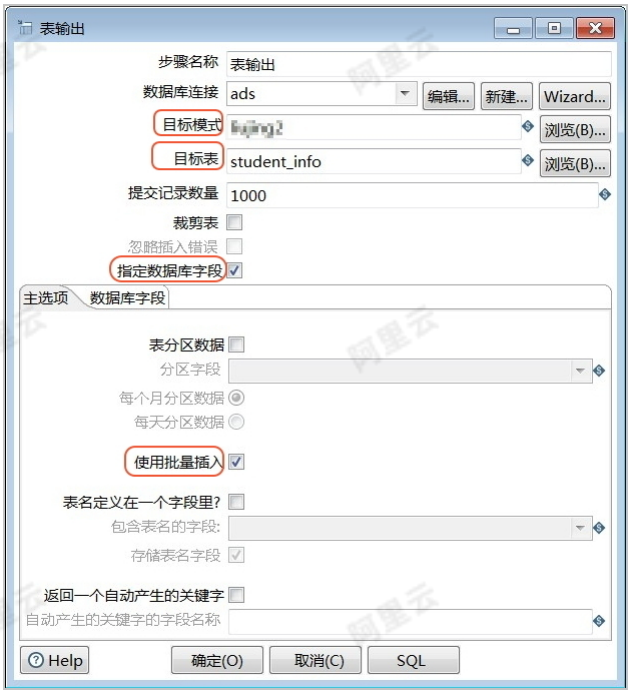
6. 在kettle左侧**核心对象**的**输出**中，找到**表输出**，并将其拖动入到工作区。



7. 新建一条**Excel输入**到**表输出**的连接线。



8. 双击**表输出**，在**表输出**对话框中进行参数配置。



- 目标模式：云原生数据仓库 AnalyticDB MySQL 版2.0集群的名字，不支持浏览选择。
- 目标表：云原生数据仓库 AnalyticDB MySQL 版2.0集群中的表，不支持浏览选择。
- 勾选指定数据库字段。
- 勾选使用批量插入。

在**表输出**的**数据库字段**选项卡中，单击**获取字段**和**输入字段映射**，映射Excel文件的列与云原生数据仓库 AnalyticDB MySQL 版2.0集群中表的列名间的映射关系。

9. 单击白色三角箭头运行转换，观察运行日志和运行状态。

   待本地数据成功导入云原生数据仓库 AnalyticDB MySQL 版2.0集群后，您就可以使用云原生数据仓库 AnalyticDB MySQL 版进行数据分析。

相关信息

- 使用DTS同步RDS MySQL数据
- 使用数据集成迁移数据至分析型数据库MySQL版
- 使用kettle将本地数据导入分析型数据库MySQL版
- 在程序中通过AnalyticDB MySQL版Client高效写入数据

6.4. 通过Logstash写入数据到AnalyticDB MySQL 2.0

本文介绍了如何通过Logstash写入数据。

背景信息

Logstash是一个开源的服务器端数据处理管道，起初用于将日志类数据写入ES中。随着开源社区的不断发展，现在Logstash可以同时从多个数据源获取数据，并对其进行转换，然后将其发送到用户想要的“存储端”。

以日志数据为例，可以通过开源logstash output插件logstash-output-jdbc将log数据导入分析型数据库MySQL版中进行进一步分析（分析型数据库MySQL版支持原生JDBC方式访问）。但经过测试发现，在日志量非常大的情况下，通过jdbc方式写入分析型数据库MySQL版的性能较低，并且非常消耗CPU的资源（因为jdbc是单条记录写入的方式）。为此，我们优化了一个基于jdbc的Logstash output plugin插件，请参见logstash-ouput-analyticdb，专门用于以聚合方式向分析型数据库MySQL版写入日志数据。

通过logstash-output-analyticdb将数据写入分析型数据库MySQL版时的性能，相较于logstash-output-jdbc有5倍的提升，并且对CPU的消耗也明显降低，本章节将为您介绍详细的操作步骤。

安装

Logstash的安装流程请参见Installing Logstash，这里主要介绍安装logstash-output-analyticdb的流程。

1. 进入logstash根目录：`cd logstash`。
2. 安装logstash-output-analyticdb：`bin/logstash-plugin install logstash-output-analyticdb`。
3. 在logstash目录下创建vendor/jar/jdbc目录：`mkdir -p vendor/jar/jdbc`。
4. 将jdbc jar拷贝到vendor/jar/jdbc中：

```
cd vendor/jar/jdbc
```

```
wget https://repo1.maven.org/maven2/mysql/mysql-connector-java/5.1.16/mysql-connector-java-5.1.16.jar
```

至此，logstash-output-analyticdb的安装步骤全都完成。

使用方式

和大多数logstash示例的使用方式一样，需要在config目录下创建一个logstash-analyticdb.conf（名字可以自行定义）的配置文件，conf文件的内容如下所示。

```
input
{
    stdin { }
}
output {
    analyticdb {
        driver_class => "com.mysql.jdbc.Driver"
        connection_string => "jdbc:mysql://HOSTNAME/DATABASE?user=USER&password=PASSWORD"
        statement => [ "INSERT INTO log (host, timestamp, message) VALUES(?, ?, ?)", "host", "@timestamp", "message" ]
    }
}
```

- connection_string：连接分析型数据库MySQL版的jdbc url。
- statement：insert SQL的声明数组。

上述配置文件的内容只是一个示例，具体配置文件的内容请根据实际场景来决定。与分析型数据库MySQL版相关的其他配置项请参见README。logstash的原使用配置项和规则，请参见logstash文档。

至此，配置任务已全部完成，接下来将启动任务。

启动任务

在logstash安装目录执行 `bin/logstash -f config/logstash-analyticdb.conf` 启动任务。

如果您在使用过程中遇到任何问题，请联系我们。联系方式：issue。

6.5. 在程序中通过AnalyticDB MySQL版Client高效写入数据到2.0集群

本文介绍了在程序中通过AnalyticDB for MySQLClient高效写入数据到2.0集群的方法。

背景信息

AnalyticDB for MySQLClient旨在为用户提供一个高效、简单地插入数据到AnalyticDB MySQL 2.0集群的方法。您只需要通过接口将数据提交给AnalyticDB for MySQLClient，便可以直接插入数据到AnalyticDB MySQL 2.0集群中。使用AnalyticDB for MySQL Client时，您无需关心分区聚合、连接池等问题，而且对数据实时写入也有更多的自主能力，不再强依赖DataHub等服务。

Maven repositories

通过Maven管理配置新的SDK版本，Maven的配置信息如下：

```
<dependency>
  <groupId>com.alibaba.cloud.analyticdb</groupId>
  <artifactId>adbclient</artifactId>
  <version>1.0.2</version>
</dependency>
```

接口列表

DatabaseConfig类

接口名	描述
setHost(String adbHost)	设置需要连接的AnalyticDB for MySQL2.0集群的主机或域名。
setPort(int port)	设置需要连接的AnalyticDB for MySQL2.0集群的端口。
setDatabase(String database)	需要连接的AnalyticDB for MySQL2.0集群的库名。
setUser(String username)	设置连接AnalyticDB for MySQL2.0集群的用户名，请填AccessKey ID。 如何获取AccessKey ID，请参见账号与权限管理。
setPassword(String pwd)	设置连接AnalyticDB for MySQL2.0集群的密码，请填AccessKey Secret。
setTable(List<String> table)	需要写入的表名List，建议小写。
setColumns(String tableName, List<String> columnList)	需要插入表的字段名，若是全字段插入，则使用 `columnList.add("*")` 即可。table列表中的所有表都需要设置小写的字段名，否则无法通过检查。
setIngnoreInsertError(boolean isIngnoreInsertError)	针对配置的所有表，设置是否忽略插入时遇到的error，默认为false。
setInsertIgnore(boolean insertIgnore)	针对配置的所有表，请根据业务场景判断是否使用 `insert ignore into` 语句。
setEmptyAsNull(boolean emptyAsNull)	针对配置的所有表，将empty数据设置为Null，默认为true。
setParallelNumber(int parallelNumber)	针对配置的所有表，设置写入AnalyticDB for MySQL2.0集群时的并发线程数，默认为4。
setLogger(Logger logger)	设置client中使用的logger对象，此处使用 `slf4j.Logger` 。
setRetryTimes(int retryTimes)	设置提交时写入AnalyticDB for MySQL2.0集群出现异常时重试的次数，默认为0。
setRetryIntervalTime(long retryIntervalTime)	设置重试间隔的时间，单位是ms，默认为0。
setCommitSize(long commitSize)	设置自动提交的SQL长度（单位Byte），默认为32 KB，一般不建议设置。
setInsertWithColumnName(boolean insertWithColumnName)	设置在拼接INSERT SQL时，是否需要带字段名，默认为true。例如，`insert into tableName (col1,col2) values ('value1','value2');` 如果为false，语句则类似 `insert into tableName values ('value1','value2');` ，此种情况下如果在表中加字段可能会导致写入失败。此设置只有在column列设置为*的情况下才会生效。

Row类

接口名	描述
setColumn(int index, Object value)	设置Row字段列表的值，必须确认字段的顺序。此种方式下，Row实例不可复用，每条数据必须使用单独的Row实例。
setColumnValues(List<Object> values)	直接将List格式数据行写入Row中。
updateColumn(int index, Object value)	更新Row字段列表的值。此方法中，Row实例可以复用，只需更新Row实例中的数据即可。

AdbClient类

接口名	描述
addRow(String tableName, Row row) / addRows(String tableName, List<Row> rows)	插入对应表的Row格式化的数据，即一条记录。数据会按照分区聚合的形式在SDK的内存中缓存，等待提交。如果SQL长度已满，则会在addRow或者addMap的时候做一次自动提交，然后将最新的数据新增进来。如果在自动提交时失败，调用方需要处理异常，并且会在异常中得到失败的数据列表。
addMap(String tableName, Map<String, String> dataMap) / addMaps(String tableName, List<Map<String, String>> dataMaps)	对应于addRow，支持map格式数据的写入。如果SQL长度已满，则会在addRow或者addMap时做一次自动提交，然后将最新的数据新增进来。如果在自动提交时失败，调用方需要处理此异常，并且会在异常中得到失败的数据列表。
addStrictMap(String table, Map<String, String> dataMap) / addStrictMaps(String tableName, List<Map<String, String>> dataMaps)	作用和addMap或addMaps类似，不同之处是传入的Map数据的key必须是表的字段。如果不是，会重新获取一次表结构信息，如果还没有改key的字段，则直接报错。其他功能与addMap或addMaps一致。如果没有动态表结构变更，不建议使用改接口，性能会有所损耗。
commit()	将缓存的数据进行提交，写入AnalyticDB for MySQL2.0集群中。若提交失败，会把执行错误的语句放在异常中抛出，调用方需要对此异常进行处理。
TableInfo getTableInfo(String tableName)	获取对应表的结构信息。
List<ColumnInfo> getColumnInfo(String tableName)	获取对应表的字段列表信息，字段类是ColumnInfo，可以通过 `columnInfo.isNullable()` 获取该字段是否能为null。
Connection getConnection() throws SQLException	从客户端连接池获取 `mysql Connection` 连接信息，调用方可以使用获得的Connection做非插入操作，使用方式和JDBC的连接使用方式一致。 重要 使用结束后需要释放掉相应的资源（如ResultSet、Statement、Connection）。

ColumnInfo类

接口名	描述
boolean isNullable()	判断该字段是否能为null。

AdbClientException错误码

错误码名	错误码值	描述
SQL_LENGTH_LIMIT	100	SQL长度超过限制的长度，默认为32KB。
COMMIT_ERROR_DATA_LIST	101	提交中某些数据出现异常，会返回异常的数据，通过 `e.getErrData()` 即可获得异常数据List<String>。此错误码在 `addMap(s)` 、 `addRow(s)` 和提交操作的时候都可能会发生，需要单独处理此错误码的异常。
COMMIT_ERROR_OTHER	102	Commit提交中的其他异常。
ADD_DATA_ERROR	103	新增数据过程中出现的异常。
CREATE_CONNECTION_ERROR	104	创建连接出现异常。
CLOSE_CONNECTION_ERROR	105	关闭连接出现异常。
CONFIG_ERROR	106	配置DatabaseConfig出现配置错误。
STOP_ERROR	107	停止实例时的报错。
OTHER	999	默认异常错误码。

示例代码

```
public class AdbClientUsage {
        public void demo() {
            DatabaseConfig databaseConfig = new DatabaseConfig();
            // AnalyticDB MySQL 2.0集群的主机名或者URL
            databaseConfig.setHost("100.100.100.100");
            // AnalyticDB MySQL 2.0集群的连接端口号
            databaseConfig.setPort(8888);
            // AnalyticDB MySQL 2.0集群的所属账号的AccessKey ID
            databaseConfig.setUser("your db username");
            // AnalyticDB MySQL 2.0集群的所属账号的AccessKey Secret
            databaseConfig.setPassword("your db password");
            databaseConfig.setDatabase("your db name");
            // 设置需要写入的表名列表
            List<String> tables = new ArrayList<String>();
            tables.add("your table name");
            tables.add("your table name 2");
            // 一旦new Client实例之后，表配置是不可修改的
            databaseConfig.setTable(tables);

            // 设置需要写入的表字段
            List<String> columns = new ArrayList<String>();
            columns.add("column1");
            columns.add("column2");
            // 如果是所有字段，字段列表使用columns.add("*")即可
            databaseConfig.setColumns("your table name", columns);
            databaseConfig.setColumns("your table name 2", Collections.singletonList("*"));

            // 如果出现插入失败，是否跳过
            //忽略插入失败可能导致数据丢失
            databaseConfig.setIgnoreInsertError(false);
              //如果该列的值为空，则直接设置为null即可。
            databaseConfig.setEmptyAsNull(true);
            // 使用insert ignore into方式进行插入
            databaseConfig.setInsertIgnore(true);
            // 提交时，写入AnalyticDB MySQL 2.0集群出现异常时重试的3次
            databaseConfig.setRetryTimes(3);
            // 重试间隔的时间为1s，单位是ms
            databaseConfig.setRetryIntervalTime(1000);
            // 初始化AdbClient，初始化实例之后，databaseConfig的配置信息无法再进行修改
            AdbClient adbClient = new AdbClient(databaseConfig);

            // 数据需要攒批，分多次添加，再提交
            for (int i = 0; i < 10; i++) {
                // Add row(s) to buffer. One instance for one record
                Row row = new Row(columns.size());
                // Set column
                // the column index must be same as the sequence of columns
                // the column value can be any type, internally it will be formatted according to column type
                row.setColumn(0, i); // Number value
                row.setColumn(1, "string value"); // String value
                // 如果SQL长度已满，则会在addRow或者addMap的时进行一次自动提交
                // 如果提交失败，则返回AdbClientException异常，错误码为COMMIT_ERROR_DATA_LIST
                adbClient.addRow("your table name", row);
            }
            Row row = new Row();
            row.setColumn(0, 10); // Number value
            row.setColumn(1, "2018-01-01 08:00:00"); //参数2的数据类型为 Date、Timestamp或者Time value
            adbClient.addRow("your table name 2", row);
            // Update column. Row实例可复用
            row.updateColumn(0, 11);
            row.updateColumn(1, "2018-01-02 08:00:00");
            adbClient.addRow("your table name 2", row);

            // 将map加入缓存中
            Map<String, String> rowMap = new HashMap<String, String>();
            rowMap.put("column1", "124");
            rowMap.put("column2", "string value");
            // 数据需要攒批，最好多次添加之后再进行提交
            adbClient.addMap("your table name", rowMap);

            //将缓存中的数据提交到AnalyticDB MySQL 2.0集群
            //数据成功提交到AnalyticDB MySQL 2.0集群之后，缓存中的数据会被清除
            try {
                adbClient.commit();
            } catch (Exception e) {
            } finally {
                adbClient.stop();
            }
        }
    }
```

注意事项

- AnalyticDB for MySQLClient本身依赖 `druid(1.1.10)` 、 `mysql-connector-java(5.1.45)` 、 `commons-lang3(3.4)` 、 `slf4j-api(1.7.25)` 和 `slf4j-log4j12(1.7.25)` ，如果在使用过程中，出现版本冲突，请检查这几个包的版本并解决冲突。

- AnalyticDB for MySQLClient SDK是非线程安全的，如果多线程调用时，需要每个线程维护自己的Client对象。

  > 说明 强烈不建议多线程共用SDK实例，除了线程安全问题外，还容易使Client成为写入性能的瓶颈。

- 数据必须在调用commit成功后才算成功写入AnalyticDB for MySQL。
- 针对Client抛出的异常，调用方要根据错误码的意义自行判断如何处理。如果是数据写入有问题，可以重复提交或者记录下有问题的数据后跳过。
- 很多时候写入线程并不是越多越好，因为业务程序会涉及到攒数据的场景，所以对内存的消耗比较明显，业务调用方一定要多关注应用程序的GC情况。
- 数据攒批数量不要太小，如果太小，分区聚合写意义就不大了。最好通过 `消息数*消息长度≈分区数*32KB` 大概得出攒批的消息数。
- DatabaseConfig配置在 `new AdbClient` 成功之后是无法进行修改，所有配置项必须在Client对象初始化之前完成配置。

6.6. 通过adbuploader将本地数据导入AnalyticDB MySQL 2.0

本文介绍如何将本地的数据文件导入云原生数据仓库 AnalyticDB MySQL 版2.0集群。

准备工作

- 安装Java环境。
- 数据是格式化的，目前只支持csv格式和特定分隔符的text文件。
- csv数据文件的编码格式转换为utf8（目前只支持utf8）。
- 针对特别大的数据文件，为提高导入性能，请将大数据文件切分成多个文件进行并发。
- 已创建数据库、创建表组、创建表，详细操作请参见快速入门。

操作步骤

1. 下载adbuploader。
2. 执行以下命令将本地数据导入分析型数据库MySQL版。

```
java -server -Xmx1g -Xms1g -Xmn256m -jar adbuploader-1.0.0.jar [options]
```

options用法

```
usage: java adbupload [option]
 -c,--conn <arg>        the connection info of your analyticdb. e.g.
                        localhost:10001
 -C,--col <arg>         the column list of table which want to insert.
                        split by ','. e.g. "col1,col2".
 -d,--database <arg>    the database of your analyticdb
 -F,--format <arg>      the format type of message. e.g. csv, text
 -h,--help              help info
 -H,--header            skip the header of files
 -n,--name <arg>        the name of source. dir or filepath
 -num <arg>             the concurrence num of writer. default is 4
 -p,--password <arg>    the AccessKeySecret of your aliyun account
 -readnum <arg>         the concurrence num of reading source if the
                        source is dir. default is 1
 -S,--source <arg>      dir or file, default file
 -s,--separator <arg>   the separator of message if the format is text.
                        default ','
 -t,--table <arg>       the table which want to insert
 -u,--username <arg>    the AccessKeyId of your aliyun account
```

参数说明

- `-c,--conn <arg>`：分析型数据库MySQL版的连接信息，通过分析型数据库MySQL版控制台查看连接信息。
- `-u,--username <arg>`：阿里云账号的AccessKeyId，用于连接分析型数据库MySQL版。
- `-p,--password <arg>`：阿里云账号的AccessKeySecret，用于连接分析型数据库MySQL版。
- `-d,--database <arg>`：在分析型数据库MySQL版上创建的数据库库名。
- `-t,--table <arg>`：需要导入的表名。
- `-C,--col <arg>`：需要写入的表的字段列表，字段的顺序需要与数据文件中记录拆分的顺序一致。
- `-S,--source <arg>`：本地数据文件或目录的类型，有dir（目录）或file（文件）两种，默认file，可不填。
- `-F,--format <arg>`：数据文件的记录格式，目前仅支持csv和text。
- `-s,--separator <arg>`：设置分隔符，如果为text文本文件，可以设置多字符的分隔符（比如##）。默认为”,”，可不填。
- `-n,--name <arg>`：数据文件（目录）的名字或路径，请用绝对路径。
- `-readnum <arg>`：当source为dir时，设置读取文件的并发数，不要超过文件数，默认为1，可不填。
- `-num <arg>`：写入分析型数据库MySQL版的并发数（并发数不超过CPU核数）。默认为4，可不填。
- `-H`：跳过文件头部的第一行（csv称为表头）。
- `-h,--help`：帮助信息。

示例

将/user/lilei/test文件（csv格式）导入到adb test的orders表的 `o_orderkey,o_custkey,o_orderstatus,o_totalprice,o_orderdate,o_orderpriority,o_clerk,o_shippriority,o_comment` 字段中。

```
java -server -Xmx4g -Xms4g -Xmn1g -jar adbuploader-1.0.0.jar -S file -n /user/lilei/test -F csv -c adb-test-56cbade4.cn-shanghai-
1.ads.aliyuncs.com:10033 -u ***** -p ***** -d adb_test -t orders -C
"o_orderkey,o_custkey,o_orderstatus,o_totalprice,o_orderdate,o_orderpriority,o_clerk,o_shippriority,o_comment" -num 32
```

6.7. 导出数据（2.0版）

6.7.1. 导出数据到MaxCompute

请参考SQL DUMP TO MAXCOMPUTE

6.7.2. 导出数据到OSS

详细请参看SQL DUMP TO OSS

6.8. DataX（2.0版）

6.8.1. DataX简介

DataX是阿里巴巴集团内被广泛使用的离线数据同步工具/平台，实现包括MySQL、Oracle、SQLServer、PostgreSQL、HDFS、Hive、AnalyticDB、HBase、TableStore、MaxCompute(ODPS)、DRDS等各种异构数据源之间高效的数据同步功能，在阿里巴巴集团内已经稳定运行了5年之久。

DataX本身作为数据同步框架，将不同数据源的同步抽象为从源头数据源读取数据的Reader插件，以及向目标端写入数据的Writer插件，理论上DataX框架可以支持任意数据源类型的数据同步工作。同时DataX插件体系作为一套生态系统， 每接入一套新数据源即可实现新加入的数据源现有的数据源互通。

RDBMS关系型数据库	MySQL	√	√	读、写
	Oracle	√	√	读、写
	SQLServer	√	√	读、写
	PostgreSQL	√	√	读、写
	DRDS	√	√	读、写
	通用RDBMS(支持所有关系型数据库)	√	√	读、写
阿里云数仓数据存储	MaxCompute	√	√	读、写
	AnalyticDB	-	√	写
	OSS	√	√	读、写
	OCS	√	√	读、写
NoSQL数据库	Table Store	√	√	读、写
	HBase0.94	√	√	读、写
	HBase1.1	√	√	读、写
	MongoDB	√	√	读、写
	Hive	√	√	读、写
无结构化存储	TxtFile	√	√	读、写
	FTP	√	√	读、写
	HDFS	√	√	读、写
	Elaticsearch	-	√	写

通过DataX同步数据至AnalyticDB

- 在底层实现上，结构化数据源和NoSQL数据源通过JDBC远程连接对应的数据源，并执行相应的SQL语句将数据从源库中select出来，然后在DataX插件内部做数据类型转换，最后通过adswriter将数据导入AnalyticDB目标表中。
- 如果是非结构化数据源，DataX通过对应的Reader获取数据文件，并将其转换为DataX传输协议传递给ADSWriter。此外，读取非结构化数据源时DataX还支持以下功能：
  - 支持txt格式的文件。
  - 支持类CSV格式文件。
  - 支持多种类型的数据读取(使用String表示)，支持列裁剪，支持列常量。
  - 支持递归读取、支持文件名过滤
  - 支持文本压缩，现有压缩格式为zip、gzip、bzip2。
    > *一个压缩包不允许多文件打包压缩。*
  - 支持并发读取多个object或者File。

在实际数据迁移时，可以根据情况修改参数来变更非结构化数据源Reader插件的通用属性配置：

- fieldDelimiter
  - 描述：读取字段分隔符
  - 必选：是
  - 默认值：,
- compress
  - 描述：文本压缩类型，默认不填写表示没有压缩。支持压缩类型为zip、gzip、bzip2。
  - 必选：否
  - 默认值：没有压缩
- encoding
  - 描述：读取文件的编码配置

- 必选：否
- 默认值：utf-8

- skipHeader
  - 描述：类CSV格式文件可能存在表头为标题情况，需要跳过，默认不跳过。
  - 必选：否
  - 默认值：false

- nullFormat
  - 描述：文本文件中无法使用标准字符串定义null(空指针)，DataX提供nullFormat定义哪些字符串可以表示为null。例如，如果配置：`nullFormat:"\N"`，则如果源头数据是"\N"，DataX视作null字段。
  - 必选：否
  - 默认值：\N

6.8.2. 通过DataX同步MySQL数据至AnalyticDB MySQL 2.0

DataX依赖以下环境，安装DataX之前请先主备好依赖环境：

- Linux/Windows/macOS JDK（1.8以上，推荐1.8）
- Python（推荐Python2.6.X）

在DataWorks 页面，单击**Quick Start**下的**DataX下载地址**，下载并安装DataX。

配置同步JSON文件

JSON同步配置模板可以使用命令：`python datax.py -r {YOUR_READER} -w{YOUR_WRITER}` 生成。例如，执行命令从MySQL同步数据到AnalyticDB中：

```
{
"job": {
"setting": {
"speed": {
"channel": 1
}
},
"content": [
{
 "reader": {
          "name":"mysqlreader",
          "parameter": {
            "username": "账号",
            "password": "密码",
            "column": [
              "*"
            ],
            "connection": [
              {
                "table": [
                  "要同步的表"
                ],
                "jdbcUrl": [
                  "填写jdbcurl（比如：jdbc:mysql://127.XX.XX.1:3306/test）"
                ]
              }
            ]
          }
        },
"writer": {
"name": "adswriter",
"parameter": {
            "url":"填写url（比如：127.XX.XX.1:10249）",
            "username": "此处填写：你自己的acessId",
            "password": "此处填写：你自己的acessKey",
            "schema": "此处填写：要同步的ADS库名，比如ads_demo",
            "table": "此处填写：你要同步的ADS表名",
            "column": [
              "*"
            ],
            "partition":"此处填写目标表的分区字段",
            "overwrite": "true",
            "writeMode": "insert",
            "lifeCycle":2
}
}
}
]
  }
}
```

参数说明

url	ADS连接信息，格式为 `ip:port` 。	为"ip:port是
schema	ADS的schema名称	是
username	AnalyticDB拥有者的AccessKeyID	是

password	AccessKeyID对应的AccessKeySecret	
table	目标表的表名称	是
partition	目标表的分区名称，当目标表为普通表时，需要指定普通表的分区。	否
writeMode	insert，实时导入模式。	是
column	目的表字段列表，可以为[“*”]，或者具体的字段列表，例如[“a”, “b”,”c”]。	是
batchSize	AnalyticDB 批量insert数据的条数。	是
bufferSize	DataX数据收集缓冲区大小，缓冲区的目的是攒一个较大的Buffer，源头的数据首先进入到此Buffer中进行排序，排序完成后再提交到AnalyticDB。排序是根据AnalyticDB的分区列模式进行的，排序的目的是数据顺序对AnalyticDB服务端更友好（出于性能考虑）。 BufferSize缓冲区中的数据会经过batchSize批量提交到ADB中，一般如果要设置bufferSize，设置bufferSize为batchSize数量的多倍。	当writeMode为insert时，该值才会生效。

DataX与AnalyticDB内部类型转换对应表

Long	int、tinyint、 smallint、 int、bigint
Double	float、double
String	varchar
Date	date、timestamp
Boolean	bool
Bytes	无

从各数据源向AnalyticDB迁移数据

从MySQL迁移数据至AnalyticDB

以下示例将阿里云RDS for MySQL数据迁移至AnalyticDB，JSON文件名为rds2ads.json，配置如下所示：

```
{
"job": {
"content": [
{
 "reader": {
          "name":"mysqlreader",
          "parameter": {
          "column": [
              "id",
              "name",
              "age",
              "create_time",
              "update_time"
            ],
            "connection": [
              {
                "jdbcUrl": [
                  "填写jdbcurl（比如：jdbc:mysql://127.XX.XX.1:3306/test）"
                ],
                "table": ["t_user"]

              }
            ]
            "username": "账号",
            "password": "密码",

          }
        },
"writer": {
"name": "adswriter",
"parameter": {
            "url":"填写url（例如：127.XX.XX.1:10249）",
            "username": "此处填写：你自己的acessId",
            "password": "此处填写：你自己的acessKey",
            "schema": "此处填写：要同步的ADS库名，比如ads_demo",
            "table": "此处填写：你要同步的ADS表名",
            "column": [
              "*"
            ],
            "partition":"此处填写目标表的分区字段",
            "overwrite": "true",
            "writeMode": "insert",
            "lifeCycle":2
}
}
}
],
"setting": {
    "speed": {
        "channel": "1"
    }
}
  }
}
```

上述JSON配置好后，启动datax迁移数据：`#python datax.py./rds2ads.json`

从MaxCompute迁移数据至AnalyticDB

从MaxCompute向AnalyticDB迁移数据时，需要先进行以下授权：

由于AnalyticDB需要获取odps待同步表的结构和数据信息，因此**AnalyticDB所属账号需要有待同步表的describe和select权限** 。

生成如下配置模板：

6.9. AnalyticDB MySQL 2.0集群间的数据库迁移

以下示例将向您介绍如何把一个AnalyticDB MySQL 2.0集群中的数据迁移至另外一个AnalyticDB MySQL 2.0集群。

准备工作

- 开通所属地域相同的OSS、AnalyticDB、DataWorks服务。
  为实现整库迁移，您需要在同一地域创建两个AnalyticDB MySQL集群。
- 在OSS中新建文件夹，将用于存储AnalyticDB数据的所有文件上传至文件夹中。如何新建文件夹，请参见新建文件夹。
  本示例创建的Bucket为iujing-test-analyticdb，数据文件为shipping.csv，存于Bucket的table目录下。

实施步骤

1. 将AnalyticDB数据DUMP到OSS中
2. 配置OSS数据源
3. 配置AnalyticDB数据源
4. 通过数据集成将OSS中的数据导入AnalyticDB中

通过上述四个步骤即可将ads_database数据库中的数据成功迁移至另外一个ads_sz_test数据库中。

将AnalyticDB数据DUMP到OSS中

您可以通过MySQL命令行、AnalyticDB支持的客户端DBeaver或者DMS For AnalyticDB执行以下SQL将AnalyticDB数据导入OSS中的指定文件。

```
/*+ dump-col-del=[,],
    dump-row-del=[\n],
    dump-oss-accesskey-id=ACCESS_KEY_ID,
    dump-oss-accesskey-secret=ACCESS_KEY_SECRET,
    engine=MPP,
    return-dump-record-count=TRUE,
    dump-header=[DUMP DATA OVERWRITE INTO 'oss://oss-cn-shenzhen-internal.aliyuncs.com/liujing-test-analyticdb/table/shipping.csv']
*/
SELECT *
FROM shipping;
```

上述示例是将AnalyticDB中的shipping表数据导入OSS中，OSS Bucket为analyticdb-bucket，object为table，存储数据的文件为shipping.csv。

若有多张表需要导入，请参照上述示例，修改相应参数，依次导入数据。

更多dump信息，请参见DUMP TO OSS。

配置OSS数据源

操作步骤请参见配置OSS数据源。

配置AnalyticDB数据源

操作步骤请参见配置AnalyticDB数据源。

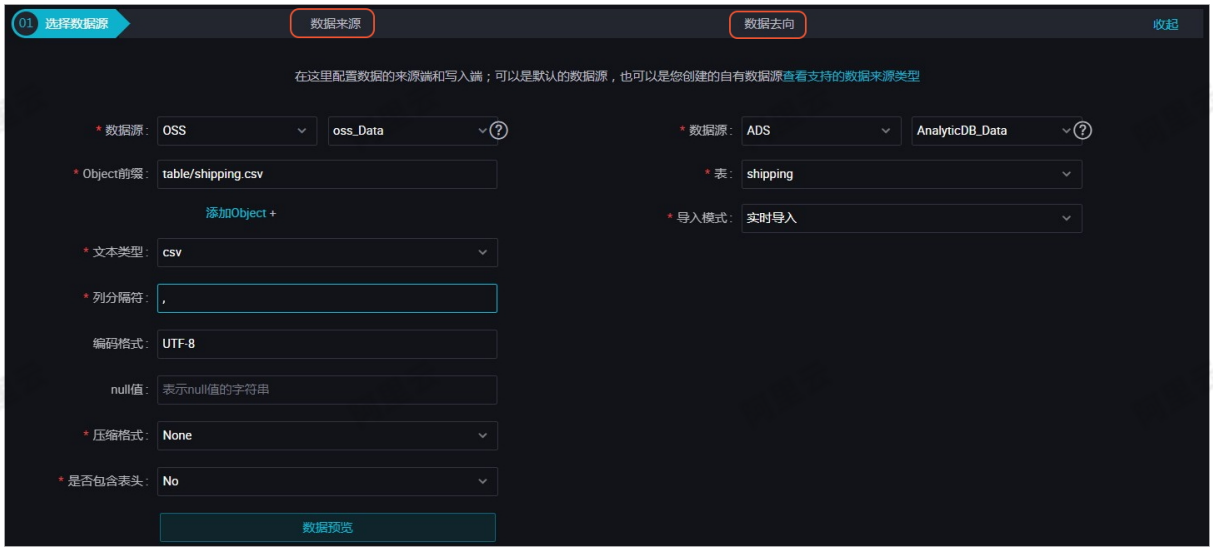
> *请根据实际情况修改相应参数。*

通过数据集成将OSS中的数据导入AnalyticDB中

1. 进入DataWorks控制台，单击对应项目操作栏中的**数据开发**。
2. 单击左侧菜单栏中的**数据开发**，右键单击**业务流程**新建一个流程。

3. 右键单击**步骤2**中新建的业务流程下的**数据集成**，选择**新建数据集成节点**>**数据同步**，输入同步**节点名称**名称。

4. 双击**步骤3**中创建的节点，配置数据同步任务的**数据来源**（Reader）、**数据去向**（Writer）、**字段映射**、**通道控制**信息。



数据来源（Reader）配置信息：

数据源	选择**OSS**，系统将自动关联配置OSS数据源时设置的数据源名称。
Object前缀	OSS的Object信息，此处可以支持填写多个Object。 例如，Bucket中有table文件夹，文件夹中有shipping.csv文件，则Object直接填table/shipping.csv。 当指定单个OSS Object时，数据来源暂时只能使用单线程进行数据抽取。后期将考虑在非压缩文件情况下针对单个Object可以进行多线程并发读取。 当指定多个OSS Object时，数据来源暂支持使用多线程进行数据抽取。线程并发数通过通道数指定。 当指定通配符时，OSS Reader尝试遍历出多个Object信息。详情请参见OSS概述。
文本类型	文件的类型，此处为.csv。
编码格式	读取文件的编码配置。
null值	文本文件中无法使用标准字符串定义null（空指针），数据同步系统提供nullFormat定义哪些字符串可以表示为null。。
压缩格式	文本压缩类型，默认不填写（即不压缩）。支持压缩类型为gzip、bzip2和zip。
是否包含表头	类CSV格式文件可能存在表头为标题情况，需要跳过，设置为No。

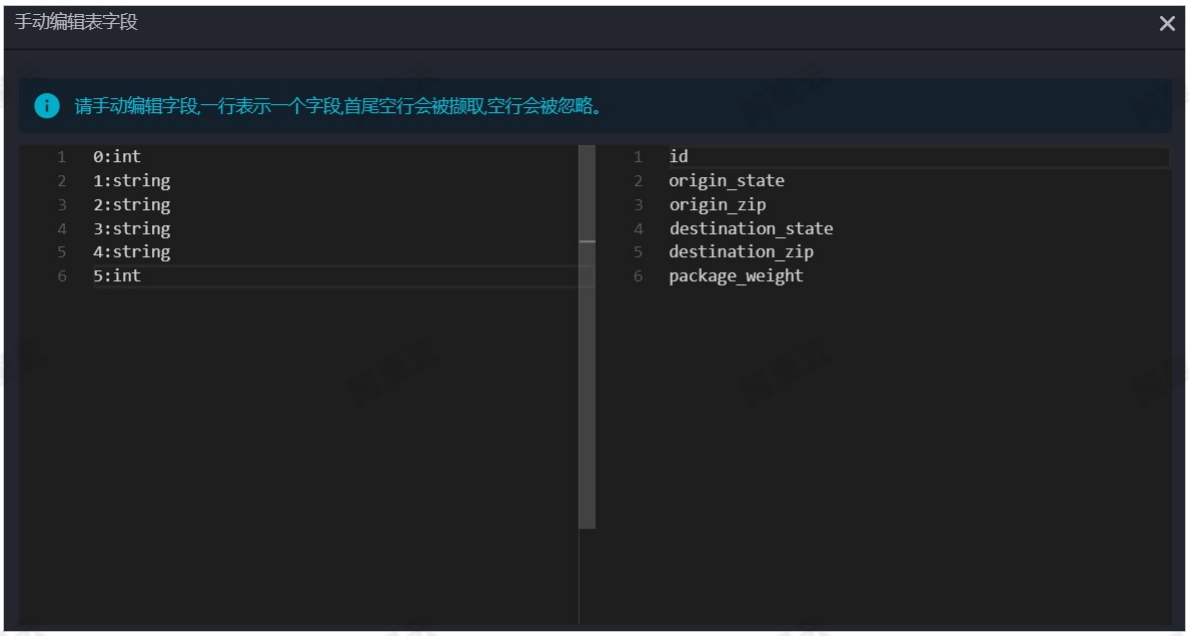
数据去向（Writer）配置信息：

数据源	选择ADS，系统将自动关联配置AnalyticDB数据源时设置的数据源名称。
表	选择AnalyticDB中的一张表，将OSS中的数据同步至该表中。
导入模式	根据AnalyticDB中表的更新方式设置导入模式，本例为实时导入。

字段映射配置信息：

> *注意列与列之间映射的字段类型是否有做数据兼容。*

如果OSS字段显示不全，单击**编辑**按钮，在**手动编辑表字段**窗口，手动增加或者修改字段和数据类型。



同行映射	自动将同一行的数据设置映射关系。
自动排版	设置完映射关系后，字段排序展示。

通道控制配置信息：

DMU	任务运行所需要的资源量。
作业并发数	配置的时候会结合读取端指定的切分建，将数据分成多个Task，多个Task同时运行，以达到提速的效果。
同步速率	设置同步速率可保护读取端数据库，以避免抽取速度过大，给读取端造成太大的压力。同步速率建议限流，结合源库的配置，请合理配置抽取速率。
错误记录数超过	

任务资源组	

5. 单击**保存**和**提交**，配置任务需要的其他信息。
6. 完成同步任务的配置后，先**保存**和**提交**节点，再单击**运行**，开始导入操作。

后续操作

成功将数据导入AnalyticDB后，您就可以使用AnalyticDB对数据进行计算分析。

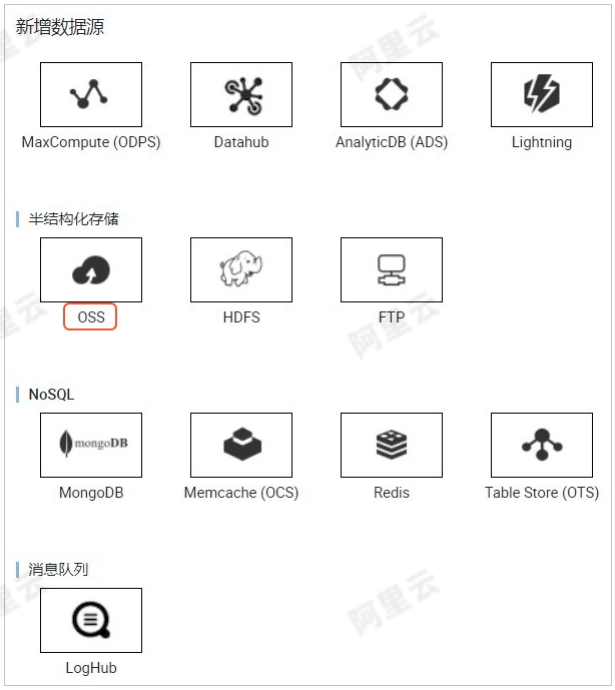
6.10. 使用OSS和DataWorks实现AnalyticDB MySQL 2.0整库迁移

6.10.1. 配置OSS数据源

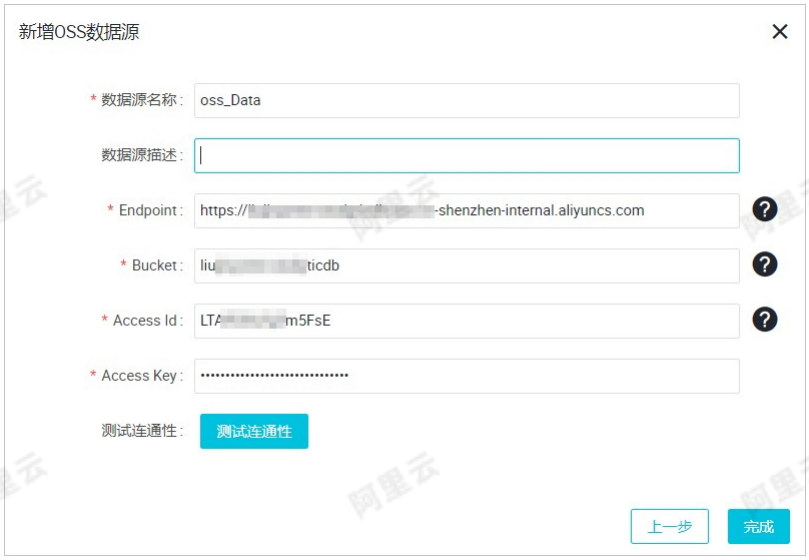
本文介绍如何配置OSS数据源。

1. 登录DataWorks控制台，单击对应项目栏中的**进入数据集成**。
   本例项目名为liujing_test_analyticdb。

2. 单击**新曾数据源**，数据源选择**OSS**。



3. 在**新增OSS数据源**页面，进行以下参数配置。



详细的参数配置如下表所示。

配置项	说明
数据源名称	为数据源指定一个名字，便与后续管理。
数据源描述	添加数据源描述，该项为可选填项。
Endpoint	OSS EndPoint（地域节点），格式为 `http://oss.aliyuncs.com` 。
Bucket	OSS Bucket名字。
Access Id	OSS Bucket创建者的Access Id。
Access Key	Access Id对应的Key。 如何获取AK信息，请参见如何获取主账号AccessKeyId和AccessKeySecret或者如何获取子账号AccessKeyId和AccessKeySecret。

4. 完成上述参数配置后，单击**测试连通性**进行连通性测试，测试通过后单击**完成**。

6.10.2. 通过DLA将OSS数据迁移至AnalyticDB MySQL 2.0

Data Lake Analytic（DLA）作为云上数据处理的枢纽，通过DLA可以读取存储在阿里云对象存储服务（OSS）中的数据，对其进行清洗得到您需要的数据。再通过DLA将清洗后的数据写入AnalyticDB MySQL 2.0集群，利用AnalyticDB强大的计算引擎，对数据进行毫秒级计算分析。

本文档将详细介绍如何通过DLA将OSS中的数据迁移至AnalyticDB。

> 重要　DLA支持的OSS数据类型有LOG、CSV和JSON。

DLA支持的Region和VPC对照表

目前只有华东1（杭州）、华东2（上海）、华北2（北京）三个Region的DLA支持连接AnalyticDB MySQL 2.0集群。要求AnalyticDB MySQL 2.0必须开通以下表中Region对应的VPC，否则DLA无法连接AnalyticDB MySQL 2.0集群。

华东1（杭州）	cn-hangzhou-g	vpc-bp1g66t4f0onrvbht2et5	vsw-bp1nh5ri8di2q7tkof474
华东2（上海）	cn-shanghai-d	vpc-uf6wxkgst74es59wqareb	vsw-uf6m7k4fcq3pgd0yjfdnm
华北2（北京）	cn-beijing-g	vpc-2zeawsrpzbelyjko7i0ir	vsw-2zea8ct4hy4hwsrcpd52d

准备工作

1. 开通DLA、AnalyticDB以及OSS服务，且DLA、AnalyticDB、OSS所属Region相同。本示例中三个服务所属Region均为华东2（上海）。
2. 在DLA中创建服务访问点，即创建VPC专有网络。详情请参见创建服务访问点。
3. 为AnalyticDB MySQL集群创建VPC专有网络，详情请参见创建VPC专业网络。
4. 如果您的AnalyticDB MySQL集群所属VPC与DLA相同，则跳过本步骤，继续下面的步骤。通过MySQL命令行工具或者客户端连接分析型数据库MySQL版，执行以下SQL修改AnalyticDB MySQL集群的VPC，使DLA能够连接AnalyticDBMySQL集群。本示例使用DLA在华东2（上海）的VPC。

```
ALTER DATABASE ads_database set zone_id='cn-shanghai-d' vpc_id='vpc-uf6wxkgst74es59wqareb' vswitch_id='vsw-uf6m7k4fcq3pgd0yjfdnm';
```

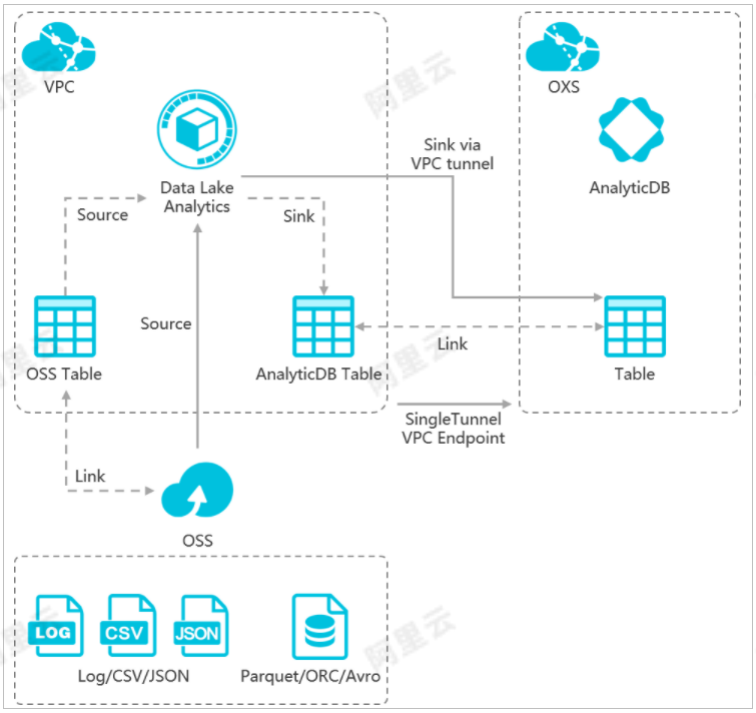
5. 在AnalyticDBMySQL集群中创建表，详情请参见创建表，用于存储OSS数据。

   本示例中的表为shipping和order_table表，表结构信息如下所示。

```
CREATE TABLE ads_database.shipping (
id bigint NOT NULL COMMENT '',
origin_state varchar NOT NULL COMMENT '',
origin_zip varchar NOT NULL COMMENT '',
destination_state varchar NOT NULL COMMENT '',
destination_zip varchar NOT NULL COMMENT '',
package_weight int NOT NULL COMMENT '',
PRIMARY KEY (id)
)
PARTITION BY HASH KEY (id) PARTITION NUM 23
TABLEGROUP ads
OPTIONS (UPDATETYPE='realtime')
COMMENT ''
```

```
CREATE TABLE ads_database.order_table (
  customer_id bigint NOT NULL COMMENT '',
  order_id varchar NOT NULL COMMENT '',
  order_time date NOT NULL COMMENT '',
  order_amount double NOT NULL COMMENT '',
  order_type varchar NOT NULL COMMENT '',
  address varchar NOT NULL COMMENT '',
  city varchar NOT NULL COMMENT '',
  order_season bigint COMMENT '',
  PRIMARY KEY (customer_id)
)
PARTITION BY HASH KEY (customer_id) PARTITION NUM 32
TABLEGROUP ads
OPTIONS (UPDATETYPE='realtime')
COMMENT ''
```

实施流程



注意事项

- AnalyticDB为主键覆盖逻辑，若整个INSERT FROM SELECT的ETL任务失败，需要整体重试。
- 在AnalyticDB端查询写入数据时，会有一定的延迟，具体延迟时间取决于AnalyticDB中ECU的规格。
- 建议将ETL任务尽量切成小的单位分批执行。例如，OSS数据200GB，在业务允许的情况下，200GB的数据切成100个文件夹，每个文件夹2GB数据。在DLA中建100张表，100张表分别做ETL，单个ETL任务失败，只重试单个ETL任务。
- OSS数据迁移至AnalyticDB后，需要修改AnalyticDB VPC，否则除了DLA可以连接AnalyticDB以外，其他阿里云服务将无法通过AnalyticDB当前VPC访问AnalyticDB。

```
ALTER DATABASE ads_database set zone_id='cn-shanghai-d' vpc_id='vpc-uf6wxkgst*****' vswitch_id='vsw-uf6m7k4fcq3pgd0y****';
```

步骤一：在DLA中创建AnalyticDB数据库映射

> **说明** 如果您的AnalyticDB MySQL集群所属VPC与DLA所属VPC不同，请按准备工作步骤4所示，修改AnalyticDB MySQL集群的VPC，否则DLA中无法创建AnalyticDB MySQL数据库映射。

登录DLA控制台，**登录DMS**。如何获取登录使用的账号密码，请参见快速入门相关内容），在DLA中创建一个AnalyticDB的数据库连接。语法如下：

```
CREATE SCHEMA `ads_database_schema` WITH DBPROPERTIES
(
  CATALOG = 'ads',
  LOCATION = 'jdbc:mysql://ads-database-********-vpc.cn-shanghai-1.ads.aliyuncs.com:10001/ads_database',
  USER='AnalyticDB AccessKey ID',
  PASSWORD='AnalyticDB AccessKey Secret'
);
```

- `CATALOG = 'ads'`：创建的是DLA到AnalyticDB的数据库连接。
- LOCATION：AnalyticDB的连接信息，格式为 `连接地址:端口号/AnalyticDB名字` ，通过AnalyticDB 控制台获取。

步骤二：在DLA中创建AnalyticDB外表

DLA数据库建完之后，接下来就可以建外表。外表分为两种情况：

- 外表的表名、字段名或者字段类型与AnalyticDB中的表信息不一致。
- 外表的表名、字段名或者字段类型与AnalyticDB中的表信息完全一致。

DLA中的表名、列名与AnalyticDB中表信息一致（建议一致）

通过以下示例在DLA中创建shipping外表：

```
CREATE EXTERNAL TABLE shipping (
  id bigint NOT NULL COMMENT '',
  origin_state varchar NOT NULL COMMENT '',
  origin_zip varchar NOT NULL COMMENT '',
  destination_state varchar NOT NULL COMMENT '',
  destination_zip varchar NOT NULL COMMENT '',
  package_weight int NOT NULL COMMENT '',
  PRIMARY KEY (id)
)
```

注意：

- EXTERNAL：创建的表是外表。
- 表名、字段名（列名）、字段类型均与AnalyticDB中的表信息完全一致。

DLA中的表名、列名与AnalyticDB中表信息不一致

通过以下示例在DLA中创建order_table的外表order_table1。其中，order_table1中字段名与order_table不同，order1_id数据类型与order_id不同。

```
CREATE EXTERNAL TABLE order_table1 (
  customer1_id bigint NOT NULL COMMENT '',
  order1_id bigint NOT NULL COMMENT '',
  order1_time date NOT NULL COMMENT '',
  order1_amount double NOT NULL COMMENT '',
  order1_type varchar NOT NULL COMMENT '',
  address1 varchar NOT NULL COMMENT '',
  city1 varchar NOT NULL COMMENT '',
  order1_season bigint COMMENT '',
  PRIMARY KEY (customer1_id)
)
tblproperties (
   table_mapping = 'ads_database.order_table',
   column_mapping = 'customer1_id,customer_id; order1_id,order_id; order1_time:order_time,
   order1_amount:order_amount, order1_type:order_type, address1:address,
   city1:city,order1_season:order_season'
);
```

- EXTERNAL：创建的表是外表
- tblproperties：外表与源表的映射关系
- table_mapping：表名映射
- column_mapping：字段（列）映射

步骤三：在DLA中创建到OSS连接

```
CREATE SCHEMA oss_data_schema with DBPROPERTIES(
  catalog='oss',
 location = 'oss://oss_bucket_name/table/'
 );
```

- `catalog='oss'`：创建的是DLA到OSS的连接。
- location：数据文件所在的OSS Bucket的目录，需以 `/` 结尾表示目录。

  后续建表的LOCATION所指向的数据文件，必须在这个OSS目录下。本示例中table就是OSS中存放文件的Object。

步骤四：在DLA中创建指向OSS文件的外表

本示例中dla_table_1、dla_table_2分别与AnalyticDB中的shipping、order_table表信息一致。且OSS Bucket中table Object下已经上传好dla_table_1、dla_table_2文件。

```
CREATE EXTERNAL TABLE IF NOT EXISTS dla_table_1 (
  id bigint NOT NULL COMMENT '',
  origin_state varchar NOT NULL COMMENT '',
  origin_zip varchar NOT NULL COMMENT '',
  destination_state varchar NOT NULL COMMENT '',
  destination_zip varchar NOT NULL COMMENT '',
  package_weight int NOT NULL COMMENT ''
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION 'oss://oss_bucket_name/table/';
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS dla_table_2 (
   customer_id bigint NOT NULL COMMENT '',
   order_id varchar NOT NULL COMMENT '',
   order_time date NOT NULL COMMENT '',
   order_amount double NOT NULL COMMENT '',
   order_type varchar NOT NULL COMMENT '',
   address varchar NOT NULL COMMENT '',
   city varchar NOT NULL COMMENT '',
   order_season bigint COMMENT '',
   PRIMARY KEY (customer_id)
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION 'oss://oss_bucket_name/table/';
```

步骤五：在DLA中执行INSERT FROM SELECT将OSS中的数据迁移至AnalyticDB

通过MySQL命令行工具或者DMS for Data Lake Analytics控制台执行INSERT FROM SELECT语句。

注意事项

- INSERT FROM SELECT通常为长时运行任务，建议通过**异步执行**方式。
- 使用MySQL命令行时，需要在命令行中指定 `-c` 参数，用来识别MySQL语句前的hint。

```
mysql -<hxxx> -<Pxxx> -<uxxx> -pxxx <db_name> -c
```

  - h：DLA的**连接串信息**，通过DLA控制台获取。如果您的MySQL客户端安装在与DLA同地域的ECS上，且ECS所属VPC与DLA所属VPC相同，则使用DLA **VPC专有网络**连接串。否则，使用DLA**经典网络**连接串。
  - P：DLA的连接**端口**，通过DLA控制台获取。
  - u：用户名，如何获取用户名，请参见快速入门相关内容。
  - db_name：DLA中的数据库名。
- 如果在INSERT INTO子句和SELECT子句中没有指定列信息，请确保源表和目标表的列定义顺序一致，列类型对应匹配。
- 如果在INSERT INTO子句和SELECT子句中指定列的信息，请确保两者中的列的顺序符合业务需要，列类型对应匹配。

写入数据

异步执行以下SQL，将OSS中table目录下dla_table_1文件中的数据插入AnalyticDB中ads_database数据库中的shipping表。

```
-- 执行OSS到AnalyticDB的全量数据插入
/*+run-async=true*/
INSERT INTO ads_database_schema.shipping
SELECT * FROM oss_data_schema.dla_table_1;
```

异步执行以下SQL，将OSS中table目录下dla_table_1文件中 `order_amount > 2` 的数据插入AnalyticDB中ads_database数据库中的order_table表。

```
-- 执行OSS到AnalyticDB的数据插入，包含对OSS数据的筛选逻辑
/*+run-async=true*/
INSERT INTO ads_database_schema.order_table1 (customer1_id, order1_id, order1_time, order1_amount,order1_type,address1,city1,order1_season)
SELECT customer_id, order_id, order_time, order_amount,order_type,address,city,order_season
FROM oss_data_schema.dla_table_2
WHERE order_amount > 2
LIMIT 10000;
```

查询写入结果

异步执行INSERT FROM SELECT语句，将返回一个task id，在AnalyticDB通过task id轮询任务执行情况，如果status为**SUCCESS**，则任务完成。

```
SHOW query_task WHERE id = 'q201812241524sh385b7d4c0049869';
```

后续操作

将OSS数据迁移至AnalyticDB后，您就可以使用AnalyticDB MySQL提供的各项功能进行数据分析计算。

6.11. 通过日志服务同步ECS日志数据到AnalyticDB MySQL 2.0

本文以ECS上的Nginx日志为例，介绍如何通过日志服务将日志数据写入AnalyticDB MySQL 2.0集群。

背景信息

日志作为一种特殊的数据，对处理历史数据、诊断问题以及了解系统活动等有着非常重要的作用。对数据分析人员、开发人员或者运维人员而言，日志都是其工作过程中必不可缺的数据来源。

阿里云从用户角度出发，支持通过日志服务（Simple Log Service，简称SLS）将云服务器ECS（Elastic Compute Service）上的Nginx访问日志、Log4j日志、Apache日志以及结构化文本等文件实时同步至AnalyticDB for MySQL，从而使用AnalyticDB for MySQL进行实时日志分析。

前提条件

- 在LOG中完成以下准备工作。
  i. 首次使用日志服务时，您需要使用阿里云账号登录日志服务产品详情页，单击**立即开通/登录**，进入购买页面，然后单击**立即开通**即可成功购买日志服务，系统自动跳转到日志服务控制台。

  > **说明**
  > 如果您之前已经开通过日志服务，请直接从创建Project开始。

  ii. 参考LOG快速入门，创建Project、创建Logstore。
  本示例创建test-ecs-log Project和nginx-logstore Logstore。
- 开启并获取当前阿里云账号的Access Key。
  i. 登录日志服务控制台，将鼠标放在右上方的用户名区域，在弹出的快捷菜单中选择**AccessKey 管理**。
  ii. 系统弹出安全提示对话框，单击**继续使用AccessKey**，单击**查看Secret**，页面显示**AccessKeyId**和**AccessKeySecret**。
- 参照分析型数据库MySQL版快速入门、创建表组和创建表，在AnalyticDB for MySQL中创建目标数据库、表组以及表。

实施步骤

步骤一：配置日志Logtail

Logtail接入服务是日志服务提供的日志采集Agent，通过日志服务控制台帮助您实时采集阿里云ECS、自建IDC或者其他云厂商等服务器上的日志。

在ECS服务器上安装Logtail客户端，然后添加Logtail采集配置，Logtail即可采集日志到日志服务。Logtail日志采集流程为：监听文件、读取文件、处理日志、过滤日志、聚合日志以及发送数据到日志服务。

1. 在ECS中安装Logtail。
2. 登录日志服务控制台，单击Project名称。
3. 在**概览**页面，单击**接入数据**，选择**Nginx-文本日志**。
4. 在**选择日志空间**步骤中，**日志库Logstore**设置为之前创建的nginx-logstore Logstore。
5. 单击**下一步**，在**创建机器组**步骤中，选择**ECS机器**，系统自动拉取与日志服务地域相同的ECS实例，勾选您需要的ECS实例ID，单击**立即执行**，完成后单击**确认安装完毕**。
6. 设置**机器组名称**和ECS服务器的**IP地址**，单击**下一步**。

   > **说明**
   > 您可以将多台ECS服务器的IP地址添加到**IP地址**中。

7. 在**我的机器组**中选中**源机器组**，然后单击右移按钮将其移动到**应用机器组**中，单击**下一步**。
8. 在**Logtail配置**中，按照页面提示进行参数配置。
   参数含义请参见采集并分析Nginx访问日志。
9. 确认**Nginx键名称**。
   日志服务自动提取Nginx日志中的键名，请确认是否正确。

   > **说明**
   > Nginx日志格式中的 `$request` 会被提取为 `request_method` 和 `request_uri` 两个键。

0. 确认**是否丢弃解析失败日志**，并单击**下一步**。
   开启该功能后，解析失败的Nginx访问日志不会上传到日志服务；关闭该功能后，Nginx访问日志解析失败时将上传原始Nginx访问日志到日志服务。
   确保日志机器组心跳正常，即可预览采集上来的数据。

步骤二：将日志投递到AnalyticDB for MySQL

通过日志服务数据处理模块的导出功能，您可以将Logstore中采集到的日志投递到AnalyticDB for MySQL。

1. 登录日志服务控制台，单击Project名称。
2. 在**日志库**列表中，单击目标Logstore名称前的**>** > **数据处理** > **导出** > **AnalyticDB**，选择需要的AnalyticDB for MySQL集群。
   您需要为日志服务授权，允许日志数据写入AnalyticDB for MySQL。
   i. 单击AnalyticDB后的**+**，系统自动提示您进行授权操作，单击**权限**。
   ii. 在云资源访问授权页面，单击**同意授权**，将角色**AliyunAnalyticDBAccessingLogRole**授予日志服务。
3. 在**日志库**列表中，单击目标Logstore名称前的**>** > **数据处理** > **导出** > **AnalyticDB**后的**+**，在数据投递参数配置页面进行参数配置。

参数	说明
投递名称	为数据投递取一个名字，便于后续管理。
投递描述	为数据投递添加一段描述。
集群版本	选择2.0。
数据库名称	目标AnalyticDB for MySQL数据库的名字。
表组名	目标AnalyticDB for MySQL中的表组。
表名	目标AnalyticDB for MySQL中的表名，投递过来的日志数据将存储在该表中。

AccessKey ID	当前阿里云账号的AccessKey ID，请参见获取阿里云账号的AccessKey ID和AccessKey Secret获取AccessKey ID和AccessKey Secret。
AccessKey Secret	当前阿里云账号的AccessKey Secret。
字段映射	日志字段与AnalyticDB for MySQL表字段的映射。 可以单击**增加一条**添加字段。
投递开始时间	设置日志数据的投递时间。
是否过滤脏数据	是否过滤那些不符合要求的数据。

4. 完成上述参数配置后，单击**确定**，系统将在设定的投递时间点将日志数据实时投递到AnalyticDB for MySQL。

6.12. 使用DTS同步RDS MySQL数据到AnalyticDB MySQL 2.0

本文以RDS MySQL数据源为例，介绍如何通过DTS将RDS MySQL数据实时同步至AnalyticDB for MySQL中。

前提条件

- 根据RDS MySQL快速入门，准备好测试数据。
- 根据AnalyticDB MySQL版2.0快速入门完成数据库创建、表组创建等准备工作。

背景信息

数据传输（Data Transmission）DTS的数据同步功能旨在帮助用户实现两个数据源之间的数据实时同步。数据同步功能可应用于异地多活、数据异地灾备、本地数据灾备、数据异地多活、跨境数据同步、查询与报表分流、云BI及实时数据仓库等多种业务场景。

通过数据同步功能，您可以将MySQL、PolarDB-X（原DRDS）以及PolarDB中的数据同步至AnalyticDB for MySQL中，其中MySQL可以是RDS MySQL、其他云厂商或线上IDC自建MySQL或者ECS自建MySQL。

同步对象的选择粒度可以为库、表、列，您可以根据需要同步某几个表的数据，或者只同步表中的某几列数据。

数据同步支持库、表、列名映射，您可以进行不同库名之间的数据或两个不同表名之间的数据同步。

注意事项

- 配置同步链路过程中，如果目标表中列信息与源表不同，DTS支持字段映射功能。详细步骤，请参见库表列映射。
- 如果需要同步的表数量较少且分析型数据库MySQL版表结构与源端表差异较大的话，可以在分析型数据库MySQL版中提前创建表，配置同步链路时候需要把下述步骤6中结构初始化的勾选项去掉即可。

数据同步流程介绍

通过DTS实时同步RDS MySQL数据到AnalyticDB for MySQL需要如下步骤：

1. 步骤一：创建DTS同步作业。
2. 步骤二：配置同步链路。
3. 步骤三：查看同步数据。

步骤一：创建DTS同步作业

创建DTS同步作业需要用户支付一定的费用，DTS支持两种付费方式：包年包月（预付费）和按量付费。关于两种付费方式的价格详情，请参见DTS产品定价。本文以按量付费为例，介绍创建同步作业的详细步骤。

1. 进入DTS产品详情页，单击**立即购买**。
2. 根据业务需要在售卖页面上进行参数配置，配置完成后单击**立即购买**。

参数	说明
功能	数据同步。
源实例	MySQL。
源实例区域	本示例选择华东1（杭州）。
目标实例	AnalyticDB for MySQL。
目标实例区域	本示例选择华东1（杭州）。
同步拓扑	单向同步。
同步链路规格	本例选择large。
网络类型	专线。

3. 在**确认订单**页面，阅读并选中服务协议，根据提示完成支付流程。

步骤二：配置同步链路

1. 登录DTS控制台。

2. 在**数据传输**页面，单击左侧导航栏中的**数据同步**。
3. **选择地域**。
4. 在**同步作业**列表中，单击目标实例右侧的**配置同步链路**，在**选择同步通道的源及目标实例**页面进行参数配置，详细的参数配置如下表所示。

参数	说明
同步作业名称	可选项。
实例类型	本例选择RDS。
源实例地区	本例为华东1（杭州）。
实例ID	指定源实例的ID。
数据库账号	RDS账号。
数据库密码	RDS密码。
连接方式	非加密连接。
实例类型	AnalyticDB for MySQL。
目标实例地区	本例为华东1（杭州）。
数据库	本例为 `ads_DataBase` 。

5. 完成上述参数配置后，单击**授权白名单并进入下一步**。
6. 在**选择同步对象**页面，完成下面两步骤配置后，单击**下一步**。
   i. 选中**结构初始化**和**全量数据初始化**。
   ii. 在**源库对象**中把同步的表移动到右侧的**已选择对象**中。
7. 在配置表信息页面，设置如下参数。

参数	说明
类型	分区表或者维度表。
主键	支持复合主键，保证数据唯一。
分区列	选取参考一级分区列选择。
分区数	建议128。

8. 完成上述参数配置后，单击**预检查并启动**，弹出**预检查**页面。
   i. 如果预检查显示**失败**，可以根据提示DTS预检查信息进行排错处理。
   ii. 预检查全部成功后，单击**关闭**。

步骤三：查看同步数据

1. 返回DTS控制台，在同步列表中的同步概况中查看同步延时和速度。
2. 进入AnalyticDB MySQL版控制台，在 `ads_DataBase` 数据库中可以查看到同步过来的数据表。

7.2.0版向量分析

7.1. 功能概述

实现原理

分析型数据库MySQL版的向量分析旨在帮助您实现非结构化数据的近似检索和分析，其实现原理是通过AI算法提取非结构化数据的特征，然后利用特征向量唯一标识非结构化数据，向量间的距离用于衡量非结构化数据之间的相似度。

向量分析继承了分析型数据库MySQL版的MPP查询架构以及全索引结构，通过SIMD指令加速、高效索引算法、混合检索CBO策略以及低成本存储技术，帮助您实现高性能、低成本的非结构化数据近似查询和分析。

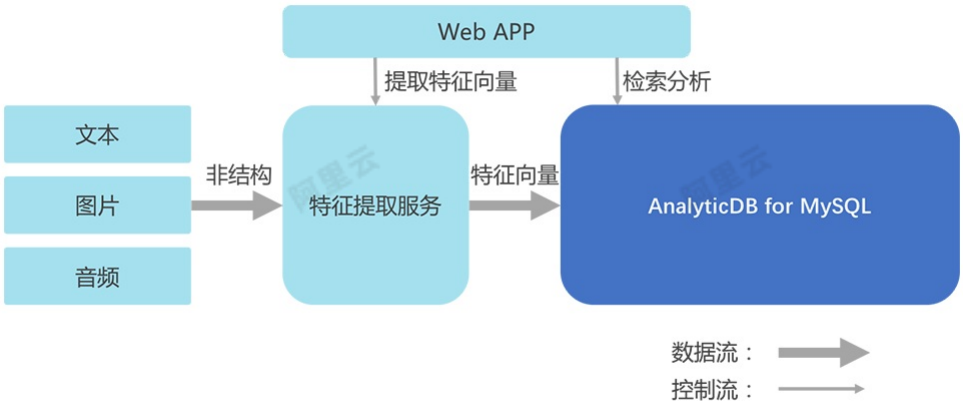
典型应用场景

通过分析型数据库MySQL版向量分析，您可以非常容易地搭建各种智能化应用，例如以下几种应用。

- 以图搜图，通过图片检索图片。
- 声纹匹配，通过音频检索音频。
- 基于语义的文本检索和推荐，通过文本检索近似文本。
- 文件去重，通过文件指纹去除重复文件。
- 商品图片分析，在大量图片中分析哪些图片包含了同一个商品。

典型架构

分析型数据库MySQL版向量分析功能的典型应用架构如下图所示。



7.2. 功能优势

分析型数据库MySQL版向量分析功能在通用性、性能优化和产品化上与普通向量检索系统相比有以下优势。

- 高维向量数据的高准度和高性能

  以典型的人脸512维向量为例，分析型数据库MySQL版向量分析提供百亿向量100 QPS、50毫秒响应时间（RT）约束下99%的数据召回率；两亿向量1000 QPS、1秒RT约束下99%的数据召回率。
- 结构化和非结构化混合检索

  例如，可以检索与输入图片中的连衣裙相似度最高、价格在100元到200元之间且上架时间在最近1个月以内的产品。
- 支持数据实时更新

  传统的向量分析系统中数据只能按照T+1更新，不支持数据实时写入。分析型数据库MySQL版向量分析支持数据实时更新和查询。
- 支持向量分析碰撞

  分析型数据库MySQL版向量分析支持KNN-Join SQL，即比较一批向量与另外一批向量的相似度，类似于Spark中的KNN-Join操作。

  典型的应用场景有商品去重，计算新加入的商品与历史商品库中有哪些是相似的。人脸聚类，计算一段时间内的人脸库中，哪些人脸是同一个人。
- 易用性

  分析型数据库MySQL版向量分析申请即可使用，支持标准SQL，简化开发流程。同时，分析型数据库MySQL版向量分析内置常用特征提取和属性提取，也支持集成第三方特征提取服务。

高维向量数据的高准度和高性能

分析型数据库MySQL版向量分析支持两种模式：无损模式和微损模式。无损模式下，保证100%的数据召回率。微损模式下，分析型数据库MySQL版会为向量构建索引，索引会带来召回率的轻微下降，保证99%的数据召回率。您可以参考下表合理选择向量分析模式。

特点及场景	模式	数据召回率	数据量	QPS
小数据量（写入数据较少）、高QPS、精度无损。典型应用场景： • 人脸门禁考勤 • 违禁图片库 • 黑名单库	无损	100%	百万级别	单节点100 QPS，利用维度表实现线性扩展。

中等规模数据量、高QPS、有较少精度损失。典型应用场景： • 实时库 • 全国人口库	微损	99%	亿级别	1000 QPS
海量数据（写入较多）、低QPS、精度有损。典型应用场景： 历史库	微损	99%	百亿级别	100 QPS

结构化和非结构化混合检索

以检索与输入图片中的连衣裙相似度最高、价格在100元到200元之间且上架时间在最近1个月以内的产品为例，介绍结构化和非结构化混合检索。

有下列商品库，其中商品表products的字段如下所示。

字段	类型	说明
Id	Char(64)	商品
Name	Varchar(256)	商品名字
Price	Float	价格
InTime	DateStamp	入库时间
Url	Varchar(256)	图片链接
Feature	Float(512)	图片特征

向量检索SQL如下：

```
SelectId,Pricefrom ann
    (SELECT *
    FROM products
    WHERE price >100
            AND price <=200
            AND InTime>'2019-03-0100：00：00'
            AND InTime<='2019-03-3100：00：00',Feature,'{1.0,2.0,…,512.0}',100,'DISTANCE_THRESHOLD=0.2');
```

其中 `ann` 为向量检索自定义函数，包含下列五个参数：

- 子查询：

```
SELECT *
FROM products
WHERE price >100
        AND price <=200
        AND InTime>'2019-03-0100：00：00'
        AND InTime<='2019-03-3100：00：00';
```

- 向量列名：`Feature`
- 查询向量：`'{1.0,2.0,…,512.0}'`
- Topk：100
- 参数列表：`DISTANCE_THRESHOLD=0.2`，距离阈值为0.2。

数据实时更新

分析型数据库MySQL版向量分析支持数据实时更新，写入数据后可以立即进行查询。数据的写入方式与传统数据库一样，使用INSERT语句插入向量数据。写入性能可线性扩展，单个H8节点每秒可写入1000条记录。

向量分析碰撞

以上述商品库为例，为实现商品去重，需要检索最近一天加入的商品与上个月的商品库中有哪些商品是相似的。SQL如下所示。

```
SELECT A.id AS ida,
        B.id AS idb from(select*from products
WHERE inTime >'2019-03-31 00：00：00') A
CROSS JOIN (SELECT * FROM products WHERE InTime>='2019-02-01 00：00：00' and InTime<'2019-02-29 00：00：00') B WHERE Vector_Distance(A.feature,
B.feature,'DotProduct')>0.9
```

SQL详解：

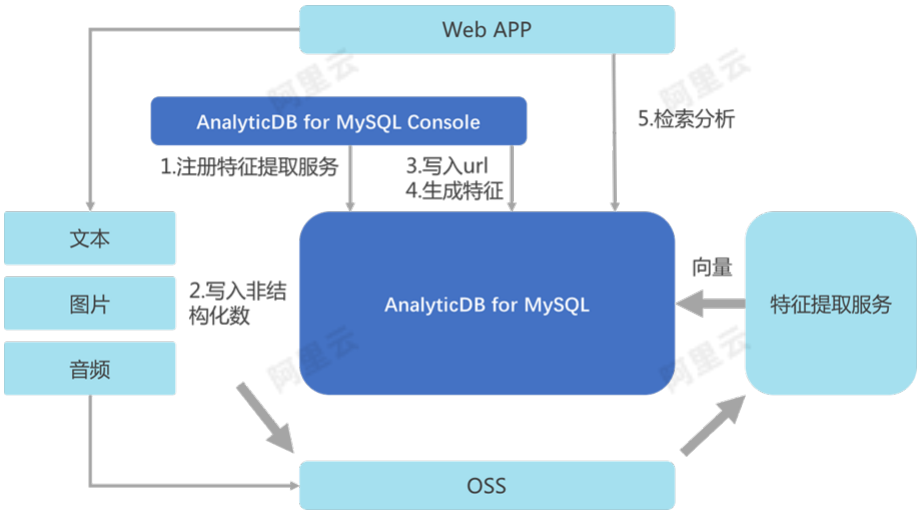
上述SQL把 `'2019-03-31 00：00：00'` 之后写入的数据与二月份的数据做笛卡尔积，把向量点积大于0.9的商品的对应ID提取出来。

易用性

向量分析完整继承了分析型数据库MySQL版的所有商业工具和生态，并支持常用的特征提取模型和第三方特征提取服务。

- 申请即可使用：开通分析型数据库MySQL版服务即可使用向量分析。
- 分析型数据库MySQL版全面兼容MySQL协议和SQL 2003，降低开发成本。
- 内置常用特征提取模块，支持集成第三方特征提取服务。

为了让您对非结构化数据拥有更多的自主控制权，您可以把非结构化数据保存在OSS或者图片服务器上（下图使用OSS），非结构化数据的保存地址即URL存储在分析型数据库MySQL版中，整体架构如下所示。



1. 通过分析型数据库MySQL版控制台注册特征提取服务。
2. 非结构化数据保存到OSS，同时返回访问的URL。
3. 非结构化数据的存储地址即URL保存在分析型数据库MySQL版中。
4. 通过Web App调用分析型数据库MySQL版的自定义函数生成向量特征，分析型数据库MySQL版后台通过调用特征提取服务从OSS读取非结构化数据，提取特征，并把特征向量保存在分析型数据库MySQL版中。所有这些操作只需要一条SQL便可轻松完成，SQL语句示例如下。

```
SELECT CLOTHES_FEATURE_EXTRACT_V1('https://xxx/1036684144_687583347.jpg');
INSERT INTO product(id, url, feature) VALUES (0, 'http://xxx/1036684144_687583347.jpg', CLOTHES_FEATURE_EXTRACT_V1(url));
```

SQL详解：

`CLOTHES_FEATURE_EXTRACT_V1` 为商品特征提取的自定义函数，传入商品图片URL，提取商品特征向量，该向量可以用来做商品检索和属性提取。

对于常用的人脸特征提取、文本特征提取BERT模型以及服装特征提取也已经内置分析型数据库MySQL版服务中，您也可以使用您自己的特征提取服务。

7.3. 竞品分析

本文讨论PostgreSQL相似图像搜索插件imgsmlr和文本转化为向量插件word2vec对比分析型数据库MySQL版向量分析在使用过程中存在哪些问题。

PostgreSQL imgsmlr

imgsmlr是PostgreSQL上的一个以图搜图插件，实现了从图片转向量和向量检索的功能，且所有操作都使用SQL封装，您可以很容易地搭建一个简易的以图搜图应用。

- ***使用过程中存在的问题***
  - imgsmlr存在以下问题：
    - 图片特征提取算法粗燥，准确率和数据召回率较低。

      imgsmlr抽取图像特征的算法，以64*64的图像为例，imgsmlr取16个区域的平均值，生成16个浮点数，作为图像特征值。这种特征抽取算法，只适用于对图像检索准确度要求极低的情况，对目前广泛使用的基于卷积神经网络图像识别来说，完全无法实现其要求。
    - imgsmlr图片检索算法使用Gist索引，仅适用于向量维度较低的情况，例如16维，而人脸识别基本在256维以上。

      低维度的向量对索引非常友好，但是维度较低会丢失大量的图像信息，导致检索精度显著下降。维度上升对数据存储和计算提出了巨大的挑战，显然imgsmlr无法胜任。
    - 只支持数据批量导入，不支持数据实时更新。

PostgreSQL word2vec

word2vec支持使用Python实现文本转向量功能，且集成Facebook AI Research（简称FAIR）开源的Faiss向量索引库，实现了多种向量索引算法，例如Product quantizer (PQ) in flat mode（简称PQ）、IVFADC (coarse quantizer+PQ on residuals)（简称IVFADC）。

word2vec还具备向量分析能力，提供了KNN-Join，详情请参见FREDDY。

- ***实现逻辑***
  - word2vec的实现逻辑较为简洁：
    a. 通过封装的Python接口把文本转为向量。
    b. 通过调用Faiss封装的Python接口提取向量的索引数据，例如量化编码，粗聚类。
    c. 建立两张表：向量表、向量索引表。

       向量表保存向量、文本以及文本id。

       向量索引表保存PQ编码和粗聚类。
    d. 导入数据时，同时写入向量表和向量索引表。
    e. 检索时提供一个用户自定义函数，首先通过向量索引表做检索，查询粗聚类中心；然后做PQ扫描，得到top k向量id；最后通过向量ID到向量表检索得到文本。
- ***使用过程中存在的问题***

- word2vec可以快速实现向量检索功能，但是存在下列问题：
  - word2vec使用PostgreSQL PG的块页式内存和磁盘管理，性能较低。

    PostgreSQL的内存page只有8k，而修改page size需要重新编译代码，加大page size对磁盘刷盘效率的影响。以128 Byte的代码为例，PostgreSQL的内存page只能保存64条记录，扫描PG编码时，会有多次page访问。
  - 使用方式不够友好，不支持数据实时写入。

    向量和索引作为一个独立的表，需要额外写入。 还需要先计算好粗中心和PG编码，然后写入数据库，而不是把粗中心和编码作为索引由PostgreSQL来生成，增加了额外工作。
  - Faiss索引的效率较低，分析型数据库MySQL版向量分析是基于图的索引算法，其性能是Faiss索引性能的10倍。

7.4. 使用案例

7.4.1. 商品属性提取和多模搜索

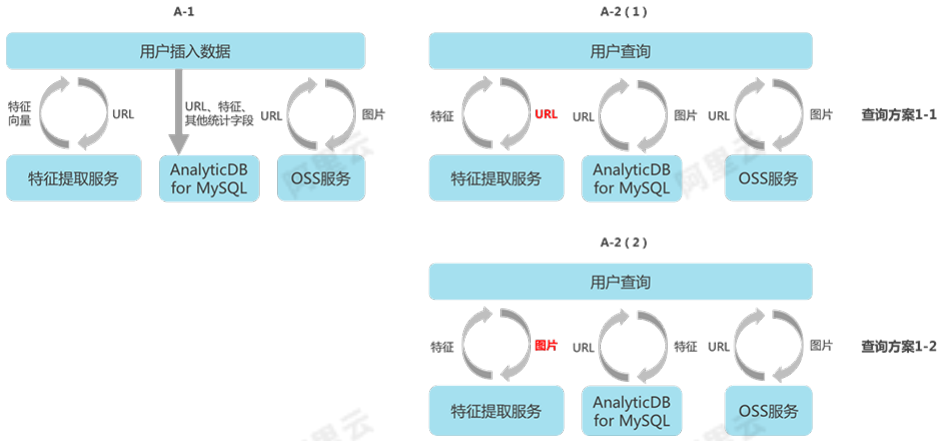
系统需求

电商卖家在准备新商品资料时，需要拍摄商品照片、标注商品类别和属性。以女装为例，需要设置子类目是裙子或者衬衣，衬衣需要标注是圆领、V领、长袖、短袖、风格等多种属性。随着商品类别的增长，将积累大量的图片素材。如果这些图片素材没有被很好地管理和利用，则经常会出现找不到之前已经准备好的素材，需要重新拍摄的情况。

分析型数据库MySQL版可以帮助电商卖家管理和检索图片素材，可以根据图片类别、属性或者相似图片做多模检索。例如，检索与输入图片最相似且价格在200元～300元之间的所有商品的图片。

实现架构

分析型数据库MySQL版作为商品属性提取和图片管理的核心组件，数据读写流程如下所示。



插入数据

应用端通过以下步骤向商品库中插入数据。

1. 应用端调用OSS服务，将图片插入OSS，获得对应的URL。

   > **说明** 当前只支持HTTP和HTTPS协议的URL。
2. 应用端调用特征提取服务，获得图片抽象后的特征向量。
3. 应用端调用分析型数据库MySQL版服务将步骤一中的URL和步骤二中的特征向量一起插入商品库。

查询数据

应用端可以采用上述架构图中的任意一种方案从商品库中查询数据，查询步骤如下所示。

1. 应用端调用OSS服务，将要查询的图片插入OSS，获得对应URL。

   > **说明** 当前只支持HTTP和HTTPS协议的URL。
2. 应用端调用特征提取服务，获得URL对应图片抽象后的特征向量，同时也支持直接传入图片获取特征向量。
3. 应用端调用分析型数据库MySQL版，获得相似特征向量图片对应的URL列表。
4. 应用端根据URL列表，调OSS服务，获取并返回图片。

示例

以下是一个简单的商品属性提取示例。

1. 通过以下SQL创建商品表 `products` 。

```
CREATE TABLE products (
  id bigint COMMENT '',
  image_url varchar COMMENT '',
  properties varchar COMMENT '',
  feature float[512] COMMENT '' COLPROPERTIES (DistanceMeasure='SquaredEuclidean',ExtractFrom='CLOTHES_FEATURE_EXTRACT_V1(image_url)'),
  PRIMARY KEY (id)
)
PARTITION BY HASH KEY (id) PARTITION NUM 2
CLUSTERED BY (id)
TABLEGROUP ads
OPTIONS (UPDATETYPE='realtime')
```

2. 插入数据

   分析型数据库MySQL版提供两种插入数据的方法。

   - 先调用用户自定义函数获取特征向量，然后插入数据。

     ```
     Select CLOTHES_FEATURE_EXTRACT_V1('https://example.com/img.jpg');
     ```

     > **说明** 可以先通过SELECT确认特征向量是否正确。

     ```
     insert into products (id, image_url, feature) values (10, 'https://xxx/img.jpg', CLOTHES_FEATURE_EXTRACT_V1 ('https://xxx/img.jpg'));
     ```

   - 根据输入的URL自动获取特征向量，然后插入数据。使用这种方法插入数据时，需要在建表时制定 `feature` 列特征以获取用户自定义函数和URL列，类似上述 `ExtractFrom='CLOTHES_FEATURE_EXTRACT_V1(image_url)'` 。

     ```
     insert into products(id, image_url) values(1, 'https://example.com/img.jpg');
     ```

3. 查询数据

   分析型数据库MySQL版支持SELECT查询中携带用户自定义函数传入图片URL。例如，检索与图片链接 `https://xxx/img.jpg` 最相似的前10条记录的商品 `id` 和 `image_url` 。

   ```
   Select id, image_url from ann(products, feature, CLOTHES_FEATURE_EXTRACT_V1 ('https://xxx/img.jpg'), 10);
   ```

7.4.2. 基于向量分析的个性化推荐系统

前提条件

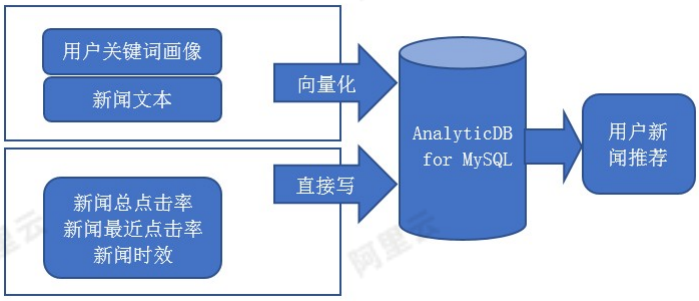
目标集群为云原生数据仓库 AnalyticDB MySQL 版。

背景信息

互联网时代个性化推荐已经渗透到人们生活的方方面面，例如常见的“猜你喜欢”、“相关商品”等。互联网能够对用户投其所好，向用户推荐他们最感兴趣的内容，实时精准地把握用户兴趣。目前很多成功的手机App都引入了个性化推荐算法，例如，新闻类的有今日头条新闻客户端、网易新闻客户端、阿里UC新闻客户端等；电商类的有拼多多、淘宝、天猫等。分析型数据库MySQL版推出的向量分析可以帮助您实现上述个性化推荐系统。

个性化推荐系统概述

以个性化新闻推荐系统为例，一篇新闻包含新闻标题、内容等内容，可以先通过NLP（Natural Language Processing，自然语言处理）算法，从新闻标题和新闻内容中提取关键词。然后，利用分析型数据库MySQL版向量内置的文本转换为向量函数，将从新闻标题和新闻内容中提取出的关键词转换为新闻向量导入分析型数据库MySQL版向量数据库中，用于用户新闻推荐，具体实现流程如下图所示。



整个新闻推荐系统由以下两个步骤实现。

1. **构建分析型数据库MySQL版向量库，得到用户特征向量。**

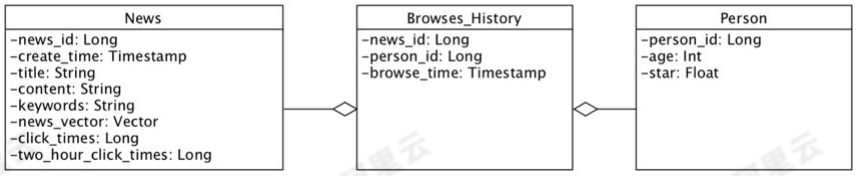
   通过分析用户历史浏览数据，构建相应的用户画像，建立用户偏好模型，得到用户特征向量。对于新闻推荐系统而言，可以从用户的浏览日志中得到用户历史浏览新闻详情，再从每条历史浏览新闻中提取关键词，建立用户画像。例如，某用户浏览了多条CBA（China Basketball Association，中国男子职业篮球联赛）季后赛新闻，这些新闻中包含了CBA、篮球、球星、体育等关键词，通过这些关键词可以得出该用户是一个CBA球迷。通过分析型数据库MySQL版向量将这些文本关键词转换为向量并导入到分析型数据库MySQL版向量库中，得到用户特征向量。

2. **根据分析型数据库MySQL版向量数据库和逻辑回归预测模型，将用户感兴趣的新闻推荐给用户。**

   通过分析型数据库MySQL版向量数据库，可以从互联网检索出前500条用户没有浏览过的新闻，但是这500条新闻却是该用户最感兴趣的新闻。然后，从这500条新闻中提取每条新闻的创建时间和点击率，最后，根据逻辑回归预测模型（该模型来自于用户以往的浏览的历史记录中），将用户感兴趣的新闻推荐给用户。

分析型数据库MySQL版内置的文本转换为向量函数采用BERT（Bidirectional Encoder Representations from Transformers）模型，同时支持中文和英文两种语言。该模型基于大量的语料进行训练，其中包含了语义信息，而且其查询精度比简单的TF-IDF（term frequency–inverse document frequency）算法高。

个性化推荐系统中数据库表结构设计



上图是个性化新闻推荐系统中分析型数据库MySQL版数据库表结构设计，包含了三张表 `news` 、 `person` 、 `browses_history` ，分别存储新闻信息、用户基本信息、用户浏览记录。

- **`news` 表**

  `news` 表存储新闻信息，包含新闻ID（ `news_id` ）、新闻创建时间（ `create_time` ）、新闻名字（ `title` ）、新闻内容（ `content` ）、新闻关键词（ `keywords` ）、总的用户点击数（ `click_times` ）、两个小时内的用户点击数（ `two_hour_click_times` ）、新闻向量（ `news_vector` ）。分析型数据库MySQL版根据新闻名称和内容得到新闻关键词 `keywords` ，然后，将新闻关键词转换为向量（ `news_vector` ）。向 `news` 表中插入数据时，系统自动根据关键词转换为向量，将向量和其他新闻信息一起插入 `news` 表。

```
CREATE TABLEGROUP recommendation_system_group;
CREATE TABLE news (
  news_id bigint primary key,
  create_time timestamp,
  title varchar(100),
  content varchar(200),
  keywords varchar(50),
  click_times bigint,
  two_hour_click_times bigint,
  news_vector float[768] COLPROPERTIES (DistanceMeasure='SquaredEuclidean',ExtractFrom='text_feature_extract_v1(keywords)'),
  ANN INDEX feature_index (news_vector) Algorithm=IVFPQ,
  primary key (news_id)
)
PARTITION BY HASH KEY (news_id) PARTITION NUM 2
CLUSTERED BY (news_id)
TABLEGROUP recommendation_system_group
OPTIONS (UPDATETYPE='realtime');
```

- **`browses_history` 表**

  `browses_history` 表存储用户浏览的新闻情况，包括新闻ID（ `news_id` ）、用户ID（ `person_id` ）、用户浏览新闻的时间（ `browse_time` ）。

```
CREATE TABLE browses_history (
  browse_id bigint,
  news_id bigint,
  person_id bigint,
  browse_time timestamp,
  primary key (browse_id)
)
PARTITION BY HASH KEY (browse_id) PARTITION NUM 2
CLUSTERED BY (browse_id)
TABLEGROUP recommendation_system_group
OPTIONS (UPDATETYPE='realtime');
```

- **`person` 表**

  `person` 表存储用户信息，包括用户的ID（ `person_id` ）、年龄（ `age` ）、星级（ `star` ）。

```
CREATE TABLE person(
  person_id bigint,
  age bigint,
  star float,
  primary key (person_id)
)
PARTITION BY HASH KEY (person_id) PARTITION NUM 2
CLUSTERED BY (person_id)
TABLEGROUP recommendation_system_group
 OPTIONS (UPDATETYPE='realtime');
```

个性化推荐系统实施步骤

结合上述个性化新闻推荐系统中分析型数据库MySQL版数据库表结构设计，整个系统的实施步骤如下所示。

- ***步骤一：从新闻中抽取新闻特征向量***

- 分析型数据库MySQL版通过内置的文本转换为向量函数，抽取新闻特征向量，然后将新闻特征向量存入新闻表 `news` 中。例如，执行以下 `SELECT` 将返回文本 `AnalyticDB向量数据库is very good!` 对应的特征向量。

  ```
  select TEXT_FEATURE_EXTRACT_V1('AnalyticDB向量数据库is very good!');
  ```

  例如，通过以下两个步骤将新闻信息存入新闻表 `news` 表中。

  a. **提取新闻关键词。**

  由于分析型数据库MySQL版暂时不支持关键词提取函数，您可以调用jieba（结巴中文NLP系统）中的关键词抽取函数 （`jieba.analyse.extract_tags(title + content, 3)`） 提取关键词。

  b. **执行INSERT将新闻信息（包含关键词和新闻特征向量）存入新闻表news表中。**

  ```
  insert into news(news_id, create_time, title, content, keywords, click_times, two_hour_click_times)
  values(1, now(),'什么是云原生数据仓库AnalyticDB MySQL版', '云原生数据仓库AnalyticDB MySQL版是融合数据库、大数据技术于一体的云原生企业级数据仓库服务。',
  '数据仓库 AnalyticDB MySQL', 123, 3);
  ```

  > 重要 执行 `INSERT` 时，分析型数据库MySQL版自动将新闻关键词转换为新闻特征向量，并将其存入向量库。

- ***步骤二：提取用户特征向量***

  a. **提取用户浏览关键词。**

  根据用户的新闻浏览日志，我们很容易得到用户的浏览关键词。例如，执行以下 `SELECT` 得到用户 `person_id` 为 `9527` 的浏览关键词。

  ```
  select keywords
  from Person p, Browses_History bh, News n
  where p.person_id = bh.person_id and bh.news_id = n.news_id and p.person_id = 9527
  ```

  将用户浏览关键词全部提取出来之后，就可以得到用户总的浏览关键词 。例如，用户 `person_id` 为 `9527` 浏览了关键词为 `CBA 体育` 、`总决赛` 、`上海` 、`广州` 的新闻。然后通过文本转换为向量函数，将用户 `person_id` 为 `9527` 浏览的关键词转换成向量。

  b. **将用户浏览关键词转换为用户特征向量。**

  ```
  select TEXT_FEATURE_EXTRACT_V1 ('CBA 体育 总决赛 上海 广州')
  ```

- ***步骤三：根据用户特征向量获取新闻推荐结果***

  通过用户特征向量，到新闻表 `news` 中查询相关的新闻信息。例如，执行以下 `SELECT` 将返回和用户相关的前500条新闻，同时系统也会过滤掉用户已经阅读过的文章。

  ```
  select id, title, content, ((now()-create_time)*w1 + click_times/(now()-create_time)*w2 + two_hour_click_times/(now()-create_time)*w3 + an
  n_distance * w4) as rank_score
  from ANN (News, news_vector,TEXT_FEATURE_EXTRACT_V1 ('CBA 体育 总决赛 上海 广州'), 500)
  order by rank_score desc;
  ```

  参数说明：

  - `ann_distance` ：用户与新闻的相关度。
  - `create_time` ：新闻的创建时间。
  - `click_times/(now()-create_time)` ：新闻热度点击率。
  - `two_hour_click_times/(now()-create_time)` ：新闻近期热度点击率。
  - `w1、w2、w3、w4` ：逻辑回归模型学习中各个属性的权重。

  获取新闻推荐结果之后，应用就可以将用户感兴趣的新闻推荐给用户了。

7.4.3. 黑名单监控告警

本文介绍如何通过向量分析实现黑名单监控告警功能。

1. 开通AnalyticDB服务。
   - 只有ECU类型为H8的集群支持向量功能，其他类型ECU不支持向量功能。
   - 只支持在普通表（实时表）中定义向量列。
2. 创建普通表之前，需要先创建表组，否则系统执行建表语句时将提示出错。

   相同表组下普通表的HASH分区数必须相同。

操作步骤

1. 连接分析型数据库MySQL版创建黑名单表。

   通过以下SQL创建黑名单表 `BLACK_LIST` 。

   ```
   CREATE TABLE black_list
   (
      id bigint,
      name varchar,
      image_url varchar,
      gender varchar,
      feature float[1024]
   COLPROPERTIES (DistanceMeasure='SquaredEuclidean',ExtractFrom='FACE_FEATURE_EXTRACT_V1(image_url)'),
      ANN INDEX feature_index (feature) Algorithm=Native_FLAT,
      PRIMARY KEY (id)
   )
   PARTITION BY HASH KEY (id)
   PARTITION NUM 2
   CLUSTERED BY (id)
   TABLEGROUP ads
   ```

   `feature float[1024] COLPROPERTIES (DistanceMeasure='SquaredEuclidean',ExtractFrom='FACE_FEATURE_EXTRACT_V1(image_url)')` 定义向量列，用于获取用户自定义函数和传入图片的URL，详细的参数解释请参见CREATE TABLE。

2. 插入数据。

   分析型数据库MySQL版提供两种插入数据的方法。

   - 调用用户自定义函数 `FACE_FEATURE_EXTRACT_V1()` 从图片中抽取人脸特征向量，然后将结果写入分析型数据库MySQL版。

     ```
     INSERT INTO black_list (id, name, image_url, gender, feature) VALUES (10, 'Zhang San', 'https://xxx/img.jpg', 'male',
     FACE_FEATURE_EXTRACT_V1 ('https://xxx/face_img.jpg'));
     ```

   - 根据输入图片的URL自动获取特征向量，然后插入数据。
     使用这种方法插入数据时，需要在建表时指定 `feature` 列及其特征，用于获取用户自定义函数和传入图片的URL，类似上述建表时定义的 `ExtractFrom='FACE_FEATURE_EXTRACT_V1(image_url)'` 。

     ```
     INSERT INTO black_list (id, name, image_url, gender) VALUES (10, 'Zhang San', 'https://xxx/face_img.jpg', 'male');
     ```

   关于插入数据请参见插入数据。

3. 查询数据。
   分析型数据库MySQL版支持SELECT查询中携带用户自定义函数传入图片URL。例如，检索人脸图片链接为 `https://xxx/face_img.jpg` 且性别为男性的记录。

```
SELECT id, name
FROM ann
  ( SELECT * FROM black_list WHERE gender='male',
   feature,
   FACE_FEATURE_EXTRACT_V1('https://xxx/face_img.jpg'),
   1,
   'DISTANCE_THRESHOLD=0.6'
  )
```

从向量表中检索图片数据时，需要使用OSS图片的URL，有以下两种方法获取OSS图片的URL。

- 直接将图片上传到OSS，获取访问URL。
  使用该方法时，需要提前将OSS Bucket授权给 `vdb_support` 账户，如何授权请参见使用Bucket Policy授权其他用户访问OSS资源。
- 使用OSS SDK生成包含签名的URL，详情请参见如何在URL中包含签名。
  包含签名的URL示例： `http://oss-example.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?OSSAccessKeyId=nz2pc56s936**91&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D`

7.5. 用户指南

7.5.1. 定义向量列（2.0版）

本文将通过具体示例，为您介绍如何通过CREATE TABLE定义向量列。

前提条件

- 只有ECU类型为H8的集群支持向量功能，其他类型ECU不支持向量功能。
- 只支持在普通表（实时表）中定义向量列。
- 创建普通表之前，需要先创建表组，否则系统执行建表语句时将提示出错。
- 相同表组下普通表的HASH分区数必须相同。

示例

在 `TEST_GROUP` 表组下创建一级分区表 `TEST_TABLE` ，通过 `feature float[4] COLPROPERTIES(DistanceMeasure=SquaredL2)` 定义向量列。

```
create table test_table (
rowkey varchar,
feature float[4] COLPROPERTIES(DistanceMeasure=SquaredL2),
seed_name varchar,cluster_label varchar,
ANN INDEX ecnn_index(feature) Algorithm=Graph_hnsw,
PRIMARY KEY (rowkey)
)
PARTITION BY HASH KEY (rowkey) PARTITION NUM 32
TABLEGROUP test_group
OPTIONS (UPDATETYPE='realtime')
```

在 `TEST_GROUP` 表组下创建二级分区表 `TEST_TABLE2` ，通过 `feature float[4] COLPROPERTIES(DistanceMeasure=SquaredL2)` 定义向量列。

```
create table test_table2 (
rowkey varchar,
feature float[4] COLPROPERTIES(DistanceMeasure=SquaredL2),
seed_name varchar,
cluster_label varchar,
dt long,ANN INDEX ecnn_index(feature) Algorithm=Graph_hnsw,
PRIMARY KEY (rowkey, dt )
)
PARTITION BY HASH KEY (rowkey) PARTITION NUM 32
SUBPARTITION BY LIST KEY (dt)
SUBPARTITION OPTIONS (available_partition_num = 4)
TABLEGROUP test_group
OPTIONS (UPDATETYPE='realtime')
```

参数

- `feature` ：向量列的名称，用户自定义。
- `float[4]` ：向量列的数据类型和向量的维数，用户自定义。
- `COLPROPERTIES` ：系统关键字，定义向量列在进行ANN查询和构建索引时采用的向量距离计算公式。

- `DistanceMeasure`：系统关键字，`DistanceMeasure`的值为向量距离计算公式，默认值为：
  - `int[]`：`Hammin`
  - `byte[]`、`short[]`或者`float[]`：`SquaredEuclidean`

> **重要**
> 向量距离计算公式定义后不支持更改

AnalyticDB for MySQL支持的向量距离计算公式如下表所示。

距离计算公式	计算公式	适用数据类型
SquaredEuclidean（简称SquaredL2）	(x1-y1)^2+(x2-y2)^2+...	byte[]、short[]或者float[]
Hamming	Integer.bitCount(x1^y1)+Integer.bitCount(x2^y2)+Integer.bitCount(x3^y3)+...	int[]
DotProduct	x1*y1+x2*y2+x3*y3+...	byte[]、short[]或者float[] 对数据的L2范数有如下要求： ▪ float[]的L2范数为1 ▪ byte[]的L2范数为127 ▪ short[]的L2范数为32767

- ANN：系统关键字。
- INDEX：系统关键字。
- ecnn_index：索引名，用户自定义。
- Algorithm：向量距离计算公式使用的算法。AnalyticDB for MySQL支持的向量距离计算公式算法如下表所示。

算法	适用场景	适用数据
Native_FLAT（SSE线性计算）	适用于单表数据量小于10万条、向量维度为256左右的小数据量场景。	int[]、short[]、byte[]、float[]
GRAPH_HNSW	适用于单表数据量在百万级别到千万级别之间，对向量维度敏感的中等规模数据量场景。	int[]、short[]、byte[]、float[]
VG_PQ	适用于单表数量在千万级别以上，向量维度高的大规模、超大规模数据量场景。	short[]、byte[]、float[]

向量算法（IndexAlgorithm）支持的向量距离计算公式（DistanceMeasure）如下表所示。

算法	SquaredEuclidean	DotProduct	Hamming
Native_FLAT	支持	支持	支持
GRAPH_HNSW	支持	支持	支持
VG_PQ	支持	不支持	不支持

7.5.2. 插入数据到向量列（2.0版）

数据识别方式

您可以通过明文或者编码方式向向量列写入数据或者查询向量列数据。

> *注意：必须使用单引号将数据引用起来，数据的维数与 **定义向量列** 时指定的维度相同。*

- 明文：`'[1,2,3,4]'` 或者 `'{1.0,2.0,3.0,4.0}'`
- Base64编码：`'MTIzNA=='`

  以下使用Base64解码完成byte[]数据类型和float[]数据类型的解码。

```
public static String getBase64(float array[]) {
        ByteBuffer bb = ByteBuffer.allocate(array.length * 4);
        bb.order(ByteOrder.LITTLE_ENDIAN);
        for(int i = 0; i < array.length; i++) {
            bb.putFloat(array[i]);
        }
        return Base64.getEncoder().encodeToString(bb.array());
    }

    public static float[] decodeBase64(String str) {
        byte[] buf = Base64.getDecoder().decode(str);
        ByteBuffer bb = ByteBuffer.wrap(buf);
        bb.order(ByteOrder.LITTLE_ENDIAN);
        float[] ret = new float[buf.length / 4];
        for(int i = 0; i < ret.length; i++) {
            ret[i] = bb.getFloat();
        }
        return ret;
    }
```

插入数据

您可以通过INSERT语句向向量表中插入数据，通过DELETE删除数据。例如，通过以下SQL向 `TEST_TABLE` 表中插入两条数据。

> *注意：向向量表中插入数据时，向量列的维数必须与 **定义向量列** 时指定的维度相同，否则系统将提示出错。*

```
insert into test_table values('rowkey1', '[0.5,0.6,0.3,0.1]','seednmae',  'label')
insert into test_table values('rowkey1', 'AAAAP5qZGT+amZk+zczMPQ==','seedname',  'label')
```

7.5.3. 查询向量数据（2.0版）

向向量表中插入数据后，您可以通过ANN查询的方式查询向量数据。

示例

- 以下SQL将从 `TEST_TABLE` 表中检索 `float_feature` ，返回与输入向量 `'[0.5,0.6,0.3,0.1]'` 最接近的前10条记录。 `ann_distance` 是输入向量和返回结果之间的真实相似度，相似度计算方法由 `float_feature` 列定义指定。

  ```
  select id, ann_distance from
  ann(test_table, float_feature, '[0.5,0.6,0.3,0.1]', 10);
  ```

- 返回向量计算距离小于0.2的记录。

  ```
  select id, ann_distance from
  ann(test_table, float_feature, '[0.5,0.6,0.3,0.1]', 10,'DISTANCE_THRESHOLD=0.2');
  ```

- 返回数据写入时间小于24小时且向量计算距离小于0.2的记录。

  ```
  select id, ann_distance from
  ann(test_table, float_feature, 'AAAAP5qZGT+amZk+zczMPQ==', 10,'DISTANCE_THRESHOLD=0.2,IGNORE_INC=true');
  ```

ANN查询支持的参数

参数（不区分大小写）	作用	使用方法
DISTANCE_THRESHOLD	`ann_distance` 的筛选阈值。	例如 `DISTANCE_THRESHOLD=0.5` ，系统根据 `DistanceMeasure` 自动判断并筛选符合条件的数据。
IGNORE_INC	数据写入流量非常大时，为保证检索响应时间，您可以选择不查询部分实时数据（一般指数据写入时间小于24小时的数据）。	`IGNORE_INC=true`
PREFER_INDEX	在同一个向量列上定义多个索引时，您可以使用 `PREFER_INDEX` 参数建议系统采用哪一个索引算法，也可以通过 `PREFER_INDEX` 参数强制不使用索引。	例如建议系统采用 `PREFER_INDEX=FAST_INDEX` 索引算法。 `PREFER_INDEX=NOINDEX` ，强制不使用索引。
DISTANCE_MEASURE	必选参数，设置ANN查询使用的 `DISTANCE_MEASURE` （向量距离计算公式）。	`DISTANCE_MEASURE` 参数的值可能是 `SquaredL2` 、 `Hamming` 或者 `DotProduct` ，必须与CREATE TABLE中 `DistanceMeasure` 设置的值相同。

向量分析碰撞

通过JOIN实现向量碰撞，以下SQL从 `TEST_TABLE` 表中检索向量 `DotProduct` 距离小于0.8的向量集合。

```
/*+ engine= mpp*/select a.feature,
b.feature,
vector_distance(a.feature, b.feature, 'DotProduct')
 from `test_table`  as a cross join `test_table`  as b
where vector_distance(a.feature, b.feature, 'DotProduct')< 0.8
```

8.2.0版SQL手册

8.1. 数据类型（2.0版）

varchar长度不得超过16KB，否则可能会出现字段为null。如果此列超过16KB又不能过滤掉，可以设置该列去掉索引或者设置为全文索引。去掉索引后建议该列不要在查询中进行筛选和计算。

1. `boolean` 布尔类型，值只能是 `0`或`1` 。取值0的逻辑意义为 `假` ，取值1的逻辑意义为 `真` ，存储字节数1比特位。
2. `tinyint` 微整数类型，取值范围 `-128`到`127` ，存储字节数1字节。
3. `smallint` 整数类型，取值范围 `-32768`到`32767` ，存储字节数2字节。
4. `int` 整数类型，取值范围 `-2147483648`到`2147483647` ，存储字节数4字节。
5. `bigint` 大整数类型，取值范围 `-9223372036854775808`到`9223372036854775807` ，存储字节数8字节。
6. `float` 单精度浮点数，取值范围 `-3.402823466E+38`到`-1.175494351E-38` ， `0` ， `1.175494351E-38`到 `3.402823466E+38` ，IEEE标准，存储字节数4字节。
7. `double` 双精度浮点数，取值范围 `-1.7976931348623157E+308`到`-2.2250738585072014E-308` ， `0` ， `2.2250738585072014E-308` 到 `1.7976931348623157E+308` ，IEEE标准，存储字节数8字节。
8. `decimal(m,d)` ， `m` 是数值的最大精度，取值范围为 `1~1000` ； `d` 是小数点右侧数字的位数，要求 `d≤m` 。
9. `varchar` 变长字符串类型。
0. `date` 日期类型，取值范围 `'1000-01-01'` 到 `'9999-12-31'` ，支持的数据格式为 `'YYYY-MM-DD'` ，存储字节数为4字节。
1. `time` 时间类型，取值范围 `'00:00:00'` 到 `'23:59:59'` ，支持的数据格式为 `'HH:MM:SS'` ，存储字节数为4字节。
2. `timestamp` 时间戳类型，取值范围 `'1970-01-01 00:00:01' UTC` 到 `'2038-01-19 03:14:07' UTC` ，支持的数据格式为 `'YYYY-MM-DD HH:MM:SS'` ，存储字节数为4字节。

与MySQL数据类型对比

分析型数据库MySQL版数据类型	MySQL版数据类型	差异
boolean	bool、boolean	一致。
tinyint	tinyint	一致。
smallint	smallint	一致。
int	int、integer	一致。
bigint	bigint	一致。
float	float[(m,d)]	分析型数据库MySQL版不支持自定义m和d，MySQL支持。
double	double[(m,d)]	分析型数据库MySQL版不支持自定义m和d，MySQL支持。
decimal	decimal	分析型数据库MySQL版支持的最大精度为1000，MySQL支持的最大精度为65。
varchar	varchar	分析型数据库MySQL版有长度限制。
date	date	一致。
time	time	取值范围不同。
timestamp	timestamp	分析型数据库MySQL版精确到秒，MySQL支持自定义精度。

注意事项

varchar长度不得超过16KB，否则可能会出现字段为null。如果此列超过16KB又不能过滤掉，可以设置该列去掉索引和全文索引。

8.2. CREATE TABLEGROUP（2.0版）

AnalyticDB for MySQL `CREATE TABLEGROUP` 用于创建普通表组。

语法

```
create tablegroup tablegroup_name;
```

参数

- tablegroup_name为表组名，表组名应满足以下要求：
  - 表组名以字母开头，字母或数字结尾（不能以下划线结尾）；可包含字母、数字或下划线（_），长度不超过64个字符。
  - 表组名中不能包含双下划线（__）。
  - 同一个数据库中，表组名唯一。
- 维度表组由系统创建，表组命名为dbname_dimension_group。

示例

创建TRADE表组，存放交易表。示例如下：

```
create tablegroup trade;
```

相关文章

- 通过DMS界面创建表组
- 名词解释

8.3. CREATE TABLE（2.0版）

AnalyticDB for MySQL `CREATE TABLE` 用于创建普通表。

创建维度表

语法

```
CREATE DIMENSION TABLE table_name (
    column_name column_type [NOT NULL][DEFAULT 'default'][COMMENT 'comment'][, …],
    [FULLTEXT INDEX index_name (column_name),]
    primary key (column_name[, …])
)
```

参数

- CREATE DIMENSION TABLE：`DIMENSION` 关键字，表示创建的表是维度表。
- column_type：列类型，分析型数据库MySQL版支持的列类型，请参见数据类型。
- NOT NULL：可选项，列属性。定义了 `NOT NULL` 的列不允许值为 `NULL` 。未定义此属性时，默认值为 `NULL` 。
- DEFAULT ‘default’：可选项，列属性。
- COMMENT ‘comment’：可选项，列属性。
- FULLTEXT INDEX：可选项，指定列建立全文索引，索引名字为 `index_name` 。 `column_name` 的类型支持Varchar或clob，建议为Varchar。
- primary key：指定主键，可以为联合主键。

示例

```
CREATE DIMENSION TABLE goods (
    goods_id bigint  comment  '货物编号',
    price double comment '价格',
    class bigint comment '类别',
    name varchar comment '名称',
    update_time  timestamp comment '上新时间',
    FULLTEXT INDEX name_fulltext (name),
    primary key (goods_id)
)
```

创建普通表

语法

```
CREATE TABLE table_name (
    column_name column_type [NOT NULL][DEFAULT 'default'][COMMENT 'comment'][, …],
    [FULLTEXT INDEX index_name (col_name),]
    primary key (column_name[, … ])
)
PARTITION BY HASH KEY(column_name)
[PARTITION NUM N]
TABLEGROUP tablegroup_name
```

参数

- column_type：列类型。分析型数据库MySQL版支持的列类型，请参见数据类型。
- NOT NULL：可选项，列属性。定义了 `NOT NULL` 的列不允许值为 `NULL` ，默认为 `NULL` 。
- DEFAULT ‘default’：可选项，列属性。
- COMMENT ‘comment’：可选项，列属性。
- FULLTEXT INDEX：可选项，指定列建立全文索引，索引名字为 `index_name` 。 `column_name` 的类型支持Varchar或clob，建议为Varchar。
- primary key：指定主键，可以为联合主键，普通表的主键中必须含有分区列。
- HASH KEY：指定分区列，详情请参见一级分区的规划和设计。
- PARTITION NUM：可选项，一级分区数，默认为128。
- TABLEGROUP：表所归属的表组。

> **重要** 创建普通表之前，需要先创建表组，否则系统执行建表语句时将提示出错。如何创建表组，请参见创建表组。

示例

在 `table_group` 表组下，新建CUSTOMER表。

```
CREATE TABLE customer (
   customer_id bigint NOT NULL COMMENT '顾客ID',
   customer_name varchar NOT NULL COMMENT '顾客姓名',
   phone_num bigint NOT NULL COMMENT '电话',
   city_name varchar NOT NULL COMMENT '所属城市',
   sex int NOT NULL COMMENT '性别',
   id_number varchar NOT NULL COMMENT '身份证号码',
   home_address varchar NOT NULL COMMENT '家庭住址',
   office_address varchar NOT NULL COMMENT '办公地址',
   age int NOT NULL COMMENT '年龄',
   login_time timestamp NOT NULL COMMENT '登录时间',
   FULLTEXT INDEX address_fulltext (home_address),
   PRIMARY KEY (customer_id，phone_num)
)
PARTITION BY HASH KEY (customer_id)
TABLEGROUP table_group
COMMENT '客户信息表';
```

相关文章

- 保留字（2.0版）
- 列的最佳实践
- 一级分区的规划和设计
- 数据类型
- 全文检索最佳实践

8.4. DROP TABLEGROUP（2.0版）

AnalyticDB for MySQL `DROP TABLEGROUP` 用于删除普通表组。

语法

```
drop tablegroup tablegroup_name;
```

参数

- tablegroup_name：为表组名称。
- 维度表组不能删除，删除普通表组之前需要先清空其中的表。

示例

删除TRADE表组。

```
drop tablegroup trade;
```

8.5. DROP TABLE（2.0版）

AnalyticDB for MySQL `DROP TABLE` 用于删除普通表。

语法

```
drop table table_name;
```

参数

table_name：表名。

示例

删除CUSTOMER表。

```
drop table customer;
```

8.6. SELECT（2.0版）

8.6.1. SELECT语法（2.0版）

AnalyticDB for MySQL SELECT用于返回表和用户定义的函数中的行，具体语法如下

```
[ WITH with_subquery [, ...] ]
SELECT
[ [ ALL | DISTINCT ] |  select_expr [ AS output_name ] [, ...] ]
[ FROM table_reference [, ...] ]
[ WHERE condition ]
[ GROUP BY expression [, ...] ]
[ HAVING condition ]
[ { UNION | ALL | INTERSECT | EXCEPT | MINUS } query ]
[ ORDER BY expression
[ ASC | DESC ]
[ LIMIT { number | ALL } ]
```

本节包含以下几个部分：

- WITH子句
- MATCH...AGAINST
- GROUP BY
- HAVING
- UNION、INTERSECT、 EXCEPT
- ORDER BY
- LIMIT
- JOIN
- 子查询

8.6.2. GROUP BY（2.0版）

GROUP BY子句标识查询的分组列。必须在查询中使用标准计算聚合函数（SUM、AVG和COUNT）时声明分组列。

```
GROUP BY expression [, ...]
```

- GROUP BY中列或表达式列表必须匹配查询列表中的非聚合表达式的列。
  例如，考虑以下简单查询中，查询列表包含两个聚合表达式。第一个聚合表达式使用SUM函数，第二个聚合表达式使用COUNT函数。必须将其余两个列（LISTID 和 EVENTID）声明为分组列。

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by listid, eventid
order by 3, 4, 2, 1
limit 5;

listid | eventid | revenue | numtix
-------+---------+---------+--------
  89397|      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

- GROUP BY子句中的表达式也可以使用序号来引用选择列表。
  例如，上一个示例的缩略形式

```
 select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by 1,2
order by 3, 4, 2, 1
limit 5;

listid | eventid | revenue | numtix
-------+---------+---------+--------
89397  |      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

8.6.3. HAVING（2.0版）

HAVING子句与聚合函数以及GROUP BY子句共同使用，用来去掉不满足条件的分组。在分组和聚合计算完成后，HAVING对分组进行过滤。

```
[ HAVING condition ]
```

以下示例查询CUSTOMER表，并进行分组，查出账户余额大于指定值的记录：

```
SELECT count(*), mktsegment, nationkey,
       CAST(sum(acctbal) AS bigint) AS totalbal
FROM customer
GROUP BY mktsegment, nationkey
HAVING sum(acctbal) > 5700000
ORDER BY totalbal DESC;

 _col0 | mktsegment | nationkey | totalbal
-------+------------+-----------+----------
  1272 | AUTOMOBILE |        19 |  5856939
  1253 | FURNITURE  |        14 |  5794887
  1248 | FURNITURE  |         9 |  5784628
  1243 | FURNITURE  |        12 |  5757371
  1231 | HOUSEHOLD  |         3 |  5753216
  1251 | MACHINERY  |         2 |  5719140
  1247 | FURNITURE  |         8 |  5701952
(7 rows)
```

> **说明**
> - HAVING子句条件中引用的列必须为分组列或引用了聚合函数结果的列。
> - HAVING子句必须与聚合函数以及GROUP BY子句一起使用，用来对GROUP BY的分组进行过滤，去掉不满足条件的分组。

8.6.4. UNION、INTERSECT和EXCEPT（2.0版）

UNION、INTERSECT和EXCEPT用于将多个查询语句的结果集进行组合，从而形成一个最终结果。

```
query
{ UNION [ ALL ] | INTERSECT | EXCEPT | MINUS }
query
```

参数

- UNION：返回两个查询表达式的集合运算。
- UNION ALL：ALL关键字用于保留由UNION生成的任何重复行。
- INTERSECT：返回派生自两个查询表达式的行的集合运算。返回结果中将丢弃未同时由两个表达式返回的行。
- EXCEPT| MINUS：返回派生自两个查询表达式之一的行的集合运算。返回的结果行必须存在于第一个结果表而不存在于第二个结果表中。MINUS和EXCEPT完全同义。

计算顺序

- UNION和EXCEPT集合运算符是左关联的，如果未指定圆括号来影响优先顺序，则将以从左到右的顺序来计算这些集合运算符的组合。例如，在以下查询中，首先计算T1

和T2的UNION，然后对UNION结果执行EXCEPT操作：

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

- 在同一个查询中使用运算符组合时，INTERSECT运算符优先于UNION和EXCEPT运算符。例如，以下查询将计算T2和T3的交集，然后计算得到的结果与T1 的并集：

```
select * from t1
union
select * from t2
intersect
select * from t3
order by c1;
```

- 通过添加圆括号，可以强制实施不同的计算顺序。在以下示例中，将T1和T2的并集结果与T3执行交集运算，并且查询可能会生成不同的结果。

```
(select * from t1
union
select * from t2)
intersect
(select * from t3)
order by c1;
```

8.6.5. ORDER BY（2.0版）

ORDER BY子句用于对查询结果集进行排序，ORDER BY中每个表达式由列名或列序号（从1开始）组成。

```
[ ORDER BY expression
[ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[ LIMIT { count | ALL } ]
```

8.6.6. LIMIT（2.0版）

LIMIT分句用于限制最终结果集的行数。LIMIT ALL和略去LIMIT分句的结果一样。以下示例为查询一个大表，LIMIT子句限制它只输出5行。

```
SELECT orderdate FROM orders LIMIT 5;

o_orderdate
-------------
 1996-04-14
 1992-01-15
 1995-02-01
 1995-11-12
 1992-04-26
(5 rows)
```

8.6.7. JOIN（2.0版）

以下查询是FROM子句中的两个子查询的内部联接，此查询查找不同类别活动（音乐会和演出）的已售门票数和未售门票数。

```
select catgroup1, sold, unsold
from
(select catgroup, sum(qtysold) as sold
from category c, event e, sales s
where c.catid = e.catid and e.eventid = s.eventid
group by catgroup) as a(catgroup1, sold)
join
(select catgroup, sum(numtickets)-sum(qtysold) as unsold
from category c, event e, sales s, listing l
where c.catid = e.catid and e.eventid = s.eventid
and s.listid = l.listid
group by catgroup) as b(catgroup2, unsold)

on a.catgroup1 = b.catgroup2
order by 1;
```

8.6.8. 子查询（2.0版）

以下示例在WHERE子句中包含一个表子查询，该子查询生成多个行。在本示例中，行只包含一列，但表子查询可以包含多个列和行，就像任何其他表一样。

以下查询查找门票销量排名前10位的卖家。

```
select firstname, lastname, cityname, max(qtysold) as maxsold
from users join sales on users.userid=sales.sellerid
where users.city not in(select venuecity from venue)
group by firstname, lastname, city
order by maxsold desc, city desc
limit 10;

firstname  | lastname |      cityname  | maxsold
-----------+----------+----------------+---------
Noah       | Guerrero | Worcester      |       8
Isadora    | Moss     | Winooski       |       8
Kieran     | Harrison | Westminster    |       8
Heidi      | Davis    | Warwick        |       8
Sara       | Anthony  | Waco           |       8
Bree       | Buck     | Valdez         |       8
Evangeline | Sampson  | Trenton        |       8
Kendall    | Keith    | Stillwater     |       8
Bertha     | Bishop   | Stevens Point  |       8
Patricia   | Anderson | South Portland |       8
(10 rows)
```

8.6.9. 全文检索（2.0版）

AnalyticDB for MySQL支持通过SQL语法进行全文检索，本文介绍如何在已经创建全文索引的 `col_name` 列中检索关键词。

语法

```
SELECT
[ [ ALL | DISTINCT ] |  select_expr [ AS output_name ] [, ...] ]
[ FROM table_reference [, ...] ]
[ WHERE match(column_name[, …]) against('words') ]
```

> **说明**
> - `match(column_name[, …])` 已经在建表语句中创建了全文索引的列的名字，可以填写一个列名也可以填写多个列名。比如 `match(body)` 表示只在 `body` 这一列中进行检索， `match(title, body)` 表示在 `title` 、 `body` 两列中分别进行检索。
> - `against('words')` 进行检索的关键词，比如 `against（'浙江省杭州市'）` 表示要检索 `浙江省杭州市` 。

示例

对 `articles_test` 表中 `author` 、 `title` 和 `body` 三列分别创建全文索引，建表语句如下。

```
CREATE TABLE articles_test (
  id bigint COMMENT '',
  author varchar COMMENT '',
  title varchar COMMENT '',
  body varchar COMMENT '',
  comment varchar COMMENT '',
  create_time timestamp COMMENT '',
  FULLTEXT INDEX author_fulltext (author),
  FULLTEXT INDEX body_fulltext (body),
  FULLTEXT INDEX title_fulltext (title),
  PRIMARY KEY (id)
)
PARTITION BY HASH KEY(id)
TABLEGROUP test_group
COMMENT '';
```

使用如下命令在 `articles_test` 表中插入数据。

```
INSERT INTO articles_test (id, author, title, body, comment, create_time) VALUES(0, '张三', '浙江省杭州市春天美景推荐', '浙江省杭州市拥有美丽的西湖，是全国有名的风景胜地，春天的景色尤其迷人', '好地方', '2018-02-01 10:10:13');
INSERT INTO articles_test (id, author, title, body, comment, create_time) VALUES(1, '张三', '江西九江夏天美丽景色', '庐山风景区坐落于江西九江，北濒长江，南傍鄱阳湖，素有"匡庐奇秀甲天下山"之美称。', '好地方', '2018-09-13 10:10:13');
INSERT INTO articles_test (id, author, title, body, comment, create_time) VALUES(2, '李四', 'x省秋天美丽景色', '那山，那水，那满眼的绿，那不同于红砖白墙的色彩在华山烨烨生辉', '好地方', '2018-09-30 10:10:13');
INSERT INTO articles_test (id, author, title, body, comment, create_time) VALUES(3, '王五', '《建国大业》简介', '中华人民共和国位于亚洲东部，太平洋西岸，是工人阶级领导的、以工农联盟为基础的人民民主专政的社会主义国家，成立于1949年（己丑年）10月1日', '国富民强', '2018-10-18 10:10:13');
INSERT INTO articles_test (id, author, title, body, comment, create_time) VALUES(4, '王五', '地理信息大全', '共和国山地、高原和丘陵约占陆地面积的67%，盆地和平原约占陆地面积的33%。山脉多呈东西和东北一西南走向', '国富民强', '2018-09-30 23:10:13');
INSERT INTO articles_test (id, author, title, body, comment, create_time) VALUES(5, '李四', '浙江杭州秋天哪里风景最美', '杭州一年中最美的季节不是春天吗？这个我不辩驳，也承认杭州的春天确实很美，我也很喜欢春天这个季节的杭州城。但是，和火红般的秋季比起来，我还是更喜欢秋天的杭州城。', '好地方', '2018-02-01 10:10:13');
INSERT INTO articles_test (id, author, title, body, comment, create_time) VALUES(9, '李四', '浙江杭州千岛湖', '梅峰岛是千岛湖风景区登高观湖揽胜的最佳处，素有"不上梅峰观群岛，不识千岛真面目"', '好地方', '2018-02-01 10:10:13');
```

在 `body` 列中检索包含 `浙江省杭州市` 的数据，语句如下：

```
SELECT * FROM articles_test WHERE MATCH(body) AGAINST('浙江省杭州市');
```

返回结果如下：

```
+----+-------+------------------------+------------------------------------------------------------------------------------------------------
--------------------------------------------------+---------+-------------------+
|id  |author |title                   |body
|comment  |create_time         |
+----+-------+------------------------+------------------------------------------------------------------------------------------------------
--------------------------------------------------+---------+-------------------+
|5   |李四   |浙江杭州秋天哪里风景最美  |杭州一年中最美的季节不是春天吗？这个我不辩驳，也承认杭州的春天确实很美，我也很喜欢春天这个季节的杭州城。但是，和火红般的秋季比
起来，我还是更喜欢秋天的杭州城。 |好地方   |2018-02-01 10:10:13|
|0   |张三   |浙江省杭州市春天美景推荐  |浙江省杭州市拥有美丽的西湖，是全国有名的风景胜地，春天的景色尤其迷人
|好地方    |2018-02-01 10:10:13|
```

在 `title` 和 `body` 列中检索包含 `春天` 的数据，语句如下：

```
SELECT * FROM articles_test WHERE MATCH(title, body) AGAINST('春天');
```

返回结果如下：

```
+----+-------+------------------------+------------------------------------------------------------------------------------------------------
--------------------------------------------------+---------+-------------------+
|id  |author |title                   |body
|comment  |create_time         |
+----+-------+------------------------+------------------------------------------------------------------------------------------------------
--------------------------------------------------+---------+-------------------+
|5   |李四   |浙江杭州秋天哪里风景最美  |杭州一年中最美的季节不是春天吗？这个我不辩驳，也承认杭州的春天确实很美，我也很喜欢春天这个季节的杭州城。但是，和火红般的秋季比
起来，我还是更喜欢秋天的杭州城。 |好地方   |2018-02-01 10:10:13|
|0   |张三   |浙江省杭州市春天美景推荐  |浙江省杭州市拥有美丽的西湖，是全国有名的风景胜地，春天的景色尤其迷人
|好地方    |2018-02-01 10:10:13|
```

注意事项

- 定义了全文检索的列，仅支持 `match() against()` 操作，不支持 `=` 、 `!=` 、 `between` 、 `is null` 、 `is not null` 以及 `like` 等操作符。
- 检索语法支持使用特殊字符 `+、-、&、|、!、( )、{ }、[ ]、^、"、~、*、?、:、\、/` ，但需要对特殊字符进行转义处理。

  例如，需要检索包含 `春天/美景` 的数据，错误写法为 `match(title) against('春天 / 美景' )` ，正确写法为 `match(title) against('春天 \\/ 美景')` 。
- 设置了全文索引列的表，不能作为 `left join` 的右表。建议将 `left join` 改为 `right join` 或者将右表改为左表。
- 全文检索条件 `match() against()` 不能放在子查询外作为子查询后的过滤条件，即如下查询是不支持的：

```
SELECT * FROM (
     SELECT id, substr(text, 4) AS text
      FROM tbl
  ) A
  WHERE match(A.text) against('浙江省杭州市');
```

8.7. CTE（2.0版）

本文介绍CTE的使用方法。

CTE用于WITH子句定义一个子查询关系，可供SELECT查询引用。

以下两个查询是等价的：

```
SELECT a, b
FROM (SELECT a, MAX(b) AS b FROM t GROUP BY a) AS x;
```

```
WITH x AS (SELECT a, MAX(b) AS b FROM t GROUP BY a)
SELECT a, b FROM x;
```

WITH子句同样适用于多子查询：

```
WITH
t1 AS (SELECT a, MAX(b) AS b FROM x GROUP BY a),
t2 AS (SELECT a, AVG(d) AS d FROM y GROUP BY a)
SELECT t1.*, t2.*
FROM t1 JOIN t2 ON t1.a = t2.a;
```

WITH子句中定义的关系可以互相连接：

```
WITH
x AS (SELECT a FROM t),
y AS (SELECT a AS b FROM x),
z AS (SELECT b AS c FROM y)
SELECT c FROM z;
```

- CTE后面必须直接跟使用CTE的SQL语句（如SELECT、INSERT、UPDATE等），否则CTE将失效。
- CTE后面也可以跟其他的CTE，但只能使用一个WITH，多个CTE中间用逗号（,）分隔。
- CTE语句里暂时不支持分页。

8.8. INSERT（2.0版）

语法

```
INSERT [IGNORE]
    INTO table_name
    [( column_name [, …] )]
    [VALUES]
    [(value_list[, …])]
    [query];
```

参数

- IGNORE：可选参数，若系统中已经有相同主键的记录，新记录将会被丢弃掉。
- column_name：可选参数，列名。
- query：通过定义任何查询，将一行或多行插入到表中，查询生成的所有行都将插入到表中。

示例

- 在CUSTOMER表中插入一条数据：

```
INSERT INTO customer(customer_id,customer_name,phone_num,city_name,sex,id_number,home_address,office_address,age,login_time)
values
(002367,'杨过','13900001234','杭州',0,'987300','西湖','转塘云栖小镇',23,'2018-03-02 10:00:00');
```

- 在CUSTOMER表中插入多条数据：

```
INSERT INTO customer(customer_id,customer_name,phone_num,city_name,sex,id_number,home_address,office_address,age,login_time)
values
(002367,'李四','13900001111','杭州',0,'987300','西湖','转塘云栖小镇',23,'2018-03-02 10:00:00'),(002368,'张三','13900002222','杭州',0,'987300','西湖','转塘云栖小镇',28,'2018-08-01 11:00:00')，(002369,'王五','13900003333','杭州',1,'987300','西湖','转塘云栖小镇',35,'2018-09-12 08:11:00');
```

- 在CUSTOMER表中插入多条数据，可以省略列名：

```
INSERT INTO
customer values(002367,'李四','13900001111','杭州',0,'987300','西湖','转塘云栖小镇',23,'2018-03-02 10:00:00'),(002368,'张三','13900002222','杭州',0,'987300','西湖','转塘云栖小镇',28,'2018-08-01 11:00:00')，(002369,'王五','13900003333','杭州',1,'987300','西湖','转塘云栖小镇',35,'2018-09-12 08:11:00');
```

- INSERT query示例请参见INSERT SELECT FROM（2.0版）。

注意事项

- 如果不指定列名，则要插入的值必须依照CREATE TABLE语句中声明的顺序。
- 目前INSERT语句支持的函数有：CURDATE、SYSDATE、LOCALTIME/LOCALTIMESTAMP/NOW。

8.9. DELETE（2.0版）

`DELETE` 用于删除表中的记录。

语法

```
DELETE FROM table_name
[ WHERE condition ]
```

示例

- 删除CUSTOMER表中name为张三的数据：

```
DELETE FROM customer WHERE name='张三';
```

- 删除CUSTOMER表中所有的行：

```
DELETE FROM customer WHERE 1=1;
```

注意事项

- 删除全表数据且表数据量非常大时会造成严重性能问题。一般表记录数超过10万行，建议通过删表重建方式替代删除全表数据；
- DELETE暂时不支持带表的别名；
- DELETE WHERE条件中暂时不支持子查询和函数；
- 二级分区表的DELETE语句必须包含二级分区条件。删除TRADE表2月份的数据（biz_date为二级分区列）

```
DELETE FROM trade where biz_date>=20180201 and biz_date<=20180228;
```

- 频繁使用DELETE操作也会造成性能问题，强烈建议WHERE条件中带上主键，性能会达到最优；

8.10. ALTER（2.0版）

增加列

表创建好之后目前支持增加列。

语法

```
ALTER TABLE table_name ADD column_name data_type;
```

示例

商品表GOODS中增加一列num，类型为bigint。

```
ALTER TABLE goods ADD num bigint;
```

修改二级分区数

最大二级分区数可以在建表后进行修改。

语法

```
ALTER TABLE table_name subpartition_available_partition_num = N;
```

参数

N为新的二级分区数。

示例

TRADE表以天为二级分区单位，增量数据保存时间由30天改为60天。

```
ALTER TABLE trade subpartition_available_partition_num = 60;
```

相关文章

二级分区表

8.11. INSERT SELECT FROM（2.0版）

如果用户数据在其他表中已经存在，AnalyticDB MySQL 2.0支持通过查询方式实现数据复制。

语法

```
INSERT INTO table_name
[( column_name [, …] )]
query;
```

参数

column_name：列名，如果只需要从源表插入数据到目标表的部分列，确保SELECT子句查询的列与INSERT子句提供的列相匹配（顺序、数据类型）。

示例

- 提供列名的方式，从ORDER表中复制某几列数据到新订购表NEW_ORDER中。

```
INSERT INTO new_order (customer_id, order_id, order_time, order_amount, order_type, address, city, order_season)
SELECT customer_id, order_id, order_time, order_amount,order_type, address, city, order_season FROM order
WHERE order.order_season = '201804';
```

- 不提供列名的方式，从ORDER表中复制所有列数据到新订购表NEW_ORDER中。

```
INSERT INTO new_order
SELECT customer_id, order_id, order_time, order_amount, order_type, address, city, order_season  FROM order
WHERE order.order_season = '201807';
```

8.12. DUMP TO OSS（2.0版）

本文介绍何如将AnalyticDB MySQL 2.0的数据导出到对象存储（OSS）。

语法

```
/*+ dump-col-del=[,],
    dump-row-del=[\n],
    dump-oss-accesskey-id=ACCESS_KEY_ID,
    dump-oss-accesskey-secret=ACCESS_KEY_SECRET,
    engine=MPP,
    [return-dump-record-count=TRUE,]
    dump-header=[DUMP DATA [OVERWRITE] INTO 'oss://endpoint/bucket_name/filename']
*/ sql
```

参数

- dump-col-del：指定列分符，可选参数。
- dump-row-del：指定行分隔符，可选参数。
- dump-oss-accesskey-id：有OSS写某个object权限的access_key_id，必选参数。
- dump-oss-accesskey-secret：有OSS写某个object权限的access_key_secret，必选参数。
- engine=MPP：计算引擎，必选参数。
- return-dump-record-count=TRUE：显示行数，可选参数。
- dump-header：必选参数。
- overwrite：可选参数。当指定OVERWRITE时，会对原有文件进行覆盖写，不指定时会对原有文件进行追加写。
- endpoint：OSS内网的Endpoint，请参见公共云下OSS Region和Endpoint对照表，必选参数。无论同地域还是跨地域，Endpoint一定是内网Endpoint（有些地域不支持跨域访问）。

示例

将 `cx_test3` 表中年龄大于18岁客户的姓名和手机号导出到testBucketName下的CUSTOMER文件中（覆盖写）。数据库与OSS同地域，都在上海地域。

```
/*+ dump-oss-accesskey-id=gfCrBlzhaVz****,
    dump-oss-accesskey-secret=pWMAPt15gU8IBox9bj9rpjGU****,
    engine=MPP,
    dump-header=[DUMP DATA OVERWRITE INTO 'oss://oss-cn-shanghai-internal.aliyuncs.com/<testBucketName>/customer.csv']
*/
SELECT customer_id,customer_name,phone_num
FROM cx_test3
WHERE age >18;
```

注意事项

如果导出OSS没有数据且无任何报错，请检查语法是否有问题。语法中 `return-dump-record-count=TRUE` 和 `OVERWRITE` 外面的方括号表示可选项，如果需要加上这两个参数必须把外面的方括号去掉，具体写法参考如上例子。

8.13. DUMP TO MAXCOMPUTE（2.0版）

本文介绍何如将AnalyticDB MySQL 2.0的数据导出到MaxCompute中。

语法

```
/*+
    engine=MPP,
    [return-dump-record-count=TRUE,]
    dump-header=[DUMP DATA [OVERWRITE] INTO 'odps://project_name/table_name']
*/ sql
```

参数

- `engine=MPP` ：计算引擎，必选参数。
- `return-dump-record-count=TRUE` ：显示导出的数据行数，可选参数。
- `dump-header` ：必选参数。

授权

进行数据导出操作时，您所使用的账号和分析型数据库MySQL版公共账号 `garuda_data@aliyun.com` 和 `garuda_build@aliyun.com` 需要拥有MaxCompute表的Describe，Select，Alter，Update和Drop权限。

以下SQL为 `garuda_data@aliyun.com` 和 `garuda_build@aliyun.com` 账号授权，可以参照以下SQL为用户账号授权。

> **说明** 如果用户使用的账号是RAM用户，RAM用户格式为： `ram$account_name:subaccount_name` 。更多账号授权信息，请参见为阿里云账号/RAM用户授权。

```
USE project_name;
ADD USER ALIYUN$garuda_data@aliyun.com;
GRANT createInstance ON project project_name TO USER ALIYUN$garuda_data@aliyun.com;
GRANT Describe,Select,alter,update,drop ON TABLE table_name TO USER ALIYUN$garuda_data@aliyun.com;
```

```
USE project_name;
ADD USER ALIYUN$garuda_build@aliyun.com;
GRANT createInstance ON project project_name TO USER ALIYUN$garuda_build@aliyun.com;
GRANT Describe,Select,alter,update,drop ON TABLE table_name TO USER ALIYUN$garuda_build@aliyun.com;
```

示例

- 将CUSTOMER表中年龄大于18岁的客户的姓名和手机号导出到MaxCompute的trade project下的CUSTOMER中。

```
/*+
    engine=MPP,
    return-dump-record-count=TRUE,
    dump-header=[DUMP DATA OVERWRITE INTO 'odps://trade/customer']
*/
    SELECT customer_id,customer_name,phone_num
    FROM customer
    WHERE age >18;
```

- 将CUSTOMER表中年龄大于18的客户信息导入MaxCompute的trade project下的分区表CUSTOMER中，其中 `dt` 和 `nation` 为MaxCompute目标表的分区字段。

```
/*+
    engine=MPP,
    return-dump-record-count=TRUE,
    dump-header=[DUMP DATA OVERWRITE INTO 'odps://trade/customer/dt=20090627/nation=china']
*/
    SELECT customer_id,customer_name,phone_num
    FROM customer
    WHERE age >18;
```

8.14. SHOW（2.0版）

查询用户的数据库列表。指定 `EXTRA` 参数，输出关于数据库的更多信息，例如创建者ID、数据库连接信息等。

SHOW DATABASES

查询用户的数据库列表。指定 `EXTRA` 参数，输出关于数据库的更多信息，例如创建者ID、数据库连接信息等。

```
SHOW DATABASES [EXTRA];
```

SHOW TABLEGROUPS

查询用户当前数据库下的表组列表。

```
SHOW TABLEGROUPS [IN db_name];
```

SHOW TABLES

查询用户当前数据库（或者表组）下的表的列表。

```
SHOW TABLES [IN db_name.tablegroup_name];
```

SHOW COLUMNS

查询指定表的列信息。

```
SHOW COLUMNS IN table_name;
```

SHOW CREATE TABLE

查询指定表的DDL。

```
SHOW CREATE TABLE table_name;
```

SHOW ALL CREATE TABLE

查询当前数据库中所有表的DDL。

```
SHOW ALL CREATE TABLE db_name;
```

SHOW PROCESSLIST MPP

查询当前正在运行的MPP任务。指定 `/*+cross-frontnode=true*/` 时，查询当前数据库实例所有正在运行的MPP任务；不指定 `/*+cross-frontnode=true*/` 时，仅查询当前连接的FRONT NODE节点实例运行的MPP任务。

```
[/*+cross-frontnode=true*/] SHOW PROCESSLIST MPP
```

8.15. GRANT（2.0版）

GRANT用于为阿里云账号或者RAM子账号授予一定的数据库权限。

语法

```
GRANT privilege_type [(column_name[, …])]
ON privilege_level
TO user[, …]
```

参数

- `privilege_type`：权限类型，请参见ACL权限体系。
- `column_name[, …]`：可选参数，`privilege_type` 为 `select` 时，`column_name` 为具体的列名，即针对具体列授予 `SELECT` 权限。
- `privilege_level`：被授权对象层级。
  - `database_name.*`：数据库级别的权限。
  - `table_name`：表级别的权限。
- `user`：阿里云账号或者RAM子账号。
  - 阿里云账号的账号格式为 `ALIYUN$account_name`，其中 `ALIYUN$` 为阿里云账号前缀，标识该账号为阿里云账号；`account_name` 为阿里云账号的账号名，例如 `ALIYUN$doc_test`。
  - RAM子账号的账号格式为 `RAM$account_name:subaccount_name`，其中 `RAM$` 为RAM子账号前缀，标识该账号为RAM子账号；`account_name` 为阿里云账号名；`subaccount_name` 为RAM子账号的账号名。例如 `RAM$doc_test:account1`。

示例

- 以下示例为分析型数据库MySQL版账号授予CUSTOMER表的all权限。

```
  GRANT all ON  TO 'ALIYUN$analyticdb_support';
```

- 以下示例为分析型数据库MySQL版账号授予CUSTOMER表的 `DESCRIBE` 和 `SELECT` 权限。

```
GRANT describe,select ON customer TO 'ALIYUN$analyticdb_support';
```

- 以下示例为分析型数据库MySQL版账号授予CUSTOMER表的 `customer_id` 和 `sex` 列的 `SELECT` 权限：

```
GRANT select (customer_id, sex) ON customer TO 'ALIYUN$analyticdb_support';
```

- 以下示例将数据库 `ads_database` 的 `SELECT`、`INSERT` 权限授予阿里云账号 `ALIYUN$doc_test`。

```
  GRANT SELECT on ads_database.* to 'ALIYUN$doc_test';
  GRANT INSERT on ads_database.* to 'ALIYUN$doc_test';
```

- 以下示例中阿里云账号 `ALIYUN$doc_test` 将数据库 `ads_database` 的 `SELECT` 权限授予RAM子账号 `RAM$doc_test:account1`。

```
  GRANT SELECT on ads_database.* to 'RAM$doc_test:account1';
```

8.16. REVOKE（2.0版）

`REVOKE` 用于为阿里云账号或者RAM子账号回收一定的数据库权限。

语法

```
REVOKE privilege_type [(column_name[, …])]
ON privilege_level
FROM user[, …]
```

参数

- `privilege_type`：权限类型，详情参见ACL权限体系表格的**操作**栏。
- `column_name[, …]`：可选参数，当 `privilege_type` 为 `select` 时候，可以指定具体列，回收指定列的权限。
- `privilege_level`：被授权对象层级，如需回收数据库级别的权限，则写成 `database_name.*`；如需回收表级别权限，则写成 `table_name`。

示例

回收AnalyticDB for MySQL 2.0支持账号在CUSTOMER表中的 `describe` 和 `select` 权限。

```
REVOKE describe,select ON customer FROM 'ALIYUN$analyticdb_support';
```

8.17. SHOW GRANTS（2.0版）

`SHOW GRANTS` 用于查看当前或指定账号的权限。

语法

```
SHOW GRANTS
  [FOR user]
    [ON privilege_level]
```

参数

- `user` ：阿里云账号或者RAM子账号。
  - 阿里云账号的账号格式为 `ALIYUN$account_name` ，其中 `ALIYUN$` 为阿里云账号前缀，标识该账号为阿里云账号； `account_name` 为阿里云账号的账号名，例如 `ALIYUN$doc_test` 。
  - RAM子账号的账号格式为 `RAM$account_name:sub_account_name` ，其中 `RAM$` 为RAM子账号前缀，标识该账号为RAM子账号； `account_name` 为阿里云账号名； `sub_account_name` 为RAM子账号的账号名。例如 `RAM$doc_test:lj_test_sub` 。
- `privilege_level` ：被授权对象层级。
  - `database_name.*` ：数据库级别的权限。
  - `table_name` ：表级别的权限。

示例

- 查看当前登录账号下的数据库db_test的权限。

```
  show grants on *
  show grants on db_test.*
```

- 查看当前登录账号下的student表的权限。

```
  show grants on student
  show grants on db.student
```

- 查看阿里云账号sqream_test账号下数据库db_test的权限。

```
  show grants for 'ALIYUN$sqream_test' on db_test.*
```

- 查看阿里云账号sqream_test账号下student表的权限。

```
  show grants for 'ALIYUN$sqream_test' on student
  show grants for 'ALIYUN$sqream_test' on db.student
```

- 查看阿里云账号sqream_test中terraform子账号下数据库db_test的权限。

```
  show grants for 'RAM$sqream_test:terraform' on db_test.*
```

- 查看阿里云账号sqream_test中terraform子账号下student表的权限。

```
  show grants for 'RAM$sqream_test:terraform' on student
  show grants for 'RAM$sqream_test:terraform' on db.student
```

9.2.0版系统函数

9.1. 时间日期函数（2.0版）

9.1.1. 获取当前日期时间函数（2.0版）

- CURRENT_DATE/CURDATE：返回当前日期。
- CURRENT_TIME/CURTIME：返回当前时间。
- CURRENT_TIMESTAMP：返回当前时间戳。
- CURRENT_TIMEZONE：返回当前时区。
- FROM_UNIXTIME：返回unixtime时间戳。
- LOCALTIME：返回本地时间。
- LOCALTIMESTAMP：返回当前本地时间戳。
- NOW：返回当前时间戳。
- TO_UNIXTIME：转换为unix时间戳。
- UTC_DATE：返回utc日期。
- UTC_TIME：返回utc时间。
- UTC_TIMESTAMP：返回utc时间戳。

CURRENT_DATE/CURDATE

```
current_date()
```

- 命令说明：返回当前日期
- 返回值类型：DATE
- 示例：

```
select current_date()
+--------------------+
| _col0              |
+--------------------+
| 2018-12-27         |
```

CURRENT_TIME/CURTIME

```
current_time()
```

- 命令说明：返回当前时间
- 返回值类型：TIME
- 示例：

```
select current_time()
+--------------------+
| _col0              |
+--------------------+
| 13:52:56           |
```

CURRENT_TIMESTAMP

```
current_timestamp()
```

- 命令说明：返回当前时间戳
- 返回值类型：TIMESTAMP
- 示例：

```
select current_timestamp()
+------------------------+
| _col0                  |
+------------------------+
| 2018-12-27 13:54:42    |
```

CURRENT_TIMEZONE

```
current_timezone()
```

- 命令说明：返回当前时区
- 返回值类型：VARCHAR
- 示例：

```
select current_timezone()
+--------------------+
| _col0              |
+--------------------+
| Asia/Shanghai      |
```

FROM_UNIXTIME

```
from_unixtime()
```

- 命令说明：返回unixtime时间戳
- 返回值类型：TIMESTAMP
- 示例：

```
select from_unixtime(1522292553)
+-------------------------+
| _col0                   |
+-------------------------+
| 2018-03-29 11:02:33     |
```

LOCALTIME

```
localtime()
```

- 命令说明：返回本地时间
- 返回值类型：TIME
- 示例：

```
select localtime()
+--------------------+
| _col0              |
+--------------------+
| 13:58:17           |
```

LOCALTIMESTAMP

```
localtimestamp()
```

- 命令说明：返回当前本地时间戳
- 返回值类型：TIMESTAMP
- 示例：

```
select localtimestamp()
+----------------------+
| _col0                |
+----------------------+
| 2018-12-27 13:59:33  |
```

NOW

```
now()
```

- 命令说明：返回当前时间戳
- 返回值类型：TIMESTAMP
- 示例：

```
select now()
+-------------------------+
| _col0                   |
+-------------------------+
| 2018-12-27 14:01:00     |
```

TO_UNIXTIME

```
to_unixtime(now())
```

- 命令说明：转换为unix时间戳
- 返回值类型：DOUBLE
- 示例：

```
select to_unixtime(now())
+--------------------+
| _col0              |
+--------------------+
| 1545890562         |
```

UTC_DATE

```
UTC_DATE(now())
```

- 命令说明：返回utc日期
- 返回值类型：VARCHAR
- 示例：

```
select UTC_DATE()
+--------------------+
| _col0              |
+--------------------+
| 2018-12-27         |
```

UTC_TIME

```
UTC_TIME()
```

- 命令说明：返回utc时间
- 返回值类型：VARCHAR
- 示例：

```
select UTC_TIME()
+--------------------+
| _col0              |
+--------------------+
| 06:12:71           |
```

UTC_TIMESTAMP

```
utc_timestamp()
```

- 命令说明：返回utc时间戳
- 返回值类型：VARCHAR
- 示例：

```
select utc_timestamp()
+---------------------+
| _col0               |
+---------------------+
| 2018-12-27 06:12:18 |
```

9.1.2. 截取函数（2.0版）

DATE_TRUNC

```
DATE_TRUNC('unit', timestamp)
```

- 命令说明：根据指定的日期部分（如小时、周或月）截断时间戳表达式或文本。
- 参数说明：unit可以是SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER或者YEAR。
  - SECOND：返回秒数对应的时间
  - MINUTE：返回分钟数对应的时间
  - HOUR：返回当天整点时刻
  - DAY：返回当天零点
  - WEEK：返回周一零点
  - MONTH：返回当月第一天零点
  - QUARTER：返回本季度第一天零点
  - YEAR：返回本年度第一天零点
- 示例：

```
select date_trunc('second', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-12-27 14:08:53|
+---------------------+
select date_trunc('minute', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-12-27 14:15:00|
+---------------------+
select date_trunc('hour', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-12-27 14:00:00|
+---------------------+
select date_trunc('day', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-12-27 00:00:00|
+---------------------+
select date_trunc('week', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-12-24 00:00:00|
+---------------------+
select date_trunc('month', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-12-01 00:00:00|
+---------------------+
select date_trunc('quarter', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-10-01 00:00:00|
+---------------------+
select date_trunc('year', now());
+---------------------+
| _col0               |
+---------------------+
| 2018-01-01 00:00:00|
```

9.1.3. 间隔函数（2.0版）

- DATE_ADD：返回增加指定时间间隔后的日期。
- ADDDATE：返回增加指定单位（例如day，minute）时长后的日期时间。
- ADDTIME：返回两个日期时间相加后的结果。
- PERIOD_ADD：在原有年月（YYYYMM）基础上增加指定月份。
- PERIOD_DIFF：返回两个年月（YYYYMM）相减后的月份。
- DATE_DIFF：返回两个日期时间的间隔。
- SUBTIME：返回两个格式相同的日期时间参数相减后的结果。
- SUBDATE/DATE_SUB：返回减去指定天数后的日期时间。
- TIMESTAMPADD：返回增加指定单位（例如week）时长后的日期时间。
- TIMESTAMPDIFF：返回减去指定单位（例如week）时长后的日期时间。

DATE_ADD

```
date_add(unit, value, datetime)
```

- 命令说明：返回增加指定时间间隔后的日期。
- 参数说明：
  - unit：单位，类型为VARCHAR。
    涉及的时间单位类型有MILLISECOND、SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER、YEAR。
  - value：类型为BIGINT。
  - datetime：日期时间字段，类型为TIMESTAMP、TIME或DATE。

    > **重要** 参数unit必须在参数datetime日期时间字段精度范围内。例如，不支持：`date_add('second', 31, current_date())` 以及 `date_add('day', 12, current_time())` 。
- 示例：

```
SELECT DATE_ADD('DAY', 31, current_date()) as col1, current_date() as col2;
+------------------------+------------------------+
| col1                   | col2                   |
+------------------------+------------------------+
| 2019-01-08             | 2018-12-08             |
SELECT DATE_ADD('second', 31, current_time()) as col1, current_time() as col2;
+------------------------+------------------------+
| col1                   | col2                   |
+------------------------+------------------------+
| 16:37:55               | 16:37:24               |
SELECT DATE_ADD('second', 31, current_timestamp()) as col1, current_timestamp() as col2;
+------------------------+------------------------+
| col1                   | col2                   |
+------------------------+------------------------+
| 2018-12-08 16:39:02.0 | 2018-12-08 16:38:31.0 |
```

ADDDATE

```
ADDDATE(date, INTERVAL expr unit)
ADDDATE(timestamp,INTERVAL expr unit)
```

- 命令说明：返回添加指定时间间隔后的日期，支持TIMESTAMP、DATE两种类型。

  > **重要** 参数2应在参数1的精度范围内。

- 示例：

```
SELECT addDate(now(), interval '2' day) as col1;
+------------------------+
| col1                   |
+------------------------+
| 2018-12-10 16:45:42.3 |
SELECT addDate(now(), interval '2' minute) as col1;
+------------------------+
| col1                   |
+------------------------+
| 2018-12-08 16:48:20.4 |
```

ADDTIME

```
ADDTIME(expr1,expr2)
```

- 命令说明：返回expr1 + expr2的结果。
- 参数说明：参数为VARCHAR类型
- 示例：

```
SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002')  as col1;
+--------------------------+
| col1                     |
+--------------------------+
| 2008-01-02 01:01:00.999 |
SELECT ADDTIME('23:59:59.999999', '1:1:1.000002') as col1;
+--------------------------+
| col1                     |
+--------------------------+
| 01:01:00.999             |
```

PERIOD_ADD

```
PERIOD_ADD(YYYYMM, monthNum)
```

- 命令说明：在原有年月基础上新增monthNum月份。
- 参数说明：
  - YYYYMM：类型为bigint，例如201803。
  - monthNum：类型为bigint，月数，即在原有年月基础上新增monthNum月份。
- 示例：

```
SELECT PERIOD_ADD(200801,20) as col1;
+---------+
| col1    |
+---------+
| 200909 |
```

PERIOD_DIFF

```
PERIOD_DIFF(YYYYMM, YYYYMM)
```

- 命令说明：返回参数1减去参数2的月份。
- 参数说明：YYYYMM类型为BIGINT。
- 示例：

```
SELECT PERIOD_DIFF(200802,200703) as col1;
+------+
| col1 |
+------+
|   11 |
```

DATE_DIFF

```
date_diff(unit, datetime1, datetime2)
```

- 命令说明：返回减去指定时间间隔后的日期。
- 参数说明：
  - unit：类型为VARCHAR。
  - datetime1、datetime2：类型为DATE或者TIMESTAMP。

    > **重要** 注意若datetime1、datetime2均为DATE类型，unit最小只能到DAY。

- 示例：

```
SELECT date_diff('day', current_date(), date_add('day', 2, current_date())) as col1;
+------+
| col1 |
+------+
|    2 |
SELECT date_diff('minute', now(), date_add('day', 2, now())) as col1;
+------+
| col1 |
+------+
| 2880 |
```

SUBTIME

```
SUBTIME(expr1,expr2)
```

- 命令说明：返回expr1 - expr2后的时间，expr1与expr2格式相同。
- 示例：

```
SELECT SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002') as col1;
+--------------------------+
| col1                     |
+--------------------------+
| 2007-12-30 22:58:58.999 |
```

SUBDATE/DATE_SUB

```
SUBDATE(date,INTERVAL expr unit)
SUBDATE(expr,days)
```

- 命令说明：返回参数1减去指定天数（参数2）后的日期。
- 参数说明：类型为TIMESTAMP、DATE或者VARCHAR。
- 示例：

```
SELECT DATE_SUB(now(), INTERVAL 31 DAY) as col1;
+-----------------------+
| col1                  |
+-----------------------+
| 2018-11-07 16:56:36.0 |
SELECT SUBDATE(current_date(), INTERVAL 31 DAY) as col1;
+-----------------------+
| col1                  |
+-----------------------+
| 2018-11-07            |
SELECT SUBDATE(now(), 31) as col1;
+-----------------------+
| col1                  |
+-----------------------+
| 2018-11-07 16:58:17.0 |
```

TIMESTAMPADD

```
TIMESTAMPADD(unit,interval,datetime_expr)
```

- 命令说明：返回增加指定单位interval后的日期时间。
- 参数说明：类型为VARCHAR、DATE、TIMESTAMP或者TIME。
- 示例：

```
SELECT TIMESTAMPADD(WEEK,1,'2003-01-02') as col1;
+------------+
| col1       |
+------------+
| 2003-01-09 |
SELECT TIMESTAMPADD(MONTH,1,'2003-01-02') as col1;
+------------+
| col1       |
+------------+
| 2003-02-02 |
```

TIMESTAMPDIFF

```
TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)
```

- 命令说明：返回datetime_expr2 - datetime_expr1的整数差，单位为unit。
- 参数说明：类型为VARCHAR、DATE、TIMESTAMP或者TIME。
- 示例：

```
SELECT TIMESTAMPDIFF(SECOND,curtime(),'20:39:39') as col1;
+-------+
| col1  |
+-------+
| 13120 |
SELECT TIMESTAMPDIFF(DAY, now(), '2018-03-28 20:39:39') as col1;
+------+
| col1 |
+------+
| -254 |
SELECT TIMESTAMPDIFF(DAY, curdate(), '2018-03-28') as col1;
+------+
| col1 |
+------+
| -255 |
```

9.1.4. 时间格式及转换函数（2.0版）

- DATE_FORMAT：按照指定格式，将日期时间类型的数据转换为字符串类型。
- TO_CHAR：按照指定格式，显示日期时间。
- SEC_TO_TIME：返回秒数对应的时间。
- TIME_TO_SEC：将时间转换为当天的秒数。
- STR_TO_DATE：按照指定格式，将字符串类型的数据转换为DATE类型。
- TIME
- TIMESTAMP
- TO_DAYS：返回从0年开始以来的天数。

DATE_FORMAT

```
date_format(datetime, format)
```

- 命令说明：按照format指定的格式，将日期时间格式化成字符串。
- 参数说明：
  - datetime：类型为TIMESTAMP或者DATE。
  - format：格式字符串，格式转换符请参见格式符。
- 示例：

```
SELECT date_format(now(), '%Y-%m-%d %H:%i:%s') as col1, now() as col2;
+---------------------+---------------------+
| col1                | col2                |
+---------------------+---------------------+
| 2018-12-08 17:06:21 | 2018-12-08 17:06:21.0 |
SELECT date_format(current_date(), '%Y-%m-%d %H-%i-%s') as col1, current_date() as col2;
+---------------------+---------------------+
| col1                | col2                |
+---------------------+---------------------+
| 2018-12-08 00-00-00 | 2018-12-08          |
```

TO_CHAR

```
to_char(datetime, format)
```

- 命令说明：按照format指定的格式，将日期时间格式化成字符串。
- 参数说明：
  - datetime：类型为TIMESTAMP或者DATE。
  - format：格式字符串，格式转换符请参见格式符。
- 示例：

```
SELECT to_char(now(), '%Y-%m-%d %H:%i:%s') as col1;
+---------------------+
| col1                |
+---------------------+
| 2018-12-08 17:08:56 |
SELECT to_char(current_date(), '%Y-%m-%d %H:%i:%s') as col1;
+---------------------+
| col1                |
+---------------------+
| 2018-12-08 00:00:00 |
```

SEC_TO_TIME

```
SEC_TO_TIME(seconds)
```

- 命令说明：返回秒数对应的时间。
- 参数说明：seconds类型为BIGINT。
- 示例：

```
SELECT SEC_TO_TIME(11111) as col1;
+----------+
| col1     |
+----------+
| 03:05:11 |
```

TIME_TO_SEC

```
TIME_TO_SEC(time)
```

- 命令说明：将时间转换为当天的秒数。
- 参数说明：time类型为VARCHAR。
- 示例：

```
SELECT TIME_TO_SEC('22:23:00') as col1;
+--------+
| col1   |
+--------+
| 80580 |
```

STR_TO_DATE

```
STR_TO_DATE(str,format)
```

- 命令说明：按照format格式，将字符串类型的数据转换为DATE类型。
- 返回值类型：DATE。
- 示例：

```
SELECT STR_TO_DATE('01,5,2013','%d,%m,%Y') as col1;
+------------+
| col1       |
+------------+
| 2013-05-01 |
```

TIME

```
TIME(expr)
```

```
SELECT TIME('2003-12-31 01:02:03') as col1;
+----------+
| col1     |
+----------+
| 01:02:03 |
```

TIMESTAMP

```
TIMESTAMP(expr)
TIMESTAMP(expr1,expr2)
```

示例：

```
SELECT TIMESTAMP('2003-12-31 01:02:03') as col1;
+------------------------+
| col1                   |
+------------------------+
| 2003-12-31 01:02:03.0 |
SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:01') as col1;
+------------------------+
| col1                   |
+------------------------+
| 2004-01-01 00:00:01    |
```

TO_DAYS

```
TO_DAYS(date)
```

- 命令说明：返回从0年开始的天数。
- 参数说明：date类型为VARCHAR、DATE或者TIMESTAMP。
- 示例：

```
SELECT TO_DAYS('2007-10-07') as col1;
+--------+
| col1   |
+--------+
| 733321 |
SELECT TO_DAYS(curdate()) as col1;
+--------+
| col1   |
+--------+
| 737401 |
```

相关文档

格式符

9.1.5. 抽取函数（2.0版）

YEAR

- 命令说明：返回日期时间字段中的年。
- 示例：

```
SELECT year(now()),year(current_date());
+-------------------------+
| _col0          _col1    |
+-------------------------+
| 2018           2018     |
```

QUARTER

- 命令说明：返回年份日期的季度值，范围为1～4。
- 示例：

```
SELECT quarter(current_date()),quarter(now());
+-------------------------+
| _col0           _col1   |
+-------------------------+
| 4                4      |
```

MONTH

- 命令说明：返回日期时间中的月份。
- 示例：

```
SELECT month(now());
+--------------------+
| _col0              |
+--------------------+
| 12                 |
```

DAY/DAY_OF_MONTH

- 命令说明：返回指定日期是当月的第几天。
- 示例：

```
SELECT day(current_date()),day_of_month(now());
+--------------------+
| _col0     _col1    |
+--------------------+
| 27         27      |
```

DAY_OF_WEEK/DOW

- 命令说明：返回指定日期是星期几。
- 示例：

```
SELECT day_of_week('2018-11-02'),dow(current_date());
+--------------------+
| _col0      _col0   |
+--------------------+
| 6           5      |
```

DAY_OF_YEAR/DOY

- 命令说明：返回当前系统时间是本年的第几天。
- 示例：

```
SELECT day_of_year(now()),day_of_year(current_date());
+--------------------+
| _col0      _col1   |
+--------------------+
| 361         361    |
```

HOUR

- 命令说明：返回日期时间中的小时。
- 示例：

```
SELECT hour(now()),hour(current_time());
+--------------------+
| _col0     _col1    |
+--------------------+
| 15         15      |
```

MINUTE

- 命令说明：返回指定日期时间中的分钟。
- 示例：

```
SELECT minute('2018-1-2 10:23:10'),minute(now())
+--------------------+
| _col0    _col1     |
+--------------------+
| 23        19       |
```

SECOND

- 命令说明：返回日期时间中的秒。
- 示例：

```
SELECT second(current_time()),second('2018-2-3 10:10:10');
+--------------------+
| _col0   _col1      |
+--------------------+
| 51       10        |
```

9.1.6. 其他函数（2.0版）

UNIX_TIMESTAMP

```
UNIX_TIMESTAMP [date]
```

- 命令说明：返回1970-01-01 00:00:00 UTC以来的秒数。
- 示例：

```
SELECT UNIX_TIMESTAMP('2015-11-13 10:20:19.1') as col1
+--------------------+
| _col0              |
+--------------------+
| 1447381219         |
```

9.1.7. 格式符（2.0版）

格式符	说明
%a	Abbreviated weekday name (Sun .. Sat)
%b	Abbreviated month name (Jan .. Dec)
%c	Month, numeric (0 .. 12)
%d	Day of the month, numeric (00 .. 31)
%e	Day of the month, numeric (0 .. 31)
%f	Fraction of second (6 digits for printing: 000000 .. 999000; 1 - 9 digits for parsing: 0 .. 999999999)
%H	Hour (00 .. 23)
%h	Hour (01 .. 12)
%I	Hour (01 .. 12)
%i	Minutes, numeric (00 .. 59)
%j	Day of year (001 .. 366)
%k	Hour (0 .. 23)
%l	Hour (1 .. 12)
%M	Month name (January .. December)

%m	Month, numeric (00 .. 12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM)
%S	Seconds (00 .. 59)
%s	Seconds (00 .. 59)
%T	Time, 24-hour (hh:mm:ss)
%v	Week (01 .. 53), where Monday is the first day of the week; used with %x
%W	Weekday name (Sunday .. Saturday)
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal % character
%x	x, for any x not listed above

9.2. 字符串函数（2.0版）

9.2.1. 字符串函数（2.0版）

AnalyticDB for MySQL支持以下字符串函数。

- CHR：返回Unicode编码对应的字符串。
- CONCAT：连接字符串。
- GROUP_CONCAT：通常与group by一起使用，用于将group by产生的同一个分组中的值连接起来，返回一个字符串结果。
- LENGTH：返回字符串长度。
- LOWER：将字符串转换为小写。
- UPPER：将字符串转换为大写。
- LPAD：左拼接字符串。
- RPAD：右拼接字符串。
- TRIM/LTRIM/RTRIM：删除字符串前后所有的空格/前导空格/后置空格。
- REPLACE：删除字符串中的所有指定子串。
- REVERSE：将字符串逆序。
- SPLIT：将字符串按分隔符进行分隔，并返回数组。
- POSITION：返回字符串中子字符串的第一次出现的起始位置。
- CHAR：返回十进制数字对应的字符形成的字符串。
- HEX：返回十六进制字符串。
- INITCAP：将字符串首字符转换为大写。
- INSTR：返回指定子串首次（或者指定次数）出现在字符串中的位置。
- ASCII：返回字符或者字符串最左边字符对应的ASCII值。
- MID：作用同SUBSTR/SUBSTRING。
- REPEAT：返回字符串重复多次的字符串。
- STRPOS：返回字符串中子字符串的第一次出现的起始位置。
- SUBSTR/SUBSTRING：返回一个从指定位置开始的指定长度的子字符串。
- TRANSLATE将字符表达式值中，指定字符替换为新字符。

CHR

```
chr(n)
```

- 命令说明：返回Unicode编码为n的字符。
- 返回值类型：VARCHAR。
- 示例：

```
 SELECT Chr(65),  Chr(66);
```

返回值如下：

```
+-------+-------+
| _col0 | _col1 |
+-------+-------+
| A     | B     |
+-------+-------+
```

CONCAT

```
concat(string1, …, stringN)
```

- 命令说明：字符串连接操作，与标准SQL的连接运算符 `||` 功能相同。

- 返回值类型：VARCHAR。
- 示例：

```
SELECT Concat('aliyun', ', ', 'analyticdb')
```

返回值如下：

```
+--------------------------------------+
| _col0                                |
+--------------------------------------+
| aliyun, analyticdb                   |
+--------------------------------------+
```

GROUP_CONCAT

```
group_concat([distinct] 要连接的字段)
```

- 命令说明：group_concat函数通常与group by一起使用，用于将group by产生的同一个分组中的值连接起来，返回一个字符串结果。通过使用distinct可以排除重复值。
- 返回值类型：VARCHAR。
- 示例：

```
SELECT * FROM user
```

返回值如下：

```
+-------------+
| id |username|
+-------------+
| 4  |  lucy  |
| 3  |  Lucy  |
| 2  |  lily  |
| 1  |  lily  |
+-------------+
```

```
select group_concat(distinct username)from user GROUP BY 'username' ;
```

返回值如下：

```
+-----------------------+
|      _col0            |
+-----------------------+
|    Lucy,Lily          |
+-----------------------+
```

LENGTH

```
length(string)
```

- 命令说明：返回字符串string的长度。

> **说明**
> MySQL中一个中文字符length为3，分析型数据库MySQL版中为1。

- 返回值类型：BIGINT。
- 示例：

```
SELECT length('aliyun')
```

返回值如下：

```
+--------------------+
| _col0              |
+--------------------+
| 6                  |
+--------------------+
```

LOWER

```
lower(string)
```

- 命令说明：将字符串string中的字母转换为小写。
- 返回值类型：VARCHAR。
- 示例：

```
SELECT lower('Aliyun');
```

返回值如下：

```
+--------------------+
| _col0              |
+--------------------+
| aliyun             |
+--------------------+
```

UPPER

```
upper(string)
```

- 命令说明：将字符串string中的字母转换为大写。
- 返回值类型：VARCHAR。
- 示例：

```
SELECT upper('Aliyun');
```

返回值如下：

```
+-----------+
| _col0     |
+-----------+
| ALIYUN    |
+-----------+
```

LPAD

```
lpad(string, size, padstring)
```

- 命令说明：
  - 将字符串string左边拼接padstring直到长度达到size，并返回填充后的字符串。
  - 如果size比string长度小，则截断。size不能为负数，padstring非空。

> **说明**
> MySQL中一个中文字符length为3，分析型数据库MySQL版中为1。

- 返回值类型：VARCHAR。
- 示例：

```
 SELECT Lpad('Aliyun',9,'#');
```

返回值如下：

```
+-----------+
| _col0     |
+-----------+
| ###Aliyun |
+-----------+
```

RPAD

```
rpad(string, size, padstring)
```

- 命令说明：
  - 将字符串string右边拼接padstring直到长度达到size，并返回填充后的字符串。
  - 如果size比string长度小，则截断。size不能为负数，padstring非空。

> **说明**
> MySQL中一个中文字符length为3，分析型数据库MySQL版中为1。

- 返回值类型：VARCHAR。
- 示例：

```
SELECT rpad('Aliyun',9,'#');
```

返回值如下：

```
+--------------+
| _col0        |
+--------------+
| Aliyun###    |
+--------------+
```

TRIM/LTRIM/RTRIM

```
trim(string)
ltrim(string)
rtrim(string)
```

- 命令说明：

- `trim(string)`：删除字符串string前后所有的空格。
- `ltrim(string)`：删除字符串string所有前导空格。
- `rtrim(string)`：删除字符串string所有后置空格。

- 返回值类型：VARCHAR。
- 示例：

```
  SELECT Trim('   135 544    '), Ltrim('   135 544    '), Rtrim('   135 544    ');
```

返回值如下：

```
+---------+-------------+------------+
| _col0   | _col1       | _col2      |
+---------+-------------+------------+
| 135 544 | 135 544     |    135 544 |
+---------+-------------+------------+
```

REPLACE

```
replace(string, search)
replace(string, search, replace)
```

- 命令说明：
  - `replace(string, search)`：删除字符串string中的所有search子串。
  - `replace(string, search, replace)`：将字符串string中所有子串search替换为replace。
- 返回值类型：VARCHAR。
- 示例：

```
S  SELECT REPLACE('helloworld', 'world'),
         REPLACE('helloworld', 'world', ' World!!!'),
         REPLACE('helloworld', 'notFound', ' World');
```

返回值如下：

```
+-------+----------------+------------+
| _col0 | _col1          | _col2      |
+-------+----------------+------------+
| hello | hello World!!! | helloworld |
+-------+----------------+------------+
```

REVERSE

```
reverse(string)
```

- 命令说明：返回string逆序后的字符串。
- 返回值类型：VARCHAR。
- 示例：

```
 SELECT Reverse('123456'),  Reverse(''), Reverse(Cast (NULL AS VARCHAR));
```

返回值如下：

```
+--------+-------+-------+
| _col0  | _col1 | _col2 |
+--------+-------+-------+
| 654321 |       | NULL  |
+--------+-------+-------+
```

SPLIT

```
split(string, delimiter)
split_part(string, delimiter, index)
split_to_map(string, entryDelimiter, keyValueDelimiter)
```

- 命令说明：
  - `split(string, delimiter)`：将字符串string按分隔符delimiter进行分隔，并返回数组。
  - `split_part(string, delimiter, index)`：将字符串string按分隔符delimiter分隔，并返回分隔后数组下标为index的子串，index以1开头，如果大于字段数则返回null。
  - `split_to_map(string, entryDelimiter, keyValueDelimiter)`：通过entryDelimiter和keyValueDelimiter拆分字符串并返回map。entryDelimiter将字符串分解为key-value对，keyValueDelimiter将每对分隔成key、value。
- 返回值类型：VARCHAR或map<varchar, varchar>。
- 示例：

```
SELECT Split('1#2#3', '#'), Split('#1#2#3#', '#'),
          Split('123', '#');
```

返回值如下：

```
+-----------+---------------+-----------+
| _col0     | _col1         | _col2     |
+-----------+---------------+-----------+
| [1, 2, 3] | [, 1, 2, 3, ] | [123]     |
+-----------+---------------+-----------+
```

```
    SELECT Split_part('A#B#C', '#', 2),
           Split_part('A#B#C', '#', 4);
```

返回值如下：

```
+---------------------+--------------------------+
| _col0               | _col1                    |
+---------------------+--------------------------+
| B                   | NULL                     |
+---------------------+--------------------------+
```

```
    SELECT Split_to_map('k1:v1,k2:v2', ',', ':'),
           Split_to_map('', ',', ':');
```

返回值如下：

```
+---------------------+--------------------------+
| _col0               | _col1                    |
+---------------------+--------------------------+
| {k1=v1, k2=v2}      | {}                       |
+---------------------+--------------------------+
```

POSITION

```
position(substring IN string)
```

- 命令说明： 返回字符串中子字符串的第一次出现的起始位置，位置从1开始，如果未找到则返回0。
- 返回值类型：BIGINT。
- 示例：

```
  SELECT position('o' IN 'helloworld'),
         position('or' IN'helloworld'),
         position('x' IN 'helloworld');
```

返回值如下：

```
+-------+-------+--------+
| _col0 | _col1 | _col2  |
+-------+-------+--------+
|  5    |  7    | 0      |
+-------+-------+--------+
```

CHAR

```
char(N,N,..)
```

- 命令说明： 返回每个数字代表的字符组成的字符串。
- 返回值类型：VARCHAR。
- 示例：

```
  SELECT CHAR(97,110,97,108,121,116,105,99,100,98);
```

返回值如下：

```
+------------+
| _col0      |
+------------+
| analyticdb |
+------------+
```

HEX

```
hex(bigint)
```

- 命令说明：返回16进制字符串。
- 返回值类型：VARCHAR。
- 示例：

```
  SELECT Hex(10),
         Hex(1234);
```

返回值如下：

```
+-------+-------+
| _col0 | _col1 |
+-------+-------+
| a     | 4d2   |
+-------+-------+
```

INITCAP

```
initcap(string)
```

- 命令说明：输入字符串首字符转大写。
- 返回值类型：VARCHAR。
- 示例：

```
  SELECT Initcap('the soap'),
         Initcap('the,soap'),
         Initcap('thesoap');
```

返回值如下：

```
+----------+----------+---------+
| _col0    | _col1    | _col2   |
+----------+----------+---------+
| The Soap | The,Soap | Thesoap |
+----------+----------+---------+
```

INSTR

```
instr(string, substring)
instr(string, substring, position)
instr(string, substring, position, occurrence)
```

- 命令说明：

`instr(string, substring)`：返回string中匹配substring的第一个位置信息。

`instr(string, substring, position)`：从string字符的第position位置开始搜索，返回string中匹配substring的位置信息。

`instr(string, substring, position, occurrence)`：从string字符的第position位置开始搜索第occurrence次出现的位置，返回string中匹配substring的位置信息。

- 返回值类型：BIGINT。
- 示例：

```
  SELECT Instr('CORPORATE FLOOR', 'OR'),
         Instr('CORPORATE FLOOR', 'OR', 3),
         Instr('CORPORATE FLOOR', 'OR', 3, 2);
```

返回值如下：

```
+----------+----------+---------+
| _col0    | _col1    | _col2   |
+----------+----------+---------+
| 2        | 5        | 14      |
+----------+----------+---------+
```

ASCII

```
ASCII(string)
```

- 命令说明：返回字符或者字符串最左边字符对应的ASCII值。
- 返回值类型：BIGINT。
- 示例：

```
  SELECT Ascii('d'),
         Ascii('dx'),
         Ascii('1');
```

返回值如下：

```
+-------+-------+-------+
| _col0 | _col1 | _col2 |
+-------+-------+-------+
|   100 |   100 |    49 |
+-------+-------+-------+
```

MID

```
MID(string, start, length)
```

- 命令说明：作用同substr(string, start, length)。
- 返回值类型：VARCHAR。
- 示例：

```
  SELECT Mid('helloworld', 6, 3),
         Mid('helloworld', -6, 2);
```

返回值如下：

```
+-------+-------+
| _col0 | _col1 |
+-------+-------+
| wor   | ow    |
+-------+-------+
```

REPEAT

```
repeat(string,int)
```

- 命令说明：返回重复多次的字符串。
- 返回值类型：VARCHAR。
- 示例：

```
  SELECT Repeat('a', 3),
         Repeat('xyz', 2),
         Repeat(Cast(NULL AS VARCHAR), 2);
```

返回值如下：

```
+-------+--------+-------+
| _col0 | _col1  | _col2 |
+-------+--------+-------+
| aaa   | xyzxyz | NULL  |
+-------+--------+-------+
```

STRPOS

```
strpos(string, substring)
```

- 命令说明：返回字符串中子字符串的第一次出现的起始位置，位置从1开始 ，如果未找到则返回0。
- 返回值类型：BIGINT。
- 示例：

```
  SELECT Strpos('helloworld', 'o'),
         Strpos('helloworld', 'or'),
         Strpos('helloworld', 'x');
```

返回值如下：

```
+-------+-------+-------+
| _col0 | _col1 | _col2 |
+-------+-------+-------+
|     5 |     7 |     0 |
+-------+-------+-------+
```

SUBSTR/SUBSTRING

```
substr(string, start)
substr(string, start, length)
```

- 命令说明：
  - `substr(string, start)` ：返回从start位置开始到字符串结束的子串。位置从1开始，如果start为负数，则起始位置从字符串的末尾开始倒数。
  - `substr(string, start, length)` 返回从start位置开始长度为length的子串，位置从1开始。如果start为负数，则起始位置从字符串的末尾开始倒数。

> **说明**
> MySQL中一个中文字符的length为3，分析型数据库MySQL版中为1。

- 返回值类型：VARCHAR。
- 示例：

```
  SELECT Substr('helloworld', 6),
         Substr('helloworld', 6, 3),
         Substr('helloworld', -6),
         Substr('helloworld', -6, 2),
         Substring('helloworld', -6, 2);
```

返回值如下：

```
+-------+-------+--------+-------+-------+
| _col0 | _col1 | _col2  | _col3 | _col4 |
+-------+-------+--------+-------+-------+
| world | wor   | oworld | ow    | ow    |
+-------+-------+--------+-------+-------+
```

TRANSLATE

```
translate(expr, from_string, to_string)
```

- 命令说明：将expr字符串中，符合from_string的字符，替换为to_string。
- 返回值类型：VARCHAR。
- 示例：

```
  SELECT Translate('acbd', 'ab', 'AB'),
         Translate('acbdaa', 'ab', 'AB'),
         Translate('acbd', 'abc', 'A'),
         Translate('acbd', 'abc', '');
```

返回值如下：

```
+-------+--------+-------+-------+
| _col0 | _col1  | _col2 | _col3 |
+-------+--------+-------+-------+
| AcBd  | AcBdAA | Ad    |    d  |
+-------+--------+-------+-------+
```

9.2.2. 字符串运算符（2.0版）

运算符||

运算符 || 用于字符串连接，|| 左右都是字符串类型。

示例

```
SELECT customer_name,
       customer_name || 'name',
       'str1' || 'str2',
       'str' || Cast(100 AS VARCHAR)
FROM   t_fact_customers
WHERE  customer_id < 4;

+---------------+---------------+----------+--------+
| customer_name | _col1         | _col2    | _col3  |
+---------------+---------------+----------+--------+
| 王小二          | 王小二name      | str1str2 | str100 |
| 李春梅          | 李春梅name      | str1str2 | str100 |
| 张大山          | 张大山name      | str1str2 | str100 |
+---------------+---------------+----------+--------+
```

9.3. 数值函数（2.0版）

- ABS：绝对值函数
- MOD：求余
- ROUND：四舍五入
- SQRT：平方根
- CBRT：立方根
- E：自然对数
- LN：自然对数
- LOG：对数
- LOG2：以2为底的对数
- LOG10：以10为底的对数
- PI：返回pi
- POWER：指数函数
- RANDOM：随机函数
- RADIANS：角度转度
- DEGREES：弧度转度
- SIGN：符号函数
- CEILING/CEIL：向上取整
- FLOOR：向下取整
- TRUNCATE：截断函数
- COS：余弦
- COSH：双曲余弦
- ACOS：反余弦
- TAN：正切
- ATAN：反正切
- ATAN2：除法后的反正切值
- ASIN：反正弦
- UUID：返回uuid值
- TO_BASE：根据base，把数字转为字符串
- FROM_BASE：根据base，把字符串转为数字

ABS

```
abs(tinyint)
abs(smallint)
abs(int)
abs(bigint)
abs(double)
```

- 命令说明：求绝对值
- 返回值类型：LONG或DOUBLE
- 示例：

```
select abs(-9);
+-------------+
| _col0       |
+-------------+
|   9         |
```

MOD

- 命令说明：求余
- 示例：

```
SELECT mod(cast(4.5 as tinyint), 3);
+--------+
| _col0 |
+--------+
|     2 |
SELECT mod(cast(4.5 as smallint), 3);
+--------+
| _col0 |
+--------+
|     2 |
SELECT mod(cast(4.5 as int), 3);
+--------+
| _col0 |
+--------+
|     2 |
SELECT mod(cast(4.5 as bigint), 3);
+--------+
| _col0 |
+--------+
|     2 |
SELECT mod(cast(4.5 as double), 3);
+--------+
| _col0 |
+--------+
|   1.5 |
```

ROUND

```
round(double)
round(tinyint)
round(smallint)
round(int)
round(bigint)
```

- 命令说明：四舍五入
- 返回值类型：BIGINT或DOUBLE
- 示例：

```
SELECT round(cast(4.5 as tinyint), 3);
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as smallint), 3);
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as int), 3);
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as bigint), 3);
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as double), 3);
+--------+
| _col0 |
+--------+
|   4.5 |
SELECT round(cast(4.5 as tinyint));
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as smallint));
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as int));
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as bigint));
+--------+
| _col0 |
+--------+
|     5 |
SELECT round(cast(4.5 as double));
+--------+
| _col0 |
+--------+
|   5.0 |
```

SQRT

```
sqrt(double)
```

- 命令说明：求平方根
- 返回值类型：DOUBLE
- 示例：

```
select sqrt(4);
+--------+
| _col0 |
+--------+
|   2.0 |
```

CBRT

```
cbrt(double)
```

- 命令说明：求立方根
- 返回值类型：DOUBLE
- 示例：

```
select cbrt(8);
+--------+
| _col0 |
+--------+
|   2.0 |
```

E

```
e()
```

- 命令说明：求自然对数

- 返回值类型：DOUBLE
- 示例：

```
select e();
+-------------------+
| _col0             |
+-------------------+
| 2.718281828459045 |
```

LN

```
ln(double)
```

- 命令说明：求自然对数
- 返回值类型：DOUBLE
- 示例：

```
select ln(2.718281828459045);
+-------+
| _col0 |
+-------+
|   1.0 |
```

LOG

```
log(double)
```

- 命令说明：求对数
- 返回值类型：DOUBLE
- 示例：

```
select log(10,100);
+-------+
| _col0 |
+-------+
|   2.0 |
```

LOG2

```
log2(double)
```

- 命令说明：求以2为底的对数
- 返回值类型：DOUBLE
- 示例：

```
select log2(8);
+-------+
| _col0 |
+-------+
|   3.0 |
```

LOG10

```
log10(double)
```

- 命令说明：求以10位底的对数
- 返回值类型：DOUBLE
- 示例：

```
select log10(100);
+-------+
| _col0 |
+-------+
|   2.0 |
```

PI

```
pi()
```

- 命令说明：返回pi
- 返回值类型：DOUBLE
- 示例：

```
select pi();
+-------------------+
| _col0             |
+-------------------+
| 3.141592653589793 |
```

POWER

```
power(double, double)
```

- 命令说明：指数函数
- 返回值类型：DOUBLE
- 示例：

```
select power(1.2,3.4);
+--------------------+
| _col0              |
+--------------------+
| 1.858729691979481 |
```

RANDOM

```
random()
random(tinyint)
random(smallint)
random(int)
random(bigint)
```

- 命令说明：随机函数
- 返回值类型：BIGINT
- 示例：

```
select random();
+---------------------+
| _col0               |
+---------------------+
| 0.5709993917553757 |
select random(cast(3 as tinyint));
+---------------------+
| _col0               |
+---------------------+
|     2               |
select random(cast(3 as smallint));
+---------------------+
| _col0               |
+---------------------+
|     0               |
select random(cast(3 as int));
+---------------------+
| _col0               |
+---------------------+
|     1               |
select random(cast(3 as bigint));
+---------------------+
| _col0               |
+---------------------+
|     0               |
```

RADIANS

```
radians(double)
```

- 命令说明：角度转度
- 返回值类型：DOUBLE
- 示例：

```
select radians(60.0);
+---------------------+
| _col0               |
+---------------------+
| 1.0471975511965976 |
```

DEGREES

```
degrees(double)
```

- 命令说明：把弧度转化为度
- 返回值类型：DOUBLE
- 示例：

```
select degrees(1.3);
+--------------------+
| _col0              |
+--------------------+
| 74.48451336700703 |
```

SIGN

```
sign(bigint)
sign(int)
sign(smallint)
sign(tinyint)
sign(double)
```

- 命令说明：符号函数
- 返回值类型：BIGINT或DOUBLE
- 示例：

```
SELECT sign(cast(4.5 as bigint));
+--------+
| _col0 |
+--------+
|     1 |
SELECT sign(cast(4.5 as int));
+--------+
| _col0 |
+--------+
|     1 |
SELECT sign(cast(4.5 as smallint));
+--------+
| _col0 |
+--------+
|     1 |
SELECT sign(cast(4.5 as tinyint));
+--------+
| _col0 |
+--------+
|     1 |
SELECT sign(cast(4.5 as double));
+--------+
| _col0 |
+--------+
|   1.0 |
```

CEILING/CEIL

```
ceiling(tinyint)
ceiling(smallint)
ceiling(int)
ceiling(bitint)
ceiling(double)
```

- 命令说明：向上取整
- 返回值类型：LONG
- 示例：

```
select ceiling(2.3);
+--------+
| _col0 |
+--------+
|     3 |
```

FLOOR

```
floor(tinyint)
floor(smallint)
floor(int)
floor(bigint)
floor(double)
```

- 命令说明：向下取整
- 返回值类型：LONG
- 示例：

```
select floor(7.8);
+--------+
| _col0 |
+--------+
|     7 |
```

TRUNCATE

```
truncate(double)
```

- 命令说明：截断函数
- 返回值类型：LONG或DOUBLE
- 示例：

```
select truncate(2.3);
+--------+
| _col0 |
+--------+
|   2.0 |
select truncate(2.3456,2);
+---------+
| _col0  |
+---------+
| 2.3400 |
SELECT truncate(cast(4.5 as tinyint), 3);
+--------+
| _col0 |
+--------+
|     5 |
SELECT truncate(cast(4.5 as smallint), 3);
+--------+
| _col0 |
+--------+
|     5 |
SELECT truncate(cast(4.5 as int), 3);
+--------+
| _col0 |
+--------+
|     5 |
SELECT truncate(cast(4.5 as bigint), 3);
+--------+
| _col0 |
+--------+
|     5 |
```

COS

```
cos(double)
```

- 命令说明：求余弦
- 返回值类型：DOUBLE
- 示例：

```
select cos(1.3);
+---------------------+
| _col0               |
+---------------------+
| 0.26749882862458735 |
```

COSH

```
cosh(double)
```

- 命令说明：求双曲余弦
- 返回值类型：DOUBLE
- 示例：

```
select cosh(1.3);
+--------------------+
| _col0              |
+--------------------+
| 1.9709142303266285 |
```

ACOS

```
acos(double)
```

- 命令说明：求反余弦函数值

  > **说明** 小数点后第16位与MySQL不同。

- 返回值类型：DOUBLE
- 示例：

```
select acos(0.5);
+--------------------+
| _col0              |
+--------------------+
| 1.0471975511965979 |
```

TAN

```
tan(double)
```

- 命令说明：求正切

  > **说明** 与MySQL第16位精度不同。

- 返回值类型：DOUBLE
- 示例：

```
select tan(8);
+--------------------+
| _col0              |
+--------------------+
| -6.799711455220379 |
```

ATAN

```
atan(double)
```

- 命令说明：求反正切函数值

> 说明 MySQL16位精度，分析型数据库MySQL版14位精度。

- 返回值类型：DOUBLE
- 示例：

```
select atan(0.5);
+--------------------+
| _col0              |
+--------------------+
| 0.4636476090008061 |
```

ATAN2

```
atan2(double, double)
```

- 命令说明：参数1除参数2后的反正切值
- 返回值类型：DOUBLE
- 示例：

```
select atan2(0.5,0.3);
+--------------------+
| _col0              |
+--------------------+
| 1.0303768265243125 |
```

ASIN

```
asin(double)
```

- 命令说明：求反正弦函数值

> 说明 小数点后第16位与MySQL不同。

- 返回值类型：DOUBLE
- 示例：

```
select asin(0.5);
+--------------------+
| _col0              |
+--------------------+
| 0.5235987755982989 |
```

UUID

```
uuid()
```

- 命令说明：求uuid
- 返回值类型：VARCHAR
- 示例：

```
select uuid();
+--------------------------+
| _col0                    |
+--------------------------+
| M5be8ec4f80c98679b3badcb3 |
```

TO_BASE

```
to_base(bigint, bigint)
```

- 命令说明：根据base将数字转为字符串
- 返回值类型：VARCHAR
- 示例：

```
SELECT to_base(8,8);
+-------+
| _col0 |
+-------+
| 10    |
```

FROM_BASE

```
from_base(varchar, bigint)
```

- 命令说明：根据base把字符串转为数字
- 返回值类型：BIGINT
- 示例：

```
SELECT from_base('10',8);
+-------+
| _col0 |
+-------+
|     8 |
```

10.2.0版最佳实践

10.1. 全文检索最佳实践（2.0版）

AnalyticDB MySQL版2.0除了支持基本的全文检索方式，还支持以下方式：

- 按近似度排序
- 结果集过滤
- 多列查询
- 短语查询、精确匹配
- 逻辑操作符AND OR NOT
- 结构化、非结构化联合检索
- 高级SQL语法：结构化、非结构化GROUP BY, JOIN, UNION

按近似度排序

SQL语义规定在不带ORDER BY的情况下不会按照近似度排序。

如果需要将结果按照近似度从高到低排序，则加上 `ORDER BY DESC` ，示例如下。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(title) against ('浙江省杭州市')
ORDER BY match(title) against ('浙江省杭州市') DESC
LIMIT 100;
```

结果集过滤

全文检索会召回所有跟关键词近似的结果。在某些数据量很大的场景中，命中关键词的结果集可能也很大，但是往往只需要取出近似度较高的部分结果。

分析型数据库MySQL版提供了结果集过滤的功能，如下例子中where match() against() > 0.9 表示只取近似度排在前10%的结果，过滤掉了90%的低近似度结果。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(title) against ('浙江省杭州市') > 0.9
ORDER BY match(title) against ('浙江省杭州市') DESC
LIMIT 100;
```

多列查询

在title，body两列中同时检索”浙江省杭州市”关键字，返回近似度排在前10%的结果行，并且将结果行按照近似度逆序排列。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(title,body) against ('浙江省杭州市') > 0.9
ORDER BY match(title, body) against ('浙江省杭州市') DESC
LIMIT 100;
```

短语查询、精确匹配

默认情况下，分析型数据库MySQL版全文检索会对词语进行分词后，再进行检索。比如检索”中华人民共和国”会被分词为”中华”、”人民”、”共和国”三个词分别检索。但是某些特殊情况下，可能需要完全精确的短语匹配。比如只希望返回完全精确匹配”中华人民共和国”的结果，而不希望返回分词后的检索结果，则使用短语查询语法。语法格式为：在检索关键词上加双引号。

基本查询：分词后再检索。注意关键词的写法：没有双引号。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(body) against ('中华人民共和国') > 0.9
ORDER BY match(body) against ('中华人民共和国') DESC
LIMIT 100;
```

短语查询：精确匹配，不会做分词处理。注意关键词的写法，有双引号。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(body) against ('"中华人民共和国"') > 0.9
ORDER BY match(body) against ('"中华人民共和国"') DESC
LIMIT 100;
```

逻辑操作符 AND OR NOT

分析型数据库MySQL版支持利用逻辑操作符对结果进行控制。目前支持的逻辑操作符包括： `AND OR NOT` 。

其中AND表示要求操作符两边的关键词都必须出现。OR表示操作符两边的关键词出现一个即可。NOT表示右侧的关键词不能出现。

语法为： `<word1> AND/OR/NOT <word2>`

> 重要
> 1. AND/OR/NOT必须大写；
> 2. AND/OR/NOT两边必须有空格。

要求 “浙江省”和”杭州市”都必须出现。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(body) against ('浙江省 AND 杭州市') > 0.9
ORDER BY match(body) against ('浙江省 AND 杭州市') DESC
LIMIT 100;
```

要求“浙江省”，”杭州市”出现一个即可。效果等同于如“浙江省杭州市”，“浙江省杭州市”。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(body) against ('浙江省 OR 杭州市') > 0.9
ORDER BY match(body) against ('浙江省 OR 杭州市') DESC
LIMIT 100;
```

要求“浙江省”一定不能出现。

```
SELECT id, title, author, body
FROM articles_test
WHERE match(body) against ('杭州市 NOT 浙江省') > 0.9
ORDER BY match(body) against ('杭州市 NOT 浙江省') DESC
LIMIT 100;
```

结构化、非结构化联合检索

分析型数据库MySQL版支持对结构化列、非结构化列进行联合条件检索。

要求查询create_time在20180201-20180930之间的、body命中“浙江省杭州市”、且comment不为空的数据行。

```
SELECT id, title, author, body
FROM articles_test
WHERE create_time > '2018-02-01 00:00:00'
        AND create_time < '2018-09-30 00:00:00'
        AND match(body) against ('浙江省杭州市') > 0.9
        AND comment is not null
ORDER BY match(body) against ('浙江省杭州市') DESC
LIMIT 100;
```

要求在body中检索“浙江省杭州市”，且在author中检索“张三”。

```
SELECT id, author, body
FROM articles_test
WHERE match(body) against ('浙江省杭州市') > 0.9
        AND match(author) against('张三') > 0.9
ORDER BY match(body) against ('浙江省杭州市') DESC, match(author) against('张三') DESC
LIMIT 100;
```

高级SQL语法：结构化、非结构化融合GROUP BY, JOIN, UNION

分组

```
SELECT author, body, max(match(body) against ('浙江省杭州市')) as score
FROM articles_test
WHERE match(body) against ('浙江省杭州市') > 0.9
GROUP BY author, body
ORDER BY score DESC
LIMIT 100;
```

表连接：在文章表articles_test中检索“浙江省杭州市”， 在作者信息表author_info中检索“张三”，并且利用ID列将两表进行join。

```
CREATE TABLE author_info (
  id bigint COMMENT '',
  name varchar COMMENT '',
  addr varchar COMMENT '',
  FULLTEXT INDEX name_fulltext (name),
  FULLTEXT INDEX addr_fulltext (addr),
  PRIMARY KEY (id)
)
PARTITION BY HASH KEY(id)
PARTITION NUM 8
CLUSTERED BY (id)
TABLEGROUP test_group
OPTIONS (UPDATETYPE='realtime')
COMMENT '';

insert into author_info(id, name, addr) values(0, '张三', '湖南省张家界市武陵源区');

SELECT articles_test.id, articles_test.title, author_info.name, author_info.addr
FROM articles_test
    JOIN author_info
    ON articles_test.id = author_info.id
WHERE
    match (articles_test.body) against ('浙江省杭州市')
    AND match(author_info.name) against ('张三')
    AND articles_test.create_time between '2018-01-01 00:00:00' and '2018-09-30 00:00:00';
```

合并

```
SELECT id, title
FROM articles_test
WHERE
    match (articles_test.body) against ('浙江省杭州市')
UNION ALL
SELECT id, addr
FROM author_info
WHERE
    match(author_info.name) against ('张三');
```

10.2. 表结构设计及最佳实践（2.0版）

10.2.1. 一级分区的规划和设计（2.0版）

本章节介绍一级分区表的规划和设计，其中主要是一级分区列的选取。

AnalyticDB MySQL 2.0一级分区表采用HASH分区，可指定任意一列（不支持多列）作为分区列。HASH分区通过标准CRC算法计算出CRC值，并将CRC值与分区数作模计算，得出每条记录的分区号。空值的HASH值与字符串-1相同。以下按照优先级从高到底列出一级分区列的选择依据。

1. 选择值分布均匀的列作为分区列，请勿选择分布不均匀的列作为分区列。
2. 建议选择一级分区列的数据类型为tinyint、smallint、int、bigint或者varchar。
3. 如果是多个普通表（不包括维度表）JOIN，则选择参与JOIN的列作为分区列。

   如果是多列JOIN ，则根据查询重要程度或查询性能要求（例如：某SQL的查询频率特别高）来选择分区列，以保证基于分区列的JOIN具有较好的查询性能。
4. 选择GROUP BY或DISTINCT包含的列作为分区列。
5. 如果常用的SQL包含某列的等值或IN查询条件，则选择该列作为分区列。以下例子则选择 `id` 列作为分区列。

   ```
   select * from table where id=123 and …;
   select * from table where user in(1, 2,3);
   ```

数据倾斜带来的影响

如果一级分区列选择不合理会导致用户表数据倾斜，带来如SQL查询长尾、后台数据上线超时和单节点资源不足等诸多问题，对查询性能影响非常大也会给用户带来资源的浪费。

如何评估表数据是否倾斜

登录分析型数据库MySQL版控制台，单击**监控信息**>**DB表概览**，查看**表明细**部分，对于存在数据倾斜的表已经标红处理，需要用户及时更换分区列。名字没有标红的表，用户也可以单击表名来查看数据分布。

10.2.2. 列的最佳实践（2.0版）

优化原理

分析型数据库MySQL版处理数值类型的性能远好于处理字符串类型。原因在于：

- 数值类型定长，占用内存少，存储空间小。
- 数值类型计算更快，尤其是join时。
- 从内部索引机制上，字符串类型适合等值查询和范围查询情况，而时间，数值类型性能更高，建议用户尽可能使用数值类型，减少使用字符串类型。

方法

常见将字符串转换为数值类型方法：

- 包含字符前缀或后缀，例如E12345，E12346等。可以直接去掉前缀或者将前缀映射为数字。
- 该列只有少数几个值，例如国家名。可以对每个国家编码，每个国家对应一个唯一数字。
- 时间/日期类型数据，避免使用varchar字符类型存储，尽量使用date，timestamp或者int类型存储时间类型。
- 对于地理经度维度的使用，需要通过地理函数查询情况，数据类型采用double数据类型。

10.3. 典型业务最佳实践（2.0版）

10.3.1. 电子商务

本章节以简化的业务场景来介绍电子商务的最佳实践。

业务场景和需求

本例中简化的业务场景包含三个数据表：客户信息表、订单表、商品类型表（维度表）。

业务场景要求如下：

- 统计不同客户群体在不同时间段的数据。
- RT小于60秒。
- 每秒查询率QPS（Query Per Second）：偶尔查询。

表的逻辑设计

- **客户信息表（t_fact_customers）**

  数据量：约有五亿客户。

字段名	数据类型	说明
customer_id	varchar	客户编号
customer_name	varchar	客户姓名
phone_number	varchar	电话

address	varchar	地址
last_login_time	timestamp	最近登录时间
age	int	年龄

- **订单表（t_fact_orders）**

  数据量：每日订单量5000万，需要存储三年的数据，总数据量约550亿。

字段名	数据类型	说明
order_id	varchar	订单编号
customer_id	varchar	客户编号
goods_id	bigint	商品编号
numbers	bigint	数量
total_price	double	总价
order_time	timestamp	订单时间
order_date	bigint	订单日期（二级分区）

- **商品类型表（t_dim_goods）**

  商品总数约100万。

字段名	数据类型	说明
goods_id	bigint	商品编号
price	double	单价
class	bigint	类型
name	varchar	名称
update_time	timestamp	更新时间

表的物理设计

物理设计的目的是获得最佳性能，在进行表的物理设计前，必须了解表的查询SQL，才能决策表的最佳分片策略。假设本例中大部分查询都需要关联查询客户表和订单表，那么我们优先考虑以 `customer_id` 进行一级分区，每个节点的计算都是本地数据。

- **创建表组ads_demo**

  表组为 `ads_demo` ，两个副本，SQL查询超时为30秒。创建表组的SQL语句如下：

```
create tablegroup ads_demo;
```

- **创建表**
  - **创建t_fact_customers表**

    每个分区数据量 `390万=5亿/128` 个一级分区。创建 `t_fact_customers` 表的SQL语句如下：

```
CREATE TABLE t_fact_customers (
customer_id varchar COMMENT '',
customer_name varchar COMMENT '',
phone_number varchar COMMENT '',
address varchar COMMENT '',
last_login_time timestamp COMMENT '',
age int COMMENT '',
PRIMARY KEY (customer_id)
)
PARTITION BY HASH KEY (customer_id) PARTITION NUM 128
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

  - **创建t_fact_orders表**

    二级分区键 `order_date` 为bigint数据类型。根据数据量和存储总时间，按月（201712）间隔，每月一个二级分区，每个二级分区数据量为： `1193万 = 550亿/（128个一级分区）/(3年×12个月)` 。创建表的SQL语句如下：

```
CREATE TABLE t_fact_orders (
order_id varchar COMMENT '',
customer_id varchar COMMENT '',
goods_id bigint COMMENT '',
numbers bigint COMMENT '',
total_price double COMMENT '',
order_time timestamp COMMENT '',
order_date bigint COMMENT '',
PRIMARY KEY (order_id,customer_id,order_date)
)
PARTITION BY HASH KEY (customer_id) PARTITION NUM 128
SUBPARTITION BY LIST KEY (order_date)
SUBPARTITION OPTIONS (available_partition_num = 90)
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

- **创建t_dim_goods商品信息表（维度表）**

  建表语句如下：

```
CREATE DIMENSION TABLE t_dim_goods (
goods_id bigint comment '',
price double comment '',
class bigint comment '',
name VARCHAR comment '',
update_time TIMESTAMP comment '',
primary key (goods_id)
)
OPTIONS (UPDATETYPE='realtime');
```

查询SQL

按年龄段统计在一段时间内订单的客户数量、订单销售总额。

```
SELECT
case when cus.age <= 30 then '<20' when cus.age>20
and cus.age <= 30 then '20-30' when cus.age>30
and cus.age <= 40 then '30-40' else '>40' end as age_range,
COUNT(distinct cus.customer_id),
SUM(total_price)
FROM
t_fact_customers cus LEFT JOIN t_fact_orders ord ON cus.customer_id =ord.customer_id
WHERE ord.order_time >= '2017-10-01 00:00:00' AND ord.order_time < '
2017-11-01 00:00:00'
AND ord.order_date = 201710
GROUP BY age_range;
```

> **说明** 可以增加二级分区条件，进行二级分区裁剪，提高查询效率。

10.3.2. 物流快递

本章节以简化的业务场景来介绍物流快递的最佳实践。

- 本例中简化的业务场景：仅包括邮件状态表用于实时监控和更新邮件包裹状态，包裹从订单、收单、装车、投递等分为不同的状态。
- 业务要求：每个机构实时统计当前不同状态的邮件数量。
- 业务性能要求：
  - 实时统计新增数据，且要求一秒以内返回结果（即RT<1s）。
  - 查询并发量：QPS>300 。

表的逻辑设计—邮件状态表（t_fact_mail_status）

要求：

- 单日数据量约一亿记录。
- 有15万机构或快递员，每个机构的邮件量为100到3000不等。
- 存储30天的数据。

为满足高QPS，从设计上采用大宽表、冗余字段，并且避免表关联。

字段名	字段类型	说明
mail_id	varchar	邮件订单唯一码
scan_timestamp	timestamp	收件时间
biz_date	bigint	日期（二级分区）
dlv_person_name	varchar	投递员名称
rg_code	varchar	机构编码
rg_name	varchar	机构名称
receiver_name	varchar	收件人名
receiver_phone	varchar	收件人电话
receiver_addr 收件人地址	varchar	收件人地址
product_no 产品编码	varchar	产品编码
mag_no	varchar	包裹编号
op_1_timestamp	bigint	状态1操作时间
op_2_timestamp	bigint	状态2操作时间
op_3_timestamp	bigint	状态3操作时间
op_4_timestamp	bigint	状态4操作时间
op_5_timestamp	bigint	状态5操作时间

表的物理设计

创建邮件状态表 t_fact_mail_status ，需要规划表的一级分区、二级分区。业务查询主要按机构进行查询，QPS要求高。综上所述， 我们选择按rg_code进行一级分区。

```
CREATE TABLE t_fact_mail_status (
mail_id varchar COMMENT '',
scan_timestamp timestamp COMMENT '',
biz_date bigint COMMENT '',
rg_code varchar COMMENT '',
rg_name varchar COMMENT '',
dlv_person_name varchar COMMENT '',
receiver_name varchar COMMENT '',
receiver_phone varchar COMMENT '',
receiver_addr varchar COMMENT '',
product_no varchar COMMENT '',
mag_no varchar COMMENT '',
op_1_timestamp bigint COMMENT '',
op_2_timestamp bigint COMMENT '',
op_3_timestamp bigint COMMENT '',
op_4_timestamp bigint COMMENT '',
op_5_timestamp bigint COMMENT '',
PRIMARY KEY (mail_id,rg_code,biz_date)
)
PARTITION BY HASH KEY (rg_code) PARTITION NUM 128
SUBPARTITION BY LIST KEY (biz_date)
SUBPARTITION OPTIONS (available_partition_num = 30)
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

查询SQL

要求：

- 查询当天某机构号
- 不同状态的邮件数据量
- 核查需要操作的邮件量
- RT<1s 并且高QPS

综上所述，须利用分区裁剪，使单个查询在不同的节点上运行。SQL语句示例如下：

```
select sum(case when t.op_1_timestamp >= 20171128000000 and t.
op_1_timestamp <= 20171128235900
and(( t.op_2_timestamp is null or t.op_2_timestamp >
20171128235900 or t.op_2_timestamp < 20171128000000)
and( t.op_3_timestamp is null or t.op_3_timestamp >
20171128235900 or t.op_3_timestamp < 20171128000000)
and( t.op_4_timestamp is null or t.op_4_timestamp >
20171128235900 or t.op_4_timestamp < 20171128000000))
then 1 else 0 end ) as cn
from t_fact_mail_status t
where t.rg_code = '21111101' and t.biz_date = 20171128;
```

10.3.3. 交通

本章节以简化的业务场景来介绍车辆通行的最佳实践。

业务场景和需求

本例中简化的业务场景：记录主要交通路口（卡口）的车辆通行信息。要求：

- 每日有一亿的实时增量数据，数据需要存储五年。
- 可实时查询每个路口车辆通行情况。例如：某一时间段通过该路口的车辆总数、车辆详单信息等。
- 可按车辆拍照号码查询一段时间内的车辆的详单信息。

表的逻辑设计

车辆信息表（t_fact_vehicle_info）

字段名	字段类型	说明
uuid	varchar	唯一标识
access_time	varchar	接收时间
gate_number	varchar	卡口编码
device_num	varchar	-
pass_time	varchar	过车时间
vehicle_number	varchar	号牌号码
vehicle_num_type	varchar	号牌种类
speed	double	行驶速度
picture_url	varchar	-
pt_month	-	-

表的物理设计

- 按车辆拍照进行HASH一级分区，可均匀的将数据分布到所有分区上。
- 按卡口编号设置聚集列。
- 二级分区按月存储，存储五年（60个月的数据），设置二级分区数为61。

创建表组

根据业务场景和要求，创建以下表组和表：

```
create tablegroup app_group;
```

创建t_fact_vehicle_info表

```
CREATE TABLE t_fact_vehicle_info (
uuid varchar NOT NULL COMMENT '唯一标识',
access_time varchar DEFAULT '0' COMMENT '接收时间',
gate_number varchar DEFAULT '0' COMMENT '卡口编码',
device_num varchar DEFAULT '0' COMMENT '',
pass_time varchar DEFAULT '0' COMMENT '过车时间',
vehicle_number varchar DEFAULT '' COMMENT '号牌号码',
vehicle_num_type varchar DEFAULT '' COMMENT '号牌种类',
speed double DEFAULT 0 COMMENT '行驶速度',
picture_url varchar DEFAULT '' COMMENT '',
pt_month bigint COMMENT '',
PRIMARY KEY (uuid,vehicle_number,pt_month)
)
PARTITION BY HASH KEY (vehicle_number) PARTITION NUM 256
SUBPARTITION BY LIST KEY (pt_month)
SUBPARTITION OPTIONS (available_partition_num = 61)
CLUSTERED BY (gate_number)
TABLEGROUP app_group
OPTIONS (UPDATETYPE='realtime')
COMMENT '过车信息';
```

查询SQL

查询某车牌号码某一段时间内的详单数据：

```
select * from t_fact_vehicle_info
where vehicle_number='xxxxx'
and pass_time between '2017-10-10 09:00:00'
and '2018-03-01 10:00:00'
and pt_month between 20171010 and 20180301;
```

查询30分钟内某一路口通过车辆总数信息：

```
select count(*) from t_fact_vehicle_info
where gate_number='xxxxx'
and pass_time between '2018-03-01 09:00:00' and '2018-03-01 09:30:00'
and pt_month =20180301;
```

11.2.0版常见问题

11.1. 产品和业务限制

限制项	描述	例外申请方式
购买分析型数据库MySQL版的限制	账户余额大于等于500元现金	请联系技术支持
开通分析型数据库MySQL版的用户限制	用户需实名认证	无
可创建的最大分析型数据库MySQL版数	3个	联系技术支持申请更多
单个分析型数据库MySQL版可购买的ECU数量最大值	C8：16个	联系技术支持申请更多
一次性申请的ECU个数上限	C8：8个	无需
单个分析型数据库MySQL版每天最大申请扩容或缩容	12次	无例外
单个分析型数据库MySQL版连续24小时导入数据量限制	2 ecuCount diskSize	无例外
单个分析型数据库MySQL版最大并发导入任务数（提交超过此数量任务排队）	C8 2个，S2n 3个	没有高配
表或分区单次导入最大数据量	`min(系统最大值,ecuCountdiskSize 0.2)`	联系技术支持
单个分析型数据库MySQL版最多表数	256	联系技术支持
单个表组总表数	256	联系技术支持
单表最大列数	1024	暂无高配
最大一级分区数	255	暂无高配
单表最大二级分区	1095	联系技术支持
字符串单列上限	8 KB	无例外

分析型数据库MySQL版中磁盘水位与数据读写之间有哪些关系？

- 当 `磁盘水位≥90%` 时，系统将拒绝用户的写数据请求，即写入数据会报错，读数据不受影响。
- 当 `80%≤磁盘水位<90%` 时，AnalyticDB for MySQL系统为防止磁盘被占满而同时影响用户读写请求，系统规定此时用户可以写入数据，但是无法查询这部分数据，待 `磁盘水位<80%` 时才可以查询该数据。
- 分析型数据库MySQL版支持云监控，用户可以通过配置磁盘监控告警，为监控项设置合理的报警规则和通知方式。一旦发生磁盘异常便会立刻为您发出报警通知，让您及时知晓磁盘水位并管理磁盘空间，保证业务正常运行。

和Oracle、MySQL关系型数据库相比，哪些是分析型数据库MySQL版当前不支持的？

- 分析型数据库MySQL版不支持无符号类型（unsigned），由于MySQL和Oracle支持，所以在迁移时可以将源表的字段修改为varchar类型或者bigint类型。
- INSERT支持部分函数。
- 不支持函数索引。
- 不支持存储过程。

分析型数据库MySQL版一个数据库支持创建多少个普通表，如果超过限制数量怎么办？

表数量与节点类型和数量都有关系，后续会自行判断表数量的上限，目前每个数据库默认256个表，需要增加可以联系技术支持解决。

分析型数据库MySQL版性能如何？

分析型数据库MySQL版是分布式架构，支持计算和存储资源水平扩展，查询和写入性能与硬件资源量正相关，接近线性比例关系。在计算资源充足的情况下，特定目标单表查询近3个月内的数据，返回前10000条，响应时间不超过5秒，并发度不低于100。单台前端机入库能力峰值不低于3万行/秒或18 MB/秒（每条按600字节计算，全天按4万秒计算），集群入库能力峰值不低于42万行/秒或250 MB/秒（按每条600字节计算，全天按照4万秒计算）。

11.2. 基本数据库对象及概念

本文汇总了AnalyticDB for MySQL集群中关于基本数据库对象及概念相关的常见问题和答案。

- **是否支持将表从一个表组移到另一个表组？**

  暂不支持。
- **是否支持修改表名？**

  暂不支持。
- **是否支持修改表的一级分区数？**

  当前不支持动态修改，只能删表重建。
- **如何选取普通表的一级分区数以达到最佳计算性能和数据存储均匀分布？**

  AnalyticDB for MySQL中一级分区数默认为128，您无需感知一级分区数，也无需选择。
- **二级分区表的二级分区是否可以是日期类型（date）？**

  每个二级分区列值的通常为日期格式的bigint数值型，例如 `2015091210` 。二级分区列只能是Bigint（long）类型，不能是Date或Time类型。
- **二级分区列的生命周期是多少？**

  当有新的二级分区数据被加载时，分析型数据库MySQL版会以二级分区列的键值排序，删除最小键值的二级分区。
- **为什么要限制二级分区数？**

  过多的二级分区会占用计算节点大量的内存，导致系统容易不稳定。所以我们一般建议二级分区数不要超过90个。单二级分区的数据记录数建议为300万条到2000万之间。假设表A一级分区数为64，二级分区数为90，那么该表最优数据量区间为 `64*90*[300万,2000万]` ，即 `[172.8亿,1152亿]` ，因此二级分区极大地增大了单表的纪录上限。

  如果在实际使用中二级分区数过多，建议增大二级分区键的分区粒度，比如将按日分区改为按月分区、将按月分区改为按年分区等。

11.3. 资源模型相关

分析型数据库MySQL版支持弹性扩缩容和升降配，可以做到用户无感知和业务无影响。

是否允许只对内存进行扩容

分析型数据库MySQL版的实例规格有多种，C4、C8、S2n和S8n，不同的资源规格对应的实例在CPU、内存和磁盘三个维度上会有区别，当某个维度遇到瓶颈时不能单独扩容那个维度。

升级是否需要停业务

分析型数据库MySQL版采用分布式高可用低延时机制，支持在线多节点滚动升级，升级不会影响业务运行。

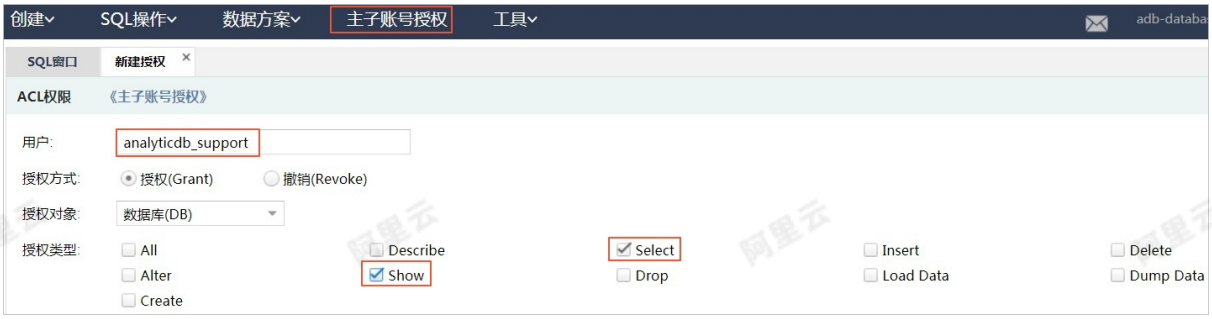
11.4. 账号与权限管理

如何为AnalyticDB for MySQL 2.0支持账号授权

当您在使用AnalyticDB for MySQL过程中遇到问题需要工程师进行问题诊断时，您需要将AnalyticDB for MySQL支持账号 `analyticdb_support` 的 `SELECT` 、 `SHOW` 权限授予工程师，以便工程师查看问题详情，帮助您解决问题。授权方式如下所示。

1. 登录分析型数据库MySQL版控制台。
2. 在**集群列表**页面，找到目标集群，单击右侧的**登录数据库**。
3. 在DMS for AnalyticDB中，单击**主子账号授权**，按照下图所示进行 `analyticdb_support` 账号授权。

> 重要 授权analyticdb_support的只读（SELECT、SHOW）权限即可，问题诊断完毕后需要您及时回收权限。



为主账号授予GRANT、SELECT等权限

主账号的账号格式为 `ALIYUN$account_name` ，其中 `ALIYUN$` 为主账号前缀，标识该账号为阿里云主账号； `account_name` 为主账号的账号名。

例如，将ads_database的 `GRANT OPTION` 、 `SELECT` 、 `INSERT` 权限授予主账号ALIYUN$doc_test。

```
GRANT GRANT OPTION on ads_database.* to 'ALIYUN$doc_test';
GRANT SELECT on ads_database.* to 'ALIYUN$doc_test';
GRANT INSERT on ads_database.* to 'ALIYUN$doc_test';
```

为RAM子账号授予GRANT、SELECT等权限

RAM子账号的账号格式为 `RAM$account_name:subaccount_name` ，其中为子账 `RAM$` 号前缀，标识该账号为RAM子账号； `account_name` 为主账号名； `subaccount_name` 为RAM子账号的账号名。

例如，主账号 `ALIYUN$doc_test` 将 `ads_database` 的 `GRANT OPTION` 和 `SELECT` 权限授予RAM子账号 `RAM$doc_test:liujing` 。

```
GRANT GRANT OPTION on ads_database.* to 'RAM$doc_test:liujing';
GRANT SELECT on ads_database.* to 'RAM$doc_test:liujing';
```

获取主账号的AccessKeyId和AccessKeySecret

1. 主账号登录阿里云控制台。
2. 将鼠标放在右上方的用户名区域，在弹出的快捷菜单中选择**accesskeys**。
3. 系统弹出安全提示对话框，单击继续使用**AccessKey**。页面显示**AccessKeyId**和**AccessKeySecret**。

获取子账号的AccessKeyId和AccessKeySecret

1. 登录
2. 单击用户名进入用户管理页面，找到页面下端的用户**AccessKey**。
3. 单击创建新的**AccessKey**。

通过REVOKE命令移除了表的权限，为何仍可以查询表？

AnalyticDB for MySQL 2.0权限按库> 表组 > 表 > 列的顺序依次向下继承。需要注意的是，如果某个账号拥有数据库级别的权限但撤销其中某个表的权限，则这个账号依然有该表的权限。

您可以通过以下命令查看账号的授权情况，查看某个账号是否拥有数据库级别的权限。

```
show grants for 'ALIYUN$sqream_support' on *;
```

为什么新增用户后，使用新用户账号无法执行命令？

新增用户后，需要为新增用户授权，然后才能使用该账号执行命令。如何授权请参见GRANT

报错信息 `{"errorCode":"40001","errorMsg":"aliyunID not exist"}` 如何处理？

该错误说明阿里云账号不存在，请检查命令中的指定账号是否正确。

报错信息 `calling ak.sdk getServiceStatus error xxx` 如何处理？

以下两种原因导致上述错误：
- 当前账号不是阿里云账号。
- 阿里云UMMAK服务不稳定，建议等待几分钟后重试。

报错信息 `You don't have privilege for connecting database 'xxx', userId=ALIYUN$126*********270, schemaId=***, user=xxx.` 如何处理？

上述报错提示您无权限连接AnalyticDB for MySQL 2.0。您当前使用的AK对应账号没有数据库的权限，请检查AK是否正确，以及AK对应的账号是否拥有数据库的权限。

主账号没有授予子账号GRANT授权，为什么子账号仍有操作权限？

以下两种原因导致上述错误：
- 在RAM中为子账号配置了权限，请前往RAM控制台。
- 主账号授予子账号GRANT授权，可以使用主账号执行SHOW GRANTS。

11.5. 查询报错问题

本文汇总了AnalyticDB for MySQL集群中的常见查询报错问题及处理建议。

查询报错 `QUERY_EXCEED_LIMIT ErrMsg:groups 100000001 exceed limit => 10000000`

用户在执行SQL查询用limit处理分页时，如果start值限制10000无法获取10000以后的数据，如：`LIMIT 1000000,20`。分析型数据库MySQL版对分页数量有限制，即查询返回结果集限制10000行。分析型数据库MySQL版会对select语句查询的返回结果集做全局最大限制，如果不加limit或limit函数超过10000，则只能返回10000行。
- 可以在查询中添加注解 `/*+limitmax=<最大值>*/`，例如 `/*+limitmax=2000000*/select * from ar_express3 limit 1000000,20`` 。
- 请联系技术支持申请调整默认返回限制。

查询时遇到 `memory is not enough`

由于分析型数据库MySQL版查询时，大量数据存在内存中。当SQL所需处理单表或者多表join时的结果较大时，计算节点内存会成为系统瓶颈。分析型数据库MySQL版CN节点为避免SQL压垮系统，会进行自我保护，自动将查询消耗内存较大的SQL fail掉，保证其它查询正常。因此当查询分析型数据库MySQL版报错提示 `memory is not enough`，有以下2种建议：
- 优化SQL，尽量不返回可有可无的结果集，同时尽量对不关心的数据添加过滤条件。若没有很好的处理方法，请联系分析型数据库MySQL版技术支持。
- 若在现有业务基础上无法进行SQL优化，可考虑DB资源扩容，甚至是调整资源模型规格。

查询时报错，提示 `scanRows exceed limit`

分析型数据库MySQL版查询时报错，错误信息为：`ErrMsg:ErrCode:2001 ErrType:QUERY_EXCEED_LIMIT ErrMsg:scanRows exceed limit: xxx >`

为避免用户输入的SQL误写或性能较差，从而导致扫描表的大量数据集，分析型数据库MySQL版会进行scan限制。当SQL scan的行数达到一定数量时，会fail掉SQL的执行并提示用户 `scanRows exceed limit` 错误信息。当遇到该错误时，建议进行以下改进：
- 从SQL本身进行优化，尽量添加更多的过滤条件，筛选出需要关注的数据进行计算。
- 判断select的列是否都有必要获取，因为一个SQL可scan的数据条数=系统默认值/select的列数，若列数减少则可scan的数据条数更多。
- 若SQL自身无法进行优化，但依旧触发该错误，可联系分析型数据库MySQL版技术支持对单个分析型数据库MySQL版的系统默认值进行调整，此操作可能会造成数据库性能下降。

查询提示错误 `META_COLUMN_NOT_EXIST`

在分析型数据库MySQL版表新增字段后，查询提示找不到新添加的列，具体是什么原因？表正在上线，但是上线失败或者CN副本在重启，都有可能出现 `META_COLUMN_NOT_EXIST`。
- **对于普通表**

  一级普通表：新增字段且执行optimize成功后可查询到新增字段（老数据为null）。

  二级普通表：新增字段且需触发最大二级分区的optimize，待optimize成功后可查询到新增字段。
- 对于未上线或者CN副本重启导致的报错，请检查上线情况或者重启情况。

> **说明**
> optimize操作为异步后台操作（目前未开放接口给用户，可联系分析型数据库MySQL版技术支持），需等待任务完成后再做查询。

查询刚刚创建的新表时报错 `table is not ready`

新创建的表，可以INSERT数据，但查询该表报错 `table is not ready`。

```
ERROR 30007 (HY000): ProcessId=20171130140151010154169109099990872 31 RT_ROUTER_ERROR tableId=28 DBId=1405 message=real time table is not ready.
```

一般表建好以后，还需分配相应的资源，需等待几分钟表才能正常使用，否则就会报 `not ready` 错误。

查询报错 `NO_NODES_AVAILABLE`

报错关键字：`NO_NODES_AVAILABLE`

报错信息：

```
ERROR 1105 (HY000): MPP_QUERY_FAILED Retry_time=1 message=MPP engine error code: 30101, message: QueryError{message=Node for bucket is offline: [], sqlState=null, errorCode=65541, errorName=NO_NODES_AVAILABLE, errorType=INTERNAL_ERROR, errorLocation=null, failureInfo=mpp.client.FailureInfo@13887b2e}
```

一般是数据库内部某个计算节点压力较大导致该节点暂时离线，系统能够自动修复。请用户过5~10分钟后重试。如果长时间出现该报错，请联系分析型数据库MySQL版技术支持。

查询性能不稳定

在分析型数据库MySQL版中执行SQL查询语句时，时慢时快。引起分析型数据库MySQL版查询不稳定的因素一般有以下几种：
- 用户执行SQL时首次较慢，之后查询明显比第一次快，这是因为分析型数据库MySQL版自身带有缓存。第一次查询时会将数据缓存到内存中，若下次查询所需要的数据依然在内存中时，此时不用再次读磁盘，因此访问速度很快。
- 用户使用的是普通表，且写入大量数据后未执行optimize操作，此时执行SQL会随着插入数据多而变得越来越慢。这是因为分析型数据库MySQL版的实时数据增量部分默认未建索引，因此执行查询时会频繁访问磁盘。为保证查询速度，建议当用户执行过大量INSERT或DELETE语句后，立即触发一次optimize table操作。

- 用户对普通表执行大量的DELETE语句，而这些DELETE的条件中没有指定分区键，导致系统在所有分区上执行DELETE，耗费CN上大量CPU和内存资源。此时其他查询也有可能变慢或超时。
- 按上述排查后还存在问题，请联系分析型数据库MySQL版技术支持。

查询报错 `IN expr values exceed limit`

带有 `IN ()` 条件的查询报错 `IN expr values exceed limit` ，默认情况下，如果 `IN ()` 中的值的个数大于50，则会报该错误。如果要修改此限制，请联系分析型数据库MySQL版技术支持。

11.6. 应用开发向导

连接云原生数据仓库MySQL版推荐的方式是 `druid-jdbc` 或 `tddl` ？

当使用JDBC连接池连接云原生数据仓库MySQL版时，推荐使用Druid连接池，且尽量使用最新版本。请参考Druid。

关于Druid连接池配置，请务必按照如下配置项进行配置：

```
- maxActive:100（最大值根据业务并发量来定，建议该值大于等于业务并发数）
- initialSize:5
- maxWait:60000
- minIdle:10
- maxIdle:20
- timeBetweenEvictionRunsMillis:2000
- minEvictableIdleTimeMillis:600000
- maxEvictableIdleTimeMillis:900000
- validationQuery: show status like '%Service_Status%';
- testWhileIdle:true
- testOnBorrow:false
- testOnReturn:false
- removeAbandoned:true
- removeAbandonedTimeout:180
```

使用SQL开发工具访问云原生数据仓库MySQL版，推荐使用哪种连接方式？

云原生数据仓库MySQL版完全兼容MySQL协议，基本上能访问MySQL的客户端都能用来访问云原生数据仓库MySQL版，推荐使用官方客户端DMS for AnalyticDB和开源工具DBeaver连接云原生数据仓库MySQL版。

如何解决访问云原生数据仓库MySQL版遇到的报错 `Communications link failure` ？

报错文本如下：

```
Communications link failure，Thelast packet successfully received from the server was 0 millisecond ago.Thelast packet successfully sent to the server was YYYY millisecond ago
```

1. 上述报错可以确定用户的JDBC URL错误或者网络不通。可以使用MySQL客户端测试是否可以连接数据库。
2. 检查本地的MySQL驱动版本；我们支持的版本有如下所示。

   5.0系列：5.0.2，5.0.3，5.0.4，5.0.5，5.0.7，5.0.8

   5.1系列：
   5.1.1，5.1.2，5.1.3，5.1.4，5.1.5，5.1.6，5.1.7，5.1.8，5.1.11，5.1.12，5.1.13，5.1.14，5.1.15，5.1.16，5.1.17，5.1.18，5.1.19，5.1.20，5.1.21，5.1
   5.1.32, 5.1.33, 5.1.34
3. 是否采用连接池连接云原生数据仓库MySQL版，若不是，请采用连接池连接，推荐采用Druid。
4. 应用是否多线程访问云原生数据仓库MySQL版，线程之间是否共享使用Connection对象,如有，请修改，强烈建议不要在多线程之间共享使用Connection对象。
5. 如果报错中的 `millisecond` 值不为0，则基本由耗时很长的查询导致，请在代码中加入重试机制。重试机制示例请参见Java访问-带重试的JDBC连接示例。

云原生数据仓库MySQL版SQL语句中怎么加多个hint？

多个hint可以通过逗号进行分隔，示例如下。

```
/+engine=MPP, mppNativeInsertFromSelect=true/INSERT INTO db_name.target_table_name (col1, col2, col3)
SELECT col1, col2, col3 FROM db_name.source_table_name
WHERE col4 ='123';
```

如何解决使用MySQL Client时Hint无法正常生效的问题？

用户在进行MPP下DUMP的时候，通过MySQL客户端连接无法正常DUMP，hint没有被识别。

使用MySQL客户端连接云原生数据仓库MySQL版，需要指定 `-c` 参数进行连接，否则后续查询的hint可能无效。

12.2.0版附录

12.1. 保留字（2.0版）

上述保留字大小写不敏感。

- 数据库名保留字

```
SYSDB,ADMIN
```

- 列保留字

```
ADD,ADDONINDEX,AGAINST,ALTER,ALL,ANY,AND,AS,ASC,BEFORE,BEGIN,BINARY,BOOLEAN,BY,BETWEEN,BLOB,BOTH,CALL,CASCADE,CAST,CHANGE,CHAR,CHARACTER,CHECK
ATION,COLLATE,CONDITION,

  CONNECTION,CONSTRAINT,CONTAINS,CONTINUE,CONVERT,COMPUTE,CROSS,CURRENT,CURRENT_DATE,CURRENT_TIME,

  CURRENT_TIMESTAMP,CURRENT_USER,CURSOR,CREATE,COLUMN,COLUMNS,CASE,DATABASES,DATABASE,DATE,DAY_HOUR,DAY_MICROSECOND,DAY_MINUTE,

  DAY_SECOND,DEC,DECIMAL,DECLARE,DEFAULT,DELAYED,DETERMINISTIC,DISTINCTROW,DIV,DO,DOUBLE,DELETE,


DUAL,DROP,DISTINCT,DESC,DESCRIBE,DICTIONARY,DIMENSION,EACH,ELSEIF,ENCLOSED,ESCAPED,ESCAPE,EXIT,EXCEPT,ELSE,END,ENGINE,EXISTS,EXPLAIN,EXTRACT,


  FETCH,FIRST,FOLLOWING,FLOAT,FOR,FORCE,FROM,FOREIGN,FULLTEXT,FALSE,FUNCTION,FULL,GLOBAL,GRANT,GOTO,GROUP,


HOUR_MICROSECOND,HOUR_MINUTE,HOUR_SECOND,HAVING,IF,IGNORE,IDENTIFIED,IN,IS,INFILE,INT,INTO,INSERT,INTEGER,ITERATE,INDEX,INDEXES,INTERSECT,INTE
INNER,JOIN,KEEP,KILL,KEY,KEYS,LANGUAGE,LAST,LABEL,

  LEADING,LEAVE,LOAD,LOCALTIME,LOCALTIMESTAMP,LOCK,LIMIT,LONG,LOOP,LEFT,LIKE,MATCH,MINUTE_MICROSECOND,

  MINUTE_SECOND,MOD,MERGE,MODIFIES,MINUS,MODE,NATURAL,NULL,NOT,NULLS,
  OFFSET,OPTIMIZE,OPTION,OUT,OPEN,OUTFILE,OUTER,ON,OR,ORDER,OVER,OVERWRITE,PRECEDING,PRECISION,REPLACE,RANGE,PROCEDURE,PRIMARY,READ,REAL,

  REFERENCES,REGEXP,RELEASE,RENAME,REPEAT,REQUIRE,RESTRICT,RETURN,RLIKE,RIGHT,ROW,ROWS,SELECT,SCHEMA,

  SECOND+MICROSECOND,SEPARATOR,SESSION,SET,SQLSTATE,STATUS,STORED,SYSTEM,SMALLINT,SPATIAL,SHOW,SPECIFIC,

  SQL,SOME,TINYINT,TINYBLOB,TABLE,TABLEGROUP,TABLES,TIME,TIMESTAMP,TRIGGER,TO,TOP,TRUNCATE,TRUE,THEN,UNLOCK,


UNBOUNDED,UNSIGNED,USE,USING,UNTIL,UPDATE,UNDO,UNIQUE,UNION,VARCHAR,VALUES,VARIABLES,VIEW,WHILE,WITH,WITHIN,WHEN,WHERE,WRITE,XOR,YEAR_MONTH
```

12.2. 错误码表（2.0版）

DDL相关错误码

范围	说明
18000 ~ 18100	DDL CREATE语句用户错误。
18600 ~ 18799	DDL ALTER语句用户错误。
18800 ~ 18899	DDL DROP语句用户错误。
19000 ~ 19599	DDL CREATE语句系统错误。
19600 ~ 19799	DDL ALTER语句系统错误。
19800 ~ 19899	DDL DROP语句系统错误。

DDL CREATE语句用户错误

错误码	错误信息	解决办法
18000	DenyAccessException:You do not have[xxx] access to resource[xxx]	无对特定数据库资源的特定操作权限，请确认，或按需要进行赋权限操作。
18001	Not support CREATE DATABASE in db: schema	无法在该数据连接上进行CREATE DATABASE操作，请检查所用数据库连接是否正确。
18002	非ADS user导致的失败信息。	操作必须由ADS user进行，请确认当前使用的UMM账号是ADS用户账号。
18003	NA	NA
18004	Illegal options in CREATE DATABASE:	非法的CREATE DATABASE命令选项参数，请参考建库文档进行修改。
18005	NA	NA
18006	xxx database already exists.	目标数据库已经存在，请确认数据库是否重名。
18007	Target database does not exist.	目标数据库不存在，请确认数据库名是否正确。
18008	Table group ‘schema.tablegroup’ already exists.	目标表组不存在，请确认表组名是否正确。

18009	IllegalParameterException: parameterName: xxx, message: xxx	DDL语句参数不正确，请参考DDL文档，或进一步联系技术支持。
18010	参数值非法的详细错误信息。	DDL语句参数值非法，请按详细提示信息修改，或进一步联系技术支持。
18011	SUBPARTITION is not supported by DIMENSION table.	维度表不支持二级分区，请修改。
18012	TABLEGROUP must not be specified for DIMENSION table.	维度表建表语句不能指定表组，维度表均归属于系统默认维度表组，请修改。
18013	The minimum PARTITION NUM allowed for fact table is xxx, but xxx was defined.	不满足分区表的最小分区数定义，请修改。
18014	Table 'table' already exists.	目标表已经存在，请确认表是否重名。
18015	xxx is the dimension table group, could not be used.	自定义表组不能使用系统默认维度表组名，请修改。
18016	NA	NA
18017	Exceed the tables limitation (xxx) of database 'xxx'	目标数据库下的表数量已经到上限，不可继续建表，请联系技术支持。
18018	Exceed the tables limitation (xxx) of table group 'xxx'	目标数据库表组下的表数量已经到上限，不可继续建表，请联系技术支持。
18019	NA	NA
18020	Duplicated column definition	DDL中有重复列定义，请修改。
18021	There are xxx columns, which is not in the valid range: 1 to xxx	建表语句列数超限，请联系技术支持。
18022	Partition column does not exist	DDL中定义的分区列名不在定义的列中，请检查并修改。
18023	Column type is invalid	列数据类型非法，请参考DDL文档中关于支持的数据类型进行修改。
18024	DDL语句语法错误的详细信息。	DDL语句语法错误，请参考DDL文档进行修改，或进一步联系技术支持。
18025	No database selected.	相关操作未找到目标数据库，请检查数据库连接是否正确包含目标数据库信息，或者操作是否指定目标数据库。
18026	Table group should be created first.	建分区表时指定的表组不存在，需先建表组。
18027	相关命名的详细错误信息。	数据库对象命名错误，请按照提示进行修改，或进一步联系技术支持。
18028	Target sub-partition column does not exist.	指定的二级分区列列名不在定义的列中，请检查并修改。
18029	Only LONG/BIGINT is allowed for subpartition column data type.	二级分区列数据类型只能为LONG/BIGINT，请修改。
18030	Sub-partition number is illegal.	二级分区数非法，请修改。
18031	Exceed maximum user database limitation: xxx, user: xxx	用户下数据库总数已经达到上限，不可继续建库，请联系技术支持。
18032	No partition information provided for none dimension table.	建分区表时缺少分区信息子句，请参考DDL文档进行修改。
18033	Not support create table without tablegroup	建分区表时缺少表组子句，请参考DDL文档进行修改。
18034	Wrong table options	建表指定的options子句中参数错误，请参考DDL文档进行修改。
18035	PRIMARY KEY does not exist for real time table.	建实时表时缺少主建定义，请参考DDL文档进行修改。
18036	PRIMARY KEY does not contain partition column "xxx" for real time partition table.	建实时表时，主建定义中未包含分区列，请修改。
18037	Subpartition must not exist for real time table.	实时表暂不支持二级分区，请联系技术支持。
18038	NA	NA
18039	Table option LIFECYCLE is not allowed for non real time table.	建非实时表时不能指定LIFECYCLE参数，请修改。
18040	Old version of CREATE TABLE statement is blocked, please use the new version.	老版本建表语句被禁止，请参考DDL文档，使用当前的建表语句语法。
18041	CLUSTER BY column does not exist	建表语句中，CLUSTER BY指定的列不在定义的列中，请检查并修改。
18042	It is not allowed to create real time table on database: xxx	禁止在目标数据库中建实时表，请联系技术支持。
18043	Index column was not in the column definition list	索引列不在定义的列中，请检查并修改。
18044	Duplicated index name	DDL中有重复索引定义，请修改。
18045	The maximum PARTITION NUM allowed for fact table is xxx, but xxx was defined.	建分区表的分区数超过上限，请修改，或进一步联系技术支持。
18046	There are xxx sub partition number in definition, which is not in the valid range: x to x	二级分区数超过上限，请修改，或进一步联系技术支持。

18047	Invalid default value for xxx	列定义中的默认值表达式非法，请修改。
18048	MULTIVALUE/TIME/DATE is not allowed for partition column data type.	多值列、TIME、DATE类型的列不能作为分区列，请修改。
18049	Subpartition column must be one of the primary key columns.	建实时表时，二级分区列必须是主建列之一，请修改。
18050	相关功能还不支持的详细信息。	请参考提示的详细信息，或进一步联系技术支持。
18051	CTAS_LOAD_DATA_TIMEOUT schema=xxx table=xxx	CTAS执行的LOAD DATA阶段超时，请重试，或进一步联系技术支持。
18052	CTAS_META_CHECK_THREAD_ERROR message=xxx	CTAS执行的元数据校验阶段超时，请重试，或进一步联系技术支持。
18053	CTAS_LOAD_DATA_FAILED schema=xxx table=xxx jobState=xxx	CTAS执行的LOAD DATA阶段失败，请重试，或进一步联系技术支持。
18054	CTAS_SELECT_SQL_ANALYZE_ERROR schema=xxx table=xxx message=xxx	CTAS语句的SELECT子句语法分析失败，请检查语法，或进一步联系技术支持。
18055	CTAS_INSERT_THREAD_ERROR message=xxx	CTAS执行的INSERT阶段失败，请重试，或进一步联系技术支持。
18056	CTAS_INSERT_TIMEOUT schema=xxx table=xxx timeoutDuration=xxx	CTAS_INSERT_THREAD_ERROR message=xxx
18057	SUBPARTITION column conflicts with PARTITION column	二级分区列与一级分区列冲突，请修改。
18058	No column definition.	DDL语句无列定义，请参考DDL文档修改。
18059	Illegal table partition type: xxx, only HASH is allowed.	建分区表时，错误的分区类型，目前仅支持HASH分区类型，请修改。
18060	Illegal table sub partition type: xxx, only LIST is allowed.	建二级分区表时，错误的二级分区类型，目前仅支持LIST分区类型，请修改。
18061	Invalid index type: xxx	不支持的索引类型，请参考DDL文档修改。
18062	CTAS_RETRIEVE_COL_DEF_FAIL message=xxx	CTAS执行的列定义获取阶段失败，请检查SELECT部分的语法，或重试，或进一步联系技术支持。
18063	Illegal worker label value: xxx, value pattern should be “read:write”; Illegal worker ratio value: xxx, value pattern should be “number:number”	读写分离读写比例参数错误，请参考DDL文档修改。
18064	EXECUTE_CREATE_CACHE_TABLE_ERROR message=xxx query=xxx	建cache表失败，请参考DDL文档检查语法，或进行重试，或进一步联系技术支持。
18065	Lack options in CREATE DATABASE: xxx, options should be contained both ‘worker_labels’ and ‘worker_ratios’ or not.	建读写分离库时，读写分离参数错误，请根据提示和DDL文档修改。
18067	NA	NA
18068	Wrong format for external catalog properties.	外部数据源catalog创建语句的properties子句格式错误，请参考DDL文档修改。
18069	xxx external catalog already exists.	外部数据源catalog已经存在，请确认catalog是否重名。
18070	NA	NA
18071	Could not use the existing schema name	外部数据源catalog名字不能和已存在的数据库重名，请修改。
18072	Table option UPDATETYPE is unknown. Only ‘realtime’ and ‘batch’ are supported.	表选项UPDATETYPE只支持realtime和batch，请修改。
18073	Invalid FULLTEXT index column data type xxx of column xxx, VARCHAR was expected.	FULLTEXT全文索引列只支持VARCHAR类型，请修改。
18074	Database name length exceeds the limitation of ‘xxx’, current length is ‘xxx’.	CREATE DATABASE指定的目标数据库名长度超限，请修改。
18075	Illegal ‘dbNameBytesMaxLength’ zk value: xxx, value type should be numeric.	CREATE DATABASE的目标数据库名长度限制ZK配置数据类型错误，请联系技术支持。
18076	Target view ‘xxx’ already exist.	CREATE VIEW的目标视图已经存在，请确认或修改。
18077	Table with the same name ‘xxx.xxx’ already exists.	CREATE VIEW的目标视图名和已经存在表名冲突，请确认或修改。
18100	Realtime table count exceeds limit in this database: xxx, ecu type: xxx	目标数据库实时表数量超过了ecu的上限，无法继续建实时表，请联系技术支持。

DDL ALTER语句用户错误

错误码	错误信息	解决办法
18600	DenyAccessException:You do not have[xxx] access to resource[xxx]	无对特定数据库资源的特定操作权限，请确认，或按需要进行赋权限操作。
18601	NA	NA

18602	DDL语句语法错误的详细信息。	DDL语句语法错误，请参考DDL文档进行修改，或进一步联系技术支持。
18603	No database selected.	相关操作未找到目标数据库，请检查数据库连接是否正确包含目标数据库信息，或者操作是否指定目标数据库。
18604	目标数据库对象不存在的详细信息。	目标数据库对象不存在，请检查。
18605	参数值非法的详细错误信息。	DDL语句参数值非法，请按详细提示信息修改，或进一步联系技术支持。
18606	参数值非法的详细错误信息。	DDL语句参数值非法，请按详细提示信息修改，或进一步联系技术支持。
18607	AlterRepeatException: columnName: xxx repeat.	列名重复，请确认目标表无该名字的列。
18608	AlterRepeatException: indexName: xxx repeated.	索引目标列已经定义过索引，不能再添加其他索引。
18609	NA	NA
18610	Column type is invalid:	列数据类型非法，请参考DDL文档中关于支持的数据类型进行修改。
18611	DROP COLUMN is not supported yet.	暂不支持ALTER TABLE DROP COLUMN。
18612	Illegal index type:	不支持的索引类型，请参考DDL文档修改。
18613	Do not allow to add column to real time table:	暂不支持对实时表进行加列，请联系技术支持。
18614	Invalid default value for xxx	列定义中的默认值表达式非法，请修改。
18615	Add PK column is not allowed for real time table.	实时表不允许通过ALTER加主建列，请修改。
18616	Add virtual column is not allowed for real time table.	实时表不允许通过ALTER加虚拟列，请修改。
18617	非法的ALTER语句参数的详细信息。	ALTER语句参数非法，请根据提示参考DDL文档修改。
18618	Index “xxx” already exist on xxx.xxx	索引名重复，请确认目标表上无该名字。
18619	The operation is not supported yet.	不支持的ALTER操作，请联系技术支持。
18620	Resize Operation is not allowed in this database.	当前用户不允许对目标数据库进行变更资源操作，请联系技术支持。
18621	There are xxx sub partition number in definition, which is not in the valid range: x to x	二级分区数超过上限，请修改，或进一步联系技术支持。
18622	Target is not sub partition table.	目标表不是二级分区表，无法进行二级分区数调整操作。
18623	Illegal frontnode_rw_instance_ratio value: xxx, value pattern should be “number:number”	读写分离读写比例参数错误，请参考DDL文档修改。

DDL DROP语句用户错误

错误码	错误信息	解决办法
18800	No database selected.	相关操作未找到目标数据库，请检查数据库连接是否正确包含目标数据库信息，或者操作是否指定目标数据库。
18801	Database ‘xxx’ was not found.	目标数据库不存在，请确认。
18802	Can not drop non-empty database.	不能DROP非空的数据库，先DROP完库中的表，之后再DROP该库。
18803	Could not explicitly drop dimension group ‘xxx’.	不能DROP系统默认的维度表组。
18804	Target table group does not exist:	目标表组不存在，请确认。
18805	Can not drop non-empty tablegroup.	不能DROP非空的表组，先DROP完表组中的表，之后再DROP该表组。
18806	Illegal drop target type, only DATABASE/TABLEGROUP/TABLE/EXTERNAL CATALOG is allowed.	DROP的目标数据库对象类型非法，只允许DROP DATABASE/TABLEGROUP/TABLE/EXTERNAL CATALOG。
18807	Target table does not exist:	目标表不存在，请确认。
18808	Drop failed. xxx message=xxx	删除cache表失败，请重试，或进一步联系技术支持。
18809	Drop external catalog failure due to:	删除外部数据源catalog失败，请重试，或进一步联系技术支持。

DDL CREATE语句系统错误

错误码	错误信息	解决办法
19000	LOCK_SERVICE_ERROR message=Acquire lock failed.	DDL操作获取ZK锁失败，请稍后重试，或进一步联系技术支持。
19001	NA	NA
19002	NO_ALB_INSTANCE message=No available ALB load balancer instance found.	创建目标库时，未找到集群可用的load balancer实例，无法创建，请联系技术支持。
19003	NA	NA

19004	Illegal SLB VIP front end port: xxx, the valid port range is xxx-xxx	在元数据中指定VIP/PORT创建目标库时，指定的PORT不在合理范围内，请联系技术支持。
19005	ALB_OPERATION_FAIL message=xxx	创建目标库时，创建VIP步骤失败，请联系技术支持。
19006	Add DNS resolve record failed.	创建目标库时，绑定DNS域名步骤失败，请联系技术支持。
19007	NA	NA
19008	NA	NA
19009	NA	NA
19010	分配数据库对象ID失败的详细信息。	为创建目标数据库对象分配ID失败，请稍后重试，或进一步联系技术支持。
19011	CREATE statement is blocked.	DDL CREATE语句被禁止，请联系技术支持。
19012	CREATE DATABASE statement is disabled.	DDL CREATE DATABASE语句被禁止，请联系技术支持。
19013	CREATE TABLEGROUP statement is disabled for database ‘xxx’.	目标数据库的DDL CREATE TABLEGROUP语句被禁止，请联系技术支持。
19014	CREATE TABLE statement is disabled for database ‘xxx’.	目标数据库的DDL CREATE TABLE语句被禁止，请联系技术支持。
19015	CREATE EXTERNAL CATALOG statement is disabled for database ‘xxx’.	目标数据库的DDL CREATE EXTERNAL CATALOG语句被禁止，请联系技术支持。
19599	CREATE操作的其他类型失败的详细信息。	CREATE操作遇到其他类型失败，请联系技术支持。

DDL ALTER语句系统错误

错误码	错误信息	解决办法
19600	NA	NA
19601	ALTER statement is blocked.	DDL ALTER语句被禁止，请联系技术支持。
19602	ALTER DATABASE statement is disabled.	DDL ALTER DATABASE语句被禁止，请联系技术支持。
19603	ALTER TABLEGROUP statement is disabled for database ‘xxx’.	目标数据库的DDL ALTER TABLEGROUP语句被禁止，请联系技术支持。
19604	ALTER TABLE statement is disabled for database ‘xxx’.	目标数据库的DDL ALTER TABLE语句被禁止，请联系技术支持。
19605	Invalid FULLTEXT index column data type xxx of column xxx, VARCHAR was expected.	FULLTEXT全文索引列只支持VARCHAR类型，请修改。
19699	ALTER操作的其他类型失败的详细信息。	ALTER操作遇到其他类型失败，请联系技术支持。

DDL DROP语句系统错误

错误码	错误信息	解决办法
19800	NA	NA
19801	DROP statement is blocked.	DDL DROP语句被禁止，请联系技术支持。
19802	DROP DATABASE statement is disabled.	DDL DROP DATABASE语句被禁止，请联系技术支持。
19803	DROP TABLEGROUP statement is disabled for database ‘xxx’.	目标数据库的DDL DROP TABLEGROUP语句被禁止，请联系技术支持。
19804	DROP TABLE statement is disabled for database ‘xxx’.	目标数据库的DDL DROP TABLE语句被禁止，请联系技术支持。
19805	DROP EXTERNAL CATALOG statement is disabled for database ‘xxx’.	目标数据库的DDL DROP EXTERNAL CATALOG语句被禁止，请联系技术支持。
19899	DROP操作的其他类型失败的详细信息。	DROP操作遇到其他类型失败，请联系技术支持。

DML相关错误码

范围	说明
0 ~ 20999	DML用户错误。
30000 ~ 60009	DML系统错误。

DML用户错误

错误码	错误信息	解决办法
1044	Can not use this command here !	不支持的MySQL协议命令，请确认。
1045	其他语句执行异常	请联系技术支持。
1046	No database specified in FROM table	查询未能找到目标表的归属DB，请检查数据库连接使用的DB，或者直接为目标表前面加上DB前缀。

1143	Access deny for accessing database 'schema' from this node.	请联系技术支持，确认是否进行了DB节点绑定操作。
1146	Table xxx doesn't exist.	请确认目标表xxx在当前连接和访问的DB下确实存在。
1147	No COLUMN:column found in TABLE table	请确认目标表table确实包含列column。
1148	Column xxx does not exist.	查询语句中Column分析的相关错误请联系技术支持查看执行的SQL。
1235	SQL where expression contains/in items count exceed limit:	1.查询语句中LIMIT子句值超过配置允许的最大值，请修改；2. 查询语句中CONTAINS/IN子句中的项目数超过配置允许的最大值，请修改减少项目数。
1236	SQL feature NOT supported yet: COUNT(DISTINCT xx) without join tables' partition columns	查询语句中xxx相关语法功能不支持。
1236	SQL feature NOT supported yet: SELECT * or SELECT <table>.* with UNION/INTERSECT/MINUS.	NA
1236	SQL feature NOT supported yet: SELECT agg_func() ... UNION/UNION ALL/INTERSECT/MINUS SELECT agg_func() ...	NA
1236	SQL feature NOT supported yet: SELECT ... LIMIT X UNION/UNION ALL/INTERSECT/MINUS SELECT ... LIMIT X	NA
1236	SQL feature NOT supported yet: FULL JOIN	NA
1236	SQL feature NOT supported yet: RIGHT JOIN	NA
1236	SQL feature NOT supported yet: SELECT (A JOIN B ...)	NA
1236	SQL feature NOT supported yet: xxx	NA
20000	[USER ERROR] Invalid SQL.	SQL语法异常，请检查并修改，或进一步联系技术支持。
20001	[USER ERROR] Invalid data value type.	SQL中有数据的值和其对应的数据列类型不匹配，请根据提示修改。
20002	[USER ERROR] Invalid hint.	下发到COMPUTENODE的查询HINT非法，请检查SQL，或进一步联系技术支持。
20003	[USER ERROR] Invalid UDF.	COMPUTENODE不支持的UDF，请确认，或进一步联系技术支持。
20004	[USER ERROR] Query items exceed limitation.	COMPUTENODE计算超限，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20005	[USER ERROR] Query must GROUP BY column.	SQL语句错误，按照提示，必须GROUP BY提示的列。
20006	[USER ERROR] Query misses join condition.	SQL语句错误，join子句缺少ON条件。
20007	[USER ERROR] Query misses join index.	已经不再出现该错误，如出现，请联系技术支持。
20008	NA	NA
20009	[USER ERROR] Unsupported query syntax.	SQL语法错误，请参考SQL语法文档，进行修改。
20010	Parsing insert statement failed.	NA
20011	[USER ERROR] fact table partition column was not provided for INSERT statement.	目标表为分区表时，INSERT语句的列集合中，必须包含分区列。
20012	[USER ERROR] Invalid column value class type (fastsql xxx	非法的列值，列值和列类型不匹配，请修改。
20013	INSERT语句语法错误的详细信息。	INSERT语句语法错误，请按照提示修改。
20014	INSERT statement is blocked.	INSERT语句被禁止，请联系技术支持。
20015	RT_UPN_HANDLER_INVALID message=Data buffer handler was not found.	INSERT执行失败的详细信息。请重试，或进一步联系技术支持。
20016	[USER ERROR] Target table was not found at this time.	已经不再出现该错误，如出现，请联系技术支持。
20017	Delete statement is blocked.	DELETE语句被禁止，请联系技术支持。
20018	DELETE语句语法错误的详细信息。	DELETE语句语法错误，请按照提示修改。
20019	DELETE执行失败的详细信息。	请重试，或进一步联系技术支持。
20020	[USER ERROR] Target table was not found at this time.	已经不再出现该错误，如出现，请联系技术支持。
20021	[USER ERROR] Target table has no primary key.	目标实时表没有主建列，请联系技术支持。
20022	[USER ERROR] Must and only contains AND joined EQUAL-TO predicates of all columns within the primary key:	严格的DELETE where条件检查，请联系技术支持更改系统配置。
20023	[USER ERROR] column was not found: column=xxx	INSERT语句中，目标列不存在，请修改。
20024	[USER ERROR] column was not found: column=	严格的DELETE where条件检查时，DELETE语句中，目标列不存在，请修改，并联系技术支持更改系统配置。

20025	[USER ERROR] column value is invalid: column=xxx, type=xxx, value=xxx	严格的DELETE where条件检查时，DELETE语句中，目标列的值和类型不匹配，请修改，并联系技术支持更改系统配置。
20026	Real time data FLUSH is blocked.	FLUSH语句被禁止，请联系技术支持。
20027	Real time data MERGE is blocked.	MERGE语句被禁止，请联系技术支持。
20028	INSERT is not allowed when the real time table is not ready.	已经不再出现该错误，如出现，请联系技术支持。
20029	DELETE is not allowed when the real time table is not ready.	已经不再出现该错误，如出现，请联系技术支持。
20030	[USER ERROR] Target table is not real time table.	目标表不是实时表，无法INSERT。
20031	[USER ERROR] Target table is not real time table.	目标表不是实时表，无法INSERT。
20032	[USER ERROR] column value lists do not match.	INSERT语句中列集合的列表和VALUES子句中值的个数不匹配，请确认修改。
20033	Insert failed due to encoding exception:	INSERT语句中数据编码有问题，请确认，或进一步联系技术支持。
20034	[USER ERROR] Target table has no primary key.	目标实时表没有主建列，请联系技术支持。
20035	[USER ERROR] miss primary key column:	INSERT语句的列集合未包含所有的主建列，请修改。
20036	RT_DATA_WRITE_TIMEOUT schema=xxx table=xxx	INSERT执行超时，请重试，或进一步联系技术支持。
20037	[USER ERROR] null value is not allowed for NOT NULL column: column=xxx	INSERT语句中，NOT NULL的列插入了NULL值，请修改。
20038	[USER ERROR] Illegal argument.	COMPUTENODE执行下发SQL时出现参数错误，请参考SQL语法文档，或进一步联系技术支持。
20039	[USER ERROR] Unsupported query operation.	COMPUTENODE执行下发SQL时出现语法错误，请参考SQL语法文档，或进一步联系技术支持。
20040	No database selected.	请检查是否在数据库连接中指定了目标数据库。
20041	No database selected.	请检查是否在数据库连接中指定了目标数据库。
20042	COUNT DISTINCT on non-partitioning column exceeds the partition data limitation:	进行非分区列COUNT DISTINCT操作时，单分区返回的明细数据超过了限制，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20043	UDF_SYS_ROWCOUNT exceeds the partition data limitation:	SQL中包含UDF_SYS_ROWCOUNT函数时，单分区返回的明细数据超过了限制，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20044	[USER ERROR] Must contains WHERE expression.	DELETE语句中未写WHERE子句，请修改。
20045	[USER ERROR] sub select condition was not support for delete statement.	已经不再出现该错误，如出现，请联系技术支持。
20046	SQL feature NOT supported yet: GROUP_CONCAT without grouping tables' partition columns.	使用group_concat函数时，order by不起作用。
20047	SQL feature NOT supported yet: COUNT(DISTINCT x1, x2, ...) against multiple non-partitioning columns	不支持对多个非分区列的COUNT DISTINCT操作。
20048	Unsupported expression in HAVING: xxx	HAVING子句表达式非法的详细信息。请参考SQL文档进行修改。
20049	SQL dump data selecting columns count exceed limit:	DUMP DATA语句中的SELECT列数超过上限，请限制并修改，或进一步联系技术支持。
20050	SQL selecting columns count exceed limit:	查询语句中的SELECT列数超过上限，请限制并修改，或进一步联系技术支持。
20051	NA	NA
20052	UDF_SYS_SUM exceeds the partition data limitation:	SQL中包含UDF_SYS_SUM函数时，单分区返回的明细数据超过了限制，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20053	INSERT is not allowed since the compute node disk is almost full: worker=%s disk_used=%s disk_size=%s usage_ratio=%s	COMPUTENODE实例节点存储空间到上限，无法继续INSERT数据，请联系技术支持。
20054	Unknown column xxx in 'ORDER BY clause'	ORDER BY子句中的列非法，请检查SQL并修改。
20055	SQL feature NOT supported yet: SELECT ... ORDER BY ... UNION/UNION ALL/INTERSECT/MINUS SELECT	不支持的UNION语法：UNION子查询中带ORDER BY子句。
20056	Unknown column xxx in 'GROUP BY clause'	GROUP BY子句中的列非法，请检查SQL并修改。
20057	NA	NA
20058	RT_UPN_ALTERTABLE_ERROR db=xxx table=xxx part=xxx message=xxx	向BUFFERNODE发ALTER TABLE语句时，遇到encoding异常，请联系技术支持。
20059	[USER ERROR] null value is not allowed for sub partitioning column: column=xxx	INSERT实时数据时，二级分区列对应的值不能为NULL，请修改。
20060	[USER ERROR] sub partitioning column value is out of range: column=xxx	INSERT实时数据时，二级分区列对应的值不在合法的范围，请修改。

20061	[USER ERROR] sub partitioning column value is invalid: column=	INSERT实时数据时，二级分区列对应的值不合法，请修改。
20062	[USER ERROR] sub partition column was not provided for INSERT statement.	INSERT实时数据时，如果目标表是二级分区表，插入的列和值的集合必须包含二级分区列，请修改。
20063	[USER ERROR] sub partitioning column value is out of range: column=xxx, type=LONG/BIGINT, value=xxx	已经不再出现该错误，如出现，请联系技术支持。
20064	[USER ERROR] sub partitioning column value is invalid: column=xxx, type=LONG/BIGINT, value=xxx	已经不再出现该错误，如出现，请联系技术支持。
20065	[USER ERROR] sub partitioning column EQUAL-TO predicate was not provided.	已经不再出现该错误，如出现，请联系技术支持。
20066	ORDER BY item ‘xxx’ was not found in GROUP BY clause.	查询语句包含GROUP BY和ORDER BY子句时，ORDER BY中的列必须出现在GROUP BY子句中。
20067	SQL GROUP-BY column expected:xxx	SQL语句错误，按照提示，必须GROUP BY提示的列。
20068	INSERT is not allowed since the compute node stops insert: worker=%s. Compute node message: %s.	COMPUTENODE设置了停止实时数据写入的标志，导致INSERT数据失败，请联系技术支持。
20069	[USER ERROR] Row expected exceeds limit.	查询语句的LIMIT子句（若不写，系统自动补上LIMIT子句，LIMIT默认值为10000）的值超过配置的上限，请限制并修改，或进一步联系技术支持。
20070	Only support single INSERT statement.	一次只能执行一个INSERT语句。
20071	执行INSERT FROM SELECT报错的详细信息。	请根据报错的详细信息，联系技术支持。
20072	Only TOP priority query is allowed.	只允许TOP优先级的查询，请联系技术支持。
20073	Exceeded max AND/OR combined predicate number:xxx	WHERE子句中AND/OR连接的谓词过滤条件数超过了允许的上限，请做限制。
20074	Reject execution of sql with key, statement: xxx	包含特定关键字的查询被拒绝执行，请确认并联系技术支持，确认这些关键已经被配置用来过滤SQL。
20075	[USER ERROR] all primary key columns value are NULL:	INSERT的数据记录中，不允许所有主键列全为NULL，请修改。
20076	[USER ERROR] sub partition keys count exceed the table limit for a duration (second): limit=xxx	在一个特定的周期内（默认为一天），目标表的INSERT数据的目标二级分区总数超限，默认为10个，请确认，或联系技术支持。
20077	[USER ERROR] delete statement count exceeds the table limit for a duration (second): limit=xxx duration=xxx	在一个特定的周期内（默认为一天），目标表的DELETE语句总数超限，默认为10000000，请确认，或联系技术支持。
20078	20078 No replica reported from COMPUTENODE.	MPP查询中COMPUTENODE节点未汇报路由信息，请联系技术支持。
20079	[USER ERROR] Total row expected exceeds limit + offset.	LIMIT m OFFSET n或者LIMIT n, m中，m + n超过了limitMax的查询限制，请确认修改查询LIMIT OFFSET的限制，或联系技术支持。
20080	Restart command is illegal. Please check its syntax.	Restart命令不合法，请检查该命令的语法。
20081	[USER ERROR] Insert record volume has exceeded the database limit for a duration(second): schema=%s totalRecordCount=%s recordlimit=%s duration=%s	在一个特定的周期内（默认为一天），目标DB的INSERT数据的总记录数超限，默认为2亿条 * COMPUTENODE节点数，请确认，或联系技术支持。
20082	[USER ERROR] Insert data size volume has exceeded the database limit for a duration(second): schema=%s totaDataSize=%s dataSizelimit=%s duration=%s	在一个特定的周期内（默认为一天），目标DB的INSERT数据（整个INSERT语句的字节数）的总大小超限，默认为100GB * COMPUTENODE节点数，请确认，或联系技术支持。
20091	[USER ERROR] column value length exceeds the limit: column=xxx	NA
20200	[USER ERROR] partition function exec error: xxx	二级分区列在元数据中不存在，请检查列名是否正确，或进一步联系技术支持。
20500	MPP_QUERY_CANCELED message=Query has been canceled!	MPP查询被CANCEL，请重试。

DML系统错误

错误码	错误信息	解决办法
30000	[SYSTEM ERROR] Execution timeout.	查询执行超时，请稍后重试，或联系技术支持。
30001	[SYSTEM ERROR] Dump service initialization error.	COMPUTENODE初始化TFS DUMP操作失败，请重试，或联系技术支持。
30002	[SYSTEM ERROR] Dump service write error.	COMPUTENODE执行TFS DUMP操作失败，请重试，或联系技术支持。
30003	[SYSTEM ERROR] Table hash partition count is invalid:	目标分区表的分区数非法，请联系技术支持。
30004	[SYSTEM ERROR] Target hash partition number is invalid:	INSERT语句计算分区列hash值时失败，请联系技术支持。
30005	[USER ERROR] sub partitioning column is invalid in the meta.	二级分区列在元数据中不存在，请检查列名是否正确，或进一步联系技术支持。

30006	Invalid column value:	列对应的值非法，请检查列值是否符合列的数据类型。
30007	RT_ROUTER_ERROR table=xxx DB=xxx message=real time table is not ready.	COMPUTENODE还未汇报目标实时表的版本心跳，请稍后重试，或进一步联系技术支持。
30008	实时数据相关功能调用BUFFERNODE API报的详细错误信息。	根据详细错误信息查看BUFFERNODE的错误码表，进一步联系技术支持。
30009	NA	NA
30010	[SYSTEM ERROR] Column data was not found.	COMPUTENODE处理下发查询时找不到目标表的某个列的元数据，请稍后重试，或进一步联系技术支持。
30011	[SYSTEM ERROR] Table hash partition count is invalid:	目标分区表的分区数非法，请联系技术支持。
30012	[SYSTEM ERROR] Target hash partition number is invalid:	已经不再出现该错误，如出现，请联系技术支持。
30013	Cluster nodes route were not established completely: A. system is starting, please wait; B. COMPUTENODEs were GCing or have already crashed.	集群服务还未启动完成，请稍等；或有部分COMPUTENODE实例GC，请联系技术支持。
30014	BROADCAST_SUBQUERY_TIMEOUT schema=xxx process=xxx query_block=xxx	小表广播模式查询的子查询超时，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
30015	BROADCAST_SUBQUERY_ERROR schema=xxx process=xxx query_block=xxx message=xxx	小表广播模式查询的子查询失败，请检查SQL，或进一步联系技术支持。
30015	Select localnode by resource, all unavailable=	NA
30016	CACHE_QUERY_FAILED message=xxx	Cache表查询失败，请检查SQL，或进一步联系技术支持。
30017	COMPUTENODE执行下发查询失败的详细信息。	请检查SQL，或进一步联系技术支持。
30018	CACHE_QUERY_FAILED message=xxx	Cache表查询失败，请检查SQL，或进一步联系技术支持。
30021	Query canceled in COMPUTENODE	查询被中断，请重试。
30022	Query exceeded scan limitation	查询超过扫描行数限制，请增加过滤条件缩小扫描范围，或进一步联系技术支持。
30023	Insert into buffernode failed	InsertIntoSelect插入到buffernode时报错，请重试，或进一步联系技术支持。
30024	Unknown serFormat. serFormat:	NA
30035	Query parameter is null.	NA
30036	NO_VALID_REPLICA part=xxx processId=xxx	NA
30037	NO_REPLICAS_TO_BE_SELECTED part=xxx processId=xxx xxxxx	NA
30100	HTTP client to MPP engine has gone.	MPP查询时，MPP client的HTTP连接断开，请重试，或进一步联系技术支持。
30101	MPP查询失败的详细信息。	请检查SQL，或进一步联系技术支持。
30102	MPP执行INSERT FROM SELECT失败的详细信息。	请检查INSERT FROM SELECT语句，或进一步联系技术支持。
30999	[SYSTEM ERROR] Other error.	其他查询错误，请联系技术支持。
31000	SQL planning partition routing error.	路由系统错误，请联系技术支持。
31000	ROUTER_ERROR table=xxx DB=xxx	NA
31001	ROUTER_ERROR table=xxx DB=xxx	NA
31002	SQL planning partition routing error: target hint partition ‘xxx’ was not found for the hint localnode ‘xxx’	NA
31003	SQL planning partition routing error: target hint partition ‘xxx’ was not found.	NA
32000	DATA_NOT_FOUND table_id=table_id db_id=db_id part_id=part_id	请检查目标表db_id.table_id是否上线完成。
32000	DATA_NOT_BUILD table=xxx DB=xxx	NA
32000	NO_COMPUTENODE_INSTANCE	NA
32000	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx processId=xxx 2ND_PICK_UP_BATCH.	NA
32000	DATA_NOT_FOUND No partition HB reported. table_id=xxx db_id=xxx part_id=xxx	NA
32001	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx processId=xxx FIRST_PART_REP_BATCH.	NA
32002	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx processId=xxx REMAINING_PART_REP_BATCH.	NA

32003	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx all part_replicas is empty.	NA
32004	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx this part_replica is empty.	NA
32007	RT_ROUTER_ERROR tableId=xxx dbId=xxx message=real time table is not ready.	NA
50000	QUERY_WARNING, server is not ready now.	FRONTNODE节点尚未启动完成，请稍后再发起查询。
55000	Partition merging error:	FRONTNODE节点进行分区结果合并操作异常，请联系技术支持。
60001	Must provide dump-header hint for cache table dump.	对CACHE表进行dump data操作时，必须通过/*+dump-header=[DUMP DATA …]*/ SELECT …的方式，请修改。
60002	Failed of getting upload id for xxx dump.	Dump data在获取upload id阶段失败，请参考详细信息，或联系技术支持。
60003	Must provide OSS dump parameters hints.	OSS dump必须在hint中提供OSS服务参数，请修改。
60004	No block ID available.	ODPS dump在get upload id阶段未获得block id，请联系技术支持。
60005	ODPS dump error. Message=xxx	OSS dump异常，查看详细异常消息，或联系技术支持。
60006	RT_DATA_WRITE_ERROR message=xxx	NA
60007	Insert failed with null result. xxxx	NA
60008	Insert failed with xxx data records xxx	NA
60008	Insert failed with xxx data records xxx	NA
60009	[USER ERROR] No values found.	NA

ACL相关错误码

范围	说明
18900 ~ 18999	ACL操作相关用户错误

ACL操作相关用户错误

错误码	错误信息	解决办法
18900	System privilege must equal with “.“	当ACL对象为System类型的resource，对象必须为`.`。
18901	Invalid resource type.	非法的ACL对象类型，必须为System或Table。
18902	xxx is not allowed, should be “ /db_name/db_name.table_name/db_name.table_group_name/table_name/table_group_name”	非法的ACL对象名，请按照提示修改。
18903	ACL对象不存在的详细提示信息。	ACL对象不存在，请确认ACL对象名是否正确。
18904	AclException: User[xxx] do not have[xxx] access to resource[xxx]	ACL鉴权失败的详细信息，请确认操作对象相关的ACL权限。
18905	Only support operation on connected db.	只允许在该数据库对应的数据库连接上操作，请检查数据库连接，或进一步联系技术支持。
18906	非法账号相关的详细信息。	账号非法，请确认是否为合法的阿里云ADS账号。
18907	Creator must be the DB Admin when creating database for delegated user!	通过委托模式创建目标库，创建账号必须是DB Admin账号，请确认并联系技术支持。
18908	Do not support specific ACL user @ target host:	GRANT不支持特定user@特定host的模式。
18909	ALB ACL was only supported under SYSDB connection.	ALB相关的黑白名单GRANT操作只能在SYSDB连接下进行。
18910	VIP/port was not found for this database: xxx; ALB load balancer instance was not found for VIP: xxx	ALB相关的黑白名单GRANT操作时，VIP/PORT或相关load balancer对象不存在，请确认并修改。
18911	‘ALL’ privilege is invalid on column, only ‘SELECT’ is allowed.	列上不支持ALL权限。
18912	RAMException: Deny for having no privilege.	RAM子账号不是ADS账号，请确认，或联系技术支持。
18913	RAMException: Unknown RAM code [xxx]	未识别的RAM操作返回码，请联系技术支持。
18914	RAMException: Invalid sub user, should be like ‘RAM$parent_user_account:sub_user_account’ or ‘RAM$sub_user_account’	RAM子账号格式错误，请按照提示修改。
18915	NA	NA
18916	RAMException: Sub user account does not match with parent.	RAM子账号与父账号不匹配，请联系技术支持。
18917	RAMException: Sub user account does not have xxx privilege.	用户无RAM子账号权限，请确认，或进行子账号赋权。

18918	RAMException: Sub user account does not allow xxx	RAM子账号ACL功能未开启，请联系技术支持。
18919	RAMException: Sub user account name is invalid.	RAM子账号在元数据中的值非法，请联系技术支持。
18920	The account of database owner is not allowed to be removed.	创建数据库的账号不能被删除。
18921	AclException:GRANT_ERROR	授权错误，请联系技术支持。
18922	AclException:REVOKE_ERROR	回收权限错误，请联系技术支持。

系统相关错误码

范围	说明
39900 ~ 39949	系统操作相关用户错误。
39950 ~ 39999	系统操作相关系统错误。

系统操作相关用户错误

错误码	错误信息	解决办法
39900	Wrong parameter for query, only SYNCCACHE [size=new_cache_size_in_MB] is allowed.	SYNCCACHE语句语法错误，请修改。
39901	Wrong parameter for query, CLEARCACHE command format is “CLEARCACHE db=schema tablegroup=table_group” or “CLEARCACHE db=schema table=table”	CLEARCACHE语句语法错误，请修改。
39902	Wrong parameter for query, FLUSH command format is “FLUSH db=schema table=table timeout=timeout_duration_ms”	FLUSH语句语法错误，请修改。
39903	Wrong parameter for query, MERGE command format is “MERGE db=schema table=table”	MERGE语句语法错误，请修改。
39904	No target table for OPTIMIZE TABLE command/Wrong OPTIMIZE TABLE command, syntax should be “OPTIMIZE TABLE [dbname.]table_name1 [, [dbname.]table_name2]”	OPTIMIZE TABLE语句语法错误，请修改。
39905	MPP引擎执行SHOW CATALOGS相关的详细错误信息	请联系技术支持。
39906	Resource Manager返回的详细错误信息，包含Resource Manager自身的错误码和详细描述。	查看Resource Manager错误码表，或进一步联系技术支持。
39907	TABLE_NOT_FOUND schema=xxx table=xxx	相关操作的目标表不存在，请检查。
39908	Target table is not realtime table.	相关实时表操作的目标表不是实时表，请检查。
39910	Cannot find worker db information.	无法找到系统后台WorkerDB，请联系技术支持。
39911	Show tables command is only allowed to show tables under current database.	该命令只允许显示当前DB下的表。

系统操作相关系统错误

错误码	错误信息	解决办法
39950	SYNCCACHE操作详细的错误信息。	请联系技术支持。
39951	CLEARCACHE操作详细的错误信息。	请联系技术支持。
39952	NA	请联系技术支持。
39953	NA	请联系技术支持。
39954	RT_UPN_FLUSH_ERROR message=target table does not exist./result=xxx DBID=xxx TableID=xxx PartitionCount=xxx	实时表FLUSH操作失败，请联系技术支持。
39955	FLUSH_TIMEOUT message=Flush version check timeout.	实时表FLUSH操作超时，实时数据强制版本同步慢，导致超时，请联系技术支持。
39956	Flush version check error:	实时表FLUSH操作版本确认失败，请联系技术支持。
39957	RT_UPN_FLUSH_ERROR db=schema table=table part=part_num message=Flush failed due to exception:	实时表FLUSH操作中，向BUFFERNODE写入目标版本信息失败，请联系技术支持。
39958	RT_UPN_ALTERTABLE_ERROR db=schema table=table part=part_num message=Alter table failed due to exception:	实时表FLUSH操作中，向BUFFERNODE写入ALTER指令失败，请联系技术支持。
39959	SLB操作详细错误信息。	查看SLB错误码表，或进一步联系技术支持。
39960	SLB操作详细错误信息。	查看SLB错误码表，或进一步联系技术支持。
39961	SLB操作详细错误信息。	查看SLB错误码表，或进一步联系技术支持。
39962	DNS_CMD_SYNTAX_ERRROR message=Incorrect DNS command syntax, the correct syntax should be “DNS [ADD | DELETE] domain ip”.	请联系技术支持。

39963	Delete DNS resolve record failed.	DNS命令执行失败，请联系技术支持。
39964	DNS_PARM_ERROR message=xxx	DNS命令执行失败，请联系技术支持。
39965	ALB_OPERATION_FAIL message=All load balancers are fully loaded. Please extend with new load balancers.	SLB的负载均衡实例全部满载（每个实例的VIP到上限），请联系技术支持进行负载均衡实例扩容。
39966	INSTANCE_VPC_DNS_PROCESS_ERROR	vpc vip申请错误，请联系技术支持。
39967	SHOW PLANCACHE STATUS执行失败的详细原因信息。	SHOW PLANCACHE STATUS执行失败，请联系技术支持。
39968	SHOW PLANCACHE PLAN执行失败的详细原因信息。	SHOW PLANCACHE PLAN执行失败，请联系技术支持。
39999	NA	请联系技术支持。

12.3. 系统运算符（2.0版）

本文介绍系统运算符的含义和用法。

逻辑运算符

运算符	描述	例子
AND	a和b两者都为true则返回True	a AND b
OR	a或b有一个为true就返回True	a OR b
NOT	a为false则返回True	NOT a

Null值对逻辑运算符的影响

当运算符AND的输入中包含FALSE时必然返回FALSE，否则如果输入中包含了NULL，则会返回NULL，示例如下。

```
SELECT CAST(null AS boolean) AND true; -- null

SELECT CAST(null AS boolean) AND false; -- false

SELECT CAST(null AS boolean) AND CAST(null AS boolean); -- null
```

当运算符OR的输入中包含TRUE时必然返回TRUE，否则如果输入中包含了NULL，则会返回NULL，示例如下。

```
SELECT CAST(null AS boolean) OR CAST(null AS boolean); -- null

SELECT CAST(null AS boolean) OR false; -- null

SELECT CAST(null AS boolean) OR true; -- true
```

下表详细列举了NULL参与AND和OR逻辑运算的结果。

a	b	a AND b	a OR b
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

NOT运算符在NULL上的操作示例如下：

```
SELECT NOT CAST(null AS boolean); -- null
```

下表详细列举了NULL参与NOT逻辑运算的结果。

a	NOT a
TRUE	FALSE
FALSE	TRUE
NULL	NULL

比较运算符

运算符	描述

<	小于
>	大于
<=	小于等于
>=	大于等于
=	等于
<>	不等于
!=	不等于（非标准语法）

区间运算符： BETWEEN

BETWEEN运算符用来判断值是否在区间里， 使用语法固定为 `value BETWEEN min AND max` ，示例如下：

```
SELECT 3 BETWEEN 2 AND 6;
```

上例中的查询语句等同于如下语句：

```
SELECT 3 >= 2 AND 3 <= 6;
```

判断值是否在区间外可以使用 `NOT BETWEEN` ，示例如下：

```
SELECT 3 NOT BETWEEN 2 AND 6;
```

上例中的查询语句等同于如下语句：

```
SELECT 3 < 2 OR 3 > 6;
```

在 `BETWEEN` or `NOT BETWEEN` 运算符中出现NULL都会使结果为NULL，示例如下：

```
SELECT NULL BETWEEN 2 AND 4; -- null

SELECT 2 BETWEEN NULL AND 6; -- null
```

`BETWEEN` and `NOT BETWEEN` 运算符同样可以作用于字符串型数据判断，示例如下：

```
SELECT 'Paul' BETWEEN 'John' AND 'Ringo'; -- true
```

必须注意的是， `BETWEEN` and `NOT BETWEEN` 运算符的输入数据类型必须相同。例如，判断 `'John' is between 2.3 and 35.2` 会使数据库报错。

IS NULL和IS NOT NULL

`IS NULL` 和 `IS NOT NULL` 运算符用于测试值是否为NULL（未定义）。这两个运算符对所有数据类型有效。

用 `IS NULL` 判断 `NULL` 值会返回 `TRUE` ：

```
select NULL IS NULL; -- true
```

但是其他任务非 `NULL` 值都会返回 `FALSE` ，示例如下：

```
SELECT 3.0 IS NULL; -- false
```

IS DISTINCT FROM和IS NOT DISTINCT FROM

在SQL语义中 `NULL` 代表一个未知的值，所以任何比较运算符在接收 `NULL` 输入时都会返回 `NULL` 。 `IS DISTINCT FROM` 和 `IS NOT DISTINCT FROM` 运算符会把 `NULL` 当成特殊的值， 当这两个运算符的输入中包含 `NULL` 时，它们仍然会返回True或者False，示例如下：

```
SELECT NULL IS DISTINCT FROM NULL; -- false

SELECT NULL IS NOT DISTINCT FROM NULL; -- true
```

在上面的例子中， 两个 `NULL` 值被认为是重复的。当你的比较运算数据中可能包含 `NULL` 值时，你可以使用这两个运算符来确保获得 `TRUE` 或者 `FALSE` 结果。

下表详细列举了 `NULL` 参与比较运算符时产生的结果。

a	b	a = b	a <> b	a DISTINCT b	a NOT DISTINCT b
1	1	TRUE	FALSE	FALSE	TRUE
1	2	FALSE	TRUE	TRUE	FALSE
1	NULL	NULL	NULL	TRUE	FALSE
NULL	NULL	NULL	NULL	FALSE	TRUE

GREATEST和LEAST

这两个函数不在标准的SQL语法中，是很常见的扩展。 和其他比较运算符一样，当函数的输入中出现 `NULL` 时函数返回结果为 `NULL` 。 在一些其他数据库，例如PostgreSQL中，只有当输入全部为 `NULL` 时才返回 `NULL` 。

比较函数支持以下数据类型： DOUBLE、BIGINT、VARCHAR、TIMESTAMP、TIMESTAMP WITH TIME ZONE、DATE

greatest(value1, value2, ... valueN)

返回参数中的最大值。

least(value1, value2, ... valueN)

返回参数中的最小值。

比较运算符支持的量词： ALL、ANY和SOME

`ALL` 、 `ANY` 、 `SOME` 0量词可以和比较运算符结合使用：

```
expression operator quantifier ( subquery )
```

例如：

```
SELECT 'hello' = ANY (VALUES 'hello', 'world'); -- true

SELECT 21 < ALL (VALUES 19, 20, 21); -- false

SELECT 42 >= SOME (SELECT 41 UNION ALL SELECT 42 UNION ALL SELECT 43); -- true
```

下表列举了比较运算符和量词组合的一些含义：

表达式	含义
A = ALL (...)	返回true当A等于所有匹配数据时。
A <> ALL (...)	返回true当A不等于所有匹配数据时。
A < ALL (...)	返回true当A小于所有匹配数据时。
A = ANY (...)	返回true当A等于任意某个匹配数据。 语义和 A IN (...) 相同。
A <> ANY (...)	返回true当A不等于任意某个匹配数据。
A < ANY (...)	返回true当A小于任意某个匹配数据。

`ANY` 和 `SOME` 含义相同，可以互换使用。

字符串运算符

运算符 `||` 完成字符串连接操作。

数学运算符

运算符	描述
+	加
-	减
*	乘
/	除（整形除法会截断）
%	模数（余数）

日期时间运算符

运算符	示例	结果
+	`date '2012-08-08' + interval '2' day`	`2012-08-10`
	`time '01:00' + interval '3' hour`	`04:00:00.000`
	`timestamp '2012-08-08 01:00' + interval '29' hour`	`2012-08-09 06:00:00.000`
	`timestamp '2012-10-31 01:00' + interval '1' month`	`2012-11-30 01:00:00.000`
	`interval '2' day + interval '3' hour`	`2 03:00:00.000`
	`interval '3' year + interval '5' month`	`3-5`
-	`date '2012-08-08' - interval '2' day`	`2012-08-06`
	`time '01:00' - interval '3' hour`	`22:00:00.000`
	`timestamp '2012-08-08 01:00' - interval '29' hour`	`2012-08-06 20:00:00.000`
	`timestamp '2012-10-31 01:00' - interval '1' month`	`2012-09-30 01:00:00.000`
	`interval '2' day - interval '3' hour`	`1 21:00:00.000`
	`interval '3' year - interval '5' month`	`2-7`

数组运算符

Subscript Operator： []

`[]` 操作符用来获取数组的某个元素：

```
SELECT my_array[1] AS first_element
```

Concatenation Operator: ||

`||` 操作符可以将相同类型的数组或元素连接：

```
SELECT ARRAY [1] || ARRAY [2]; -- [1, 2]
SELECT ARRAY [1] || 2; -- [1, 2]
SELECT 2 || ARRAY [1]; -- [2, 1]
```

MAP运算符

Subscript Operator: []

`[]` 运算符用于取出map中指定键的值：

```
SELECT name_to_age_map['Bob'] AS bob_age;
```

12.4. 元数据库数据字典（2.0版）

云原生数据仓库MySQL版的元数据库分为记载性能相关信息的performance_schema库和记载元数据的information_schema库，并和MySQL的元数据库有一定的兼容性，但并不是100%一致。

查询元数据库可以直接在JDBC连接中使用SQL语句进行查询。查询云原生数据仓库MySQL版中某表近期的10次数据导入的状态，示例如下：

```
SELECT state
FROM information_schema.current_job
WHERE table_schema='db_name'
  AND table_name='table_name'
ORDER BY start_time DESC
LIMIT 10
```

SCHEMATA

SCHEMATA表提供了关于数据库的信息。

FIELD	TYPE	ALLOW_NULL	PK	DEFAULT_VALUE	COMMENT
CLUSTER_NAME	varchar(16)	NO	无	NULL	集群名称
CATALOG_NAME	varchar(16)	YES	无	NULL	CATALOG名称
SCHEMA_NAME	varchar(64)	NO	无	NULL	SCHEMA名称
DEFAULT_CHARACTER_SET_NAME	varchar(64)	YES	无	UTF-8	默认字符集
DEFAULT_COLLATION_NAME	varchar(64)	YES	无	OFF	默认排序规则
id	int(11)	NO	PRI	NULL	ID
CREATOR_ID	varchar(64)	YES	无	NULL	创建者ID
CREATOR_NAME	varchar(64)	YES	无	NULL	创建者名称
SQL_PATH	varchar(255)	YES	无	NULL	SQL路径
DOMAIN_URL	varchar(255)	YES	无	NULL	域URL
QUERY_TYPE	varchar(64)	YES	无	NULL	查询类型
DISABLED	tinyint(1)	YES	无	0	禁用标记
LOAD_DISABLED	tinyint(1)	YES	无	0	禁止装载标记
TABLE_COUNT	int(11)	YES	无	NULL	表个数
TABLE_GROUP_COUNT	int(11)	YES	无	NULL	表组个数
MAX_TABLE_COUNT_LIMIT	int(11)	YES	无	256	最大表个数限制
MAX_TABLE_GROUP_COUNT_LIMIT	int(11)	YES	无	256	最大表组个数限制
QPS	int(11)	YES	无	-1	每秒查询数
LOAD_FACTOR	double	YES	无	-1	装载因子
INITIAL_CAPACITY	int(11)	YES	无	-1	初始化大小
INCREMENTAL_CAPACITY	int(11)	YES	无	-1	增量大小
DB_ASSIGN_STRATEGY	varchar(255)	YES	无	NULL	数据库分配策略
SHARED	tinyint(1)	YES	无	0	是否是共享实例
CREATE_TIME	timestamp	NO	无	CURRENT_TIMESTAMP	创建时间
UPDATE_TIME	timestamp	NO	无	0000-00-0000:00:00	修改时间

如果用户需要了解当前数据库的基本信息包括当前表的个数，表组的个数，最大表的限制，最大表组的限制，每秒查询数等信息的时候可以查询这张表。例如：

```
SELECT CLUSTER_NAME, SCHEMA_NAME, DEFAULT_CHARACTER_SET_NAME, DEFAULT_COLLATION_NAME, TABLE_COUNT, TABLE_GROUP_COUNT, MAX_TABLE_COUNT_LIMIT, MAX_TABLE_GROUP_COUNT_LIMIT, QPS FROM information_schema.schemata WHERE SCHEMA_NAME = 'xxx';
```

TABLES

TABLES表提供数据库表信息。该部分数据包括表的元数据与部分表对应数据的元数据，如分区信息等。

FIELD	TYPE	ALLOW_NULL	PK	DEFAULT_VALUE	COMMENT
CLUSTER_NAME	varchar(16)	NO	PRI	NULL	集群名称
TABLE_SCHEMA	varchar(128)	NO	PRI	NULL	所属SCHEMA的名称
TABLE_GROUP	varchar(128)	YES	无	NULL	所属表组的名称
TABLE_NAME	varchar(128)	NO	PRI	NULL	表名称
TABLE_SCHEMA_ID	varchar(128)	YES	无	NULL	所属SCHEMA的ID
TABLE_GROUP_ID	varchar(128)	YES	无	NULL	所属表组的ID
TABLE_ID	varchar(128)	YES	无	NULL	表的ID
TABLE_TYPE	varchar(128)	YES	无	NULL	标记：分区表PARTITION_TABLE、维度表DIMENSION_TABLE
UPDATE_TYPE	varchar(128)	YES	无	NULL	标记：批量表batch、实时表realtime
ONLINE_GROUP	varchar(128)	YES	无	NULL	在线分组
PARTITION_TYPE	varchar(128)	YES	无	NULL	分区类型：DIM、HASH
PARTITION_COLUMN	varchar(128)	YES	无	NULL	分区列名称
PARTITION_COUNT	int(11)	YES	无	NULL	分区数量
IS_SUB_PARTITION	tinyint(1)	YES	无	NULL	标记：是否是二级分区
SUB_PARTITION_TYPE	varchar(128)	YES	无	NULL	二级分区类型
SUB_PARTITION_COLUMN	varchar(128)	YES	无	NULL	二级分区列名称
SUB_PARTITION_COUNT	int(11)	YES	无	NULL	二级分区数量
CREATE_TIME	timestamp	NO	无	CURRENT_TIMESTAMP	创建时间
UPDATE_TIME	timestamp	NO	无	0000-00-0000:00:00	更新时间
CREATOR_ID	varchar(128)	YES	无	NULL	创建者ID
CREATOR_NAME	varchar(128)	YES	无	NULL	创建者名称
CURRENT_VERSION	bigint(20)	YES	无	NULL	当前版本
PREVIOUS_VERSION	bigint(20)	YES	无	NULL	上一版本
MIN_REDUNDANCY	int(11)	YES	无	2	最小副本数
COMMENTS	varchar(255)	YES	无	NULL	说明
CLUSTER_BY_COLUMNS	varchar(255)	YES	无	NULL	聚簇列名称
PRIMARY_KEY_COLUMNS	varchar(255)	YES	无	NULL	主键列名称
MAX_COLUMN_COUNT_LIMIT	int(11)	YES	无	990	列最多个数限制
EXECUTE_TIMEOUT	int(11)	YES	无	30000	执行超时时间
from_ctas	tinyint(1)	YES	无	NULL	是否创建于CTAS

- 如果用户需要了解某一个表的基本信息包括表的类型，一级分区键，二级分区键，分区类型，批量表或是实时表等信息的时候可以查询这张表。例如：

```
SELECT CLUSTER_NAME, TABLE_SCHEMA, TABLE_GROUP, TABLE_NAME, TABLE_TYPE, ONLINE_GROUP, PARTITION_TYPE, PARTITION_COLUMN, PARTITION_COUNT, SUB_PARTITION_TYPE, SUB_PARTITION_COLUMN, SUB_PARTITION_COUNT FROM information_schema.tables WHERE TABLE_SCHEMA = 'xxx' AND TABLE_GROUP = 'xxx' AND TABLE_NAME = 'xxx';
```

- 如果不知道具体的表名，只是希望了解某一个SCHEMA下有多少张不同类型的表，可以查询这张表。例如：

```
SELECT DISTINCT TABLE_GROUP, TABLE_NAME FROM information_schema.tables WHERE TABLE_SCHEMA = 'xxx';
```

- 如果希望了解某一个SCHEMA下面哪些表是维度表，哪些表是分区表，可以查询这张表。例如：

```
SELECT TABLE_GROUP, TABLE_NAME FROM information_schema.tables WHERE TABLE_SCHEMA = 'xxx' AND TABLE_TYPE = 'DIMENSION_TABLE';
```

- 如果希望了解某一个SCHEMA下面哪些表是批量表，哪些表是实时表，可以查询这张表。例如：

```
SELECT TABLE_GROUP, TABLE_NAME FROM information_schema.tables WHERE TABLE_SCHEMA = 'xxx' AND UPDATE_TYPE = 'realtime';
```

TABLE_GROUPS

该表存储了所有的表组的详细信息。

FIELD	TYPE	ALLOW_NULL	PK	DEFAULT_VALUE	COMMENT
CLUSTER_NAME	varchar(16)	NO	PRI	NULL	集群名称
TABLE_SCHEMA	varchar(64)	NO	PRI	NULL	所属SCHEMA
TABLE_GROUP	varchar(64)	NO	PRI	NULL	表组名称
TABLE_SCHEMA_ID	varchar(64)	YES	无	NULL	所属SCHEMA的ID
TABLE_GROUP_ID	varchar(64)	YES	无	NULL	所属表组的ID
CREATOR_ID	varchar(64)	YES	无	NULL	创建者ID
CREATOR_NAME	varchar(64)	YES	无	NULL	创建者名称
CREATE_TIME	timestamp	NO	无	CURRENT_TIMESTAMP	创建时间
UPDATE_TIME	timestamp	NO	无	0000-00-0000:00:00	更新时间
TABLE_COUNT	int(11)	YES	无	NULL	包含表的数量
MAX_TABLE_COUNT_LIMIT	int(11)	YES	无	256	最大表数量限制
MIN_REDUNDANCY	int(11)	YES	无	2	最小副本数
EXECUTE_TIMEOUT	bigint(20)	YES	无	30000	执行超时时间

- 如果希望了解某一个SCHEMA下面有多少个不同的表组，可以根据TABLE_SCHEMA查询TABLE_GROUP列。
- 如果希望了解某一个表组下包含了多少张表，可以根据TABLE_SCHEMA查询TABLE_COUNT列。

COLUMNS

该表存储了所有的表中字段的详细信息。

FIELD	TYPE	ALLOW_NULL	PK	DEFAULT_VALUE	COMMENT
CLUSTER_NAME	varchar(16)	NO	PRI	NULL	集群名称
TABLE_CATALOG	varchar(8)	YES	无	NULL	CATALOG名称
TABLE_SCHEMA	varchar(64)	NO	PRI	NULL	所属SCHEMA
TABLE_GROUP	varchar(64)	YES	无	NULL	所属表组
TABLE_NAME	varchar(64)	NO	PRI	NULL	所属表名
COLUMN_NAME	varchar(64)	NO	PRI	NULL	列名
ORDINAL_POSITION	int(11)	YES	无	NULL	在表中的位置
COLUMN_DEFAULT	varchar(255)	YES	无	NULL	默认列
IS_NULLABLE	tinyint(1)	YES	无	1	是否允许空
DATA_TYPE	int(11)	YES	无	NULL	数据类型名称
TYPE_NAME	varchar(64)	YES	无	NULL	列类型名称
CHARACTER_MAXIMUM_LENGTH	int(11)	YES	无	NULL	字符最大长度
CHARACTER_OCTET_LENGTH	int(11)	YES	无	NULL	字符八进制长度
NUMERIC_PRECISION	int(11)	YES	无	NULL	数值精度
NUMERIC_SCALE	int(11)	YES	无	NULL	数值范围
COLUMN_TYPE	varchar(64)	YES	无	NULL	列类型
COLUMN_COMMENT	varchar(255)	YES	无	NULL	列说明
DISABLE_INDEX	tinyint(1)	YES	无	0	是否禁用索引
DISABLE_DETAIL	tinyint(1)	YES	无	0	禁用细节
IS_PRIMARYKEY	tinyint(1)	YES	无	0	是否是主键
IS_PRESORT	tinyint(1)	YES	无	NULL	是否是预分类
PRESORT_ORDER	smallint(6)	YES	无	NULL	预分类订单
IS_VIRTUAL	tinyint(1)	YES	无	0	是否虚拟
IS_MULTIVALUED	tinyint(1)	YES	无	NULL	是否是多值列
DELIMITER	varchar(64)	YES	无	NULL	分隔符
IS_DELETED	tinyint(1)	YES	无	0	是否被删除

INDEXES	varchar(255)	YES	无	NULL	索引
CREATE_TIME	timestamp	NO	无	CURRENT_TIMESTAMP	创建时间
UPDATE_TIME	timestamp	NO	无	0000-00-0000:00:00	更新时间

如果希望了解某一个表包含的所有列信息，可以根据TABLE_CATALOG，TABLE_GROUP和TABLE_SCHEMA查询需要的列信息。

12.5. 18900~18999ACL操作相关用户错误（2.0版）

错误码	错误信息	解决办法
18900	System privilege must equal with “.“	当ACL对象为System类型的resource，对象必须为 . 。
18901	Invalid resource type.	非法的ACL对象类型，必须为System或Table。
18902	xxx is not allowed, should be “ /db_name/db_name.table_name/db_name.table_group_name/table_name/table_group_name”	非法的ACL对象名，请按照提示修改。
18903	ACL对象不存在的详细提示信息。	ACL对象不存在，请确认ACL对象名是否正确。
18904	AclException: User[xxx] do not have[xxx] access to resource[xxx]	ACL鉴权失败的详细信息，请确认操作对象相关的ACL权限。
18905	Only support operation on connectted db.	只允许在该数据库对应的数据库连接上操作，请检查数据库连接，或进一步联系技术支持。
18906	非法账号相关的详细信息。	账号非法，请确认是否为合法的阿里云ADS账号。
18907	Creator must be the DB Admin when creating database for delegated user!	通过委托模式创建目标库，创建账号必须是DB Admin账号，请确认并联系技术支持。
18908	Do not support specific ACL user @ target host:	GRANT不支持特定user@特定host的模式。
18909	ALB ACL was only supported under SYSDB connection.	ALB相关的黑白名单GRANT操作只能在SYSDB连接下进行。
18910	VIP/port was not found for this database: xxx; ALB load balancer instance was not found for VIP: xxx	ALB相关的黑白名单GRANT操作时，VIP/PORT或相关load balancer对象不存在，请确认并修改。
18911	‘ALL’ privilege is invalid on column, only ‘SELECT’ is allowed.	列上不支持ALL权限。
18912	RAMException: Deny for having no privilege.	RAM子账号不是ADS账号，请确认，或提交工单。
18913	RAMException: Unknown RAM code [xxx]	未识别的RAM操作返回码，请提交工单。
18914	RAMException: Invalid sub user, should be like ‘RAM$parent_user_account:sub_user_account’ or ‘RAM$sub_user_account’	RAM子账号格式错误，请按照提示修改。
18915	NA	NA
18916	RAMException: Sub user account does not match with parent.	RAM子账号与父账号不匹配，请确认。
18917	RAMException: Sub user account does not have xxx privilege.	用户无RAM子账号权限，请确认，或进行子账号赋权。
18918	RAMException: Sub user account does not allow xxx	RAM子账号ACL功能未开启，请提交工单。
18919	RAMException: Sub user account name is invalid.	RAM子账号在元数据中的值非法，请提交工单。
18920	The account of database owner is not allowed to be removed.	创建数据库的账号不能被删除。
18921	AclException:GRANT_ERROR	授权错误，请提交工单。
18922	AclException:REVOKE_ERROR	回收权限错误，请提交工单。

12.6. DDL相关错误码（2.0版）

12.6.1. 19600~19799DDL ALTER语句系统错误

本文介绍19600~19799DDL ALTER语句系统错误及解决方法。

错误码	错误信息	解决办法
19600	NA	NA
19601	ALTER statement is blocked.	DDL ALTER语句被禁止，请联系技术支持。
19602	ALTER DATABASE statement is disabled.	DDL ALTER DATABASE语句被禁止，请提交工单。
19603	ALTER TABLEGROUP statement is disabled for database ‘xxx’.	目标数据库的DDL ALTER TABLEGROUP语句被禁止，请提交工单。
19604	ALTER TABLE statement is disabled for database ‘xxx’.	目标数据库的DDL ALTER TABLE语句被禁止，请提交工单。
19605	Invalid FULLTEXT index column data type xxx of column xxx, VARCHAR was expected.	FULLTEXT全文索引列只支持VARCHAR类型，请修改。
19699	ALTER操作的其他类型失败的详细信息。	ALTER操作遇到其他类型失败，请联系技术支持。

12.6.2. 19000~19599DDL CREATE语句系统错误

本文介绍19000~19599DDL CREATE语句系统错误及解决办法。

错误码	错误信息	解决办法
19000	LOCK_SERVICE_ERROR message=Acquire lock failed.	DDL操作获取ZK锁失败，请稍后重试，或进一步联系技术支持。
19001	NA	NA
19002	NO_ALB_INSTANCE message=No available ALB load balancer instance found.	创建目标库时，未找到集群可用的load balancer实例，无法创建，请联系技术支持。
19003	NA	NA
19004	Illegal SLB VIP front end port: xxx, the valid port range is xxx-xxx	在元数据中指定VIP/PORT创建目标库时，指定的PORT不在合理范围内，请联系技术支持。
19005	ALB_OPERATION_FAIL message=xxx	创建目标库时，创建VIP步骤失败，请联系技术支持。
19006	Add DNS resolve record failed.	创建目标库时，绑定DNS域名步骤失败，请联系技术支持。
19007	NA	NA
19008	NA	NA
19009	NA	NA
19010	分配数据库对象ID失败的详细信息。	为创建目标数据库对象分配ID失败，请稍后重试，或进一步联系技术支持。
19011	CREATE statement is blocked.	DDL CREATE语句被禁止，请联系技术支持。
19012	CREATE DATABASE statement is disabled.	DDL CREATE DATABASE语句被禁止，请提交工单。
19013	CREATE TABLEGROUP statement is disabled for database ‘xxx’.	目标数据库的DDL CREATE TABLEGROUP语句被禁止，请提交工单。
19014	CREATE TABLE statement is disabled for database ‘xxx’.	目标数据库的DDL CREATE TABLE语句被禁止，请提交工单。
19015	CREATE EXTERNAL CATALOG statement is disabled for database ‘xxx’.	目标数据库的DDL CREATE EXTERNAL CATALOG语句被禁止，请提交工单。
19599	CREATE操作的其他类型失败的详细信息。	CREATE操作遇到其他类型失败，请联系技术支持。

12.6.3. 19800~19899DDL DROP语句系统错误

本文介绍19800~19899DDL DROP语句系统错误信息和解决方法。

错误码	错误信息	解决办法
19800	NA	NA
19801	DROP statement is blocked.	DDL DROP语句被禁止，请联系技术支持。
19802	DROP DATABASE statement is disabled.	DDL DROP DATABASE语句被禁止，请提交工单。
19803	DROP TABLEGROUP statement is disabled for database ‘xxx’.	目标数据库的DDL DROP TABLEGROUP语句被禁止，请提交工单。
19804	DROP TABLE statement is disabled for database ‘xxx’.	目标数据库的DDL DROP TABLE语句被禁止，请提交工单。
19805	DROP EXTERNAL CATALOG statement is disabled for database ‘xxx’.	目标数据库的DDL DROP EXTERNAL CATALOG语句被禁止，请提交工单。
19899	DROP操作的其他类型失败的详细信息。	DROP操作遇到其他类型失败，请联系技术支持。

12.6.4. 18600~18799DDL ALTER语句用户错误

本文介绍18600~18799DDL ALTER语句用户错误信息和解决方案。

错误码	错误信息	解决办法
18600	DenyAccessException:You do not have[xxx] access to resource[xxx]	无对特定数据库资源的特定操作权限，请确认，或按需要进行赋权限操作。
18601	NA	NA
18602	DDL语句语法错误的详细信息。	DDL语句语法错误，请参考DDL文档进行修改，或进一步联系技术支持。
18603	No database selected.	相关操作未找到目标数据库，请检查数据库连接是否正确包含目标数据库信息，或者操作是否指定目标数据库。
18604	目标数据库对象不存在的详细信息。	目标数据库对象不存在，请检查。
18605	参数值非法的详细错误信息。	DDL语句参数值非法，请按详细提示信息修改，或进一步联系技术支持。

18606	参数值非法的详细错误信息。	DDL语句参数值非法，请按详细提示信息修改，或进一步联系技术支持。
18607	AlterRepeatException: columnName: xxx repeat.	列名重复，请确认目标表无该名字的列。
18608	AlterRepeatException: indexName: xxx repeated.	索引目标列已经定义过索引，不能再添加其他索引。
18609	NA	NA
18610	Column type is invalid:	列数据类型非法，请参考DDL文档中关于支持的数据类型进行修改。
18611	DROP COLUMN is not supported yet.	暂不支持ALTER TABLE DROP COLUMN。
18612	Illegal index type:	不支持的索引类型，请参考DDL文档修改。
18613	Do not allow to add column to real time table:	暂不支持对实时表进行加列，请联系技术支持。
18614	Invalid default value for xxx	列定义中的默认值表达式非法，请修改。
18615	Add PK column is not allowed for real time table.	实时表不允许通过ALTER加主建列，请修改。
18616	Add virtual column is not allowed for real time table.	实时表不允许通过ALTER加虚拟列，请修改。
18617	非法的ALTER语句参数的详细信息。	ALTER语句参数非法，请根据提示参考DDL文档修改。
18618	Index “xxx” already exist on xxx.xxx	索引名重复，请确认目标表上无该名字的
18619	The operation is not supported yet.	不支持的ALTER操作，请联系技术支持。
18620	Resize Operation is not allowed in this database.	当前用户不允许对目标数据库进行变更资源操作，请联系技术支持。
18621	There are xxx sub partition number in definition, which is not in the valid range: x to x	二级分区数超过上限，请修改，或进一步联系技术支持。
18622	Target is not sub partition table.	目标表不是二级分区表，无法进行二级分区数调整操作。
18623	Illegal frontnode_rw_instance_ratio value: xxx, value pattern should be “number:number”	读写分离读写比例参数错误，请参考DDL文档修改。

12.6.5. 18000~18100DDL CREATE语句用户错误

本文介绍18000~18100DDL CREATE语句用户错误信息和解决办法。

错误码	错误信息	解决办法
18000	DenyAccessException:You do not have[xxx] access to resource[xxx]	无对特定数据库资源的特定操作权限，请确认，或按需要进行赋权限操作。
18001	Not support CREATE DATABASE in db: schema	无法在该数据连接上进行CREATE DATABASE操作，请检查所用数据库连接是否正确。
18002	非ADS user导致的失败信息。	操作必须由ADS user进行，请确认当前使用的UMM账号是ADS用户账号
18003	NA	NA
18004	Illegal options in CREATE DATABASE:	非法的CREATE DATABASE命令选项参数，请参考建库文档进行修改。
18005	NA	NA
18006	xxx database already exists.	目标数据库已经存在，请确认数据库是否重名。
18007	Target database does not exist.	目标数据库不存在，请确认数据库名是否正确。
18008	Table group ‘schema.tablegroup’ already exists.	目标表组不存在，请确认表组名是否正确。
18009	IllegalParameterException: parameterName: xxx, message: xxx	DDL语句参数不正确，请参考DDL文档，或进一步联系技术支持。
18010	参数值非法的详细错误信息。	DDL语句参数值非法，请按详细提示信息修改，或进一步联系技术支持。
18011	SUBPARTITION is not supported by DIMENSION table.	维度表不支持二级分区，请修改。
18012	TABLEGROUP must not be specified for DIMENSION table.	维度表建表语句不能指定表组，维度表均归属于系统默认维度表组，请修改。
18013	The minimum PARTITION NUM allowed for fact table is xxx, but xxx was defined.	不满足分区表的最小分区数定义，请修改。
18014	Table ‘table’ already exists.	目标表已经存在，请确认表是否重名。
18015	xxx is the dimension table group, could not be used.	自定义表组不能使用系统默认维度表组名，请修改。
18016	NA	NA
18017	Exceed the tables limitation (xxx) of database ‘xxx’	目标数据库下的表数量已经到上限，不可继续建表，请联系技术支持。
18018	Exceed the tables limitation (xxx) of table group ‘xxx’	目标数据库表组下的表数量已经到上限，不可继续建表，请联系技术支持。

18019	NA	NA
18020	Duplicated column definition	DDL中有重复列定义，请修改。
18021	There are xxx columns, which is not in the valid range: 1 to xxx	建表语句列数超限，请联系技术支持。
18022	Partition column does not exist	DDL中定义的分区列名不在定义的列中，请检查并修改。
18023	Column type is invalid	列数据类型非法，请参考DDL文档中关于支持的数据类型进行修改。
18024	DDL语句语法错误的详细信息。	DDL语句语法错误，请参考DDL文档进行修改，或进一步联系技术支持。
18025	No database selected.	相关操作未找到目标数据库，请检查数据库连接是否正确包含目标数据库信息，或者操作是否指定目标数据库。
18026	Table group should be created first.	建分区表时指定的表组不存在，需先建表组。
18027	相关命名的详细错误信息。	数据库对象命名错误，请按照提示进行修改，或进一步联系技术支持。
18028	Target sub-partition column does not exist.	指定的二级分区列列名不在定义的列中，请检查并修改。
18029	Only LONG/BIGINT is allowed for subpartition column data type.	二级分区列数据类型只能为LONG/BIGINT，请修改。
18030	Sub-partition number is illegal.	二级分区数非法，请修改。
18031	Exceed maximum user database limitation: xxx, user: xxx	用户下数据库总数已经达到上限，不可继续建库，请联系技术支持。
18032	No partition information provided for none dimension table.	建分区表时缺少分区信息子句，请参考DDL文档进行修改。
18033	Not support create table without tablegroup	建分区表时缺少表组子句，请参考DDL文档进行修改。
18034	Wrong table options	建表指定的options子句中参数错误，请参考DDL文档进行修改。
18035	PRIMARY KEY does not exist for real time table.	建实时表时缺少主建定义，请参考DDL文档进行修改。
18036	PRIMARY KEY does not contain partition column “xxx” for real time partition table.	建实时表时，主建定义中未包含分区列，请修改。
18037	Subpartition must not exist for real time table.	实时表暂不支持二级分区，请联系技术支持。
18038	NA	NA
18039	Table option LIFECYCLE is not allowed for non real time table.	建非实时表时不能指定LIFECYCLE参数，请修改。
18040	Old version of CREATE TABLE statement is blocked, please use the new version.	老版本建表语句被禁止，请参考DDL文档，使用当前的建表语句语法。
18041	CLUSTER BY column does not exist	建表语句中，CLUSTER BY指定的列不在定义的列中，请检查并修改。
18042	It is not allowed to create real time table on database: xxx	禁止在目标数据库中建实时表，请联系技术支持。
18043	Index column was not in the column definition list	索引列不在定义的列中，请检查并修改。
18044	Duplicated index name	DDL中有重复索引定义，请修改。
18045	The maximum PARTITION NUM allowed for fact table is xxx, but xxx was defined.	建分区表的分区数超过上限，请修改，或进一步联系技术支持。
18046	There are xxx sub partition number in definition, which is not in the valid range: x to x	二级分区数超过上限，请修改，或进一步联系技术支持。
18047	Invalid default value for xxx	列定义中的默认值表达式非法，请修改。
18048	MULTIVALUE/TIME/DATE is not allowed for partition column data type.	多值列、TIME、DATE类型的列不能作为分区列，请修改。
18049	Subpartition column must be one of the primary key columns.	建实时表时，二级分区列必须是主建列之一，请修改。
18050	相关功能还不支持的详细信息。	请参考提示的详细信息，或进一步联系技术支持。
18051	CTAS_LOAD_DATA_TIMEOUT schema=xxx table=xxx	CTAS执行的LOAD DATA阶段超时，请重试，或进一步联系技术支持。
18052	CTAS_META_CHECK_THREAD_ERROR message=xxx	CTAS执行的元数据校验阶段超时，请重试，或进一步联系技术支持。
18053	CTAS_LOAD_DATA_FAILED schema=xxx table=xxx jobState=xxx	CTAS执行的LOAD DATA阶段失败，请重试，或进一步联系技术支持。
18054	CTAS_SELECT_SQL_ANALYZE_ERROR schema=xxx table=xxx message=xxx	CTAS语句的SELECT子句语法分析失败，请检查语法，或进一步联系技术支持。
18055	CTAS_INSERT_THREAD_ERROR message=xxx	CTAS执行的INSERT阶段失败，请重试，或进一步联系技术支持。

18056	CTAS_INSERT_TIMEOUT schema=xxx table=xxx timeoutDuration=xxx	CTAS_INSERT_THREAD_ERROR message=xxx
18057	SUBPARTITION column conflicts with PARTITION column	二级分区列与一级分区列冲突，请修改。
18058	No column definition.	DDL语句无列定义，请参考DDL文档修改。
18059	Illegal table partition type: xxx, only HASH is allowed.	建分区表时，错误的分区类型，目前仅支持HASH分区类型，请修改。
18060	Illegal table sub partition type: xxx, only LIST is allowed.	建二级分区表时，错误的二级分区类型，目前仅支持LIST分区类型，请修改。
18061	Invalid index type: xxx	不支持的索引类型，请参考DDL文档修改。
18062	CTAS_RETRIEVE_COL_DEF_FAIL message=xxx	CTAS执行的列定义获取阶段失败，请检查SELECT部分的语法，或重试，或进一步联系技术支持。
18063	Illegal worker label value: xxx, value pattern should be “read:write”; Illegal worker ratio value: xxx, value pattern should be “number:number”	读写分离读写比例参数错误，请参考DDL文档修改。
18064	EXECUTE_CREATE_CACHE_TABLE_ERROR message=xxx query=xxx	建cache表失败，请参考DDL文档检查语法，或进行重试，或进一步联系技术支持。
18065	Lack options in CREATE DATABASE: xxx, options should be contained both ‘worker_labels’ and ‘worker_ratios’ or not.	建读写分离库时，读写分离参数错误，请根据提示和DDL文档修改。
18067	NA	NA
18068	Wrong format for external catalog properties.	外部数据源catalog创建语句的properties子句格式错误，请参考DDL文档修改。
18069	xxx external catalog already exists.	外部数据源catalog已经存在，请确认catalog是否重名。
18070	NA	NA
18071	Could not use the existing schema name	外部数据源catalog名字不能和已存在的数据库重名，请修改。
18072	Table option UPDATETYPE is unknown. Only ‘realtime’ and ‘batch’ are supported.	表选项UPDATETYPE只支持realtime和batch，请修改。
18073	Invalid FULLTEXT index column data type xxx of column xxx, VARCHAR was expected.	FULLTEXT全文索引列只支持VARCHAR类型，请修改。
18074	Database name length exceeds the limitation of ‘xxx’, current length is ‘xxx’.	CREATE DATABASE指定的目标数据库名长度超限，请修改。
18075	Illegal ‘dbNameBytesMaxLength’ zk value: xxx, value type should be numeric.	CREATE DATABASE的目标数据库名长度限制ZK配置数据类型错误，请提工单。
18076	Target view ‘xxx’ already exist.	CREATE VIEW的目标视图已经存在，请确认或修改。
18077	Table with the same name ‘xxx.xxx’ already exists.	CREATE VIEW的目标视图名和已经存在表名冲突，请确认或修改。
18100	Realtime table count exceeds limit in this database: xxx, ecu type: xxx	目标数据库实时表数量超过了ecu的上限，无法继续建实时表，请联系技术支持。

12.6.6. 18800~18899DDL DROP语句用户错误

本文介绍18800~18899DDL DROP语句用户错误信息和解决办法。

错误码	错误信息	解决办法
18800	No database selected.	相关操作未找到目标数据库，请检查数据库连接是否正确包含目标数据库信息，或者操作是否指定目标数据库。
18801	Database ‘xxx’ was not found.	目标数据库不存在，请确认。
18802	Can not drop non-empty database.	不能DROP非空的数据库，先DROP完库中的表，之后再DROP该库。
18803	Could not explicitly drop dimension group ‘xxx’.	不能DROP系统默认的维度表组。
18804	Target table group does not exist:	目标表组不存在，请确认。
18805	Can not drop non-empty tablegroup.	不能DROP非空的表组，先DROP完表组中的表，之后再DROP该表组。
18806	Illegal drop target type, only DATABASE/TABLEGROUP/TABLE/EXTERNAL CATALOG is allowed.	DROP的目标数据库对象类型非法，只允许DROP DATABASE/TABLEGROUP/TABLE/EXTERNAL CATALOG。
18807	Target table does not exist:	目标表不存在，请确认。
18808	Drop failed. xxx message=xxx	删除cache表失败，请重试，或进一步联系技术支持。
18809	Drop external catalog failure due to:	删除外部数据源catalog失败，请重试，或进一步联系技术支持。

12.7. DML相关错误码（2.0版）

12.7.1. 30000~60009DML系统错误

本文介绍30000~60009DML系统错误信息和解决办法。

错误码	错误信息	解决办法
30000	[SYSTEM ERROR] Execution timeout.	查询执行超时，请稍后重试，或联系技术支持。
30001	[SYSTEM ERROR] Dump service initialization error.	COMPUTENODE初始化TFS DUMP操作失败，请重试，或联系技术支持。
30002	[SYSTEM ERROR] Dump service write error.	COMPUTENODE执行TFS DUMP操作失败，请重试，或联系技术支持。
30003	[SYSTEM ERROR] Table hash partition count is invalid:	目标分区表的分区数非法，请联系技术支持。
30004	[SYSTEM ERROR] Target hash partition number is invalid:	INSERT语句计算分区列hash值时失败，请联系技术支持。
30005	[USER ERROR] sub partitioning column is invalid in the meta.	二级分区列在元数据中不存在，请检查列名是否正确，或进一步联系技术支持。
30006	Invalid column value:	列对应的值非法，请检查列值是否符合列的数据类型。
30007	RT_ROUTER_ERROR table=xxx DB=xxx message=real time table is not ready.	COMPUTENODE还未汇报目标实时表的版本心跳，请稍后重试，或进一步联系技术支持。
30008	实时数据相关功能调用BUFFERNODE API报的详细错误信息。	根据详细错误信息查看BUFFERNODE的错误码表，进一步联系技术支持。
30009	NA	NA
30010	[SYSTEM ERROR] Column data was not found.	COMPUTENODE处理下发查询时找不到目标表的某个列的元数据，请稍后重试，或进一步联系技术支持。
30011	[SYSTEM ERROR] Table hash partition count is invalid:	目标分区表的分区数非法，请联系技术支持。
30012	[SYSTEM ERROR] Target hash partition number is invalid:	已经不再出现该错误，如出现，请联系技术支持。
30013	Cluster nodes route were not established completely: A. system is starting, please wait; B. COMPUTENODEs were GCing or have already crashed.	集群服务还未启动完成，请稍等；或有部分COMPUTENODE实例GC，请联系技术支持。
30014	BROADCAST_SUBQUERY_TIMEOUT schema=xxx process=xxx query_block=xxx	小表广播模式查询的子查询超时，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
30015	BROADCAST_SUBQUERY_ERROR schema=xxx process=xxx query_block=xxx message=xxx	小表广播模式查询的子查询失败，请检查SQL，或进一步联系技术支持。
30015	Select localnode by resource, all unavailable=	
30016	CACHE_QUERY_FAILED message=xxx	Cache表查询失败，请检查SQL，或进一步联系技术支持。
30017	COMPUTENODE执行下发查询失败的详细信息。	请检查SQL，或进一步联系技术支持。
30018	CACHE_QUERY_FAILED message=xxx	Cache表查询失败，请检查SQL，或进一步联系技术支持。
30021	Query canceled in COMPUTENODE	查询被中断，请重试。
30022	Query exceeded scan limitation	查询超过扫描行数限制，请增加过滤条件缩小扫描范围，或进一步联系技术支持。
30023	Insert into buffernode failed	InsertIntoSelect插入到buffernode时报错，请重试，或进一步联系技术支持。
30024	Unknown serFormat. serFormat:	
30035	Query parameter is null.	
30036	NO_VALID_REPLICA part=xxx processId=xxx	
30037	NO_REPLICAS_TO_BE_SELECTED part=xxx processId=xxx xxxxx	
30100	HTTP client to MPP engine has gone.	MPP查询时，MPP client的HTTP连接断开，请重试，或进一步联系技术支持。
30101	MPP查询失败的详细信息。	请检查SQL，或进一步联系技术支持。
30102	MPP执行INSERT FROM SELECT失败的详细信息。	请检查INSERT FROM SELECT语句，或进一步联系技术支持。
30999	[SYSTEM ERROR] Other error.	其他查询错误，请联系技术支持。
31000	SQL planning partition routing error.	路由系统错误，联系技术支持。
31000	ROUTER_ERROR table=xxx DB=xxx	
31001	SQL planning partition routing error: no partition was found for the hint localnode	

31002	SQL planning partition routing error: target hint partition ‘xxx’ was not found for the hint localnode ‘xxx’	
31003	SQL planning partition routing error: target hint partition ‘xxx’ was not found.	
32000	DATA_NOT_FOUND table_id=table_id db_id=db_id part_id=part_id	请检查目标表db_id.table_id是否上线完成。
32000	DATA_NOT_BUILD table=xxx DB=xxx	
32000	NO_COMPUTENODE_INSTANCE	
32000	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx processId=xxx 2ND_PICK_UP_BATCH.	
32000	DATA_NOT_FOUND No partition HB reported. table_id=xxx db_id=xxx part_id=xxx	
32001	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx processId=xxx FIRST_PART_REP_BATCH.	
32002	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx processId=xxx REMAINING_PART_REP_BATCH.	
32003	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx all part_replicas is empty.	
32004	DATA_NOT_FOUND table_id=xxx db_id=xxx part_id=xxx this part_replica is empty.	
32007	RT_ROUTER_ERROR tableId=xxx dbId=xxx message=real time table is not ready.	
50000	QUERY_WARNING, server is not ready now.	FRONTNODE节点尚未启动完成，请稍后再发起查询。
55000	Partition merging error:	FRONTNODE节点进行分区结果合并操作异常，请联系技术支持。
60001	Must provide dump-header hint for cache table dump.	对CACHE表进行dump data操作时，必须通过/*+dump-header=[DUMP DATA ...]*/ SELECT ...的方式，请修改。
60002	Failed of getting upload id for xxx dump.	Dump data在获取upload id阶段失败，请参考详细信息，或进一步提工单。
60003	Must provide OSS dump parameters hints.	OSS dump必须在hint中提供OSS服务参数，请修改。
60004	No block ID available.	ODPS dump在get upload id阶段未获得block id，请提工单。
60005	ODPS dump error. Message=xxx	OSS dump异常，查看详细异常消息，或进一步提工单。
60006	RT_DATA_WRITE_ERROR message=xxx	
60007	Insert failed with null result. xxxx	
60008	Insert failed with xxx data records xxx	
60008	Insert failed with xxx data records xxx	
60009	[USER ERROR] No values found.	

12.7.2. 0~ 20999DML用户错误

本文介绍0~ 20999DML用户错误信息和解决办法。

错误码	错误信息	解决办法
1044	Can not use this command here !	不支持的MySQL协议命令，请确认。
1045	其他语句执行异常	请提交工单。
1046	No database specified in FROM table	查询未能找到目标表的归属DB，请检查数据库连接使用的DB，或者直接为目标表前面加上DB前缀。
1143	Access deny for accessing database ‘schema’ from this node.	请联系技术支持，确认是否进行了DB节点绑定操作。
1146	Table xxx doesn’t exist.	请确认目标表xxx在当前连接和访问的DB下确实存在。
1147	No COLUMN:column found in TABLE table	请确认目标表table确实包含列column。
1148	Column xxx does not exist.	查询语句中Column分析的相关错误请联系技术支持查看执行的SQL。
1235	SQL where expression contains/in items count exceed limit:	1.查询语句中LIMIT子句值超过配置允许的最大值，请修改； 2. 查询语句中CONTAINS/IN子句中的项目数超过配置允许的最大值，请修改减少项目数。
1236	SQL feature NOT supported yet: COUNT(DISTINCT xx) without join tables’ partition columns	查询语句中xxx相关语法功能不支持。

1236	SQL feature NOT supported yet: SELECT * or SELECT <table>.* with UNION/INTERSECT/MINUS.	无
1236	SQL feature NOT supported yet: SELECT agg_func() ... UNION/UNION ALL/INTERSECT/MINUS SELECT agg_func() ...	无
1236	SQL feature NOT supported yet: SELECT ... LIMIT X UNION/UNION ALL/INTERSECT/MINUS SELECT ... LIMIT X	无
1236	SQL feature NOT supported yet: FULL JOIN	无
1236	SQL feature NOT supported yet: RIGHT JOIN	无
1236	SQL feature NOT supported yet: SELECT (A JOIN B ...)	无
1236	SQL feature NOT supported yet: xxx	无
20000	[USER ERROR] Invalid SQL.	SQL语法异常，请检查并修改，或进一步联系技术支持。
20000	[USER ERROR] Invalid SQL.	SQL语法异常，请检查并修改，或进一步联系技术支持。
20001	[USER ERROR] Invalid data value type.	SQL中有数据的值和其对应的数据列类型不匹配，请根据提示修改。
20002	[USER ERROR] Invalid hint.	下发到COMPUTENODE的查询HINT非法，请检查SQL，或进一步联系技术支持。
20003	[USER ERROR] Invalid UDF.	COMPUTENODE不支持的UDF，请确认，或进一步联系技术支持。
20004	[USER ERROR] Query items exceed limitation.	COMPUTENODE计算超限，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20005	[USER ERROR] Query must GROUP BY column.	SQL语句错误，按照提示，必须GROUP BY提示的列。
20006	[USER ERROR] Query misses join condition.	SQL语句错误，join子句缺少ON条件。
20007	[USER ERROR] Query misses join index.	已经不再出现该错误，如出现，请联系技术支持。
20008	NA	NA
20009	[USER ERROR] Unsupported query syntax.	SQL语法错误，请参考SQL语法文档，进行修改。
20010	Parsing insert statement failed.	NA
20011	[USER ERROR] fact table partition column was not provided for INSERT statement.	目标表为分区表时，INSERT语句的列集合中，必须包含分区列。
20012	[USER ERROR] Invalid column value class type (fastsql xxx	非法的列值，列值和列类型不匹配，请修改。
20013	INSERT语句语法错误的详细信息。	INSERT语句语法错误，请按照提示修改。
20014	INSERT statement is blocked.	INSERT语句被禁止，请联系技术支持。
20015	RT_UPN_HANDLER_INVALID message=Data buffer handler was not found.	INSERT执行失败的详细信息。请重试，或进一步联系技术支持。
20016	[USER ERROR] Target table was not found at this time.	已经不再出现该错误，如出现，请联系技术支持。
20017	Delete statement is blocked.	DELETE语句被禁止，请联系技术支持。
20018	DELETE语句语法错误的详细信息。	DELETE语句语法错误，请按照提示修改。
20019	DELETE执行失败的详细信息。	请重试，或进一步联系技术支持。
20020	[USER ERROR] Target table was not found at this time.	已经不再出现该错误，如出现，请联系技术支持。
20021	[USER ERROR] Target table has no primary key.	目标实时表没有主建列，请联系技术支持。
20022	[USER ERROR] Must and only contains AND joined EQUAL-TO predicates of all columns within the primary key:	严格的DELETE where条件检查，请联系技术支持更改系统配置。
20023	[USER ERROR] column was not found: column=xxx	INSERT语句中，目标列不存在，请修改。
20024	[USER ERROR] column was not found: column=	严格的DELETE where条件检查时，DELETE语句中，目标列不存在，请修改，并联系技术支持更改系统配置。
20025	[USER ERROR] column value is invalid: column=xxx, type=xxx, value=xxx	严格的DELETE where条件检查时，DELETE语句中，目标列的值和类型不匹配，请修改，并联系技术支持更改系统配置。
20026	Real time data FLUSH is blocked.	FLUSH语句被禁止，请联系技术支持。
20027	Real time data MERGE is blocked.	MERGE语句被禁止，请联系技术支持。
20028	INSERT is not allowed when the real time table is not ready.	已经不再出现该错误，如出现，请联系技术支持。
20029	DELETE is not allowed when the real time table is not ready.	已经不再出现该错误，如出现，请联系技术支持。
20030	[USER ERROR] Target table is not real time table.	目标表不是实时表，无法INSERT。

20031	[USER ERROR] Target table is not real time table.	目标表不是实时表，无法INSERT。
20032	[USER ERROR] column value lists do not match.	INSERT语句中列集合的列表和VALUES子句中值的个数不匹配，请确认修改。
20033	Insert failed due to encoding exception:	INSERT语句中数据编码有问题，请确认，或进一步联系技术支持。
20034	[USER ERROR] Target table has no primary key.	目标实时表没有主建列，请联系技术支持。
20035	[USER ERROR] miss primary key column:	INSERT语句的列集合未包含所有的主建列，请修改。
20036	RT_DATA_WRITE_TIMEOUT schema=xxx table=xxx	INSERT执行超时，请重试，或进一步联系技术支持。
20037	[USER ERROR] null value is not allowed for NOT NULL column: column=xxx	INSERT语句中，NOT NULL的列插入了NULL值，请修改。
20038	[USER ERROR] Illegal argument.	COMPUTENODE执行下发SQL时出现参数错误，请参考SQL语法文档，或进一步联系技术支持。
20039	[USER ERROR] Unsupported query operation.	COMPUTENODE执行下发SQL时出现语法错误，请参考SQL语法文档，或进一步联系技术支持。
20040	No database selected.	请检查是否在数据库连接中指定了目标数据库。
20041	No database selected.	请检查是否在数据库连接中指定了目标数据库。
20042	COUNT DISTINCT on non-partitioning column exceeds the partition data limitation:	进行非分区列COUNT DISTINCT操作时，单分区返回的明细数据超过了限制，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20043	UDF_SYS_ROWCOUNT exceeds the partition data limitation:	SQL中包含UDF_SYS_ROWCOUNT函数时，单分区返回的明细数据超过了限制，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20044	[USER ERROR] Must contains WHERE expression.	DELETE语句中未写WHERE子句，请修改。
20045	[USER ERROR] sub select condition was not support for delete statement.	已经不再出现该错误，如出现，请联系技术支持。
20046	SQL feature NOT supported yet: GROUP_CONCAT without grouping tables' partition columns.	使用group_concat函数时，order by不起作用。
20047	SQL feature NOT supported yet: COUNT(DISTINCT x1, x2, ...) against mulitple non-partitioning columns	不支持对多个非分区列的COUNT DISTINCT操作。
20048	Unsupported expression in HAVING: xxx	HAVING子句表达式非法的详细信息。请参考SQL文档进行修改。
20049	SQL dump data selecting columns count exceed limit:	DUMP DATA语句中的SELECT列数超过上限，请限制并修改，或进一步联系技术支持。
20050	SQL selecting columns count exceed limit:	查询语句中的SELECT列数超过上限，请限制并修改，或进一步联系技术支持。
20051	NA	NA
20052	UDF_SYS_SUM exceeds the partition data limitation:	SQL中包含UDF_SYS_SUM函数时，单分区返回的明细数据超过了限制，请优化SQL，根据业务场景提高SQL的筛选率，或进一步联系技术支持。
20053	INSERT is not allowed since the compute node disk is almost full: worker=%s disk_used=%s disk_size=%s usage_ratio=%s	COMPUTENODE实例节点存储空间到上限，无法继续INSERT数据，请联系技术支持。
20054	Unknown column xxx in 'ORDER BY clause'	ORDER BY子句中的列非法，请检查SQL并修改。
20055	SQL feature NOT supported yet: SELECT ... ORDER BY ... UNION/UNION ALL/INTERSECT/MINUS SELECT	不支持的UNION语法：UNION子查询中带ORDER BY子句。
20056	Unknown column xxx in 'GROUP BY clause'	GROUP BY子句中的列非法，请检查SQL并修改。
20057	NA	NA
20058	RT_UPN_ALTERTABLE_ERROR db=xxx table=xxx part=xxx message=xxx	向BUFFERNODE发ALTER TABLE语句时，遇到encoding异常，请联系技术支持。
20059	[USER ERROR] null value is not allowed for sub partitioning column: column=xxx	INSERT实时数据时，二级分区列对应的值不能为NULL，请修改。
20060	[USER ERROR] sub partitioning column value is out of range: column=xxx	INSERT实时数据时，二级分区列对应的值不在合法的范围，请修改。
20061	[USER ERROR] sub partitioning column value is invalid: column=	INSERT实时数据时，二级分区列对应的值不合法，请修改。
20062	[USER ERROR] sub partition column was not provided for INSERT statement.	INSERT实时数据时，如果目标表是二级分区表，插入的列和值的集合必须包含二级分区列，请修改。
20063	[USER ERROR] sub partitioning column value is out of range: column=xxx, type=LONG/BIGINT, value=xxx	已经不再出现该错误，如出现，请联系技术支持。
20064	[USER ERROR] sub partitioning column value is invalid: column=xxx, type=LONG/BIGINT, value=xxx	已经不再出现该错误，如出现，请联系技术支持。

20065	[USER ERROR] sub partitioning column EQUAL-TO predicate was not provided.	已经不再出现该错误，如出现，请联系技术支持。
20066	ORDER BY item 'xxx' was not found in GROUP BY clause.	查询语句包含GROUP BY和ORDER BY子句时，ORDER BY中的列必须出现在GROUP BY子句中。
20067	SQL GROUP-BY column expected:xxx	SQL语句错误，按照提示，必须GROUP BY提示的列。
20068	INSERT is not allowed since the compute node stops insert: worker=%s. Compute node message: %s.	COMPUTENODE设置了停止实时数据写入的标志，导致INSERT数据失败，请联系技术支持。
20069	[USER ERROR] Row expected exceeds limit.	查询语句的LIMIT子句（若不写，系统自动补上LIMIT子句，LIMIT默认值为10000）的值超过配置的上限，请限制并修改，或进一步联系技术支持。
20070	Only support single INSERT statement.	一次只能执行一个INSERT语句。
20071	执行INSERT FROM SELECT报错的详细信息。	请根据报错的详细信息，联系技术支持。
20072	Only TOP priority query is allowed.	只允许TOP优先级的查询，请联系技术支持。
20073	Exceeded max AND/OR combined predicate number:xxx	WHERE子句中AND/OR连接的谓词过滤条件数超过了允许的上限，请做限制。
20074	Reject execution of sql with key, statement: xxx	包含特定关键字的查询被拒绝执行，请确认并联系技术支持，确认这些关键已经被配置用来过滤SQL。
20075	[USER ERROR] all primary key columns value are NULL:	INSERT的数据记录中，不允许所有主键列全为NULL，请修改。
20076	[USER ERROR] sub partition keys count exceed the table limit for a duration (second): limit=xxx	在一个特定的周期内（默认为一天），目标表的INSERT数据的目标二级分区总数超限，默认为10个，请确认，或进一步提工单。
20077	[USER ERROR] delete statement count exceeds the table limit for a duration (second): limit=xxx duration=xxx	在一个特定的周期内（默认为一天），目标表的DELETE语句总数超限，默认为10000000，请确认，或进一步提工单。
20078	20078 No replica reported from COMPUTENODE.	MPP查询中COMPUTENODE节点未汇报路由信息，请提工单。
20079	[USER ERROR] Total row expected exceeds limit + offset.	LIMIT m OFFSET n或者LIMIT n, m中，m + n超过了limitMax的查询限制，请确认修改查询LIMIT OFFSET的限制，或进一步提工单。
20080	Restart command is illegal. Please check its syntax.	Restart命令不合法，请检查该命令的语法。
20081	[USER ERROR] Insert record volume has exceeded the database limit for a duration(second): schema=%s totalRecordCount=%s recordlimit=%s duration=%s	在一个特定的周期内（默认为一天），目标DB的INSERT数据的总记录数超限，默认为2亿条 * COMPUTENODE节点数，请确认，或进一步提工单。
20082	[USER ERROR] Insert data size volume has exceeded the database limit for a duration(second): schema=%s totaDataSize=%s dataSizelimit=%s duration=%s	在一个特定的周期内（默认为一天），目标DB的INSERT数据（整个INSERT语句的字节数）的总大小超限，默认为100GB * COMPUTENODE节点数，请确认，或进一步提工单。
20091	[USER ERROR] column value length exceeds the limit: column=xxx	无
20200	[USER ERROR] partition function exec error: xxx	二级分区列在元数据中不存在，请检查列名是否正确，或进一步联系技术支持。
20500	MPP_QUERY_CANCELED message=Query has been canceled!	MPP查询被CANCEL，请重试。

12.8. 系统相关错误码（2.0版）

12.8.1. 39950 ~ 39999 系统操作相关系统错误

错误码	错误信息	解决办法
39950	SYNCCACHE操作详细的错误信息。	请联系技术支持。
39951	CLEARCACHE操作详细的错误信息。	请联系技术支持。
39952		请联系技术支持。
39953		请联系技术支持。
39954	RT_UPN_FLUSH_ERROR message=target table does not exist./result=xxx DBID=xxx TableID=xxx PartitionCount=xxx	实时表FLUSH操作失败，请联系技术支持。
39955	FLUSH_TIMEOUT message=Flush version check timeout.	实时表FLUSH操作超时，实时数据强制版本同步慢，导致超时，请联系技术支持。
39956	Flush version check error:	实时表FLUSH操作版本确认失败，请联系技术支持。
39957	RT_UPN_FLUSH_ERROR db=schema table=table part=part_num message=Flush failed due to exception:	实时表FLUSH操作中，向BUFFERNODE写入目标版本信息失败，请联系技术支持。

错误码	错误信息	解决办法
39958	RT_UPN_ALTERTABLE_ERROR db=schema table=table part=part_num message=Alter table failed due to exception:	实时表FLUSH操作中，向BUFFERNODE写入ALTER指令失败，请联系技术支持。
39959	SLB操作详细错误信息。	查看SLB错误码表，或进一步联系技术支持。
39960	SLB操作详细错误信息。	查看SLB错误码表，或进一步联系技术支持。
39961	SLB操作详细错误信息。	查看SLB错误码表，或进一步联系技术支持。
39962	DNS_CMD_SYNTAX_ERRROR message=Incorrect DNS command syntax, the correct syntax should be “DNS [ADD | DELETE] domain ip”.	请联系技术支持。
39963	Delete DNS resolve record failed.	DNS命令执行失败，请联系技术支持。
39964	DNS_PARM_ERROR message=xxx	DNS命令执行失败，请联系技术支持。
39965	ALB_OPERATION_FAIL message=All load balancers are fully loaded. Please extend with new load balancers.	SLB的负载均衡实例全部满载（每个实例的VIP到上限），请提交工单进行负载均衡实例扩容。
39966	INSTANCE_VPC_DNS_PROCESS_ERROR	vpc vip申请错误，请联系技术支持。
39967	SHOW PLANCACHE STATUS执行失败的详细原因信息。	SHOW PLANCACHE STATUS执行失败，请联系技术支持。
39968	SHOW PLANCACHE PLAN执行失败的详细原因信息。	SHOW PLANCACHE PLAN执行失败，请联系技术支持。
39999		请联系技术支持。

12.8.2. 39900 ~ 39949 系统操作相关用户错误

错误码	错误信息	解决办法
39900	Wrong parameter for query, only SYNCCACHE [size=new_cache_size_in_MB] is allowed.	SYNCCACHE语句语法错误，请修改。
39901	Wrong parameter for query, CLEARCACHE command format is “CLEARCACHE db=schema tablegroup=table_group” or “CLEARCACHE db=schema table=table”	CLEARCACHE语句语法错误，请修改。
39902	Wrong parameter for query, FLUSH command format is “FLUSH db=schema table=table timeout=timeout_duration_ms”	FLUSH语句语法错误，请修改。
39903	Wrong parameter for query, MERGE command format is “MERGE db=schema table=table”	MERGE语句语法错误，请修改。
39904	No target table for OPTIMIZE TABLE command/Wrong OPTIMIZE TABLE command, syntax should be “OPTIMIZE TABLE [dbname.]table_name1 [, [dbname.]table_name2]”	OPTIMIZE TABLE语句语法错误，请修改。
39905	MPP引擎执行SHOW CATALOGS相关的详细错误信息	请联系技术支持。
39906	Resource Manager返回的详细错误信息，包含Resource Manager自身的错误码和详细描述。	查看Resource Manager错误码表，或进一步联系技术支持。
39907	TABLE_NOT_FOUND schema=xxx table=xxx	相关操作的目标表不存在，请检查。
39908	Target table is not realtime table.	相关实时表操作的目标表不是实时表，请检查。
39910	Cannot find worker db information.	无法找到系统后台WorkerDB，请联系技术支持。
39911	Show tables command is only allowed to show tables under current database.	该命令只允许显示当前DB下的表。

12.9. 常见术语（2.0版）

本文介绍AnalyticDB for MySQL的基本概念。

基本概念

以下列出了数据库、ECU、表组、表和分区等概念。

- **数据库（DataBase）**

  在AnalyticDB for MySQL中，数据库是最高层的对象，按数据库进行资源的分配和管理。每个数据库独享一个服务进程，实现用户间资源的隔离。

  AnalyticDB for MySQL中数据库又称之为实例，一个数据库就是一个实例，一个实例由若干个ECU节点组成。
- **ECU**

  弹性计算单元（Elastic compute units，简称ECU）是ADB 2.0用来衡量实例计算能力的元单位。

  一个数据库由若干个同一类型的ECU节点组成，例如数据库A，可能由4个C8组成，或者6个S2N组成，每个ECU节点拥有固定的磁盘和内存资源。
- **表组**

  表组是指一系列可发生关联的数据表的集合。ADB 2.0为了管理相关联的数据表，引入了表组的概念。表组类似于传统数据库中的Schema，支持以下表组。
  - 维度表组（系统自带）：自带维度概念的表（例如省份表、银行表等）可以放到维度表组下。
  - 普通表组：一般会把需要关联的普通表放在相同普通表组中，建议这个表组中的所有普通表的一级分区数一致，join性能会有很大提升。
- **表（Table）**

在表组之下是表的概念，有两种类型的表。

- 维度表：带有维度概念的表称为维度表，例如银行表。默认每个ECU节点放置一份全量的维度表数据，因此维度表可以和任何普通表进行关联。
  由于维度表会消耗更多的存储资源，所以维度表的数据量大小有限制，要求维度表单表数据量不超过100万行。
- 普通表：普通表就是分区表，为充分利用分布式系统的查询能力而设计的一种表。普通表默认是指一级分区表，如果有增量数据导入需求，可以创建二级分区表。

- **分区**

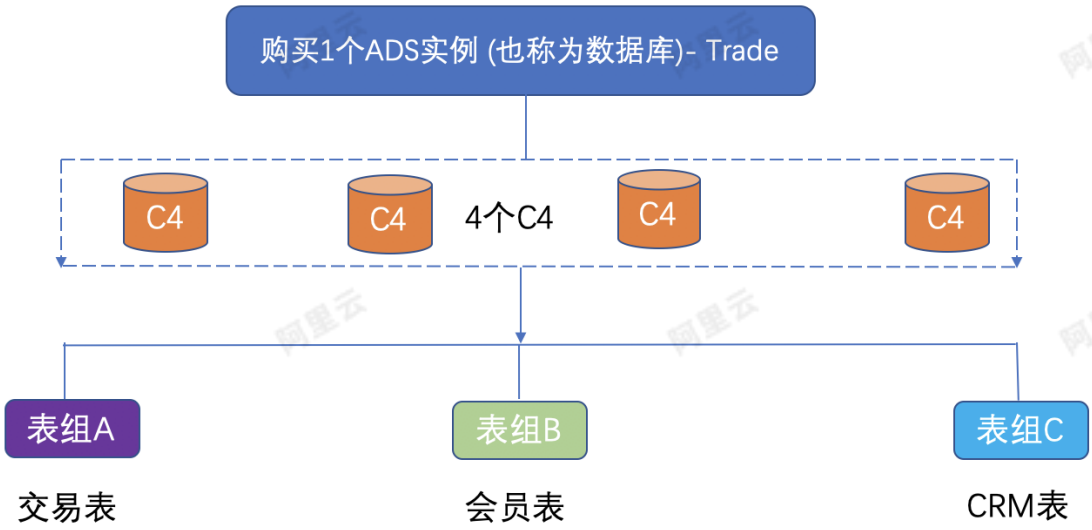
  普通表才有分区的概念。

  支持两级分区策略。
  - 一级分区采用HASH算法，单表数据量在60亿条以内，推荐您使用一级分区，通常一级分区已足够。
  - 二级分区采用LIST算法。
- **主键**

  表中必须包含主键字段，通过主键进行记录的唯一性判断。主键由业务ID、一级分区键组成，有些情况业务ID与一级分区相同。对于记录量特别大的表，从存储空间和INSERT性能考虑，一定要减少主键的字段数。
- **示例**

  以一个电商公司购买了一个分析型数据库MySQL版Trade为例，帮助您理解上述概念。

  i. 某用户在阿里云购买1个名为Trade的分析型数据库MySQL版（也称之为1个ADS实例），如图所示，Trade由4个C4节点构成。



  ii. C4是ECU规格类型，我们还提供C8、S2N、S8N三种不同规格的ECU。

  iii. 数据库Trade下面可以规划多个表组（类似Schema概念），不同表组用于存放不同的业务表。

  iv. Trade数据库创建完毕后，系统会默认创建一个维度表组，所有维度相关的表，可以放到维度表组下。普通表按照上述第3点的规则来管理。