TDSQL MYSQL 版（TDSQL）

产品文档

文档目录

错误码

产品简介
产品概述

最近更新时间: 2025-02-18 16:02:00

简介

TDSQL MYSQL 版是部署在云上的一种支持自动水平拆分、Shared Nothing 架构的分布式数据库。TDSQL MYSQL 版即业务获取的是完整的逻辑库表，而后端会将库表均匀的拆分到多个物理分片节点。TDSQL MYSQL 版默认部署主备架构，提供容灾、备份、恢复、监控、迁移等全套解决方案，适用于 TB 或 PB 级的海量数据库场景。

> **注意：**
>
> 由于TDSQL MYSQL 版正在更新其英文简称，您可能观察到：
>
> - 云数据库MySQL/MariaDB的实例ID开头为：tdsql-
> - TDSQL MYSQL 版的实例ID开头为：tdsqlshard-或dcdb-

解决问题

单机数据库瓶颈

面对互联网类业务百万级以上的用户量，单机数据库由于硬件和软件的限制，数据库在数据存储容量、访问容量、容灾等方面都会随着业务的增长而到达瓶颈。

TDSQL MYSQL 版目前单分片最大可支持6TB存储，如果性能或容量不足以支撑业务发展时，在控制台自动升级扩容。升级过程中，您无需关心分布式系统内的数据迁移，均衡和路由切换。升级完成后访问 IP 不变，仅在自动切换时存在秒级闪断，您仅需确保有重连机制即可。

应用层分片开发工作量大

应用层分片将业务逻辑和数据库逻辑高度耦合，给当前业务快速迭代带来极大的开发工作量。 基于 TDSQL MYSQL 版透明自动拆分的方案，开发者只需要在第一次接入时修改代码，后续迭代无需过多关注数据库逻辑，可以极大减少开发工作量。

开源方案或 NoSQL 难题

选择开源或 NoSQL 产品也能够解决数据库瓶颈，这些产品免费或者费用相对较低，但可能有如下问题：

- 产品 bug 修复取决于社区进度。
- 您的团队是否有能持续维护该产品的人，且不会因为人事变动而影响项目。
- 关联系统是否做好准备。
- 您的业务重心是什么，投入资源来保障开源产品的资源管控和生命周期管理、分布式逻辑、高可用部署和切换、容灾备份、自助运维、疑难排查等是否是您的业务指标。

TDSQL MYSQL 版支持 Web 控制台，提供完善的数据备份、容灾、一键升级等功能，完善的监控和报警体系，大部分故障都通过自动化程序处理恢复。

产品优势

最近更新时间: 2025-02-18 16:02:00

超高性能

- 整个实例性能随着分片数量增加线性扩展。
- 不存在中间件 + 数据库方案中的性能瓶颈，即 Proxy 也可以做线性扩展。
- 强同步性能与异步同步相当，能让您在数据不丢失的情况下，也拥有较高的性能。

专业可靠

- 经过各类核心业务10余年大规模产品的验证，包括社交、电商、支付、音视频等。
- 提供完善的数据备份、容灾、一键升级等功能。
- 完善的监控和报警体系，大部分故障都通过自动化程序处理恢复。
- 支持分布式数据库领域领先功能，如分布式多表 JOIN、小表广播、分布式事务、SQL 透传等。

简单易用

- 除少量语法与原生 MySQL、MariaDB 不同外，使用起来如使用单机数据库，分片过程对业务透明且无需干预。
- 兼容 MySQL 协议（支持 MySQL、MariaDB 等内核）。
- 支持 Web 控制台，读写分离能力、专有运维管理指令等。

应用场景

最近更新时间: 2025-02-18 16:02:00

当前版本 TDSQL MYSQL 版目前仅适用于 OLTP 场景的业务，例如，交易系统、前台系统；不适用于 ERP、BI 等存在大量 OLAP 业务的系统。

大型应用（超高并发实时交易场景）

电商、金融、O2O、社交应用、零售、SaaS 服务提供商，普遍存在用户基数大（百万级或以上）、营销活动频繁、核心交易系统数据库响应日益变慢的问题，制约业务发展。

TDSQL MYSQL 版提供线性水平扩展能力，能够实时提升数据库处理能力，提高访问效率，峰值 QPS 达1500万+，轻松应对高并发的实时交易场景。微信支付、财付通、腾讯云金融专区充值等都是使用的 TDSQL MYSQL 版架构的数据库。

物联网数据（PB 级数据存储访问场景）

在工业监控和远程控制、智慧城市的延展、智能家居、车联网等物联网场景下，传感监控设备多、采样率高、数据规模大。通常存储一年的数据就可以达到 PB 级甚至 EB，而传统基于 x86 服务器架构和开源数据库的方案根本无法存储和使用如此大的数据量。

TDSQL MYSQL 版提供的容量水平扩展能力，可以有效的帮助用户以低成本（相对于共享存储方案）存储海量数据。

文件索引（万亿行数据毫秒级存取）

一般来说，作为云服务平台，存在大量的图片、文档、视频数据，数据量都在亿级 - 万亿级，服务平台通常需要将这些文件的索引存入数据库，并在索引层面提供实时的新增、修改、读取、删除操作。

由于服务平台承载着其他客户的访问，服务质量和性能要求极高。传统数据库无法支撑如此规模的访问和使用，TDSQL MYSQL 版超高性能和扩展能力并配合强同步能力，有效的保证平台服务质量和数据一致性。

高性价比商业数据库解决方案

政务机构、大型企业、银行等行业为了支持大规模数据存储和高并发数据库访问，对小型机和高端存储依赖极强。而互联网企业通过低成本 x86 服务器和开源软件即可做到商业数据库相同甚至更高的能力。

TDSQL MYSQL 版适用于诸如国家级或省级业务系统汇聚、大型企业电商和渠道平台、银行的互联网业务和交易系统等场景。

基本原理
水平分表

最近更新时间: 2025-02-18 16:02:00

概述

水平拆分方案，实际上是分布式数据库的基础原理，他的每个节点都参与计算和数据存储，且每个节点都仅计算和存储一部分数据。因此，无论业务的规模如何增长，我们仅需要在分布式集群中不断的添加设备，用新设备去应对增长的计算和存储需要即可。

水平切分

水平切分（分表）：是按照某种规则，将一个表的数据分散到多个物理独立的数据库服务器中，形成"独立"的数据库"分片"。多个分片共同组成一个逻辑完整的数据库实例。

- 常规的单机数据库中，一张完整的表仅在一个物理存储设备上读写。

- 分布式数据库中，根据在建表时设定的分表键，系统将根据不同分表键自动分布到不同的物理分片中，但逻辑上仍然是一张完整的表。

- 在 TDSQL 中，数据的切分通常就需要找到一个分表键（shardkey）以确定拆分维度，再采用某个字段求模（HASH）的方案进行分表，而计算 HASH 的某个字段就是 shardkey。 HASH 算法能够基本保证数据相对均匀地分散在不同的物理设备中。

写入数据（ SQL 语句含有 shardkey ）

1. 业务写入一行数据。
2. 网关通过对 shardkey 进行 hash。
3. 不同的 hash 值范围对应不同的分片（调度系统预先分片的算法决定）。
4. 数据根据分片算法，将数据存入实际对应的分片中。

数据聚合

数据聚合：如果一个查询 SQL 语句的数据涉及到多个分表，此时 SQL 会被路由到多个分表执行，TDSQL 会将各个分表返回的数据按照原始 SQL 语义进行合并，并将最终结果返回给用户。

> **说明：**
>
> 执行 SELECT 语句时，建议您在 where 条件带上 shardKey 字段，否则会导致数据需要全表扫描然后网关才对执行结果进行聚合。全表扫描响应较慢，对性能影响很大。

读取数据（有明确 shardkey 值）

1. 业务发送 select 请求中含有 shardkey 时，网关通过对 shardkey 进行 hash。
2. 不同的 hash 值范围对应不同的分片。
3. 数据根据分片算法，将数据从对应的分片中取出。

读取数据（无明确 shardkey 值）

1. 业务发送 select 请求没有 shardkey 时，将请求发往所有分片。
2. 各个分片查询自身内容，发回 Proxy 。
3. Proxy 根据 SQL 规则，对数据进行聚合，再答复给网关。

读写分离

最近更新时间: 2025-02-18 16:02:00

功能简介

当处理大数据量读请求的压力大、要求高时，可以通过读写分离功能将读的压力分布到各个从节点上。 TDSQL 默认支持读写分离功能，架构中的每个从机都能支持只读能力，如果配置有多个从机，将由网关集群（Proxy）自动分配到低负载从机上，以支撑大型应用程序的读取流量。

基本原理

读写分离基本的原理是让主节点（Master）处理事务性增、改、删操作（INSERT、UPDATE、DELETE），让从节点（Slave）处理查询操作（SELECT）。

只读账号

只读帐号是一类仅有读权限的帐号，默认从数据库集群中的从机（或只读实例）中读取数据。 通过只读帐号，对读请求自动发送到备机，并返回结果。

弹性拓展

最近更新时间: 2025-02-18 16:02:00

概述

TDSQL MYSQL 版支持在线实时扩容，扩容方式分为新增分片和现有分片扩容两种方式，整个扩容过程对业务完全透明，无需业务停机。扩容时仅部分分片存在秒级的只读或中断，整个集群不会受影响。

扩容过程

TDSQL MYSQL 版主要是采用自研的自动再均衡技术保证自动化的扩容和稳定。

新增分片扩容

1. 控制台单击扩容后，系统根据负载和容量计算出 A 节点（实际上可能影响多个节点）存在瓶颈。
2. 根据新加 G 节点配置，将 A 节点部分数据搬迁（从备机）到 G 节点。
3. 数据完全同步后，A、G 节点校验数据库，存在一至几十秒的只读，但整个服务不会停止。
4. 调度通知 proxy 切换路由。

现有分片扩容

基于现有分片的扩容相当于更换了一块更大容量的物理分片。 基于现有分片的扩容没有增加分片，不会改变划分分片的逻辑规则和分片数量。

1. 按需要升级的配置分配一个新的物理分片（以下简称新分片）。
2. 将需要升级的物理分片（以下简称老分片）的数据、配置等同步数据到新分片中。
3. 数据同步完成后，会在网关做路由切换，之后将使用新分片。

强同步

最近更新时间: 2025-02-18 16:02:00

背景

传统数据复制方式有如下三种：

- 异步复制：应用发起更新请求，主节点（Master）完成相应操作后立即响应应用，Master 向从节点（Slave）异步复制数据。
- 强同步复制：应用发起更新请求，Master 完成操作后向 Slave 复制数据，Slave 接收到数据后向 Master 返回成功信息，Master 接到 Slave 的反馈后再应答给应用。Master 向 Slave 复制数据是同步进行的。
- 半同步复制：正常情况下数据复制方式采用强同步复制方式，当 Master 向 Slave 复制数据出现异常的时候（Slave 不可用或者双节点间的网络异常）退化成异步复制。当异常恢复后，异步复制会恢复成强同步复制。

存在问题

当 Master 或 Slave 不可用时，以上三种传统数据复制方式均有几率引起数据不一致。

数据库作为系统数据存储和服务的核心能力，其可用性要求非常高。在生产系统中，通常都需要用高可用方案来保证系统不间断运行，而数据同步技术是数据库高可用方案的基础。

解决方案

MAR 强同步复制方案是腾讯云金融专区 自主研发的基于 MySQL 协议的异步多线程强同步复制方案，只有当备机数据完全同步（日志）后，才由主机给予应用事务应答，保障数据正确安全。

原理示意图如下：

MAR 强同步方案在性能上优于其他主流同步方案，具体数据详情可参见“强同步性能对比数据”。主要特点如下：

- 一致性的同步复制，保证节点间数据强一致性。
- 对业务层面完全透明，业务层面无需做读写分离或同步强化工作。
- 将串行同步线程异步化，引入线程池能力，大幅度提高性能。
- 支持集群架构。
- 支持自动成员控制，故障节点自动从集群中移除。
- 支持自动节点加入，无需人工干预。
- 每个节点都包含完整的数据副本，可以随时切换。
- 无需共享存储设备。

实例架构

最近更新时间: 2025-02-18 16:02:00

实例架构

基于可用性部署要求，每个物理分片均为主从架构，即一个实例由多个物理分片组成，一个物理分片由一主节点多个从节点组成。其默认从节点数量大于等于1，小于等于5。即理论可分配一主一从（2节点）至一主五从（6节点）。

> 请注意：物理分片与某些分布式数据库的（逻辑）分片并非同意定义。(逻辑）分片通常是指的一个大表水平拆分的最小粒度；而物理分片是指一组物理资源。

实例跨可用区部署

实例默认部署在相同可用区。部分场景下，主、从节点可以分别部署在两个不同可用区。

其默认分配逻辑如下：

	主可用区节点数	从可用区节点数
一主一从	1	1
一主二从	2	1
一主三从	2	2
一主四从	3	2
一主五从	3	3

更多的节点意味着更高的可用性和更高的读性能。

- 如主从需支持强同步(不可退化)，需大于等于3个节点。
- 如果您需要区别于上述可用区部署要求，可以基于资源独享方式进行申请，具体可以咨询工作人员。

地域选择

最近更新时间: 2025-02-18 16:02:00

地域

简介

地域（Region）是指物理的数据中心的地理区域。不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。 您可以查看下表或者通过 API 接口 查询地域列表 查看完整的地域列表。

相关特性

- 不同地域之间的网络完全隔离，不同地域之间的云产品默认不能通过内网通信。
- 不同地域之间的云产品，可以通过 公网 IP 访问 Internet 的方式进行通信。处于不同私有网络的云产品，可以通过对等连接进行通信，此通信方式更较为高速、稳定。
- 负载均衡 当前默认支持同地域流量转发，绑定本地域的云服务器。如果开通 跨地域绑定 功能，则可支持负载均衡跨地域绑定云服务器。

可用区

简介

可用区（Zone）是指在同一地域内电力和网络互相独立的物理数据中心。其目标是能够保证可用区间故障相互隔离（大型灾害或者大型电力故障除外），不出现故障扩散，使得用户的业务持续在线服务。通过启动独立可用区内的实例，用户可以保护应用程序不受单一位置故障的影响。您可以通过 API 接口 查询可用区列表 查看完整的可用区列表。

相关特性

处于相同地域不同可用区，但在同一个私有网络下的云产品之间均通过内网互通，可以直接使用 内网 IP 访问。

> 内网互通是指同一账户下的资源互通，不同账户的资源内网完全隔离。

地域和可用区明细

地域	可用区	ZoneId
上海金融专区 `shjr` 50000001	上海金融专区一区 `shjraz1`	50010001
	上海金融专区二区 `shjraz2`	50010002
深圳金融专区 `szjr` 50000002	深圳金融专区一区 `szjraz1`	50020001

资源独享

最近更新时间: 2025-02-18 16:02:00

资源独享简介

独享集群数据库（Database Dedicated Cluster），简称**资源独享**，可以让您以**独享的形式享有一套物理集群（完整）**的资源，并在该方案下直接购买、创建数据库实例。以满足您对独特架构设计、资源独享、物理安全、行业监管等需求。其与公有云多租户的方案区别可以用下图表示：

独享集群支持范围

> **说明：**
>
> **购买独享集群后，您可以在其上灵活创建多种自定义规格的云数据库。**

现已支持分配：云数据库MySQL/MariaDB、TDSQL MYSQL 版两种类型的实例。同时，请注意如下情况：

- 您将独占对应物理服务器，其他（无权限）租户将不可使用您的物理服务器。
- 云数据库租户管理系统：您将与其他租户共享。该系统部署在云上更高等级的管理网络中，是您提供灵活的运维能力的基础，因此请放心使用。
- 备份和日志存储：您将与其他租户共享。当然，联系工作人员并付费的情况下，您也可以申请独占。
- 云网关/安全网关：部署在数据库前端的提供智能负载均衡和安全管理的虚拟网络设备，为您的数据库提供 VIP（唯一虚拟 IP）、VPC、主备切换、安全防护等能力，您将与其他用户共享。

如何购买独享集群

目前暂不提供自助购买能力，需人工在后台系统进行分配，如有需请咨询工作人员。

快速入门

最近更新时间: 2025-02-18 16:02:00

快速入门指引

操作项	操作说明
创建实例	介绍通过 TDSQL 控制台创建实例的操作。
初始化实例	介绍通过TDSQL控制台进行初始化实例操作。该功能在3.10.0版本已经支持实例创建后自动初始化，可跳过此操作。
连接实例	介绍在 TDSQL 控制台通过windows和Linux的不同方式连接实例的操作。
查看实例	介绍通过TDSQL控制台查看实例的基本信息和操作。
表介绍	介绍分布式实例支持的表类型。

购买指导
计费概述

最近更新时间: 2025-02-18 16:02:00

> **注意：**
>
> 部分版本未开启计费能力，仅支持计量能力。

计费方式

分布式数据库 TDSQL 提供如下计费模式：

计费模式	付费模式	适用场景
按量计费	后付费模式，即先按需申请资源使用，在结算时会按您的实际资源使用量收取费用。	适合业务量有瞬间大幅波动的业务场景，用完可立即释放实例，节省成本。

按量计费实例采用线性讲价方式。

节点规格

详细内容请联系工作人员。

产品定价

最近更新时间: 2025-02-18 16:02:00

> **注意：**
>
> 部分版本未开启计费能力，仅支持计量能力。

计费公式

总费用 = 节点价格 × 节点数量 × 分片数量 = (节点内存 × 内存价格 + 节点磁盘 × 磁盘价格) × 节点数量 × 分片数量

> **说明：**
>
> 节点数量为实例规格的主从个数总和。如一主一从为2节点，一主二从为3节点。

计费项

计费项	计费项	说明
实例费用	内存规格费用	在购买页选择的实例规格的费用，支持按量计费线性价格。

节点按量计费价

详细内容请咨询工作人员。

计费示例

公式中的小时费用仅为示例，详细内容请咨询工作人员。

【按量计费示例】：在北京地域下，购买1个按量计费分布式数据库 TDSQL（一主一从版），其节点内存为2GB、硬盘为500GB，分片数量为2，使用时长为400个小时。

则所需支付的费用计算如下：

实例费用：(2GB × XX元/GB/小时 + 500GB × XX元/GB/小时) × 2个节点 × 2个分片 × 96小时 = XX元

实例升级

最近更新时间: 2025-02-18 16:02:00

在线扩容

TDSQL支持在线实时扩容，扩容方式分为新增分片和对现有分片扩容两种方式：

- **新增分片**：新增加一个物理分片，支持针对指定分片进行拆分数据（原有分片配置不变），数据将自动均衡到新分片，无需手工操作。
- **扩容分片**：即对某个物理分片的配置进行升级，升级后，数据将不会自动均衡。主要用于解决单一分片负载和容量过高的问题。

> **说明：**
>
> 暂不支持一次性新增多个物理分片，敬请期待。 扩容分片支持指定时间切换功能，您可以提前发起任务并设置在业务低谷期切换，以降低对您业务影响。

为避免一次性升级全部分片可能导致业务全局中断，TDSQL在线扩容当前仅支持同时新增或扩容一个分片，您需要在当前分片升级完成后才能进行下一个分片。虽然此种方案比较复杂，但根据下方"自动再均衡"原理，整个实例仅影响一部分业务，而不会造成全局中断，因为扩容时仅受影响分片存在秒级的只读（或中断），其他分片的读写业务不会受影响。

自动再均衡

TDSQL主要是采用自研的自动再均衡技术（rebalance）保证自动化的扩容和稳定，以新增分片为例，扩容过程如下图：

1. 控制台单击扩容后，系统根据负载和容量计算出 A 节点（实际上可能影响多个节点）存在瓶颈；
2. 根据新加 G 节点配置，将 A 节点部分数据搬迁（从备机）到 G 节点（可使用系统自动计算值或手工分配）；
3. 数据完全同步后，AG 校验数据库，（存在一到几十秒的只读），但整个服务不会停止。
4. 调度通知 proxy 切换路由。

为确保业务不停以及数据一致性，TDSQL的整个迁移过程采用移存量数据、迁移增量数据、数据检验、再追增量、切换路由、清理六个步骤循环迭代进行。如下图所示：

升级配置

最近更新时间: 2025-02-18 16:02:00

实例购买

操作步骤

1. 登录 分布式数据库 TDSQL 控制台。
2. 单击【新建】购买实例。

> **说明：**
>
> 实例最多一次性购买64个物理分片，超过64个物理分片需 提交工单。

3. 实例购买后，需初始化实例方可正常运行。

实例升级

操作介绍

实例升级操作指将现有 TDSQL 实例的规格升级到更高规格。因分布式数据库是由多个分片组成，因此其实例升级有"新增分片，扩容分片"两种方案。

升级过程中服务不会终止，但某些分片会发生若干秒的只读现象，建议您在低峰期进行升级。

> **说明：**
>
> **云数据库管理平台创建的实例,升级配置时，不可在赤兔管理台操作，需在云数据库管理平台进行升级操作。**

操作步骤

方案一：新增分片

1. 在 TDSQL 控制台，单击实例名或操作列的【管理】，进入实例管理页面。

2. 选择【分片管理】页，在操作列单击【新增分片】，在弹出的对话框选择新增分片的规格和数量，即可升级。

> **说明：**

> 新增分片，请手动选择新增分片的规格和数量，暂不支持自动均衡所有分片的数据。

方案二：扩容分片

1. 在 TDSQL 控制台，单击实例名或操作列的【管理】，进入实例管理页面。

2. 选择【分片管理】页，在操作列单击【调整分片】，在弹出的对话框选择新调整分片的规格，即可升级。

> 说明：
>
> 调整分片指不增加分片数量，但将单个分片的规格扩容到更大。

操作指南
实例管理
创建实例

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍通过 TDSQL 控制台创建实例的操作。

操作步骤

1. 登录 TDSQL 控制台，单击**新建**进入购买页。
2. 根据需求指定数据库实例信息，单击**立即购买**。
   - 地域：实例部署的地域，建议与需要对接的云服务器保持一致。
   - 可用区：在同一地域内电力和网络互相独立的物理数据中心，建议与需要对接的云服务器保持一致。
   - 网络类型：实例所处的网络，建议与需要对接的云服务器保持一致。VPC 网络选择后不可更改，VPC 相关操作请参见 管理私有网络。
   - 实例版本：请参见 实例架构，备机越多，可用性越高。

> **注意：**
>
> 实例架构如果选择一主零备，一般为测试环境或异地灾备环境，并且该架构没有备份功能，请谨慎操作。

- 分片配置：请参见分片配置，计费详情请参见 计费概述。
- 硬盘：默认采用SSD盘（本地盘）。
- 所属项目：是否为必填项可参见《配置管理_用户操作手册_01》，修改tcloud_view_project_required参数属性。
- 参数模板：支持选择不同的参数模板
- 支持字符集：选择 MySQL 数据库支持的字符集。如需在UTF8格式下支持emoji等，请选择UTF8MB4字符集。其他非常见字符集请通过参数设置或请工作人员在后台协助设置。
- 开启强同步：开启强同步可以保证在主机故障时备机数据的一致性，至少需要2个节点方可正常运行。只有大于等于一主二从（3节点）才可配置强同步（不可退化）。
- 购买数量：一次性可购买的实例个数，为避免误操作，一次购买设有上限，如需更多个数，请多次购买。

iii. 在核对信息页确认无误后，单击**立即购买**。
iv. 在**购买实例确认**页面确认信息无误后，单击**立即购买**进行支付。
v. 支付成功后，实例会自动进行创建和初始化操作，返回 TDSQL 实例列表，等待**监控/状态**变为**运营中**，即可进行实例管理操作。

初始化实例

最近更新时间: 2025-02-18 16:02:00

操作场景

本文为您介绍通过 TDSQL 控制台初始化实例的操作。

> **说明：**
>
> 目前从3.10.0版开始实例创建后会自动初始化，用户可跳过此步骤。该操作仅针对旧版未进行初始化操作的实例。

操作步骤

1. 登录 TDSQL 控制台，在实例列表选择**未初始化**的实例，选中复选框，点击列表上方**更多操作**，随后选择**批量初始化**。

2. 在弹出的初始化对话框，根据需要选择配置后，单击**确认**进行初始化实例。

   - 支持字符集：选择 MySQL 数据库支持的字符集。

     如需在UTF8格式下支持emoji等，请选择UTF8MB4字符集。其他非常见字符集请通过参数设置或请工作人员在后台协助设置。

   - 表名大小写敏感：数据库表名大小写是否敏感。

   - 开启强同步：开启强同步可以保证在主机故障时备机数据的一致性，至少需要2个节点方可正常运行。
     - 目前支持强同步、强同步（跨AZ节点优先）、异步复制。
     - 强同步（跨AZ节点优先）仅可用于跨AZ实例，对于单AZ实例无法选择。

   - 强同步可退化

     仅有一个从节点或异步复制，不可关闭或使用强同步可退化。

3. 返回实例列表，待实例**监控/状态**变为**运行中】**，表明初始化实例完成。

查看实例

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍通过 TDSQL 控制台查看实例状态、所属地域、CPU架构、配置、计费模式等信息。

操作步骤

1. 进入TDSQL控制台，点击实例列表，进入实例管理界面。

2. 可通过实例列表查看状态、实例状态信息、所属地域、CPU架构、配置、计费模式等信息。

- 查询实例信息

  点击列表右上方的**搜索框**，单机搜索框，选择资源属性进行过滤，点击回车或搜索图标，进行搜索。

- 刷新列表信息

  点击列表右上方的**刷新**图标，点击刷新按钮进行刷新列表。

- 自定义配置列表字段

  点击列表右上方**设置**图标，点击进入，可自定义展示用户需要的列。

- 导出实例信息

  点击列表右上方**下载**图标，点击，选择需要的下载条件，进行下载。

- 编辑和查看实例标签
  - 选中所需实例，点击操作列**更多**，选中**编辑标签**进行标签维护，可维护多个标签。
  - 查看标签可单击实例名称或者操作列**管理**按钮，进入实例详情界面，基本信息中可查看维护的标签。

- 查看实例详情

  选中所需实例，单击实例名称或者操作列**管理**按钮，进入实例详情界面，查看更多实例信息。

- 重启实例

  选中所要重启的实例，点击操作列**更多**，选中**重启**按钮。如需同时重启多个实例，选中左侧复选框，点击列表上方**重启**按钮，进行重启。

> **警告：**
>
> 重启实例将导致当前连接中断、未提交事务将回滚，并有可能造成业务中断，重启时间将持续5秒~5分钟，请谨慎操作。

- 销毁\退货实例

  选中需销毁的实例，点击操作列**更多**，选中**销毁\退货**按钮。实例彻底销毁后数据将无法找回，请谨慎操作。详细参见销毁实例

- 连接实例

  选中所需实例，点击操作列**登录**，可进入可视化界面，进行数据管理操作。详细参见连接实例 ，数据库管理服务(DMC)。

连接实例

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍如何连接实例，连接后可对数据进行操作。

> **说明：**
>
> 连接实例前要创建账号，账号创建详见创建账号。

连接方式

创建账户后，参照如下方式进行连接

- 腾讯云金融专区 数据库管理服务(DMC)
- Windows 端，以命令行连接、客户端连接和 JDBC 驱动连接三种方式，以JDBC驱动为例。
- Linux 端，以命令行连接为例。

数据库管理服务(DMC)

进入TDSQL控制台，查看实例列表，选中所需访问实例的操作列**登录**，输入账号密码。进入DMC管理界面。此访问方式为可视化界面，用户可以更加直观方便的管理实例。

Windows JDBC 驱动连接

TDSQL 支持程序驱动连接，本文以 Java 使用 JDBC Driver for MySQL 连接 TDSQL 为例。

1. 在 MySQL 官网 下载一个 JDBC 的 jar 包，将其导入 Java 引用的 Library 中。

2. 调用 JDBC 代码如下：

```
public static final String url = "内网地址";
//调用JDBC驱动
public static final String name = "com.mysql.jdbc.Driver";
public static final String user = "用户名";
public static final String password = "密码";
//JDBC连接
Class.forName("com.mysql.jdbc.Driver");
Connection conn=DriverManager.getConnection("url, user, password");
//关闭连接
conn.close();
```

Linux 命令行连接

使用云服务器 CVM 访问数据库的内网地址。CVM 和数据库要在同一地域、同一账号且同一网络类型（都是基础网络或都在同一个 私有网络 VPC ）。

1. 以 CentOS 7.2 64位系统的 CVM 为例，利用 CentOS 自带的包管理软件 Yum 去腾讯云金融专区的镜像源下载安装 MySQL 客户端。
2. 执行以下命令安装客户端：

3. 命令行显示 complete 后，表示 MySQL 客户端安装完成。

4. 输入命令 `mysql -h内网地址 -P端口 -u用户名 -p密码 -c` 连接 TDSQL，下一步即可进行分表操作。
   - 使用MySQL登录命令时，请务必增加-c参数，这样可以使用注释透传功能。
   - 注释透传指支持透传 SQL语句到对应的一个或者多个物理分片（Set），并透传到分表键（Shardkey）对应的分片（Set）中的操作方式。

内网地址转换

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍在业务需要修改数据库实例访问地址时，可使用内网地址转换功能来调整网络。

> **注意：**
>
> - 修改实例内网地址属于高危操作，请务必在业务低谷期谨慎操作。修改内网地址后，原地址将持续生效24小时（除非被另外业务占用），请尽快切换业务配置。
> - 私有网络 VPC 选择后不可更改。

修改内网地址

在 VPC 网络下，云数据库支持修改内网地址。通过实例列表选择所需实例，进入**实例详情**页的**内网地址**处，单击编辑按钮更改，前提条件是当前子网仍有可用 IP。

VPC 网络内切换子网

支持实例在 VPC 网络中切换子网。通过实例列表选择所需实例，进入**实例详情**页的**所属网络**处，单击**更换子网**更改，前提条件是目标 VPC 子网内仍有可用 IP。

由于产品支持同城双活架构，建议您优先选择与业务服务器相同，或与主节点所在地域相同的 VPC 子网。

数据复制方式

最近更新时间: 2025-02-18 16:02:00

数据复制方式

数据复制方式又名数据同步方式，指在数据库高可用方案下，主从数据节点数据复制的机制。云数据库 MySQL/MariaDB 目前支持：

- **异步复制**：应用发起更新（含增加、删除、修改操作）请求，Master 完成相应操作后立即响应应用，Master 向 Slave 异步复制数据。因此异步复制方式下，Slave 不可用不影响主库上的操作，而 Master 不可用有概率会引起数据不一致。
- **强同步（不可退化）复制**：应用发起更新请求，Master 完成操作后向 Slave 复制数据，Slave 接收到数据后向 Master 返回成功信息，Master 接到 Slave 的反馈后再应答给应用。Master 向 Slave 复制数据是同步进行的，因此 Slave 不可用会影响 Master 上的操作，而 Master 不可用不会引起数据不一致。

> **注意：**
>
> 使用强同步复制时，如果主库与从库自建网络中断或从库出现问题，主库也会被锁住（hang），而此时如果只有一个主库或一个从库，是无法做高可用方案的。因为单一服务器服务，如果故障则直接导致部分数据完全丢失，不符合金融级数据安全要求。

- **强同步（可退化）复制**：业务系统中批处理、事务大量写入数据会导致从机严重延迟；加上强同步（ 不可退化 ）仅余单节点会被锁住；这些原本保证数据一致性的机制可能导致业务系统异常。 为解决这个问题，云数据库提供在强同步机制上可退化为异步的方案，从机延迟大于等于十五秒时，系统自动将强同步退化为异步；从机延迟小于十五秒时，系统自动将异步升级为强同步。强同步（可退化）是一种高效的、保证数据最终一致性的方案。

> **注意：**
>
> 此处与 Google 开源的半同步机制不同点在于，强同步采用的是线程池，且不占用工作线程模式，且退化方案优于半同步。

修改数据复制方式

云数据库 MySQL/MariaDB 一主一从仅提供强同步（ 可退化 ）、异步复制两种方案；如需数据一致性，请购买一主二从三节点版本。

修改方式有两种场景：

- 新建实例时，可以在**新建**阶段设定数据复制方式。
- 编辑实例，通过实例列表，进入实例详情页，找到可用性信息->数据复制方式，进行修改。

> **说明：**
>
> 修改过程不影响实例正常运行，修改后小于等于五秒即生效。

分片管理

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍如何管理实例里的不同 set。目前分布式只支持 hash 分片方式，分片总数在实例创建时指定，set 管理页面展示了 实例下每个 set 的分片分布。

操作步骤

1. 进入TDSQL控制台，点击实例列表，选择所需实例。
2. 点击**实例ID/名称**或操作列**管理**，点击分片管理，查看分片列表。
3. 点击分片ID或操作列的**管理**可查看每个分片的详情。
   - 调整分片配置，点击操作列的**调整分片配置**可调整分片的规格，硬盘大小等，并设置切换时间。
   - 新增分片，缓解其他分片负载压力，点击操作列的**新增分片**。
   - 分片的备份信息进行查看，点击操作列**管理**,点击**备份与恢复**，可查看冷备，binlog备份，并可对备份存储时间进行调整。备份具体操作详见备份与恢复备份与恢复。

销毁实例

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍当您不需要某个实例时，可以对实例进行退还，被退还的实例会被隔离。对于隔离中的实例，您可以根据不同场景和需求进行续费、恢复或者释放实例，隔离到期后会彻底销毁，请谨慎操作。

操作步骤

1. 进入TDSQL控制台，点击实例列表，进入实例管理界面。
2. 选择需销毁/退还实例，点击操作列**更多**->**销毁/退货**

> **警告：**
>
> - 实例彻底销毁后数据将无法找回，请提前备份实例数据。
> - 实例彻底销毁后 IP 资源将同时释放，如果该实例有相关的灾备实例，灾备实例将会断开同步连接，自动升级为主实例。

> **说明：**
>
> - 实例退还后，状态变为**隔离**，默认保留72小时。
> - 隔离状态的实例即让实例无法使用（但并非销毁或删除）,实例隔离后将不可被访问，您可以在控制台恢复实例，隔离后资源空间不会被释放且保留最基本的数据副本。隔离到期后，实例彻底销毁。

> **注意：**
>
> 退还后，实例的状态一旦变为“已隔离”时，就不再产生与该实例相关的费用。
>
> - 隔离后，实例 IP 被释放，再次恢复可能无法获得原有 IP。
> - 隔离后，实例无法进行升级、修改参数、创建修改帐号、回档、添加 SET、修改实例名等修改操作。

3. 返回实例列表，实例状态变为“已隔离”，隔离期间可选择**恢复/开机**，隔离到期后，实例彻底销毁。

> **注意：**
>
> 恢复实例是在实例被隔离后恢复实例至正常运行的操作。恢复可能需要几分钟时间，另外，恢复实例可能会重新分配 IP，而非隔离前的 IP。

4. 回收站功能

租户侧执行销毁实例后，实例进入回收站，实例停止服务和计费，进入隔离状态。

进入回收站后，再下线实例会直接删除，如不操作，一定时间之后自动删除。放入回收站的实例重新开机后，重新开始计费。

> **注意：**
>
> 早期版本租户无法直接在租户侧删除实例。而租户运维端或自动删除实例时，会导致订单错误。因此增加回收站功能。

灾备/只读实例

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将为您介绍如何通过控制台创建、管理灾备/只读实例。针对业务连续服务和数据可靠性有强需求或是监管需要的场景，TDSQL MySQL提供跨可用区、跨地域灾备/只读实例，帮助用户以较低的成本提升业务连续服务的能力，同时提升数据的可靠性。

适用场景

- 异地灾备：灾备实例可用于对业务和数据进行多地备份，来保证数据的安全性。 当一个可用区/地域发生故障，可以迅速切换到跨可用区、跨地域的灾备实例，尽可能减少故障对业务的影响。
- 就近接入：业务在一个可用区/地域作为主实例写入，另外的可用区/地域作为只读实例，为用户提供就近接入、异地读能力，改善访问速度。
- 多地域部署：TDSQL MySQL 提供多地域部署能力，当一个可用区/地域遇到网络波动或者不可用的情况，能根据业务情况手动切换到另一个可用区/地域上。

功能特点

- 提供独立的数据库连接地址，灾备/只读实例可提供读访问能力，用于就近接入、数据分析等场景，设备冗余成本低。
- 一个主实例可以创建多个灾备/只读实例，部署在不同的地域、可用区。
- 灾备/只读实例支持一主一从、一主两从高可用架构，避免了数据库的单点风险。
- 如果主实例发生故障，可在数秒内激活灾备/只读实例，恢复完整读写功能。
- 灾备/只读实例通过内网专线同步，具有较低的同步时延和更高的稳定性，同步链路质量远优于公网网络。
- 目前推广期专线流量费用免费，商业化收费时间将另行通知。

功能限制

- 灾备/只读实例暂不支持：参数设置、帐号管理功能。
- 灾备/只读实例数据库版本默认保持与主实例相同，实例规格、硬盘大小需要大于等于主实例。

操作步骤

1. 登录 TDSQL控制台，进入实例列表，单击**实例名**或操作列所在的**管理**，进入实例管理页面。

   在实例详情页的实例架构图中，单击**添加灾备/只读实例**，进入实例购买页。

2. 在购买页中，设置灾备/只读实例的“计费模式”、“地域”等基本信息，单击**立即购买**。

> **注意**：
>
> - 创建灾备 / 只读实例会立即同步数据，创建时长受数据量的影响，期间主实例的控制台操作会被锁定，请妥善安排。
> - 暂只支持整个实例数据同步，请确保磁盘空间充足。
> - 请确保主实例状态为运行中并且没有任何任务执行，不然同步任务有可能失败。
> - 强同步建议同城实例之间使用，否则可能影响性能。

3. 支付完成后，返回实例列表，等待实例自动初始化成功后，**监控/状态**变为运行钟，即可进行后续操作。

4. 灾备/只读实例管理

   - **查看灾备/只读实例** 灾备/只读实例可在其所在地域查看，也可在实例列表选择**实例类型**筛选出该地域全部灾备/只读实例。

- **查看从属关系** 在实例详情页的实例架构图中，可查看从属关系。
- **灾备/只读实例功能** 灾备/只读实例提供实例详情、监控告警、数据安全性、备份与恢复、性能优化功能。
- **灾备/只读实例升级为主实例**

  当您需要切换灾备/只读实例为主实例时，可在控制台主动切换灾备/只读实例为主实例。

  - 登录TDSQL控制台 ，进入实例列表，选择所需灾备/只读实例，单击**实例名**或操作列**管理**，进入实例管理页面。
  - 在实例管理页面，单击右上角的**切换为主实例**，即可将灾备/只读实例升级为主实例。切换后将断开与原主实例的同步连接，恢复实例数据库数据写入能力和完整的 MariaDB 功能。

> **警告：**
>
> **同步连接断开后不可重连，请谨慎操作。**

就近接入

最近更新时间: 2025-02-18 16:02:00

操作场景

多可用区部署时若开启就近接入，在本可用区有 Proxy 节点可以访问的情况下，只访问本可用区的 Proxy，从而降低延迟。默认关闭，只对新建立的连接生效。

> **注意：**
>
> 本功能仅在多可用区下支持，单可用区无此开关。

操作步骤

1. 进入TDSQL控制台，点击实例列表，选择所需实例。
2. 在**实例详情**>**可用性信息**中，找到**就近接入**选项，选择打开或关闭。

数据库管理
创建账号

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍通过TDSQL控制台创建账号，以及修改账号权限的操作。

- 因信息安全要求，暂不支持通过命令行 `insert into mysql.user` 、 `grant` 、 `drop` 创建、修改账号。
- 创建账号后可用于连接实例。

操作步骤

1. 登录 TDSQL 控制台，在实例列表选择选择实例，单击实例名或操作列的**管理**，进入实例管理页面。
2. 选择**账号管理**页，单击**创建账号**。
3. 在**创建账号**对话框，输入账号名、创建只读账号、主机、密码、备注，确认无误后单击**确认,下一步**。

> **注意**：
>
> - **账号名**：信息安全角度不建议采用admin、test等名称。从安全考虑，云数据库MySQL/MariaDB不提供root账号。
> - **主机**：可理解为 HOST，支持 IP、IP 段、%三种形式；%代表结尾符，例如，要支持10.10.10.1 - 10.10.10.254的所有主机IP，可以输入10.10.10.%或%。(10.10.10%为错误语法）。如需支持相同账号多个主机，请使用克隆账号功能创建多个。主机名实际是网络出口地址，支持%的匹配方式，代表所有 IP 均可访问。
> - **创建为只读帐号**：选中表示该账号只能接受读请求（select）。
> - **密码**：密码强度支持配置，默认关闭。通过变量pwd_rule_is_open可控制打开，详见《TDSQL MYSQL版本 运维手册》私有变量 章节。
> - 同一个账号不同的主机 IP 需要独立设置权限，可通过账号管理中【克隆账号】功能，快速克隆相似的账号以及设置权限。

4. 进入**修改权限**对话框，根据需求分配权限后，单击**保存设置**即可完成权限分配。若需稍后设置权限，单击**之后设置**即可。

> **说明**：
>
> - 权限分为全局权限和对象级权限。
> - 权限管理可以细化到列级，点击对象级特权进行配置。
> - 权限包括数据库常见的19种权限，可以为表、视图、函数、触发器等对象设置权限。
> - 基于信息安全考虑，建议您每3个月更换一次账号密码。

5. 返回账号列表，单击**修改权限**可以修改用户权限，单击**克隆账号**可以完全复制当前账号权限来新建一个帐号，单击**更多**可以重置密码和删除账号。

读写分离

最近更新时间: 2025-02-18 16:02:00

操作场景

为保证数据安全，需要创建只读账号，只可进行数据查询操作。

操作步骤

1. 登录 TDSQL 控制台，进入实例列表，单击实例名或操作列的**管理**，进入实例管理页面。
2. 选择**帐号管理**页，单击**创建帐号**。
3. 在弹出的对话框设置帐号信息，**创建为只读帐号**设置为**是**，单击**确认，下一步**。
4. 在弹出的对话框，可以设置**只读请求分配策略**，定义在备机故障（或延迟较大）时的读策略。
   - 选择**主机**则备机延迟超时时从主机读取。
   - 选择**直接报错**则备机延迟超时时报错。
   - 选择**只从备机读取**则忽略延迟参数，一直从备机读取（一般用于拉取 binlog 同步）。
   - 定义**只读备机延迟参数**，定义数据同步延迟时间，并与**只读请求分配策略**中的**主机**及**直接报错**两种策略配合使用。

基于注释的读写分离

在每条需要从机“读”的 SQL 前，增加 **/*slave*/** 字段，即可自动将“读”请求分配到从机。并且 mysql 连接命令后面增加 **-c**

```
-- 主机读
select * from emp order by sal，deptno desc；
-- 从机读
/*slave*/ select * from emp order by sal，deptno desc；
```

- 该功能仅支持从机读（select），不支持其他操作，非 select 语句将失败。
- mysql 后面要增加 -c 参数来解析注释。
- /*slave*/ 必须为小写，语句前后无空格。
- 从机出现异常而影响到 MAR（强同步）机制时，从机读操作将自动切换回主机。

> **说明：**
>
> 本版本新增若干基于/*slave*/读写分离策略，详情请参考：《TDSQL MySQL 10.3.20 分布式开发指南》

参数配置

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍如何维护数据库基本参数，比如最大连接数max_connection，事务隔离级别tx_isolation等。根据系统给出的**参考修改值**，可视化的进行参数配置。

操作步骤

1. 进入TDSQL控制台，点击实例列表，选择所需实例。
2. 点击**实例ID/名称**或操作列**管理**，点击参数配置，查看参数列表。
3. 根据系统给出的参考修改值，可单条修改和点击上方的**批量修改参数**。点击参数明旁的图标，可显示参数的作用。

表类型

最近更新时间: 2025-02-18 16:02:00

TDSQL分布式实例支持创建分表、单表和广播表。

分表：即水平拆分表（shard表），该表从业务视角是一张完整的逻辑表，但proxy根据分表键（shardkey）和不同的分表策略，将数据分布到不同的节点（set）中。

单表：又名 Noshard 表，无需拆分，且没有做任何特殊处理的表，目前分布式实例将该表默认存放在第一个物理节点组（set）中。

广播表：又名小表广播功能，即设置为广播表后，该表的所有操作都将广播到所有节点（set）中，每个 set 都有该表的全量数据，常用于业务系统的配置表等。

> **注意**：
>
> 在分布式实例中，如果两张表分表键相等，这意味着，两张表相同的分表键对应的行，一定存储于相同的物理节点组中。这种场景通常被称为组拆分（groupshard），会极大提高业务联合查询等语句的处理效率。
>
> 由于单表默认放置在第一个 set 上，如果在分布式实例中建立了大量的单表，则会导致第一个 set 的负载太大。 除特殊情况外，建议在分布式实例中尽量都使用分表。
>
> 分表键（shardkey）选择的限制请参考开发指南.

- 在分布式实例中，如果两张表**分表键**相等，这意味着，两张表相同的分表键对应的行，一定存储于相同的物理节点组中。这种场景通常被称为组拆分（groupshard），会极大提高业务联合查询等语句的处理效率。
- 由于单表默认放置在第一个 set 上，如果在分布式实例中建立了大量的单表，则会导致第一个 set 的负载太大。 除特殊情况外，建议在分布式实例中尽量都使用分表。
- 分表键（shardkey）选择的限制请参考开发指南.

分表操作

最近更新时间: 2025-02-18 16:02:00

插入数据

insert 字段必须包含分表键(shardkey)，否则会拒绝执行。

向刚刚建立的表中插入数据，代码示例如下：

```
mysql> insert into test1(id,name) values (1,'Alice');
Query OK,1 rows affected(0.08 sec)

-- 未指定shardkey情况
mysql> insert into test1(name,addr) values('example','shenzhen');
ERROR 7013 (HY000): Proxy ERROR:get_shardkeys return error
```

查询数据

查询数据时，最好带上分表键(shardkey)，proxy网关将自动路由到对应分片，此时效率最高。否则，会自动全表扫描，然后在网关进行结果聚合，效率较低。

查询数据代码示例如下：

```
mysql> select id from test1 where id=1;
```

删除数据

delete 必须带有 where 条件，where 条件建议带上分表键。

删除代码示例如下：

```
mysql> delete from test1 where id=1;
Query OK, 1 row affected (0.02 sec)
```

建表

分表为例，建分表时，需指明分表键（shardkey），代码示例如下：

```
mysql> create table test1(id int primary key,name varchar(20),addr varchar(20))shardkey=id;
Query OK,0 rows affected(0.15 sec)
```

监控告警
指标监控

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍实例的指标监控，为方便用户查看和掌握实例的运行信息，MySQL/MariaDB 提供丰富的指标监控项。

操作步骤

1. 进入TDSQL控制台，点击实例列表，进入实例管理界面。
2. 点击**实例ID/名称**或操作列**管理**，点击监控告警->指标监控。
3. 用户查询维护可选择分钟、小时、天、周、月维度进行查看实例、分片、节点的监控指标，也可进行数据对比，支持同比、环比、自定义日期对比。

> **说明：**
>
> 每个指标监控看板可单独配置告警策略。

- 监控数据导出，选择下方每个指标监控看板，点击右上角导出标识图标。
- 已提供监控指标如下

指标名	单位	注意事项
客户端总连接数(DB Connections)	个	客户端到数据库服务器的连接总数。
SQL总数	次/秒	所有 DDL、DML、DCL 的总数量。
SQL错误数	个	所有 DDL、DML、DCL 中运行错误的总数量，该值如果过大，请尽快检查业务日志。
SQL 成功数	次/秒	所有 DDL、DML、DCL 中运行成功的总数量。
耗时小于5ms请求数	次/秒	执行时间小于5ms的请求数。
耗时5到20ms请求数	次/秒	执行时间5-20ms的请求数。
耗时20到30ms请求数	次/秒	执行时间20~30ms的请求数。
耗时大于30ms请求数	次/秒	执行时间大于30ms的请求数。
活跃线程数	个	客户端到数据库服务器的活跃连接总数。
DELETE请求数	次/秒	Delete请求数。
数据磁盘空间利用率	%	指数据占用空间、日志占用空间、临时占用空间、系统文件占用空间占购买磁盘空间的比例，建议小于80%，否则需增加磁盘空间。
innodb缓冲池预读页次数	次	用于分析 innodb 存储引擎当前性能的指标。
innodb缓冲池读页次数	次	用于分析 innodb 存储引擎当前性能的指标。

指标名	单位	注意事项
innodb磁盘读页次数	次	用于分析 innodb 存储引擎当前性能的指标。
innodb执行DELETE行数	行	用于分析 innodb 存储引擎当前性能的指标。
innodb执行INSERT行数	行	用于分析 innodb 存储引擎当前性能的指标。
innodb执行READ行数	行	用于分析 innodb 存储引擎当前性能的指标。
innodb执行UPDATE行数	行	用于分析 innodb 存储引擎当前性能的指标。
NSERT请求数	次/秒	累加实例各个分片主节点的INSERT请求数。
IO利用率	%	IO利用率。
慢查询数	次	SQL 语句执行时间超过 long_query_time 设置值的语句数据量，详细情况可至性能优化页面查看详情。
主从切换次数	次	发生主机切换到从机的情况。
可用缓存空间	GBytes	实际采集为 Innodb_buffer 的可用空间，因数据库通常采用 LRU 调度方案，正常情况下该值将趋于零；处理大事务时，该值可能为负，即数据库内存使用超过实际分配值。
缓存命中率	%	SELECT 或预处理查询直接从内存中获取数据的比例，建议大于90%，否则需增加内存规格。
REPLACE_SELECT请求数	次/秒	replace select 语句的总数量。
REPLACE请求数	次/秒	replace 语句的总数量。
汇总主备节点总请求数	次/秒	累加实例所有主节点总请求数和所有备节点的Select请求数。
SELECT请求数	次/秒	select 语句的总数量。
备延迟	秒	从机与主机数据延迟，强同步的原理是将数据写入从机 binlog 便返回事务应答，此时数据尚未完全写入磁盘，因此仍然会有延迟。
可用数据磁盘空间	GBytes	可用数据磁盘空间。
CPU利用率	%	MySQL/MariaDB 采用灵活的 CPU 限制，允许您的实例闲时使用设备额外 CPU 资源，此时 CPU 利用率会超过100%。
当前打开连接数	个	show processlist得到的session个数。
DB连接使用率	%	ThreadsConnected/ConnMax。
最大连接数	个	最大连接数。
已用Binlog日志磁盘空间	GBytes	已用Binlog日志磁盘空间。
剩余Binlog日志磁盘空间	GBytes	剩余Binlog日志磁盘空间。

指标名	单位	注意事项
UPDATE请求数	次/秒	UPDATE请求数。

实例告警

最近更新时间: 2025-02-18 16:02:00

操作场景

用户可对关心的指标设置告警，通过配置告警对象、触发条件，触发实例告警。

操作步骤

1. 进入TDSQL控制台，点击实例列表，进入实例管理界面。
2. 点击**实例ID/名称**或操作列**管理**，点击监控告警->实例告警。
   - 可根据天数条件筛选，查看一段时间的告警信息。
   - 设置告警，两种方式进入配置：
     - 当前页面，右上角，点击告警设置，跳转**新建告警策略**页面，策略类型选择**云数据库/TDSQL**
     - 通过指标监控界面，点击每个指标监控看板右上角**配置告警**图标。

性能优化
性能检测

最近更新时间: 2025-02-18 16:02:00

操作场景

性能检测是针对数据库实例性能及运行状况综合分析的服务，针对 SQL 语句的性能、CPU 使用率、IOPS 使用率、内存使用率、磁盘空间使用率、连接数、锁信息、热点表、事务等进行综合分析，性能检测提供智能的诊断及优化功能，能最大限度发现数据库存在或潜在的健康问题。

目前如下版本实例支持此功能：

- TDSQL MySQL 版
- 云数据库 MariaDB

> **注意：**
>
> 针对某些检测结论，性能检测报告提供了一系列的优化建议，请您在应用这些建议前谨慎测试，以防加重实例的性能问题。

操作步骤

登录TDSQL控制台，进入实例列表，点击**实例ID/名称**或操作列**管理**，选择**性能优化**>**性能检测**页，可进行性能检测。

- **健康评分：**您可以看到当前数据库性能综合打分，满分100分；长期低于60分请注意优化业务或数据库配置。
- **报告生成、查看与保存：**您可以任意创建报告；并查看最近一次生成的报告；报告可以通过另存网页下载到本地查阅。

性能检测主要功能

资源分析

分析一定时间内数据库实例资源（CPU、磁盘、连接）的使用情况，并综合评分。

由于多数实例默认开启了闲时超用策略，因此您可能观察到最大 CPU 使用率可能超过100%。如果您的 CPU 长期超过100%，且平均值高于建议值，建议您尽快扩容。

系统状态

梳理实例关键指标，并列举其状态、出现时间和提出对应修改建议。

表空间分布

列出按数据空间倒序的当前 TOP 10 表，协助您分析超大表情况。

冗余索引检测

列出当前可能的冗余索引（区分度小于1%的冗余），并提出优化建议。

由于查询语句要先查询索引，再通过索引去查询表，所以，如果索引列相同数据过多不利于减少筛选的数据量，反而不如直接全表扫描性能快。

死锁诊断

死锁诊断通过诊断 `show engine innodb status` 获取 DB 最后一次死锁信息，如果死锁发生时间在用户选择的诊断时间段内，便展示出来。

死锁出现频率过高代表事务内的 SQL 在并发执行场景中的持锁容易产生环路，根本解决方案是修改 SQL 运行逻辑顺序，优化加锁机制，减少死锁产生概率。临时解决方案是 kill 掉领头的阻塞会话。

锁等待诊断

当前时间段内的锁等待超过60s的报告。

- 有锁等待是正常现象，但有时候业务会出现 `Lock wait timeout exceeded;try restarting transaction` 锁等待超时等报错。MySQL 的 innodb 锁信息保存在系统库 `information_schema` 中的 `innodb_trx`，`innodb_lock_waits`，`innodb_locks` 三张表下，锁等待诊断通过分析诊断 set 主 DB 中的三张表的锁依赖关系，找出持有锁时间超过一定阈值，并阻塞的其他会话的领头事务信息和会话信息，以及被阻塞事务的会话信息，并 kill 掉该领头会话。
- 当前锁等待只支持 InnoDB 引擎。

长会话诊断

列通过诊断 set 主 DB 中的 `information_schema.processlist` 获取 Command 不是 Sleep，但执行时间（Time）超过10s的会话。

最佳解决长会话的手段是优化 SQL，并在业务代码中主动植入 session 失效配置，当然，您也可以通过调整 `interactive_timeout`、`wait_timeout` 两个参数，让过期 session 主动失效。

慢查询分析

基于执行次数倒序，列出当前 TOP 20 的慢查询语句。

慢查询可以通过 `long_query_time` 配置调整；慢查询产生的原因产生较多。通常，如果您的实例消耗资源合理且慢查询较多，则建议您关注业务 SQL、索引是否合理；如果您实例消耗性能较高且慢查询较多，建议您关注实例配置是否合理，并优化业务 SQL、索引等。慢查询数据可以在慢查询分析功能下，查询更多详细数据。

DB 状态检查

检查当前数据库 DB 层的健康状态。

其他

列出需要 DBA 关注的其他值。

慢查询分析

最近更新时间: 2025-02-18 16:02:00

操作场景

将超过指定时间的 SQL 查询语句称为“慢查询”，对应语句称为“慢查询语句”，而数据库管理员（DBA）对慢查询语句进行分析并找到慢查询出现原因的过程叫做慢查询分析。

操作步骤

1. 登录TDSQL控制台，进入实例列表，点击**实例ID/名称**或操作列**管理**
2. 选择**性能优化**>**慢查询分析**页，可进行慢查询分析。慢查询分析可拆分到某一个库，某个分片，主机\从机进行分析。

主要参数说明

主要默认设置

- 慢查询功能：默认开启。
- 慢查询时间（long_query_time）：默认配置为1秒；即慢查询语句查询时间超过1秒的才被记录。
- 分析数据输出延迟：1分钟 - 5分钟。
- 日志记录时长：30天，根据备份和日志设置周期决定。

分析列表字段说明

- 校验值（checksum）：表示慢查询语句的一串序列数字，默认64bit。
- 抽象后的慢查询语句（fingerprint）：隐去用户数据以后的慢查询语句。
- 数据库：出现慢查询语句的数据库。
- 帐号：出现慢查询语句的帐号。
- 最后执行时间（last_seen）：时间范围内，最后一次出现慢查询语句的时间。
- 首次执行时间（first_seen）：时间范围内，第一次出现慢查询语句的时间。
- 总次数（ts_cnt）：时间范围内，慢查询语句出现的次数。
- 总次数占比：时间范围内，慢查询语句占所有慢查询语句次数的占比。
- 总时间（query_time_sum）：时间范围内，慢查询语句查询总耗时。
- 总时间占比：时间范围内，慢查询语句查询总耗时的占比。
- 平均时间（query_time_avg）：慢查询语句总时间除以总次数的平均时间。
- 最小时间（query_time_min）：慢查询语句出现的最小时间。
- 最大时间（query_time_max）：慢查询语句出现的最大时间。
- 总锁时间（lock_time_sum）：慢查询语句出现锁的总耗时。
- 总锁时间占比：时间范围内，慢查询语句占所有慢查询语句锁时间的占比。
- 平均锁时间（lock_time_avg）：慢查询语句总锁时间除以总锁次数的平均时间。
- 最小锁时间（lock_time_min）：慢查询语句锁出现的最小时间。
- 最大锁时间（lock_time_max）：慢查询语句锁出现的最大时间。
- 发送行数（Rows_sent_sum）：该条慢查询语句发送的数据行数总和。
- 扫描行数（Rows_examined_sum）：该条慢查询语句扫描的数据行数总和。

备份与恢复
备份方式

最近更新时间: 2025-02-18 16:02:00

自定义备份时间

- 登录 TDSQL 控制台，进入实例列表，单击实例名或操作列的**管理**，进入实例管理页面。
- 选择**分片管理**页，单击分片 ID或操作列的**管理**，进入分片管理页面。
- 选择**备份与恢复**>**备份和日志设置**页，单击**存储时间**旁的编辑图标可设置存储时间。
  - 备份周期：目前默认每天执行备份任务。
  - 存储时间：数据和日志备份文件保留的天数，默认为7天，可设置保存最大天数时间为 [1,7]。

备份类型

TDSQL MySQL 版支持全量备份和增量备份。备份采用 LZ4 方式压缩，使用方式请参见解压备份和日志文件。

全量备份

全量备份可设置备份程序启动时刻和备份保存时长，默认启动时刻为凌晨00：00 - 05:00性能较低时刻，备份保存时长默认为7天。

增量备份

增量备份以 binlog 方式提供，binlog 实时生成（将占用一定数据盘空间），并定期上传至云数据库备份系统。

下载备份和日志文件

最近更新时间: 2025-02-18 16:02:00

操作场景

用户可通过分布式数据库 TDSQL 控制台下载云数据库的冷备数据、binlog、日志文件。

操作步骤

1. 登录 TDSQL 控制台，进入实例列表，单击实例名或操作列的**管理**，进入实例管理页面。
2. 选择**分片管理**页，单击分片 ID，进入分片管理页面。
3. 选择**冷备列表**或**Binlog 列表**，在操作列单击**下载**。
4. 在弹出的下载对话框，提供 VPC 网络地址，单击**复制**获取地址。
   - 为保证数据安全，目前暂只提供内网地址，地址有效期为15分钟，过期后请刷新页面重新获取，VPC 网络地址请在 VPC 网络进行访问。
   - 推荐您复制下载地址，并登录到云数据库所在 VPC 下的 CVM（Linux 系统）中，运用 wget 命令进行下载。
5. 下载慢查询日志、错误日志，选择**性能优化**->**慢查询日志**或**错误日志**页。操作同上3、4步骤。若文件大小为0KB，无记录，则无法下载。

解压备份和日志文件

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍针对备份文件，介绍如何进行解压。

操作步骤

出于压缩性能和压缩比的综合考虑，MySQL/MariaDB 的备份文件和日志文件（binlog 文件）采用 LZ4（Extremely Fast Compression algorithm）工具进行压缩，您可以选用 LZ4 工具进行解压。由于常见的解压工具不支持该格式，本文特别给出解压工具和操作指引。

Windows

下载工具

工具下载地址

安装工具

双击 zip 文档，解压后得到 LZ4installv1.4.exe ，双击运行，按指引完成安装。 如果只是解压我们的文件，最后一步的复选框可以忽略。

解压文件

右键单击需要解压的 lz4 文件，选择 **Decode with LZ4** 项即可完成解压。

Linux

安装工具

腾讯云金融专区云服务器的 yum 库中有 LZ4 组件，登录云服务器执行如下命令即可安装。

```
$ yum install lz4
```

直接执行 **lz4** 返回类似如下图所示，表示安装正确。

解压文件

执行如下命令即可完成解压。

```
$ lz4 -d xxx.lz4
```

利用备份文件恢复实例

最近更新时间: 2025-02-18 16:02:00

操作场景

TDSQL MySQL 版、MariaDB可通过回档功能来查看历史数据，如果您需要在本地恢复您的数据库实例，可按照本文的步骤进行操作恢复历史数据。

操作步骤

准备服务器

如您需在本地恢复数据库实例，请确保服务器基本配置如下：

- CPU：2核或以上。
- 内存：4G或以上。
- 磁盘空间：必须超过数据库已用空间并留足系统所需的临时空间。
- 操作系统：centos。

准备数据库

下文以安装 MariaDB 10.0.10 为例：

1.添加 yum 源。

```
vi /etc/yum.repos.d/mariadb-10.0.10.repo):
# MariaDB 10.0 CentOS repository list - created 2016-05-30 02:16 UTC
# http://imgcache.finance.cloud.tencent.com:80downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
# baseurl = http://imgcache.finance.cloud.tencent.com:80yum.mariadb.org/10.0/centos7-amd64
baseurl = http://imgcache.finance.cloud.tencent.com:80archive.mariadb.org/mariadb-10.0.10/yum/centos6-amd64/
gpgkey=http://imgcache.finance.cloud.tencent.com:80yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=0
```

2.检查配置 yum 源对应的 MariaDB 版本是否为10.0.10。

```
yum makecache
yum info MariaDB-server
```

3.安装 MariaDB-server。

```
yum install MariaDB-server
```

> **说明：**
>
> 提示与旧版本冲突时，需要先移除之前的安装包，例如 `yum remove mariadb-libs`

安装辅助工具

1.安装 MariaDB 客户端。

```
yum install MariaDB-client
```

2.安装 LZ4 解压软件，请参见 解压备份文件和日志文件。LZ4 默认安装在 `mysqlagent/bin` 目录下，也可以将其放置在 `/usr/bin` 目录下，当环境变量引用。

```
yum install -y lz4
percona-xtrabackup
yum install http://imgcache.finance.cloud.tencent.com:80www.percona.com/downloads/percona-release/redhat/642959856401
40800.1-3/percona-release-0.1-3.noarch.rpm
yum install percona-xtrabackup
```

下载备份

在 MySQL/MariaDB 控制台，单击实例名进入实例管理页，在【备份与恢复】页获取备份下载地址。 下载命令示例：

```
wget --content-disposition 'http://imgcache.finance.cloud.tencent.com:801x.2xx.0.27:8083/2/noshard1/set_1464144850_587/14
64552298xxxxxxxx'
```

通过备份文件恢复数据库（未加密）

1.进入备份文件下载目录，通过 lz4 解压冷备文件

```
lz4 -d set_1464144850_587.1464552298.xtrabackup.lz4
```

2.使用 xbstream 工具解压到临时目录 xtrabackuptmp

```
mkdir xtrabackuptmp/
mv set_1464144850_587.1464552298.xtrabackup xtrabackuptmp/
xbstream -x < set_1464144850_587.1464552298.xtrabackup
```

解压完后，目录和文件内容如下：

3.使用 innobackupex 应用日志

```
mkdir /root/dblogs_tmp
innobackupex --apply-log --use-memory=1G --tmpdir='/root/dblogs_tmp/' /root/xtrabackuptmp/
```

操作成功后，会显示 `completed OK!` ，如下所示：

4.停止数据库，清空数据文件

```
service mysql stop
```

清空数据文件（数据目录、表空间目录、日志目录）：

```
mkdir /var/lib/mysql-backup
mv /var/lib/mysql/* /var/lib/mysql-backup
```

5.修改数据库参数文件

修改数据库参数文件（/etc/my.cnf.d/server.cnf），具体参数数值请参考解压文件中 backup-my.cnf 的参数。**不能直接用 backup-my.cnf 替换参数文件。**

```
[mysqld]
skip-name-resolve
```

```
datadir=/var/lib/mysql
innodb_checksum_algorithm=innodb
innodb_log_checksum_algorithm=innodb
innodb_data_file_path=ibdata1:2G:autoextend
innodb_log_files_in_group=4
innodb_log_file_size=1073741824
innodb_page_size=4096
innodb_log_block_size=512
innodb_undo_tablespaces=0
```

6.使用 innobackupex 加载镜像

```
innobackupex --defaults-file=/etc/my.cnf --move-back /root/xtrabackuptmp/
```

成功后，显示 `Complete OK!` ，如下所示：

7.启动数据库

```
chmod 777 -R /var/lib/mysql
service start mysql
```

如果遇到启动失败，则需要检查错误信息，修复后再重新启动。

8.连接数据库查看数据

数据库启动后，您可能需要通过原来的账号密码来连接数据库查看数据。

通过备份文件恢复数据库（已加密）

数据透明加密（TDE）当前仅支持 5.7 、8.0.24版本，后续将陆续开放。您可以通过在 MariaDB 控制台 的实例管理页的数据安全性 > 数据加密进行访问。

开启数据加密后，暂时不支持用备份文件在本地恢复数据库实例，推荐采用 回档数据库 进行恢复。

回档数据库

最近更新时间: 2025-02-18 16:02:00

回档说明

- TDSQL 可以根据备份和日志保持情况，回档到7天内的任意时刻（回档时间取决于您的备份和日志文件实际存储时长）。通过数据库回档能力，可以最大程度地减少系统损失。
- TDSQL MySQL版回档功能不会影响现网生产实例，可直接回档到私有云创建的一个新的按量计费实例。回档的新按量计费实例为一个标准的实例，用户可以根据需要自行选择配置。

限制条件

- 回档、创建实例和实例切换过程中，现网生产实例的部分管理功能将不可用，操作完成即可恢复数据库管理操作。
- 回档操作有可能会对二进制日志（binlog）进行强行分片，即未达到100MB也会被备份为一个独立文件。
- 回档后的新购买实例会具备现网生产实例的参数信息（如账户、数据库参数等），请注意账号管理。

回档实例

1. 登录 TDSQL 控制台，单击实例名进入实例管理页。
2. 选择**备份与恢复** > **克隆实例**，单击**新建克隆实例**。
3. 在弹出的对话框，设置克隆时间，单击**确定**。
4. 在新购实例页面，根据需要变更配置，单击**立即购买**后等待实例回档完成。
5. 操作回档后，可在**备份与恢复** > **回档实例**页查看生成的回档实例，回档实例在实例列表中也可以正常查看。

安全管理
数据安全性
安全组

最近更新时间: 2025-02-18 16:02:00

操作场景

安全组的设置用来管理云服务器是否可以被访问，您可以通过配置安全组的入站和出站规则，设置您的服务器是否可以被访问以及访问其他网络资源。安全组具体详细见安全组。

操作步骤

1. 进入TDSQL控制台，点击实例列表，选择所需实例。
2. 点击**实例ID/名称**或操作列**管理**，点击**数据安全性**->**安全组】**
   - 可以查看出站和入站规则
   - 点击安全组ID可跳转页面，对安全组规则进行编辑。

> **注意：**
>
> 云数据库安全组不需要指定端口号或协议，已设置端口号的安全组规则对云数据库不生效。

数据加密

最近更新时间: 2025-02-18 16:02:00

操作场景

本文将介绍如何将核心数据，特别是企业重要数据资产，进行加密。

> **说明：**
>
> - 当前仅MySQL/Percona 5.7或以上兼容版本支持数据加密功能；由于加密会消耗较多CPU性能，请充分测试后再启动。
> - 加密算法缺省为 AES，如有国密需求，请到参数配置处将 innodb_encryption_algorithm 参数调整成 SM4；当前仅 MySQL 8.0 内核支持国密算法；调整不会影响存量加密库表的加密算法。

操作步骤

1. 进入TDSQL控制台，点击实例列表，选择所需实例。
2. 点击**实例ID/名称**或操作列**管理**，点击**数据安全性**->**数据加密**
   - 该功能是对数据文件进行加密，比如Innodb引擎下的 .idb文件
   - 该功能开启之后，用户需要手动对数据库表进行加解密操作。
   - 加密后 用户可对比.idb文件查看，发现加密后的文件为乱码显示。

```
-- 表创建时进行加密
CREATE TABLE t1 (c1 INT) ENCRYPTION='Y';
-- 表创建后进行加密处
ALTER TABLE t1 ENCRYPTION='Y';
-- 表解密
ALTER TABLE t1 ENCRYPTION='N';
```

> **注意：**
>
> - 数据加密功能可能产生密钥管理服务（KMS）费用
> - 如需关闭本功能，需解密全部表空间，且关闭KMS服务后，提交工单处理

访问管理
概述

最近更新时间: 2025-02-18 16:02:00

如果您在云上中使用到了云数据库、云服务器、私有网络等服务，这些服务由不同的人管理，但都共享您的云账号密钥，将存在以下问题：

- 您的密钥由多人共享，泄密风险高。
- 您无法限制他人的访问权限，易产生误操作造成安全风险。

这个时候，访问管理应运而生。 有关访问管理 CAM 的更多介绍，请参见 CAM 概述。

如果您不需要对子账户进行云数据库相关资源的访问管理，您可以跳过此章节。跳过这些部分并不影响您对文档中其余部分的理解和使用。

接入CAM 后，可通过子账号实现不同的人管理不同的服务，以避免出现以上的问题。默认情况下，子账号没有使用云数据库实例以及云数据库相关资源的权限。因此，我们就需要创建策略来允许子账号使用他们所需要的资源或者权限。

策略是定义和描述一条或多条权限的语法规范，策略通过授权一个用户或者一组用户来允许或拒绝使用指定资源。CAM 策略的更多基本信息请参见 策略语法，使用信息请参见 策略。

入门

CAM 策略必须授权使用一个或多个云数据库操作或者必须拒绝使用一个或多个云数据库操作。同时还必须指定可以用于操作的资源（可以是全部资源，某些操作也可以是部分资源），策略还可以包含操作资源所设置的条件。

> **说明：**
>
> - 建议用户使用 CAM 策略来管理云数据库资源和授权云数据库操作，对于存量分项目权限的用户体验不变，但不建议再继续使用分项目权限来管理资源与授权操作。
> - 云数据库暂时不支持相关生效条件设置。

策略结构

最近更新时间: 2025-02-18 16:02:00

策略语法

CAM 策略配置示例：

```
{
"version":"2.0",
"statement":
[
{
"effect":"effect",
"action":["action"],
"resource":["resource"],
"condition": {"key":{"value"}}
}
]
}
```

- **版本 version**：必填项，目前允许值为"2.0"（该值实际代表 CAM 接受的云 API 版本）。
- **语句 statement**：用来描述一条或多条权限的详细信息。该元素包括 effect、action、resource、condition 等多个其他元素的权限或权限集合。一条策略有且仅有一个 statement 元素。
- **操作 action**：用来描述允许或拒绝的操作。操作 action 实际填入的是以 "dcdb:" 前缀描述，TDSQL MySQL版 API 为后缀的一串字符串 。该元素是必填项。
- **资源 resource**：描述授权的具体数据。资源是用六段式描述。每款产品的资源定义详情会有所区别。有关如何指定资源的信息，请参阅您编写的资源声明所对应的产品文档。该元素是必填项。
- **生效条件 condition**：描述策略生效的约束条件。条件包括操作符、操作键和操作值组成。条件值可包括时间、IP 地址等信息。有些服务允许您在条件中指定其他值。该元素是非必填项。
- **影响 effect**：描述声明产生的结果是"允许"还是"显式拒绝"。包括 allow（允许）和 deny（显式拒绝）两种情况。该元素是必填项。

> **说明**：
>
> 由于历史原因，分布式数据库 TDSQL（曾用名 DCDB）在访问管理的接口关键词为 dcdb。

云数据库的操作

在云数据库策略语句中，您可以从支持云数据库的任何服务中指定任意的 API 操作。对于云数据库，请使用以 dcdb: 为前缀的 API。例如：dcdb:CreateDBInstance （创建实例-包年包月）或者 dcdb:CloseDBExtranetAccess（关闭外网访问）。

1. 如果您要在单个语句中指定多个操作的时候，请使用英文逗号将它们隔开，如下所示：

```
"action":["dcdb:action1","dcdb:action2"]
```

2. 您也可以使用通配符指定多项操作。例如，您可以指定名字以单词" Describe "开头的所有操作，如下所示：

```
"action":["dcdb:Describe*"]
```

3. 如果您要指定云数据库中所有操作，请使用 * 通配符，如下所示：

```
"action"：["dcdb:*"]
```

云数据库的资源

每个 CAM 策略语句都有适用于自己的资源。 资源的一般形式如下：

```
qcs:project_id:service_type:region:account:resource
```

- **project_id**：描述项目信息，仅为了兼容 CAM 早期逻辑，无需填写。
- **service_type**：产品简称，如 dcdb。
- **region**：地域信息，如 ap-guangzhou。请参考地域相关信息。
- **account**：资源拥有者的主帐号信息，如 uin/653339763。
- **resource**：各产品的具体资源详情，如 instance/instance_id1 或者 instance/*。

例如，

1. 您可以使用特定实例（dcdb-k05xdcta）在语句中指定它，如下所示：

```
"resource":[ "qcs::dcdb:ap-guangzhou:uin/653339763:instance/dcdb-k05xdcta"]
```

2. 您还可以使用 * 通配符指定属于特定账户的所有实例，如下所示：

```
"resource":[ "qcs::dcdb:ap-guangzhou:uin/653339763:instance/*"]
```

3. 您要指定所有资源，或者如果特定 API 操作不支持资源级权限，请在 Resource 元素中使用 * 通配符，如下所示：

```
"resource": ["*"]
```

4. 如果您想要在一条指令中同时指定多个资源，请使用英文逗号将它们隔开，如下所示为指定两个资源的例子：

```
"resource":["resource1","resource2"]
```

下表描述了云数据库能够使用的资源和对应的资源描述方法。 在下表中，$ 为前缀的单词均为代称。

- 其中，project 指代的是项目 ID。
- 其中，region 指代的是地域。
- 其中，account 指代的是账户 ID。

资源	授权策略中的资源描述方法
实例	`qcs::dcdb:$region:$account:instance/$instanceId`

支持的资源级权限

最近更新时间: 2025-02-18 16:02:00

由于历史原因，TDSQL MySQL 版（曾用名 DCDB）在访问管理的接口关键词为 dcdb。

资源级权限指的是能够指定允许用户对哪些资源具有执行操作的能力。云数据库部分支持资源级权限，这意味着对于某些云数据库操作，您可以控制何时允许用户执行操作（基于必须满足的条件）或是允许用户使用的特定资源。下表将向您介绍云数据库可授权的资源类型。

CAM 中可授权的资源类型：

资源类型	授权策略中的资源描述方法
云数据库实例相关	`qcs::dcdb:$region:$account:instance/*` `qcs::dcdb:$region:$account:instance/$instanceId`

下表将介绍当前支持资源级权限的云数据库 API 操作，以及每个操作支持的资源和条件密钥。指定资源路径的时候，您可以在路径中使用 * 通配符。

> **说明：**
>
> 表中未列出的云数据库 API 操作，即表示该云数据库 API 操作不支持资源级权限。针对不支持资源级权限的云数据库 API 操作，您仍可以向用户授予使用该操作的权限，但策略语句的资源元素必须指定为 *。

下列操作可支持资源级权限

API名	描述
ActiveHourDCDBInstance	恢复后付费实例
AssociateSecurityGroups	绑定安全组
CheckIpStatus	检查私有网络 IP是否可用
CloneAccount	克隆实例账户
CopyAccountPrivileges	复制账号权限
CreateAccount	创建账号
CreateDCDBInstance	创建预付费实例
CreateHourDCDBInstance	创建后付费实例
CreateTmpDCDBInstance	回档实例
DeleteAccount	删除账号
DeleteTmpInstance	删除临时实例
DescribeAccountPrivileges	查询账号权限
DescribeAccounts	查询账号列表
DescribeBatchDCDBRenewalPrice	实例批量续费询价
DescribeDatabaseObjects	查询数据库对象列表

API名	描述
DescribeDatabases	查询数据库列表
DescribeDatabaseTable	查询数据库表信息
DescribeDBDetailMetrics	查询云数据库详细监控指标
DescribeDBEncryptAttributes	查询实例数据加密状态
DescribeDBLogFiles	获取日志列表
DescribeDBMetrics	查询云数据库监控指标
DescribeDBParameters	查看数据库参数
DescribeDBSecurityGroups	查询实例安全组信息
DescribeDBSlowLogAnalysis	获取慢查询记录详情
DescribeDBSlowLogs	查询慢查询日志列表
DescribeDBSyncMode	查询同步模式
DescribeDBTmpInstances	获取实例回档生成的临时实例
DescribeDCDBBinlogTime	获取实例回档时可选的时间
DescribeDCDBInstanceDetail	获取实例详情
DescribeDCDBInstances	查询实例列表
DescribeDCDBPrice	新购分布式数据库实例询价
DescribeDCDBRenewalPrice	续费实例询价
DescribeDCDBSaleInfo	查询分布式数据库可售卖地域和可用区信息
DescribeDCDBShards	查询分片信息
DescribeDCDBUpgradePrice	查询升级分布式数据库实例价格
DescribeFlow	查询流程状态
DescribeLatestCloudDBAReport	获取最新的性能检测报告
DescribeLogFileRetentionPeriod	查看备份日志备份天数
DescribeOrders	查询订单信息
DescribeProjectSecurityGroups	查询项目安全组信息
DescribeShardSpec	查询分布式数据库可售卖分片规格
DestroyHourDCDBInstance	销毁按量计费实例
DisassociateSecurityGroups	安全组批量解绑云资源

API名	描述
FlushBinlog	切分 Binlog
GrantAccountPrivileges	设置账号权限
InitDCDBInstances	初始化实例
IsolateHourDCDBInstance	销毁后付费实例
KillSession	杀死指定会话
ModifyAccountDescription	修改数据库账号备注
ModifyAutoRenewFlag	修改自动续费标记
ModifyDBEncryptAttributes	修改实例数据加密属性
ModifyDBInstanceName	修改实例名称
ModifyDBInstanceSecurityGroups	修改云数据库安全组
ModifyDBParameters	修改数据库参数
ModifyDBSyncMode	修改同步模式
ModifyInstanceNetwork	修改实例所属网络
ModifyInstanceRemark	修改实例备注
ModifyInstanceVip	修改实例 Vip
ModifyInstanceVport	修改实例 VPORT
ModifyLogFileRetentionPeriod	修改备份日志保存天数
RenewDCDBInstance	续费分布式数据库实例
ResetAccountPassword	重置账号密码
RestartDBInstances	重启实例
StartSmartDBA	启动性能检测任务
SwitchRollbackInstance	切换回档实例
UpgradeDCDBInstance	升级分布式数据库
UpgradeHourDCDBInstance	升级后付费实例

控制台示例

最近更新时间: 2025-02-18 16:02:00

云数据库访问管理策略示例

您可以通过使用 CAM 策略让用户拥有在云数据库控制台中查看和使用特定资源的权限。该部分的示例能够使用户使用控制台的特定部分的策略。

由于历史原因，分布式数据库 TDSQL（曾用名 DCDB）在访问管理的接口关键词为 dcdb。

创建自定义策略语法

1. 进入**访问管理**->**策略管理。**
2. 进入策略语法配置页面，单击**新建自定义策略。**
3. 在弹出的对话框，选择**按策略语法创建。**
4. 选择空白模板并单击**下一步。**
5. 填入对应的策略语法。

关联子账号/协作者并验证

创建策略完成后，选择关联用户/组。关联完成后，更换浏览器（或主机），通过使用子账号/协作者验证验证是否正常。如果策略语法写作无误，您可以观察到：

- 您能正常访问预期目标产品和资源，并拥有预期的全部功能。
- 访问其他未授权产品或资源时提示“您没有权限执行此操作”。
- 为避免多个策略语法影响，建议一次只让子账号关联一个策略。
- 修改某账号访问控制权限后，预估会有1分钟以内的延迟。

附录：常用的策略语法

放通云数据库的全部实例全部功能策略

如果您想让用户拥有创建和管理云数据库实例的权限，您可以对该用户使用名称为 QcloudDCDBFullAccess 的策略。

策略语法如下：

```
{
"version": "2.0",
"statement": [
{
"action": [
"dcdb:*"
],
"resource": "*",
"effect": "allow"
}
]
}
```

云数据库全部实例仅查询功能策略

如果您只想让用户拥有查询云数据库实例的权限，但是不具有创建、删除和修改的权限，您可以对该用户使用名称为 QcloudDCDBInnerReadOnlyAccess 的策略。

策略语法如下：

```
{
"version": "2.0",
"statement": [
{
"action": [
"dcdb:Describe*"
],
"resource": "*",
"effect": "allow"
}
]
}
```

以上策略是通过让用户分别对云数据库中所有以单词" Describe "开头的所有操作进行 CAM 策略授权来达到目的。

授权用户拥有特定地域云数据库的操作权限策略

如果您想要授权用户拥有特定地域的云数据库的操作权限，可将以下策略关联到该用户。以下策略允许用户拥有对广州地域的云数据库机器的操作权限。

```
{
"version": "2.0",
"statement": [
{
"action": "dcdb:*",
"resource": "qcs::dcdb:ap-guangzhou::*",
"effect": "allow"
}
]
}
```

授权用户拥有若干特定地域云数据库的操作权限策略

如果您想要授权用户拥有特定地域的云数据库的操作权限，可将以下策略关联到该用户。以下策略允许用户拥有对广州地域的云数据库机器的操作权限。

```
{
"version": "2.0",
"statement": [
{
"action": "dcdb:*",
"resource": "qcs::dcdb:ap-guangzhou::*","qcs::dcdb:ap-chengdu::*",
"effect": "allow"
}
]
}
```

授权用户拥有特定云数据库的操作权限策略

如果您想要授权用户拥有特定云数据库操作权限，可将以下策略关联到该用户。以下策略允许用户拥有对 ID 为 dcdb-xxx，广州地域的云数据库实例的操作权限：

```
{
"version": "2.0",
"statement": [
{
"action": [
```

```
"dcdb:*"
],
"resource": "qcs::dcdb:ap-chengdu::instance/dcdb-fwr62n3i",
"effect": "allow"
}
]
}
```

授权用户拥有若干云数据库的操作权限策略

如果您想要授权用户拥有批量云数据库操作权限，可将以下策略关联到该用户。以下策略允许用户拥有对 ID 为 dcdb-xxx、dcdb-yyy，广州地域的云数据库实例的操作权限和对 ID 为 dcdb-zzz，北京地域的云数据库实例的操作权限。

```
{
"version": "2.0",
"statement": [
{
"action": "dcdb:*",
"resource": ["qcs::dcdb:ap-guangzhou::instance/dcdb-xxx", "qcs::dcdb:ap-guangzhou::instance/dcdb-yyy", "qcs::dcdb:ap-beijing::instance/dcdb-zzz"],
"effect": "allow"
}
]
}
```

授权用户拥有若干云数据库的若干操作权限策略

如果您想要授权用户拥有批量云数据库操作权限，可将以下策略关联到该用户。以下策略允许用户拥有对 ID 为 dcdb-xxx、dcdb-yyy，广州地域的云数据库实例的操作权限和对 ID 为 dcdb-zzz，北京地域的云数据库实例的操作权限。

```
{
"version": "2.0",
"statement": [
{
"action": "dcdb:Describe*","dcdb:Create*",
"resource": ["qcs::dcdb:ap-guangzhou::instance/dcdb-xxx", "qcs::dcdb:ap-guangzhou::instance/dcdb-yyy", "qcs::dcdb:ap-beijing::instance/dcdb-zzz"],
"effect": "allow"
}
]
}
```

> **说明：**
>
> 当前全部支持 API 接口详见API接口文档。

拒绝用户拥有云数据库的创建账号权限

如果您想要拒绝某用户拥有云数据库的创建账号权限，即配置 `&quot;effect&quot;: &quot;deny&quot;` 。

```
{
"version": "2.0",
"statement": [
{
"action": "dcdb:CreateAccount",
"resource": "*",
"effect": "deny"
}
```

```
]
}
```

其他自定义策略

如果您觉得预设策略不能满足您所想要的要求，您也可以创建自定义策略。自定义的策略语法如下：

```
{
"version": "2.0",
"statement": [
{
"action": [
"Action"
],
"resource": "Resource",
"effect": "Effect"
}
]
}
```

- Action 中换成您要进行允许或拒绝的操作。
- Resource 中换成您要授权的具体资源。
- Effect 中换成允许或者拒绝。

最佳实践
从单机实例导入到分布式实例

最近更新时间: 2025-02-18 16:02:00

由于分布式数据库的分布式架构对用户透明，一般情况下，只需要预先建好表结构。可以使用 mysqldump、或其他 Navicat、SQLyog 等 MySQL 客户端进行迁移。迁移步骤如下：

1. 准备导入导出环境
2. 导出源表的表结构和数据
3. 修改建表语句并在目的表中创建表结构
4. 导入数据

准备导入导出环境

准备迁移数据前，您需要准备好如下环境：

- 云服务器
- 建议配置 CPU 2核，内存8GB，磁盘500GB以上（取决于数据量大小）
- Linux
- 安装 MySQL 客户端
- 如果您迁移的数据量较小（< 10GB），也可以通过外网（互联网）直接导入，无需准备
- 分布式数据库 TDSQL
- 根据预期选择大小，并根据源库字符集，表名大小写，innodb_page_size 大小进行初始化
- 创建帐号，该帐号建议开启全局所有权限
- 必要时开启外网 IP

导出源表的表结构和数据

演示环境

- 操作库：caccts
- 操作表：t_acct_water_0
- 源库：单实例 MySQL
- 目标库：TDSQL for Percona、MariaDB

在源库导出表结构

通过命令 `mysqldump -u username -p password -d -S dbname tablename &gt; tablename.sql` 导出表结构。

```
//命令实例
mysqldump -utest -ptest1234 -d -S /data/4003/prod/mysql.scok caccts t_acct_water_0 > table.sql
```

在源库导出表数据

通过命令 `mysqldump -c -u username -p password -d dbname tablename &gt; tablename.sql` 导出表数据。

```
//命令实例
mysqldump -c -t -utest -ptest1234 -S /data/4003/prod/mysql.scok caccts t_acct_water_0 > data.sql
```

> 导出数据必须通过 mysqldump 工具导出，并且加上 -c 参数，因为这样导出的数据行都带有列名字段，不带列名字段的 sql 会被 TDSQL for Percona、MariaDB 拒绝掉。-t 参数的意义是不导出表结构，只导出表数据。

上传文件至云服务器某目录

上传前，您需开启 CVM 外网访问地址，并参见 Linux 系统通过 SCP 上传文件到 Linux 云服务器 上传文件，您至少需要上传刚刚导出的：

- 表结构 sql：table.sql
- 数据 sql：data.sql

修改建表语句并在目的表中创建表结构

打开刚导出的表结构文件 table.sql，参考如下格式语句添加主键和 shardkey，并另存为 tablenew.sql。

```
CREATE TABLE (
列名称1 数据类型,
列名称2 数据类型,
列名称3 数据类型,
.... ,
PRIMARY KEY('列名称n') )
ENGINE=INNODB DEFAULT CHARSET=xxxx
shardkey=keyname
```

> 必须要设置主键，必须指定 shardkey，必须注意表名大小问题，建议删除多余注释，否则建表可能不成功。

导入数据

连接 TDSQL for Percona、MariaDB 实例

在 CVM 上使用 `mysql -u username -p password -h IP -P port` 登录 MySQL 服务器，然后使用 `use dbname` 进入数据库。

> 导入数据前需要先创建库。

导入表结构

使用刚刚上传的文件，用 `source` 命令导入数据。

1. 先导入表结构： `source /文件路径/tablenew.sql；`
2. 再导入数据： `source /文件路径/data.sql；`
3. 校验导入情况： `select count(*) from tablename;`

> 需先导入建表语句，再导入数据。也可以通过 mysql 的 source 命令直接导入 sql。

其他方案

整体来说，只要能够在导入数据前，在目的表先创建对应的表结构（需指定 shardkey），就可以比较顺利的导入数据。

Mydumper/MyLoader工具使用说明
版本

最近更新时间: 2025-02-18 16:02:00

mydumper 版本号

```
[root@db1 mydumper]# ./mydumper --version
mydumper-2.0.20-3-R020D003
base on community mydumper 0.11.6, built against MySQL 8.0.23
```

myloader 版本号

```
[root@controller mydumper]# ./myloader --version
myloader-2.0.20-3-R020D003
base on community myloader 0.11.6, built against MySQL 8.0.23
```

导入导出命令

最近更新时间: 2025-02-18 16:02:00

INSERT 语句格式

只导出表结构和数据库创建语句：
./mydumper --host=xxx --port=xxx --user=xxx --password=xxx --less-locking --chunk-filesize=1024 --ignore-sysdb=1 --threads=4 --outputdir=/data/backup/schema -B xxx -T xxx --no-data
举例：
./mydumper --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --less-locking --ignore-sysdb=1 --chunk-filesize=1024 --threads=4 --outputdir=/data/backup/schema -B test -T emp --no-data
说明：以上命令将test数据库上的emp表导出到/data/backup/schema目录下，导出文件test-schema-create.sql包含创建test数据库的DDL语句；导出文件test.emp-schema.sql包含创建emp表结构的DDL语句，不包含构造表数据的insert语句

只导出表数据
./mydumper --host=xxx --port=xxx --user=xxx --password=xxx --less-locking --chunk-filesize=1024 --ignore-sysdb=1 --threads=4 --outputdir=/data/backup/data --complete-insert -B xxx -T xxx --no-schemas
举例：
./mydumper --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --less-locking --ignore-sysdb=1 --chunk-filesize=1024 --threads=4 --outputdir=/data/backup/data --complete-insert -B test -T emp --no-schemas
说明：以上命令将test数据库上的emp表数据导出到/data/backup/data目录下，导出文件test.emp.sql包含构造emp表数据的insert语句

同时导出表数据和表结构创建语句：
./mydumper --host=xxx --port=xxx --user=xxx --password=xxx --less-locking --chunk-filesize=1024 --ignore-sysdb=1 --threads=4 --outputdir=/data/backup/data_schema --complete-insert -B xxx -T xxx
举例：
./mydumper --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --less-locking --ignore-sysdb=1 --chunk-filesize=1024 --threads=4 --outputdir=/data/backup/data_schema --complete-insert -B test -T emp
说明：以上命令同时导出表数据和表结构创建语句，命令中避免带入--no-schemas、--no-data选项

只导入表结构
./myloader --host=xxx --port=xxx --user=xxx --password=xxx --directory=/data/backup/schema --threads=4 --enable-binlog
举例：
./myloader --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --directory=/data/backup/schema --threads=4 --enable-binlog

只导入表数据
./myloader --host=xxx --port=xxx --user=xxx --password=xxx --directory=/data/backup/data --threads=4 --enable-binlog
举例：
./myloader --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --directory=/data/backup/data --threads=4 --enable-binlog

CSV 格式

###导出CSV格式的数据
./mydumper --host=172.27.25.11 --port=15002 --user=test1 --password=test1 -B test --tables-list customer --load-data --field

```
s-terminated-by="," --fields-enclosed-by='"' --lines-starting-by="---" --lines-terminated-by="\n" --fields-escaped-by='\' -o /d
ata/backup/data/
```

> **说明：**
>
> CSV 格式的数据可以使用 loaddata工具 导入，详情参考《TDSQL MYSQL版 V10.3.20.x.x-Load_data工具使用说明》。

备份生成的文件

最近更新时间: 2025-02-18 16:02:00

mydumper 备份的结果就是在指定的备份目录下生成一系列文件，下面对常用生成的文件说明：

- metadata：元数据信息，包括备份的开始和结束时间等
- dump_progress.info：mydumper 数据导出进度
- load_progress.info：myloader 数据导入进度
- dbname-schema-create.sql：建库语句（CREATE DATABASE dbname）
- dbname.tablename[.n].sql：插入数据语句（INSERT INTO tablename VALUES），由于 mydumper 支持对表分片，对于每个分片都会生成一个名称类似 dbname.tablename.00001.sql 的文件
- dbname.tablename-schema.sql：建表语句（CREATE TABLE tablename）
- dbname.viewname-schema-view.sql：建视图语句（CREATE VIEW viewname）
- dbname.viewname-schema.sql：建表语句（CREATE TABLE viewname）
- dbname.tablename-schema-triggers.sql：建触发器语句（CREATE TRIGGER）
- dbname-schema-post.sql：建函数、过程和事件语句（CREATE FUNCTION，CREATE PROCEDURE，CREATE EVENT）

参数说明
mydumper

最近更新时间: 2025-02-18 16:02:00

参数选项

- -B, --database 导出单库
- -T, --tables-list 以逗号分隔的多个表名
- -O,--omit-from-file 包含要跳过的 database.table 条目列表的文件，每行一个（在应用正则表达式选项之前跳过）
- -o, --outputdir 导出文件存放目录
- -s, --statement-size 每条INSERT语句的大小,单位字节,默认 1000000
- -r, --rows 尝试分割表,每个块的行数由该参数指定，该参数会屏蔽--chunk-filesize
- -F, --chunk-filesize 尝试分割表,每个表的大小由该参数指定，单位MB，该参数不能和--load-data一起使用
- --max-rows 限制表中每个块的行数，默认1000000
- -c, --compress 将备份出来的数据文件，经压缩后存放
- -e, --build-empty-files 空表也生成dump文件，此选项在V1.0D002版本之后废弃，统一用 ignore-emptytb
- --ignore-emptytb 忽略空表不导出，此选项默认不开启，如需开启，设置 --ignore-emptytb=1
- -x, --regex 'db.table' 正则匹配
- -i, --ignore-engines 忽略以逗号分隔的多种存储引擎的数据表
- -N, --insert-ignore 导出sql采用INSERT IGNORE
- -m, --no-schemas 不导出表结构文件
- -M, --table-checksums 转储表校验和与数据
- -d, --no-data 不导出数据文件
- --order-by-primary 如果不存在主键，则按主键或唯一键对数据进行排序
- -G, --triggers 导出触发器
- -E, --events 导出事件
- -R, --routines 导出存储过程和函数
- -W, --no-views 不导出视图
- -k, --no-locks 不执行任何临时共享读锁. 警告：该参数会导致备份数据不一致
- --no-backup-locks 不使用percona 特有的backup locks，还是使用传统的flush table with read lock
- --less-locking 最小化在innodb引擎表上的锁持有时间
- --long-query-retries 重试检查长查询，默认0（不重试）
- --long-query-retry-interval 重试长查询检查前的等待时间，以秒为单位，默认 60
- -l, --long-query-guard 设置长查询判定阈值, 默认60s，在执行flush tables with read lock前，如果开启了kill-long-queries，为了避免加全局读锁失败，会将当前执行时间超过60秒的sql强行结束
- -K, --kill-long-queries 在执行flush tables with read lock前,kill长查询(默认行为是遇到长查询报错退出)
- -D, --daemon 以守护进程模式运行
- -X, --snapshot-count 快照数量，默认 2
- -I, --snapshot-interval 每次dump的时间间隔 (单位为分钟), 需要 --daemon 参数, 默认 60
- -L, --logfile log文件名，默认使用stdout
- --tz-utc TIMESTAMP类型转换时区到UTC+0, 默认开启, 设置--skip-tz-utc可关闭.
- --skip-tz-utc 不转换TIMESTAMP时区
- --use-savepoints 采用savepoint以减少元数据锁使用, 需要 SUPER 权限
- --success-on-1146 针对表不存在错误，采用warning而不是error

- --lock-all-tables 使用 LOCK TABLE for all 替代 FTWRL
- -U, --updated-since 使用 Update_time 仅导出过去U天有更新的表
- --trx-consistency-only 将所有类型的表都当做innodb表备份
- --complete-insert 构造insert语句带字段
- --split-partitions 将分区转储到单独的文件中。 此选项覆盖分区表的 --rows 选项。
- --set-names 设置名称，使用有风险，默认二进制
- -h, --host 主机名
- -u, --user 用户名
- -p, --password 密码
- -a, --ask-password 向用户索要密码
- -P, --port TCP端口号
- -S, --socket 连接采用socket文件
- -t, --threads 备份线程数，默认4
- -C, --compress-protocol 压缩协议
- -V, --version 打印程序版本并退出
- -v, --verbose 额外输出信息, 0 = silent, 1 = errors, 2 = warnings, 3 = info, default 2
- --load-data(原csv-output) 空参或设置为1时，输出csv格式，并不再输出sql格式，默认不开启，该参数不能和--trunk_filesize一起使用
- --fields-escaped-by (原csv-escaped) 控制字符转义字符，仅支持单字符,默认""，设置为空则不进行转义 ,默认为""
- --fields-enclosed-by (原csv-fields-enclosed) 字段包围符，仅支持单字符,默认为""
- --fields-terminated-by (原csv-fields-terminated) 字段结束符,默认为"\t"
- --lines-starting-by (原csv-lines-starting) 行起始符默认为""
- --lines-terminated-by (原csv-lines-terminated) 行结束符，默认为"\n"
- --statement-terminated-by 该参数很少会被使用到
- --sync-wait 在 SESSION 级别设置的 WSREP_SYNC_WAIT 值
- --hdfs-path 指定hdfs路径
- --ignore-sysdb 忽略mysql,sysdb,sys库
- --cos-bucket Bucket name in tencent cos storage service, 外部客户无需关注此选项
- --cos-secretid Secret key in tencent cos storage service, 外部客户无需关注此选项
- --cos-secretkey Secret ID in tencent cos storage service, 外部客户无需关注此选项
- --cos-region Region name in tencent cos storage service, 外部客户无需关注此选项
- --l5_server_name l5 server name, 外部客户无需关注此选项
- --where 导出查询语句使用where条件,例如:--where="a=1 and b=2"
- --whereCondition 选项where的别名
- --no-check-generated-fields 不会执行与生成字段相关的查询。如果您已生成列，将导致恢复问题
- --defaults-file 指定mysql default file
- --ssl 使用 SSL 连接
- --ssl-mode 与服务器的连接所需的安全状态：DISABLED、PREFERRED、REQUIRED、VERIFY_CA、VERIFY_IDENTITY
- --key 密钥文件的路径名
- --cert 证书文件的路径名
- --ca 证书颁发机构文件的路径名
- --capath 包含 PEM 格式的受信任 SSL CA 证书的目录的路径名
- --cipher 用于 SSL 加密的允许密码列表
- --tls-version 服务器允许哪些协议用于加密连接
- --stream 写入文件后，它将通过 STDOUT 流式传输
- --no-delete 流式操作完成后不会删除文件

举例

测试表：

```
CREATE TABLE `customer` (
`cust_id` int NOT NULL,
`name` varchar(200) DEFAULT NULL,
`job_id` int DEFAULT NULL,
`job_name` varchar(300) DEFAULT NULL,
PRIMARY KEY (`cust_id`),KEY `j_idx` (`name`) USING BTREE) ENGINE=InnoDB;

insert into customer(cust_id,name,job_id,job_name) values(100,'A',20,'C');
insert into customer(cust_id,name,job_id,job_name) values(200,'B',30,'NET');
insert into customer(cust_id,name,job_id,job_name) values(300,'C',40,'JAVA');
insert into customer(cust_id,name,job_id,job_name) values(400,'D',50,'PM');
```

文件导出到 hdfs下的目录：

```
su - tdsql
./mydumper --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --less-locking --ignore-sysdb=1 --threads=4 --outp
utdir=/data/backup/data -B test -T customer --no-schemas --hdfs-path /recover
```

导出文件按行数切分：

```
./mydumper --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --less-locking --ignore-sysdb=1 --ignore-emptytb=
1 --rows=50 --threads=4 --outputdir=/data/backup/data --complete-insert -B test -T customer --no-schemas
```

正则表达式，不备份包含 mysql、xa、query_rewrite 等关键字开头的数据库：

```
./mydumper --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --less-locking --threads=1 --ignore-sysdb=1 --outp
utdir=/data/backup/data --regex='^(?!(mysql|xa|query_rewrite))' --no-schemas
```

使用压缩方式导出数据：

```
./mydumper --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --less-locking --ignore-sysdb=1 --ignore-emptytb=
1 --threads=1 --outputdir=/data/backup/data --complete-insert -B test -T customer --no-schemas --compress=1
```

使用CSV格式导出数据：

```
./mydumper --host=172.27.25.11 --port=15002 --user=test1 --password=test1 -B test --tables-list customer --load-data --field
s-terminated-by="," --fields-enclosed-by='"' --lines-starting-by="---" --lines-terminated-by="\n" --fields-escaped-by='\' -o /d
ata/backup/data2
```

注意事项

- --load-data 和 chunk_filesize 不能同时指定。
- --database、--tables-list、--db-list 不能同时指定。
- timestamp 类型默认导出会把时区转换到 UTC+0，可以通过--skip-tz-utc 改变这一行为，即不进行时区转换
- 对于 big5、cp932、gbk、sjis、gb18031编码类型的表，由于编码字符集中含有和控制字符相等的字节，可能会出现导出后无法用 load data 导入的情况

- --fields-escaped-by 设置转义字符，该转义的目的是为了避免字段内字符和格式控制字符冲突。如果设置为空，则不进行任何转义，如果数据和格式控制字符冲突，则可能会出现导出后无法用 load data 导入的情况，因此最好保持默认或者设置非空转移字符

myloader

最近更新时间: 2025-02-18 16:02:00

参数选项

- -?, --help：显示帮助选项
- -d, --directory：指定要加载的数据目录
- -q, --queries-per-transaction：指定每个事务的查询数阀值，默认为1000
- -o, --overwrite-tables：加载数据时，如果数据库中已存在的表，则先执行 drop table 语句，注意：需要备份表结构(即，mydumper 备份时不得使用 -m, --no-schemas 选项)
- -B, --database：需要恢复到哪个数据库，该数据库为目标数据名称
- -s, --source-db：需要恢复哪个数据库，该数据库为源数据库名称
- -e, --enable-binlog：恢复数据时是否记录到 binlog，默认不开启，使用此参数开启记录到 binlog
- -h, --host：定恢复实例的 IP 地址
- -u, --user：指定恢复实例的数据库帐号名
- -p, --password：指定恢复实例的数据库帐号密码
- -P, --port：指定恢复实例的数据库 TCP/IP 端口
- -S, --socket：指定恢复实例的数据库 socket 文件
- -t, --threads：载入数据时使用的线程数，默认为4
- -C, --compress-protocol：使用压缩协议连接恢复实例，减少传输数据量
- -V, --version：显示版本号
- -v, --verbose：输出的精度模式, 0 = silent, 1 = errors, 2 = warnings, 3 = info，默认为2
- --defaults-file：指定默认的 my.cnf 配置文件

举例

导入时指定元数据库和目标数据库：

```
./myloader --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --directory=/data/backup/data --threads=4 --enable-binlog --source-db=test --database=newdb
```

导入时覆盖掉原表：

```
./myloader --host=10.0.1.9 --port=15002 --user=test1 --password=test1 --directory=/data/backup/data --threads=4 --enable-binlog --overwrite-tables=1
```

Load_data工具使用说明

工具简介

最近更新时间: 2025-02-18 16:02:00

版本号

```
[root@db1 load_data]# ./load_data --version
version: "load_data-2.0.21-3-R040D006"
```

简介

Load_data是TDSQL团队开发的兼容开源版本的一套导入导出工具。

使用Load_data工具请注意其依赖以下两个功能库。

- ubuntu系列： `apt-get install expat`
- centos系列： `yum install expat expat-devel -y`

功能介绍

最近更新时间: 2025-02-18 16:02:00

- 支持单表（Noshard表），广播表（global表），分表（groupshard表），二级分区表（subshard表）以及range/list全局分区表的导入。
- 支持进度条显示，包括实时导入记录数，总进度百分比，实时导入速率，耗时等。
- 支持命令行导入（类似 `--ip=127.0.0.1` ）和文件参数导入(配置文件 `load.ini`) 。
- 支持自定义导入线程，分割块(chunk_size)大小等参数。
- 支持导入部分字段，使用 `--fields ="(id,k,@jump,pad)"` 参数跳过第三列数据只导入3个字段，或者 `--fields ="(id,k)"` 导入两个字段。
- 支持参数设定日志等级，当日志等级设置为debug时（ `--log_level=2` ）可以查看详细的报错信息，定位到具体的行。报错或者报警告的子文本默认保存在 `./tmp_error` 或者 `./tmp_warning` 目录下。
  - 可以通过设置参数 `--tmp_file_path` , `--log_file_path` 控制临时文件和日志存储目录，例如设置了一下参数后，如果存在历史文件，就可以在 `.../loaddata/tmp/` 目录下，发现 `tmp_error` 或者 `tmp_warning` 目录； `.../loaddata/log/` 目录下发现日志。

```
# ./load_data --ip=127.0.0.1 --password="test" --user=test --port=15006 --file=/data/file.txt --log_level=2 --log_file_path="./log" --tmp_file_path="./tmp" --field_enclosed="\""
```

- 支持后台运行，通过设置 `--is_daemon=true` 开启后台模式，开启后可通过信号量(`kill -s SIGUSR1 pid`)的方式获得实时的导入进度。同时log日志每隔5min打印进度。
- 支持执行前置sql,通过参数 `--prefix_sqls` 能在导入前执行sql，对于设置 `net_read` 或者 `write_timeout` 等参数也可以在此后追加,以分号隔开每个sql。目前默认运行以下指令：

```
"SET unique_checks = 0; SET foreign_key_checks = 0;SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED"
```

- 支持文件夹导入
- 支持位置参数
- 支持自动重试
- 支持断点续传
- 支持设定出错时是退出线程还是继续执行其他range导入
- 支持参数开启文件列数与字段数检查功能
- 支持密码加密，例如运行 `./load_data --password=test --encrypt_pwd` 会使用默认密钥将密码test加密，返回加密后的密码密文。同时支持指定加密密钥 `encrypt_secret_key` 。
- 支持使用加密密码进行导入，例如导入时password设置为密文，则指定 `use_encrypt_pwd=1` 后可以顺利导入。同时支持指定解密密钥 `encrypt_secret_key` 。

执行条件

最近更新时间: 2025-02-18 16:02:00

1. 执行load_data的用户需要有bin目录下的创建目录、创建文件的权限。
2. TDSQL实例需要做如下设置（导入结束后需手动恢复）：

- 调整复制模式为异步复制
- 开启Set免切设置
- 尽量调大数据库超时参数net_read_timeout/net_write_timeout/innodb_lock_wait_timeout
- 尽量调大数据库binlog拦截参数binlog_write_threshold
- 数据库关闭双1参数sync_binlog/innodb_flush_log_at_trx_commit
- 数据库开启LOCAL方式导入数据参数local_infile
- 数据库磁盘空间检查（预计总量占用2～3倍CSV文件空间）
- MySQL 8.0占用内存过多，需要预留足够内存

导入命令

最近更新时间: 2025-02-18 16:02:00

导入指令有3种：命令行导入，位置参数导入和配置文件导入（执行./load_data 可以看到help信息。）

- 命令行导入：

```
./load_data --ip=127.0.0.1 --port=15026 --user=test --password=test --db_table=loaddata.sbtest_shard --file=sbtest_4l.txt --fie
ld_enclosed="\"" --field_terminated="," --log_level=2
```

- 位置参数导入（支持mode0/mode1/mode2/mode3/mode4）

```
./load_data mode0/mode1 proxy_host proxy_port user password db_table shardkey_index file field_terminate filed_enclosed [sp
lit_size] [escaped by]
example:
example:./load_data mode1 10.10.10.10 3336 test test123 shard.table 1 '/tmp/datafile' ' ' ''
mode0: 只拆分数据，不进行数据导入。
mode1：用insert ignore导入的方式，并且skip_error是false，遇见错误直接退出。
mode2：用insert ignore导入的方式，并且skip_error是true，遇见错误不停止。
mode3：用replace 导入方式，并且skip_error是false，遇见错误直接退出。
mode4：用replace 导入方式，并且skip_error是true，遇见错误不停止。
```

- 配置文件导入（其中load.ini格式参考配置文件load.ini）：

```
./load_data --config=load.ini
```

配置文件load.ini格式如下：

```
[Server]
ip=127.0.0.1
port=15006
user=test
password=test
db_table=test.test
file=test.txt
thread_num=2
field_terminated="," (字段分隔符为双引号)
field_enclosed="""" (字段括起符为双引号)
log_level=2
```

配置参数说明

最近更新时间: 2025-02-18 16:02:00

常用参数

- --help 说明：获取帮助信息
- --version 说明：获取版本信息
- --ip=127.0.0.1 说明：proxy的ip地址
- --port=15006 说明：proxy的端口
- --user=test 说明：登陆proxy的用户名
- --password=test 说明：登陆proxy的密码
- --db_table=test.test 说明：指定需要导入的库名和表名，如test.sbtest
- --file=data200M.txt 说明：导入文件的位置，如--file=/data2/load_new/2.txt;当为目录时，表示文件夹导入。
- --field_terminated=" " 说明：字段间隔符，如空格(" "),逗号(","),制表符("\t"),感叹号("|!")等

> **说明：**
>
> 部分shell工具无法识别双引号带感叹号("|!")的场景，此时需要在执行loaddata前，关闭histexpand设置（该设置执行历史替换时打开!和!!扩展，默认为开启）
>
> > `# set +H #执行前关闭histexpand` `# echo $- himBH` `# set -o histexpand off` `#./load_data #执行loaddata命令` `# set -H #执行后复原histexpand` `# set -o #查看当前设置情况allexport ` `histexpand on`

- --field_enclosed=" " 说明：字段括起符，如为空(" "),双引号号("""")等，注意：双引号在命令行和配置文件中有区别
- --config=load.ini 说明：配置文件导入模式，配置此参数后，其他直接读取配置文件的参数配置。

其他参数

- --set_length= "a(1:5),b(6:10),c(11:15)" 说明：固定长度字符解析入库时，将该表字段a每一行的第1到5字节导入，字段为b第6到10字节导入，字段为c第11到15字节导入为字段。如果要更换字段顺序，可以改写set_length: b(2:5),a(7:9),c(11:13)
- --remove_space_mode=0 说明：固定长度字符解析入库时，设置去除空格的方式。取值为：-1（去除左边空格），0（默认值，不去除左右空格），1（去除右边空格），2（去除左右两边空格）

> **注意：**
>
> set_length、remove_space_mode 当前版本仅兼容单表和广播表，且需要为utf8编码。
>
> 此处a、b、c为字段名的示例，请以实际为准；

- --tmp_file_path=/文件夹路径 控制临时存储目录。
- --log_file_path=/文件夹路径 控制日志存储目录。
- --thread_num=1 说明：导入线程数，默认值为4

--chunk_size=10 说明：导入块大小（KB），默认值为与文件大小相关的一个分段函数,当设置的chunk_size<=1280k时，其都是以128k运行的。

- --escaped_by="\" 说明：转义字符，默认值为反双斜线
- --lines_terminated="\n" 说明：行间隔符,支持"\n","\r","\r\n"等
- --fields_optionally_enclosed=false 说明：是否选择性括住CHAR、VARCHAR和TEXT等字符型字段
- --prefix_sqls="SET foreign_key_checks" 说明：前置运行sql，默认运行"SET unique_checks = 0", "SET foreign_key_checks = 0","SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED"这3条sql。注意：该参数在命令行和配置文件中有区别，在使用配置文件时，需要去掉引号。
- --replace_duplicates=false 说明：是否开启替换模式，替换已存在的记录
- --fields="(id,k,@jump,pad)" 说明：导入部分字段(id,k,@jump,pad)，该参数在使用配置文件时，需要去掉引号
- --log_level=5 说明：设置日志等级，默认为LOG_INFO，开启debug日志请将等级设置为2
- --is_daemon=false 说明：是否开启后台运行模式，默认关闭
- --retry=1 说明：设置重试模式。-1表示当导入出现错误时一直重连；0表示不重试；1表示重试一段时间，需结合retry_time一起使用
- --retry_time=10 说明：当retry为1时，指定重试时间,单位是min，默认一个10分钟
- --skip_error=0 说明：是否跳过错误，即当发生错误时，是停止导入（0）还是跳过错误（1）
- --Breakpoint=0 说明：是否开启断点重传
- --column_check=0 说明：是否开启字段检查功能，注意此功能与导入部分字段功能冲突，无法同时使用。注意：该参数在单表和广播表导入时无效。
- --client_timeout=1000 说明，mysql api的超时参数，默认是1000s
- --character_set="gbk" 说明：指定导入文件编码格式，该参数在使用配置文件时，需要去掉引号
- --suffix_sqls="SET column2 = @var1/100" 说明：执行导入sql时追加的后缀信息，通常都需要和fields一起使用; 在使用配置文件时，需要去掉引号。
- --use_encrypt_pwd=0 说明：是否使用了加密的password，默认未使用。当use_encrypt_pwd=0时表示password为明文;use_encrypt_pwd=1时表示使用了加密的password。
- --encrypt_secret_key="abcdefgh" 说明:当password为加密密码时，支持指定解密密钥，不输入时使用默认密钥解密。在使用配置文件时，需要去掉引号。注意：密钥长度至少8个字符
- --encrypt_pwd 说明：通过命令行指令获取密码加密后的密文，该参数只在命令行参数时有效
- --generate_auto_inc_data=false 说明：当缺少自增列数据时，是否自动补充数据，默认进行报错处理不补充数据

- --zk_iplist="127.0.0.1:2181" 说明：当需要补充自增列数据导入时，需要设置zk的iplist信息
- --ignore_lines=2 说明: 跳过文件前n行，例如跳过文件前10行则–ignore_lines=10，注意：ignore和断点续传功能冲突，无法同时使用

举例说明

最近更新时间: 2025-02-18 16:02:00

假设有数据库loaddata和表test：

```
CREATE DATABASE LOADDATA;
USE LOADDATA;
CREATE TABLE `test` (
`a` int NOT NULL,
`b` int DEFAULT NULL,
`c` int NOT NULL,
PRIMARY KEY (`a`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1. 已有数据文件test1.txt格式如下：

```
"1","11","111"
"2","22","222"
```

将test.txt导入test表可以使用下面指令开启2个线程导入：

```
./load_data --ip=10.0.1.9 --port=15002 --user=test --password=test --db_table=loaddata.test --file=/data1/test1.txt --field_enc
losed="\"" --field_terminated="," --log_level=2
```

2. 已有数据文件test2.txt格式如下：

```
"3","aaa","11","111"
"4","bbb","22","222"
```

想要跳过test2.txt的第二列数据，将数据导入表test，可以使用以下指令：

```
./load_data --ip=10.0.1.9 --port=15002 --user=test --password=test --db_table=loaddata.test --file=/data1/test2.txt --field_enc
losed="\"" --field_terminated="," --fields="(a,@jump,b,c)" --log_level=2
```

3. 已有数据文件test3.txt格式如下：

"5",,"111"

"6",,"222"

想将第二列的空值导入表中为NULL，可以使用fields参数结合suffix_sqls参数来实现，具体指令如下：

```
./load_data --ip=10.0.1.9 --port=15002 --user=test --password=test --db_table=loaddata.test --file=/data1/test3.txt --field_enc
losed="\"" --field_terminated="," --fields="(a, @b, c)" --suffix_sqls="SET b = NULLIF(@b,'')" --log_level=2
```

4. 获取密码加密后的密文：

```
./load_data --password=test --encrypt_pwd --encrypt_secret_key="abcdefgh"
```

说明：使用密钥"abcdefgh"将密码test加密，如果不指定encrypt_secret_key则使用默认密钥

5.使用mydumper导出CSV格式数据后，再使用loaddata导入：

###导出数据：
./mydumper --host=172.27.25.11 --port=15002 --user=test1 --password=test1 -B loaddata --tables-list test --load-data -o /data/backup/data2

###导入数据：
./load_data --ip=172.27.25.11 --port=15002 --user=test1 --password=test1 --db_table=loaddata.test --file=/data/backup/data2/loaddata.test.00000.dat --log_level=2

如何选择TDSQL实例配置和分片配置

最近更新时间: 2025-02-18 16:02:00

TDSQL 选型概述

TDSQL 由分片（sharding）组成，分片的规格和分片数量决定了 TDSQL 实例的处理能力。理论上来讲：

- TDSQL 实例读写并发性能 = ∑（某规格分片性能 * 某规格分片数量）
- TDSQL 实例事务性能 = ∑（某规格分片事务性能 * 70% * 某规格分片数量）

因此，分片规格越高、分片数量越多，实例的处理能力越强。而分片性能，主要与 CPU / 内存 相关，并以 QPS 为基础衡量指标，我们在分片性能说明章节，给出了大致性能指标。

TDSQL 分片规格的选择

TDSQL 分片规格的选择，主要从三个方面需求来决定：1、性能需求；2、容量需求；3、其他要求。

性能需求：通过预判至少6个月的性能规模和可能增长，您可以确定您分布式实例所需总 CPU / 内存 规模。 **容量需求**：通过预判至少1年的容量规模和可能增长，您可以确定您分布式实例所需总 磁盘 规模。 **其他要求**：我们建议一个分片**至少存储5000W行数据**，并考虑到业务中所需的广播表、单表，和节点内 join 等业务需求。

> 建议您先确保让单个分片配置较大，而分片数量较少。

综合上述来看，我们预估您可能有如下几种业务特点，推荐策略如下：

- 使用 TDSQL 做功能性测试，且对性能没有特别要求：2个分片，每个分片配置为：**内存/磁盘：2GB/25GB。**
- 业务发展初期，总数据规模较小但增长快的选型：2个分片，每个分片配置为：**内存/磁盘：16GB/200GB。**
- 业务发展稳定，根据业务实际情况选型：4个分片，每个分片配置等于：**当前业务峰值*增长率/4**

TDSQL 分片性能测试

数据库基准性能测试为 sysbench 0.5 工具。

修改说明：对 sysbench 自带的 otlp 脚本做了修改，读写比例修改为 1：1，并通过执行测试命令参数 oltp_point_selects 和 oltp_index_updates 控制读写比例。本文测试用例的均采用4个 select 点，1个 update 点，读写比例保持 4:1。

产品性能与物理服务器配置有关联关系，本性能测试采用（24core，512GB内存，nvme-SSD7200GB磁盘）

vCPU（核）	内存(GB)	存储空间(GB)	数据集(GB)	客户端数	单客户端并发数	QPS	TPS
1	2GB	100GB	46GB	1	128	1880	351
2	4GB	200GB	76GB	1	128	3983	797
2	8GB	200GB	142GB	1	128	6151	1210
4	16GB	400GB	238GB	1	128	10098	2119
4	32GB	700GB	238GB	2	128	20125	3549
8	64GB	1T	378GB	2	128	37956	7002
12	96GB	1.5T	378GB	3	128	51026	10591
16	120GB	2T	378GB	3	128	81050	15013
24	240GB	3T	567GB	4	128	96891	17698

vCPU（核）	内存(GB)	存储空间(GB)	数据集(GB)	客户端数	单客户端并发数	QPS	TPS
48	480GB	6T	567GB	6	128	140256	26599

此处 TPS 为单机 TPS，并非测试的是分布式事务的 TPS。 根据运营策略要求，当前 TDSQL 的（部分）实例都采用闲时超用技术，所以您可能在您的监控中看到 CPU 利用率超过100%的情况。

开发指南
概述

最近更新时间: 2025-02-18 16:02:00

文档说明

本手册涵盖TDSQL连接方式，SQL语句开发编写等内容。目的是指导应用开发。

范围

本手册适用于使用TDSQL分布式实例的应用开发人员、数据库应用设计人员、数据库管理员等。

数据库连接方式
使用客户端

最近更新时间: 2025-02-18 16:02:00

TDSQL通过Proxy接口提供和MySQL兼容的连接方式，用户可以通过IP地址、端口号、用户名以及密码连接TDSQL 系统，连接语句如下：

语法：

mysql -hhost_ip -Pport -uusername -ppassword -c

示例：

mysql -h10.10.10.10 -P3306 -utest12 -ptest123 -c

> **注意：**
>
> 使用MySQL登录命令时，请务必增加-c参数，这样可以使用注释透传功能。

使用中间件

最近更新时间: 2025-02-18 16:02:00

在Tomcat的server.xml中配置数据库连接时，推荐JDBC连接串如下：jdbc:mysql://ip:port/db_name?user=your_username&password=your_password&useLocalSessionStates=true&useUnicode=true&characterEncoding=utf-8&serverTimezone=Asia/Shanghai"

其他参数说明：

参数	含义	缺省值	推荐值
useLocalSessionState	配置驱动程序是否使用autocommit，read_only和transaction isolation的内部值(jdbc端的本地值),避免JDBC driver每次都去检查target database是否是ReadOnly,autocommit	false	true
rewriteBatchedStatements	用于保证jdbc driver可以批量执行SQL，按需配置	false	按需配置，建议true
allowMultiQueries	该配置允许使用";"，在一条语句中分隔多个查询，实现多语句执行	false	按需配置，建议true
useUnicode	是否使用Unicode字符集	false	按需配置，建议设置true
characterEncoding	字符编码格式	无	按需配置，建议设置utf-8
serverTimezone	时区	local	按需配置，建议中国区部署设置为Asia/Shanghai
netTimeoutForStreamingResults	当使用StremResultSet结果集时，建议配置该参数，保证使用数据库的默认超时时间	600	0（即应用端不配置，直接使用数据库服务器超时时间）
useCursorFetch	是否使用cursor来拉取数据。(分布式不支持游标)	false	false
useSSL	与数据库之间连接是否使用加密连接。建议互联网部署应用开启加密连接。开启后由于数据链路加密传输，影响部分性能。非互联网应用按需配置。说明：tdsql网关节点进已进行适配，默认开启usessl后，jdbc参数中无需配置allowPublicKeyRetrieval=true	默认开启，即useSSL=true(或sslMode=PREFERRED)	按需配置（关闭方式：useSSL=false(或sslMode=DISABLED)）

其他连接方式

最近更新时间: 2025-02-18 16:02:00

分布式实例高度兼容 MySQL 的连接协议和语法，支持 SSL 加密，也支持 JDBC、ODBC、PHP、Python 等相关协议和语法，例如：

```
private final String USERNAME = "your_username";
private final String PASSWORD = "your_password";
private final String DRIVER = "com.mysql.jdbc.Driver";
private final String URL = "jdbc:mysql://ip:port?userunicode=true&characterEncoding=utf8mb4";
private Connection connection;
private PreparedStatement pstmt;
private ResultSet resultSet;
```

语言结构

最近更新时间: 2025-02-18 16:02:00

分布式实例支持所有MySQL使用的文字格式，包括如下：

- String Literals
- Numeric Literals
- Date and Time Literals
- Hexadecimal Literals
- Bit-Value Literals
- Boolean Literals
- NULL Values

String Literals格式

String Literals 是一个 bytes 或者 characters 的序列，两端被单引号 ' 或者双引号 " 包围，目前TDSQL不支持ANSI_QUOTES SQL MODE，双引号 " 包围的始终认为是String Literals，而不是Identifier。

不支持 character set introducer格式，即：[charsetname]'string' [COLLATE collation_name]格式。

支持如下转义字符：

- \0: ASCII NUL (X'00') 字符
- \': 单引号
- \": 双引号
- \b: 退格符号
- \n: 换行符
- \r: 回车符
- \t: tab 符（制表符）
- \z: ASCII 26 (Ctrl + Z)
- : 反斜杠 \
- %: %
- _: _

Numeric Literals格式

数值字面值包括Integer类型 、 Decimal 类型、浮点数字面值。

Integer 可以包括“ . ”作为小数点分隔，数字前加字符“ -” 、“ +” 来表示正数或者负数。

精确数值字面值可以表示多种格式，如格式：1, .2, 3.4, -5, -6.78, +9.10。

科学记数法，如格式：1.2E3, 1.2E-3, -1.2E3, -1.2E-3。

Date and Time Literals格式

Date支持如下格式：

'YYYY-MM-DD' or 'YY-MM-DD' 'YYYYMMDD' or 'YYMMDD' YYYYMMDD or YYMMDD

例如：'2012-12-31', '2012/12/31', '2012^12^31', '2012@12@31' '20070523', '070523'

Datetime、Timestamp支持如下格式：

'YYYY-MM-DD HH:MM:SS' or 'YY-MM-DD HH:MM:SS' 'YYYYMMDDHHMMSS' or 'YYMMDDHHMMSS'

YYYYMMDDHHMMSS or YYMMDDHHMMSS

例如：'2012-12-31 11:30:45', '2012^12^31 11+30+45', '2012/12/31 113045', '2012@12@31 11^30^45'，19830905132800

Hexadecimal Literals格式

Hexadecimal Literals支持的格式如下：

X'01AF' X'01af' x'01AF' x'01af' 0x01AF 0x01af

Bit-Value Literals格式

Bit-Value Literals支持的格式如下：

b'01' B'01' 0b01

Boolean Literals格式

常量 True=1 和 False =0，其不区分大小写。

```
mysql> SELECT TRUE, true, FALSE, false;
+------+------+-------+-------+
| TRUE | TRUE | FALSE | FALSE |
+------+------+-------+-------+
| 1 | 1 | 0 | 0 |
+------+------+-------+-------+
1 row in set (0.03 sec)
```

NULL Values

NULL 代表数据为空，不区分大小写，与命令 \N(不区分大小写) 同义。

> **注意：**
>
> NULL 跟 0 的意义不一样，跟空字符串 '' 的意义也不一样。

数据类型
数字类型

最近更新时间: 2025-02-18 16:02:00

分布式实例兼容整型、浮点型和定点型三种数字类型，具体兼容类型如下：

- 整型支持INTEGER、 INT、SMALLINT、TINYINT、 MEDIUMINT、BIGINT七种类型，相关信息详见如下表。

类型	字节数	最小值(有符号/无符号)	最大值(有符号/无符号)
TINYINT	1	-128/70392116334850048	127/255
SMALLINT	2	-32768/70392116334850048	32767/65535
MEDIUMINT	3	-8388608/70392116334850048	8388607/16777215
INT	4	-2147483648/70392116334850048	2147483647/4294967295
BIGINT	8	-9223372036854775808/70392116334850048	9223372036854775807/18446744073709551615

- 浮点型支持FLOAT和DOUBLE，格式支持FLOAT(M,D) 、REAL(M,D) 、 DOUBLE PRECISION(M,D)。
- 定点型支持DECIMAL和NUMERIC，格式DECIMAL(M,D)。

字符类型

最近更新时间: 2025-02-18 16:02:00

TDSQL支持的字符类型：CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、TINYBLOB、TINYTEXT，MEDIUMBLOB、MEDIUMTEXT、LONGBLOB、LONGTEXT、ENUM、SET。

其中 CHAR 和 VARCHAR 最为常用，LOB 和 TEXT 类型不建议使用。

CHAR 和 VARCHAR 类型相似，但存储和检索的方式不同。 它们在最大长度和是否保留尾随空格方面也不同。

CHAR 和 VARCHAR 类型声明的长度指示要存储的最大字符数。 例如，CHAR(30) 最多可容纳 30 个字符。CHAR 列的长度固定为您在创建表时声明的长度。长度可以是 0 到 255 之间的任何值。存储 CHAR 值时，它们会用空格右填充到指定的长度。

VARCHAR 列中的值是可变长度的字符串。长度可以指定为 0 到 65,535 之间的值。

日期类型

最近更新时间: 2025-02-18 16:02:00

TDSQL支持如下时间类型：

类型	日期格式	日期范围
YEAR	YYYY	1901 ~ 2155
TIME	HH:MM:SS	-838:59:59 ~ 838:59:59
DATE	YYYY-MM-DD	1000-01-01 ~ 9999-12-3
DATETIME	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59
TIMESTAMP	YYYY-MM-DD HH:MM:SS	1980-01-01 00:00:01 UTC ~ 2040-01-19 03:14:07 UTC

JSON数据类型

最近更新时间: 2025-02-18 16:02:00

支持存储Json格式的数据类型，以便更加有效的对Json进行处理，同时又能提早检查错误。

语句如下：

> **注意：**
>
> 对Json类型的字段进行排序时，不支持混合类型排序。
>
> 例如，不能将 String 类型和 Int 类型做比较，同类型排序只支持数值类型和 String 类型，其它类型排序暂不处理。

```
mysql> CREATE TABLE t1 (jdoc JSON,a int key);
Query OK, 0 rows affected (0.30 sec)

mysql> INSERT INTO t1 (jdoc,a)VALUES('{"key1": "value1", "key2": "value2"}',1);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t1 (jdoc,a)VALUES('{"key1": "value1", "key2": 2}',2);

mysql> select * from t1;
+--------------------------------------+---+
| jdoc | a |
+--------------------------------------+---+
| {"key1": "value1", "key2": "value2"} | 1 |
| {"key1": "value1", "key2": 2} | 2 |
+--------------------------------------+---+
2 rows in set (0.00 sec)
```

字符集和时区

最近更新时间: 2025-02-18 16:02:00

TDSQL 在后端存储支持 MySQL 的所有字符集和字符序。

具体显示如下：

```
mysql> show character set;
+----------+---------------------------------+---------------------+--------+
| Charset | Description | Default collation | Maxlen |
+----------+---------------------------------+---------------------+--------+
| big5 | Big5 Traditional Chinese | big5_chinese_ci | 2 |
| dec8 | DEC West European | dec8_swedish_ci | 1 |
| cp850 | DOS West European | cp850_general_ci | 1 |
| hp8 | HP West European | hp8_english_ci | 1 |
| koi8r | KOI8-R Relcom Russian | koi8r_general_ci | 1 |
| latin1 | cp1252 West European | latin1_swedish_ci | 1 |
| latin2 | ISO 8859-2 Central European | latin2_general_ci | 1 |
| swe7 | 7bit Swedish | swe7_swedish_ci | 1 |
| ascii | US ASCII | ascii_general_ci | 1 |
| ujis | EUC-JP Japanese | ujis_japanese_ci | 3 |
| sjis | Shift-JIS Japanese | sjis_japanese_ci | 2 |
| hebrew | ISO 8859-8 Hebrew | hebrew_general_ci | 1 |
| tis620 | TIS620 Thai | tis620_thai_ci | 1 |
| euckr | EUC-KR Korean | euckr_korean_ci | 2 |
| koi8u | KOI8-U Ukrainian | koi8u_general_ci | 1 |
| gb2312 | GB2312 Simplified Chinese | gb2312_chinese_ci | 2 |
| greek | ISO 8859-7 Greek | greek_general_ci | 1 |
| cp1250 | Windows Central European | cp1250_general_ci | 1 |
| gbk | GBK Simplified Chinese | gbk_chinese_ci | 2 |
| latin5 | ISO 8859-9 Turkish | latin5_turkish_ci | 1 |
| armscii8 | ARMSCII-8 Armenian | armscii8_general_ci | 1 |
| utf8 | UTF-8 Unicode | utf8_general_ci | 3 |
| ucs2 | UCS-2 Unicode | ucs2_general_ci | 2 |
| cp866 | DOS Russian | cp866_general_ci | 1 |
| keybcs2 | DOS Kamenicky Czech-Slovak | keybcs2_general_ci | 1 |
| macce | Mac Central European | macce_general_ci | 1 |
| macroman | Mac West European | macroman_general_ci | 1 |
| cp852 | DOS Central European | cp852_general_ci | 1 |
| latin7 | ISO 8859-13 Baltic | latin7_general_ci | 1 |
| utf8mb4 | UTF-8 Unicode | utf8mb4_general_ci | 4 |
| cp1251 | Windows Cyrillic | cp1251_general_ci | 1 |
| utf16 | UTF-16 Unicode | utf16_general_ci | 4 |
| utf16le | UTF-16LE Unicode | utf16le_general_ci | 4 |
| cp1256 | Windows Arabic | cp1256_general_ci | 1 |
| cp1257 | Windows Baltic | cp1257_general_ci | 1 |
| utf32 | UTF-32 Unicode | utf32_general_ci | 4 |
| binary | Binary pseudo charset | binary | 1 |
| geostd8 | GEOSTD8 Georgian | geostd8_general_ci | 1 |
| cp932 | SJIS for Windows Japanese | cp932_japanese_ci | 2 |
| eucjpms | UJIS for Windows Japanese | eucjpms_japanese_ci | 3 |
| gb18030 | China National Standard GB18030 | gb18030_chinese_ci | 4 |
+----------+---------------------------------+---------------------+--------+
41 rows in set (0.02 sec)
```

查看当前连接的相关字符集：

```
mysql> show variables like "%char%";
+--------------------------+----------------------------------------------------+
| Variable_name | Value |
+--------------------------+----------------------------------------------------+
| character_set_client | latin1 |
| character_set_connection | latin1 |
| character_set_database | utf8 |
| character_set_filesystem | binary |
| character_set_results | latin1 |
| character_set_server | utf8 |
| character_set_system | utf8 |
| character_sets_dir | /data/tdsql_run/8812/percona-5.7.17/share/charsets/ |
+--------------------------+----------------------------------------------------+
设置当前连接的相关字符集：
mysql> set names utf8;
Query OK, 0 rows affected (0.03 sec)

mysql> show variables like "%char%";
+--------------------------+----------------------------------------------------+
| Variable_name | Value |
+--------------------------+----------------------------------------------------+
| character_set_client | utf8 |
| character_set_connection | utf8 |
| character_set_database | utf8 |
| character_set_filesystem | binary |
| character_set_results | utf8 |
| character_set_server | utf8 |
| character_set_system | utf8 |
| character_sets_dir | /data/tdsql_run/8811/percona-5.7.17/share/charsets/ |
+--------------------------+----------------------------------------------------+
```

> **注意：**
>
> TDSQL 不支持通过命令行设置参数，需要通过赤兔管理台进行设置。

通过设置 time_zone 变量修改时区相关的属性：

```
mysql> show variables like '%time_zone%';
+------------------+--------+
| Variable_name | Value |
+------------------+--------+
| system_time_zone | CST |
| time_zone | SYSTEM |
+------------------+--------+
2 rows in set (0.00 sec)

mysql> create table test.tt (ts timestamp, dt datetime,c int key) shardkey=c;
Query OK, 0 rows affected (0.49 sec)

mysql> insert into test.tt (ts,dt,c)values ('2017-10-01 12:12:12', '2017-10-01 12:12:12',1);
Query OK, 1 row affected (0.09 sec)

mysql> select * from test.tt;
+---------------------+---------------------+------+
| ts | dt | c |
+---------------------+---------------------+------+
| 2017-10-01 12:12:12 | 2017-10-01 12:12:12 | 1 |
+---------------------+---------------------+------+
```

```
1 row in set (0.04 sec)

mysql> set time_zone = '+12:00';
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like '%time_zone%';
+------------------+--------+
| Variable_name | Value |
+------------------+--------+
| system_time_zone | CST |
| time_zone | +12:00 |
+------------------+--------+
2 rows in set (0.00 sec)

mysql> select * from test.tt;
+---------------------+---------------------+------+
| ts | dt | c |
+---------------------+---------------------+------+
| 2017-10-01 16:12:12 | 2017-10-01 12:12:12 | 1 |
+---------------------+---------------------+------+
1 row in set (0.06 sec)
```

函数运算符

最近更新时间: 2025-02-18 16:02:00

分布式实例支持以下7种类型的函数：

- 流程控制函数（Control Flow Functions）
- 字符串函数（String Functions）
- 数字函数（Numeric Functions and Operators）
- 日期时间函数（Date and Time Functions）
- 聚合函数（Aggregate (GROUP BY) Functions）
- 位函数（Bit Functions and Operators）
- 转换函数（Cast Functions and Operators）

以上类型的各函数具体描述如下：

流程控制函数（Control Flow Functions）

函数名	描述
CASE	Case operator
IF()	If/else construct
IFNULL()	Null if/else construct
NULLIF()	Return NULL if expr1 = expr2

字符串函数（String Functions）

函数名	描述
ASCII()	Return numeric value of left-most character
BIN()	Return a string containing binary representation of a number
BIT_LENGTH()	Return length of argument in bits
CHAR()	Return the character for each integer passed
CHAR_LENGTH()	Return number of characters in argument
CHARACTER_LENGTH()	Synonym for CHAR_LENGTH()
CONCAT()	Return concatenated string
CONCAT_WS()	Return concatenate with separator
ELT()	Return string at index number

函数名	描述
EXPORT_SET()	Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string
FIELD()	Return the index (position) of the first argument in the subsequent arguments
FIND_IN_SET()	Return the index position of the first argument within the second argument
FORMAT()	Return a number formatted to specified number of decimal places
FROM_BASE64()	Decode to a base-64 string and return result
HEX()	Return a hexadecimal representation of a decimal or string value
INSERT()	Insert a substring at the specified position up to the specified number of characters
INSTR()	Return the index of the first occurrence of substring
LCASE()	Synonym for LOWER()
LEFT()	Return the leftmost number of characters as specified
LENGTH()	Return the length of a string in bytes
LIKE	Simple pattern matching
LOAD_FILE()	Load the named file
LOCATE()	Return the position of the first occurrence of substring
LOWER()	Return the argument in lowercase
LPAD()	Return the string argument, left-padded with the specified string
LTRIM()	Remove leading spaces
MAKE_SET()	Return a set of comma-separated strings that have the corresponding bit in bits set
MATCH	Perform full-text search
MID()	Return a substring starting from the specified position
NOT LIKE	Negation of simple pattern matching
NOT REGEXP	Negation of REGEXP
OCT()	Return a string containing octal representation of a number
OCTET_LENGTH()	Synonym for LENGTH()
ORD()	Return character code for leftmost character of the argument
POSITION()	Synonym for LOCATE()
QUOTE()	Escape the argument for use in an SQL statement
REGEXP	Pattern matching using regular expressions
REPEAT()	Repeat a string the specified number of times

函数名	描述
REPLACE()	Replace occurrences of a specified string
REVERSE()	Reverse the characters in a string
RIGHT()	Return the specified rightmost number of characters
RLIKE	Synonym for REGEXP
RPAD()	Append string the specified number of times
RTRIM()	Remove trailing spaces
SOUNDEX()	Return a soundex string
SOUNDS LIKE	Compare sounds
SPACE()	Return a string of the specified number of spaces
STRCMP()	Compare two strings
SUBSTR()	Return the substring as specified
SUBSTRING()	Return the substring as specified
SUBSTRING_INDEX()	Return a substring from a string before the specified number of occurrences of the delimiter
TO_BASE64()	Return the argument converted to a base-64 string
TRIM()	Remove leading and trailing spaces
UCASE()	Synonym for UPPER()
UNHEX()	Return a string containing hex representation of a number
UPPER()	Convert to uppercase
WEIGHT_STRING()	Return the weight string for a string

数字函数（Numeric Functions and Operators）

函数名	描述
ABS()	Return the absolute value
ACOS()	Return the arc cosine
ASIN()	Return the arc sine
ATAN()	Return the arc tangent
ATAN2(), ATAN()	Return the arc tangent of the two arguments
CEIL()	Return the smallest integer value not less than the argument

函数名	描述
CEILING()	Return the smallest integer value not less than the argument
CONV()	Convert numbers between different number bases
COS()	Return the cosine
COT()	Return the cotangent
CRC32()	Compute a cyclic redundancy check value
DEGREES()	Convert radians to degrees
DIV	Integer division
/	Division operator
EXP()	Raise to the power of
FLOOR()	Return the largest integer value not greater than the argument
LN()	Return the natural logarithm of the argument
LOG()	Return the natural logarithm of the first argument
LOG10()	Return the base-10 logarithm of the argument
LOG2()	Return the base-2 logarithm of the argument
-	Minus operator
MOD()	Return the remainder
%, MOD	Modulo operator
PI()	Return the value of pi
+	Addition operator
POW()	Return the argument raised to the specified power
POWER()	Return the argument raised to the specified power
RADIANS()	Return argument converted to radians
RAND()	Return a random floating-point value
ROUND()	Round the argument
SIGN()	Return the sign of the argument
SIN()	Return the sine of the argument
SQRT()	Return the square root of the argument
TAN()	Return the tangent of the argument
*	Multiplication operator
TRUNCATE()	Truncate to specified number of decimal places

函数名	描述
-	Change the sign of the argument

日期时间函数（Date and Time Functions）

函数名	描述
ADDDATE()	Add time values (intervals) to a date value
ADDTIME()	Add time
CONVERT_TZ()	Convert from one time zone to another
CURDATE()	Return the current date
CURRENT_DATE(),	CURRENT_DATE Synonyms for CURDATE()
CURRENT_TIME(), CURRENT_TIME	Synonyms for CURTIME()
CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP	Synonyms for NOW()
CURTIME()	Return the current time
DATE()	Extract the date part of a date or datetime expression
DATE_ADD()	Add time values (intervals) to a date value
DATE_FORMAT()	Format date as specified
DATE_SUB()	Subtract a time value (interval) from a date
DATEDIFF()	Subtract two dates
DAY()	Synonym for DAYOFMONTH()
DAYNAME()	Return the name of the weekday
DAYOFMONTH()	Return the day of the month (0-31)
DAYOFWEEK()	Return the weekday index of the argument
DAYOFYEAR()	Return the day of the year (1-366)
EXTRACT()	Extract part of a date
FROM_DAYS()	Convert a day number to a date
FROM_UNIXTIME()	Format Unix timestamp as a date
GET_FORMAT()	Return a date format string
HOUR()	Extract the hour
LAST_DAY	Return the last day of the month for the argument

函数名	描述
LOCALTIME(), LOCALTIME	Synonym for NOW()
LOCALTIMESTAMP, LOCALTIMESTAMP()	Synonym for NOW()
MAKEDATE()	Create a date from the year and day of year
MAKETIME()	Create time from hour, minute, second
MICROSECOND()	Return the microseconds from argument
MINUTE()	Return the minute from the argument
MONTH()	Return the month from the date passed
MONTHNAME()	Return the name of the month
NOW()	Return the current date and time
PERIOD_ADD()	Add a period to a year-month
PERIOD_DIFF()	Return the number of months between periods
QUARTER()	Return the quarter from a date argument
SEC_TO_TIME()	Converts seconds to 'HHimgSS' format
SECOND()	Return the second (0-59)
STR_TO_DATE()	Convert a string to a date
SUBDATE()	Synonym for DATE_SUB() when invoked with three arguments
SUBTIME()	Subtract times
SYSDATE()	Return the time at which the function executes
TIME()	Extract the time portion of the expression passed
TIME_FORMAT()	Format as time
TIME_TO_SEC()	Return the argument converted to seconds
TIMEDIFF()	Subtract time
TIMESTAMP()	With a single argument, this function returns the date or datetime expression; with two arguments, the sum of the arguments
TIMESTAMPADD()	Add an interval to a datetime expression
TIMESTAMPDIFF()	Subtract an interval from a datetime expression
TO_DAYS()	Return the date argument converted to days
TO_SECONDS()	Return the date or datetime argument converted to seconds since Year 0
UNIX_TIMESTAMP()	Return a Unix timestamp
UTC_DATE()	Return the current UTC date

函数名	描述
UTC_TIME()	Return the current UTC time
UTC_TIMESTAMP()	Return the current UTC date and time
WEEK()	Return the week number
WEEKDAY()	Return the weekday index
WEEKOFYEAR()	Return the calendar week of the date (1-53)
YEAR()	Return the year
YEARWEEK()	Return the year and week

聚合函数（Aggregate (GROUP BY) Functions）

函数名	描述
AVG()	Return the average value of the argument
COUNT()	Return a count of the number of rows returned
MAX()	Return the maximum value
MIN()	Return the minimum value
SUM()	Return the sum

位函数（Bit Functions and Operators）

函数名	描述
BIT_COUNT()	Return the number of bits that are set
&	Bitwise AND
~	Bitwise inversion
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

转换函数（Cast Functions and Operators）

函数名	描述
BINARY	Cast a string to a binary string
CAST()	Cast a value as a certain type
CONVERT()	Cast a value as a certain type

SQL语言（分布式）
常用SQL
数据库定义语言（DDL）
CREATE

最近更新时间: 2025-02-18 16:02:00

CREATE DATABASE

本节介绍CREATE DATABASE语法。

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
[create_option] ...

create_option: [DEFAULT] {
CHARACTER SET [=] charset_name
| COLLATE [=] collation_name
}
```

> **注意：**
>
> - CREATE DATABASE 创建具有给定名称的数据库。 要使用此语句，您需要对数据库具有 CREATE 权限。 CREATE SCHEMA 是 CREATE DATABASE 的同义词。
> - 如果数据库存在并且您没有指定 IF NOT EXISTS，则会发生错误。
> - 在具有活动 LOCK TABLES 语句的会话中不允许 CREATE DATABASE。
> - CHARACTER SET 选项指定默认的数据库字符集。 COLLATE 选项指定默认的数据库排序规则。要查看可用的字符集和排序规则，请使用 SHOW CHARACTER SET 和 SHOW COLLATION 语句

示例：

```
create database d2 default charset 'utf8mb4';
```

CREATE TABLE

TDSQL分布式实例支持创建分表、单表和广播表。分表即自动水平拆分的表（Shard表），水平拆分是基于分表键采用一致性 Hash方式，根据计算后的值分配到不同的节点组中的一种技术方案。可以将满足对应条件的行将存储在相同的物理节点组中。这种场景称为组拆分（Groupshard）,可以迅速提高应用层联合查询等语句的处理效率。TDSQL支持LIST、RANGE、HASH三种类型的一级分区，同时支持RANGE、LIST两种格式的二级分区。

创建一级range| list分区表语法

> **注意：**
>
> DB 5.7版本不支持TDSQL_DISTRIBUTED BY range|list的语法。

```
CREATE TABLE [IF NOT EXISTS] tbl_name
[(create_definition)]
[local_table_options]
TDSQL_DISTRIBUTED BY range|list (column_name) [partition_options]

create_definition: {
col_name column_definition
| {INDEX | KEY} [index_name] [index_type] (key_part,...)
[index_option] ...
| [INDEX | KEY] [index_name] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] PRIMARY KEY
[index_type] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
[index_name] [index_type] (key_part,...)
[index_option] ...
}

column_definition: {
data_type [NOT NULL | NULL] [DEFAULT]
[AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
[COLLATE collation_name]
[COLUMN_FORMAT {FIXED | DYNAMIC | DEFAULT}]
[ENGINE_ATTRIBUTE [=] 'string']
| data_type
[UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
}

key_part: {col_name [(length)]} [ASC | DESC]

index_type:
USING {BTREE}

index_option: {
index_type | COMMENT 'string'
}
[local_table_options]
Local_table_option: {AUTO_INCREMENT [=] value
| [DEFAULT] CHARACTER SET [=] charset_name
| [DEFAULT] COLLATE [=] collation_name
| COMMENT [=] 'string'
| ENGINE [=] engine_name
| ROW_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}
| STATS_AUTO_RECALC [=] {DEFAULT | 0 | 1}
| STATS_PERSISTENT [=] {DEFAULT | 0 | 1}
| STATS_SAMPLE_PAGES [=] value)
}
partition_options:
PARTITION BY
| RANGE{(expr)}
| LIST{(expr)}
[(partition_definition [, partition_definition] ...)]

partition_definition:
PARTITION partition_name
[VALUES
```

```
{LESS THAN {(expr | value_list) | MAXVALUE}
|
IN (value_list)}]
[[STORAGE] ENGINE [=] engine_name]
[COMMENT [=] 'string']
```

创建分区表

一级分区

在TDSQL中，分表也叫一级分区表。一级hash分区使用shardkey关键字指定拆分键。

一级HASH分区

支持类型

- – DATE，DATETIME
- – TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT
- – CHAR，VARCHAR

> **注意：**
>
> - Shardkey 字段必须是主键以及所有唯一索引的一部分
> - Shardkey字段的值不能为中文，因为Proxy不会转换字符集，所以不同字符集可能会路由到不同的分区
> - Shardkey=a 需放在SQL语句的最后

示例

```
DROP TABLE IF EXISTS employees_hash;
CREATE TABLE `employees_hash` (
`id`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id)
) shardkey=id;
```

语法

```
CREATE TABLE [IF NOT EXISTS] tbl_name
[(create_definition)]
[local_table_options]
shardkey=column_name

create_definition: {
col_name column_definition
| {INDEX | KEY} [index_name] [index_type] (key_part,...)
[index_option] ...
| [INDEX | KEY] [index_name] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] PRIMARY KEY
[index_type] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
[index_name] [index_type] (key_part,...)
[index_option] ...
```

```
}

column_definition: {
data_type [NOT NULL | NULL] [DEFAULT]
[AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
[COLLATE collation_name]
[COLUMN_FORMAT {FIXED | DYNAMIC | DEFAULT}]
[ENGINE_ATTRIBUTE [=] 'string']
| data_type
[UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
}

key_part: {col_name [(length)]} [ASC | DESC]

index_type:
USING {BTREE}

index_option: {
index_type | COMMENT 'string'
}
[local_table_options]
Local_table_option: {AUTO_INCREMENT [=] value
| [DEFAULT] CHARACTER SET [=] charset_name
| [DEFAULT] COLLATE [=] collation_name
| COMMENT [=] 'string'
| ENGINE [=] engine_name
| ROW_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}
| STATS_AUTO_RECALC [=] {DEFAULT | 0 | 1}
| STATS_PERSISTENT [=] {DEFAULT | 0 | 1}
| STATS_SAMPLE_PAGES [=] value)
}
```

一级RANGE | LIST分区

> **注意：**
>
> - DB 5.7版本不支持TDSQL_DISTRIBUTED BY range|list的语法。
> - tdsql_distributed by ...语法放置于create table ...的末尾
> - 如果分区键是char或者varchar类型，建议长度不超255。
> - 语句中指定的s1和s2是每个set的别名，基于实现原理，s1、s2不能自定义，只能按照顺序依次命名为s1、s2...
> - set的别名可通过/proxy/show status;获取到

支持类型

- – DATE，DATETIME，TIMESTAMP
- – TINYINT, SMALLINT, MEDIUMINT, INT, and BIGINT
- – CHAR，VARCHAR

```
-- RANGE分区
create table t1(a int key, b int) tdsql_distributed by range(a) (s1 values less than(100), s2 values less than(200));


-- LIST分区
CREATE TABLE `employees_list` (
`id`int NOT NULL,
```

```
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id)
)
TDSQL_DISTRIBUTED BY LIST(id) (
s1 VALUES IN (1,3,5),
s2 VALUES IN (2,4,6)
);
--查看set_1624363222_1和set_1624363251_3的别名分别为s1和s2：
```

> **禁止：**
>
> 避免使用TIMESTAMP类型作为分区键，因为timestamp受到时区的影响，同时只能使用到2038年

语法

```
CREATE TABLE [IF NOT EXISTS] tbl_name
[(create_definition)]
[local_table_options]
TDSQL_DISTRIBUTED BY range|list (column_name) [partition_options]

create_definition: {
col_name column_definition
| {INDEX | KEY} [index_name] [index_type] (key_part,...)
[index_option] ...
| [INDEX | KEY] [index_name] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] PRIMARY KEY
[index_type] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
[index_name] [index_type] (key_part,...)
[index_option] ...
}

column_definition: {
data_type [NOT NULL | NULL] [DEFAULT]
[AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
[COLLATE collation_name]
[COLUMN_FORMAT {FIXED | DYNAMIC | DEFAULT}]
[ENGINE_ATTRIBUTE [=] 'string']
| data_type
[UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
}

key_part: {col_name [(length)]} [ASC | DESC]

index_type:
USING {BTREE}

index_option: {
index_type | COMMENT 'string'
}
[local_table_options]
Local_table_option: {AUTO_INCREMENT [=] value
| [DEFAULT] CHARACTER SET [=] charset_name
| [DEFAULT] COLLATE [=] collation_name
```

```
| COMMENT [=] 'string'
| ENGINE [=] engine_name
| ROW_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}
| STATS_AUTO_RECALC [=] {DEFAULT | 0 | 1}
| STATS_PERSISTENT [=] {DEFAULT | 0 | 1}
| STATS_SAMPLE_PAGES [=] value)
}
partition_options:
PARTITION BY
| RANGE{(expr)}
| LIST{(expr)}
[(partition_definition [, partition_definition] ...)]

partition_definition:
PARTITION partition_name
[VALUES
{LESS THAN {(expr | value_list) | MAXVALUE}
|
IN (value_list)}]
[[STORAGE] ENGINE [=] engine_name]
[COMMENT [=] 'string']
```

二级分区

二级分区是将特定条件的数据进行分区处理，目前TDSQL支持Range和List两种格式的二级分区，具体建表语法和MySQL分区语法类似。

> **注意：**
>
> DB 5.7版本不支持TDSQL_DISTRIBUTED BY range|list的语法。

二级RANGE分区

Range支持类型

– DATE，DATETIME，TIMESTAMP

—支持year，month，day函数，此时传入的值转换为年月日，然后和分表信息进行对比

– TINYINT, SMALLINT, MEDIUMINT, INT , BIGINT

> **注意：**
>
> 分区使用小于符号“<”，如果要存储当年数据（例如，2017），需要创建小于往后一年（<2018）的分区，用户只需创建到当前的时间分区。TDSQL会自动增加后续分区，默认往后创建3个分区，以Year为例，TDSQL会自动往后创建3年（2018年、2019年、2020年）的分区，后续也会自动增减。

示例

```
-- 一级hash二级range分区：
DROP TABLE IF EXISTS employees_hash_range;
CREATE TABLE `employees_hash_range` (
`id`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id)
) shardkey=id
PARTITION BY RANGE (month(fired)) (
PARTITION p0 VALUES LESS THAN (202106),
PARTITION p1 VALUES LESS THAN (202107)
```

```
);
-- 一级list二级range分区:
DROP TABLE IF EXISTS employees_list_range;
CREATE TABLE `employees_list_range` (
`id`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id,fired)
)
PARTITION BY RANGE (month(fired)) (
PARTITION p0 VALUES LESS THAN (202106),
PARTITION p1 VALUES LESS THAN (202107)
)
TDSQL_DISTRIBUTED BY LIST(id) (
s1 VALUES IN (1,3,5),
s2 VALUES IN (2,4,6)
);
-- 一级range二级range分区:
DROP TABLE IF EXISTS employees_range_range;
CREATE TABLE `employees_range_range` (
`id`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id,fired)
)
PARTITION BY RANGE (month(fired)) (
PARTITION p0 VALUES LESS THAN (202106),
PARTITION p1 VALUES LESS THAN (202107)
)
TDSQL_DISTRIBUTED BY RANGE(id) (
s1 VALUES LESS THAN (6),
s2 VALUES LESS THAN (11)
);

-- 一级range二级range分区和子分区
DROP TABLE if exists tb_sub_ev;
CREATE TABLE tb_sub_ev (
id int NOT NULL,
purchased date NOT NULL,
PRIMARY KEY (id,purchased)
) ENGINE=InnoDB
PARTITION BY RANGE (YEAR(purchased))
SUBPARTITION BY HASH (TO_DAYS(purchased))
(PARTITION p0 VALUES LESS THAN (1990)
(SUBPARTITION s0 ENGINE = InnoDB,
SUBPARTITION s1 ENGINE = InnoDB),
PARTITION p1 VALUES LESS THAN (2000)
(SUBPARTITION s2 ENGINE = InnoDB,
SUBPARTITION s3 ENGINE = InnoDB))
TDSQL_DISTRIBUTED BY RANGE(id) (s1 values less than ('100'),s2 values less than ('1000'));
```

二级LIST分区

List支持类型

– DATE，DATETIME，TIMESTAMP —支持年月日函数

– TINYINT, SMALLINT, MEDIUMINT, INT , BIGINT

示例

```
-- 一级hash二级list分区：
DROP TABLE IF EXISTS employees_hash_list;
CREATE TABLE `employees_hash_list` (
`id`int NOT NULL,
`region`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id)
) shardkey=id
PARTITION BY LIST (region) (
PARTITION pRegion_1 VALUES IN (10, 30),
PARTITION pRegion_2 VALUES IN (20, 40)
);
-- 一级list二级list分区:
DROP TABLE IF EXISTS employees_list_list;
CREATE TABLE `employees_list_list` (
`id`int NOT NULL,
`region`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id, region)
)
PARTITION BY LIST (region) (
PARTITION pRegion_1 VALUES IN (10, 30),
PARTITION pRegion_2 VALUES IN (20, 40)
)
TDSQL_DISTRIBUTED BY LIST(id) (
s1 VALUES IN (1,3,5),
s2 VALUES IN (2,4,6)
);

一级range二级list分区:
DROP TABLE IF EXISTS employees_range_list;
CREATE TABLE `employees_range_list` (
`id`int NOT NULL,
`region`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id,region)
)
PARTITION BY LIST (region) (
PARTITION pRegion_1 VALUES IN (10, 30),
PARTITION pRegion_2 VALUES IN (20, 40)
)
TDSQL_DISTRIBUTED BY RANGE(id) (
s1 VALUES LESS THAN (6),
s2 VALUES LESS THAN (11)
);
```

语法

```
-- 一级hash分区 二级range|list分区：
CREATE TABLE [IF NOT EXISTS] tbl_name
 [(create_definition)]
 [local_table_options]
shardkey=column_name
[partition_options]
```

```
--一级range|list 二级range|list分区：
CREATE TABLE [IF NOT EXISTS] tbl_name
[(create_definition)]
[local_table_options]
[partition_options] TDSQL_DISTRIBUTED BY range|list (column_name)

create_definition: {
col_name column_definition
| {INDEX | KEY} [index_name] [index_type] (key_part,...)
[index_option] ...
| [INDEX | KEY] [index_name] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] PRIMARY KEY
[index_type] (key_part,...)
[index_option] ...
| [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
[index_name] [index_type] (key_part,...)
[index_option] ...
}

column_definition: {
data_type [NOT NULL | NULL] [DEFAULT]
[AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
[COLLATE collation_name]
[COLUMN_FORMAT {FIXED | DYNAMIC | DEFAULT}]
[ENGINE_ATTRIBUTE [=] 'string']
| data_type
[UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
}

key_part: {col_name [(length)]} [ASC | DESC]

index_type:
USING {BTREE}

index_option: {
index_type | COMMENT 'string'
}
[local_table_options]
Local_table_option: {AUTO_INCREMENT [=] value
| [DEFAULT] CHARACTER SET [=] charset_name
| [DEFAULT] COLLATE [=] collation_name
| COMMENT [=] 'string'
| ENGINE [=] engine_name
| ROW_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}
| STATS_AUTO_RECALC [=] {DEFAULT | 0 | 1}
| STATS_PERSISTENT [=] {DEFAULT | 0 | 1}
| STATS_SAMPLE_PAGES [=] value)
}
partition_options:
PARTITION BY
| RANGE{(expr)}
| LIST{(expr)}
[SUBPARTITION BY
{HASH(expr)
|(column_list) }
]
```

```
[(partition_definition [, partition_definition] ...)]

partition_definition:
PARTITION partition_name
[VALUES
{LESS THAN {(expr | value_list) | MAXVALUE}
|
IN (value_list)}]
[[STORAGE] ENGINE [=] engine_name]
[COMMENT [=] 'string' ]
[(subpartition_definition [, subpartition_definition] ...)]

subpartition_definition:
SUBPARTITION logical_name
[[STORAGE] ENGINE [=] engine_name]
[COMMENT [=] 'string']
```

创建广播表

广播表又名小表广播功能，创建时需要指定noshardkey_allset关键字。创建广播表后，每个节点都有该表的全量数据，且该表的所有操作都将广播到所有物理分片（set）中。

广播表主要用于提升跨节点组（ Set ）的Join 操作的性能，常用于配置表等。

示例

```
DROP TABLE IF EXISTS global_table_a;
CREATE TABLE global_table_a (a int, b int key) shardkey=noshardkey_allset;
```

创建单片表

普通表：又名单片表（Noshard表），创建时无须指定shardkey或者tdsql_distributed by关键字。单片表无需拆分且没有做任何特殊处理的表。其语法和MySQL完全一样，所有该类型表的全量数据默认存放在第一个物理节点组（Set）中。

示例

```
DROP TABLE IF EXISTS noshard_table;
CREATE TABLE noshard_table (a int, b int key);
```

创建临时表

临时表：创建表时可以使用 TEMPORARY 关键字。 TEMPORARY 表仅在当前会话中可见，并在会话关闭时自动删除。这意味着两个不同的会话可以使用相同的临时表名称，而不会相互冲突或与现有的同名非临时表发生冲突。（现有表是隐藏的，直到临时表被删除。）

> **注意：**
>
> - 需要使用注释透传才可创建临时表。关于注释透传功能请参考6.5节。
> - 使用/sets:allsets/创建的临时表，查询时可以指定任意setid。而如果使用/sets:setid/，则查询临时表时只能指定对应的setid。

示例

```
--使用/*sets:allsets*/创建临时表：
MySQL [test]> /*sets:allsets*/ DROP TABLE IF EXISTS new_tmp_tbl;
Query OK, 0 rows affected (0.01 sec)

MySQL [test]> /*sets:allsets*/ CREATE TEMPORARY TABLE new_tmp_tbl(id int primary key);
Query OK, 0 rows affected (0.00 sec)

MySQL [test]> /*sets:set_1624363222_1*/ select * from new_tmp_tbl;
Empty set (0.01 sec)

MySQL [test]> /*sets:set_1624363251_3*/ select * from new_tmp_tbl;
Empty set (0.00 sec)

MySQL [test]> /*sets:set_1626536042_12*/ select * from new_tmp_tbl;
Empty set (0.00 sec)

MySQL [test]> /*sets:allsets*/ select * from new_tmp_tbl;
Empty set (0.00 sec)

--使用/*sets:setid*/创建临时表：
MySQL [test]> /*sets:set_1624363222_1*/ CREATE TEMPORARY TABLE new_tmp_tbl(id int primary key);
Query OK, 0 rows affected (0.01 sec)

MySQL [test]> /*sets:set_1624363222_1*/ select * from new_tmp_tbl;
Empty set (0.01 sec)

MySQL [test]> /*sets:set_1624363251_3*/ select * from new_tmp_tbl;
ERROR 1146 (42S02): Table 'test.new_tmp_tbl' doesn't exist

MySQL [test]> /*sets:set_1626536042_12*/ select * from new_tmp_tbl;
ERROR 1146 (42S02): Table 'test.new_tmp_tbl' doesn't exist

MySQL [test]> /*sets:allsets*/ select * from new_tmp_tbl;
ERROR 1146 (42S02): Table 'test.new_tmp_tbl' doesn't exist
```

CREATE INDEX

通常，在使用 CREATE TABLE 创建表本身时在表上创建所有索引。该准则对于 InnoDB 表尤其重要，其中主键决定了数据文件中行的物理布局。 CREATE INDEX 使您能够向现有表添加索引。

语法：

```
CREATE [UNIQUE ] INDEX index_name
[index_type]
ON tbl_name (key_part,...)
[index_option]
[algorithm_option | lock_option] ...

key_part: {col_name [(length)]} [ASC | DESC]

index_option: {
index_type | COMMENT 'string'
}

index_type:
```

```
USING {BTREE}

algorithm_option:
ALGORITHM [=] {DEFAULT | INPLACE | COPY}

lock_option:
LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
```

> **注意：**
>
> - CREATE INDEX 不能用于创建 PRIMARY KEY；对于主键，请改用 ALTER TABLE。
> - 对于INNODB存储引擎，允许的索引类型为BTREE。

示例

```
创建测试表：
DROP TABLE IF EXISTS customer;
CREATE TABLE customer(cust_id int key,name varchar(200),job_id int,job_name varchar(300)) shardkey=cust_id;

使用using语句指定index_type，若不指定，默认为BTREE：
CREATE INDEX j_idx ON customer (name) USING BTREE;

创建列前缀索引：
CREATE INDEX idx_part_name ON customer (name(10));

创建降序索引：
CREATE INDEX idx_name_desc ON customer (name desc);

创建升序索引：
CREATE INDEX idx_name_asc ON customer (name asc);

创建唯一索引：
CREATE UNIQUE INDEX uniq_idx_job_id on customer(cust_id,job_id);

创建组合索引：
CREATE INDEX idx_cust on customer(name,job_name);

使用COMMENT语句指定索引页合并阈值：
CREATE INDEX j_idx_com ON customer (name) COMMENT 'MERGE_THRESHOLD=40';
```

CREATE VIEW

语法如下：

```
CREATE
[OR REPLACE]
VIEW view_name [(column_list)]
AS select_statement
```

> **注意：**
>
> CREATE VIEW 语句创建一个新视图，如果给出OR REPLACE 子句，则替换现有视图。 如果视图不存在，CREATE OR REPLACE VIEW 与 CREATE VIEW 相同。 如果视图确实存在，CREATE OR REPLACE VIEW 将替换它。

示例：

```
MySQL [test]> create view v1 as select * from employee;
Query OK, 0 rows affected (0.01 sec)
```

CREATE PROCEDURE

语法如下：

```
CREATE
[DEFINER = user]
PROCEDURE sp_name ([proc_parameter[,...]])
[characteristic ...] routine_body
proc_parameter:
[ IN | OUT | INOUT ] param_name type
type:
Any valid MySQL data type
characteristic: {
COMMENT 'string'
| LANGUAGE SQL
| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
}
routine_body:
Valid SQL routine statement
```

> **注意：**
>
> 这些语句用于创建存储过程。 默认情况下，存储过程与默认数据库相关联。 要将存储过程与给定数据库显式关联，请在创建时将名称指定为db_name.sp_name。

示例：

```
create database world;
use world;
create table cities(countryCode varchar(20),countryname varchar(20),city_code varchar(20) primary key,city_name varchar(20)) s
hardkey=city_code;

insert into world.cities(countryCode,countryname,city_code,city_name) values('CHN','CHINA','SH','SHANGHAI');
insert into world.cities(countryCode,countryname,city_code,city_name) values('CHN','CHINA','BJ','BEIJING');
insert into world.cities(countryCode,countryname,city_code,city_name) values('CHN','CHINA','SZ','SHENZHEN');
insert into world.cities(countryCode,countryname,city_code,city_name) values('CHN','CHINA','GZ','GUANGZHOU');
insert into world.cities(countryCode,countryname,city_code,city_name) values('CHN','CHINA','CD','CHENGDU');

--创建procedure
/*sets:allsets*/CREATE PROCEDURE citycount (IN country CHAR(3), OUT cities INT)
BEGIN
SELECT COUNT(*) INTO cities FROM world.cities
WHERE CountryCode = country;
END
//
--调用procedure
MySQL [world]> /*sets:allsets*/ CALL citycount('CHN', @cities)//

--查看调用结果，5条记录存储在3个set上：
```

```
MySQL [world]> /*sets:allsets*/SELECT @cities//
+---------+-------------------+
| @cities | info |
+---------+-------------------+
| 1 | set_1624363222_1 |
| 2 | set_1626536042_12 |
| 2 | set_1624363251_3 |
+---------+-------------------+
3 rows in set (0.01 sec)
```

DROP

最近更新时间: 2025-02-18 16:02:00

Drop database

语法如下：

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

> **注意：**
>
> DROP DATABASE 删除数据库中的所有表并删除数据库。 对此语句要非常小心！ 要使用 DROP DATABASE ，您需要 DROP database 的权限。 DROP SCHEMA 是DROP DATABASE的同义词。
>
> 删除数据库时， 不会自动删除专门为数据库授予的权限，必须手动删除它们。

示例：

```
DROP DATABASE test;
```

Drop table

语法如下：

```
DROP TABLE [IF EXISTS]
tbl_name [, tbl_name] ...
[RESTRICT | CASCADE]
```

> **注意：**
>
> - DROP TABLE 删除一个或多个表。 您必须拥有 DROP 每个表 的 权限。
> - 对于每个表，它将删除表定义和所有表数据。 如果表已分区，则该语句将删除表定义，其所有分区，存储在这些分区中的所有数据以及与已删除表关联的所有分区定义。
> - 删除表也会删除表的任何触发器。
> - DROP TABLE 导致隐式提交。
> - 删除表时，不会自动删除专门为该表授予的权限 。 必须手动删除它们。
> - 所有 innodb_force_recovery 设置都不支持 DROP TABLE
> - RESTRICT 和 CASCADE 关键字什么也不做。 它们被允许使从其他数据库系统移植更容易。

示例：

```
DROP TABLE test;
drop table test RESTRICT;
drop table test5 CASCADE;
```

Drop index

语法如下：

```
DROP INDEX index_name ON tbl_name
[algorithm_option | lock_option] ...

algorithm_option:
ALGORITHM [=] {DEFAULT | INPLACE | COPY}

lock_option:
LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
```

> **注意：**
>
> 要删除主键，索引名称始终为 PRIMARY，必须将其指定为带引号的标识符，因为 PRIMARY 是保留字：DROP INDEX PRIMARY ON t;

示例：

```
MySQL [test]> show create table customer\G;
*************************** 1. row ***************************
Table: customer
Create Table: CREATE TABLE `customer` (
`cust_id` int(11) NOT NULL,
`name` varchar(200) COLLATE utf8_bin DEFAULT NULL,
`job_id` int(11) DEFAULT NULL,
`job_name` varchar(300) COLLATE utf8_bin DEFAULT NULL,
PRIMARY KEY (`cust_id`),
UNIQUE KEY `uniq_idx_job_id` (`cust_id`,`job_id`),
KEY `idx_cust` (`name`,`job_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=cust_id
1 row in set (0.00 sec)

MySQL [test]> drop index uniq_idx_job_id on customer;
Query OK, 0 rows affected (0.04 sec)

MySQL [test]> drop index idx_cust on customer;
Query OK, 0 rows affected (0.08 sec)
```

ALTER

最近更新时间: 2025-02-18 16:02:00

ALTER TABLE

本章介绍ALTER相关用法。ALTER TABLE 更改表的结构。 例如，您可以添加或删除列、创建或销毁索引、更改现有列的类型或重命名列或表本身。 您还可以更改特征，例如用于表或表注释的存储引擎。

但是请注意：线上系统的DDL变更请通过赤兔管理控制台的online-ddl模块进行。

语法如下：

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

alter_option: {
table_options
| ADD [COLUMN] col_name column_definition
[FIRST | AFTER col_name]
| ADD [COLUMN] (col_name column_definition,...)
| ADD {INDEX | KEY} [index_name]
[index_type] (key_part,...) [index_option] ...
| ALGORITHM [=] {DEFAULT | INSTANT | INPLACE | COPY}
| CHANGE [COLUMN] old_col_name new_col_name column_definition
[FIRST | AFTER col_name]
| [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
| {DISABLE | ENABLE} KEYS
| DROP [COLUMN] col_name
| DROP {INDEX | KEY} index_name
| LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
| MODIFY [COLUMN] col_name column_definition
[FIRST | AFTER col_name]
| ORDER BY col_name [, col_name] ...
}

partition_options:
partition_option [partition_option] ...

partition_option: {
ADD PARTITION (partition_definition)
| DROP PARTITION partition_names
| TRUNCATE PARTITION {partition_names | ALL}
}

key_part: {col_name [(length)]} [ASC | DESC]

index_type:
USING {BTREE}

index_option: {
index_type | COMMENT 'string'
}

table_options:
```

```
table_option [[,] table_option] ...

table_option: {AUTO_INCREMENT [=] value
| [DEFAULT] CHARACTER SET [=] charset_name
| [DEFAULT] COLLATE [=] collation_name
| COMMENT [=] 'string'
| COMPRESSION [=] {'ZLIB' | 'LZ4' | 'NONE'}
| ENGINE [=] engine_name
| KEY_BLOCK_SIZE [=] value
| ROW_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}
| STATS_AUTO_RECALC [=] {DEFAULT | 0 | 1}
| STATS_PERSISTENT [=] {DEFAULT | 0 | 1}
| STATS_SAMPLE_PAGES [=] value)
}
```

注意：

- 要使用 ALTER TABLE ，你需要 ALTER ， CREATE 和 INSERT 权限。
- 不支持改变shardkey类型、删除shardkey的操作
- 一级分区，语法和单表一样，只能改变db上表结构，不能改变数据分布方式。
- 二级分区，支持添加和删除分区，语法和单表一样，range分区只能向后追加。

示例：

```
--创建一级hash分区表
DROP TABLE IF EXISTS sbtest1;
CREATE TABLE `sbtest1`
(`k` bigint(20) NOT NULL,
`id` bigint(20) NOT NULL,
`c` char(120) NOT NULL,
`pad` char(60) NOT NULL,
`balance` int(11) NOT NULL,
`lastModifyTime` datetime,
PRIMARY KEY (`k`,`id`),
KEY `k_1` (`k`))
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;

--添加删除索引
alter table sbtest1 add index idx_blc (balance);
alter table sbtest1 drop index idx_blc;

--修改表字段类型
alter table sbtest1 modify column pad varchar(50);

--增加一个新列为第一列
alter table sbtest1 add column col1 INT NOT NULL first;

--增加一个到指定列之后
alter table sbtest1 add column col_after_pad INT NOT NULL after pad;

--修改表增加字段
alter table sbtest1 add column mark varchar(50);

--修改表字段名字
alter table sbtest1 change column k k_new1 bigint(20);

--修改表删除字段
alter table sbtest1 drop column mark;
```

```
--重组表
ALTER TABLE sbtest1 ENGINE = InnoDB;

--更改 InnoDB 表以使用压缩行存储格式：
ALTER TABLE sbtest1 ROW_FORMAT = COMPRESSED;

--添加（或更改）表注释：
ALTER TABLE sbtest1 COMMENT = 'New table comment';
```

示例：

```
创建二级分区表：
DROP TABLE if exists customers_1;
CREATE TABLE customers_1 (
first_name VARCHAR(25) primary key,
last_name VARCHAR(25),
street_1 VARCHAR(30),
street_2 VARCHAR(30),
city_name VARCHAR(15),
renewal DATE
) shardkey=first_name
PARTITION BY LIST (city_name) (
PARTITION pRegion_1 VALUES IN('BJ', 'GZ', 'SZ'),
PARTITION pRegion_2 VALUES IN('SH', 'CD'),
PARTITION pRegion_3 VALUES IN('GY'),
PARTITION pRegion_4 VALUES IN('HZ')
);

删除分区：
ALTER TABLE customers_1 drop partition pRegion_4;

增加分区：
ALTER TABLE customers_1 add partition (partition pRegion_4 VALUES IN('TJ'));

截断分区：
ALTER TABLE customers_1 truncate partition pRegion_4;
```

示例：

```
创建二级分区表：
DROP TABLE IF EXISTS employees_list_range;
CREATE TABLE `employees_list_range` (
`id`int NOT NULL,
`city` varchar(10),
`fired` DATE NOT NULL DEFAULT '1970.01.01',
PRIMARY KEY(id,fired)
)
PARTITION BY RANGE (month(fired)) (
PARTITION p0 VALUES LESS THAN (202106),
PARTITION p1 VALUES LESS THAN (202107)
)
TDSQL_DISTRIBUTED BY LIST(id) (
s1 VALUES IN (1,3,5),
s2 VALUES IN (2,4,6)
);

删除分区：
ALTER TABLE employees_list_range drop partition p1;
```

增加分区：
ALTER TABLE employees_list_range add partition(partition p2 values less than (202108));

截断分区：
ALTER TABLE employees_list_range truncate partition p0;

TRUNCATE

最近更新时间: 2025-02-18 16:02:00

语法如下：

```
TRUNCATE [TABLE] tbl_name
```

> **注意：**
>
> - 需要有drop权限
> - 截断操作会导致隐式提交，因此无法回滚
> - 第一次执行truncate若失败，则进行第二次truncate

示例：

```
truncate table t1;
```

数据库操作语言（DML）

最近更新时间: 2025-02-18 16:02:00

本节主要介绍 DML 语句中常用的Select（查询）、Insert（插入）、Replace（替换）、Update（更新）及Delete（删除）指令。

SELECT

基础查询语法

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
select_expr [, select_expr] ...
[FROM table_references
[PARTITION partition_list]]
[WHERE where_condition]
[GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
[ASC | DESC], ... [WITH ROLLUP]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[FOR {UPDATE | SHARE}
[OF tbl_name [, tbl_name] ...]
[NOWAIT | SKIP LOCKED]
| LOCK IN SHARE MODE]
```

示例：

```
drop table if exists test1;
create table test1 ( a int key, b int, c char(20) ) shardkey=a;
drop table if exists test2;
create table test2 ( a int key, d int, e char(20) ) shardkey=a;

insert into test1 (a,b,c) values(1,2,"record1"),(2,3,"record2");
insert into test2 (a,d,e) values(1,3,"test2_record1"),(2,3,"test2_record2");

select t1.a,t1.b,t1.c,t2.a,t2.d,t2.e from test1 t1 join test2 t2 on t1.b=t2.d;


select t1.a,t1.b,t1.c from test1 t1 where t1.a in (select a from test2);

select t1.a,t1.b,t1.c from test1 t1 where exists (select t2.a,t2.d,t2.e from test2 t2 where t2.a=t1.b);

select t1.a, count(1) from test1 t1 where exists (select t2.a,t2.d,t2.e from test2 t2 where t2.a=t1.a) group by t1.a;


select distinct count(1) from test1 t1 where exists (select t2.a,t2.d,t2.e from test2 t2 where t2.a=t1.a) group by t1.a;

select count(distinct t1.a) from test1 t1 where exists (select t2.a,t2.d,t2.e from test2 t2 where t2.a=t1.a);
```

join

TDSQL支持对 SELECT 语句和多表 DELETE 和 UPDATE 操作的join。

分表间join示例

如果分表之间带有分表键相等的条件，则相当于单机Join。

示例：

```
--构建两张测试表：
DROP TABLE IF EXISTS `test_join_shard_table1`;
CREATE TABLE `test_join_shard_table1` (
`id` int(10) NOT NULL,
`b` varchar(10) NOT NULL DEFAULT '',
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_join_shard_table1 (id, b, c) VALUES
(1,"test1",1), (2,"test2",2), (3,"test3",3),
(4,"test4",4), (5,"test5",5), (6,"test6",6),
(7,"test7",7), (8,"test8",8), (9,"testX",11);

DROP TABLE IF EXISTS `test_join_shard_table2`;
CREATE TABLE `test_join_shard_table2` (
`id` int(10) NOT NULL,
`d` datetime,
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_join_shard_table2 (id, d, c) VALUES
(1,NOW(),1), (2,NOW(),2), (3,NOW(),3),
(4,NOW(),4), (5,NOW(),5), (6,NOW(),6),
(7,NOW(),7), (8,NOW(),8), (9,NOW(),10);

--检查分布式测试表的数据分布情况：
/*sets:allsets*/ select * from test_join_shard_table1;
/*sets:allsets*/ select * from test_join_shard_table2;

--执行带INNER JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
INNER JOIN test_join_shard_table2 test2
ON test1.c=test2.c
ORDER BY NAME;

--执行带LEFT JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
LEFT JOIN test_join_shard_table2 test2
ON test1.c=test2.c
ORDER BY NAME;

--执行带RIGHT JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
RIGHT JOIN test_join_shard_table2 test2
ON test1.c=test2.c
ORDER BY NAME;
```

```
--执行带FULL JOIN的SELECT查询语句，笛卡尔积
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
CROSS JOIN test_join_shard_table2 test2
ORDER BY NAME;
```

分表和广播表join示例

跨分片的分表与广播表，效果相当于单机 Join。

示例：

```
--构建两张测试表：
DROP TABLE IF EXISTS `test_join_shard_table1`;
CREATE TABLE `test_join_shard_table1` (
`id` int(10) NOT NULL,
`b` varchar(10) NOT NULL DEFAULT '',
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_join_shard_table1 (id, b, c) VALUES
(1,"test1",1), (2,"test2",2), (3,"test3",3),
(4,"test4",4), (5,"test5",5), (6,"test6",6),
(7,"test7",7), (8,"test8",8), (9,"testX",11);

DROP TABLE IF EXISTS `test_join_group_table2`;
CREATE TABLE `test_join_group_table2` (
`id` int(10) NOT NULL,
`d` datetime,
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=noshardkey_allset;
INSERT INTO test_join_group_table2 (id, d, c) VALUES
(1,NOW(),1), (2,NOW(),2), (3,NOW(),3),
(4,NOW(),4), (5,NOW(),5), (6,NOW(),6),
(7,NOW(),7), (8,NOW(),8), (9,NOW(),10);

--检查分布式测试表的数据分布情况：
/*sets:allsets*/ select * from test_join_shard_table1;
/*sets:allsets*/ select * from test_join_group_table2;

--执行带INNER JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
INNER JOIN test_join_group_table2 test2
ON test1.c=test2.c
ORDER BY NAME;

--执行带LEFT JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
LEFT JOIN test_join_group_table2 test2
ON test1.c=test2.c
ORDER BY NAME;

--执行带RIGHT JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
RIGHT JOIN test_join_group_table2 test2
```

```
ON test1.c=test2.c
ORDER BY NAME;

--执行带FULL JOIN的SELECT查询语句，笛卡尔积
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
CROSS JOIN test_join_group_table2 test2
ORDER BY NAME;
```

分表和单表join示例

示例：

```
--构建两张测试表：
DROP TABLE IF EXISTS `test_join_shard_table1`;
CREATE TABLE `test_join_shard_table1` (
`id` int(10) NOT NULL,
`b` varchar(10) NOT NULL DEFAULT '',
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_join_shard_table1 (id, b, c) VALUES
(1,"test1",1), (2,"test2",2), (3,"test3",3),
(4,"test4",4), (5,"test5",5), (6,"test6",6),
(7,"test7",7), (8,"test8",8), (9,"testX",11);

DROP TABLE IF EXISTS `test_join_noshard_table2`;
CREATE TABLE `test_join_noshard_table2` (
`id` int(10) NOT NULL,
`d` datetime,
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
INSERT INTO test_join_noshard_table2 (id, d, c) VALUES
(1,NOW(),1), (2,NOW(),2), (3,NOW(),3),
(4,NOW(),4), (5,NOW(),5), (6,NOW(),6),
(7,NOW(),7), (8,NOW(),8), (9,NOW(),10);

--检查分布式测试表的数据分布情况：
/*sets:allsets*/ select * from test_join_shard_table1;
--检查单片表的数据：
select * from test_join_noshard_table2;

--执行带INNER JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
INNER JOIN test_join_noshard_table2 test2
ON test1.c=test2.c
ORDER BY NAME;

--执行带LEFT JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
LEFT JOIN test_join_noshard_table2 test2
ON test1.c=test2.c
ORDER BY NAME;

--执行带RIGHT JOIN的SELECT查询语句
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
```

```
FROM test_join_shard_table1 test1
RIGHT JOIN test_join_noshard_table2 test2
ON test1.c=test2.c
ORDER BY NAME;

--执行带FULL JOIN的SELECT查询语句，笛卡尔积
SELECT test1.id, test1.b AS NAME, test2.d AS TIME
FROM test_join_shard_table1 test1
CROSS JOIN test_join_noshard_table2 test2
ORDER BY NAME;
```

跨分片update/delete join示例

示例：

```
--创建测试表：
DROP TABLE IF EXISTS `test_join_shard_table1`;
CREATE TABLE `test_join_shard_table1` (
`id` int(10) NOT NULL,
`b` varchar(10) NOT NULL DEFAULT '',
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_join_shard_table1 (id, b, c) VALUES
(1,"test1",1), (2,"test2",2), (3,"test3",3),
(4,"test4",4), (5,"test5",5), (6,"test6",6),
(7,"test7",7), (8,"test8",8), (9,"testX",11);

DROP TABLE IF EXISTS `test_join_shard_table2`;
CREATE TABLE `test_join_shard_table2` (
`id` int(10) NOT NULL,
`d` datetime,
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_join_shard_table2 (id, d, c) VALUES
(1,NOW(),1), (2,NOW(),2), (3,NOW(),3),
(4,NOW(),4), (5,NOW(),5), (6,NOW(),6),
(7,NOW(),7), (8,NOW(),8), (9,NOW(),10);

--检测分布式测试表的数据分布情况
/*sets:allsets*/ select * from test_join_shard_table1;
/*sets:allsets*/ select * from test_join_shard_table2;

--UPDATE...JOIN...ON...SET语句，单字段：
UPDATE test_join_shard_table1 test1
INNER JOIN test_join_shard_table2 test2
ON test1.c=test2.c SET test1.b="TEXTXXXXX"
WHERE test1.id>7;
SELECT * FROM test_join_shard_table1;

--UPDATE...JOIN...ON...SET语句，同一表多字段
UPDATE test_join_shard_table1 test1
INNER JOIN test_join_shard_table2 test2
ON test1.c=test2.c
SET test1.b="TEXTSSSS", test1.c=88
WHERE test1.id>7;
SELECT * FROM test_join_shard_table1;
```

```
--DELETE...FROM...JOIN...ON语句
DELETE test1 FROM test_join_shard_table1 test1
INNER JOIN test_join_shard_table2 test2
ON test1.c=test2.c
WHERE test1.id>7;
SELECT * FROM test_join_shard_table1;
```

union语法

UNION 将来自多个 SELECT 语句的结果组合到一个结果集中。

语法如下：

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

> **注意：**
>
> 参与UNION的表所select的列的个数需要保持一致。
>
> UNION 结果集的列名取自第一个 SELECT 语句的列名。

示例:

```
DROP TABLE IF EXISTS t1;
create table t1 (a int primary key, b int) shardkey=a;
DROP TABLE IF EXISTS t2;
create table t2 (a int primary key, b int) shardkey=a;
select * from t1 where t1.a in (select a from t2) union select * from t2 where t2.a>22;
```

各种表的组合场景：

```
分表：
DROP TABLE IF EXISTS s1;
create table s1 (a int primary key, b int) shardkey=a;
DROP TABLE IF EXISTS s2;
create table s2 (a int primary key, b int) shardkey=a;
单表：
DROP TABLE IF EXISTS ns1;
create table ns1 (a int primary key, b int);
DROP TABLE IF EXISTS ns2;
create table ns2 (a int primary key, b int);
广播表：
DROP TABLE IF EXISTS g1;
create table g1 (a int primary key, b int) shardkey=noshardkey_allset;
DROP TABLE IF EXISTS g2;
create table g2 (a int primary key, b int) shardkey=noshardkey_allset;
二级分区表：
DROP TABLE IF EXISTS p1;
create table p1 (a int, b int, PRIMARY KEY(a)) shardkey=a PARTITION BY range (b) (PARTITION p0 values less than (100), PARTITI
ON p1 values less than (200));
DROP TABLE IF EXISTS p2;
create table p2 (a int, b int, PRIMARY KEY(a)) shardkey=a PARTITION BY range (b) (PARTITION p0 values less than (100), PARTITI
ON p1 values less than (200));
```

```
各种类型表之间的union
select * from s1 union select * from s2;
select * from ns1 union select * from ns2;
select * from g1 union select * from g2;
select * from s1 union select * from ns1;
select * from p1 union select * from p2;
select * from s1 where not exists (select * from s2 where s2.a=s1.a order by s2.a) or b<10 union select * from s2 where s2.a>22;
select a, sum(b) from s1 group by a union select * from s2;
select a, sum(b) from s1 union select * from s2;
select distinct(a) from s1 union select a from s2;
select distinct(a), b from s1 union select a,b from s2;
```

子查询

语法如下：

```
SELECT ...
FROM table
WHERE expr operator
(SELECT select_list FROM table)
```

> **注意：**
>
> 一般情况下，由于子查询效率不高，尽量使用join的代替子查询

示例：

```
DROP TABLE if exists `test_shard_table1`;
CREATE TABLE `test_shard_table1` (
`id` int(10) NOT NULL,
`b` varchar(10) NOT NULL DEFAULT '',
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_shard_table1 (id, b, c) VALUES
(1,"test1",1), (2,"test2",2), (3,"test3",3),
(4,"test4",4), (5,"test5",5), (6,"test6",6),
(7,"test7",7), (8,"test8",8), (9,"testX",11);

DROP TABLE if exists `test_shard_table2`;
CREATE TABLE `test_shard_table2` (
`id` int(10) NOT NULL,
`d` datetime,
`c` int(10) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin shardkey=id;
INSERT INTO test_shard_table2 (id, d, c) VALUES
(1,NOW(),1), (2,NOW(),2), (3,NOW(),3),
(4,NOW(),4), (5,NOW(),5), (6,NOW(),6),
(7,NOW(),7), (8,NOW(),8), (9,NOW(),10);

SELECT COUNT(B)
FROM test_shard_table1
WHERE id IN
(SELECT c FROM test_shard_table2 WHERE id>5);
```

```
SELECT MAX(c), MIN(c)
FROM test_shard_table1 WHERE c IN
(SELECT c FROM test_shard_table2 WHERE id<8)
AND id>4 ORDER BY c;
```

INSERT

语法如下：

```
INSERT [IGNORE]
[INTO] tbl_name
[PARTITION (partition_name [, partition_name] ...)]
[(col_name [, col_name] ...)]
{{VALUES | VALUE} (value_list) [, (value_list)] ...
|
VALUES row_constructor_list
}
[ON DUPLICATE KEY UPDATE assignment_list]

INSERT [IGNORE]
[INTO] tbl_name
[PARTITION (partition_name [, partition_name] ...)]
SET assignment_list
[ON DUPLICATE KEY UPDATE assignment_list]

INSERT [IGNORE]
[INTO] tbl_name
[PARTITION (partition_name [, partition_name] ...)]
[(col_name [, col_name] ...)]
{SELECT ... | TABLE table_name}
[ON DUPLICATE KEY UPDATE assignment_list]

value:
{expr | DEFAULT}

value_list:
value [, value] ...

row_constructor_list:
ROW(value_list)[, ROW(value_list)][, ...]

assignment:
col_name = [row_alias.]value

assignment_list:
assignment [, assignment] ...
```

> **注意：**
>
> 对于分片表执行insert命令时，字段必须包含Shardkey，否则系统会拒绝执行SQL命令，因为Proxy无法判断SQL语句发送的后端数据库节点位置

示例：

```
--测试不带shardkey字段：
MySQL [test]> DROP TABLE IF EXISTS test1;
Query OK, 0 rows affected (0.12 sec)

MySQL [test]> create table test1(a int not null primary key,b int,c char(10)) shardkey=a;
Query OK, 0 rows affected (2.64 sec)

MySQL [test]> insert into test1 (b,c) values(10,"record3");
ERROR 683 (HY000): Proxy ERROR: Get shardkeys return error: insert/replace must contain shardkey column

MySQL [test]> insert into test1 (a,c) values(40,"records5");
Query OK, 1 row affected (0.03 sec)


--测试不携带ignore，会发生主键冲突
MySQL [test]> drop table if exists t1_1_1;
Query OK, 0 rows affected (0.10 sec)
MySQL [test]> create table t1_1_1 (a int primary key, b int) shardkey=a;
Query OK, 0 rows affected (0.18 sec)
MySQL [test]> drop table if exists t1_1_2;
Query OK, 0 rows affected (0.07 sec)
MySQL [test]> create table t1_1_2 (a int primary key) shardkey=a;
Query OK, 0 rows affected (0.18 sec)

MySQL [test]> insert into t1_1_1 (a,b) values (1,0),(2,0),(3,1);
Query OK, 3 rows affected (0.01 sec)

MySQL [test]> select * from t1_1_1;
+---+------+
| a | b |
+---+------+
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
+---+------+
3 rows in set (0.00 sec)

MySQL [test]> insert into t1_1_2 select b from t1_1_1;
ERROR 913 (HY000): Proxy ERROR:Join internal error: Duplicate entry '0' for key 'PRIMARY'

--携带ignore，会写入部分数据，重复的数据只写一次
MySQL [test]> insert ignore into t1_1_2 select b from t1_1_1;
Query OK, 2 rows affected, 1 warning (0.00 sec)

MySQL [test]> select * from t1_1_2 order by a;
+---+
| a |
+---+
| 0 |
| 1 |
+---+
2 rows in set (0.00 sec)
```

REPLACE

语法如下：

```
REPLACE
[INTO] tbl_name
[PARTITION (partition_name [, partition_name] ...)]
[(col_name [, col_name] ...)]
{{VALUES | VALUE} (value_list) [, (value_list)] ...
|
VALUES row_constructor_list
}

REPLACE
[INTO] tbl_name
[PARTITION (partition_name [, partition_name] ...)]
SET assignment_list

REPLACE
[INTO] tbl_name
[PARTITION (partition_name [, partition_name] ...)]
[(col_name [, col_name] ...)]
{SELECT ... | TABLE table_name}

value:
{expr | DEFAULT}

value_list:
value [, value] ...

row_constructor_list:
ROW(value_list)[, ROW(value_list)][, ...]

assignment:
col_name = value

assignment_list:
assignment [, assignment] ...
```

> **注意：**
>
> 对于分片表执行Replace命令时，字段必须包含Shardkey，否则系统会拒绝执行SQL命令，因为Proxy无法判断SQL语句发送的后端数据库节点位置

示例：

```
--测试不带shardkey字段：
MySQL [test]> DROP TABLE IF EXISTS test5;
MySQL [test]> create table test5(a int not null primary key,b int,c char(10)) shardkey=a;
Query OK, 0 rows affected (0.27 sec)

MySQL [test]> replace into test5 (b,c) values(10,"record3");
ERROR 683 (HY000): Proxy ERROR: Get shardkeys return error: insert/replace must contain shardkey column

MySQL [test]> replace into test5(a,b,c) values(3,40,"record1");
Query OK, 1 row affected (0.00 sec)

--测试加载多条数据
MySQL [test]> replace into test5(a,b,c) values(4,50,"record2"),(5,60,"record3"),(6,70,"record4"),(7,80,"record5"),(8,90,"record6"),(9,
100,"record7");
```

```
Query OK, 6 rows affected (0.00 sec)

--测试replace select语句
drop table if exists t1_1_1;
create table t1_1_1 (a int not null primary key, b char(10)) shardkey=a;
drop table if exists t1_1_2;
create table t1_1_2 (a int not null primary key, b char(10)) shardkey=a;

insert into t1_1_1 (a,b) values (1,"t1:1"),(3,"t1:3");
insert into t1_1_2 (a,b) values (2,"t2:2"), (3,"t2:3");
replace into t1_1_1 select * from t1_1_2;
```

DELETE

语法如下：

```
DELETE [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
[PARTITION (partition_name [, partition_name] ...)]
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

> **注意**：
>
> 为了安全考虑，分表和广播表执行delete指令的时候必须带“ where ”条件，否则系统拒绝执行该SQL命令

示例：

```
--测试不带shardkey的delete
MySQL [test]> DROP TABLE IF EXISTS test3;
MySQL [test]> create table test3(a int not null primary key,b int,c char(10)) shardkey=a;

MySQL [test]> insert into test3(a,b,c) values (1,2,'A');
Query OK, 1 row affected (0.00 sec)

MySQL [test]> delete from test3;
ERROR 913 (HY000): Proxy ERROR:Join internal error: delete query has no where clause

MySQL [test]> delete from test3 where a=1;
Query OK, 1 rows affected (0.00 sec)

--测试包含子查询的delete
drop table if exists t1_1;
create table t1_1 (a int primary key, b int) shardkey=a;
drop table if exists t1_2;
create table t1_2 (a int primary key, b int) shardkey=a;
insert into t1_1 (a,b) values (20,20);
insert into t1_2 (a,b) values (20,20);
insert into t1_1 (a,b) values (19,19);
insert into t1_2 (a,b) values (19,19);
insert into t1_1 (a,b) values (18,18);
insert into t1_2 (a,b) values (18,18);
insert into t1_1 (a,b) values (17,17);
insert into t1_2 (a,b) values (17,17);
insert into t1_1 (a,b) values (16,16);
```

```
insert into t1_2 (a,b) values (16,16);
insert into t1_1 (a,b) values (15,15);
insert into t1_2 (a,b) values (15,15);
insert into t1_1 (a,b) values (14,14);
insert into t1_2 (a,b) values (14,14);
insert into t1_1 (a,b) values (13,13);
insert into t1_2 (a,b) values (13,13);
insert into t1_1 (a,b) values (12,12);
insert into t1_2 (a,b) values (12,12);
insert into t1_1 (a,b) values (11,11);
insert into t1_2 (a,b) values (11,11);
insert into t1_1 (a,b) values (10,10);
insert into t1_2 (a,b) values (10,10);
insert into t1_1 (a,b) values (9,9);
insert into t1_2 (a,b) values (9,9);
insert into t1_1 (a,b) values (8,8);
insert into t1_2 (a,b) values (8,8);
insert into t1_1 (a,b) values (7,7);
insert into t1_2 (a,b) values (7,7);
insert into t1_1 (a,b) values (6,6);
insert into t1_2 (a,b) values (6,6);
insert into t1_1 (a,b) values (5,5);
insert into t1_2 (a,b) values (5,5);
insert into t1_1 (a,b) values (4,4);
insert into t1_2 (a,b) values (4,4);
insert into t1_1 (a,b) values (3,3);
insert into t1_2 (a,b) values (3,3);
insert into t1_1 (a,b) values (2,2);
insert into t1_2 (a,b) values (2,2);
insert into t1_1 (a,b) values (1,1);
insert into t1_2 (a,b) values (1,1);
delete from t1_1 where a in (select b from t1_2 where a<10);
delete from t1_1 where exists(select 1 from t1_2 where t1_1.a=t1_2.b and t1_2.a>8);

--测试携带和不携带ignore的delete
drop table if exists t8_1;
create table t8_1 (a int NOT NULL, b int, primary key (a));
drop table if exists t8_2;
create table t8_2 (a int NOT NULL, b int, primary key (a));
drop table if exists t8_3;
create table t8_3 (a int NOT NULL, b int, primary key (a));
insert into t8_1 (a,b) values (0, 10),(1, 11),(2, 12);
insert into t8_2 (a,b) values (33, 10),(0, 11),(2, 12);
insert into t8_3 (a,b) values (1, 21),(2, 12),(3, 23);

--不带ignore的情况
MySQL [test]> delete t8_1.*, t8_2.* from t8_1,t8_2 where t8_1.a = t8_2.a and t8_1.b <> (select b from t8_3 where t8_1.a < t8_3.
a);
ERROR 1242 (21000): Subquery returns more than 1 row

--携带ignore的情况
MySQL [test]> delete ignore t8_1.*, t8_2.* from t8_1,t8_2 where t8_1.a = t8_2.a and t8_1.b <> (select b from t8_3 where t8_1.a <
t8_3.a);
Query OK, 2 rows affected, 2 warnings (0.01 sec)
```

UPDATE

语法如下：

```
UPDATE [IGNORE] table_reference
SET assignment_list
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]

value:
{expr | DEFAULT}

assignment:
col_name = value

assignment_list:
assignment [, assignment] ...
```

> **注意：**
>
> - 分区表不支持更新shardkey，需用显示开启事务，再执行delete和insert替代update
> - 分区表不支持update set的值为子查询
> - 为了安全考虑，分表和广播表执行update指令的时候必须带“ where ”条件，否则系统拒绝执行该SQL命令

示例：

```
--测试update的累加
DROP TABLE IF EXISTS t1_1;
CREATE TABLE t1_1
(place_id int (10) unsigned NOT NULL,
shows int(10) unsigned DEFAULT '0' NOT NULL,
ishows int(10) unsigned DEFAULT '0' NOT NULL,
ushows int(10) unsigned DEFAULT '0' NOT NULL,
clicks int(10) unsigned DEFAULT '0' NOT NULL,
iclicks int(10) unsigned DEFAULT '0' NOT NULL,
uclicks int(10) unsigned DEFAULT '0' NOT NULL,
ts timestamp,PRIMARY KEY (place_id,ts))
shardkey=place_id;

INSERT INTO t1_1 (place_id,shows,ishows,ushows,clicks,iclicks,uclicks,ts)VALUES (1,0,0,0,0,0,0,20000928174434);

UPDATE t1_1 SET shows=shows+1,ishows=ishows+1,ushows=ushows+1,clicks=clicks+1,iclicks=iclicks+1,uclicks=uclicks+1 WH
ERE place_id=1 AND ts>="2000-09-28 00:00:00";

--测试带有子查询的update
drop table if exists t1_1;
create table t1_1 (a int primary key, b int) shardkey=a;
drop table if exists t1_2;
create table t1_2 (a int primary key, b int) shardkey=a;
drop table if exists t1_3;
create table t1_3 (a int primary key, b int) shardkey=a;
insert into t1_1(a, b) values (10, 10);
insert into t1_1(a, b) values (9, 9);
insert into t1_1(a, b) values (8, 8);
insert into t1_1(a, b) values (7, 7);
insert into t1_1(a, b) values (6, 6);
```

```
insert into t1_1(a, b) values (5, 5);
insert into t1_1(a, b) values (4, 4);
insert into t1_1(a, b) values (3, 3);
insert into t1_1(a, b) values (2, 2);
insert into t1_1(a, b) values (1, 1);
insert into t1_2 select * from t1_1;
insert into t1_3 select * from t1_1;
update t1_1 set b=1 where exists(select * from t1_2 where t1_1.a=t1_2.a order by 1) limit 3;
update t1_1 set b=-1 where a in (select b from t1_2 order by 1) order by a limit 3;

--update不支持更新主键
MySQL [test]> update t1_1 set a=b where exists(select 1 from t1_2 where a=t1_1.b);
ERROR 658 (HY000): Proxy ERROR: Join internal error: cannot update primary key

--update不支持更新shardkey
MySQL [test]> update t1_1 set a=200 where b=1;
ERROR 682 (HY000): Proxy ERROR: Something went wrong: can not update the shardkey

--显示开启事务用delete/insert替代update
begin;
delete from t1_1 where b=1;
insert into t1_1(a,b) values(200,1);
commit;

--不支持update列表中带有sum的子查询
MySQL [test]> update t1_1 set b=(select max(b) from t1_2 where t1_2.a=t1_1.a) where 1;
ERROR 658 (HY000): Proxy ERROR: Join internal error: do not support subquery/sum in update list

--多表更新语法，但只更新一个表
MySQL [test]> update t1_1, t1_2 set t1_1.b=-1 where t1_1.a=t1_2.b and t1_2.a<3;
Query OK, 0 rows affected (0.01 sec)

--不支持order by和limit
MySQL [test]> update t1_1, t1_2 set t1_1.b=-1 where t1_1.a=t1_2.b and t1_2.a<3 order by t1_1.a limit 3;
ERROR 658 (HY000): Proxy ERROR: Join internal error: Incorrect usage of UPDATE and ORDER

--不支持更新多个表
MySQL [test]> update t1_1, t1_2 set t1_1.b=-1, t1_2.b=-1 where t1_1.a=t1_2.b and t1_2.a<3;
ERROR 658 (HY000): Proxy ERROR: Join internal error: multi update is not supported yet.

--更新一个表，但value引用了另外一个表
MySQL [test]> update t1_1, t1_2 set t1_1.b= t1_2.b+1 where t1_1.a=t1_2.b and t1_2.a<3;
Query OK, 2 rows affected (0.01 sec)

--不支持list分区表更新分区键
drop table if exists list_user;
CREATE TABLE list_user
(id int, name varchar(255),
city varchar(255), primary key(id))
shardkey=id
PARTITION by list(city)
(PARTITION p0 values in ('Beijin','Shanghai','Shenzhen'),
PARTITION p1 values in ('Nanjin', 'Chongqing','Wuhan'));
insert into list_user (id, name,city) values (1,'Rain','Beijin'),(22,'Storm','Beijin'),(103,'wind','Nanjin');

MySQL [test]> update list_user set city='Nanjin' where id in (select id from list_user,t1_1 where t1_1.a=list_user.id and t1_1.a <3
);
ERROR 913 (HY000): Proxy ERROR:Join internal error: sub partitioned table do not support such update yet!
```

```
MySQL [test]> update list_user set city='Nanjin' where id=1;
ERROR 682 (HY000): Proxy ERROR: Something went wrong: can not update the subshardkey

--不支持范围分区表更新分区键
drop table if exists range_part;
create table range_part
(a int, b int, PRIMARY KEY(a))
shardkey=a
PARTITION BY range (b)
(PARTITION p0 values less than (100),
PARTITION p1 values less than (200));
insert into range_part (a,b) values (1,11),(22,2),(103,1);

MySQL [test]> update range_part set b=11 where a in (select a from range_part,t1_1 where t1_1.a=range_part.id and t1_1.a <3
);
ERROR 913 (HY000): Proxy ERROR:Join internal error: sub partitioned table do not support such update yet!

MySQL [test]> update range_part set b=11 where a=103;
ERROR 682 (HY000): Proxy ERROR: Something went wrong: can not update the subshardkey
```

效用声明

最近更新时间: 2025-02-18 16:02:00

DESCRIBE 语句

DESCRIBE 用于获取表结构信息：

示例：

```
mysql> DESCRIBE City;
+------------+----------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+----------------+
| Id | int(11) | NO | PRI | NULL | auto_increment |
| Name | char(35) | NO | | | |
| Country | char(3) | NO | UNI | | |
| District | char(20) | YES | MUL | | |
| Population | int(11) | NO | | 0 | |
+------------+----------+------+-----+---------+----------------+
```

EXPLAIN 语句

语法

```
{EXPLAIN | DESCRIBE | DESC}
tbl_name [col_name | wild]

{EXPLAIN | DESCRIBE | DESC}
[explain_type]
{explainable_stmt | FOR CONNECTION connection_id}

{EXPLAIN | DESCRIBE | DESC} ANALYZE [FORMAT = TREE] select_statement

explain_type: {
FORMAT = format_name
}

format_name: {
TRADITIONAL
| JSON
| TREE
}

explainable_stmt: {
SELECT statement
| TABLE statement
| DELETE statement
| INSERT statement
```

```
| REPLACE statement
| UPDATE statement
}
```

> **注意：**
>
> 查看执行计划，SQL不会真正执行
>
> 在只读的DB上，无法查看写SQL的执行计划

示例：

```
DROP TABLE if exists employees;
CREATE TABLE employees (
id INT key NOT NULL,
fname VARCHAR(30),
lname VARCHAR(30),
hired date,
separated DATE NOT NULL DEFAULT '9999-12-31',
job_code INT,
store_id INT
)
shardkey=id;

MySQL [test]> explain select id,fname,lname from employees where id=20\G;
*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: NULL
partitions: NULL
type: NULL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: NULL
filtered: NULL
Extra: no matching row in const table
info: set_1624363251_3, explain select id,fname,lname from `test`.`employees` where (id = 20)
1 row in set (0.00 sec)
```

执行计划中各字段含义：

- id：执行行顺序，按1,2,3,4...进行排序。在所有组中，id值越大，优先级越高，越先执行。id如果相同，可以认为是一组，从上往下顺序执行
- select_type：select的类型。
- table：输出记录的表，对应行正在访问哪一个表，表名或者别名，可能是临时表或者union合并结果集
- partitions：符合的分区
- type：显示的是访问类型，访问类型表示以何种方式去访问数据，例如全表扫描
- possible_keys：优化器可能使用到的索引
- key：优化器实际选择的索引
- key_len：表示索引中使用的字节数，可以通过key_len计算查询中使用的索引长度
- ref：显示索引的哪一列被使用了，如果可能的话，是一个常数
- rows：优化器预估的记录数量，根据表的统计信息及索引使用情况，大致估算出找出所需记录需要读取的行数
- filtered：该 filtered 列指示将按表条件过滤的表行的估计百分比。 最大值为100，这意味着不会对行进行过滤。 值从100开始减少表示过滤量增加
- Extra：额外的显示选项

- info：网关下推，记录了实际发往的set名称和sql信息，info这个一列信息是分布式实例执行计划特有的

网关下推示例

测试表准备

```
--创建测试表
drop table if exists t1;
create table t1(a int key, b int) shardkey=a;
drop table if exists t2;
create table t2(a int key, b int) shardkey=a;

--集群的结构，包含2个set
```

select 查询的下推

1. 指定了shardkey的单表查询。根据shardkey的哈希值计算出目标set，然后将查询直接下推给目标set执行。

```
-- info字段展示了发送的目标set，以及下推的查询
MySQL [test]> explain select * From t1 where a=1\G;
*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: NULL
partitions: NULL
type: NULL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: NULL
filtered: NULL
Extra: no matching row in const table
info: set_1624363222_1, explain select * from `test`.`t1` where (a = 1)
1 row in set (0.00 sec)
```

2. 未指定shardkey的单表查询。将查询广播给所有目标set执行。

```
-- 广播给两个set执行，因此返回了两条记录，其中info字段展示了发送的目标set，以及下推的查询
MySQL [test]> explain select * From t1 where b=1\G;
*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: t1
partitions: p0,p1,p2,p3,p4,p5,p6,p7
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
filtered: 100.00
Extra: Using where
info: set_1624363222_1, explain select * from `test`.`t1` where (b = 1)
*************************** 2. row ***************************
```

```
id: 1
select_type: SIMPLE
table: t1
partitions: p8,p9,p10,p11,p12,p13,p14,p15
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
filtered: 100.00
Extra: Using where
info: set_1624363251_3, explain select * from `test`.`t1` where (b = 1)
2 rows in set (0.01 sec)
```

3. 多表连接查询。当shardkey相等时，将查询直接下推给db执行。

```
-- 广播给两个set执行，因此返回了两条记录，其中info字段展示了发送的目标set，以及下推的查询
-- shardkey相等，但shardkey未指定明确的值，因此广播给所有set执行。
MySQL [test]> explain select * from t1, t2 where t1.a=t2.a;

-- shardkey相等，且shardkey指定了明确的值，因此shardkey的哈希值转给目标set执行。
MySQL [test]> explain select * from t1, t2 where t1.a=t2.a and t1.a=1;

-- shardkey相等，且shardkey指定了多个明确的值，因此shardkey的哈希值转给多个目标set执行。
MySQL [test]> explain select * from t1, t2 where t1.a=t2.a and t1.a in (1,2,3);

-- shardkey相等，且shardkey指定了多个明确的值，但当前网关在计算shardkey的值时会忽略'or'谓词，因此将广播给所有set执行。
MySQL [test]> explain select * from t1, t2 where t1.a=t2.a and (t1.a=1 or t1.a=2);
```

4. 常用聚合函数，包括sum、count、avg、max以及min的下推。

```
-- 网关将查询广播给所有set，并对set返回的聚合结果进行累加
MySQL [test]> explain select count(1) from t1;

-- 网关将avg转换为sum、count，并广播给所有set执行，再根据set返回的sum、count值计算出全局的avg
MySQL [test]> explain select avg(a) from t1;

-- 多表连接时，表的shardkey相等，网关将查询广播给所有set执行，并对set返回的聚合结果进行累加
MySQL [test]> explain select sum(t1.a) from t1, t2 where t1.a=t2.a;

-- 多表连接时，表的shardkey相等，且shardkey指定了明确的值，网关将查询转发给目标set执行
MySQL [test]> explain select sum(t1.a) from t1, t2 where t1.a=t2.a and t1.a=1;

-- 网关将查询广播给所有set执行，再对set返回的结果进行归并排序，计算出每个分组的全局sum值
MySQL [test]> explain select sum(a) from t1 group by b\G;
*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: t1
partitions: p8,p9,p10,p11,p12,p13,p14,p15
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
```

```
filtered: 100.00
Extra: Using temporary; Using filesort
info: set_1624363251_3, explain select sum(a),b, COLLATION(b) from `test`.`t1` group by b order by b
*************************** 2. row ***************************
id: 1
select_type: SIMPLE
table: t1
partitions: p0,p1,p2,p3,p4,p5,p6,p7
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
filtered: 100.00
Extra: Using temporary; Using filesort
info: set_1624363222_1, explain select sum(a),b, COLLATION(b) from `test`.`t1` group by b order by b
2 rows in set (0.00 sec)
```

5. distinct的下推。

```
-- 将distinct下推给set执行，同时额外追加order by操作。网关对set返回的有序元组进行归并排序和去重，从而得到全局去重的结果。
MySQL [test]> explain select distinct b from t1\G;
*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: t1
partitions: p8,p9,p10,p11,p12,p13,p14,p15
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
filtered: 100.00
Extra: Using temporary; Using filesort
info: set_1624363251_3, explain select distinct b from `test`.`t1` order by b
*************************** 2. row ***************************
id: 1
select_type: SIMPLE
table: t1
partitions: p0,p1,p2,p3,p4,p5,p6,p7
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
filtered: 100.00
Extra: Using temporary; Using filesort
info: set_1624363222_1, explain select distinct b from `test`.`t1` order by b
2 rows in set (0.00 sec)
```

6. 子查询的下推。

```
-- 通过等值传递，能够推断出父查询和子查询中表的shardkey相等时，则网关将查询下推给db执行。
-- 注意：由于实现方式的不同，部分查询的explain的结果为json的形式。其中DBQuery字段描述了下推到db执行的查询。
```

```
-- IN子查询
MySQL [test]> explain select * from t1 where t1.a in (select a from t2)\G;
*************************** 1. row ***************************
trace: [
{
"ProxyDeduplicate " : "false",
"Query" : "set_1624363222_1 set_1624363251_3 , Select `t1`.`a`, `t1`.`b` from `test`.`t1` where (`test`.`t1`.`a`) in (select `test`.`t2`.`a` f
rom `test`.`t2`)",
"QueryMode" : "Hash"
}
]

-- EXISTS 子查询
MySQL [test]> explain select * from t1 where exists (select * From t2 where t1.a=t2.a)\G;
*************************** 1. row ***************************
trace: [
{
"ProxyDeduplicate " : "false",
"Query" : "set_1624363222_1 set_1624363251_3 , Select `t1`.`a`, `t1`.`b` from `test`.`t1` where exists(select 1 from `test`.`t2` where
(`test`.`t1`.`a` = `test`.`t2`.`a`))",
"QueryMode" : "Hash"
}
]

-- 通过等值传递，能够推断出父查询和子查询中表的shardkey相等时，则网关将查询下推给db执行。
MySQL [test]> explain select * from t1 where t1.a in (select b from t2 where t2.a=t2.b)\G;
*************************** 1. row ***************************
trace: [
{
"ProxyDeduplicate " : "false",
"Query" : "set_1624363222_1 set_1624363251_3 , Select `t1`.`a`, `t1`.`b` from `test`.`t1` where (`test`.`t1`.`a`) in (select `test`.`t2`.`b`
from `test`.`t2` where (`test`.`t2`.`a` = `test`.`t2`.`b`))",
"QueryMode" : "Hash"
}
]
```

7. distinct聚合函数的下推，例如count(distinct 表达式)、sum(distinct 表达式)等。

```
-- 不存在分组(group by)和排序(order by)操作时，网关只下推distinct查询给所有set执行.
-- 网关对set返回的结果再次去重，从而计算count(distinct b)的值。
MySQL [test]> explain select count(distinct b) from t1 \G
*************************** 1. row ***************************
trace: [
{
"AggFunc " : "count(distinct `test`.`t1`.`b`)",
"ProxyDeduplicate " : "false",
"Query" : "set_1624363222_1 set_1624363251_3 , Select DISTINCT `t1`.`b` from `test`.`t1` where 1",
"QueryMode" : "Hash"
}
]

-- 当存在分组(group by)操作时，网关下推distinct操作，并在下推的查询中额外添加order by语句。
-- 网关对set返回的有序元组按照'分组列'进行归并排序，并计算每个分组的聚合函数count(distinct b)的值。
MySQL [test]> explain select count(distinct b) from t1 group by a\G
*************************** 1. row ***************************
trace: [
{
"AggFunc " : "count(distinct `test`.`t1`.`b`)",
```

```
"DBGroupColumns " : "`test`.`t1`.`a`",
"DBSortedColumns " : "`test`.`t1`.`a`",
"ProxyDeduplicate " : "false",
"Query" : "set_1624363222_1 set_1624363251_3 , Select DISTINCT `t1`.`a`, `t1`.`b`, `test`.`t1`.`a` from `test`.`t1` where 1 order by
3",
"QueryMode" : "Hash"
}
]

-- 当同时存在分组(group by)以及排序(order by)操作时，网关按照前面的例子先计算出分组聚合操作的结果，再利用临时表对分组聚合的
结果进行排序。
-- 其中PorxyTmptable字段展示了创建的临时表；ProxyQuery展示了需要在临时表上执行的查询。
MySQL [test]> explain select a, count(distinct b) as cnt from t1 group by a order by cnt \G
*************************** 1. row ***************************
trace: [
{
"AggFunc " : "count(distinct `test`.`t1`.`b`)",
"DBGroupColumns " : "`test`.`t1`.`a`",
"DBSortedColumns " : "`test`.`t1`.`a`",
"ProxyDeduplicate " : "false",
"ProxyQuery" : "SELECT f0 ,f1 FROM proxy_tmpdb.tmptbl ORDER BY f1 ",
"ProxySortedColumns " : "count(distinct `test`.`t1`.`b`)",
"ProxyTmptable" : "CREATE TEMPORARY TABLE proxy_tmpdb.tmptbl (f0 int(11),f1 bigint)",
"Query" : "set_1624363222_1 set_1624363251_3 , Select DISTINCT `t1`.`a`, `t1`.`b`, `test`.`t1`.`a` from `test`.`t1` where 1 order by
3",
"QueryMode" : "Hash"
}
]
```

Delete/update的下推

8. 指定了shardkey值的单表查询。

```
-- 网关根据shardkey的值计算出目标set，并将查询直接下推给目标set。
-- 注意：info字段展示了目标set以及下推的查询语句
MySQL [test]> explain delete from t1 where a=1\G
*************************** 1. row ***************************
id: 1
select_type: DELETE
table: t1
partitions: p1
type: range
possible_keys: PRIMARY
key: PRIMARY
key_len: 4
ref: const
rows: 1
filtered: 100.00
Extra: Using where
info: set_1624363222_1, explain delete from `test`.`t1` where (a = 1)

MySQL [test]> explain update t1 set b=1 where a=1\G
*************************** 1. row ***************************
id: 1
select_type: UPDATE
table: t1
partitions: p1
type: range
possible_keys: PRIMARY
```

```
key: PRIMARY
key_len: 4
ref: const
rows: 1
filtered: 100.00
Extra: Using where
info: set_1624363222_1, explain update `test`.`t1` SET b=1 where (a = 1)
```

9. 没有指定shardkey值的单表查询。

```
-- 将查询广播给所有set。
-- 注意：info字段展示了目标set以及下推的查询语句
MySQL [test]> explain delete from t1 where 1\G
*************************** 1. row ***************************
id: 1
select_type: DELETE
table: t1
partitions: p0,p1,p2,p3,p4,p5,p6,p7
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
filtered: 100.00
Extra: NULL
info: set_1624363222_1, explain delete from `test`.`t1` where 1
*************************** 2. row ***************************
id: 1
select_type: DELETE
table: t1
partitions: p8,p9,p10,p11,p12,p13,p14,p15
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1
filtered: 100.00
Extra: NULL
info: set_1624363251_3, explain delete from `test`.`t1` where 1

MySQL [test]> explain update t1 set b=1 where 1\G
*************************** 1. row ***************************
id: 1
select_type: UPDATE
table: t1
partitions: p0,p1,p2,p3,p4,p5,p6,p7
type: index
possible_keys: NULL
key: PRIMARY
key_len: 4
ref: NULL
rows: 1
filtered: 100.00
Extra: NULL
info: set_1624363222_1, explain update `test`.`t1` SET b=1 where 1
*************************** 2. row ***************************
```

```
id: 1
select_type: UPDATE
table: t1
partitions: p8,p9,p10,p11,p12,p13,p14,p15
type: index
possible_keys: NULL
key: PRIMARY
key_len: 4
ref: NULL
rows: 1
filtered: 100.00
Extra: NULL
info: set_1624363251_3, explain update `test`.`t1` SET b=1 where 1
```

0. 多表更新操作，且表的shardkey相等。

```
-- 多表更新操作，且shardkey相等时，网关将查询直接下推给后端set执行。
-- 如果shardkey为一个明确的值，则根据shardkey的值计算出目标set；否则，将查询广播给所有set执行。
MySQL [test]> explain update t1, t2 set t1.b=t2.b where t1.a=t2.a and t1.a=202\G
*************************** 1. row ***************************
id: 1
select_type: UPDATE
table: NULL
partitions: NULL
type: NULL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: NULL
filtered: NULL
Extra: no matching row in const table
info: set_1624363222_1, explain update `test`.`t1` join `test`.`t2` SET t1.b=t2.b where ((t1.a = t2.a) and (t1.a = 202))
```

1. 多表更新操作，且表的shardkey不相等，或者包含子查询。网关将构建与更新操作对应的select查询，计算出被更新行的主键、被更新列的新值，再构建相应的update语句发送给set执行。因此，其下推策略同select查询

```
-- 对于如下查询，网关将构建与之对应的select查询：
-- select * from t1, t2 where t1.a=1 and t2.a=21 and t1.b != t2.b group by t1.a;
-- 然后再根据执行的结果，为t1中需要被更新的每个元组构建如下查询给set执行：
-- update t1 set t1.b=... where t1.a= ...
MySQL [test]> explain update t1, t2 set t1.b=t2.b where t1.a=1 and t2.a=21\G
*************************** 1. row ***************************
trace: [
{
"optype" : "TableRename",
"t1" : "T2e(a,b)",
"t2" : "T6(a,b)",
"timecost" : "0.031000"
},
{
"0.OpType " : "Load table",
"1.TableName " : "T6",
"2.PushedDownCond " : "( /*filter*/((`test`.`t2`.`a`=21)))",
"3.NumOfRows " : "0",
"4.AddedCond " : "`test`.`t2`.`a` is null",
"Query" : "set_1624363251_3 , select `a`,`b` from `test`.`t2` t2 where ( /*filter*/((`test`.`t2`.`a`=21))) limit 1000",
```

```
"QueryMode" : "Hash",
"timecost" : "0.640000"
},
{
"0.OpType " : "Load table",
"1.TableName " : "T2e",
"2.PushedDownCond " : "(0)",
"3.NumOfRows " : "0",
"4.AddedCond " : "`test`.`t1`.`a` is null",
"Query" : "AllSets , select `a`,`b` from `test`.`t1` t1 where (0) limit 1000",
"QueryMode" : "All",
"timecost" : "0.544000"
},
{
"Query" : " select `test`.`t1`.`a`,`test`.`t1`.`b`,`test`.`t2`.`b` from `test`.`T2e` `t1` join `test`.`T6` `t2` where 0 group by `test`.`t1`.`a` for u
pdate of `test`.`t1` ",
"timecost" : "0.001000"
}
]
```

USE 语句

语法如下：

```
use db_name
```

示例：

```
MySQL [test]> USE db1;
MySQL [test]> SELECT COUNT(*) FROM mytable;
MySQL [test]> USE db2; SELECT COUNT(*) FROM mytable;
```

注释透传

最近更新时间: 2025-02-18 16:02:00

常用语法

注释透传指支持透传 SQL语句到对应的一个或者多个物理分片（Set），并透传到分表键（Shardkey）对应的分片（Set）中的操作方式。

具体语法如下：

```
/*sets:set_1*/
/*sets:set_1,set_2*/（set名字可以通过/*proxy*/show status查询）
/*sets:allsets */
```

> **注意：**
>
> 对于分布式实例，Proxy 会对 SQL进行语法解析，但有比较严格的限制，如果用户想在某个物理分片（set）中执行SQL语句，可以使用该功能

示例：

```
MySQL [test]> DROP TABLE IF EXISTS test1;
Query OK, 0 rows affected (0.08 sec)

MySQL [test]> create table test1 (a int key, b int, c char(20)) shardkey=a;
Query OK, 0 rows affected (1.71 sec)

--加载300行数据到test1表之后：
MySQL [test]> select count(*) from test1;
+----------+
| count(*) |
+----------+
| 300 |
+----------+
1 row in set (0.12 sec)

MySQL [test]> select count(*) from test1;
+----------+
| count(*) |
+----------+
| 300 |
+----------+
1 row in set (0.11 sec)

MySQL [test]> /*sets:allsets */ select count(*) from test1;
+----------+------------------+
| count(*) | info |
+----------+------------------+
| 150 | set_1619374020_1 |
| 150 | set_1619508344_3 |
+----------+------------------+
2 rows in set (0.02 sec)

MySQL [(none)]> /*proxy*/ show status;
+----------------------------+-----------------------------------------+
```

```
| status_name | value |
+-----------------------------+-------------------------------------------+
| cluster | group_1619373877_13 |
| set_1619374020_1:ip | 10.0.0.17:4007;s1@10.0.0.16:4007@1@IDC1@0 |
| set_1619374020_1:alias | s1 |
| set_1619374020_1:hash_range | 0---31 |
| set_1619508344_3:ip | 10.0.0.17:4008;s1@10.0.0.16:4008@1@IDC1@0 |
| set_1619508344_3:alias | s2 |
| set_1619508344_3:hash_range | 32---62 |
| set | set_1619374020_1,set_1619508344_3 |
+-----------------------------+-------------------------------------------+
8 rows in set (0.00 sec)

MySQL [test]> /*sets:set_1619374020_1*/ select count(*) from test1;
+----------+-----------------+
| count(*) | info |
+----------+-----------------+
| 150 | set_1619374020_1 |
+----------+-----------------+
1 row in set (0.04 sec)

MySQL [test]> /*set_1619508344_3*/ select count(*) from test1;
+----------+
| count(*) |
+----------+
| 150 |
+----------+
1 row in set (0.11 sec)


MySQL [test]> delete from test1;
ERROR 913 (HY000): Proxy ERROR:Join internal error: delete query has no where clause

MySQL [test]> /*sets:allsets*/delete from test1;
Query OK, 300 rows affected (0.04 sec)
```

展示一致性读是否开启

> **注意：**
>
> - $port等参数根据实例信息进行配置

以下返回结果成功查到consistent_read参数的值：

```
mysql -h{proxy_ip} -u{user} -p{password} -P{proxy_port} -c
/*proxy*/ show config;
```

统计数据显示

以下返回结果成功查到统计信息：

```
mysql -h{proxy_ip} -u{user} -p{password} -P{proxy_port} -c
/*proxy*/ show statistics;
```

读写分离新增

强制访问主库

赤兔上创建普通用户后，连接网关执行sql语句：
mysql -h{proxy_ip} -u{user} -p{password} -P{proxy_port} -c
/*master*/ select 1;

同机房就近访问

语法解释

/**slave:localslave**/	优先访问同机房从库（随机访问同机房的从库），次之异机房从库（随机访问异机房的从库），否则访问主库
/**slave:localnode**/	优先随机访问同机房主库&从库（随机访问本地所有集合），次之异机房主库& 从库（随机访问异地所有的集合）
/**slave:localslave,slaveonly**/	优先访问同机房从库（随机访问同机房的从库），次之异机房从库（随机访问异机房的从库）否则失败
/**slave:localnode,slaveonly**/	优先访问同机房从库（随机访问同机房的从库），次之异机房从库（随机访问异机房的从库）否则失败
/**slave:localslave,20**/	优先访问同机房从库（随机选择小于20的延迟从库），次之异机房从库（随机选择小于20的延迟的从库），否则访问主库。
/**slave:localnode,20**/	优先随机访问同机房主库&从库（将符合20延迟的从库和主库放一起，随机访问），次之异机房主库&从库（将符合20延迟的从库和主库放一起，随机访问）
/**slave:localslave,slaveonly,20**/	优先访问同机房从库（随机访问小于20的同机房的从库），次之异机房从库（随机小于20的访问异机房的从库）否则失败
/**slave:localnode,slaveonly,20**/	优先访问同机房从库（随机访问小于20的同机房的从库），次之异机房从库（随机小于20的访问异机房的从库）否则失败

示例

赤兔上创建普通用户后，连接网关执行sql语句查询sbtest1表：

```
/*slave:localslave*/ select a,info from sbtest1;
/*slave:localnode*/ select a,b from sbtest1;
/*slave:localslave,slaveonly*/ select a,info from sbtest1;
/*slave:localnode,slaveonly*/ select b,info from sbtest1;
/*slave:localslave,20*/ select info from sbtest1;
/*slave:localnode,20*/ select a from sbtest1;
/*slave:localslave,slaveonly,20*/ select b from sbtest1;
/*slave:localnode,slaveonly,20*/ select a,b,info from sbtest1;
```

预处理

最近更新时间: 2025-02-18 16:02:00

TDSQL 支持预处理协议，使用方式与单机 MySQL 相同，例如：

- PREPARE Syntax
- EXECUTE Syntax

二进制协议的支持：

- COM_STMT_PREPARE
- COM_STMT_EXECUTE

> **注意：**
>
> 目前TDSQL只对Prepare/Execute命令做语法兼容，从性能角度的话，在分布式下建议用户尽量不要使用该种方式，直接使用文本协议。

示例：

```
MySQL [test]> DROP TABLE IF EXISTS test1;
Query OK, 0 rows affected (0.08 sec)

MySQL [test]> create table test1(a int not null primary key,b int) shardkey=a;
Query OK, 0 rows affected (1.71 sec)

MySQL [test]> insert into test1(a,b) values(5,6),(3,4),(1,2);
Query OK, 3 rows affected (0.06 sec)
Records: 3 Duplicates: 0 Warnings: 0

MySQL [test]> select a,b from test1;
+---+------+
| a | b |
+---+------+
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |
+---+------+
3 rows in set (0.02 sec)

mysql> prepare ff from "select a,b from test1 where a=?";
Query OK, 0 rows affected (0.00 sec)
Statement prepared

mysql> set @aa=3;
Query OK, 0 rows affected (0.00 sec)

mysql> execute ff using @aa;
+---+------+
| a | b |
+---+------+
| 3 | 4 |
+---+------+
1 row in set (0.06 sec)
```

全局唯一数字序列

最近更新时间: 2025-02-18 16:02:00

TDSQL支持全局唯一数字序列（auto_increment）的使用；暂时仅保证自增字段全局唯一和递增性，但是不保证单调递增（即按时间顺序的绝对递增性）。

全局唯一数字序列（auto_increment）长 8 字节，最大为 18446744073709551616，因此，您无需担心该值溢出。

> **注意：**
>
> select last_insert_id()命令只能与Shard表和广播表的自增字段一起使用，不支持与Noshard表的使用。

示例：

创建自增字段的表：

```
mysql> DROP TABLE IF EXISTS auto_inc;

mysql> create table auto_inc (a int,b int,c int auto_increment,d int,key auto(c),primary key p(a,d)) shardkey=d;
Query OK, 0 rows affected (0.12 sec)
```

插入自增字段的分表：

```
mysql> insert into auto_inc (a,b,d,c) values(1,2,3,0),(1,2,4,0);
Query OK, 2 rows affected (0.05 sec)
Records: 2 Duplicates: 0 Warnings: 0

MySQL [test]> select * from auto_inc;
+---+------+------+---+
| a | b | c | d |
+---+------+------+---+
| 1 | 2 | 1008 | 4 |
| 1 | 2 | 1007 | 3 |
+---+------+------+---+
2 rows in set (0.00 sec)
```

自增字段的空洞处理：

由于 auto_increment 仅保证自增字段全局唯一和递增性，如果在节点调度切换、重启等过程中，自增长字段中间会出现空洞，例如：

```
MySQL [test]>insert into auto_inc (a,b,d,c) values(11,12,13,0),(21,22,23,0);
Query OK, 2 rows affected (0.00 sec)

MySQL [test]> select * from auto_inc;
+----+------+------+----+
| a | b | c | d |
+----+------+------+----+
| 11 | 12 | 1009 | 13 |
| 21 | 22 | 1010 | 23 |
| 1 | 2 | 1008 | 4 |
| 1 | 2 | 1007 | 3 |
+----+------+------+----+
4 rows in set (0.00 sec)
```

可更改当前值，命令如下：

```
MySQL [test]> alter table auto_inc auto_increment=100;
Query OK, 0 rows affected (0.03 sec)
```

目前不支持通过insert into auto_inc set c=100 语法插入数据，如果用户要指定自增的值，需要使用以下语法：

```
insert into auto_inc (a,b,d,c) values(300,400,100,500);
```

通过select last_insert_id()命令获取自增值，如果用户不指定自增值，可以通过select last_insert_id()命令获取，暂不支持直接从Insert返回包获取，详见如下：

```
MySQL [test]> insert into auto_inc (a,b,d,c) values(5,6,7,8),(11,12,14,19);
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0

MySQL [test]> select * from auto_inc;
+-----+------+------+-----+
| a | b | c | d |
+-----+------+------+-----+
| 11 | 12 | 1009 | 13 |
| 5 | 6 | 8 | 7 |
| 11 | 12 | 19 | 14 |
| 300 | 400 | 500 | 100 |
| 21 | 22 | 1010 | 23 |
| 1 | 2 | 1008 | 4 |
| 1 | 2 | 1007 | 3 |
+-----+------+------+-----+
7 rows in set (0.00 sec)

MySQL [test]> select last_insert_id();
+------------------+
| last_insert_id() |
+------------------+
| 1009 |
+------------------+
1 row in set (0.00 sec)
```

sequence

最近更新时间: 2025-02-18 16:02:00

本节主要介绍创建、删除、查询和使用Sequence，以及获取显示Sequence的值。Sequence语法和MariaDB兼容，但是需保证分布式全局递增且数值唯一。

> **注意：**
>
> 目前Sequence为保证分布式全局数值唯一，导致性能较差，主要适用于并发不高的场景。

示例：

```
创建Sequence：
create tdsql_sequence test.seq1 start with 12 tdsql_minvalue 10 maxvalue 50000 tdsql_increment by 5 tdsql_nocycle;
create tdsql_sequence test.seq2 start with 12 tdsql_minvalue 10 maxvalue 50000 tdsql_increment by 1 tdsql_cycle;

查询Sequence：
show create tdsql_sequence test.seq2;

使用Sequence获取下一个数值，语句如下：
select tdsql_nextval(test.seq2);
select next value for test.seq2;

删除Sequence：
drop tdsql_sequence test.seq1;
drop tdsql_sequence test.seq2;

nextval命令可以用在insert语句中。使用如下：
MySQL [test]> DROP TABLE IF EXISTS test3;
MySQL [test]> create table test3(a int not null primary key,b int,c char(10)) shardkey=a;
MySQL [test]> insert into test3(a,c) values(1,'A');
Query OK, 1 row affected (0.00 sec)
MySQL [test]> insert into test3(a,c) values(40,'records5');
Query OK, 1 row affected (0.00 sec)

MySQL [test]> select a,c from test3;
+----+----------+
| a | c |
+----+----------+
| 1 | A |
| 40 | records5 |
+----+----------+
2 rows in set (0.00 sec)

MySQL [test]> insert into test3(a,c) values(tdsql_nextval(test.seq2),3);
Query OK, 1 row affected (0.01 sec)

Seq2的初始值为12，此次insert的值为12
MySQL [test]> select a,c from test3;
+----+----------+
| a | c |
+----+----------+
| 40 | records5 |
| 1 | A |
| 12 | 3 |
+----+----------+
3 rows in set (0.00 sec)
```

如需获取上一次的值：
```
MySQL [test]> select tdsql_lastval(test.seq2);
+----+
| 12 |
+----+
| 12 |
+----+
1 row in set (0.00 sec)

MySQL [test]> select tdsql_previous value for test.seq2;
+----+
| 12 |
+----+
| 12 |
+----+
1 row in set (0.00 sec)
```

设置下一个序列数值为2000，tdsql_setval内的第三个参数默认为1，表示2000这个值用过了，下一次不包含2000，如果为0，则下一个从2000开始。
```
MySQL [test]> select tdsql_setval(test.seq2,2000,1)
-> ;
+------+
| 2000 |
+------+
| 2000 |
+------+
1 row in set (0.01 sec)
```

设置的值只能比当前数值大，否则将返回数值为0。设置下一个序列数值时，如果比当前数值小，则系统将没有反应，例如：
```
MySQL [test]> select tdsql_nextval(test.seq2);
+------+
| 2001 |
+------+
| 2001 |
+------+
1 row in set (0.01 sec)
```

seq2设置为10，系统返回0
```
MySQL [test]> select tdsql_setval(test.seq2,10);
+---+
| 0 |
+---+
| 0 |
+---+
1 row in set (0.03 sec)
```

如果设置的比当前数值大，成功返回当前设置的值。
```
MySQL [test]> select tdsql_setval(test.seq2,2010);
+------+
| 2010 |
+------+
| 2010 |
+------+
1 row in set (0.02 sec)

MySQL [test]> select tdsql_nextval(test.seq2);
+------+
| 2011 |
```

```
+------+
| 2011 |
+------+
1 row in set (0.01 sec)
```

使用限制

最近更新时间: 2025-02-18 16:02:00

TDSQL分布式实例中所编写的SQL语句中凡是包含shardkey、parition、distributed by等关键字的会交由proxy处理，语句的剩余部分会发送到DB, 按照MYSQL语法执行。所有TDSQL分布式SQL不支持使用DELAYED和LOW_PRIORITY，不支持对于变量的引用和操作，比如 SET @c=1, @d=@c+1; SELECT @c, @d等。具体限制项请参考以下两小节。

TDSQL大类限制

- 不支持自定义函数、事件、表空间
- 不支持触发器、游标
- 不支持外键
- 不支持复合语句，例如：BEGIN END，LOOP，UNION的语句
- 不支持主备同步相关的SQL语言
- 分区需使用TDSQL分区语法建立分区

TDSQL小语法限制

TDSQL分布式实例不支持DDL、DML、管理SQL语言的部分语法，具体限制如下：

- DDL
  - 不支持CREATE TABLE ... SELECT
  - 不支持CREATE/DROP/ALTER SERVER
  - 不支持CREATE/DROP/ALTER LOGFILE GROUP
  - 不支持ALTER对分表键进行改名，但可以修改类型
  - 不支持RENAME
- DML
  - 不支持SELECT INTO OUTFILE/INTO DUMPFILE/INTO var_name
  - 不支持query_expression_options，如：HIGH_PRIORITY/STRAIGHT_JOIN/SQL_SMALL_RESULT/SQL_BIG_RESULT/SQL_BUFFER_RESULT/SQL_CACHE/SQL_NO_CACHE/SQL_CALC_FOUND_ROWS
  - 不支持窗口函数
  - 不支持非SELECT的子查询
  - 不支持不带列名的INSERT/REPLACE
  - 不支持不带WHERE条件的UPDATE/DELETE
  - 不支持LOAD DATA/XML
  - 不支持SQL中使用DELAYED和LOW_PRIORITY
  - 不支持SQL中对于变量的引用和操作，比如 SET @c=1, @d=@c+1; SELECT @c, @d
  - 不支持INDEX_HINT
  - 不支持HANDLER/DO
- 管理SQL语句
  - 不支持ANALYZE/CHECK/CHECKSUM/OPTIMIZE/REPAIR TABLE，需要用透传语法
  - 不支持CACHE INDEX
  - 不支持FLUSH

- 不支持LOAD INDEX INTO CACHE
- 不支持RESET
- 不支持SHUTDOWN
- 不支持SHOW BINARY LOGS/BINLOG EVENTS
- 不支持SHOW WARNINGS/ERRORS和LIMIT/COUNT的组合

REBALANCE与分区

最近更新时间: 2025-02-18 16:02:00

Range/List一级分区表数据重分布

针对range/list表的数据重分布：目前建议通过proxy运行重分布sql的方式完成。

proxy运行rebalance重分布sql

针对range/list一级分区表的重分布语法，支持将range分区或者list枚举范围进行搬迁。共有offline和online两种方式。offline锁表时间较长，使用数据导入导出的方式完成数据重分布。online方式不锁表，但是依赖多源同步组件实现重分布。

> **注意：**
>
> - 重分布不能和添加set（扩容操作）同时进行
> - online 方式需要部署kafka，consumer组件，并开启实例的数据订阅开关
> - 需要有涉及重分布的数据所占用的空间大小。如果全部分区都参与重分布，则是要有两倍空间
> - 仅TXSQL8.0.24内核版本可以使用该rebalance功能。

重分布基本功能

offline重分布分区：

```
alter table t_range TDSQL_DISTRIBUTED (s1 valuesless than(50), s2 values less than(200)) offline;
```

online重分布分区：

```
alter table t_range TDSQL_DISTRIBUTED (s1 values less than(50), s2 values less than(200));
```

查看正在运行的任务信息：

```
/*proxy*/show rebalance_task;
```

查看所有任务信息：

```
/*proxy*/show all rebalance_task;
```

Kill任务：

```
kill rebalance_task 具体任务id;
kill rebalance_task 12;//停止id为12的任务
```

重分布增加分区示例

> **注意：**
>
> - 至少需要具有3个set的分布式实例才可模拟以下示例
> - online 方式需要部署kafka，consumer组件
> - 开启实例的数据订阅开关（在实例详情页面可设置），用于支持online方式的rebalance功能，如下图所示：

示例：

```
--假设test数据库已有分区表t_range和t_list：
create table t_range(a int key, b int) TDSQL_DISTRIBUTED BY RANGE(a) (s1 values less than(100), s2 values less than(200));

create table t_list(a int key, b int) TDSQL_DISTRIBUTED BY LIST(a) (s1 values in('1','10','100'), s2 values in('110','150','200'));

--增加新的range分区s3：
alter table t_range add TDSQL_DISTRIBUTED (s3 values less than (300));

--增加新的list分区s3：
alter table t_list add TDSQL_DISTRIBUTED (s3 values in (40));

--s3增加range分区范围(只能在最后一个set上增加)
alter table t_range add TDSQL_DISTRIBUTED (s3 values less than (600));

--s2增加list分区范围：
alter table t_list add TDSQL_DISTRIBUTED (s2 values in (90));
```

注意：

- 不支持altertable t_range add TDSQL_DISTRIBUTED (s2 values less than (300),s2 values lessthan (400));会报语法错误。s2重复时不能通过语法解析。
- 支持altertable t_range add TDSQL_DISTRIBUTED (s2 values less than (300),s3 values lessthan (400));一次增加多个分区。

重分布删除分区示例

示例：

```
--强制删除
alter table t_range force drop TDSQL_DISTRIBUTED s3;

--删除，但是有数据会提示并停止删除操作
alter table t_list drop TDSQL_DISTRIBUTED s3;

--一次性删除多个分区
alter table t_range drop TDSQL_DISTRIBUTED s2,s3;
```

注意：

- 当全局分区表只剩下一个分区时，会拒绝删除分区的操作。
- 删除分区会清空数据，删除路由，请谨慎操作。
- 当想删除所有分区时，会拒绝：Can'tdelete all partitions。
- force删除一个分区后，会把指定分区上的表清空，但是表结构还在。
- 当有s1<100, s2<200，force drop s1后，s1上数据会清空，路由会转到s2（0-200），此功能同drop partition，往后归属。

Range表支持语法

注意：

- offline和online支持范围一致。下面以offline为例，online方式只需去掉后面的offline标志即可
- 分区表语句中指定的s1、s2、s3...是每个set的别名，基于实现原理，不能自定义，只能按照顺序依次命名为s1、s2、s3...

```
--重分布后路由：s1:0-100 s2:100-200 s3:200-300 s4:300-500
alter table t_range add TDSQL_DISTRIBUTED (s4 values less than (500));

--重分布后路由：s1:0-100 s2:100-200 s3:200-300 s4:300-400 s5:400-500
alter table t_range add TDSQL_DISTRIBUTED (s5 values less than (500));

--重分布后路由：s1:0-100 s4:100-400
alter table t_range drop TDSQL_DISTRIBUTED s2,s3;

--重分布后路由：s1:0-50 s2:50-200 s3:200-300 s4:300-400
alter table t_range TDSQL_DISTRIBUTED (s1 values less than(50), s2 values less than(200)) offline;

--重分布后路由：s1:0-300 s4:300-400
alter table t_range TDSQL_DISTRIBUTED (s1 values less than(300)) offline;

--重分布后路由：s1:0-100 s2:100-300 s4:300-400
alter table t_range TDSQL_DISTRIBUTED (s1 values less than(100), s2 values less than(300)) offline;

--重分布后路由：s1:0-50 s5:50-100 s2:100-200 s3:200-300 s4:300-400
alter table t_range TDSQL_DISTRIBUTED (s1 values less than(50), s5 values less than(100)) offline;

--重分布后路由：s2:0-50 s1:50-200 s3:200-300 s4:300-400
alter table t_range TDSQL_DISTRIBUTED (s2 values less than(50), s1 values less than(200)) offline;

--重分布后路由：s1:0-100 s2:100-200 s3:200-400
alter table t_range TDSQL_DISTRIBUTED (s1 values less than(100), s2 values less than(200),s3 values less than(400)) offline;
```

注意：

- alter table t_range TDSQL_DISTRIBUTED (s1 values less than(150)) offline;//不支持，需要写为s1 values less than(150)，s2 values less than(200)
- alter table t_range TDSQL_DISTRIBUTED (s1 values less than(100), s2 values less than(300),s4 values less than(400)) offline;和 alter table test.t_range TDSQL_DISTRIBUTED (s2 values less than(300),s4 values less than(400)) offline;的区别，前者是表示s2和s3分区重分布，s1不参与重分布；后者是s1，s2，s3重分布到s2上

List表支持语法

示例：

```
--重分布后路由：s1:1 2 3 11 12 s2:4 5 6 s3:7 8 9
alter table t_list add TDSQL_DISTRIBUTED (s1 values in ('11','12'));

--重分布后路由：s1:1 2 3 s2:4 5 6 s3:7 8 9 s4:11 12
alter table t_list add TDSQL_DISTRIBUTED (s4 values in ('11','12'));

--重分布后路由：s3:7 8 9
alter table t_list drop TDSQL_DISTRIBUTED s1,s2;

--重分布后路由：s1:1 2 6 s2:4 5 3 s3:7 8 9
alter table t_list TDSQL_DISTRIBUTED (s1 values in ('1','2','6'),s2 values in ('4','5','3')) offline;

--重分布后路由：s1:1 2 3 4 5 6 s3:7 8 9
alter table t_list TDSQL_DISTRIBUTED (s1 values in ('1','2','3','4','5','6')) offline;

--重分布后路由：s1:1 2 s2:4 5 6 s3:7 8 9 s4:3
alter table t_list TDSQL_DISTRIBUTED (s1 values in ('1','2'), s4 values in ('3')) offline;
```

List表不支持语法

```
--缺少6
alter table t_list TDSQL_DISTRIBUTED (s1 values in ('1','2'),s2 values in ('4','5','3')) offline;

--多出7，写成s1 values in ('1','2''6','7'),s2 values in ('4','5','3'),s3 values in ('8','9')则支持
alter table t_list TDSQL_DISTRIBUTED (s1 values in ('1','2''6','7'),s2 values in ('4','5','3')) offline;
```

Range表的rebalance示例

```
--通过proxy连接数据库，创建数据库和range分区表
create database if not exists rebalance;
use rebalance;
CREATE TABLE `sbtest1` (
`id` int NOT NULL,
`k` int NOT NULL DEFAULT '0',
`c` char(120) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
`pad` char(60) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
PRIMARY KEY (`id`),
KEY `k_1` (`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_bin TDSQL_DISTRIBUTED BY RANGE(id) (s1 values less than ('50
000'),s2 values less than ('100000'));

--给range分区表增加一个分区s3
alter table sbtest1 add TDSQL_DISTRIBUTED (s3 values less than ('150000'));

--执行show create table sbtest1;语句确认range分区表的三个分区范围正确

--往各个分区分别插入数据，查询数据所在分区正确（数据量多少可自行决定，只要不超过分区范围限制）
insert into sbtest1(id,k,c,pad) values (19999,1,'abc','abc');
insert into sbtest1(id,k,c,pad) values (25000,1,'abc','abc');
insert into sbtest1(id,k,c,pad) values (49999,1,'abc','abc');
insert into sbtest1(id,k,c,pad) values (50000,2,'abc','abc');
insert into sbtest1(id,k,c,pad) values (55000,2,'abc','abc');
insert into sbtest1(id,k,c,pad) values (99999,2,'abc','abc');
insert into sbtest1(id,k,c,pad) values (100000,3,'abc','abc');
insert into sbtest1(id,k,c,pad) values (120000,3,'abc','abc');
insert into sbtest1(id,k,c,pad) values (149999,3,'abc','abc');
/*sets:allsets*/select * from sbtest1;

--执行如下sql语句进行offline方式的重分布，将s1分区的[40000,50000)迁到s2分区，s2分区的[70000,100000)迁到s3分区：
alter table sbtest1 TDSQL_DISTRIBUTED (s1 values less than(40000), s2 values less than(70000),s3 values less than(150000)) offli
ne;

--执行show create table sbtest1;语句确认range分区表的三个分区范围重分布正确

--查询offline方式数据重分布结果正确
/*sets:allsets*/select * from sbtest1;

--执行如下sql语句进行online方式的重分布，将s1分区的[20000,40000)迁到s2分区，s3分区的[70000,120000)迁到s2分区：
alter table sbtest1 TDSQL_DISTRIBUTED (s1 values less than(20000), s2 values less than(120000),s3 values less than(150000));

--执行show create table sbtest1;语句确认range分区表的三个分区范围重分布正确

--查询online方式数据重分布结果正确
/*sets:allsets*/select * from sbtest1;
```

List表的rebalance示例

```
--通过proxy连接数据库，创建数据库和list分区表
create database if not exists rebalance;
use rebalance;

CREATE TABLE `t_list` (
`a` int NOT NULL,
`b` int DEFAULT NULL,
PRIMARY KEY (`a`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_bin TDSQL_DISTRIBUTED BY LIST(a) (s1 values in ('1','10','100'),s
2 values in ('110','150','200'));

--给list分区表增加一个分区s3
alter table t_list add TDSQL_DISTRIBUTED (s3 values in ('40','300','180'));

--执行show create table t_list;语句确认list分区表的三个分区范围正确

--往各个分区分别插入数据，查询数据所在分区正确
insert into t_list(a,b) values(1,1),(10,1),(100,1),(110,2),(150,2),(200,2),(40,3),(300,3),(180,3);
/*sets:allsets*/select * from t_list;

--执行如下sql语句进行offline方式的重分布，将s1分区的100迁到s2分区
alter table t_list TDSQL_DISTRIBUTED (s1 values in('1','10'), s2 values in('100','110','150','200')) offline;

--执行show create table t_list;语句确认list分区表的三个分区范围重分布正确

--查询offline方式数据重分布结果正确
/*sets:allsets*/select * from t_list;

--执行如下sql语句进行online方式的重分布，将s2分区的100，150迁到s3分区：
alter table t_list TDSQL_DISTRIBUTED (s1 values in('1','10'), s2 values in('110','200'), s3 values in('40','300','180','100','150'));

--执行show create table t_list;语句确认list分区表的三个分区范围重分布正确

--查询online方式数据重分布结果正确
/*sets:allsets*/select * from t_list;
```

其他功能介绍

1. 任务抢占：所有proxy都会执行select任务中未完成且时间戳停止时间超过租约时间（目前设置为120s）的任务，获得时间戳停止时间time1。然后尝试更新这条记录（带where update_time = time1条件）如果更新成功则代表拿到任务，开始写时间戳和自己的proxyid。如果更新失败则代表未拿到任务。
2. kill任务id：直接写任务表的is_stop=1（重试，300s超时，可配置），然后想应proxy会感知到任务需要停止，执行停止。
3. 任务迁移：当任务时间戳不更新时，时差超过租约则会被任务抢占，抢占到的proxy会重新执行重分布任务。
4. proxy重启继续执行：因为在租约内，所以不会被抢占，重启后会查找proxyid为自身，is_finish=0且还在租约内的任务写入队列执行。
5. 当前任务执行ctrl+c断开连接，会写入一个stop任务，stop任务执行时，会先从任务列表中查找是否存在该任务，如果存在，则表示当前proxy正在运行该任务，直接执行stop影响状态转换即可回滚。如果不存在，表示当前任务已经停止。

二级分区的自增

功能介绍

针对一级Hash二级Range分区自增的情况。

使用场景

使用二级分区的场景下。

Range表自动化创建分区示例

> **注意：**
>
> - 测试前确保网关参数已经打开，默认是开启的，可以通过赤兔页面查看：

```
--创建range二级分区表：
DROP DATABASE IF EXISTS test;
CREATE DATABASE test;
CREATE TABLE test.t1 (
id int primary key,
hired varchar(100)
) shardkey=id partition by range (day(hired))
(
PARTITION p0 VALUES LESS THAN (20211020),
PARTITION p2 VALUES LESS THAN (20211021)
);
--查看当前分区：
MySQL [rebalance]> show create table test.t1\G;
*************************** 1. row ***************************
Table: t1
Create Table: CREATE TABLE `t1` (
`id` int NOT NULL,
`hired` varchar(100) COLLATE utf8_bin DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_bin shardkey=id PARTITION BY RANGE ( day(hired) ) ( PARTITIO
N p0 VALUES LESS THAN (20211020), PARTITION p2 VALUES LESS THAN (20211021))
1 row in set (0.001 sec)

--等待数分钟左右，查看是否自动创建分区：
MySQL [rebalance]> show create table test.t1\G;
*************************** 1. row ***************************
Table: t1
Create Table: CREATE TABLE `t1` (
`id` int NOT NULL,
`hired` varchar(100) COLLATE utf8_bin DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_bin shardkey=id PARTITION BY RANGE ( day(hired) ) ( PARTITIO
N p0 VALUES LESS THAN (20211020), PARTITION p2 VALUES LESS THAN (20211021), PARTITION p_auto_20211029 VALUES LES
S THAN (20211030), PARTITION p_auto_20211030 VALUES LESS THAN (20211031), PARTITION p_auto_20211031 VALUES LESS T
HAN (20211101), PARTITION p_auto_20211101 VALUES LESS THAN (20211102), PARTITION p_auto_20211102 VALUES LESS THA
N (20211103))
1 row in set (0.001 sec)

--在自动化创建的分区插入数据：
```

```
INSERT INTO test.t1(id,hired) VALUES(1,'20211027');
SELECT * from test.t1;
```

LoadData工具使用说明

最近更新时间: 2025-02-18 16:02:00

依赖

依赖库信息

ubuntu系列：apt-get install expat

centos系列：yum install expat expat-devel -y

原理

noshard表和global表对数据文件不切分，只是执行导入sql，区别是对于noshard表只导入一个set，global导入所有set。使用透传语法。

导入时使用多线程，根据定义的块大小对数据文件进行位置选取，线程获取文件块位置后通过多线程执行load data infile导入。

groupshard表、subshard二级分区表和range/list全局分区表都会进行多线程的切分和导入，区别在于切分阶段groupshard表使用每行数据的shardkey进行hash，然后结合每个set的hash范围获得要发往的setname，进而切分为子文件进行通过loaddatainfile导入。subshard二级分区表会在groupshard表的基础上再次细化每个set的数据文件，按照分区范围将发往同一个set的文本再次细化为发往一个set的某一个分区的文件。range/list表则根据分区范围或者list范围定位到要发往的set，再切分为文件通过load data infile发送。

架构设计

主线程根据设定的块大小(假设为64k)先获取64k大小的分块，再往后定位到行结束符，从而获得一个分块range=(begin_pos，end_pos)。由此将一个文件分为多个range存在队列里。再根据设定的线程数生成多个线程，每个线程从队列里取得一块数据的位置，判断表类型如为noshard或者global则直接发送该块数据内的数据，不生成中间文件。如为shard，一级分区表，二级分区，则按行读取该块数据内的数据，根据发往的set/分区的不同将数据切分到子文件中，再通过load data infile发送。线程处理完一个range后，如导入不出现警告或者报错则删除子文件，再处理其他range。如出现警告或者报错，则将报错子文件存于./tmp_error目录下，将警告子文件存放在./tmp_warning目录下，再取其他range继续处理。

功能介绍

- 支持noshard单表，global复制表，groupshard分表，subshard二级分区表以及range/list全局分区表的导入。
- 支持进度条显示，包括实时导入记录数，总进度百分比，实时导入速率，耗时等。
- 支持命令行导入（类似--ip=127.0.0.1）和文件参数导入(配置文件load.ini)
- 支持自定义导入线程，分割块(chunk_size)大小等参数。
- 支持导入部分字段，使用 --fields ="(id,k,@jump,pad)"参数跳过第三列数据只导入3个字段，或者--fields ="(id,k)"导入两个字段。
- 支持参数设定日志等级，当日志等级设置为debug时（--log_level=2）可以查看详细的报错信息，定位到具体的行。报错或者报警告的子文本会报错在./tmp_data目录下。
- 支持后台运行，通过设置--is_daemon=true开启后台模式，开启后可通过信号量(kill -s SIGUSR1 pid)的方式获得实时的导入进度。同时log日志每隔5min打印进度。

- 支持执行前置sql,通过参数--prefix_sqls能在导入前执行sql，对于设置net_read/write_timeout等参数也可以在此后追加,以分号隔开每个sql。目前默认运行"SET unique_checks = 0; SET foreign_key_checks = 0;SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED"
- 支持文件夹导入
- 支持位置参数
- 支持断点续传
- 支持自动重试
- 支持设定出错时是退出线程还是继续执行其他range导入
- 支持参数开启文件列数与字段数检查功能

执行条件

1. 执行load_data的用户需要有bin目录下的创建目录、创建文件的权限。
2. TDSQL实例需要做如下设置（导入结束后需手动恢复）：
   - 调整复制模式为异步复制
   - 开启Set免切设置
   - 尽量调大数据库超时参数net_read_timeout/net_write_timeout/innodb_lock_wait_timeout
   - 尽量调大数据库binlog拦截参数binlog_write_threshold
   - 数据库关闭双1参数sync_binlog/innodb_flush_log_at_trx_commit
   - 数据库开启LOCAL方式导入数据参数local_infile
   - 数据库磁盘空间检查（预计总量占用2～3倍CSV文件空间）
   - 需要预留足够内存

导入指令

导入指令有3种：命令行导入，位置参数导入和配置文件导入（执行./load_data 可以看到help信息。）

```
命令行导入：
./load_data --ip=127.0.0.1 --port=15026 --user=test --password=test --db_table=loaddata.sbtest_shard --file=sbtest_4l.txt --field_enclosed=" " --field_terminated=" " --thread_num=128

位置参数导入（支持mode0/mode1/mode2/mode3/mode4）：
/load_data mode0/mode1 proxy_host proxy_port user password db_table shardkey_index file field_terminate filed_enclosed [split_size] [escaped by]
example:
example:./load_data mode1 10.10.10.10 3336 test test123 shard.table 1 '/tmp/datafile' ' ' ''
mode0: 只拆分数据，不进行数据导入。
mode1：用insert ignore导入的方式，并且skip_error是false，遇见错误直接退出。
mode2：用insert ignore导入的方式，并且skip_error是true，遇见错误不停止。
mode3：用replace 导入方式，并且skip_error是false，遇见错误直接退出。
mode4：用replace 导入方式，并且skip_error是true，遇见错误不停止。

配置文件导入：
./load_data –config=load.ini
其中load.ini格式参考文件load.ini
```

配置参数说明

常用参数:
--help 说明:获取帮助信息
--ip=127.0.0.1 说明:proxy的ip地址
--port=15006 说明:proxy的端口
--user=test 说明:登陆proxy的用户名
--password=test 说明:登陆proxy的秘密
--db_table=test.test 说明:指定需要导入的库名和表名，如test.sbtest
--file=data200M.txt 说明:导入文件的位置，如--file=/data2/load_new/2.txt;当为目录时，表示文件夹导入。
--field_terminated=" " 说明:字段间隔符，如空格(" "),逗号(","),制表符("\t")等
--field_enclosed=" " 说明:字段括起符，如为空(" "),双引号号("\"")等，注：双引号在命令行和配置文件中有区别
--fields_optionally_enclosed=false 说明:是否选择性括住CHAR、VARCHAR和TEXT等字符型字段
--thread_num=1 说明:导入线程数,默认值为1
--chunk_size=10 说明:导入块大小（KB）默认值为与文件大小相关的一个分段函数
--config=load.ini 说明:配置文件导入模式，配置此参数后，其他直接读取配置文件的参数配置。

不常用参数:
--escaped_by="\\" 说明:转义字符，默认值为反双斜线
--lines_terminated="\n" 说明:行间隔符
--prefix_sqls="SET foreign_key_checks" 说明:前置运行sql，默认运行"SET unique_checks = 0", "SET foreign_key_checks = 0","SET S ESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED"这3条sql。注：prefix_sqls在命令行和配置文件中有区别
--replace_duplicates=false 说明:是否开启替换模式，替换已存在的记录
--fields="(id,k,@jump,pad)" 说明: 导入部分字段(id,k,@jump,pad)
--log_level=5 说明:设置日志等级，默认为LOG_INFO，开启debug日志请将等级设置为2
--is_daemon=false 说明:是否开启后台运行模式，默认关闭
--retry=1 说明：设置重试模式。-1表示当导入出现错误时一直重连；0表示不重试；1表示重试一段时间，需结合retry_time一起使用
--retry_time=10 说明：当retry为1时，指定重试时间,单位是小时，默认一个小时
--skip_error=0 说明：是否跳过错误，即当发生错误时，是停止导入（0）还是跳过错误（1）。
--Breakpoint=0 说明：是否开启断点重传
--client_timeout=10 说明，mysql api的超时参数，默认是10s.
--column_check=0 说明：是否开启字段检查功能，注意此功能与导入部分字段功能冲突，无法同时使用

举例说明

假设有表test：
CREATE TABLE `test` (
`a` int(30) NOT NULL,
`b` int(30) DEFAULT NULL,
`c` int(30) NOT NULL,
PRIMARY KEY (`a`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

已有数据文件test1.txt格式如下：
"1","11","111"
"2","22","222"
将test.txt导入test表可以使用下面指令开启2个线程导入:
./load_data --ip=127.0.0.1 --port=15009 --user=test --password=test --db_table=loaddata.test --file=/data1/test1.txt --field_en closed="\"" --field_terminated="," --thread_num=2;

已有数据文件test2.txt格式如下：
"1","aaa","11","111"
"2","bbb","22",,"222"
想要跳过test2.txt的第二列数据，将数据导入表test，可以使用以下指令：

```
./load_data --ip=127.0.0.1 --port=15009 --user=test --password=test --db_table=loaddata.test --file=/data1/test1.txt --field_en
closed="\"" --field_terminated="," --thread_num=2 --fields="(a,@jump,b,c)";

已有数据文件test3.txt格式如下：
"1",,"111"
"2",,"222"
想将第二列的空值导入表中为NULL，可以使用fields参数来实现，具体指令如下：
./load_data --ip=127.0.0.1 --port=15009 --user=test --password=test --db_table=loaddata.test --file=/data1/test3.txt --field_en
closed="\"" --field_terminated="," --thread_num=2 --fields="(a, @b, c) SET b = NULLIF(@b,'')";
```

问题定位

- 问题描述: load data failed, errmsg: ret: -1, errno: 2013: info: query error[gone away agin]

  问题原因:导入连接中断，可以尝试降低线程数和块大小再重新导入。

- 问题描述:"Local_infile is OFF! Input /sets:allsets/set global local_infile='ON';"

  问题原因:导入权限未开启，连接proxy，运行指令/sets:allsets/set global local_infile='ON';或者使用赤兔开启权限。

- 问题描述:配置文件设置参数prefix_sqls="CREATE DATABASE IF NOT EXISTS test"，运行时query sql failed

  问题原因:配置文件中prefix_sqls参数设置时，需要去掉双引号。

日志查看

- 默认情况下，日志等级为LOG_INFO，只会显示info和warning，error信息，想要查看更多信息，可以通过--log_level参数将日志等级设为2，即可查看debug信息。

- load_data会产生两个日志文件：文件名.discard.log和文件名.log

  文件名.discard.log:收集切分时shardkey不符合要求的数据行。比如当括起符为双引号时，如果导入数据中shardkey字段缺少半边引号94",或者"","时，会认为数据格式有问题，将该条数据写入discard.log。如果开启column_check后，多列或者少列的错误行会记录到此日志。

  文件名.log:导入阶段报错或者警告的行会在文件名.log中体现，同时该日志也会有整体导入进度的信息，导入进度每隔5min打印一次，尾部也会打印进度信息。

其他

- Chunk_size的自适应使用分段函数,当用户未设置chunk_size参数时，

  当文件<=10k时，chunk_size=1k

当10k<文件<100M时，chunk_size=(文件/10k)k

当100M<文件<=10G时，chunk_size=(文件/100k)k

当10G<文件时，chunk_size=100M

- 配置文件中，如果括起符为双引号时，参数设置和命令行稍有区别，配置文件为 --field_enclosed="""，区别于命令行的--field_enclosed="""
- prefix_sqls参数的设定在使用配置文件时，需要去掉引号

常见问题

最近更新时间: 2025-02-18 16:02:00

如何选择实例规格?

- 使用 TDSQL 做功能性测试，且对性能没有特别要求：2个分片，每个分片配置为：内存/磁盘：2GB/25GB。
- 业务发展初期，总数据规模较小但增长快的选型：2个分片，每个分片配置为：内存/磁盘：16GB/200GB。
- 业务发展稳定，根据业务实际情况选型：4个分片，每个分片配置等于：当前业务峰值 * 增长率 / 4。 更多关于实例规格，请参见 TDSQL 实例及分片配置。

TDSQL 语法和 MySQL 的兼容和限制有哪些?

TDSQL 目前版本不能通过命令行进行用户权限相关的设置，需要登录 控制台 进行操作。 TDSQL 目前版本暂不支持自定义函数、触发器、外键、子查询等特性。 对 MySQL 的语法兼容详情，请参见本文“ TDSQL开发指南”章节。

分表键有何作用？

使用分表，在执行操作 select 时，最好带上分表键（shardkey），路由将自动跳转到对应分片，效率较高。若没带上亦可执行，但系统将自动全表扫描，效率较低。 使用分表，在执行操作 insert/replace 或 delete/update 时，必须包含 shardkey，否则会拒绝执行操作。执行操作 insert/replace 需要指定 shardkey，指明将数据插入的物理分片位置。执行操作 delete/update 需要指明 shardkey，作为验证，避免误删。

如何选择分表键？

分表键是在水平拆分过程中用于生成拆分规则的数据表字段，必须在建表时指定好。TDSQL 建议分表键尽可能找到数据表中的数据在业务逻辑上的主体，并确定大部分（或核心的）数据库操作都围绕这个主体的数据进行，方可使用该主体对应的字段作为拆分键，进行分表（该分表方案名为 Group-Shard）。如下图所示：

按组分表方案可以确保不同分表的某些关联数据和复杂的业务逻辑运算，可以聚合到一个物理分片内。例如，某电商平台订单表和用户表都是基于用户维度（UserID）拆分，平台就很容易通过联合查询（不会存在跨节点 join，或分布式事务）快速计算某个用户近期产生了多少订单。

一些典型选择拆分键的应用场景如下：

- 面向用户的互联网应用，是围绕用户维度来做各种操作，那么业务逻辑主体就是用户，可使用用户对应的字段作为拆分键。
- 电商应用或 O2O 应用，是围绕卖家/买家维度来进行各种操作，那么业务逻辑主体就是卖家/买家，可使用卖家/买家对应的字段作为拆分键。但请注意，某些情况下几个超大卖家占到绝大多数交易额，会导致某几个分片的负载和压力明显高于其他分片。
- 游戏类的应用，是围绕玩家维度来做各种操作，那么业务逻辑主体就是玩家，可使用玩家对应的字段作为拆分键。
- 物联网方面的应用，则是基于物联信息进行操作，那么业务逻辑主体就是传感器/SIM 卡，可使用传感器、独立设备、SIM 卡的 IMEI 作为对应的字段作为拆分键。
- 税务/工商类/社保的应用，主要是基于纳税人/法定代表人/居民的信息来开展前台业务，那么业务逻辑主体就是纳税人/法定代表人，可使用纳税人/法定代表人对应的字段作为拆分键。 以此类推，其它类型的应用场景，大多也能找到合适的业务逻辑主体作为拆分键的选择。但需要注意在选择分表键时有一定限制，详情参见本文“开发指南-TDSQL开发指南”的“建表”章节。

分表键能否更换？

一旦选择好分区字段（shardkey），就不能轻易更改。若需要修改一个表的分区字段，只能新建一个表。 若需要修改一个分表某一行中的 shardkey 值，需要先 insert 再 delete。直接操作 update 不能修改分区字段的值。

通用参考
强同步性能对比数据

最近更新时间: 2025-02-18 16:02:00

本文提供 TDSQL 分片与开源 MySQL（未经优化）的性能对比，用于做对比参考。

对比测试环境

硬件：CPU 24core、内存128GB、磁盘1.8TB SSD **网络环境：**局域网，平均网络延迟0.80ms **操作系统：**CentOS 7.0 **数据量：**10张表，每张表2180000行，每张表数据量约5.2GB，innodb buffer：30GB **开源版本：**MySQL 5.7.17 社区版（未经优化，开启半同步） **TDSQL 分片版本：**MySQL5.7（开启强同步），默认开启线程池，参数如下：

- thread_pool_max_threads=2000
- thread_pool_oversubscribe = 10
- thread_pool_stall_limit = 50
- thread_handling = 2

对比测试详细数据

1. 数据初始化参数

```
create database caccts ;
./sysbench --num-threads=500 --test=./tests/db/oltp.lua.bak --oltp-table-size=2180000 --oltp-tables-count=10 --oltp-point-selects=1 --oltp-simple-ranges=0 --oltp-sum-ranges=0 --oltp-order-ranges=0 --oltp-index-updates=1 --oltp-non-index-updates=0 --report-interval=1 --mysql-user=xxxxxx --mysql-password=xxxxxx --mysql-host=xxxxxx --mysql-db=caccts --max-time=360000 --max-requests=2000000000 prepare
```

2. 非索引更新（update）

```
./sysbench --num-threads=500 --test=./tests/db/update\_non\_index.lua --oltp-table-size=2180000 --oltp-tables-count=10 --percentile=99 --report-interval=1 --mysql-host=xxxx --mysql-user=xxx --mysql-password=xxx --mysql-db=caccts --max-time=360000 --max-requests=2000000000 --mysql-port=3306 run
```

3. 只读（select）

```
./sysbench --num-threads=500 --test=./tests/db/select.lua --oltp-table-size=2180000 --oltp-tables-count=10 --percentile=99 --report-interval=1 --mysql-host=xxxx --mysql-user=xxx --mysql-password=xxx --mysql-db=caccts --max-time=360000 -- max-requests=2000000000 --mysql-port=3306 run
```

4. 混合测试

```
./sysbench\_orig --num-threads=500 --test=./tests/db/oltp\_new.lua --oltp-read-only=off --oltp-table-size=2180000 --oltp-tables-count=10 --oltp-point-selects=1 --oltp-simple-ranges=0 --oltp-sum-ranges=0 --oltp-order-ranges=0 --oltp-distinct-ranges=0 --oltp-index-updates=1 --oltp-non-index-updates=0 --percentile=99 --report-interval=1 --mysql-host=xxxx -- mysql-user=xxx --mysql-password=xxx --mysql-db=caccts --max-time=360000 --max-requests=2000000000 --mysql-port=3306 run
```

读请求（read）测试结果

并发	版本	qps	平均响应时间（ms）	99%响应时间（ms）
50	开源 MySQL	304585	0.16	0.26
50	TDSQL	330695	0.15	0.24
100	开源 MySQL	407443	0.24	0.48
100	TDSQL	484640	0.2	0.72
200	开源 MySQL	433401	0.57	1
200	TDSQL	498215	0.55	1.22
500	开源 MySQL	428542	1.16	2.42
500	TDSQL	494874	1.01	2.61
1000	开源 MySQL	412775	2.4	6.3
1000	TDSQL	478393	2.08	4.21

写请求（updata）测试结果

并发	版本	qps	平均响应时间（ms）	99%响应时间（ms）
50	开源 MySQL	14816	3.37	4.82
50	TDSQL	28925	1.73	2.55
100	开源 MySQL	25046	3.99	6.91
100	TDSQL	43466	2.3	4
200	开源 MySQL	32690	6.12	10.86
200	TDSQL	54045	3.7	7.27
500	开源 MySQL	37192	13.44	21.1
500	TDSQL	70370	7.25	15.52
1000	开源 MySQL	35447	28.2	40.47
1000	TDSQL	69890	14.35	30.73

混合场景（OLTP 测试）测试结果

并发	版本	qps	平均响应时间（ms）	99%响应时间（ms）
50	开源 MySQL	63806	4.7	7.13
50	TDSQL	162883	1.84	3.45
100	开源 MySQL	102516	5.85	11.4
100	TDSQL	173974	3.58	6.64
200	开源 MySQL	124550	9.64	18.92

并发	版本	qps	平均响应时间（ms）	99%响应时间（ms）
200	TDSQL	208128	5.76	11.9
500	开源 MySQL	125386	23.93	39.68
500	TDSQL	232543	13.58	27.81
1000	开源 MySQL	121765	49.29	80.71
1000	TDSQL	226130	27.76	54.78

联系我们

最近更新时间: 2025-02-18 16:02:00

工单系统

当您遇到运维或技术类产品使用问题时，可以登录腾讯云金融专区官网，根据界面指引提交工单，我们将尽快响应，期待收到您的宝贵意见。 工单相关入口如下：

- 工单提交：提交工单
- 状态查询：工单列表

词汇表

最近更新时间: 2025-02-18 16:02:00

TDSQL

分布式数据库（Distributed MySQL），简称 TDSQL。

DDL

数据库模式定义语言（Data Definition Language），主要的命令有 CREATE、ALTER、DROP 等。

DML

数据库操纵语言（Data Manipulation Language），命令是 SELECT、UPDATE、INSERT、DELETE。

单表

主要用于存储一些无需分片的表。该表的数据全量存在第一个物理分片中，所有该类型的表都放在第一个物理分片中，语法和使用方法和 MySQL 完全一样，您可以把他理解为一个非分布式的表。

分片

由数据库引擎组成的物理逻辑上的实例，由一个主节点、若干备节点、若干异地备份节点组成。

分表

是指那些原有的数据量极大的表，需要切分到多个数据库节点，这样每个物理分片都有一部分数据，所有物理分片构成了完整的数据。

广播表

又名小表广播功能，设置为广播表后，该表的所有操作都将广播到所有物理分片中，每个分片都有该表的全量数据。

节点

承载分片的物理设备。当然，节点可以是物理机，也可以是虚拟机，也可能是一个小集群。

MariaDB

MariaDB 是 MySQL 最重要的分支之一，在扩展功能、存储引擎以及一些新的功能改进方面都强过 MySQL。目前 TDSQL 已支持 MariaDB 10.1 版本（兼容 MySQL 5.6）。

MySQL

MySQL 是最流行的关系型数据库管理系统之一，所使用的 SQL 语言是用于访问数据库的最常用标准化语言。

OLTP

联机事务处理（On-Line Transaction Processing），是传统的关系型数据库的主要应用，主要是基本的、日常的事务处理，例如银行交易。

OLAP

联机分析处理（On-Line Analytical Processing），是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。

Percona

Percona 完全兼容 MySQL 协议，且功能和性能上较 MySQL 有显著的提升。目前 TDSQL 已支持 Percona 5.7版本。

Proxy

TDSQL 通过 Proxy 实现自动分库分表逻辑，管理底层的多个物理数据库实例，对客户端提供唯一的兼容 MySQL 数据库服务端口。

强同步

是一种基于 MySQL 协议的异步多线程强同步复制方案。

全局唯一数字序列

简称 sequence，使用的是 unsigned long 类型，8 个字节长，目前 TDSQL 可以保证该字段全局唯一和有序递增，但不保证连续性。

Shard

一个物理的数据库实例，用户看到的一个逻辑实例由多个物理实例构成。

shardkey

分表键，TDSQL 通过采用分表键求模的方案进行分表。

水平切分

按照某种规则，将一个表的数据分散到多个物理独立的数据库服务器中，形成“独立”的数据库“分片”。多个分片共同组成一个逻辑完整的数据库实例。

API文档
TDSQL MySQL版（dcdb）
版本（2018-04-11）
API概览

最近更新时间: 2025-02-18 17:50:23

API版本

V3

其他接口

接口名称	接口功能
CheckIpStatus	检查私有网络IP是否可用
DescribeAvailableExclusiveGroups	拉取独享资源池信息
DescribeDBEncryptAttributes	查询实例数据加密状态
DescribeDBEngines	获取DB引擎版本列表
DescribeDBSecurityGroups	查询实例安全组信息
DescribeLatestCloudDBAReport	获取最新的性能检测报告
DescribeProjectSecurityGroups	查询项目安全组信息
KillSession	杀死指定会话
ModifyDBEncryptAttributes	修改实例数据加密属性
ModifyDBInstanceSecurityGroups	修改云数据库安全组
ModifyInstanceNetwork	修改实例所属网络
StartSmartDBA	启动性能检测任务

分布式数据库

接口名称	接口功能
ActiveHourDCDBInstance	恢复后付费实例
AssociateSecurityGroups	绑定安全组
CancelDcnJob	取消DCN同步

接口名称	接口功能
CloneAccount	克隆实例账户
CopyAccountPrivileges	复制账号权限
CreateAccount	创建账号
CreateHourDCDBInstance	创建后付费实例
CreateTmpDCDBInstance	回档实例
DeleteAccount	删除账号
DeleteTmpInstance	删除临时实例
DescribeAccountPrivileges	查询账号权限
DescribeAccounts	查询账号列表
DescribeDBCharsets	获取DB字符集信息列表
DescribeDBLogFiles	获取日志列表
DescribeDBParameters	查看数据库参数
DescribeDBSlowLogAnalysis	获取慢查询记录详情
DescribeDBSlowLogs	查询慢查询日志列表
DescribeDBSyncMode	查询同步模式
DescribeDBTmpInstances	获取实例回档生成的临时实例
DescribeDCDBBinlogTime	获取实例回档时可选的时间范围
DescribeDCDBInstanceDetail	获取实例详情
DescribeDCDBInstances	查询实例列表
DescribeDCDBPrice	新购分布式数据库实例询价
DescribeDCDBSaleInfo	查询分布式数据库可售卖地域和可用区信息
DescribeDCDBShards	查询分片信息
DescribeDatabaseObjects	查询数据库对象列表
DescribeDatabaseTable	查询数据库表信息
DescribeDatabases	查询数据库列表
DescribeDcnDetail	获取实例灾备详情
DescribeFlow	查询流程状态
DescribeLogFileRetentionPeriod	查看备份日志备份天数
DescribeShardSpec	查询分布式数据库可售卖分片规格
DescribeUserTasks	拉取用户任务列表

接口名称	接口功能
DescribeZonesArchInfo	可用区Cpu架构查询
DisassociateSecurityGroups	安全组批量解绑云资源
FlushBinlog	切分Binlog
GrantAccountPrivileges	设置账号权限
InitDCDBInstances	初始化实例
IsolateHourDCDBInstance	销毁后付费实例
ModifyAccountDescription	修改数据库账号备注
ModifyDBInstanceName	修改实例名字
ModifyDBParameters	修改数据库参数
ModifyDBSyncMode	修改同步模式
ModifyInstanceRemark	修改实例备注
ModifyInstanceVip	修改实例Vip
ModifyInstanceVport	修改实例VPORT
ModifyLogFileRetentionPeriod	修改备份日志保存天数
ModifyRealServerAccessStrategy	修改就近接入策略
ResetAccountPassword	重置账号密码
RestartDBInstances	重启实例
SwitchRollbackInstance	切换回档实例
UpgradeHourDCDBInstance	升级后付费实例

调用方式
接口签名v1

最近更新时间: 2025-02-18 17:50:23

tcecloud API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。 签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往云API密钥页面申请，否则无法调用云API接口。

1. 申请安全凭证

在第一次使用云API之前，请前往云API密钥页面申请安全凭证。 安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- 用户必须严格保管安全凭证，避免泄露。

申请安全凭证的具体步骤如下：

1. 登录tcecloud管理中心控制台。
2. 前往云API密钥的控制台页面
3. 在云API密钥页面，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

2. 生成签名串

有了安全凭证SecretId 和 SecretKey后，就可以生成签名串了。以下是生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

注意：这里只是示例，请根据用户实际申请的 SecretId 和 SecretKey 进行后续操作！

以云服务器查看实例列表(DescribeInstances)请求为例，当用户调用这一接口时，其请求参数可能如下:

参数名称	中文	参数值
Action	方法名	DescribeInstances
SecretId	密钥Id	AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
InstanceIds.0	待查询的实例ID	ins-09dx96dg
Offset	偏移量	0

参数名称	中文	参数值
Limit	最大允许输出	20
Version	接口版本号	2017-03-12

2.1. 对参数排序

首先对所有请求参数按参数名的字典序（ ASCII 码）升序排序。注意：1）只按参数名进行排序，参数值保持对应即可，不参与比大小；2）按 ASCII 码比大小，如 InstanceIds.2 要排在 InstanceIds.12 后面，不是按字母表，也不是按数值。用户可以借助编程语言中的相关排序函数来实现这一功能，如 php 中的 ksort 函数。上述示例参数的排序结果如下:

```
{
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

2.2. 拼接请求字符串

此步骤生成请求字符串。 将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 Action=DescribeInstances 。 **注意：“参数值”为原始值而非url编码后的值。**

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。 签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.finance.cloud.tencent.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 / 。
4. 请求字符串: 即上一步生成的请求字符串。

签名原文串的拼接规则为: 请求方法 + 请求主机 +请求路径 + ? + 请求字符串

示例的拼接结果为：

```
GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.4. 生成签名串

此步骤生成签名串。 首先使用 HMAC-SHA1 算法对上一步中获得的**签名原文字符串**进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例:

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

使用其它程序设计语言开发时，可用上面示例中的原文进行签名验证，得到的签名串与例子中的一致即可。

3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 EliP9YW3pW28FpsEdkXt/+WcGeI= ，最终得到的签名串请求参数（Signature）为：EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%3d，它将用于生成最终的请求 URL。

注意：如果用户的请求方法是 GET，或者请求方法为 POST 同时 Content-Type 为 application/x-www-form-urlencoded，则发送请求时所有请求参数的值均需要做 URL 编码，参数键和=符号不需要编码。非 ASCII 字符在 URL 编码前需要先以 UTF-8 进行编码。

注意：有些编程语言的 http 库会自动为所有参数进行 urlencode，在这种情况下，就不需要对签名串进行 URL 编码了，否则两次 URL 编码会导致签名失败。

注意：其他参数值也需要进行编码，编码采用 RFC 3986。使用 %XY 对特殊字符例如汉字进行百分比编码，其中"X"和"Y"为十六进制字符（0-9 和大写字母 A-F），使用小写将引发错误。

4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期
AuthFailure.SecretIdNotFound	密钥不存在
AuthFailure.SignatureFailure	签名错误
AuthFailure.TokenFailure	token 错误
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）

5. 签名演示

在实际调用 API 3.0 时，推荐使用配套的tcecloud SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- JavaScript
- .NET

为了更清楚的解释签名过程，下面以实际编程语言为例，将上述的签名过程具体实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK 。

最终输出的 url 可能为： `http://imgcache.finance.cloud.tencent.com:80cvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%3D&Timestamp=1465185768&Version=2017-03-12`

注意：由于示例中的密钥是虚构的，时间戳也不是系统当前时间，因此如果将此 url 在浏览器中打开或者用 curl 等命令调用时会返回鉴权错误：签名过期。为了得到一个可以正常返回的 url ，需要修改示例中的 SecretId 和 SecretKey 为真实的密钥，并使用系统当前时间戳作为 Timestamp 。

注意：在下面的示例中，不同编程语言，甚至同一语言每次执行得到的 url 可能都有所不同，表现为参数的顺序不同，但这并不影响正确性。只要所有参数都在，且签名计算正确即可。

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，即使是旧版的 API ，由于存在细节差异也会导致签名计算错误，请以对应的实际文档为准。

Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TceCloudAPIDemo {
private final static String CHARSET = "UTF-8";

public static String sign(String s, String key, String method) throws Exception {
Mac mac = Mac.getInstance(method);
SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
mac.init(secretKeySpec);
byte[] hash = mac.doFinal(s.getBytes(CHARSET));
return DatatypeConverter.printBase64Binary(hash);
}

public static String getStringToSign(TreeMap<String, Object> params) {
StringBuilder s2s = new StringBuilder("GETcvm.finance.cloud.tencent.com/?");
// 签名时要求对参数进行字典排序，此处用TreeMap保证顺序
for (String k : params.keySet()) {
s2s.append(k).append("=").append(params.get(k).toString()).append("&");
}
return s2s.toString().substring(0, s2s.length() - 1);
```

```
}

public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException {
StringBuilder url = new StringBuilder("http://imgcache.finance.cloud.tencent.com:80cvm.finance.cloud.tencent.com/?");
// 实际请求的url中对参数顺序没有要求
for (String k : params.keySet()) {
// 需要对请求串进行urlencode，由于key都是英文字母，故此处仅对其value进行urlencode
url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).append("&");
}
return url.toString().substring(0, url.length() - 1);
}

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如：params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如： params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // 公共参数
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE", "HmacSHA1")); // 公共参数
System.out.println(getUrl(params));
}
}
```

Python

注意：如果是在 Python 2 环境中运行，需要先安装 requests 依赖包： `pip install requests` 。

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import time

import requests

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

def get_string_to_sign(method, endpoint, params):
s = method + endpoint + "/?"
query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
return s + query_str

def sign_str(key, s, method):
hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
return base64.b64encode(hmac_str)

if __name__ == '__main__':
endpoint = "cvm.finance.cloud.tencent.com"
data = {
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
```

```
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : secret_id,
'Timestamp' : 1465185768, # int(time.time())
'Version': '2017-03-12'
}
s = get_string_to_sign("GET", endpoint, data)
data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
print(data["Signature"])
# 此处会实际调用，成功后可能产生计费
# resp = requests.get("http://imgcache.finance.cloud.tencent.com:80" + endpoint, params=data)
# print(resp.url)
```

接口签名v3

最近更新时间: 2025-02-18 17:50:23

tcecloud API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。 签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往云API密钥页面申请，否则无法调用云API接口。

1. 申请安全凭证

在第一次使用云API之前，请前往云API密钥页面申请安全凭证。 安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- 用户必须严格保管安全凭证，避免泄露。

申请安全凭证的具体步骤如下：

1. 登录tcecloud管理中心控制台。
2. 前往云API密钥的控制台页面
3. 在云API密钥页面，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

2. TC3-HMAC-SHA256 签名方法

注意：对于GET方法，只支持 Content-Type: application/x-www-form-urlencoded 协议格式。对于POST方法，目前支持 Content-Type: application/json 以及 Content-Type: multipart/form-data 两种协议格式，json 格式默认所有业务接口均支持，multipart 格式只有特定业务接口支持，此时该接口不能使用 json 格式调用，参考具体业务接口文档说明。

下面以云服务器查询广州区实例列表作为例子，分步骤介绍签名的计算过程。我们仅用到了查询实例列表的两个参数：Limit 和 Offset，使用 GET 方法调用。

假设用户的 SecretId 和 SecretKey 分别是：AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 和 Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

2.1. 拼接规范请求串

按如下格式拼接规范请求串（CanonicalRequest）：

```
CanonicalRequest =
HTTPRequestMethod + '\n' +
CanonicalURI + '\n' +
CanonicalQueryString + '\n' +
CanonicalHeaders + '\n' +
SignedHeaders + '\n' +
HashedRequestPayload
```

- HTTPRequestMethod：HTTP 请求方法（GET、POST ），本示例中为 GET；
- CanonicalURI：URI 参数，API 3.0 固定为正斜杠（/）；
- CanonicalQueryString：发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，本示例取值为：Limit=10&Offset=0。注意：CanonicalQueryString 需要经过 URL 编码。

- CanonicalHeaders：参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入自定义的头部参与签名以提高自身请求的唯一性和安全性。拼接规则：1）头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接；2）多个头部，按照头部 key（小写）的字典排序进行拼接。此例中为： `content-type:application/x-www-form-urlencoded\nhost:cvm.finance.cloud.tencent.com\n`
- SignedHeaders：参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。拼接规则：1）头部 key 统一转成小写；2）多个头部 key（小写）按照字典排序进行拼接，并且以分号（;）分隔。此例中为： `content-type;host`
- HashedRequestPayload：请求正文的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，对 HTTP 请求整个正文 payload 做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。注意：对于 GET 请求，RequestPayload 固定为空字符串，对于 POST 请求，RequestPayload 即为 HTTP 请求正文 payload。

根据以上规则，示例中得到的规范请求串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
GET
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.finance.cloud.tencent.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

2.2. 拼接待签名字符串

按如下格式拼接待签名字符串：

```
StringToSign =
Algorithm + \n +
RequestTimestamp + \n +
CredentialScope + \n +
HashedCanonicalRequest
```

- Algorithm：签名算法，目前固定为 TC3-HMAC-SHA256；
- RequestTimestamp：请求时间戳，即请求头部的 X-TC-Timestamp 取值，如上示例请求为 1539084154；
- CredentialScope：凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调用的产品域名一致，例如 cvm。如上示例请求，取值为 2018-10-09/cvm/tc3_request；
- HashedCanonicalRequest：前述步骤拼接所得规范请求串的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。

注意：

> 1. Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。
> 2. Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能导致运行一段时间后，请求必定失败（返回签名过期错误）。

根据以上规则，示例中得到的待签名字符串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
TC3-HMAC-SHA256
1539084154
```

```
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

2.3. 计算签名

1）计算派生签名密钥，伪代码如下

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

- SecretKey：原始的 SecretKey；
- Date：即 Credential 中的 Date 字段信息，如上示例，为2018-10-09；
- Service：即 Credential 中的 Service 字段信息，如上示例，为 cvm；

2）计算签名，伪代码如下

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

- SecretSigning：即以上计算得到的派生签名密钥；
- StringToSign：即步骤2计算得到的待签名字符串；

2.4. 拼接 Authorization

按如下格式拼接 Authorization：

```
Authorization =
Algorithm + ' ' +
'Credential=' + SecretId + '/' + CredentialScope + ', ' +
'SignedHeaders=' + SignedHeaders + ', '
'Signature=' + Signature
```

- Algorithm：签名方法，固定为 TC3-HMAC-SHA256；
- SecretId：密钥对中的 SecretId；
- CredentialScope：见上文，凭证范围；
- SignedHeaders：见上文，参与签名的头部信息；
- Signature：签名值

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5da7a
33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

最终完整的调用信息如下：

```
http://imgcache.finance.cloud.tencent.com:80cvm.finance.cloud.tencent.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, Sig
nedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.finance.cloud.tencent.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
```

```
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

3. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

错误代码	错误描述
AuthFailure.SignatureExpire	签名过期
AuthFailure.SecretIdNotFound	密钥不存在
AuthFailure.SignatureFailure	签名错误
AuthFailure.TokenFailure	token 错误
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）

4. 签名演示

Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class TceCloudAPITC3Demo {
private final static String CHARSET = "UTF-8";
private final static String ENDPOINT = "cvm.finance.cloud.tencent.com";
private final static String PATH = "/";
private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencoded";
private final static String CT_JSON = "application/json";
private final static String CT_FORM_DATA = "multipart/form-data";

public static byte[] sign256(byte[] key, String msg) throws Exception {
Mac mac = Mac.getInstance("HmacSHA256");
SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
mac.init(secretKeySpec);
return mac.doFinal(msg.getBytes(CHARSET));
}
```

```
public static void main(String[] args) throws Exception {
String service = "cvm";
String host = "cvm.finance.cloud.tencent.com";
String region = "ap-guangzhou";
String action = "DescribeInstances";
String version = "2017-03-12";
String algorithm = "TC3-HMAC-SHA256";
String timestamp = "1539084154";
//String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
// 注意时区，否则容易出错
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

// ************* 步骤 1：拼接规范请求串 *************
String httpRequestMethod = "GET";
String canonicalUri = "/";
String canonicalQueryString = "Limit=10&Offset=0";
String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host + "\n";
String signedHeaders = "content-type;host";
String hashedRequestPayload = DigestUtils.sha256Hex("");
String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryString + "\n"
+ canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
System.out.println(canonicalRequest);

// ************* 步骤 2：拼接待签名字符串 *************
String credentialScope = date + "/" + service + "/" + "tc3_request";
String hashedCanonicalRequest = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

// ************* 步骤 3：计算签名 *************
byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
byte[] secretService = sign256(secretDate, service);
byte[] secretSigning = sign256(secretService, "tc3_request");
String signature = DatatypeConverter.printHexBinary(sign256(secretSigning, stringToSign)).toLowerCase();
System.out.println(signature);

// ************* 步骤 4：拼接 Authorization *************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);

TreeMap<String, String> headers = new TreeMap<String, String>();
headers.put("Authorization", authorization);
headers.put("Host", host);
headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
headers.put("X-TC-Action", action);
headers.put("X-TC-Timestamp", timestamp);
headers.put("X-TC-Version", version);
headers.put("X-TC-Region", region);
}
}
```

Python

```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# 密钥参数
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.finance.cloud.tencent.com"
endpoint = "http://imgcache.finance.cloud.tencent.com:80" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
payload = ""
if http_request_method == "POST":
canonical_querystring = ""
ct = "json"
payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)


# ************* 步骤 3：计算签名 *************
# 计算签名摘要函数
def sign(key, msg):
return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)
```

```
# ************* 步骤 4：拼接 Authorization *************
authorization = (algorithm + " " +
"Credential=" + secret_id + "/" + credential_scope + ", " +
"SignedHeaders=" + signed_headers + ", " +
"Signature=" + signature)
print(authorization)

# 公共参数添加到请求头部
headers = {
"Authorization": authorization,
"Host": host,
"Content-Type": "application/%s" % ct,
"X-TC-Action": action,
"X-TC-Timestamp": str(timestamp),
"X-TC-Version": version,
"X-TC-Region": region,
}
```

请求结构

最近更新时间: 2025-02-18 17:50:23

1. 服务地址

地域（Region）是指物理的数据中心的地理区域。tcecloud交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 **查询地域列表** 查看完成的地域列表。

2. 通信协议

tcecloud API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

- application/json（推荐），必须使用 TC3-HMAC-SHA256 签名方法。
- application/x-www-form-urlencoded，必须使用 HmacSHA1 或 HmacSHA256 签名方法。
- multipart/form-data（仅部分接口支持），必须使用 TC3-HMAC-SHA256 签名方法。

GET 请求的请求包大小不得超过 32 KB。POST 请求使用签名方法为 HmacSHA1、HmacSHA256 时不得超过 1 MB。POST 请求使用签名方法为 TC3-HMAC-SHA256 时支持 10 MB。

4. 字符编码

均使用UTF-8编码。

返回结果

最近更新时间: 2025-02-18 17:50:23

正确返回结果

以云服务器的接口查看实例状态列表 (DescribeInstancesStatus) 2017-03-12 版本为例，若调用成功，其可能的返回如下为：

```
{
"Response": {
"TotalCount": 0,
"InstanceStatusSet": [],
"RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
}
}
```

- Response 及其内部的 RequestId 是固定的字段，无论请求成功与否，只要 API 处理了，则必定会返回。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系我们，并提供该 ID 来解决问题。
- 除了固定的字段外，其余均为具体接口定义的字段，不同的接口所返回的字段参见接口文档中的定义。此例中的 TotalCount 和 InstanceStatusSet 均为 DescribeInstancesStatus 接口定义的字段，由于调用请求的用户暂时还没有云服务器实例，因此 TotalCount 在此情况下的返回值为 0， InstanceStatusSet 列表为空。

错误返回结果

若调用失败，其返回值示例如下为：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。
- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系我们，并提供该 ID 来解决问题。

公共错误码 (TODO: 重复信息, 是否真的需要?)

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。 Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码，下表列出了公共错误码。

错误码	错误描述
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。

错误码	错误描述
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。
AuthFailure.SignatureExpire	签名过期。
AuthFailure.SignatureFailure	签名错误。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	http(s)请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

公共参数

最近更新时间: 2025-02-18 17:50:23

公共参数是用于标识用户和接口鉴权目的的参数，如非必要，在每个接口单独的接口文档中不再对这些参数进行说明，但每次请求均需要携带这些参数，才能正常发起请求。

签名方法 v3

使用 TC3-HMAC-SHA256 签名方法时，公共参数需要统一放到 HTTP Header 请求头部中，如下：

参数名称	类型	必选	描述
X-TC-Action	String	是	操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
X-TC-Region	String	是	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。
X-TC-Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。
X-TC-Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
Authorization	String	是	HTTP 标准身份认证头部字段，例如： TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 其中， - TC3-HMAC-SHA256：签名方法，目前固定取该值； - Credential：签名凭证，AKIDEXAMPLE 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service为产品名，必须与调用的产品域名一致，例如cvm； - SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部； - Signature：签名摘要。
X-TC-Token	String	否	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。

签名方法 v1

使用 HmacSHA1 和 HmacSHA256 签名方法时，公共参数需要统一放到请求串中，如下

参数名称	类型	必选	描述
Action	String	是	操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。
Region	String	是	地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。

参数名称	类型	必选	描述
Timestamp	Integer	是	当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。
Nonce	Integer	是	随机正整数，与 Timestamp 联合起来，用于防止重放攻击。
SecretId	String	是	在云API密钥上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。
Signature	String	是	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见接口鉴权文档。
Version	String	是	操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。
SignatureMethod	String	否	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。
Token	String	否	临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token 。

地域列表

地域（Region）是指物理的数据中心的地理区域。tcecloud交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 **查询地域列表** 查看完成的地域列表。

其他接口
检查私有网络IP是否可用

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(CheckIpStatus)用于查询指定的私有网络中的虚拟IP是否可用。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-07-01 14:06:13。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CheckIpStatus
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例Id，形如：dcdbt-ow728lmc
VpcId	是	否	String	目标私有网络ID
SubnetId	是	否	String	目标子网ID
Vip	是	否	String	目标虚拟IP
Ipv6Flag	否	否	Int64	**此参数对外不可见。** 是否IPv6，默认0

3. 输出参数

参数名称	类型	描述
Vip	String	透传入参
Status	Int64	Vip可用状态：0 - 可用，300 - 此ip已被子网内其他用户占用， 301 - 此ip不在子网内， 302 - ip格式错误，303 - 该IP为实例当前IP。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InternalError.GetSubnetFailed	
InvalidParameter.InstanceNotFound	
UnauthorizedOperation.PermissionDenied	

拉取独享资源池信息

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeAvailableExclusiveGroups)用于拉取独享资源池信息

默认接口请求频率限制：20次/秒。

接口更新时间：2019-08-13 22:26:08。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeAvailableExclusiveGroups
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)

3. 输出参数

参数名称	类型	描述
TotalCount	Uint64	独享资源池数目
Items	FenceInfoItem	独享资源池信息
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InvalidParameter	

查询实例数据加密状态

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeDBEncryptAttributes)用于查询实例数据加密状态。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-07 20:13:22。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBEncryptAttributes
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例Id，形如：tdsql-ow728lmc。

3. 输出参数

参数名称	类型	描述
EncryptStatus	Int64	是否启用加密，1-已开启；0-未开启。
CipherText	String	DEK密钥
ExpireDate	String	DEK密钥过期日期。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError	

错误码	描述
InternalError.OperateDatabaseFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	
ResourceUnavailable.InstanceStatusAbnormal	
UnsupportedOperation.InvalidOperation	
InternalError.GetCipherTextFailed	

获取DB引擎版本列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

获取DB引擎版本列表

默认接口请求频率限制：20次/秒。

接口更新时间：2020-04-16 21:32:56。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBEngines
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)

3. 输出参数

参数名称	类型	描述
TotalCount	Int64	items总数
Items	DBEngineInfo	DB引擎信息
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.InnerSystemError	

查询实例安全组信息

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDBSecurityGroups）用于查询实例安全组信息

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 11:55:08。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBSecurityGroups
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Product	是	否	String	数据库引擎名称，本接口取值：dcdb。
InstanceId	是	否	String	实例ID。

3. 输出参数

参数名称	类型	描述
Groups	SecurityGroup	安全组详情。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.AddInstanceInfoFailed	
FailedOperation.AssociateSecurityGroupsFailed	
FailedOperation.ClearInstanceInfoFailed	

错误码	描述
FailedOperation.DisassociateSecurityGroupsFailed	
FailedOperation.SetRuleLocationFailed	
FailedOperation.UpdateInstanceInfoFailed	
InternalError.QueryDatabaseFailed	
InternalError.ReadDatabaseFailed	
InternalError.RouteNotFound	
InvalidParameter	
InvalidParameter.InstanceNotFound	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
AuthFailure	

获取最新的性能检测报告

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeLatestCloudDBAReport）用于获取最新的性能检测报告

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 11:40:00。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeLatestCloudDBAReport
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID
ShardId	是	否	String	实例分片ID

3. 输出参数

参数名称	类型	描述
Score	Int64	本次检测的有效分数
ReportName	String	本次检测报告的名称
ReportDetail	String	本次检测报告的详情
FinishTime	Datetime	本次检测报告的完成时间
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述

错误码	描述
InternalError.DbOperationFailed	
InvalidParameter	

查询项目安全组信息

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeProjectSecurityGroups）用于查询项目安全组信息

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 11:58:03。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeProjectSecurityGroups
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Product	是	否	String	数据库引擎名称，本接口取值：dcdb。
ProjectId	是	否	Uint64	项目ID，TCE上恒传0。
Offset	否	否	Uint64	偏移量。
Limit	否	否	Uint64	拉取数量限制。
SearchKey	否	否	String	搜索条件，支持安全组id或者安全组名称。

3. 输出参数

参数名称	类型	描述
Groups	SecurityGroup	安全组详情。
Total	Uint64	符合条件的安全组总数量。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.AddInstanceInfoFailed	
FailedOperation.AssociateSecurityGroupsFailed	
FailedOperation.ClearInstanceInfoFailed	
FailedOperation.DisassociateSecurityGroupsFailed	
FailedOperation.SetRuleLocationFailed	
FailedOperation.UpdateInstanceInfoFailed	
InternalError.GetSecurityGroupDetailFailed	
InternalError.QueryDatabaseFailed	
InternalError.UpdateDatabaseFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
AuthFailure	

杀死指定会话

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（KillSession）用于杀死指定会话。

默认接口请求频率限制：20次/秒。

接口更新时间：2022-11-24 11:57:24。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：KillSession
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID
SessionId	是	否	Array of Int64	会话ID列表
ShardId	否	否	String	分片ID，与ShardSerialId两个需要设置一个
ShardSerialId	否	否	String	分片序列ID，与ShardId两者需要设置一个

3. 输出参数

参数名称	类型	描述
TaskId	Int64	任务ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	

错误码	描述
UnauthorizedOperation.PermissionDenied	
InvalidParameter	
InvalidParameter.InstanceNotFound	

修改实例数据加密属性

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(ModifyDBEncryptAttributes)用于修改实例数据加密。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-07 20:12:55。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyDBEncryptAttributes
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例Id，形如：tdsql-ow728lmc。
EncryptEnabled	是	否	Int64	是否启用数据加密，开启后暂不支持关闭。本接口的可选值为：1-开启数据加密。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError.DbOperationFailed	
InternalError.OperateDatabaseFailed	
InvalidParameter	

错误码	描述
ResourceUnavailable.BadInstanceStatus	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
UnsupportedOperation.InvalidOperation	

修改云数据库安全组

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyDBInstanceSecurityGroups）用于修改云数据库安全组

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 11:56:22。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyDBInstanceSecurityGroups
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Product	是	否	String	数据库引擎名称，本接口取值：dcdb。
SecurityGroupIds	是	否	Array of String	要修改的安全组ID列表，一个或者多个安全组Id组成的数组。
InstanceId	是	否	String	实例ID。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.AddInstanceInfoFailed	
FailedOperation.AssociateSecurityGroupsFailed	
FailedOperation.ClearInstanceInfoFailed	

错误码	描述
FailedOperation.DisassociateSecurityGroupsFailed	
FailedOperation.SetRuleLocationFailed	
FailedOperation.UpdateInstanceInfoFailed	
InternalError.QueryDatabaseFailed	
InternalError.ReadDatabaseFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
AuthFailure	

修改实例所属网络

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyInstanceNetwork）用于修改实例所属网络。

默认接口请求频率限制：20次/秒。

接口更新时间：2021-03-10 15:54:59。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyInstanceNetwork
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID
VpcId	是	否	String	希望转到的VPC网络的VpcId
SubnetId	是	否	String	希望转到的VPC网络的子网ID
Vip	否	否	String	如果需要指定VIP，填上该字段
Vipv6	否	否	String	**此参数对外不可见。** 如果需要指定VIPv6，填上该字段

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务ID，根据此FlowId通过DescribeFlow接口查询任务进行状态
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InternalError.GetSubnetFailed	
InternalError.GetVpcFailed	
InternalError.VpcOperationFailed	
InvalidParameter.VipNotInSubnet	
InvalidParameter.VipUsed	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
InvalidParameter.SubnetUnavailable	
InvalidParameter.InstanceNotFound	

启动性能检测任务

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（StartSmartDBA）用于启动性能检测任务。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 11:39:39。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：StartSmartDBA
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID
ShardId	是	否	String	实例的分片ID

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务流程ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InvalidParameter	
ResourceUnavailable.InstanceStatusAbnormal	

错误码	描述
UnauthorizedOperation.PermissionDenied	

分布式数据库
恢复后付费实例

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

解隔离后付费实例

默认接口请求频率限制：20次/秒。

接口更新时间：2020-04-16 21:25:56。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ActiveHourDCDBInstance
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceIds	是	否	Array of String	实例ID，形如dcdbt-87zif6ha

3. 输出参数

参数名称	类型	描述
SuccessInstanceIds	String	恢复成功的实例id列表
FailedInstanceIds	String	恢复失败的实例id列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError.DbOperationFailed	

错误码	描述
InvalidParameter	
InvalidParameter.InstanceNotFound	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	

绑定安全组

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口 (AssociateSecurityGroups) 用于安全组批量绑定云资源。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 11:40:28。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：AssociateSecurityGroups
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Product	是	否	String	数据库引擎名称：mariadb,cdb,cynosdb,dcdb,redis,mongodb 等。
SecurityGroupId	是	否	String	要绑定的安全组ID，类似sg-efil73jd。
InstanceIds	是	否	Array of String	被绑定的实例ID，类似tdsqlshard-lesecurk，支持指定多个实例。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.AddInstanceInfoFailed	
FailedOperation.AssociateSecurityGroupsFailed	
FailedOperation.ClearInstanceInfoFailed	

错误码	描述
FailedOperation.DisassociateSecurityGroupsFailed	

取消DCN同步

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

取消DCN同步

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 14:26:49。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CancelDcnJob
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	灾备实例ID

3. 输出参数

参数名称	类型	描述
FlowId	Int64	流程ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CreateFlowFailed	
InternalError	
InternalError.OperateDatabaseFailed	
InvalidParameter	

错误码	描述
InvalidParameter.InstanceNotFound	
ResourceUnavailable.InstanceStatusAbnormal	

克隆实例账户

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（CloneAccount）用于克隆实例账户。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-25 10:15:18。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CloneAccount
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID
SrcUser	是	否	String	源用户账户名
SrcHost	是	否	String	源用户HOST
DstUser	是	否	String	目的用户账户名
DstHost	是	否	String	目的用户HOST
DstDesc	否	否	String	目的用户账户描述

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务流程ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InternalError.GetUserListFailed	
InvalidParameter	
InvalidParameterValue.AccountAlreadyExists	
InvalidParameterValue.SuperUserForbidden	
ResourceNotFound.AccountDoesNotExist	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
UnsupportedOperation.InvalidOperation	

复制账号权限

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（CopyAccountPrivileges）用于复制云数据库账号的权限。 注意：相同用户名，不同Host是不同的账号，Readonly属性相同的账号之间才能复制权限。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-09-27 22:17:11。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CopyAccountPrivileges
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow728lmc。
SrcUserName	是	否	String	源用户名
SrcHost	是	否	String	源用户允许的访问 host
SrcReadOnly	否	否	String	源账号的 ReadOnly 属性
DstUserName	是	否	String	目的用户名
DstHost	是	否	String	目的用户允许的访问 host
DstReadOnly	否	否	String	目的账号的 ReadOnly 属性

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CopyRightError	
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InvalidParameter.GenericParameterError	
InvalidParameterValue.BadUserType	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

创建账号

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（CreateAccount）用于创建云数据库账号。一个实例可以创建多个不同的账号，相同的用户名+不同的host是不同的账号。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-01-16 20:48:59。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateAccount
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow728lmc，可以通过 DescribeDCDBInstances 查询实例详情获得。
UserName	是	否	String	AccountName
Host	是	否	String	可以登录的主机，与mysql 账号的 host 格式一致，可以支持通配符，例如 %，10.%，10.20.%。
Password	是	否	String	账号密码，由字母、数字或常见符号组成，不能包含分号、单引号和双引号，长度为6~32位。
ReadOnly	否	否	Int64	是否创建为只读账号，0：否， 1：该账号的sql请求优先选择备机执行，备机不可用时选择主机执行，2：优先选择备机执行，备机不可用时操作失败，3：只从备机读取。
Description	否	否	String	账号备注，可以包含中文、英文字符、常见符号和数字，长度为0~256字符
DelayThresh	否	否	Int64	如果备机延迟超过本参数设置值，系统将认为备机发生故障 建议该参数值大于10。当ReadOnly选择1、2时该参数生效。

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例ID，透传入参。
UserName	String	用户名，透传入参。

参数名称	类型	描述
Host	String	允许访问的 host，透传入参。
ReadOnly	Int64	透传入参。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CreateUserFailed	
FailedOperation.OssOperationFailed	
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetUserListFailed	
InvalidParameter.CharacterError	
InvalidParameter.GenericParameterError	
InvalidParameterValue.AccountAlreadyExists	
InvalidParameterValue.SuperUserForbidden	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
UnsupportedOperation.OldProxyVersion	

创建后付费实例

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

创建后付费实例

默认接口请求频率限制：20次/秒。

接口更新时间：2022-08-08 16:14:22。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateHourDCDBInstance
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
VpcId	是	否	String	虚拟私有网络 ID
SubnetId	是	否	String	虚拟私有网络子网 ID
ShardMemory	是	否	Int64	分片内存大小，单位：GB，可以通过 DescribeShardSpec 查询实例规格获得。
ShardStorage	是	否	Int64	分片存储空间大小，单位：GB，可以通过 DescribeShardSpec 查询实例规格获得。
ShardNodeCount	是	否	Int64	单个分片节点个数，可以通过 DescribeShardSpec 查询实例规格获得。
ShardCount	是	否	Int64	实例分片个数，可选范围2-8，可以通过升级实例进行新增分片到最多64个分片。
DbVersionId	否	否	String	数据库引擎版本，调用 DescribeDBEngines 查询可用的版本。
Zones	是	否	Array of String	分片节点可用区分布，最多可填两个可用区。当分片规格为一主两从时，其中两个节点在第一个可用区。
Count	否	否	Int64	欲购买实例的数量
ShardCpu	否	否	Int64	分片cpu大小，单位：核，可以通过 DescribeShardSpec 查询实例规格获得。
SecurityGroupId	否	否	String	安全组id

参数名称	必选	允许NULL	类型	描述
InstanceName	否	否	String	实例名称， 可以通过该字段自主的设置实例的名字
ExclusterId	否	否	String	资源池 Id，TCE 专用
Ipv6Flag	否	否	Int64	**此参数对外不可见。** 是否支持IPv6
ResourceTags	否	否	Array of ResourceTag	标签键值对数组
DcnRegion	否	否	String	**此参数对外不可见。** DCN源地域
DcnInstanceId	否	否	String	**此参数对外不可见。** DCN源实例ID
PlatformProjectId	否	否	String	平台项目ID
CpuArch	否	否	String	cpu架构
InitParams	否	否	Array of DBParamValue	参数列表。本接口的可选值为：character_set_server（字符集，必传），lower_case_table_names（表名大小写敏感，必传，0 - 敏感；1-不敏感），innodb_page_size（innodb数据页，默认16K），sync_mode（同步模式：0 - 异步； 1 - 强同步；2 - 强同步可退化。默认为强同步可退化）。
RsAccessStrategy	否	否	Int64	RS访问策略, 0-无策略, 1-可用区就近访问

3. 输出参数

参数名称	类型	描述
InstanceIds	String	订单对应的实例 ID 列表，如果此处没有返回实例 ID，可以通过订单查询接口获取。还可通过实例查询接口查询实例是否创建完成。
FlowId	Int64	流程id，可以根据流程id查询创建进度
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CreateOrderFailed	
FailedOperation.TagDryRunError	
FailedOperation.UserNotAuthed	
InternalError.DbOperationFailed	

错误码	描述
InternalError.GetSubnetFailed	
InternalError.GetVpcFailed	
InvalidParameter	
InvalidParameter.SpecNotFound	
InvalidParameter.SubnetNotFound	
InvalidParameterValue.IllegalCount	
InvalidParameterValue.IllegalNodeCount	
InvalidParameterValue.IllegalZone	
InvalidParameterValue.SpecIdIllegal	
InvalidParameter.VpcNotFound	
UnauthorizedOperation.PermissionDenied	

回档实例

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

回档实例

默认接口请求频率限制：20次/秒。

接口更新时间：2020-01-16 14:42:22。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：CreateTmpDCDBInstance
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	回档实例的ID
RollbackTime	是	否	String	回档时间点

3. 输出参数

参数名称	类型	描述
FlowId	Int64	任务流ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError	
InternalError.DbOperationFailed	
InternalError.InnerSystemError	

错误码	描述
InternalError.RetreateTime	
InvalidParameter	
InvalidParameter.ActionNotFound	
InvalidParameter.InstanceNotFound	
InvalidParameter.NotSupportedPayMode	
InvalidParameter.PermissionDenied	
ResourceInUse.TempInstanceExist	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

删除账号

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DeleteAccount）用于删除云数据库账号。用户名+host唯一确定一个账号。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-09-28 11:11:34。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteAccount
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID，形如：dcdbt-ow728lmc，可以通过 DescribeDCDBInstances 查询实例详情获得。
UserName	是	否	String	用户名
Host	是	否	String	用户允许的访问 host

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.DeleteUserFailed	
InternalError.CamAuthFailed	

错误码	描述
InternalError.DbOperationFailed	
InvalidParameter.GenericParameterError	
InvalidParameterValue.SuperUserForbidden	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

删除临时实例

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DeleteTmpInstance）用于删除临时实例。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-12-27 20:47:15。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DeleteTmpInstance
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：tdsql-ow728lmc。

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务流程ID。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CreateFlowFailed	
InternalError	
InvalidParameter	
InvalidParameter.ActionNotFound	

错误码	描述
InvalidParameter.PermissionDenied	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	

查询账号权限

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeAccountPrivileges）用于查询云数据库账号权限。 注意：注意：相同用户名，不同Host是不同的账号。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-03-05 19:39:39。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeAccountPrivileges
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow7t8lmc。
UserName	是	否	String	登录用户名。
Host	是	否	String	用户允许的访问 host，用户名+host唯一确定一个账号。
DbName	是	否	String	数据库名。如果为 *，表示查询全局权限（即 *.*），此时忽略 Type 和 Object 参数
Type	否	否	String	类型,可以填入 table 、 view 、 proc 、 func 和 *。当 DbName 为具体数据库名，Type为 * 时，表示查询该数据库权限（即db.*），此时忽略 Object 参数
Object	否	否	String	具体的 Type 的名称，例如 Type 为 table 时就是具体的表名。DbName 和 Type 都为具体名称，则 Object 表示具体对象名，不能为 * 或者为空
ColName	否	否	String	当 Type=table 时，ColName 为 * 表示查询表的权限，如果为具体字段名，表示查询对应字段的权限

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例ID
Privileges	String	权限列表。
UserName	String	数据库账号用户名

参数名称	类型	描述
Host	String	数据库账号Host
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetRightFailed	
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

查询账号列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeAccounts）用于查询指定云数据库实例的账号列表。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-09-28 11:12:24。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeAccounts
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID，形如：dcdbt-ow728lmc。

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例ID，透传入参。
Users	DBAccount	实例用户列表。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetUserListFailed	

错误码	描述
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

获取DB字符集信息列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

获取DB字符集信息列表

默认接口请求频率限制：20次/秒。

接口更新时间：2021-09-13 20:41:32。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBCharsets
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)

3. 输出参数

参数名称	类型	描述
TotalCount	Int64	DBCharsets总数
DBCharsetItem	DBCharsetItem	DB字符集信息
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.InnerSystemError	
InvalidParameter.ActionNotFound	

获取日志列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeDBLogFiles)用于获取数据库的各种日志列表，包括冷备、binlog、errlog和slowlog。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-03-29 14:41:29。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBLogFiles
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow7t8lmc。
ShardId	是	否	String	分片 ID，形如：shard-7noic7tv
Type	是	否	Int64	请求日志类型，取值只能为1、2、3或者4。1-binlog，2-冷备，3-errlog，4-slowlog。

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例 ID，形如：dcdbt-ow728lmc。
Type	Uint64	请求日志类型。1-binlog，2-冷备，3-errlog，4-slowlog。
Total	Uint64	请求日志总数
Files	LogFileInfo	日志文件列表
VpcPrefix	String	如果是VPC网络的实例，做用本前缀加上URI为下载地址
NormalPrefix	String	如果是普通网络的实例，做用本前缀加上URI为下载地址
ShardId	String	分片 ID，形如：shard-7noic7tv
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.CosConfiguration	
InternalError.CosSignUrl	
InternalError.DbOperationFailed	
InternalError.GetInstanceInfoFailed	
InvalidParameter.GenericParameterError	
InvalidParameter.InstanceNotFound	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.CosApiFailed	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

查看数据库参数

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeDBParameters)用于获取数据库的当前参数设置。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-11-15 15:28:19。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBParameters
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow7t8lmc。

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例 ID，形如：dcdbt-ow7t8lmc。
Params	ParamDesc	请求DB的当前参数值
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetDbConfigFailed	

错误码	描述
InvalidParameter.GenericParameterError	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

获取慢查询记录详情

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeDBSlowLogAnalysis)用于获取慢查询记录详情。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-04-16 21:31:26。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBSlowLogAnalysis
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如 dcdbt-hw0qj6m1
StartTime	是	否	Datetime	开始时间，形如 2006-01-02 15:04:05
EndTime	否	否	Datetime	结束时间，形如 2006-01-02 15:04:05，不填的话默认是当前时间
Db	是	否	String	要查询的慢查询语句对应的数据库名称
User	是	否	String	要查询的慢查询语句对应的用户名称
CheckSum	是	否	String	要查询的慢查询语句的校验和，可以通过查询慢查询日志列表获得
Slave	否	否	Int64	是否查询从机的慢查询。0-主机；1-从机，默认查询主机慢查询
ShardId	是	否	String	实例分片ID，形如 shard-nbchf68b

3. 输出参数

参数名称	类型	描述
StartTime	Datetime	慢查询SQL出现的开始时间
EndTime	Datetime	慢查询SQL出现的结束时间

参数名称	类型	描述
Data	Int64	返回的慢查询详情数据。每个点代表一个时间段内慢查询SQL出现的记录次数，一个时间段的长度由请求的开始时间和结束时间的差值决定，小于1天是5分钟一段，大于1天小于7天是30分钟一段，大于7天是2个小时一段。
Period	Int64	采样周期
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InternalError.GetInstanceInfoFailed	
InternalError.GetSlowLogFailed	
InternalError.InnerSystemError	
InvalidParameter	
InvalidParameter.IllegalTime	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	

查询慢查询日志列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeDBSlowLogs)用于查询慢查询日志列表。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-06-18 16:47:34。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBSlowLogs
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-hw0qj6m1
Offset	是	否	Uint64	从结果的第几条数据开始返回
Limit	是	否	Uint64	返回的结果条数
StartTime	是	否	Datetime	查询的起始时间，形如2016-07-23 14:55:20
EndTime	否	否	Datetime	查询的结束时间，形如2016-08-22 14:55:20。如果不填，那么查询结束时间就是当前时间
Db	否	否	String	要查询的具体数据库名称
OrderBy	否	否	String	排序指标，取值为query_time_sum或者query_count。不填默认按照query_time_sum排序
OrderByType	否	否	String	排序类型，desc（降序）或者asc（升序）。不填默认desc排序
Slave	否	否	Int64	是否查询从机的慢查询，0-主机; 1-从机。不填默认查询主机慢查询
ShardId	是	否	String	实例的分片ID，形如shard-53ima8ln

3. 输出参数

参数名称	类型	描述

参数名称	类型	描述
LockTimeSum	Float	所有语句锁时间总和
QueryCount	Int64	所有语句查询总次数
Total	Int64	总记录数
QueryTimeSum	Float	所有语句查询时间总和
Data	SlowLogData	慢查询日志数据
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InternalError.GetInstanceInfoFailed	
InternalError.GetSlowLogFailed	
InternalError.InnerSystemError	
InvalidParameter.IllegalTime	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	

查询同步模式

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDBSyncMode）用于查询云数据库实例的同步模式。

默认接口请求频率限制：20次/秒。

接口更新时间：2021-02-25 11:39:01。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBSyncMode
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	待修改同步模式的实例ID。形如：dcdbt-ow728lmc。

3. 输出参数

参数名称	类型	描述
SyncMode	Int64	同步模式：0 异步，1 强同步， 2 强同步可退化
IsModifying	Int64	是否有修改流程在执行中：1 是， 0 否。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	

错误码	描述
InternalError.GetInstanceDetailFailed	
InternalError.InnerSystemError	
InvalidParameter.GenericParameterError	
InvalidParameter.InstanceNotFound	
UnauthorizedOperation.PermissionDenied	

获取实例回档生成的临时实例

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDBTmpInstances）用于获取实例回档生成的临时实例

默认接口请求频率限制：20次/秒。

接口更新时间：2019-12-27 20:46:39。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDBTmpInstances
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID

3. 输出参数

参数名称	类型	描述
TmpInstances	TmpInstance	临时实例列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	
UnauthorizedOperation.PermissionDenied	

获取实例回档时可选的时间范围

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

获取实例回档时可选的时间范围

默认接口请求频率限制：20次/秒。

接口更新时间：2019-12-27 20:45:28。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDCDBBinlogTime
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	需要回档的实例ID

3. 输出参数

参数名称	类型	描述
StartTime	Datetime	开始时间
EndTime	Datetime	结束时间
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.LogNotExisted	
FailedOperation.OssOperationFailed	
InternalError	

错误码	描述
InternalError.DbOperationFailed	
InternalError.GetInstanceInfoFailed	
InvalidParameter	
InvalidParameter.ActionNotFound	
InvalidParameter.PermissionDenied	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceStatusAbnormal	

获取实例详情

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDCDBInstanceDetail）用于获取实例详情

默认接口请求频率限制：20次/秒。

接口更新时间：2022-08-08 16:11:51。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDCDBInstanceDetail
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID，形如dcdbt-7oaxtcb7

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例ID，形如dcdbt-7oaxtcb7
InstanceName	String	实例名称
Status	Int64	实例状态。0-实例创建中；1-异步任务处理中；2-运行中；3-实例未初始化；-1-实例已隔离
StatusDesc	String	实例目前运行状态描述
Vip	String	实例内网IP地址
Vport	Int64	实例内网端口
NodeCount	Int64	实例节点数。值为2时表示一主一从，值为3时表示一主二从
Region	String	实例所在地域名称，形如ap-guangzhou
VpcId	String	实例私有网络ID，形如vpc-r9jr0de3
SubnetId	String	实例私有网络子网ID，形如subnet-6rqs61o2

参数名称	类型	描述
WanStatus	Int64	外网状态，0-未开通；1-已开通；2-关闭；3-开通中；4-关闭中
WanDomain	String	外网访问的域名，公网可解析
WanVip	String	外网IP地址，公网可访问
WanPort	Int64	外网访问端口
ProjectId	Int64	实例所属项目ID
AutoRenewFlag	Int64	实例自动续费标志。0-正常续费；1-自动续费；2-到期不续费
ExclusterId	String	独享集群ID
PayMode	String	付费模式。prepaid-预付费；postpaid-按量计费
CreateTime	Datetime	实例创建时间，格式为 2006-01-02 15:04:05
PeriodEndTime	Datetime	实例到期时间，格式为 2006-01-02 15:04:05
DbVersion	String	数据库版本信息
IsAuditSupported	Int64	实例是否支持审计。0-不支持；1-支持
IsEncryptSupported	Int64	实例是否支持数据加密。0-不支持；1-支持
Machine	String	实例母机机器型号
Memory	Int64	实例内存大小，单位 GB，各个分片的内存大小的和
Storage	Int64	实例磁盘存储大小，单位 GB，各个分片的磁盘大小的和
StorageUsage	Float	实例存储空间使用率，计算方式为：各个分片已经使用的磁盘大小的和/各个分片的磁盘大小的和。
LogStorage	Int64	日志存储空间大小，单位GB
Pid	Int64	产品类型ID
MasterZone	String	主DB可用区
SlaveZones	String	从DB可用区
Shards	ShardBriefInfo	分片信息
Vip6	String	内网IPv6
Cpu	Int64	实例Cpu核数
Qps	Int64	实例QPS
DbEngine	String	DB引擎
Ipv6Flag	Int64	是否支持IPv6
WanVipv6	String	外网IPv6地址，公网可访问
WanStatusIpv6	Int64	外网状态，0-未开通；1-已开通；2-关闭；3-开通中；4-关闭中

参数名称	类型	描述
WanPortIpv6	Int64	外网IPv6端口
ResourceTags	ResourceTag	标签信息
DcnFlag	Int64	DCN标志，0-无，1-主实例，2-灾备实例
DcnStatus	Int64	DCN状态，0-无，1-创建中，2-同步中，3-已断开
DcnDstNum	Int64	DCN灾备实例数
PlatformProjectId	String	平台项目ID
PlatformProjectName	String	平台项目名称
InstanceType	Int64	1： 主实例（独享型），2: 主实例, 3： 灾备实例, 4： 灾备实例（独享型）
CpuArch	String	cpu架构
RsAccessStrategy	Int64	就近访问策略，0-无策略, 1-可用区就近访问
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError.GetSubnetFailed	
InternalError.OperateDatabaseFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	
UnauthorizedOperation.PermissionDenied	

查询实例列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

查询云数据库实例列表，支持通过项目ID、实例ID、内网地址、实例名称等来筛选实例。 如果不指定任何筛选条件，则默认返回10条实例记录，单次请求最多支持返回100条实例记录。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-18 15:09:56。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDCDBInstances
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceIds	否	否	Array of String	按照一个或者多个实例 ID 查询。实例 ID 形如：dcdbt-2t4cf98d
SearchName	否	否	String	搜索的字段名，当前支持的值有：instancename、vip、all。传 instancename 表示按实例名进行搜索；传 vip 表示按内网IP进行搜索；传 all 将会按实例ID、实例名和内网IP进行搜索。
SearchKey	否	否	String	搜索的关键字，支持模糊搜索。多个关键字使用换行符（'\n'）分割。
ProjectIds	否	否	Array of Int64	**此参数对外不可见。** 按项目 ID 查询
IsFilterVpc	否	否	Bool	是否根据 VPC 网络来搜索
VpcId	否	否	String	私有网络 ID， IsFilterVpc 为 1 时有效
SubnetId	否	否	String	私有网络的子网 ID， IsFilterVpc 为 1 时有效
OrderBy	否	否	String	排序字段， projectId， createtime， instancename 三者之一
OrderByType	否	否	String	排序类型， desc 或者 asc
Offset	否	否	Int64	偏移量，默认为 0
Limit	否	否	Int64	返回数量，默认为 10，最大值为 100。

参数名称	必选	允许NULL	类型	描述
ExclusterType	否	否	Int64	1非独享集群，2独享集群， 0全部
IsFilterExcluster	否	否	Bool	标识是否使用ExclusterType字段, false不使用，true使用
ExclusterIds	否	否	Array of String	独享集群ID
TagKeys	否	否	Array of String	按标签key查询
FilterInstanceType	否	否	String	实例类型过滤，1-独享实例，2-主实例，3-灾备实例，多个按逗号分隔

3. 输出参数

参数名称	类型	描述
TotalCount	Int64	符合条件的实例数量
Instances	DCDBInstanceInfo	实例详细信息列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.FenceError	
InternalError.GetSubnetFailed	
InternalError.GetVpcFailed	
InvalidParameter.GenericParameterError	
InvalidParameter.SubnetNotFound	
InvalidParameterValue.IllegalExclusterID	
InvalidParameterValue.SpecIdIllegal	
UnauthorizedOperation.PermissionDenied	

新购分布式数据库实例询价

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDCDBPrice）用于在购买实例前，查询实例的价格。 注意：当前 TCE 上 Paymode 参数需要传 postpaid。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 12:31:27。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDCDBPrice
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Zone	是	否	String	欲新购实例的可用区ID。
Count	是	否	Int64	欲购买实例的数量，目前支持购买1-10个实例
Period	是	否	Int64	欲购买的时长，单位：月。不传时默认查询1个月的价格。付费模式为 postpaid 时该参数无效，固定查询第一计费阶梯使用1小时的费用。
ShardNodeCount	是	否	Int64	单个分片节点个数大小，可以通过 DescribeShardSpec 查询实例规格获得。
ShardMemory	是	否	Int64	分片内存大小，单位：GB，可以通过 DescribeShardSpec 查询实例规格获得。
ShardStorage	是	否	Int64	分片存储空间大小，单位：GB，可以通过 DescribeShardSpec 查询实例规格获得。
ShardCount	是	否	Int64	实例分片个数，可选范围2-8，可以通过升级实例进行新增分片到最多64个分片。
Paymode	否	否	String	付费类型。postpaid：按量付费；prepaid：预付费。不传时默认为预付费（当前TCE暂不支持预付费功能）。

3. 输出参数

参数名称	类型	描述
OriginalPrice	Int64	原价，单位：分

参数名称	类型	描述
Price	Int64	实际价格，单位：分。受折扣等影响，可能和原价不同。计量模式下返回值恒为0。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.QueryPriceFailed	
InvalidParameter.GenericParameterError	
InvalidParameter.SpecNotFound	
InvalidParameterValue.SpecIdIllegal	
UnauthorizedOperation.PermissionDenied	

查询分布式数据库可售卖地域和可用区信息

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeDCDBSaleInfo)用于查询分布式数据库可售卖的地域和可用区信息。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-11-15 15:37:18。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDCDBSaleInfo
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)

3. 输出参数

参数名称	类型	描述
RegionList	RegionInfo	可售卖地域信息列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InvalidParameter.GenericParameterError	
UnauthorizedOperation.PermissionDenied	

查询分片信息

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDCDBShards）用于查询云数据库实例的分片信息。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-27 16:08:46。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDCDBShards
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID，形如：dcdbt-ow728lmc。
ShardInstanceIds	否	否	Array of String	分片ID列表。
Offset	否	否	Int64	偏移量，默认为 0
Limit	否	否	Int64	返回数量，默认为 20，最大值为 100。
OrderBy	否	否	String	排序字段， 目前仅支持 createtime
OrderByType	否	否	String	排序类型， desc 或者 asc

3. 输出参数

参数名称	类型	描述
TotalCount	Int64	符合条件的分片数量
Shards	DCDBShardInfo	分片信息列表
DcnFlag	Int64	灾备标志，0-无，1-主实例，2-灾备实例
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.FenceError	
InternalError.GetInstanceDetailFailed	
InternalError.GetInstanceInfoFailed	
InternalError.GetVpcFailed	
InternalError.InnerSystemError	
InvalidParameter.GenericParameterError	
InvalidParameterValue.IllegalExclusterID	
InvalidParameterValue.SpecIdIllegal	
UnauthorizedOperation.PermissionDenied	

查询数据库对象列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDatabaseObjects）用于查询云数据库实例的数据库中的对象列表，包含表、存储过程、视图和函数。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-04-25 11:10:48。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDatabaseObjects
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow7t8lmc。
DbName	是	否	String	数据库名称，通过 DescribeDatabases 接口获取。

3. 输出参数

参数名称	类型	描述
InstanceId	String	透传入参。
DbName	String	数据库名称。
Tables	DatabaseTable	表列表。
Views	DatabaseView	视图列表。
Procs	DatabaseProcedure	存储过程列表。
Funcs	DatabaseFunction	函数列表。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetDbObjectFailed	
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

查询数据库表信息

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDatabaseTable）用于查询云数据库实例的表信息。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-06-26 17:48:50。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDatabaseTable
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow7t8lmc。
DbName	是	否	String	数据库名称，通过 DescribeDatabases 接口获取。
Table	是	否	String	表名称，通过 DescribeDatabaseObjects 接口获取。

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例名称。
DbName	String	数据库名称。
Table	String	表名称。
Cols	TableColumn	列信息。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetTableInfoFailed	
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

查询数据库列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeDatabases）用于查询云数据库实例的数据库列表。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-11-15 15:29:07。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDatabases
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow7t8lmc。

3. 输出参数

参数名称	类型	描述
Databases	Database	该实例上的数据库列表。
InstanceId	String	透传入参。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetDbListFailed	

错误码	描述
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

获取实例灾备详情

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

获取实例灾备详情

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-07 20:12:39。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeDcnDetail
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID

3. 输出参数

参数名称	类型	描述
DcnDetails	DcnDetailItem	DCN同步详情
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError	
InternalError.DbOperationFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	

查询流程状态

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeFlow）用于查询流程状态

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 11:24:43。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeFlow
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
FlowId	是	否	Int64	异步请求接口返回的任务流程号。

3. 输出参数

参数名称	类型	描述
Status	Int64	流程状态，0：成功，1：失败，2：运行中
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
InvalidParameter.FlowNotFound	
UnauthorizedOperation.PermissionDenied	

查看备份日志备份天数

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DescribeLogFileRetentionPeriod)用于查看数据库备份日志的备份天数的设置情况。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-04-06 16:01:38。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeLogFileRetentionPeriod
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：tdsql-ow728lmc。

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例 ID，形如：tdsql-ow728lmc。
Days	Uint64	日志备份天数
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	

错误码	描述
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

查询分布式数据库可售卖分片规格

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

查询可创建的分布式数据库可售卖的分片规格配置。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-04-09 14:22:49。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeShardSpec
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)

3. 输出参数

参数名称	类型	描述
SpecConfig	SpecConfig	按机型分类的可售卖规格列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InvalidParameter.GenericParameterError	
UnauthorizedOperation.PermissionDenied	

拉取用户任务列表

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（DescribeUserTasks）用于拉取用户任务列表

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-07 20:23:07。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeUserTasks
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Statuses	否	否	Array of Int64	任务的状态列表。0-任务启动中；1-任务运行中；2-任务成功；3-任务失败
InstanceIds	否	否	Array of String	实例ID列表
FlowTypes	否	否	Array of Int64	任务类型列表，当前支持的任务类型有：0-回档任务；1-创建实例任务；2-扩容任务；3-迁移任务；4-删除实例任务；5-重启任务
StartTime	否	否	String	任务的创建时间
EndTime	否	否	String	任务的结束时间
UTaskIds	否	否	Array of Int64	任务ID的数组
Limit	否	否	Int64	每次最多返回多少条任务，取值范围为(0,100]，不传的话默认值为20
Offset	否	否	Int64	返回任务默认是按照创建时间降序排列，从偏移值Offset处开始返回

3. 输出参数

参数名称	类型	描述
TotalCount	Int64	任务总数

参数名称	类型	描述
FlowSet	UserTaskInfo	任务列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
UnauthorizedOperation.PermissionDenied	
InternalError.OperateDatabaseFailed	

可用区Cpu架构查询

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

根据主从可用区返回可用Cpu架构查询

默认接口请求频率限制：20次/秒。

接口更新时间：2021-09-03 11:51:06。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DescribeZonesArchInfo
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Zones	是	否	Array of String	可用区信息

3. 输出参数

参数名称	类型	描述
CpuArch	String	可用Cpu架构信息
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CreateOrderFailed	
InternalError.CreateFlowError	
InternalError.DbOperationFailed	
InternalError.OperateDatabaseFailed	

错误码	描述
InvalidParameter	
InvalidParameter.InstanceNotFound	
InvalidParameter.SpecNotFound	
InvalidParameterValue.ShardNotExist	
InvalidParameterValue.SpecIdIllegal	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceStatusAbnormal	

安全组批量解绑云资源

最近更新时间: 2025-02-18 17:50:23

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(DisassociateSecurityGroups)用于安全组批量解绑实例。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 14:17:17。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：DisassociateSecurityGroups
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
Product	是	否	String	数据库引擎名称，本接口取值：dcdb。
SecurityGroupId	是	否	String	安全组Id。
InstanceIds	是	否	Array of String	实例ID列表，一个或者多个实例Id组成的数组。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.AddInstanceInfoFailed	
FailedOperation.AssociateSecurityGroupsFailed	
FailedOperation.ClearInstanceInfoFailed	

错误码	描述
FailedOperation.DisassociateSecurityGroupsFailed	

切分Binlog

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

相当于在所有分片的mysqld中执行flush logs，完成切分的binlog将展示在各个分片控制台binlog列表里。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-09-01 12:01:56。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：FlushBinlog
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError	
InternalError.DbOperationFailed	
InternalError.InnerSystemError	
InvalidParameter	

错误码	描述
LimitExceeded.TooFrequentlyCalled	
ResourceUnavailable.InstanceStatusAbnormal	

设置账号权限

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（GrantAccountPrivileges）用于给云数据库账号赋权。 注意：相同用户名，不同Host是不同的账号。

默认接口请求频率限制：20次/秒。

接口更新时间：2021-02-20 15:52:56。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：GrantAccountPrivileges
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow728lmc。
UserName	是	否	String	登录用户名。
Host	是	否	String	用户允许的访问 host，用户名+host唯一确定一个账号。
DbName	是	否	String	数据库名。如果为 *，表示查询全局权限（即 *.*），此时忽略 Type 和 Object 参数
Type	否	否	String	类型,可以填入 table 、 view 、 proc 、 func 和 *。当 DbName 为具体数据库名，Type为 * 时，表示设置该数据库权限（即db.*），此时忽略 Object 参数
Object	否	否	String	具体的 Type 的名称，例如 Type 为 table 时就是具体的表名。DbName 和 Type 都为具体名称，则 Object 表示具体对象名，不能为 * 或者为空
ColName	否	否	String	当 Type=table 时，ColName 为 * 表示对表授权，如果为具体字段名，表示对字段授权
Privileges	是	否	Array of String	全局权限： SELECT，INSERT，UPDATE，DELETE，CREATE，DROP，REFERENCES，INDEX，ALTER，CREATE TEMPORARY TABLES，LOCK TABLES，EXECUTE，CREATE VIEW，SHOW VIEW，CREATE ROUTINE，ALTER ROUTINE，EVENT，TRIGGER，SHOW DATABASES 库权限： SELECT，INSERT，UPDATE，DELETE，CREATE，DROP，REFERENCES，INDEX，ALTER，CREATE TEMPORARY TABLES，LOCK TABLES，EXECUTE，CREATE VIEW，SHOW VIEW，CREATE ROUTINE，ALTER ROUTINE，EVENT，TRIGGER 表/视图权限： SELECT，INSERT，UPDATE，DELETE，CREATE，DROP，REFERENCES，INDEX，ALTER，CREATE VIEW，SHOW VIEW，TRIGGER 存储过程/函数权限： ALTER ROUTINE，EXECUTE 字段权限： INSERT，REFERENCES，SELECT，UPDATE

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.ModifyRightFailed	
FailedOperation.OssOperationFailed	
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.GetRightFailed	
InternalError.InnerSystemError	
InvalidParameter.GenericParameterError	
InvalidParameterValue.BadUserRight	
InvalidParameterValue.SuperUserForbidden	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
InvalidParameterValue.IllegalRightParam	

初始化实例

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(InitDCDBInstances)用于初始化云数据库实例，包括设置默认字符集、表名大小写敏感等。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-01-16 20:42:57。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：InitDCDBInstances
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceIds	是	否	Array of String	待初始化的实例ID列表，形如：dcdbt-ow728lmc，可以通过 DescribeDCDBInstances 查询实例详情获得。
Params	是	否	Array of DBParamValue	参数列表。本接口的可选值为：character_set_server（字符集，必传），lower_case_table_names（表名大小写敏感，必传，0 - 敏感；1-不敏感），innodb_page_size（innodb数据页，默认16K），sync_mode（同步模式：0 - 异步； 1 - 强同步；2 - 强同步可退化。默认为强同步）。

3. 输出参数

参数名称	类型	描述
FlowIds	Uint64	异步任务ID，可通过 DescribeFlow 查询任务状态。
InstanceIds	String	透传入参。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InternalError.InnerSystemError	
InvalidParameter.GenericParameterError	
InvalidParameterValue.IllegalInitParam	
ResourceUnavailable.BadInstanceStatus	
UnauthorizedOperation.PermissionDenied	

销毁后付费实例

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（IsolateHourDCDBInstance）用于销毁/退货按量计费实例。 实例将会保留3天再进行真实的销毁动作，在彻底销毁前可以通过ActiveHourDCDBInstance 恢复/开机。实例彻底销毁后数据将无法找回，请提前备份实例数据。实例彻底销毁后IP资源同时释放。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-12-27 20:43:15。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：IsolateHourDCDBInstance
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceIds	是	否	Array of String	实例uuid列表

3. 输出参数

参数名称	类型	描述
SuccessInstanceIds	String	销毁成功的实例id列表
FailedInstanceIds	String	销毁失败的实例id列表
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError.DbOperationFailed	

错误码	描述
InternalError.InnerSystemError	
InvalidParameter	
InvalidParameter.InstanceNotFound	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

修改数据库账号备注

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyAccountDescription）用于修改云数据库账号备注。 注意：相同用户名，不同Host是不同的账号。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-11-15 15:29:30。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyAccountDescription
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow728lmc。
UserName	是	否	String	登录用户名。
Host	是	否	String	用户允许的访问 host，用户名+host唯一确定一个账号。
Description	是	否	String	新的账号备注，长度 0~256。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	

错误码	描述
InvalidParameter.GenericParameterError	
ResourceNotFound.AccountDoesNotExist	
UnauthorizedOperation.PermissionDenied	

修改实例名字

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyDBInstanceName）用于修改实例名字

默认接口请求频率限制：20次/秒。

接口更新时间：2019-05-29 18:00:17。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyDBInstanceName
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID，形如tdsql-hdaprz0v
InstanceName	是	否	String	实例名称

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.OperateDatabaseFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	

错误码	描述
UnauthorizedOperation.PermissionDenied	

修改数据库参数

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(ModifyDBParameters)用于修改数据库参数。

默认接口请求频率限制：20次/秒。

接口更新时间：2021-11-01 15:21:51。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyDBParameters
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow728lmc。
Params	是	否	Array of DBParamValue	参数列表，每一个元素是Param和Value的组合，可以通过 DescribeDBParameters 查询可配置的参数和值限制

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例 ID，形如：dcdbt-ow728lmc。
Result	ParamModifyResult	各参数修改结果
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.CamAuthFailed	

错误码	描述
InternalError.DbOperationFailed	
InternalError.GetDbConfigFailed	
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

修改同步模式

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyDBSyncMode）用于修改云数据库实例的同步模式。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-04-06 21:39:17。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyDBSyncMode
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	待修改同步模式的实例ID。形如：dcdbt-ow728lmc。
SyncMode	是	否	Int64	同步模式：0 异步，1 强同步， 2 强同步可退化

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务Id，可通过 DescribeFlow 查询任务状态。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CreateFlowFailed	
InternalError.CamAuthFailed	
InternalError.GetInstanceInfoFailed	

错误码	描述
InvalidParameter.GenericParameterError	
InvalidParameterValue.BadSyncMode	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
UnsupportedOperation.InvalidOperation	
InvalidParameter.InstanceNotFound	

修改实例备注

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyInstanceRemark）用于修改实例备注。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-08 15:28:28。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyInstanceRemark
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：tdsql-ow728lmc。
Remark	是	否	String	备注

3. 输出参数

参数名称	类型	描述
Remark	String	备注
IsModify	Int64	是否修改
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError	
InternalError.DbOperationFailed	

错误码	描述
InvalidParameter	
InvalidParameter.ActionNotFound	
InvalidParameter.PermissionDenied	

修改实例Vip

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyInstanceVip）用于修改实例Vip

默认接口请求频率限制：20次/秒。

接口更新时间：2020-07-01 13:38:04。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyInstanceVip
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID
Vip	是	否	String	实例VIP
Ipv6Flag	否	否	Uint64	**此参数对外不可见。** IPv6标志

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务流程ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	

错误码	描述
InternalError.InnerSystemError	
InvalidParameter	
InvalidParameter.VipNotInSubnet	
InvalidParameter.VipUsed	
ResourceUnavailable.InstanceStatusAbnormal	

修改实例VPORT

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyInstanceVport）用于修改实例VPORT

默认接口请求频率限制：20次/秒。

接口更新时间：2020-05-21 13:54:08。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyInstanceVport
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例ID
Vport	是	否	Uint64	实例VPORT

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.SGChangeVip	
FailedOperation.VpcAddServiceFailed	
InternalError.DbOperationFailed	
InternalError.InnerSystemError	

错误码	描述
InvalidParameter	
InvalidParameter.VipUsed	
InvalidParameter.VportUsed	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	
InvalidParameter.InstanceNotFound	

修改备份日志保存天数

最近更新时间: 2025-02-18 17:50:24

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口(ModifyLogFileRetentionPeriod)用于修改数据库备份日志保存天数。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-04-06 16:00:51。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyLogFileRetentionPeriod
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：tdsql-ow728lmc。
Days	是	否	Uint64	保存的天数,不能超过30

3. 输出参数

参数名称	类型	描述
InstanceId	String	实例 ID，形如：tdsql-ow728lmc。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.OssOperationFailed	
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	

错误码	描述
InvalidParameter.GenericParameterError	
InvalidParameterValue.IllegalLogSaveDays	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

修改就近接入策略

最近更新时间: 2025-02-18 17:50:25

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ModifyRealServerAccessStrategy）用于修改就近接入策略

默认接口请求频率限制：20次/秒。

接口更新时间：2022-08-08 16:05:41。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ModifyRealServerAccessStrategy
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例Id
RsAccessStrategy	是	否	Int64	RS就近模式, 0-无策略, 1-可用区就近访问

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.GetVpcFailed	
ResourceUnavailable.InstanceStatusAbnormal	
InvalidParameter.VpcNotFound	
InternalError.OperateDatabaseFailed	

错误码	描述
InvalidParameter	

重置账号密码

最近更新时间: 2025-02-18 17:50:25

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（ResetAccountPassword）用于重置云数据库账号的密码。 注意：相同用户名，不同Host是不同的账号。

默认接口请求频率限制：20次/秒。

接口更新时间：2018-11-15 15:29:23。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：ResetAccountPassword
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	实例 ID，形如：dcdbt-ow728lmc。
UserName	是	否	String	登录用户名。
Host	是	否	String	用户允许的访问 host，用户名+host唯一确定一个账号。
Password	是	否	String	新密码，由字母、数字或常见符号组成，不能包含分号、单引号和双引号，长度为6~32位。

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.ResetPasswordFailed	

错误码	描述
InternalError.CamAuthFailed	
InternalError.DbOperationFailed	
InvalidParameter.CharacterError	
InvalidParameter.GenericParameterError	
ResourceUnavailable.InstanceAlreadyDeleted	
ResourceUnavailable.InstanceHasBeenLocked	
ResourceUnavailable.InstanceStatusAbnormal	
UnauthorizedOperation.PermissionDenied	

重启实例

最近更新时间: 2025-02-18 17:50:25

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（RestartDBInstances）用于重启数据库实例

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-07 20:27:05。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：RestartDBInstances
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceIds	是	否	Array of String	实例ID的数组
RestartTime	否	否	Datetime	重启时间，

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务ID
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError.DbOperationFailed	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceStatusAbnormal	

错误码	描述
UnauthorizedOperation.PermissionDenied	

切换回档实例

最近更新时间: 2025-02-18 17:50:25

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（SwitchRollbackInstance）用于切换回档实例。

默认接口请求频率限制：20次/秒。

接口更新时间：2023-02-07 20:17:34。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：SwitchRollbackInstance
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	源/旧实例ID，形如：tdsql-ow728lmc。
DstInstanceId	是	否	String	目标实例ID，形如：tdsql-ow728lmc。

3. 输出参数

参数名称	类型	描述
FlowId	Int64	异步任务流程ID。
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
InternalError	
InternalError.DbOperationFailed	
InternalError.InnerSystemError	

错误码	描述
InternalError.OperateDatabaseFailed	
InvalidParameter	
InvalidParameter.ActionNotFound	
InvalidParameter.PermissionDenied	
InvalidParameterValue.IllegalInstanceId	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceStatusAbnormal	

升级后付费实例

最近更新时间: 2025-02-18 17:50:25

1. 接口描述

接口请求域名： dcdb.api3.finance.cloud.tencent.com。

本接口（UpgradeHourDCDBInstance）用于升级后付费分布式数据库实例。

默认接口请求频率限制：20次/秒。

接口更新时间：2019-12-27 20:42:04。

接口只验签名不鉴权。

2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

参数名称	必选	允许NULL	类型	描述
Action	是	否	String	公共参数，本接口取值：UpgradeHourDCDBInstance
Version	是	否	String	公共参数，本接口取值：2018-04-11
Region	是	否	String	公共参数，详见产品支持的**地域列表**(TODO)
InstanceId	是	否	String	待升级的实例ID。形如：dcdbt-ow728lmc，可以通过DescribeDCDBInstances 查询实例详情获得。
UpgradeType	是	否	String	升级类型，取值范围: - ADD: 新增分片 - EXPAND: 升级实例中的已有分片 - SPLIT: 将已有分片中的数据切分到新增分片上
AddShardConfig	否	否	AddShardConfig	新增分片配置，当UpgradeType为ADD时生效。
ExpandShardConfig	否	否	ExpandShardConfig	扩容分片配置，当UpgradeType为EXPAND时生效。
SplitShardConfig	否	否	SplitShardConfig	切分分片配置，当UpgradeType为SPLIT时生效。
SwitchStartTime	否	否	Datetime	切换开始时间，格式如: "2019-12-12 07:00:00"。开始时间必须在当前时间一个小时以后，3天以内。
SwitchEndTime	否	否	Datetime	切换结束时间, 格式如: "2019-12-12 07:15:00"，结束时间必须大于开始时间。
SwitchAutoRetry	否	否	Int64	是否自动重试。 0：不自动重试 1：自动重试

3. 输出参数

参数名称	类型	描述
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。

4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

错误码	描述
FailedOperation.CreateOrderFailed	
InternalError.CreateFlowError	
InternalError.DbOperationFailed	
InternalError.OperateDatabaseFailed	
InvalidParameter	
InvalidParameter.InstanceNotFound	
InvalidParameter.SpecNotFound	
InvalidParameterValue.ShardNotExist	
InvalidParameterValue.SpecIdIllegal	
ResourceNotFound.NoInstanceFound	
ResourceUnavailable.InstanceStatusAbnormal	

数据结构

最近更新时间: 2025-02-18 17:50:25

DBParamValue

云数据库参数信息。

被如下接口引用：CreateHourDCDBInstance、InitDCDBInstances、ModifyDBParameters

名称	必选	允许NULL	类型	描述
Param	是	否	String	参数名称
Value	是	否	String	参数值

DBAccount

云数据库账号信息

被如下接口引用：DescribeAccounts

名称	必选	允许NULL	类型	描述
CreateTime	是	否	Datetime	创建时间
DelayThresh	是	否	Int64	如果备机延迟超过本参数设置值，系统将认为备机发生故障建议该参数值大于10。当ReadOnly选择1、2时该参数生效。
Description	是	否	String	用户备注信息
Host	是	否	String	用户可以从哪台主机登录（对应 MySQL 用户的 host 字段，UserName + Host 唯一标识一个用户，IP形式，IP段以%结尾；支持填入%；为空默认等于%）
ReadOnly	是	否	Int64	只读标记，0：否， 1：该账号的sql请求优先选择备机执行，备机不可用时选择主机执行，2：优先选择备机执行，备机不可用时操作失败。
UpdateTime	是	否	Datetime	最后更新时间
UserName	是	否	String	用户名

Database

数据库信息

被如下接口引用：DescribeDatabases

名称	必选	允许NULL	类型	描述
DbName	是	否	String	数据库名称

Project

项目信息描述

被如下接口引用：

名称	必选	允许NULL	类型	描述
AppId	是	否	Int64	应用Id
CreateTime	是	否	Datetime	创建时间
CreatorUin	是	否	Int64	创建者uin
Info	是	否	String	描述信息
IsDefault	是	否	Int64	是否默认项目，1 是，0 不是
Name	是	否	String	项目名称
OwnerUin	是	否	Int64	资源拥有者（主账号）uin
ProjectId	是	否	Int64	项目ID
SrcAppId	是	否	Int64	来源AppId
SrcPlat	是	否	String	来源平台
Status	是	否	Int64	项目状态,0正常，-1关闭。默认项目为3

SecurityGroup

安全组规则

被如下接口引用：DescribeDBSecurityGroups、DescribeProjectSecurityGroups

名称	必选	允许NULL	类型	描述
CreateTime	是	否	String	创建时间，时间格式：yyyy-mm-dd hh:mm:ss。
Inbound	是	否	Array of Inbound	入站规则。
Outbound	是	否	Array of Outbound	出站规则。
ProjectId	是	否	Uint64	项目ID。
SecurityGroupId	是	否	String	安全组ID。
SecurityGroupName	是	否	String	安全组名称。
SecurityGroupRemark	是	否	String	安全组备注。

ParamDesc

DB参数描述

被如下接口引用：DescribeDBParameters

名称	必选	允许NULL	类型	描述
Constraint	是	否	ParamConstraint	参数限制
Default	是	否	String	系统默认值
HaveSetValue	是	否	Bool	是否有设置过值，false:没有设置过值，true:有设置过值。
Param	是	否	String	参数名字
SetValue	是	是	String	设置过的值，参数生效后，该值和value一样。未设置过就不返回该字段。
Value	是	否	String	当前参数值

SlowLogData

慢查询条目信息

被如下接口引用：DescribeDBSlowLogs

名称	必选	允许NULL	类型	描述
CheckSum	是	否	String	语句校验和，用于查询详情
Db	是	否	String	数据库名称
ExampleSql	是	是	String	样例Sql
FingerPrint	是	否	String	抽象的SQL语句
LockTimeAvg	是	否	String	平均的锁时间
LockTimeMax	是	否	String	最大锁时间
LockTimeMin	是	否	String	最小锁时间
LockTimeSum	是	否	String	锁时间总和
QueryCount	是	否	String	查询次数
QueryTimeAvg	是	否	String	平均查询时间
QueryTimeMax	是	否	String	最大查询时间
QueryTimeMin	是	否	String	最小查询时间
QueryTimeSum	是	否	String	查询时间总和
RowsExaminedSum	是	否	String	扫描行数
RowsSentSum	是	否	String	发送行数
TsMax	是	否	String	最后执行时间
TsMin	是	否	String	首次执行时间

名称	必选	允许NULL	类型	描述
User	是	否	String	帐号

AddShardConfig

升级实例 -- 新增分片类型

被如下接口引用：UpgradeHourDCDBInstance

名称	必选	允许NULL	类型	描述
ShardCount	是	否	Int64	新增分片的数量
ShardMemory	是	否	Int64	分片内存大小，单位 GB
ShardStorage	是	否	Int64	分片存储大小，单位 GB

BriefNodeInfo

描述分片DB节点信息

被如下接口引用：DescribeDCDBInstanceNodeInfo

名称	必选	允许NULL	类型	描述
NodeId	是	否	String	DB节点ID
Role	是	否	String	DB节点角色，取值为master或者slave
ShardId	是	否	String	节点所属分片的分片ID

SpecConfigInfo

实例可售卖规格详细信息，创建实例和扩容实例时 NodeCount、Memory 确定售卖规格，硬盘大小可用区间为[MinStorage,MaxStorage]

被如下接口引用：DescribeShardSpec

名称	必选	允许NULL	类型	描述
Cpu	是	否	Int64	CPU核数
MaxStorage	是	否	Int64	数据盘规格最大值，单位 GB
Memory	是	否	Int64	内存大小，单位 GB
MinStorage	是	否	Int64	数据盘规格最小值，单位 GB
NodeCount	是	否	Uint64	节点个数，2 表示一主一从，3 表示一主二从
Pid	是	否	Int64	产品类型 Id
Qps	是	否	Int64	最大 Qps 值

名称	必选	允许NULL	类型	描述
SuitInfo	是	否	String	推荐的使用场景

UserTaskInfo

用户任务信息

被如下接口引用：DescribeUserTasks

名称	必选	允许NULL	类型	描述
Id	是	否	Int64	任务ID
AppId	是	否	Int64	用户账户ID
Status	是	否	Int64	任务状态，0-任务初始化中；1-任务运行中；2-任务成功；3-任务失败
UserTaskType	是	否	Int64	任务类型，0-实例回档；1-实例创建；2-实例扩容；3-实例迁移；4-实例删除；5-实例重启
CreateTime	是	否	Datetime	任务创建时间
EndTime	是	否	Datetime	任务结束时间
ErrMsg	是	否	String	任务错误信息
InputData	是	否	String	客户端参数
InstanceId	是	否	String	实例ID
InstanceName	是	否	String	实例名称
RegionId	是	否	Int64	地域ID

Rsip

拉取实例后端RS信息，返回IP和PORT结构体数组

被如下接口引用：DescribeDBInstanceRsip

名称	必选	允许NULL	类型	描述
Ip	是	否	String	Ip地址
Port	是	否	Int64	端口

DBEngineInfo

数据库引擎信息

被如下接口引用：DescribeDBEngines

名称	必选	允许NULL	类型	描述
Description	是	否	String	引擎描述
Name	是	否	String	引擎名称，eg. 基于MariaDB 10.1.9 分支
Type	是	否	String	引擎类型，MariaDB|Percona
Version	是	否	String	引擎版本，eg. 10.1.9

RegionInfo

售卖可用区信息

被如下接口引用：DescribeDCDBSaleInfo

名称	必选	允许NULL	类型	描述
AvailableChoice	是	否	Array of ShardZoneChooseInfo	可选择的主可用区和从可用区
Region	是	否	String	地域英文ID
RegionId	是	否	Int64	地域数字ID
RegionName	是	否	String	地域中文名
ZoneList	是	否	Array of ZonesInfo	可用区列表

Inbound

安全组入站规则

被如下接口引用：DescribeDBSecurityGroups、DescribeProjectSecurityGroups

名称	必选	允许NULL	类型	描述
Action	是	否	String	策略，ACCEPT或者DROP。
AddressModule	是	否	String	地址组id代表的地址集合。
CidrIp	是	否	String	来源Ip或Ip段，例如192.168.0.0/16。
Desc	是	否	String	描述。
Id	是	否	String	安全组id代表的地址集合。
IpProtocol	是	否	String	网络协议，支持udp、tcp等。
PortRange	是	否	String	端口。
ServiceModule	是	否	String	服务组id代表的协议和端口集合。

TmpInstance

临时实例

被如下接口引用：DescribeDBTmpInstances

名称	必选	允许NULL	类型	描述
AppId	是	是	Int64	应用ID
CreateTime	是	是	Datetime	创建时间
InstanceId	是	是	String	实例 ID，形如：tdsql-ow728lmc。
InstanceRemark	是	是	String	实例备注
Ipv6Flag	是	是	Uint64	实例IPv6标志
PeriodEndTime	是	是	Datetime	有效期结束时间
Region	是	是	String	实例所在地域
SrcInstanceId	是	是	String	源实例 ID，形如：tdsql-ow728lmc。
Status	是	是	Int64	实例状态,0:待初始化,1:流程处理中,2:有效状态,-1:已隔离，-2：已下线
StatusDesc	是	是	String	实例状态描述
TempType	是	是	Int64	0:非临时实例 ,1:无效临时实例, 2:回档成功的有效临时实例
Vip	是	是	String	实例虚IP
Vipv6	是	是	String	实例虚IPv6
Vport	是	是	Int64	实例虚端口
Zone	是	是	String	实例所在可用区

MonitorShardInfo

分片信息

被如下接口引用：DescribeDCDBInstanceShardInfo

名称	必选	允许NULL	类型	描述
ShardId	是	否	String	分片ID

ParamConstraint

参数约束

被如下接口引用：DescribeDBParameters

名称	必选	允许NULL	类型	描述
Enum	是	否	String	约束类型为enum时的可选值列表

名称	必选	允许NULL	类型	描述
Range	是	是	ConstraintRange	约束类型为section时的范围
String	是	否	String	约束类型为string时的可选值列表
Type	是	否	String	约束类型,如枚举enum，区间section

DatabaseTable

数据库表信息

被如下接口引用：DescribeDatabaseObjects

名称	必选	允许NULL	类型	描述
Table	是	否	String	表名

InstanceResource

数据库实例信息

被如下接口引用：DescribeInstances

名称	必选	允许NULL	类型	描述
AppId	是	否	Int64	账户的应用ID
ClusterName	是	否	String	实例所属集群的名字
DbVersion	是	否	String	实例的数据库版本
Id	是	否	Int64	数据库实例的数字ID
InstanceId	是	否	String	实例ID，形如 dcdbt-ifzc58fh或者tdsqlshard-ifzc58fh
InstanceName	是	否	String	实例名称
NodeCount	是	否	Int64	实例的节点个数
OriginSerialId	是	否	String	实例的原始SerialId
Region	是	否	String	实例所在地域
SerialId	是	否	String	实例的当前SerialId
ShardDetail	是	否	Array of InstanceShardInfo	分片信息
Status	是	否	Int64	实例运行状态
SubnetId	是	否	Int64	实例所属子网的数字ID
UniqueSubnetId	是	否	String	实例所属子网的字符串唯一ID
UniqueVpcId	是	否	String	实例所属VPC的字符串唯一ID

名称	必选	允许NULL	类型	描述
Vip	是	否	String	实例Vip
VpcId	是	否	Int64	实例所属VPC的数字ID
Vport	是	否	Int64	实例Vport
ZkName	是	否	String	实例所属Zk的名字
Zone	是	否	String	实例所在可用区
MonitorType	是	否	Int64	表示实例监控页面如何展示

ZonesInfo

可用区信息

被如下接口引用：DescribeDCDBSaleInfo

名称	必选	允许NULL	类型	描述
Zone	是	否	String	可用区英文ID
ZoneId	是	否	Int64	可用区数字ID
ZoneName	是	否	String	可用区中文名

SplitShardConfig

升级实例 -- 切分分片类型

被如下接口引用：UpgradeHourDCDBInstance

名称	必选	允许NULL	类型	描述
ShardInstanceIds	是	否	Array of String	分片ID数组
ShardMemory	是	否	Int64	分片内存大小，单位 GB
ShardStorage	是	否	Int64	分片存储大小，单位 GB
SplitRate	是	否	Int64	数据切分比例

ExpandShardConfig

升级实例 -- 扩容分片类型

被如下接口引用：UpgradeHourDCDBInstance

名称	必选	允许NULL	类型	描述
ShardInstanceIds	是	否	Array of String	分片ID数组

名称	必选	允许NULL	类型	描述
ShardMemory	是	否	Int64	分片内存大小，单位 GB
ShardStorage	是	否	Int64	分片存储大小，单位 GB

SpecConfig

按机型分类的规格配置

被如下接口引用：DescribeShardSpec

名称	必选	允许NULL	类型	描述
Machine	是	否	String	规格机型
SpecConfigInfos	是	否	Array of SpecConfigInfo	规格列表

ResourceTag

标签对象，包含tagKey & tagValue

被如下接口引用：CreateHourDCDBInstance、DescribeDCDBInstanceDetail

名称	必选	允许NULL	类型	描述
TagKey	是	否	String	标签键key
TagValue	是	否	String	标签值value

ShardBriefInfo

实例分片信息

被如下接口引用：DescribeDCDBInstanceDetail

名称	必选	允许NULL	类型	描述
Cpu	是	否	Int64	分片Cpu核数
CreateTime	是	否	Datetime	分片创建时间
LogDisk	是	否	Int64	分片日志磁盘空间大小，单位GB
Memory	是	否	Int64	分片内存大小，单位GB
NodeCount	是	否	Int64	分片节点个数
ProxyVersion	是	否	String	分片Proxy版本信息
ShardInstanceId	是	否	String	分片ID，形如shard-7vg1o339
ShardMasterZone	是	否	String	分片主DB可用区

名称	必选	允许NULL	类型	描述
ShardSerialId	是	否	String	分片SerialId
ShardSlaveZones	是	否	Array of String	分片从DB可用区
Status	是	否	Int64	分片运行状态
StatusDesc	是	否	String	分片运行状态描述
Storage	是	否	Int64	分片磁盘大小，单位GB
StorageUsage	是	否	Float	分片磁盘空间使用率
NodesInfo	是	是	Array of NodeInfo	DB节点信息

ShardZoneChooseInfo

分片节点可用区选择

被如下接口引用：DescribeDCDBSaleInfo

名称	必选	允许NULL	类型	描述
MasterZone	是	否	ZonesInfo	主可用区
SlaveZones	是	否	Array of ZonesInfo	可选的从可用区

Permission

描述对资源权限，仅内部使用。

被如下接口引用：AuthenticateCAM

名称	必选	允许NULL	类型	描述
IsPermitted	是	否	Int64	是否有权限，1 - 有权限， 0 - 无权限
Resource	是	否	String	资源对象

DatabaseProcedure

数据库存储过程信息

被如下接口引用：DescribeDatabaseObjects

名称	必选	允许NULL	类型	描述
Proc	是	否	String	存储过程名称

ShardInfo

分片信息

被如下接口引用：DescribeDCDBInstances

名称	必选	允许NULL	类型	描述
Cpu	是	否	Uint64	Cpu核数
Createtime	是	否	String	创建时间
Memory	是	否	Int64	内存大小，单位 GB
NodeCount	是	否	Int64	节点数，2 为一主一从， 3 为一主二从
Pid	是	否	Int64	产品类型 Id（过时字段，请勿依赖该值）
ShardId	是	否	Int64	分片数字ID
ShardInstanceId	是	否	String	分片ID
ShardSerialId	是	否	String	分片Set ID
Status	是	否	Int64	状态：0 创建中，1 流程处理中， 2 运行中，3 分片未初始化，-2 分片已删除
Storage	是	否	Int64	存储大小，单位 GB

Outbound

安全组出站规则

被如下接口引用：DescribeDBSecurityGroups、DescribeProjectSecurityGroups

名称	必选	允许NULL	类型	描述
Action	是	否	String	策略，ACCEPT或者DROP。
AddressModule	是	否	String	地址组id代表的地址集合。
CidrIp	是	否	String	来源Ip或Ip段，例如192.168.0.0/16。
Desc	是	否	String	描述。
Id	是	否	String	安全组id代表的地址集合。
IpProtocol	是	否	String	网络协议，支持udp、tcp等。
PortRange	是	否	String	端口。
ServiceModule	是	否	String	服务组id代表的协议和端口集合。

DatabaseView

数据库视图信息

被如下接口引用：DescribeDatabaseObjects

名称	必选	允许NULL	类型	描述
View	是	否	String	视图名称

LogFileInfo

拉取的日志信息

被如下接口引用：DescribeDBLogFiles

名称	必选	允许NULL	类型	描述
FileName	是	否	String	文件名
Length	是	否	Uint64	文件长度
Mtime	是	否	Uint64	Log最后修改时间
Uri	是	否	String	下载Log时用到的统一资源标识符

CommonDBInstance

http://imgcache.finance.cloud.tencent.com:80git.code.oa.com/QCloudCDB-Platform/working-dashboard/issues/66

被如下接口引用：DescribeCommonDBInstances

名称	必选	允许NULL	类型	描述
AppId	是	是	Int64	应用Id
CreateTime	是	是	String	实例创建时间
InstanceId	是	是	String	实例ID
InstanceName	是	是	String	实例名称
NetType	是	是	Int64	0-基础网络，1-VPC网络
PayMode	是	是	Int64	计费类型，1-包年包月，0-按量计费
ProjectId	是	是	Int64	项目Id
Region	是	是	String	地域
Status	是	是	String	实例状态信息，0-创建中,1-运行中，2-隔离中,下线后无法拉取实例信息
SubnetId	是	是	String	子网统一Id
Vips	是	是	Array of String	VPC网络IP
VpcId	是	是	String	VPC网络统一Id
Vport	是	是	Int64	VPC网络端口
Zone	是	是	String	可用区

DCDBShardInfo

描述分布式数据库分片信息。

被如下接口引用：DescribeDCDBShards

名称	必选	允许NULL	类型	描述
Cpu	是	否	Int64	CPU核数
CreateTime	是	否	Datetime	创建时间
InstanceId	是	否	String	所属实例Id
Memory	是	否	Int64	内存大小，单位 GB
MemoryUsage	是	否	Float	内存使用率，单位为 %
NodeCount	是	否	Int64	节点数，2 为一主一从， 3 为一主二从
Paymode	是	是	String	付费模型
PeriodEndTime	是	否	Datetime	到期时间
Pid	是	否	Int64	产品ProductID
ProjectId	是	否	Int64	项目ID
ProxyVersion	是	否	String	Proxy版本
Region	是	否	String	地域
ShardId	是	否	Int64	数字分片Id（过时字段，请勿依赖该值）
ShardInstanceId	是	否	String	全局唯一的分片Id
ShardMasterZone	是	是	String	分片的主可用区
ShardSerialId	是	否	String	分片SQL透传Id，用于将sql透传到指定分片执行
ShardSlaveZones	是	是	Array of String	分片的从可用区列表
Status	是	否	Int64	状态：0 创建中，1 流程处理中， 2 运行中，3 分片未初始化
StatusDesc	是	否	String	状态中文描述
Storage	是	否	Int64	存储大小，单位 GB
StorageUsage	是	否	Float	存储使用率，单位为 %
SubnetId	是	否	String	字符串格式的私有网络子网Id
VpcId	是	否	String	字符串格式的私有网络Id
Zone	是	否	String	可用区
Range	是	否	String	分片ShardKey的范围（总共64个哈希值），例如： 0-31，32-63
CpuArch	否	否	String	cpu架构

DatabaseFunction

数据库函数信息

被如下接口引用：DescribeDatabaseObjects

名称	必选	允许NULL	类型	描述
Func	是	否	String	函数名称

FenceInfoItem

独享资源池信息

被如下接口引用：DescribeAvailableExclusiveGroups

名称	必选	允许NULL	类型	描述
FenceId	是	否	String	独享资源池ID

DBCharsetItem

DB字符集信息

被如下接口引用：DescribeDBCharsets

名称	必选	允许NULL	类型	描述
Version	否	否	String	DB版本号
DefaultCharset	否	否	String	DB默认字符集
OptionalCharsets	否	否	String	DB可选字符集

DcnDetailItem

DCN详情条目

被如下接口引用：DescribeDcnDetail

名称	必选	允许NULL	类型	描述
DcnFlag	是	否	Int64	实例DCN标志，1-主，2-备
DcnStatus	是	否	Int64	实例DCN状态，0-无，1-创建中，2-同步中，3-已断开
InstanceId	是	否	String	实例ID
InstanceName	是	否	String	实例名称
Region	是	否	String	实例地域
Status	是	否	Int64	实例状态

名称	必选	允许NULL	类型	描述
StatusDesc	是	否	String	实例状态描述
Vip	是	否	String	实例IP地址
Vipv6	是	否	String	实例IPv6地址
Vport	是	否	Int64	实例端口
Zone	是	否	String	实例可用区
EncryptStatus	是	否	Int64	KMS开启状态，0-未开启，1-已开启

InstanceShardInfo

TDSQL实例的分片信息

被如下接口引用：DescribeInstances

名称	必选	允许NULL	类型	描述
ShardInstanceId	是	否	String	分片ID
ShardSerialId	是	否	String	分片的SerialId
Status	是	否	Int64	分片的运行状态

TableColumn

数据库列信息

被如下接口引用：DescribeDatabaseTable

名称	必选	允许NULL	类型	描述
Col	是	否	String	列名称
Type	是	否	String	列类型

ParamModifyResult

修改参数结果

被如下接口引用：ModifyDBParameters

名称	必选	允许NULL	类型	描述
Code	是	否	Int64	参数修改结果。0表示修改成功；-1表示修改失败；-2表示该参数值非法
Param	是	否	String	修改参数名字

DCDBInstanceInfo

分布式数据库实例信息

被如下接口引用：DescribeDCDBInstances

名称	必选	允许NULL	类型	描述
AppId	是	否	Int64	应用ID
AutoRenewFlag	是	否	Int64	自动续费标志
Cpu	是	否	Uint64	Cpu核数
CreateTime	是	否	Datetime	创建时间
DbEngine	是	否	String	数据库引擎
DbVersion	是	否	String	数据库引擎版本
DcnDstNum	是	是	Int64	DCN灾备实例数
DcnFlag	是	是	Int64	DCN标志，0-无，1-主实例，2-灾备实例
DcnStatus	是	是	Int64	DCN状态，0-无，1-创建中，2-同步中，3-已断开
ExclusterId	是	否	String	独享集群ID，为空表示非独享集群实例
Id	是	否	Uint64	数字实例ID（过时字段，请勿依赖该值）
InstanceId	是	否	String	实例ID
InstanceName	是	否	String	实例名称
Ipv6Flag	是	是	Uint64	实例IPv6标志
IsAuditSupported	是	否	Uint64	该实例是否支持审计。1-支持；0-不支持
IsTmp	是	否	Int64	临时实例标记，0 为非临时实例
IsolatedTimestamp	是	否	Datetime	隔离时间
Locker	是	是	Int64	实例处于异步任务状态时，表示异步任务流程ID
Memory	是	否	Int64	内存大小，单位 GB
NodeCount	是	否	Int64	节点数，2 为一主一从， 3 为一主二从
Paymode	是	否	String	付费模式
PeriodEndTime	是	否	Datetime	到期时间
Pid	是	否	Int64	产品类型 ID（过时字段，请勿依赖该值）
ProjectId	是	否	Int64	项目ID
Region	是	否	String	地域
ShardCount	是	否	Int64	分片个数

名称	必选	允许NULL	类型	描述
ShardDetail	是	否	Array of ShardInfo	分片详情
Status	是	否	Int64	状态
StatusDesc	是	否	String	状态中文描述
Storage	是	否	Int64	存储大小，单位 GB
SubnetId	是	否	Int64	Subnet数字ID
Uin	是	否	String	账号ID
UniqueSubnetId	是	否	String	字符串型的私有网络子网ID
UniqueVpcId	是	否	String	字符串型的私有网络ID
UpdateTime	是	否	Datetime	实例最后更新时间，格式为 2006-01-02 15:04:05
Vip	是	否	String	内网IP
Vipv6	是	是	String	内网IPv6
VpcId	是	否	Int64	VPC数字ID
Vport	是	否	Int64	内网端口
WanDomain	是	否	String	外网访问的域名，公网可解析
WanPort	是	否	Int64	外网端口
WanPortIpv6	是	是	Uint64	外网IPv6端口
WanStatus	是	否	Int64	外网状态，0-未开通；1-已开通；2-关闭；3-开通中
WanStatusIpv6	是	是	Uint64	外网IPv6状态
WanVip	是	否	String	外网 IP 地址，公网可访问
WanVipv6	是	是	String	外网IPv6
Zone	是	否	String	可用区
InstanceType	是	是	Int64	1： 主实例（独享型），2: 主实例, 3： 灾备实例, 4： 灾备实例（独享型）
CpuArch	否	否	String	Cpu架构

ConstraintRange

约束类型值的范围

被如下接口引用：DescribeDBParameters

名称	必选	允许NULL	类型	描述
Max	是	否	String	约束类型为section时的最大值

名称	必选	允许NULL	类型	描述
Min	是	否	String	约束类型为section时的最小值

NodeInfo

描述DB节点信息

被如下接口引用：DescribeDCDBInstanceDetail

名称	必选	允许NULL	类型	描述
NodeId	是	否	String	DB节点ID
Role	是	否	String	DB节点角色，取值为master或者slave

错误码

最近更新时间: 2025-02-18 17:50:25

功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

Error 中的 Code 表示错误码，Message 表示该错误的具体信息。

错误码列表

公共错误码

错误码	说明
AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）。
AuthFailure.MFAFailure	MFA 错误。
AuthFailure.SecretIdNotFound	密钥不存在。 请在控制台检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。
AuthFailure.SignatureExpire	签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。
AuthFailure.SignatureFailure	签名错误。 签名计算错误，请对照调用方式中的接口鉴权文档检查签名计算过程。
AuthFailure.TokenFailure	token 错误。
AuthFailure.UnauthorizedOperation	请求未 CAM 授权。
DryRunOperation	DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。
FailedOperation	操作失败。
InternalError	内部错误。
InvalidAction	接口不存在。
InvalidParameter	参数错误。
InvalidParameterValue	参数取值错误。
LimitExceeded	超过配额限制。

错误码	说明
MissingParameter	缺少参数错误。
NoSuchVersion	接口版本不存在。
RequestLimitExceeded	请求的次数超过了频率限制。
ResourceInUse	资源被占用。
ResourceInsufficient	资源不足。
ResourceNotFound	资源不存在。
ResourceUnavailable	资源不可用。
UnauthorizedOperation	未授权操作。
UnknownParameter	未知参数错误。
UnsupportedOperation	操作不支持。
UnsupportedProtocol	http(s)请求协议错误，只支持 GET 和 POST 请求。
UnsupportedRegion	接口不支持所传地域。

业务错误码

错误码	说明
InternalError.CosSignUrl	
InvalidParameterValue.IllegalCount	
FailedOperation.SGChangeVip	
InternalError.FenceError	
InvalidParameter.VipNotInSubnet	
InvalidParameter.CharacterError	
InternalError.GetDbObjectFailed	
FailedOperation.CopyRightError	
FailedOperation.AddInstanceInfoFailed	
FailedOperation.OssOperationFailed	
InternalError.DbOperationFailed	
FailedOperation.AssociateSecurityGroupsFailed	
InvalidParameter.IllegalTime	
LimitExceeded.TooFrequentlyCalled	
InvalidParameter.VportUsed	
InternalError.OperateDatabaseFailed	

错误码	说明
InternalError.GetInstanceInfoFailed	
FailedOperation.DisassociateSecurityGroupsFailed	
FailedOperation.UpdateInstanceInfoFailed	
InvalidParameter.SubnetUnavailable	
UnsupportedOperation.OldProxyVersion	
AuthFailure	
ResourceUnavailable.BadInstanceStatus	
ResourceUnavailable.InstanceHasBeenLocked	
InvalidParameterValue.IllegalZone	
FailedOperation.CreateFlowFailed	
InternalError.CamAuthFailed	
InternalError.GetDbListFailed	
InvalidParameterValue.BadSyncMode	
FailedOperation.VpcAddServiceFailed	
InvalidParameterValue.IllegalRightParam	
InternalError.GetSecurityGroupDetailFailed	
InvalidParameterValue.ShardNotExist	
InternalError.ReadDatabaseFailed	
FailedOperation.ResetPasswordFailed	
InvalidParameter.VpcNotFound	
InvalidParameterValue.SuperUserForbidden	
FailedOperation.ModifyRightFailed	
InternalError.GetTableInfoFailed	
ResourceUnavailable.CosApiFailed	
InternalError.VpcOperationFailed	
FailedOperation.CreateUserFailed	
FailedOperation.DeleteUserFailed	
FailedOperation.UserNotAuthed	
UnauthorizedOperation.PermissionDenied	
InternalError.RouteNotFound	

错误码	说明
InvalidParameterValue.AccountAlreadyExists	
InternalError.GetSubnetFailed	
InvalidParameterValue.BadUserRight	
InternalError.GetRightFailed	
FailedOperation.TagDryRunError	
InvalidParameter	
InternalError.QueryPriceFailed	
InvalidParameter.PermissionDenied	
InvalidParameter.GenericParameterError	
InvalidParameterValue.SpecIdIllegal	
InvalidParameterValue.IllegalExclusterID	
InternalError.GetCipherTextFailed	
InternalError.GetInstanceDetailFailed	
InvalidParameterValue.IllegalInstanceId	
InvalidParameterValue.IllegalLogSaveDays	
InvalidParameter.ActionNotFound	
InvalidParameter.VipUsed	
ResourceUnavailable.InstanceStatusAbnormal	
FailedOperation.CreateOrderFailed	
InternalError.GetDbConfigFailed	
ResourceInUse.TempInstanceExist	
InvalidParameter.NotSupportedPayMode	
InternalError.RetreateTime	
InternalError.QueryDatabaseFailed	
FailedOperation.LogNotExisted	
InternalError.UpdateDatabaseFailed	
InvalidParameterValue.BadUserType	
ResourceNotFound.AccountDoesNotExist	
InternalError.GetVpcFailed	
FailedOperation.SetRuleLocationFailed	

错误码	说明
InternalError.GetSlowLogFailed	
InternalError	
InternalError.CosConfiguration	
FailedOperation.ClearInstanceInfoFailed	
InternalError.GetUserListFailed	
InvalidParameter.SpecNotFound	
InvalidParameter.SubnetNotFound	
InvalidParameterValue.IllegalInitParam	
InvalidParameterValue.IllegalNodeCount	
InvalidParameter.FlowNotFound	
ResourceNotFound.NoInstanceFound	
InternalError.CreateFlowError	
UnsupportedOperation.InvalidOperation	
ResourceUnavailable.InstanceAlreadyDeleted	
InternalError.InnerSystemError	
InvalidParameter.InstanceNotFound	