TDSQL-C MySQL 版

产品简介

产品文档

腾讯云

文档目录

产品简介

产品概述

最近更新时间：2023-03-01 14:35:47

TDSQL-C MySQL 版（TDSQL-C for MySQL）是腾讯云自研的新一代云原生关系型数据库。融合了传统数据库、云计算与新硬件技术的优势，为用户提供具备高弹性、高性能、海量存储、安全可靠的数据库服务。TDSQL-C MySQL 版100%兼容 MySQL 5.7、8.0。实现超百万级 QPS 的高吞吐，最高 PB 级智能存储，保障数据安全可靠。
TDSQL-C MySQL 版采用存储和计算分离的架构，所有计算节点共享一份数据，提供秒级的配置升降级、秒级的故障恢复，单节点可支持百万级 QPS，自动维护数据和备份，最高以GB/秒的速度并行回档。
TDSQL-C MySQL 版既融合了商业数据库稳定可靠、高性能、可扩展的特征，又具有开源云数据库简单开放、高效迭代的优势。TDSQL-C MySQL 版引擎完全兼容原生 MySQL，您可以在不修改应用程序任何代码和配置的情况下，将 MySQL 数据库迁移至 TDSQL-C MySQL 版引擎。

核心设计理念

Cloud Native 应“ 云 ”而生 —— 云原生数据库面向服务架构
TDSQL-C MySQL 版数据库是搭建在腾讯云现有的高效稳定的云服务之上，能快速地搭建出高性能、高可用、高可靠的一套云数据库。
Creative “ 分 ”而 治之 —— 计算与存储分离，日志即数据库
TDSQL-C MySQL 版实现了“日志即数据库”的架构，将计算（CPU、内存）与存储分离，通过对 MySQL 内核的深度改造，卸载了不必要的功能模块，实现了无状态的计算节点，使得计算资源可以在秒级的时间内完成弹性扩展和故障恢复，并将其构建在腾讯云分布式云存储之上实现了存储资源的池化。
Comprehensive “ 兼 ”容并包 —— 全面兼容新版开源数据库
100%兼容开源数据库引擎 MySQL，还会定期实现对新版本的支持，几乎无需改动代码，即可完成现有数据库的查询、应用和工具平滑迁移，为用户极大降低数据迁移的成本和风险。
Cohesive 相“ 辅 ”相成 —— 极简的软件优化释放硬件红利
TDSQL-C MySQL 版通过数据库内核、系统架构等软件优化，有效提升了数据库性能和稳定性，较传统架构的数据库产品有了大幅提升。在相同硬件条件下性能更为出众，即先释放硬件红利，并完美适配新硬件的发展趋势，最大程度上提升数据库服务效能。
Cost Effective 事半功“ 倍 ”—— 性能成倍提升，按量计费
我们需要一个在性能上能超过传统数据库的云数据库，并且可以给用户减少成本压力，因为云计算的本身其实是要给客户一个很实惠的服务，所以 TDSQL-C MySQL 版，可实现真正的按量计费和弹性的扩缩容。

产品定价

详情请参见 TDSQL-C MySQL 版购买页。

如何使用 TDSQL-C MySQL 版

您可以通过以下方式管理 TDSQL-C MySQL 版集群，包括创建集群、创建数据库、创建帐号等。

控制台：提供可视化图形化的 Web 界面，操作方便。

API：控制台上所有的操作都可以通过 API 实现。

产品优势

最近更新时间：2023-03-01 14:33:46

本文介绍 TDSQL-C MySQL 版的产品优势，帮助您更好地了解 TDSQL-C MySQL 版。

完全兼容

TDSQL-C MySQL 版将开源数据库的计算和存储分离，存储构建在腾讯云分布式云存储服务之上，计算层全面兼容开源数据库引擎 MySQL 5.7、8.0，业务无需改造即可平滑迁移。

超高性能

单节点百万 QPS 的超高性能，可以满足高并发高性能的场景，保证关键业务的连续性，并可进一步提供读写分离以及读写扩展性。

海量存储

最高支持 PB 级的海量存储，为客户免去面对海量的数据时频繁分库分表的繁琐操作，同时支持数据压缩，在海量数据检索和写入性能上进行了大量优化。

秒级故障恢复

计算节点实现了无状态，支持秒级的故障切换和恢复，即便计算节点所在的物理机宕机也可以在一分钟之内恢复。

数据高可靠

集群支持安全组和 VPC 网络隔离。自动维护数据和备份的多个副本，保障数据安全可靠，可靠性达99.9999999%。

弹性扩展

计算节点可根据业务需要快速升降配，秒级完成扩容，结合弹性存储，实现计算资源的成本最优。

快速只读扩展

计算节点可根据业务需要快速添加只读节点，一个集群支持秒级添加或删除1个 - 15个只读节点，快速应对业务峰值和变化场景。

快照备份回档

基于数据多版本的秒级快照备份对用户的数据进行连续备份保护，免去主从架构备份回档数据的同步和搬迁，最高以GB/秒的速度极速并行回档，保证业务数据迅速恢复。

Serverless 架构

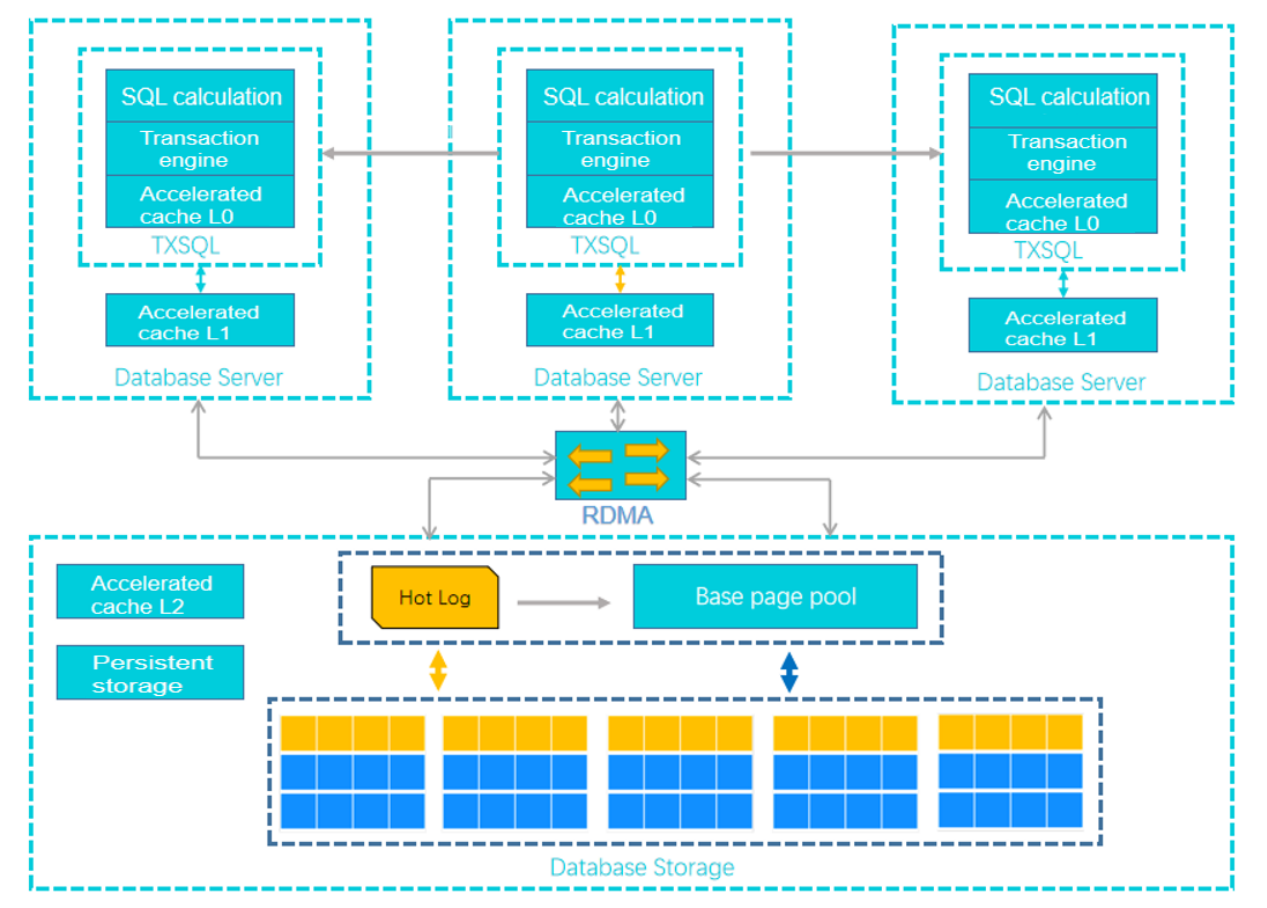
Serverless 是腾讯自研云原生数据库 TDSQL-C MySQL 版的无服务器架构版，自动扩缩容，仅按照实际使用量计费，不用不计费，轻松应对业务数据量动态变化和持续增长。

产品架构

最近更新时间：2023-03-01 14:33:46

TDSQL-C MySQL 版基于 Cloud Native 设计理念，既融合了商业数据库稳定可靠、高性能、可扩展的特征，又具有开源云数据库简单开放、高效迭代的优势。本文将介绍 TDSQL-C MySQL 版的产品架构及特点。

产品架构图



一写多读

TDSQL-C MySQL 版，一个集群中包含一个主节点和最多15个只读节点。主节点处理读写请求，只读节点仅处理读请求。

计算与存储分离

TDSQL-C MySQL 版采用计算与存储分离的设计理念，满足公共云计算环境下根据业务发展弹性扩展集群的刚性需求。数据库的计算节点（Database Engine Server）仅存储元数据，而将数据文件、Redo Log 等存储于远端的存储节点（Database Storage Server）。各计算节点之间仅需同步 Redo Log 相关的元数据信息，极大降低了主节点和只读节点间的复制延迟，而且在主节点故障时，可快速拉起新节点实现平滑替换。

自动读写分离

自动读写分离是 TDSQL-C MySQL 版提供的一个透明、高可用、自适应的负载均衡能力。通过配置数据库代理地址，SQL 请求自动转发到 TDSQL-C MySQL 版的各个节点，提供聚合、高吞吐的并发 SQL 处理能力。

高速链路互联

支持全链路 RDMA（Remote Direct Memory Access）传输，即将数据直接从一台计算机的内存传输到另一台计算机，无需双方操作系统的介入，进一步优化了关键路径的系统性能，降低请求延迟，使 I/O 性能不再成为瓶颈，存储的多个副本之间也采用 RDMA 网络。

共享分布式存储

多个计算节点共享一份数据，而不是每个计算节点都存储一份数据，极大降低了用户的存储成本。基于全新打造的分布式块存储和文件系统，存储容量可以在线平滑扩展，不会受到单个数据库服务器的存储容量限制，可承载 PB 级别的数据规模。

数据多副本强一致

数据库存储节点的数据采用多副本形式，确保数据的可靠性，并通过多副本强一致策略保证数据的一致性。数据文件采用三副本强一致，保证数据可靠性，计费仅按照“单副本”数据量统计。

产品规格

最近更新时间：2023-02-07 11:56:37

本文介绍 TDSQL-C MySQL 版的实例规格，帮助您了解 TDSQL-C MySQL 版实例的最新规格信息和历史规格信息，您可以查看本文了解各个规格的具体配置。

说明：

当前规格列表中可能存在部分已下线的规格，请以实际购买页的规格为准。

TDSQL-C MySQL 版集群下，主实例和只读实例的规格配置一样。

若您有更高的规格存储需求，请 提交工单 联系工作人员处理。

计算节点规格

包年包月/按量计费模式的计算节点规格：

计算节点规格（CPU 和内存）	支持最大存储空间（GB）		最大 IOPS	I/O 带宽
	MySQL 5.7内核小版本 < 2.0.15 MySQL 8.0内核小版本 < 3.1.2	MySQL 5.7内核小版本 ≥ 2.0.15 MySQL 8.0内核小版本 ≥ 3.1.2		
1核1GB	1000	3000	8000	1 Gbps
1核2GB	1000	3000	8000	1 Gbps
2核4GB	5000	10000	48000	6 Gbps
2核8GB	5000	10000	48000	6 Gbps
2核16GB	5000	10000	48000	6 Gbps
4核8GB	10000	30000	96000	12 Gbps
4核16GB	10000	30000	96000	12 Gbps
4核24GB	10000	30000	96000	12 Gbps
4核32GB	10000	30000	96000	12 Gbps
8核16GB	10000	50000	216000	27 Gbps
8核32GB	10000	50000	216000	27 Gbps
8核48GB	10000	50000	216000	27 Gbps

8核64GB	10000	50000	216000	27 Gbps
12核48GB	10000	80000	288000	36 Gbps
12核72GB	10000	80000	288000	36 Gbps
12核96GB	10000	80000	288000	36 Gbps
16核64GB	20000	100000	384000	48 Gbps
16核96GB	20000	100000	384000	48 Gbps
16核128GB	20000	100000	384000	48 Gbps
24核96GB	20000	150000	480000	60 Gbps
24核144GB	20000	150000	480000	60 Gbps
24核192GB	20000	150000	480000	60 Gbps
32核128GB	50000	200000	576000	72 Gbps
32核192GB	50000	200000	576000	72 Gbps
32核256GB	50000	200000	576000	72 Gbps
48核192GB	50000	300000	648000	81 Gbps
48核288GB	50000	300000	648000	81 Gbps
48核384GB	50000	300000	648000	81 Gbps
48核488GB	50000	300000	648000	81 Gbps
64核256GB	50000	400000	720000	90 Gbps
64核384GB	50000	400000	720000	90 Gbps
64核512GB	50000	400000	720000	90 Gbps
88核710GB	50000	400000	780000	98 Gbps

实例类型

最近更新时间：2023-09-12 15:01:32

本文为您介绍 TDSQL-C MySQL 版的实例类型，包括通用型以及独享型。

实例类型	描述
通用型	专享被分配的内存和磁盘，与同一物理机上的其他通用规格实例共享 CPU 资源。 通过资源复用享受规模红利，性价比较高，CPU 资源轻微复用。 磁盘大小不和 CPU 及内存绑定，可以灵活选配。
独享型	完全独享的 CPU（绑核）、内存以及磁盘资源，性能长期稳定，不会因为物理机上其它实例的行为而受到影响。 独享型的顶配是独占物理机，完全独占一台物理机的所有资源。

地域和可用区

最近更新时间：2023-08-22 15:51:53

腾讯云数据库托管机房分布在全球多个位置，这些位置节点称为地域（Region），每个地域又由多个可用区（Zone）构成。
每个地域（Region）都是一个独立的地理区域。每个地域内都有多个相互隔离的位置，称为可用区（Zone）。每个可用区都是独立的，但同一地域下的可用区通过低时延的内网链路相连。腾讯云支持用户在不同位置分配云资源，建议用户在设计系统时考虑将资源放置在不同可用区以屏蔽单点故障导致的服务不可用状态。
地域、可用区名称是对机房覆盖范围最直接的体现，为便于客户理解，命名规则如下：
地域命名采取【覆盖范围＋机房所在城市】的结构，前半段表示该机房的覆盖能力，后半段表示该机房所在或临近的城市。
可用区命名采取【城市 + 编号】的结构。

地域

腾讯云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。建议您选择最靠近您用户的地域，可降低访问时延、提高下载速度。用户启动实例、查看实例等操作都是区分地域属性的。
云产品内网通信的注意事项如下：
同地域下（保障同一账号，且同一个 VPC 内）的云资源之间可通过内网互通，可以直接使用 内网 IP 访问。
不同地域之间网络完全隔离，不同地域之间的云产品默认不能通过内网互通。
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 云联网 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

可用区（Zone）是指腾讯云在同一地域内电力和网络互相独立的物理数据中心。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。
用户启动实例时，可以选择指定地域下的任意可用区。当用户需要设计应用系统的高可靠性时（某个实例发生故障时服务保持可用），可以使用跨可用区的部署方案（例如 负载均衡、弹性 IP 等），以使另一可用区域中的实例可代为处理相关请求。

地域和可用区列表

中国

地域	可用区
华南地区（广州） ap-guangzhou	广州四区 ap-guangzhou-4
	广州六区 ap-guangzhou-6
	广州七区 ap-guangzhou-7
华东地区（上海） ap-shanghai	上海二区 ap-shanghai-2
	上海四区 ap-shanghai-4
	上海五区 ap-shanghai-5
华北地区（北京） ap-beijing	北京三区 ap-beijing-3
	北京五区 ap-beijing-5
	北京六区 ap-beijing-6
	北京七区 ap-beijing-7
华北地区（北京金融） ap-beijing-fsi	北京金融一区（仅限金融机构和企业通过 在线咨询 申请开通） ap-beijing-fsi-1
华东地区（南京） ap-nanjing	南京一区 ap-nanjing-1
西南地区（成都） ap-chengdu	成都一区 ap-chengdu-1
西南地区（重庆） ap-chongqing	重庆一区 ap-chongqing-1

港澳台地区（中国香港） ap-hongkong	香港二区 ap-hongkong-2
	香港三区 ap-hongkong-3

其他国家和地区

地域	可用区
亚太东南（新加坡） ap-singapore	新加坡三区 ap-singapore-3
	新加坡四区 ap-singapore-4
美国西部（硅谷） na-siliconvalley	硅谷一区 na-siliconvalley-1
	硅谷二区 na-siliconvalley-2
欧洲地区（法兰克福） eu-frankfurt	法兰克福一区 eu-frankfurt-1
	法兰克福二区 eu-frankfurt-2
亚太东北（首尔） ap-seoul	首尔二区 ap-seoul-2
美国东部（弗吉尼亚） na-ashburn	弗吉尼亚一区 na-ashburn-1
	弗吉尼亚二区 na-ashburn-2
亚太东北（东京） ap-tokyo	东京一区 ap-tokyo-1
	东京二区 ap-tokyo-2

如何选择地域和可用区

购买云服务时建议选择最靠近您的地域，可降低访问时延、提高下载速度。

应用场景

最近更新时间：2023-03-01 14:35:16

TDSQL-C MySQL 版为用户提供具备超高弹性、高性能、海量存储、安全可靠的数据库服务，可帮助企业轻松应对诸如商品订单等高频交易、伴随流量洪峰的快速增长业务、游戏业务、历史订单等大数据量低频查询、金融数据安全相关、开发测试、成本敏感等的业务场景。以下为您从互联网移动 APP、游戏应用、电商直播教育行业、金融保险企业来介绍 TDSQL-C MySQL 版能够应对这些业务场景的条件和优势。

互联网移动 App

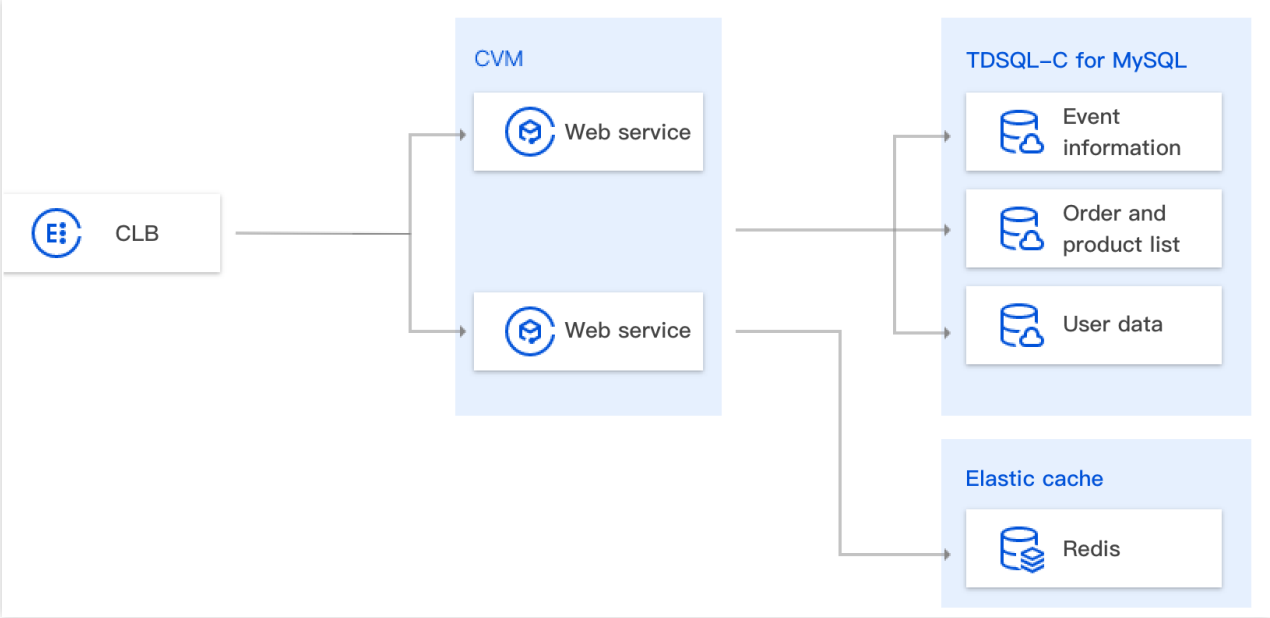
商用数据库级别的高性能、高可靠，定制开发的多项内核优化以及企业级特性保障业务平稳高效的运行，让研发人员专注于业务逻辑的开发，无后顾之忧。

解决了传统主备架构弹性能力差，业务压力大时的同步效率低，主备切换时间不可控等问题，在提供高性能的同时保证了系统的高可用性和业务的连续性。极大的减轻了运营和运维人员的工作量。

全面兼容开源数据库 MySQL，原有业务应用几乎不用更改即可接入 TDSQL-C MySQL 版，助力企业平滑上云。

自带高可用架构，自动维护数据多副本，自动进行数据的校验和修复，减少人工干预，数据可靠性达 99.9999999%。

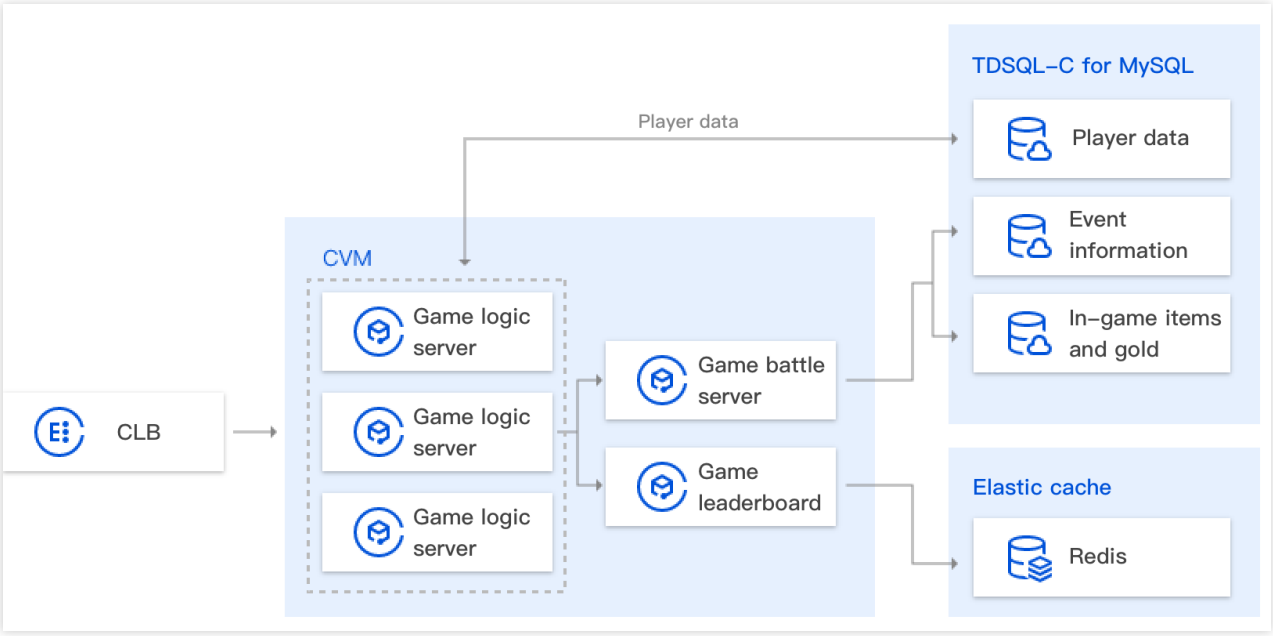
架构图



游戏应用

敏捷灵活的弹性扩展，无需预先购买存储，可根据业务需要快速升降级，快速扩容，轻松应对业务峰值。
最高 PB 级海量存储，按存储量计费，自动扩容，免去合区合服的繁琐操作，实现资源和成本的最优配置。
秒级的快照备份和快速回档能力，在多副本的基础上对用户的数据进行连续保护，是互联网和游戏行业的最佳选择。

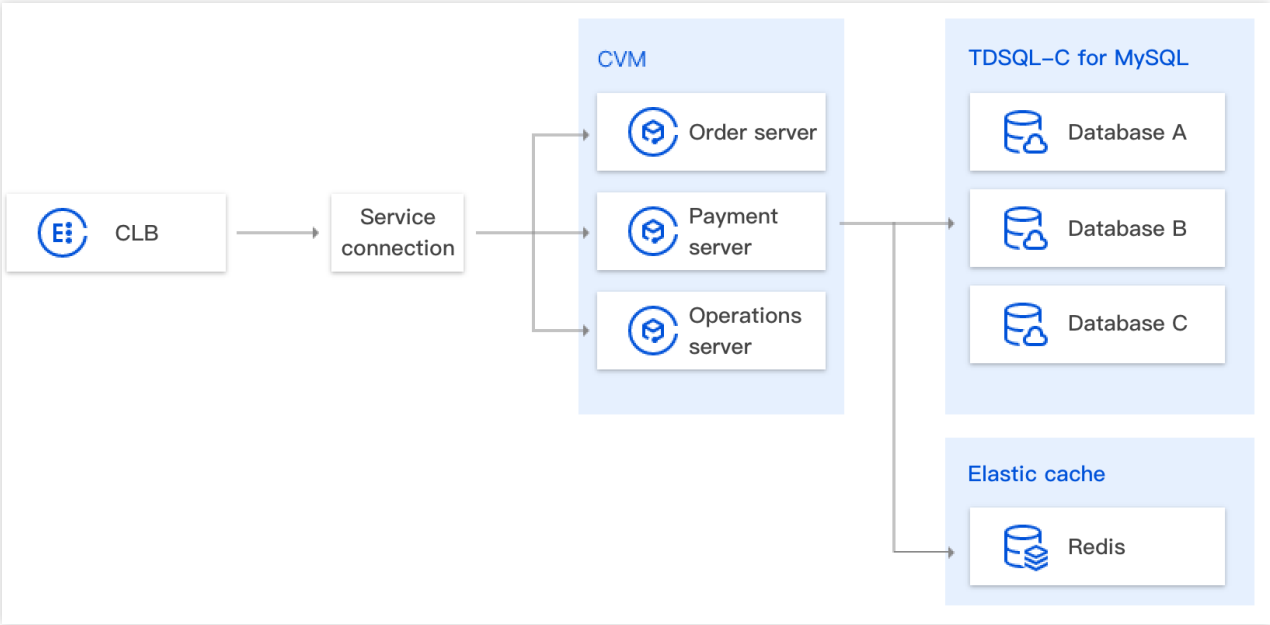
架构图



电商直播教育行业

支持秒级的升配，最多可扩展至15个节点，快速弹升 QPS 的能力。解决传统数据库的升配时间会随着存储量的大小、宿主机资源的情况而不断上升的问题。
通过引擎的优化 IOPS 能力的提升，提供高并发状态下优秀的数据写入能力，轻松应对业务峰值。
读写节点和只读节点之间采用物理复制的方式，只读节点与读写节点延迟极大降低，满足电商场景中买家卖家数据一致性读取需求。

架构图



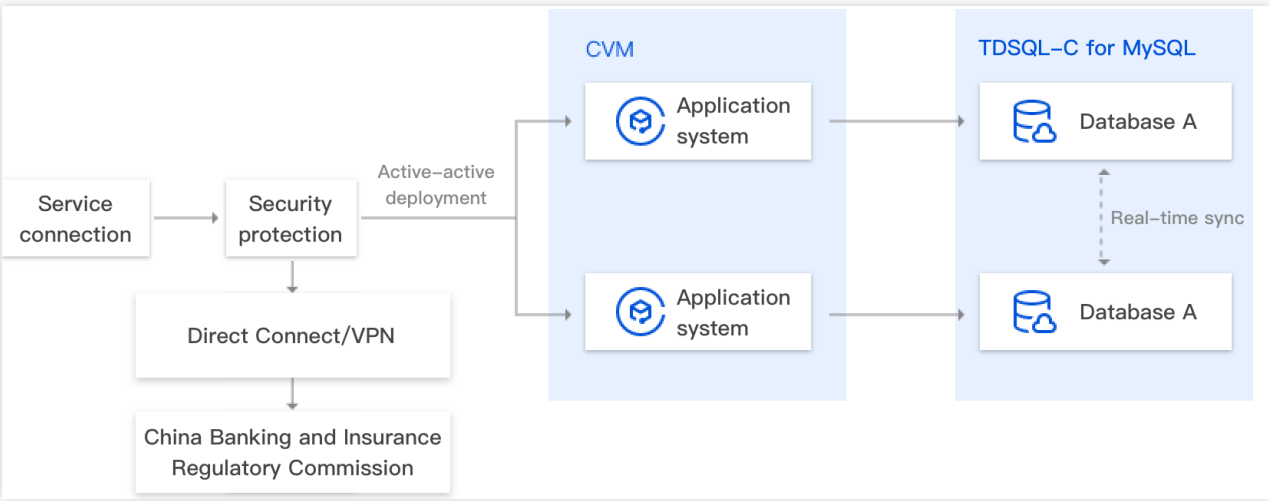
金融保险企业

多可用区架构，在多个可用区内都有数据备份，为数据库提供容灾和备份。

采用白名单、VPC 网络等全方位的手段，对数据库数据访问、存储、管理等各个环节提供安全保障。

采用共享分布式存储的设计，彻底解决了主从（Master-Slave）异步复制所带来的备库数据非强一致性的问题。

架构图



常用概念

最近更新时间：2023-03-01 14:33:46

本文为您介绍产品相关的常用概念，帮助您更好地选购和使用 TDSQL-C MySQL 版。

相关概念

腾讯云控制台：基于 Web 的用户界面。
地域：地域是指物理的数据中心。一般情况下，TDSQL-C MySQL 版实例应该和云服务器实例位于同一地域，以实现最高的访问性能。
可用区：可用区是指在某个地域内拥有独立电力和网络的物理区域。同一地域的不同可用区之间没有实质性区别。
多可用区：是在单可用区的级别上，将同一地域的多个单可用区组合成的物理区域。
读写实例：腾讯云上的 TDSQL-C MySQL 版数据库资源，一个集群中仅支持一个读写实例。
只读实例：仅提供读功能的计算节点，一个集群中可包含0 - 15个只读实例。
计费模式：实例资源的收费方式，分为包年包月、按量计费和 Serverless。
按量计费：后计费购买，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。
包年包月：预付费购买，用户根据自身对云资源的使用需求，一次性支付一个月、多个月或多年的费用。
Serverless：是腾讯云自研的新一代云原生关系型数据库 TDSQL-C MySQL 版的无服务器架构版，支持按实际计算和存储资源使用量收取费用，不用不付费。
实例类型：包括通用型和独享型。
兼容数据库版本：兼容的数据库版本，目前兼容 MySQL 5.7和8.0。
实例规格：每个计算实例的规格配置，例如2核16GB。
项目：用于对实例资源的分类和管理。
标签：是腾讯云提供的云资源管理工具，您可以从不同维度对具有相同特征的云资源进行分类、搜索和聚合，从而轻松管理云上资源。
维护时间：为保证云数据库实例的稳定性，后台系统会不定期对实例进行维护操作的一个时间范围。您可对业务实例设置自己可接受的维护时间，一般设置在业务低峰期，将对业务的影响降到最低。
安全组：对实例进行安全的访问控制，指定进入实例的 IP、协议及端口规则。
网络：由若干节点和连接这些节点的链路构成，表示诸多对象及其相互联系。出于性能安全考虑，目前仅支持私有网络（VPC）。
读写内网地址：用户 VPC 网络内，分配给数据库使用并支持读请求和写请求的 IP 和 port。
只读内网地址：用户 VPC 网络内，分配给数据库使用仅支持读请求的 IP 和 port。
读写外网地址：提供外网访问，支持读请求和写请求的 IP 和 port。
只读外网地址：提供外网访问，仅支持读请求的 IP 和 port。
端口：port，指计算机内部或交换机路由器内的端口。
数据库：是一个长期存储在计算机内的、有组织的、可共享的、统一管理的大量数据的集合。

数据库帐号：用以登录管理数据库的用户名。

字符集：包括字符编码集和字符编码，简单理解就是一个映射关系，一个编码规则。将字符集对应的码点映射为一个个二进制序列，从而使得计算机可以存储和处理。

云服务器：（Cloud Virtual Machine，CVM）是腾讯云提供的可扩展的计算服务。

监控：方便用户查看和掌握实例的运行信息，TDSQL-C MySQL 版提供丰富的性能监控项与便捷的监控功能（自定义视图、时间对比、合并监控项等）。

告警策略：在某些监控指标异常时，创建告警来及时通知您采取措施。告警在一定周期内监控某些特定指标，并根据给定的阈值，每隔若干个时间段通过多种方式（短信、邮件、电话）发送告警通知。

回收站：销毁的实例还未下线时存放的地方，可以在回收站恢复已销毁的实例。

备份：为应对文件、数据丢失或损坏等可能出现的意外情况，将数据单独贮存或形成文件副本保存。

自动备份：目前支持快照备份，通过设置备份的时间实现系统自主保存数据的方式。

手动备份：支持在任意时间通过人为手动去创建备份文件的方式，手动备份暂时仅支持全量备份。

数据库审计：记录对数据库的访问及 SQL 语句执行情况，帮助企业进行风险控制，提高数据安全等级。

使用限制

最近更新时间：2023-07-13 11:05:59

为保障集群稳定及安全的运行，TDSQL-C MySQL 版有部分使用上的约束。本文为您介绍 TDSQL-C MySQL 版的相关使用限制。

引擎限制

TDSQL-C MySQL 版仅支持 Innodb 引擎。

命名限制

限制项	限制说明
集群名	长度小于60字符。 支持输入 `中文/英文/数字/"-"/"_"/"."` 。
读写/只读实例名	长度小于60字符。 支持输入 `中文/英文/数字/"-"/"_"/"."` 。
帐号名	长度为1 - 16个字符。 由字母、数字和特殊字符组成，以字母开头，字母或数字结尾。 特殊字符为 `"_"` 。 不能和已有的帐号名重复。
数据库名	长度最长为64字符。 由小写字母，数字，中划线 `(-)` ，下划线 `(_)` 组成。 字母开头，字母或是数字结尾 。 不能和已有的数据库名重复且创建后数据库名不支持修改。

配额限制

配额	限制
只读实例	一个集群里只读实例的创建个数为0 - 15个。
标签	标签键必须唯一，最大设置20个。每次最多设置50个实例进行批量标签绑定。
备份空间免费	TDSQL-C MySQL 版备份空间暂时不会产生额外费用，后续会根据购买集群时的存储空间设

额度	置备份空间的免费额度，超出免费额度将产生额外费用。
备份保留天数	默认为7天，最大支持1830天。
日志保留天数	默认为7天，最大支持1830天。
项目	项目是以集群为维度归属的，同一集群下的多个实例归属于同一个项目下。

操作限制

限制项	限制说明
内核版本升级	版本升级完成时会涉及集群切换（即秒级数据库连接断开），建议程序有自动重连功能，并且建议选择实例可维护时间内做切换。 单个实例的表数量超过100万后，可能会造成升级失败，同时也会影响数据库监控，请合理规范表的数量，控制单个实例表数量不超过100万。 内核小版本升级后无法降级。
故障切换	当主节点出现故障时，TDSQL-C MySQL 版会切换到备节点。切换过程中有30秒以内的连接闪断，需保证您的业务具有自动重连机制，避免因为切换导致服务不可用。
切换网络	更换网络会导致集群下所有内网 IP 发生变化，系统将自动分配新的 IP 地址，需及时修改客户端程序。 旧的 IP 地址默认24小时后会失效，失效时间可在切换网络操作时设置。当设置为0更换网络后旧 IP 地址会被立即回收。 切换网络时只能选择集群所在地域和可用区内的 VPC 网络与子网。
存储空间	按量计费和 Serverless 集群下的每个计算实例规格存储空间都有上限，具体可分别参考 产品规格 和Serverless 算力配置。 包年包月计费模式下，以购买的存储空间为准。 不同计算实例规格有对应最大存储上限，如需更大的存储，可升级计算实例规格。
数据恢复	建议您在数据恢复前备份好重要数据，以免导致数据丢失。建议您通过回档或克隆集群来恢复数据。
变配	TDSQL-C MySQL 版支持原地快速升降配，特殊情况下连接有可能发生秒级闪断，请确保业务具备重连机制，建议在业务低峰执行此操作。

关键字和保留字限制

关键字指在 SQL 语句中有意义的词。**保留字**指关键字中某些特定的词（如 SELECT、DELETE 或 BIGINT）被保留到数据库对应版本里。这些**保留关键字**需要特殊处理才能作为表名和列名等标识符，如加引号，否则会出现报错，

非保留关键字不做特殊处理就可以作为标识符使用。

TDSQL-C MySQL 版的关键字和保留字与 MySQL 官网基本一致，详见 官网文档 ，便于您更好地执行 SQL。

除了官网列举的关键字和保留字以外，TDSQL-C MySQL 版新增了以下保留关键字：

CLUSTER

THREADPOOL_SYM

使用规范建议
SQL 使用规范建议

最近更新时间：2023-11-01 16:59:33

本文为您介绍在创建 TDSQL-C MySQL 集群后 SQL 使用规范建议。

数据库基本设计规范

所有的字符存储与表示，均以 utf-8 或者 utf8mb4 编码，表和字段需要有注释信息。
尽量避免使用大事务。
说明:
例如在一个事务里进行多个 select 或 update 语句，如果是高频事务，会严重影响 MySQL 并发能力，因为事务持有的锁等资源只在事务 rollback/commit 时才能释放。但同时也要评估数据写入的一致性。

数据库 SQL 查询规范

当使用 ORDER BY .. LIMIT 查询时，优先考虑通过索引优化查询语句，提高执行效率。
使用 ORDER BY、GROUP BY、DISTINCT 执行查询时，where 条件过滤出来的结果集请保持在1000行以内，否则会降低查询效率。
使用 ORDER BY、GROUP BY、DISTINCT 语句时，优先利用索引检索排序好的数据。如 where a=1 order by b 可以利用 key(a,b)。
使用 JOIN 连接查询时，where 条件尽量充分利用同一表上的索引。
说明:
例如，select t1.a, t2.b from t1,t2 where t1.a=t2.a and t1.b=123 and t2.c= 4
如果 t1.c 与 t2.c 字段相同，那么 t1上的索引 (b,c) 就只用到 b。此时如果把 where 条件中的 t2.c=4 改成 t1.c=4，那么可以用到完整的索引。这种情况可能会在字段冗余设计（反范式）时出现。
推荐使用 UNION ALL，减少使用 UNION，需要考虑是否需要对数据进行去重。
使用 UNION ALL 不对数据去重，由于少了排序操作，速度快于使用 UNION，如果业务没有去重的需求，优先使用 UNION ALL。
在代码中实现分页查询逻辑时，若 COUNT 为0应直接返回，避免执行后面的分页语句。
避免频繁对表进行 COUNT 操作。对大数据量表进行 COUNT 操作耗时会较长，一般都是秒级响应速度。如果有频繁对表进行 COUNT 操作的需求，请引入专门的计数表解决。
确定返回结果只有一条时，使用 limit 1。在保证数据无误的前提下，可以确定结果集数量时，尽量使用 limit 查询，尽可能快速返回结果。

评估 DELETE 和 UPDATE 语句效率时，可以将语句改成 SELECT 后执行 explain（explain 命令可以帮助我们分析 SQL 查询语句的执行计划和性能瓶颈）。但需注意，如果频繁执行 SELECT 语句会导致数据库性能慢，因此在使用 explain 命令分析 SQL 查询语句时，需尽量减少 SELECT 语句的执行次数，在分析 SQL 查询语句时，需要综合考虑查询效率和数据库性能，权衡利弊，选择最优的方案。

TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和日志资源少，如果删除的表上没有触发器，且进行全表删除，建议使用 TRUNCATE TABLE。

说明:

TRUNCATE TABLE 不会把删除的数据写到日志文件中。

TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同。

TRUNCATE TABLE 不能和其它 DML 写在同一个事务里。

尽量不要使用负向查询，避免全表扫描。

说明:

使用负向查询是指使用负向运算符，如：NOT, !=, <>, NOT EXISTS, NOT IN 以及 NOT LIKE 等。如果使用负向查询，无法利用索引结构做二分查找，只能做全表扫描。

避免对三个表以上执行 JOIN 连接。需要 JOIN 的字段，数据类型必须保持一致。

多表关联查询时，保证被关联的字段需要有索引；在多表 join 中，尽量选取结果集较小的表作为驱动表，用来 join 其他表。即使双表 join 也要关注表索引、SQL 性能情况。

数据库 SQL 开发规范

对于简单 SQL，优先考虑拆分。

说明:

如 OR 条件：f_phone='10000' or f_mobile='10000'，两个字段各自有索引，但只能用到其中一个。可以拆分成2个 SQL，或者使用 union all。

需要在 SQL 中进行复杂的运算或业务逻辑时，优先考虑在业务层实现。

使用合理的分页方式以提高分页效率，大页情况下不使用跳跃式分页。

说明:

例如有类似下面分页语句：

```
SELECT * FROM table1 ORDER BY ftime DESC LIMIT 10000,10;
```

这种分页方式会导致大量的 IO，因为 MySQL 使用的是提前读取策略。

推荐分页方式：即传入上一次分页的界值。

```
SELECT * FROM table1 WHERE ftime < last_time ORDER BY ftime DESC LIMIT 10;
```

在事务里使用更新语句时，尽量基于主键或 unique key，否则会产生间隙锁，内部扩大锁定范围，导致系统性能下降，产生死锁。

尽量不使用外键与级联，外键概念在应用层处理。

说明:

例如，学生表中的 student_id 是主键，那么成绩表中的 student_id 则为外键。如果更新学生表中的 student_id，同时触发成绩表中的 student_id 更新，则为级联更新。

外键与级联更新适用于单机低并发，不适合分布式、高并发集群；

级联更新是强阻塞，存在数据库更新风暴的风险，外键影响数据库的插入速度。

减少使用 in 操作，in 后的集合元素数量不超过500个。

为了减少与数据库交互的次数，可以适度采用批量 SQL 语句。例如：INSERT INTO … VALUES (XX),(XX),(XX).... (XX); 这里 XX 的个数建议100个以内。

避免使用存储过程，存储过程难以调试和扩展，更没有移植性。

避免使用触发器、事件调度器（event scheduler）和视图实现业务逻辑，这些业务逻辑应该在业务层处理，避免对数据库产生逻辑依赖。

避免使用隐式类型转换。

说明:

类型转换规则具体如下：

1. 两个参数至少有一个是 NULL 时，比较的结果也是 NULL，特殊情况是使用 <=> 对两个 NULL 做比较时会返回 1，这两种情况都不需要做类型转换。
2. 两个参数都是字符串，会按照字符串来比较，不做类型转换。
3. 两个参数都是整数，按照整数来比较，不做类型转换。
4. 十六进制的值和非数字做比较时，会被当做二进制串。
5. 参数是 TIMESTAMP 或 DATETIME，并且另外一个参数是常量，常量会被转换为 timestamp。
6. 有一个参数是 decimal 类型，如果另外一个参数是 decimal 或者整数，会将整数转换为 decimal 后进行比较，如果另外一个参数是浮点数，则会把 decimal 转换为浮点数进行比较。
7. 有其他情况下，两个参数都会被转换为浮点数再进行比较。
8. 如果一个索引建立在 string 类型上，如果这个字段和一个 int 类型的值比较，符合上述第7条。

如 f_phone 定义的类型是 varchar，但 where 语句中使用 f_phone in (098890)，两个参数都会被当成浮点型。这种情况下 string 转换后的 float，导致 MySQL 无法使用索引，导致出现性能问题。

如果是 f_user_id ='1234567' 的情况，符合上述第2条，直接把数字当字符串比较。

业务允许的情况下，事务里包含 SQL 语句越少越好，尽量不超过5个。因为过长的事务会导致锁数据较久，MySQL 内部缓存、连接消耗过多等问题。

避免使用自然连接（natural join）。

说明:

自然连接没有显示定义连接列，而是隐含，会出现难以理解及无法移植问题。

数据库索引设计规范

根据实际业务需求，减少使用无法利用索引优化的 order by 查询语句。Order by、group by、distinct 这些语句较为耗费 CPU 资源。

涉及到复杂 SQL 语句时，优先参考已有索引进行设计，通过执行 explain，查看执行计划，利用索引，增加更多查询限制条件。

使用新的 SELECT、UPDATE、DELETE 语句时，都需要通过 explain 查看执行计划中的索引使用情况，尽量避免 extra 列出现：Using File Sort，Using Temporary。当执行计划中扫描的行数超过1000时，需要评估是否允许上线。

需每日进行慢日志统计分析，处理慢日志语句。

说明:

explain 解读：

type：ALL, index, range, ref, eq_ref, const, system, NULL（从左到右，性能从差到好）。

possible_keys：指出 MySQL 能使用哪个索引在表中找到记录，查询涉及到的字段上若存在索引，则该索引将被列出，但不一定被查询使用。

key：表示 MySQL 实际决定使用的键（索引），如果没有选择索引，键是 NULL。要想强制 MySQL 使用或忽视 possible_keys 列中的索引，在查询中使用 FORCE INDEX、USE INDEX 或者 IGNORE INDEX。

ref：哪些列或常量被用于查找索引列上的值。

rows：根据表统计信息及索引选用情况，估算的找到所需的记录所需要读取的行数。

Extra：

Using temporary：表示 MySQL 需要使用临时表来存储结果集，常见于排序和分组查询。

Using filesort：MySQL 中无法利用索引完成的排序操作称为“文件排序”。

Using index：表示使用索引，如果只有 Using index，说明没有查询到数据表，只用索引表即完成了这个查询，这种情况为覆盖索引。如果同时出现 Using where，代表使用索引来查找读取记录， 也是可以用到索引的，但是需要查询到数据表。

Using where：表示条件查询，如果不读取表的所有数据，或不是仅通过索引就可以获取所有需要的数据，则会出现 Using where。如果 type 列是 ALL 或 index，而没有出现该信息，则您有可能在执行错误的查询，返回所有数据。

在 WHERE 条件列上使用函数，会导致索引失效。

说明:

如 WHERE left(name, 5) = 'zhang'，left 函数会导致 name 上的索引失效。可在业务侧修改该条件，不使用函数。当返回结果集较小时，业务侧过滤满足条件的行。

数据库权限及库表索引规范

最近更新时间：2023-11-01 17:00:20

本文为您介绍在创建 TDSQL-C MySQL 集群后的相关使用规范建议。

数据库权限规范

所有 DDL（例如：创建表，更改表结构等）只有通过评审后，由 DBA 通过数据库管理工具（DMC）执行，在业务低峰期操作上线。
权限需要进行细粒度控制，读写权限分开，运维和开发权限要分开。
DDL 操作保留操作日志。

库表规范

InnoDB 是 MySQL 中支持事务的一种引擎，如果要在 MySQL 数据库中创建表，并且需要使用事务功能，那么必须使用 InnoDB 引擎来创建表，适配 MySQL 的其它引擎不支持事务。
小数类型需使用 decimal 类型来定义，禁止使用 float 和 double。
说明:
float 和 double 在存储的时候，存在精度损失的问题，很可能在值比较的时候得到的结果有误。如果存储的数据范围超过 decimal 的范围，建议将数据拆成整数和小数分开存储。
禁用保留字，如 desc、range、match、delayed 等，请参考 MySQL 官方保留字。
数据表必须有主键，可以使用业务相关，有序且具有唯一性的字段作为主键，也可以使用业务无关的自增长字段作为主键。
说明:
无主键容易导致主库执行速度慢和复制延迟问题。
创建表时需为表中的字段设置默认值，并且将字段设置为 NOT NULL，以避免在插入数据时出现空值或缺失值的情况，数字类型默认值推荐给0，varchar 等字符类型的默认值推荐空字符串，如''。
建议表包含两个字段：create_time，update_time，且均为 datetime 类型。
说明:
数据仓库拖取数据时可以利用这两个统一字段无需询问业务。
在数据库出现意外时可以判断数据进入数据库和修改的时间，在极端情况可以帮助数据恢复的判断。
控制单表字段数量，字段上限50个。
如果存储的字符串长度几乎相等，使用 char 定长字符串类型。
字段允许适当跨表冗余，以避免关联查询，提高查询性能，但必须考虑数据一致。
说明:

冗余字段应遵循：
不是频繁修改的字段。
不是 varchar 超长字段和 text 字段。
合适的存储长度（不建议使用 LONG TEXT，BLOB 等长类型字段），不但节约数据库表空间、节约索引存储，更重要的是提升检索速度。

索引规范

避免因为字段类型不同造成的隐式转换，导致索引失效。
业务上具有唯一特性的字段，即使是多个字段的组合，建议在所有具有唯一特性字段的最小集合上建立唯一索引。
例如：一个表含有 a，b，c，d，e，f 字段，在业务上 ab 和 ef 分别是具有唯一特性的字段集合，那么最好在最小集合 ab 和 ef 上分别建立唯一索引。
说明:
即使在应用层做了完善的校验控制，只要没有唯一索引，就可能会有脏数据产生。
同时需要考虑建立的唯一索引对查询是否真正有帮助，没有帮助的索引可以考虑删除。
需要考虑多建立的索引对插入性能的影响，根据唯一性相关的数据正确性需求，以及性能需求来权衡是不是需要多建立唯一性索引。
尽量在定长的字段（如：INT）上建立索引；在 varchar 字段上建立索引时，必须指定索引长度，无需对全字段建立索引，根据实际文本区分度决定索引长度即可。
说明:
索引长度与区分度是一对矛盾体，一般对字符串类型数据，长度为20的索引区分度会高达90%以上，可以使用 `count(distinct left(列名，索引长度))/count(*)` 的区分度来确定（有区分度的放前面，没有区分度的放后面）。
页面搜索避免使用左模糊（如：SELECT * FROM users WHERE u_name LIKE ‘%hk’）或者全模糊，避免从索引扫描退化为全表扫描，如果需要请在应用层解决。
说明:
索引文件具有 B-tree 的最左前缀匹配特性，如果左边的值未确定，那么无法使用此索引。
利用覆盖索引来进行查询操作，避免回表，但是覆盖索引加的字段不能太多，要兼顾写性能。
说明:
能够建立索引的种类：主键索引、唯一索引、普通索引，而覆盖索引是一种查询的效果，利用 explain 的结果，extra 列会出现：using index。
SQL 性能优化的目标：至少要达到 range 级别，要求是 ref 级别，如果可以是 consts 最好。
创建组合索引的时候，区分度最高的在左边。
单张表的索引数量控制在5个以内，或不超过表字段个数的20%。
创建索引避免有如下误解：
宁滥勿缺。误认为一个查询就需要建一个索引。
宁缺勿滥。误认为索引会消耗空间、严重拖慢更新和新增速度。

不用唯一索引。误认为业务的唯一性一律需要在应用层通过“先查后插”方式能解决。