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：20220704

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
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	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.产品定价 07
1.1. 计费方式 07
1.2. 变更配置计费规则 07
1.3. 到期或欠费的影响 07
1.4. 续费 07
1.5. 查看消费明细 07
2.快速入门 09
2.1. 快速入门综述 09
2.2. 连接分析型数据库MySQL版 10
2.3. 创建表组 10
2.4. 创建表 10
2.5. 同步数据 13
2.6. 导出数据 13
3.用户指南 14
3.1. 计费管理 14
3.1.1. 手动续费 14
3.1.2. 自动续费 15
3.1.3. 按量付费转包年包月 16
3.2. 监控与报警 16
3.2.1. 设置报警规则 16
3.2.2. 查看事件监控 17
3.3. SQL开发规范 18
4.权限与安全 20
4.1. 账号类型 20
5.访问分析型数据库MySQL版 21
6.数据迁移 22

6.1. 通过DTS将PolarDB数据同步至分析型数据库MySQL版 22
6.1.1. 实施步骤 22
6.2. 通过Logstash写入数据 24
6.3. 在程序中通过AnalyticDB MySQL版Client高效写入数据 25
6.4. 通过adbuploader将本地数据导入分析型数据库MySQL版 28
6.5. 通过日志服务同步ECS日志数据到AnalyticDB for MySQL 28
6.6. 使用DTS同步RDS MySQL数据 36
7.向量分析 38
7.1. 功能优势 38
7.2. 竞品分析 39
8.SQL手册 41
8.1. CREATE TABLE 41
8.2. CTE 42
8.3. INSERT 42
8.4. DUMP TO OSS 43
9.系统函数 44
9.1. 字符串函数 44
9.1.1. 字符串函数 44
9.2. 数值函数 50
10.最佳实践 59
10.1. 全文检索最佳实践 59
11.常见问题 62
11.1. 产品和业务限制 62
11.2. 基本数据库对象及概念 62
11.3. 查询报错问题 63
11.4. 应用开发向导 64
12.附录 65
12.1. 错误码表 65

12.2. 系统运算符 76
12.3. 元数据库数据字典 80

1.产品定价

1.1. 计费方式

分析型数据库MySQL版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

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

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

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

您可以通过阿里云管理控制台查看分析型数据库MySQL版的收费明细。

1. 登录阿里云管理控制台。
2. 在页面右上方，选择**费用** > **进入费用中心**。
3. 单击左侧导航栏**费用账单**。
4. 在**费用账单**页面，您可以通过以下形式查看消费详情。
   - 账单总览：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：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.快速入门

2.1. 快速入门综述

分析型数据库MySQL版（AnalyticDB MySQL版）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 MySQL版 2.0，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 MySQL版 2.0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 MySQL版 2.0服务。

1. **选地域和可用区**

   您可以选择在任何一个AnalyticDB MySQL版 2.0开服的地域购买AnalyticDB MySQL版 2.0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。

   中国内地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 MySQL版 2.0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 MySQL版 2.0仅支持相同类型的ECU之间自由变更配置，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 MySQL版 2.0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 MySQL版 2.0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 MySQL版 2.0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，请参见变更配置。

4. **设置数据库名字**

   数据库名是AnalyticDB MySQL版 2.0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 MySQL版 2.0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 MySQL版 2.0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 MySQL版 2.0全面兼容MySQL协议和SQL 2003，成功购买AnalyticDB MySQL版 2.0后，您就可以像使用MySQL数据库一样快速使用AnalyticDB MySQL版 2.0。

1. **创建表组**

   表组是一系列可发生关联的数据表的集合，AnalyticDB MySQL版 2.0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 MySQL版 2.0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 MySQL版 2.0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 MySQL版 2.0时默认只能创建经典网络，您可以在使用AnalyticDB MySQL版 2.0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 MySQL版 2.0**

   您可以通过分析型数据管理控制台连接AnalyticDB MySQL版 2.0，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 MySQL版 2.0，详情请参见连接分析型数据库MySQL版。

5. **将数据导入AnalyticDB MySQL版 2.0**

   AnalyticDB MySQL版 2.0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 MySQL版 2.0，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 MySQL版 2.0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，请提交工单联系技术支持。
   - **删除数据**：AnalyticDB MySQL版 2.0支持通过删除（DELETE）普通表中的数据。
   - **更新数据**：AnalyticDB MySQL版 2.0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 MySQL版 2.0服务，详情请参见关闭分析型数据库MySQL版服务。

2.2. 连接分析型数据库MySQL版

在应用开发中，您可以通过以下方式连接分析型数据库MySQL版。

- 通过代码连接分析型数据库MySQL版。
- 通过客户端连接分析型数据库MySQL版。

应用开发中通过代码连接分析型数据库MySQL版

在应用开发中可以通过Java、PHP、Python、C#或者Druid连接分析型数据库MySQL版。

通过客户端连接分析型数据库MySQL版

支持使用客户端MySQLWorkbench、DBVisualizer或者Navicat for MySQL连接分析型数据库MySQL版。

连接说明

连接分析型数据库MySQL版时，需要获取以下连接信息。

- **连接地址和端口**

  登录分析型数据库MySQL版管理控制台，在**连接信息**处查看连接地址和端口。**通过外网连接分析型数据库MySQL版时，使用经典网络地址连接串。通过内网连接分析型数据库MySQL版时，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 MySQL版）为管理相关联的数据表，引入表组的概念。

说明

创建AnalyticDB MySQL版数据库时，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

2.4. 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（__）。

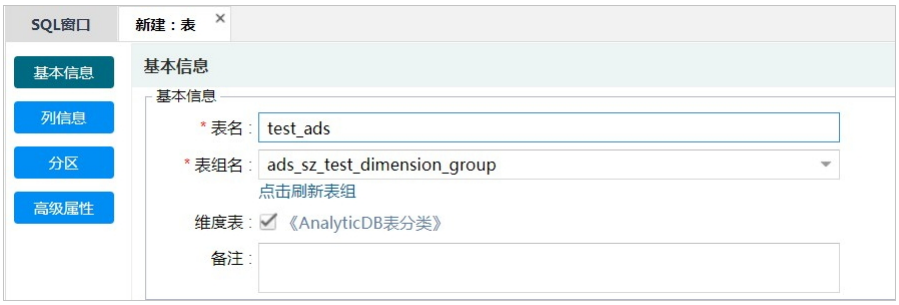
通过DMS页面创建维度表

1. 登录分析型数据管理控制台。

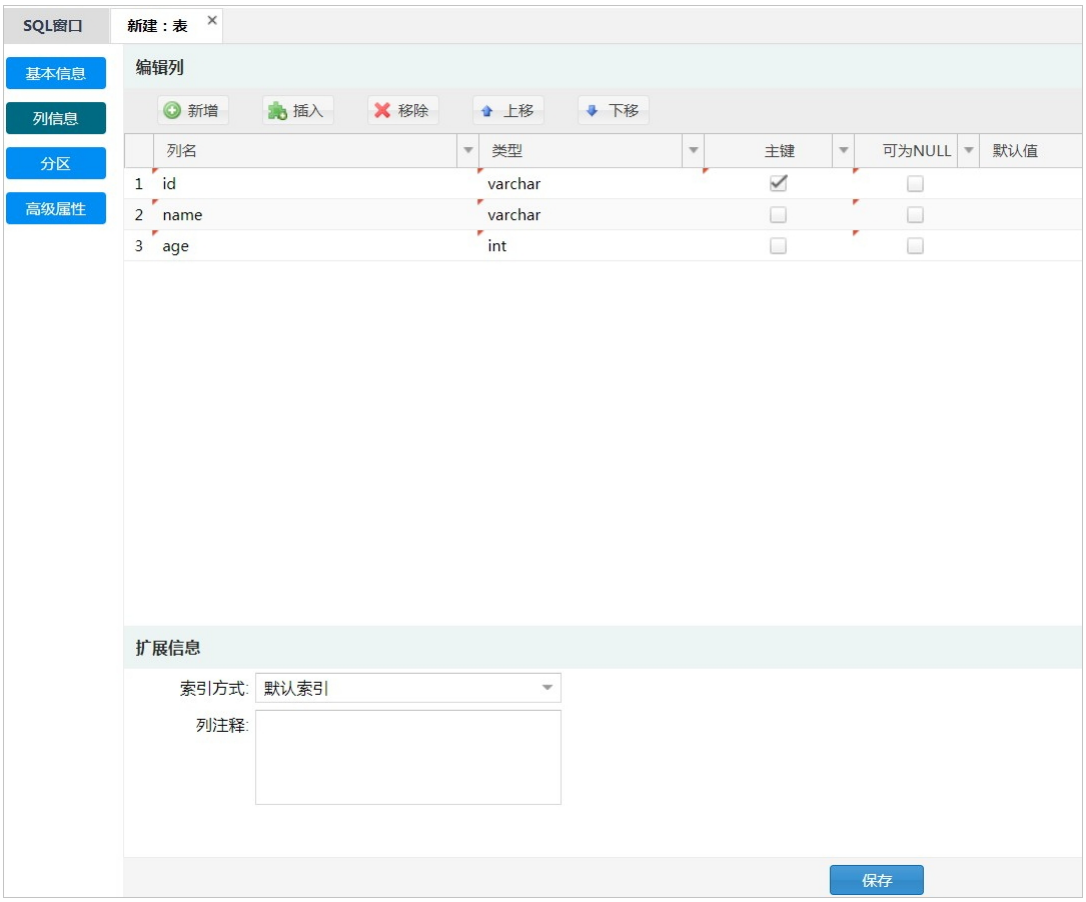
2. 在控制台左上方选择数据库所属地域。
3. 单击目标数据库右侧的**登录数据库**，在**DMS for AnalyticDB**页面的顶部菜单栏单击**创建>表**。
4. 在**新建：表**页面，按照提示进行参数配置。

   在**新建：表**页面，按照提示进行参数配置。

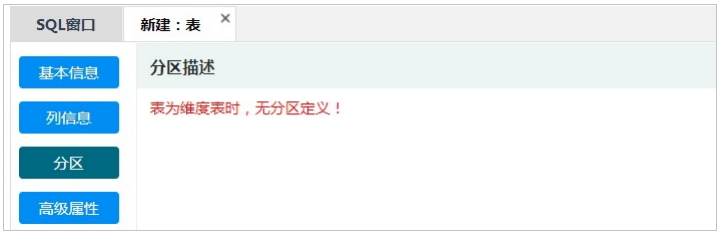
   基本信息配置：



列配置：



分区配置：



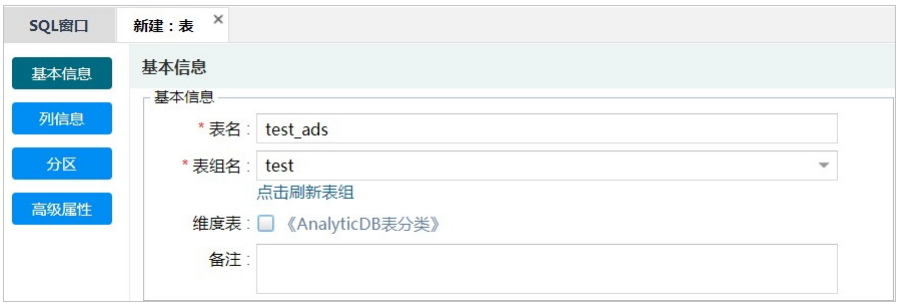
高级属性：

5. 配置好上述参数后，单击**保存**。

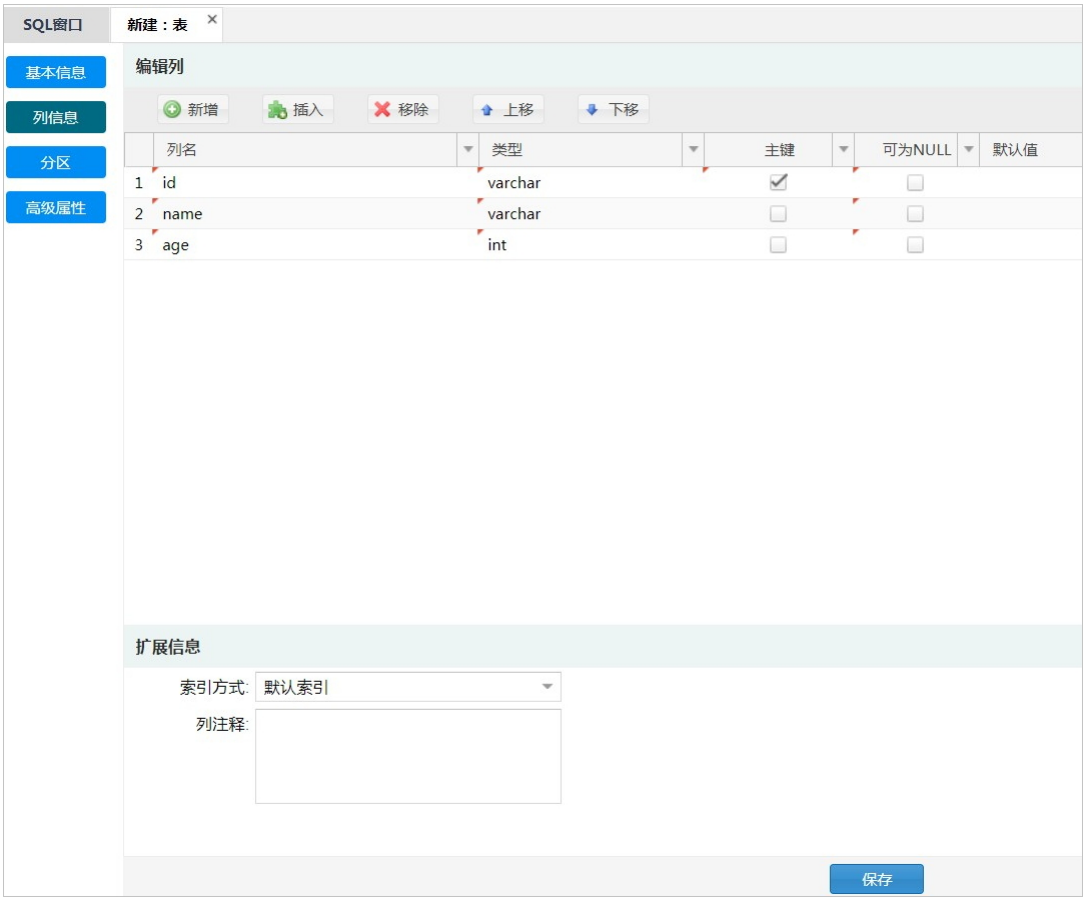
通过DMS页面创建普通表

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。

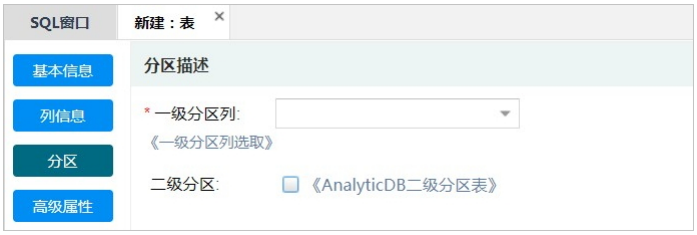
   基本信息配置：



   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.用户指南

3.1. 计费管理

3.1.1. 手动续费

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

包月套餐数据库开通自动续费后，系统将自动以选择的续费时长进行续费。例如，选择3个月的续费时长，则每次自动续费时会缴纳3个月的费用。

> **说明** 按量付费数据库没有到期时间，不涉及续费操作。

注意事项

- 若您在自动扣款日期前进行了手动续费，则系统将在下一次到期前进行自动续费。
- 自动续费功能于次日生效。若您的数据库于次日到期，为避免业务中断，请进行手动续费。

创建数据库时开通自动续费

在购买包月套餐数据库时，勾选**自动续费**。

创建数据库后开通自动续费

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**界面，选择**云原生数据仓库AnalyticDB MySQL版（包年包月）**。
4. 单击**手动续费**或**到期不续费**页签，找到目标数据库，单击右侧的**开通自动续费**。
5. 在**开通自动续费**页面，设置**自动续费时长**，然后单击**开通自动续费**。

修改自动续费

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

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. 监控与报警

3.2.1. 设置报警规则

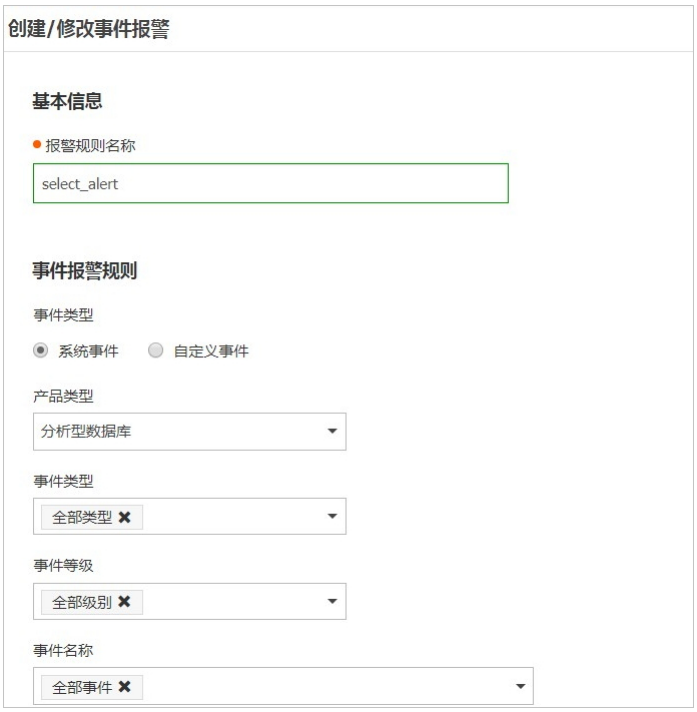
本文介绍如何通过云监控控制台为AnalyticDB for MySQL设置报警规则。

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。

3.2.2. 查看事件监控

本文介绍如何通过云监控控制台查看AnalyticDB MySQL版 2.0事件监控。通过事件监控，您可以明确知晓AnalyticDB MySQL版 2.0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 MySQL版 2.0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 MySQL版 2.0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 MySQL版 2.0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。

3.3. SQL开发规范

分析型数据库MySQL版是一个分布式、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。

在分析型数据库MySQL版中，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

分析型数据库MySQL版是列存数据库，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

分析型数据库MySQL版内部采用列存方式，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
```

4.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。

5.访问分析型数据库MySQL版

6.数据迁移

6.1. 通过DTS将PolarDB数据同步至分析型数据库MySQL版

6.1.1. 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。

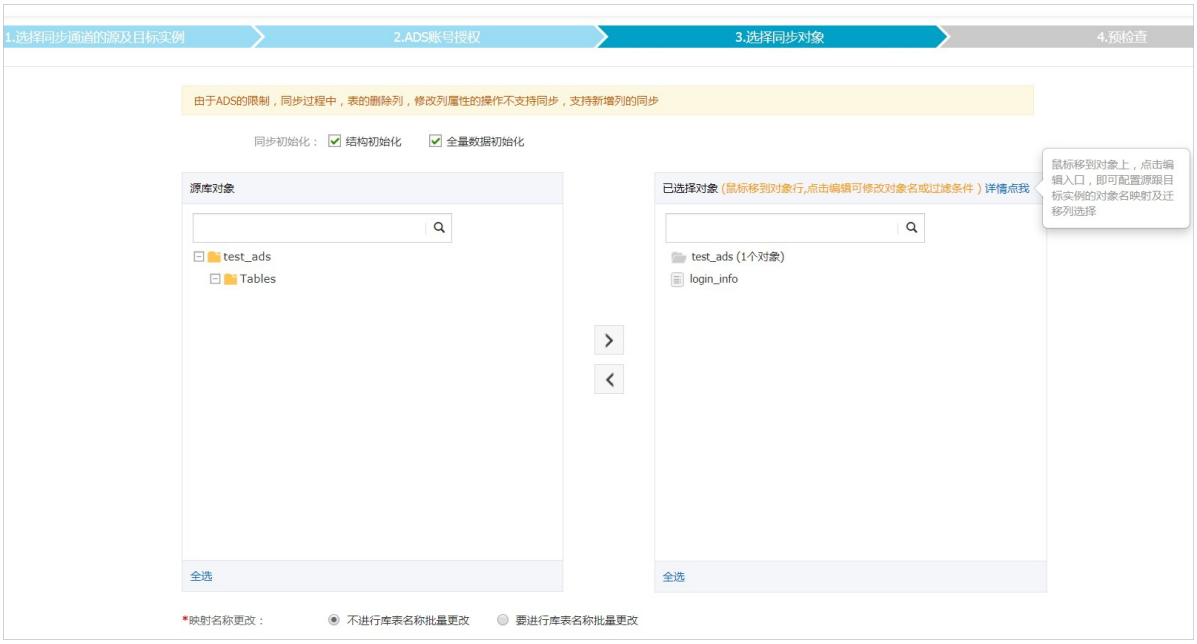
3. 在**确认订单**页面，勾选**《数据传输服务（按量付费）服务协议》**，根据提示完成支付流程。

步骤二：配置同步链路

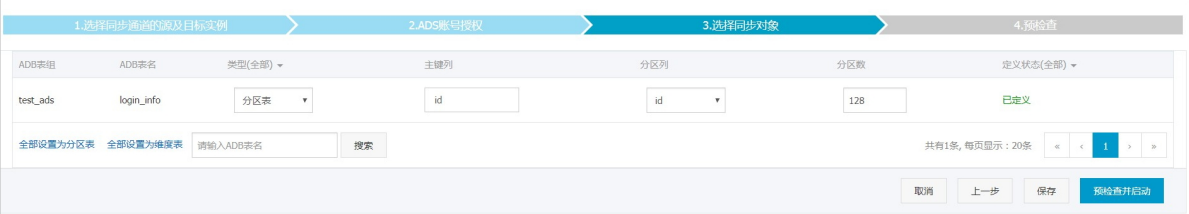
1. 登录DTS控制台。
2. 在**数据传输**页面，单击左侧导航栏中的**数据同步**。
3. 选择**地域**。
4. 在**同步作业**列表中，单击目标实例右侧的**配置同步链路**，在**选择同步通道的源及目标实例**页面进行参数配置，详细的参数配置如下表所示。

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。 C:\Users ping pc- mysql.polardb.rds.aliyuncs.com 正在 Ping pc- .mysql.polardb.rds.aliyuncs.com [172. 36] 用 有 32 字节的数据: 请求超时。
	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。
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. 通过Logstash写入数据

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 http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.36/mysql-connector-java-5.1.36.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。

6.3. 在程序中通过AnalyticDB MySQL版Client高效写入数据

本文介绍了在程序中通过AnalyticDB MySQL版Client高效写入数据的方法。

背景信息

AnalyticDB MySQL版Client旨在为用户提供一个高效、简单地插入数据到分析型数据库MySQL版的方法。您只需要通过接口将数据提交给AnalyticDB MySQL版Client，便可以直接插入数据到分析型数据库MySQL版中。使用AnalyticDB MySQL版 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 MySQL版的主机或域名。
setPort(int port)	设置需要连接的AnalyticDB MySQL版端口。
setDatabase(String database)	需要连接的AnalyticDB MySQL版库名。
setUser(String username)	设置连接AnalyticDB MySQL版使用的用户名，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 MySQL版使用的密码，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 MySQL版时的并发线程数，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 MySQL版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。

接口名	描述
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 MySQL版中。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。 **注意** 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();
            // 分析型数据库MySQL版的主机名或者URL
            databaseConfig.setHost("100.100.100.100");
            // 分析型数据库MySQL版的连接端口号
            databaseConfig.setPort(8888);
            // 分析型数据库MySQL版的所属账号的AccessKey ID
            databaseConfig.setUser("your db username");
            // 分析型数据库MySQL版的所属账号的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，写入分析型数据库MySQL版出现异常时重试的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分析型数据库MySQL版
            //数据成功提交到分析型数据库MySQL版之后，缓存中的数据会被清除
            try {
                adbClient.commit();
            } catch (Exception e) {
            } finally {
                adbClient.stop();
            }
        }
    }
```

注意事项

- AnalyticDB MySQL版Client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 MySQL版Client 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 MySQL版。
- 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。

6.4. 通过adbuploader将本地数据导入分析型数据库MySQL版

本文介绍如何将本地的数据文件导入分析型数据库MySQL版中。

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:1
0033 -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
```

6.5. 通过日志服务同步ECS日志数据到AnalyticDB for MySQL

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 for MySQL。

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（Log Service，简称LOG/原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 LOG产品详情页，单击**立即购买**，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

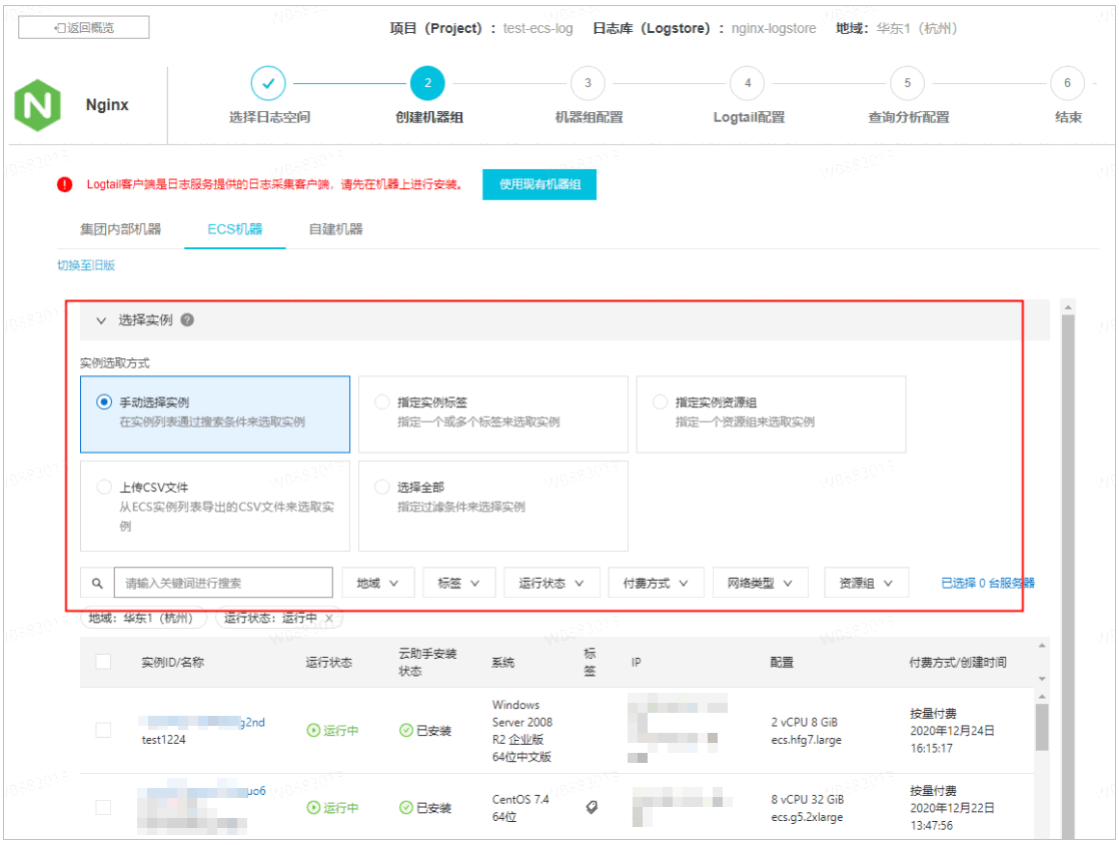
步骤一：配置日志Logtail

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。

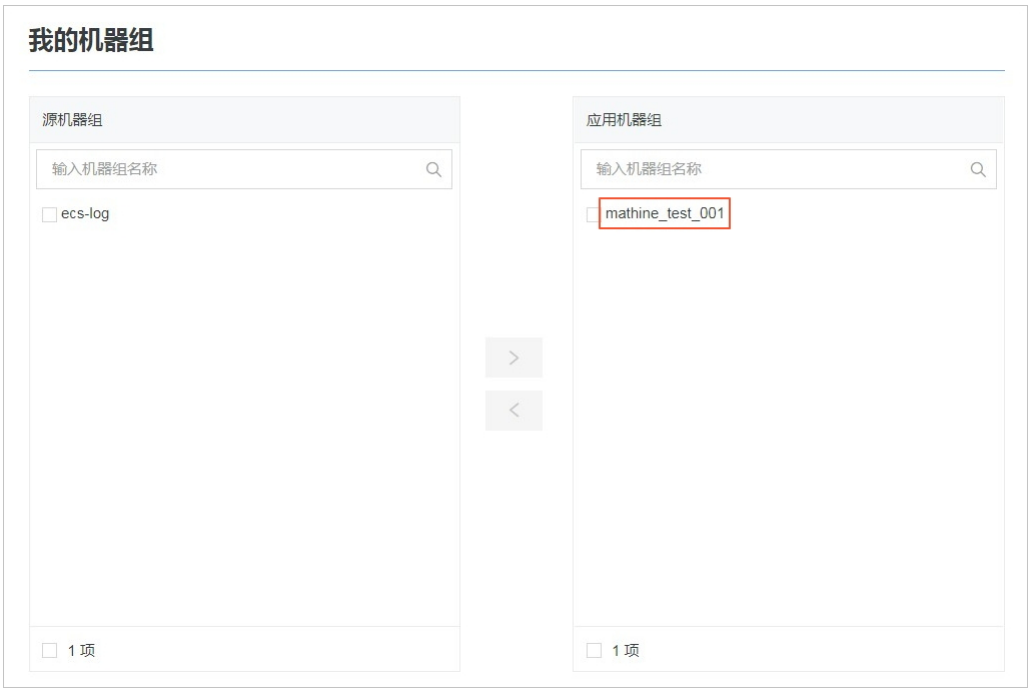
5. 单击**下一步**，在**创建机器组**步骤中，选择**ECS机器**，系统自动拉取与日志服务地域相同的ECS实例，勾选您需要的ECS实例ID，单击**立即执行**，完成后单击**确认安装完毕**。



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**。



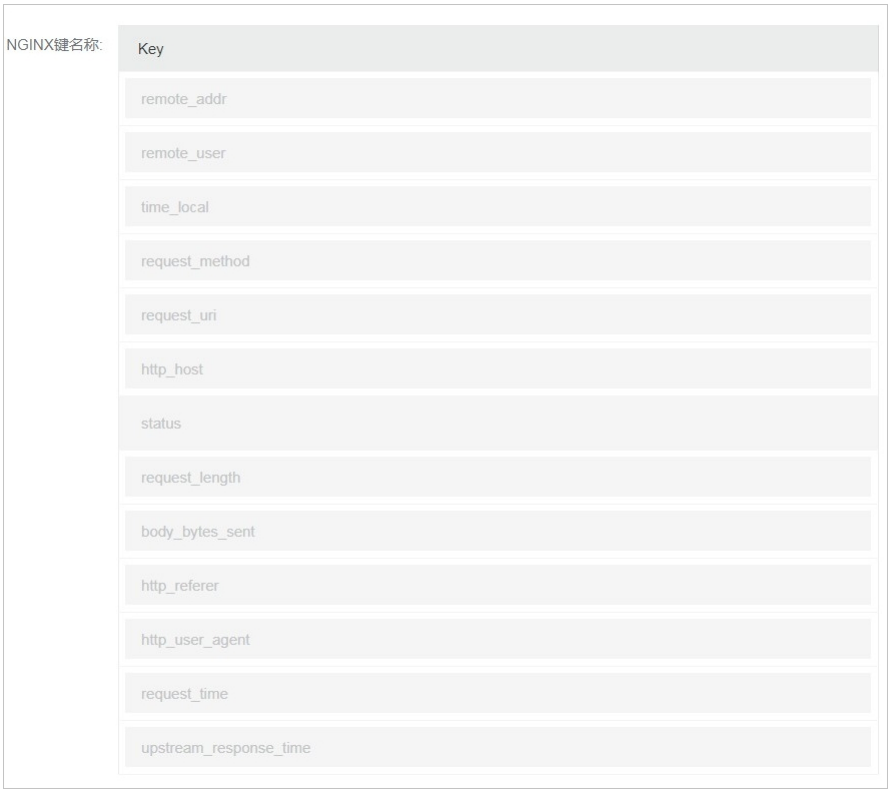
8. 在**Logtail配置**中，按照页面提示进行参数配置。

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。



10. 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。

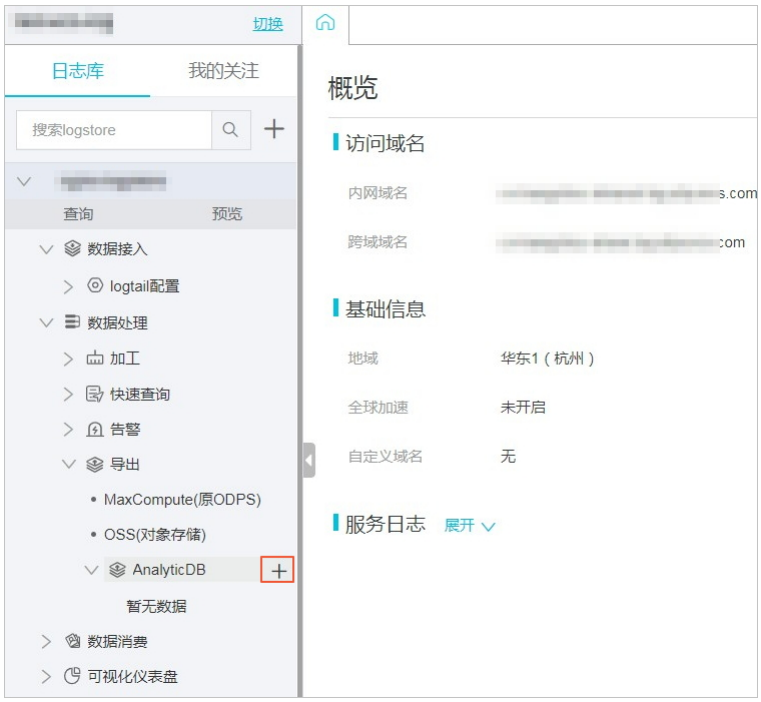
步骤二：将日志投递到AnalyticDB for MySQL

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。

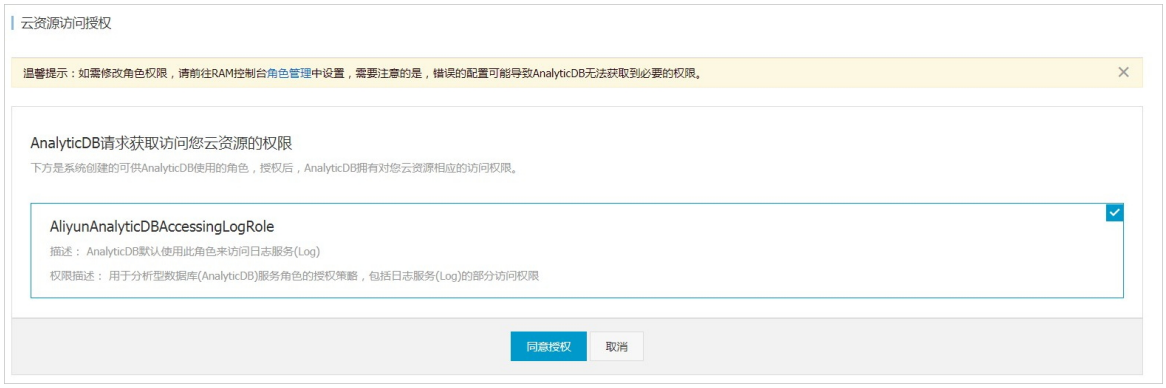
1. 登录日志服务控制台，单击Project名称。
2. 在**日志库**列表中，单击目标Logstore名称前的> > **数据处理** > **导出** > **AnalyticDB**，选择需要的AnalyticDB for MySQL集群。

   您需要为日志服务授权，允许日志数据写入AnalyticDB for MySQL。

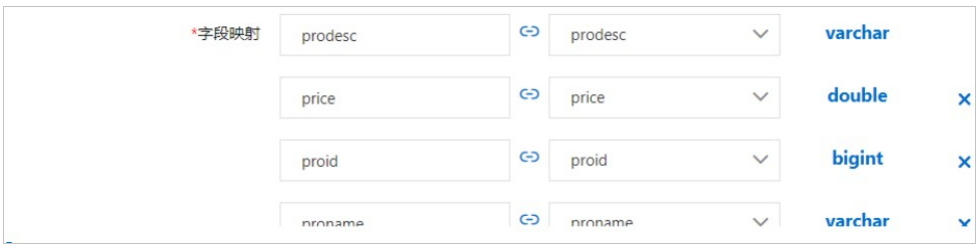
   i. 单击AnalyticDB后的**+**，系统自动提示您进行授权操作，单击**权限**。



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。

参数	说明
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。

6.6. 使用DTS同步RDS MySQL数据

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 MySQL版 2.0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 MySQL版 2.0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上述步骤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 MySQL版 2.0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 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 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.向量分析

7.1. 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 >100and price <=200andInTime>'2019-03-0100: 00: 00'andInTime<='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 >100and price <=200andInTime>'2019-03-0100: 00: 00'andInTime<='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 whereInTime>='2019
-02-01 00: 00: 00'andInTime<'2019-02-29 00: 00: 00') B whereVector_Distance(A.feature, B.feature,'DotProduct')>0.9
```

SQL详解：

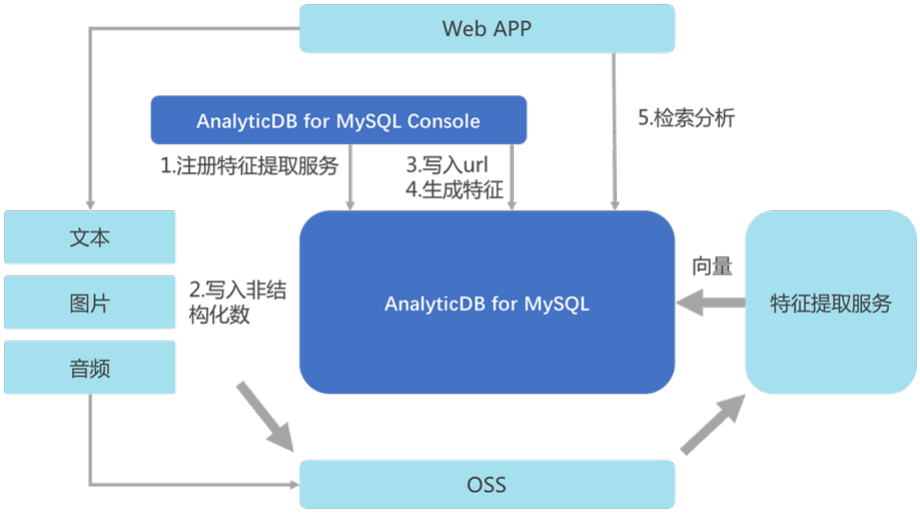
上述SQL把 `'2019-03-31 00: 00: 00'` 之后写入的数据与二月份的数据做笛卡尔积，把向量点积大于0.9的商品的对应ID提取出来。

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。

7.2. 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：

1. 通过封装的Python接口把文本转为向量。
2. 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。
3. 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。
4. 导入数据时，同时写入向量表和向量索引表。
5. 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。

8.SQL手册

8.1. CREATE TABLE

`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` ，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。

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。

> **注意** 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
- 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

8.2. CTE

CTE是一个命名的临时结果集，仅在单个SQL语句（例如SELECT，INSERT或DELETE）的执行范围内存在。CTE不作为对象存储，仅在查询执行期间持续，提供了更好的可读性和性能。

可以使用WITH子句来创建CTE，WITH子句用于定义一个子查询关系，以供SELECT查询引用，它可以扁平化嵌套查询或者简化子查询。使用WITH子句定义一个子查询后，SELECT只需要执行一遍这个子查询即可，提高查询性能。

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。

8.3. INSERT

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。

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。

8.4. DUMP TO OSS

导出数据到对象存储（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CUSTOMER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。

9.系统函数

9.1. 字符串函数

9.1.1. 字符串函数

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：

```
SELECT Concat('aliyun', ', ', 'analyticdb')
  +--------------------------------------+
  | _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：

```
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('helloworld', 'o'),
       position('helloworld', 'or'),
       position('helloworld', 'x');
```

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. 数值函数

- 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.最佳实践

10.1. 全文检索最佳实践

分析型数据库MySQL版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。如果需要将所有命中结果按照近似度从高到低排序，则使用如下SQL语法。

如果需要将结果按照近似度从高到低排序，则加上 `ORDER BY DESC` 。

```
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，如下列子中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>`

> **注意**
> 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');
```

11.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	工单
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	工单申请更多
单个分析型数据库MySQL版可购买的ECU数量最大值	C8：16个	工单申请更多
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)`	工单
单个分析型数据库MySQL版最多表数	256	提交工单
单个表组总表数	256	提交工单
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	提交工单
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 MySQL版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，需要增加可以提工单解决。

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 MySQL版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 MySQL版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. 查询报错问题

本文汇总了AnalyticDB MySQL版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/+limit max=2000000/select * from ar_express3 limit 1000000,20`。
- 提工单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=20171130140151010154169109099999087231 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: [], s
qlState=null, errorCode=65541, errorName=NO_NODES_AVAILABLE, errorType=INTERNAL_ERROR, errorLocation=null, failureInfo=mpp.client.FailureInfo@13887b2
e}
```

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，请提工单联系分析型数据库MySQL版技术支持。

11.4. 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：
   5.1.1，5.1.2，5.1.3，5.1.4，5.1.5，5.1.6，5.1.7，5.1.8，5.1.11，5.1.12，5.1.13，5.1.14，5.1.15，5.1.16，5.1.17，5.1.18，5.1.19，5.1.20，5.1.21，5.1.22，5.1.23，5.
   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.附录

12.1. 错误码表

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。

错误码	错误信息	解决办法
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。

错误码	错误信息	解决办法
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。

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。

错误码	错误信息	解决办法
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。

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。

错误码	错误信息	解决办法
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。

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。

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。

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。

错误码	错误信息	解决办法
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。

错误码	错误信息	解决办法
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。

错误码	错误信息	解决办法
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。

错误码	错误信息	解决办法
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。

错误码	错误信息	解决办法
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	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	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.。

错误码	错误信息	解决办法
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。

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。

错误码	错误信息	解决办法
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	NA	请联系技术支持。

12.2. 系统运算符

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

运算符	描述	例子
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。

表达式	含义
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;
```

12.3. 元数据库数据字典

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_TA
BLE_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

FIELD	TYPE	ALLOW_NULL	PK	DEFAULT_VALUE	COMMENT
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

FIELD	TYPE	ALLOW_NULL	PK	DEFAULT_VALUE	COMMENT
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。