TDSQL-C for MySQL

Product Introduction

Product Documentation

Contents

Product Introduction

Overview

Last updated：2023-03-01 14:33:46

TDSQL-C for MySQL is a new-generation cloud native relational database developed by Tencent Cloud. It combines the strengths of traditional databases, cloud computing, and cutting-edge hardware technologies to provide elastically scalable database services featuring high performance, security, and reliability, as well as full compatibility with MySQL 5.7 and 8.0. It can deliver a high throughput of over one million QPS and a smart storage capacity up to the petabyte level, fully ensuring data security and reliability.
TDSQL-C for MySQL uses an architecture where computing and storage resources are separated and all compute nodes share the same data. It supports configuration adjustment and disaster recovery within seconds. A single node can sustain millions of QPS, automatically maintain data and backups, and roll back gigabytes of data per second.
TDSQL-C for MySQL delivers high stability, reliability, performance, and scalability like commercial databases while featuring simplicity, openness, and efficient iteration like open-source cloud databases. Its engine is fully compatible with native MySQL, so you can migrate MySQL data to it without modifying any application code or configuration.

Core design concepts

Cloud native: TDSQL-C for MySQL is service-oriented.
TDSQL-C for MySQL is built on Tencent Cloud's existing efficient and stable cloud services. It allows you to quickly build cloud databases featuring high performance, availability, and reliability.
Creative: TDSQL-C separates computing and storage and implements the concept of log as a database.
TDSQL-C for MySQL implements the architecture of "log as a database" and separates computing (CPU and memory) from storage. Through considerable modification of the MySQL kernel, it removes unnecessary feature modules and implements stateless compute nodes, enabling elastic scaling and disaster recovery within seconds for computing resources. Moreover, it is built on Tencent Cloud's distributed cloud storage, empowering the pooling of storage resources.
Comprehensive: TDSQL-C is fully compatible with new open-source databases.
TDSQL-C for MySQL is fully compatible with open-source MySQL and will regularly implement support for new versions. This allows you to smoothly migrate the query, application, and tool capabilities of your existing database almost without modifying the code, which notably reduces the costs and risks of data migration.
Cohesive: TDSQL-C features minimalist software optimizations to release the hardware dividend.
TDSQL-C for MySQL has been effectively improved in performance and stability through optimizations of the database kernel and system architecture. Compared with database products using traditional architectures, it delivers

a better performance under the same hardware configuration, releases the hardware dividend first, and perfectly adapts to the trends in new hardware to maximize the database service efficiency.

Cost-Effective: TDSQL-C doubles the database performance and is pay-as-you-go.

Since cloud computing itself is actually a cost-effective service, TDSQL-C for MySQL can be truly pay-as-you-go and elastically scalable as a cloud database solution that outperforms traditional databases while being less expensive.

Pricing

For more information, go to the TDSQL-C for MySQL purchase page.

Usage

You can create TDSQL-C for MySQL clusters, databases, and accounts in the following ways:

Console: It offers easy-to-use web-based GUIs.

API: All operations in the console can be performed through APIs.

Strengths

Last updated：2023-03-01 14:33:46

This document describes the strengths of TDSQL-C for MySQL.

Full compatibility

TDSQL-C for MySQL separates the computing and storage capabilities of open-source databases. The storage is built on Tencent Cloud's distributed cloud storage service and the computing layer is compatible with open-source MySQL 5.7 and 8.0, so you can smoothly migrate your business without any modifications.

Ultra high performance

The ultra high performance of a single node can sustain millions of QPS, which meets the requirements of high-concurrency and high-performance scenarios, ensures the continuity of key businesses, and provides better read/write separation and read/write scalability.

Massive storage capacity

TDSQL-C for MySQL supports storage capacity at the petabyte level. It eliminates the need for frequent and tedious database sharding and table splitting operations otherwise required in the face of massive amounts of data. It also supports data compression and has been deeply optimized for data search and write.

Fast disaster recovery

Compute nodes are stateless and support failover and recovery within seconds. Even if the physical machine where a compute node resides is down, it can be recovered in less than one minute.

High data reliability

Clusters support security group and VPC network isolation and automatically maintain multiple replicas of data and backups to ensure data security and provide a data reliability of up to 99.9999999% (nine nines).

Elastic scalability

Compute nodes can be quickly upgraded/downgraded and scaled within seconds as needed, which minimizes the costs of computing resources when coupled with elastic storage.

Fast read-only scaling

1–15 read-only compute nodes can be quickly added in a cluster within seconds, enabling fast response to business surges and fluctuations.

Snapshot backup and restoration

The fast snapshot backup feature based on multiple replicas of data provides continuous backup protection for your data and eliminates the need for the sync and migration of backup data in the source-replica architecture. Up to gigabytes of data can be rolled back per second, ensuring the rapid restoration of business data.

Serverless architecture

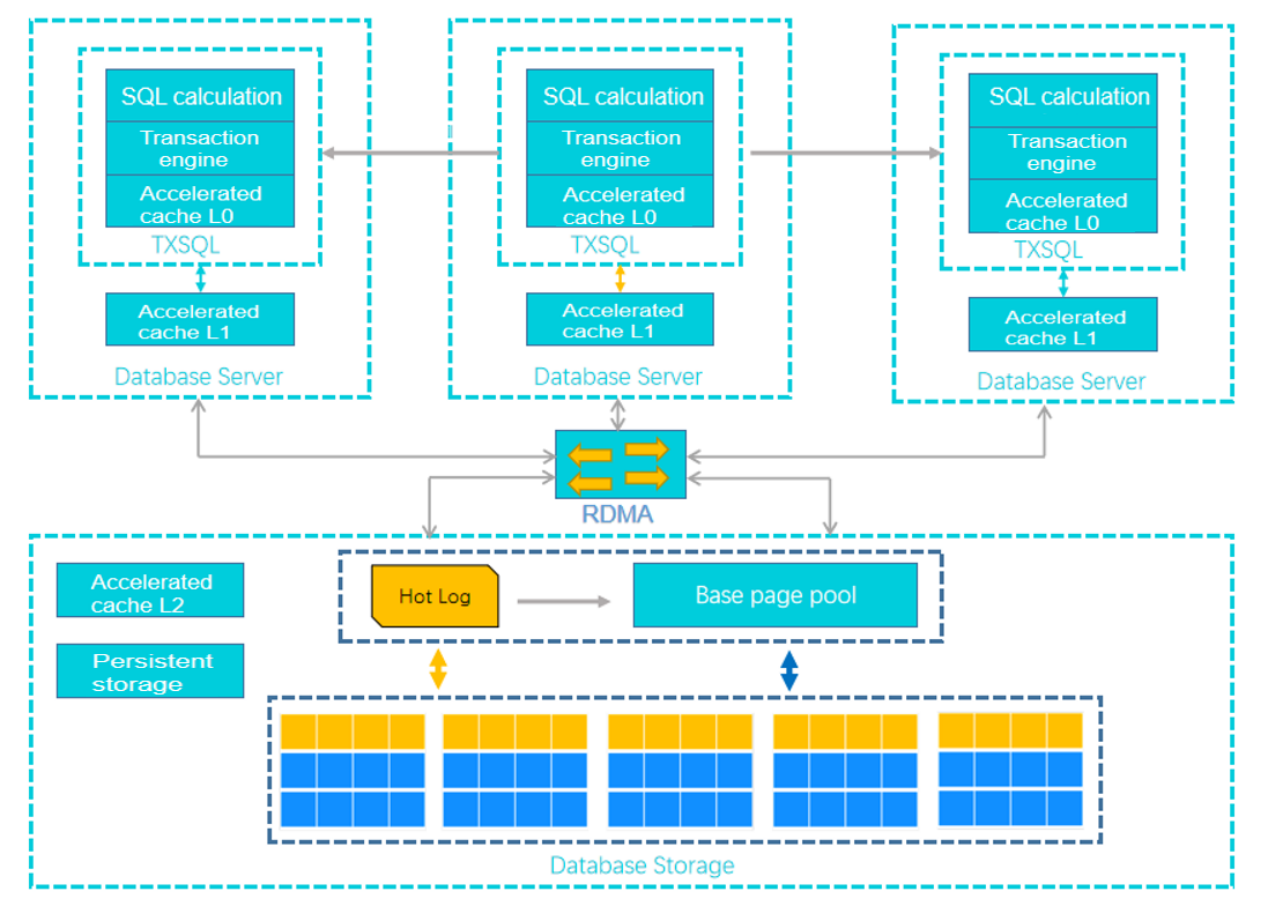
TDSQL-C for MySQL Serverless Edition uses a serverless architecture to provide cloud native database services. It scales automatically and is pay-as-you-go, so you only pay for what you use. This helps you easily cope with dynamic changes and continuous growth of your business data volume.

Architecture

Last updated：2023-03-01 14:33:46

This document describes the architecture and features of TDSQL-C for MySQL. Based on cloud native design, TDSQL-C for MySQL delivers high stability, reliability, performance, and scalability like commercial databases while featuring simplicity, openness, and efficient iteration like open-source cloud databases.

Product architecture diagram



Single-write-multiple-read

A TDSQL-C for MySQL cluster contains one source node and up to 15 read-only nodes. The former processes read and write requests, while the latter processes only read requests.

Computing/Storage separation

TDSQL-C for MySQL separates computing from storage to elastically scale clusters based on business needs in public cloud computing environments. Compute nodes (database engine servers) only store metadata, while remote storage nodes (database storage servers) store data files and redo logs. Only the metadata of redo logs needs to be synced between compute nodes. This greatly reduces the replication delay between the source node and read-only nodes. In addition, when the source node fails, a new node will be quickly started for a smooth switch.

Automatic read/write separation

Automatic read/write separation is a transparent, highly available, and adaptive load balancing feature offered by TDSQL-C for MySQL. After a database proxy address is configured, SQL requests will be automatically forwarded to the nodes in the TDSQL-C for MySQL cluster. This mechanism provides aggregated and high-throughput capabilities to process concurrent SQL requests.

Interconnection through high-speed linkage

TDSQL-C for MySQL supports full-linkage transfer with remote direct memory access (RDMA), where data is transferred from the memory of a computer directly to another computer without intervention from both operating systems required. This further optimizes the system performance at critical paths and minimizes the request delay, so that the I/O performance is no longer a bottleneck. In addition, an RDMA network is also used between stored copies.

Shared distributed storage

Data is shared by multiple compute nodes rather than stored on each compute node. This greatly reduces the storage costs. Based on the newly created distributed block storage and file system, the storage capacity can be smoothly expanded online to sustain petabytes of data without being limited by the storage capacity of a single database server.

Multi-copy strong consistency

Data is stored on the storage nodes of the database in three copies, which ensures a high data reliability. In addition, the strong multi-copy consistency policy is adopted to guarantee the data consistency. Fees are charged based on the data volume in only one copy.

Product Specifications

Last updated：2023-02-07 11:56:37

This document describes the latest and historical specifications of TDSQL-C for MySQL.

Note:

The current specification list may contain some deactivated specifications. Available specifications are listed on the purchase page.

The read-write instance and read-only instances in a TDSQL-C for MySQL cluster have the same specification configuration.

If you need a higher specification to meet your storage needs, submit a ticket for assistance.

Compute node specification

Specifications of monthly subscribed and pay-as-you-go compute nodes:

Compute Node Specification(CPU and Memory)	Supported Max Storage Space (GB)		Max IOPS	I/O Bandwidth
	MySQL 5.7 kernel minor version < 2.0.15MySQL 8.0 kernel minor version < 3.1.2	MySQL 5.7 kernel minor version ≥ 2.0.15MySQL 8.0 kernel minor version ≥ 3.1.2		
1-core 1 GB MEM	1000	3000	8000	1 Gbps
1-core 2 GB MEM	1000	3000	8000	1 Gbps
2-core 4 GB MEM	5000	10000	48000	6 Gbps
2-core 8 GB MEM	5000	10000	48000	6 Gbps
2-core 16 GB MEM	5000	10000	48000	6 Gbps
4-core 8 GB MEM	10000	30000	96000	12 Gbps
4-core 16 GB MEM	10000	30000	96000	12 Gbps
4-core 24 GB MEM	10000	30000	96000	12 Gbps
4-core 32 GB MEM	10000	30000	96000	12 Gbps

8-core 16 GB MEM	10000	50000	216000	27 Gbps
8-core 32 GB MEM	10000	50000	216000	27 Gbps
8-core 48 GB MEM	10000	50000	216000	27 Gbps
8-core 64 GB MEM	10000	50000	216000	27 Gbps
12-core 48 GB MEM	10000	80000	288000	36 Gbps
12-core 72 GB MEM	10000	80000	288000	36 Gbps
12-core 96 GB MEM	10000	80000	288000	36 Gbps
16-core 64 GB MEM	20000	100000	384000	48 Gbps
16-core 96 GB MEM	20000	100000	384000	48 Gbps
16-core 128 GB MEM	20000	100000	384000	48 Gbps
24-core 96 GB MEM	20000	150000	480000	60 Gbps
24-core 144 GB MEM	20000	150000	480000	60 Gbps
24-core 192 GB MEM	20000	150000	480000	60 Gbps
32-core 128 GB MEM	50000	200000	576000	72 Gbps
32-core 192 GB MEM	50000	200000	576000	72 Gbps
32-core 256 GB MEM	50000	200000	576000	72 Gbps

48-core 192 GB MEM	50000	300000	648000	81 Gbps
48-core 288 GB MEM	50000	300000	648000	81 Gbps
48-core 384 GB MEM	50000	300000	648000	81 Gbps
48-core 488 GB MEM	50000	300000	648000	81 Gbps
64-core 256 GB MEM	50000	400000	720000	90 Gbps
64-core 384 GB MEM	50000	400000	720000	90 Gbps
64-core 512 GB MEM	50000	400000	720000	90 Gbps
88-core 710 GB MEM	50000	400000	780000	98 Gbps

Instance Types

Last updated：2023-09-12 15:01:13

This document describes the instance types of TDSQL-C for MySQL, including general and dedicated.

Instance Type	Description
General	A general instance exclusively uses the allocated memory and disk resources and shares the CPU resources with other general instances on the same physical machine. A general instance benefits from higher specifications at a lower cost by sharing CPU resources. A general instance's disk capacity is unaffected by its CPU and memory specs.
Dedicated	A dedicated instance has exclusive access to the CPU, memory, and disk resources (if CPU pinning is enabled). It has long-term stability and is unaffected by the activities of other instances on the physical machine. A dedicated instance with the highest configurations can monopolize a physical machine and all of its resources.

Regions and AZs

Last updated：2023-08-22 15:51:37

TencentDB data centers are hosted in multiple locations worldwide. These locations are known as regions. Each region contains multiple availability zones (AZs).
Each region is an independent geographic area containing multiple isolated AZs. Separate AZs in the same region are connected via low-latency private networks. Tencent Cloud provides you with the ability to distribute Tencent Cloud resources across different locations. We recommend you place resources in different AZs to eliminate single points of failure which may lead to service unavailability.
Region name and AZ name can most directly embody the coverage of a data center. The following naming convention is used for your convenience:
A region name is composed of **region + city**. The `region` indicates the geographic area that the data center covers, while the `city` represents the city in or near which the data center is located.
AZ names utilize the format of **city + number**.

Regions

Tencent Cloud regions are completely isolated. This guarantees the maximum cross-region stability and fault tolerance. When you purchase Tencent Cloud services, we recommend you select the region closest to your end users to minimize access latency and improve download speed. Operations such as launching or viewing instances are performed at the region level.
Private network communication:
Tencent Cloud resources in the same VPC within the same region under the same account can communicate with each other over private network. They can also be accessed at the private network access.
The networks of different regions are fully isolated from each other, and Tencent Cloud services in different regions cannot communicate using private networks by default.
Tencent Cloud services in different regions can communicate with each other through public IPs over the internet, while those in different VPCs can communicate with each other through Cloud Connect Network, which is faster and more stable.
Cloud Load Balancer currently supports intra-region traffic forwarding and being bound to CVM instances in the same region by default. If you enable the cross-region binding feature, a CLB instance can be bound to CVM instances in another region.

AZs

AZs refer to Tencent Cloud's physical data centers that are in the same region. Each AZ is independently powered and have its own network resources. They are designed to ensure that failures within one AZ can be isolated from other AZs, thereby ensuring service availability and business stability and excepting the occurrences of large-scale disasters or major power failures. Users can protect their applications from being affected by failures that occur in a single location by selecting instances in independent AZs.
When launching an instance, you can select any AZ in the specified region. For high reliability, you can adopt a cross-AZ deployment solution to ensure that the service remains available when an instance in a single location fails. Examples of such solutions include CLB and EIP.

List of Regions and AZs

China

Region	AZ
South China (Guangzhou) ap-guangzhou	Guangzhou Zone 4 ap-guangzhou-4
	Guangzhou Zone 6 ap-guangzhou-6
	Guangzhou Zone 7 ap-guangzhou-7
East China (Shanghai) ap-shanghai	Shanghai Zone 2 ap-shanghai-2
	Shanghai Zone 4 ap-shanghai-4
	Shanghai Zone 5 ap-shanghai-5
North China (Beijing) ap-beijing	Beijing Zone 3 ap-beijing-3
	Beijing Zone 5 ap-beijing-5
	Beijing Zone 6 ap-beijing-6
	Beijing Zone 7 ap-beijing-7

North China (Beijing Finance) ap-beijing-fsi	Beijing Finance Zone 1 (only financial institutions and enterprises can contact us to apply for activation) ap-beijing-fsi-1
East China (Nanjing) ap-nanjing	Nanjing Zone 1 ap-nanjing-1
Southwest China (Chengdu) ap-chengdu	Chengdu Zone 1 ap-chengdu-1
Southwest China (Chongqing) ap-chongqing	Chongqing Zone 1 ap-chongqing-1
Hong Kong/Macao/Taiwan (China Region) (Hong Kong, China) ap-hongkong	Hong Kong Zone 2 ap-hongkong-2
	Hong Kong Zone 3 ap-hongkong-3

Other countries and regions

Region	AZ
Southeast Asia Pacific (Singapore) ap-singapore	Singapore Zone 3 ap-singapore-3
	Singapore Zone 4 ap-singapore-4
West US (Silicon Valley) na-siliconvalley	Silicon Valley Zone 1 na-siliconvalley-1
	Silicon Valley Zone 2 na-siliconvalley-2
Europe (Frankfurt) eu-frankfurt	Frankfurt Zone 1 eu-frankfurt-1
	Frankfurt Zone 2 eu-frankfurt-2
Northeast Asia (Seoul) ap-seoul	Seoul Zone 2 ap-seoul-2
East US (Virginia) na-ashburn	Virginia Zone 1 na-ashburn-1

	Virginia Zone 2 na-ashburn-2
Northeast Asia (Tokyo) ap-tokyo	Tokyo Zone 1 ap-tokyo-1
	Tokyo Zone 2 ap-tokyo-2

Selection of Regions and AZs

When you purchase Tencent Cloud services, we recommend you select the region closest to your end users to minimize access latency and improve download speed.

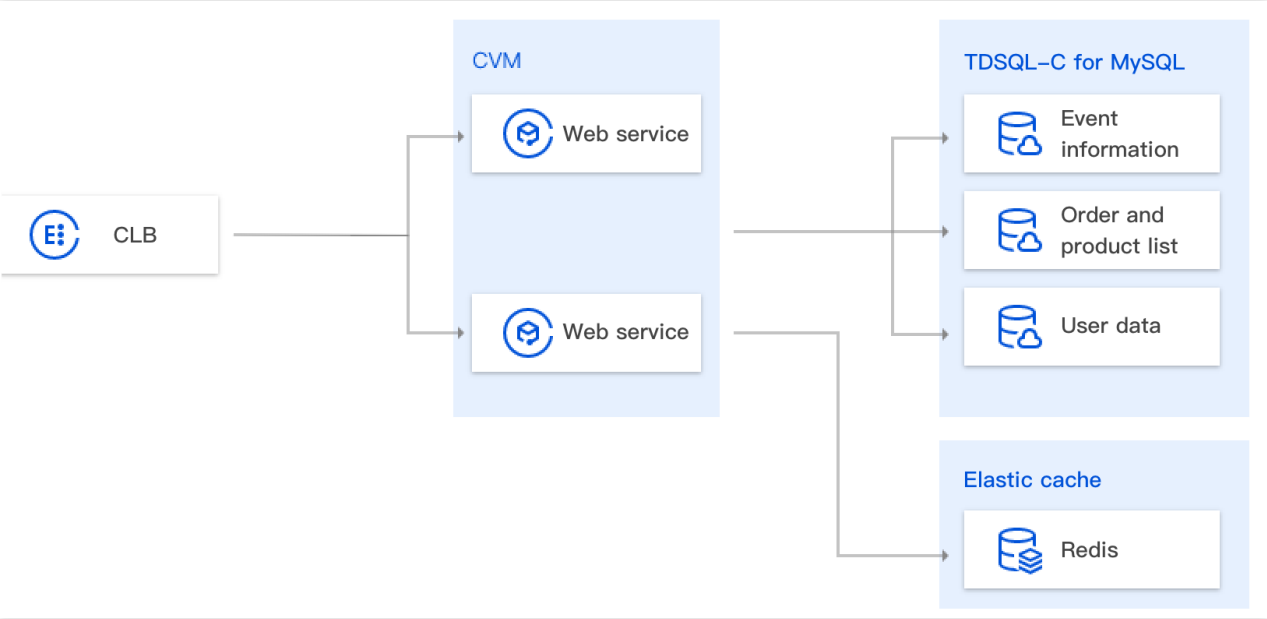
Use Cases

Last updated：2023-03-01 14:33:46

TDSQL-C for MySQL is highly elastic and performant database service that features excellent storage capacity, security, and reliability. It helps you easily tackle your needs in diverse business scenarios, such as frequent transactions like commodity orders, fast-growing businesses with traffic peaks, gaming, data-heavy infrequent queries like historical orders, financial data security protection, development, testing, and cost-sensitive businesses. TDSQL-C for MySQL offers the following benefits and strengths to suit you in the best possible way.

Mobile internet application

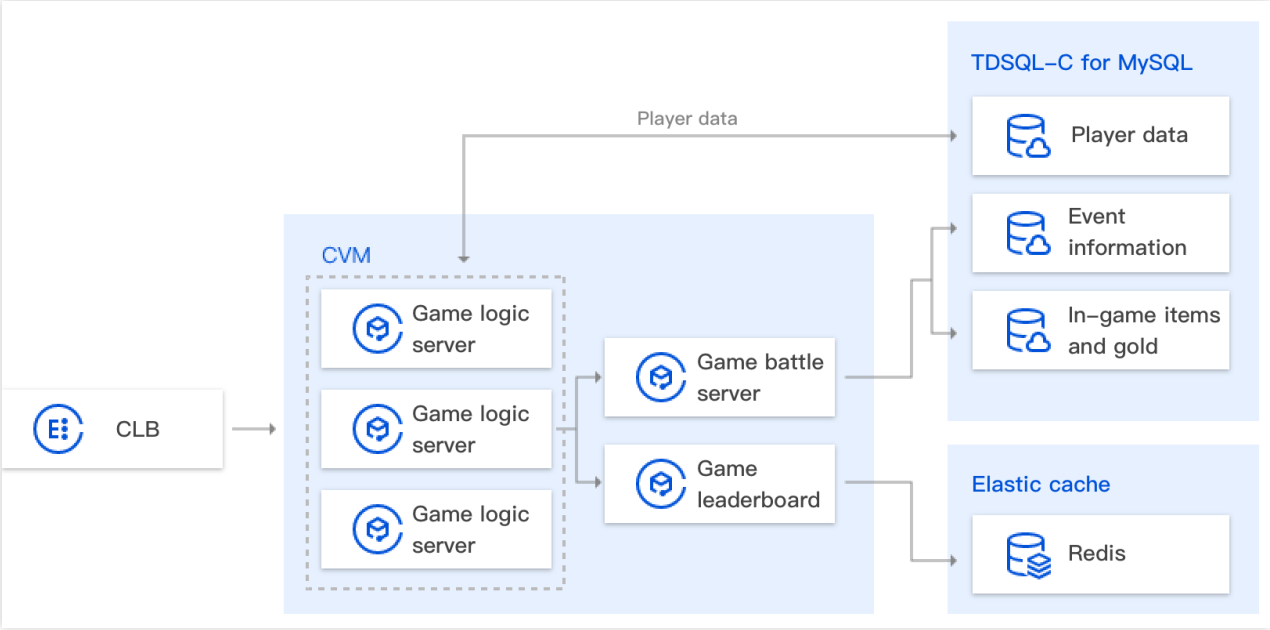
It delivers commercial database-grade high performance and reliability. Its multiple custom-developed kernel optimizations and enterprise-grade features ensure smooth and efficient business operations, so that R&D personnel can focus on the development of business logic without worries.
It solves the problems of poor flexibility, low sync efficiency under high business pressure, and uncontrollable switch time in the traditional source-replica architecture. It ensures high system availability and business continuity while delivering high performance, which greatly reduces the workload of operations and Ops personnel.
It is fully compatible with open-source MySQL, so that your existing business applications can be connected with almost no changes required, helping you cloudify your business smoothly.
It has a built-in high availability architecture and automatically maintains multiple data replicas, checks data, and fixes errors, which reduce human intervention and achieve a data reliability of up to 99.9999999% (nine nines).
Architecture diagram



Game application

It has agile and flexible scalability, which means that you don't need to purchase storage capacity in advance. It can quickly upgrade/downgrade according to your business needs for fast scaling and easy response to business peaks.
It offers a massive storage capacity for up to petabytes of data that can be automatically expanded. It eliminates the tedious operations of region and server merges and optimizes resource allocation and costs.
It features fast snapshot backup and rollback, which provides continuous protection for your data based on multiple replicas, making it an ideal database in the internet and gaming industries.

Architecture diagram



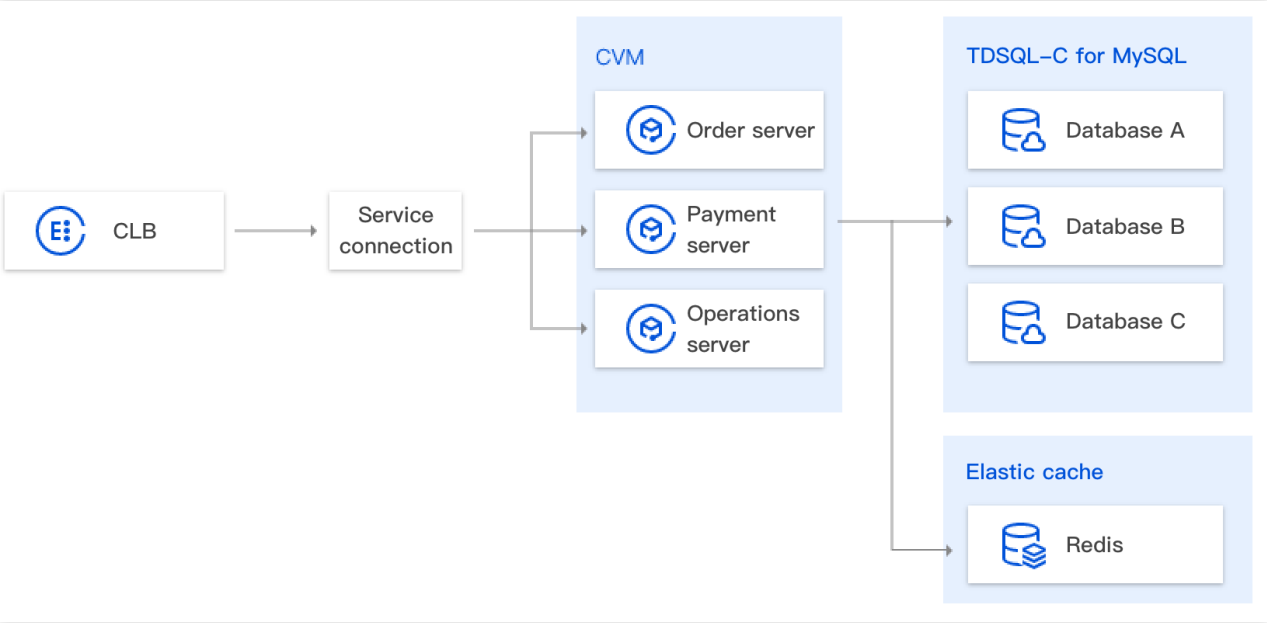
Ecommerce/Education live streaming

It can be upgraded within seconds and expanded to 15 nodes at most to quickly increase the QPS, as compared with traditional databases where configuration upgrade takes longer as the storage capacity and host resource usage grow.

It delivers an excellent data write performance under high concurrency thanks to its engine optimizations and IOPS capability enhancement, helping you tackle business surges easily.

It leverages physical replication between the read-write and read-only nodes, which greatly reduces the delay and ensures the data consistency between buyers and sellers in ecommerce scenarios.

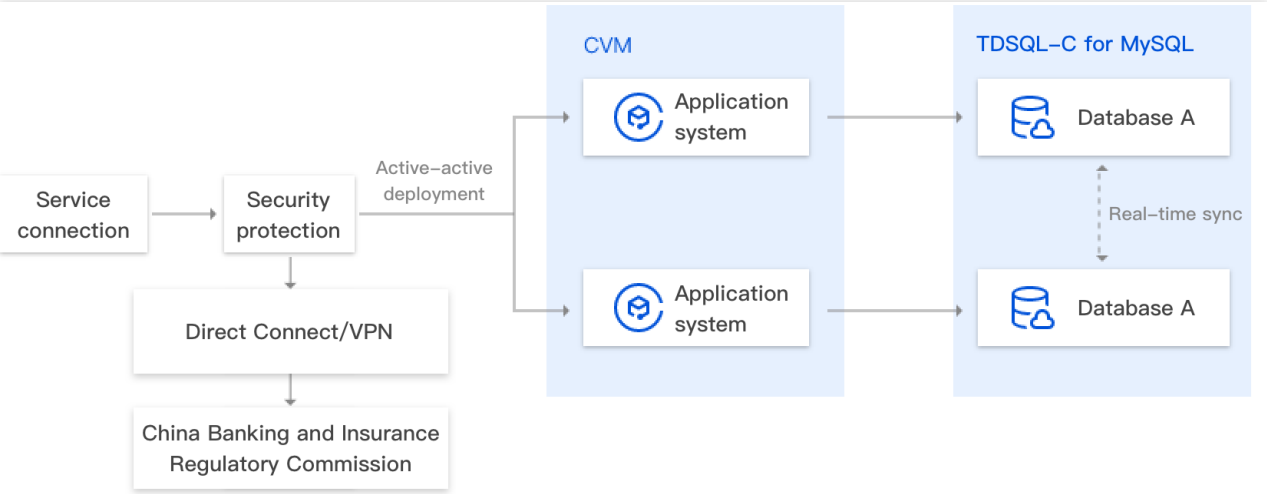
Architecture diagram



Finance and insurance

It is deployed in a multi-AZ architecture for disaster recovery and backup, with data backups stored in multiple AZs.
It adopts comprehensive methods such as allowlist and VPC to provide security guarantees for various links in database data access, storage, and management.
It uses the shared distributed storage architecture, which completely resolves the problem of poor data consistency in replica databases caused by async source-replica replication.

Architecture diagram



Common Concepts

Last updated：2023-03-01 14:33:46

This document describes the common concepts of TDSQL-C for MySQL to help you better understand and use it.

Concepts

Tencent Cloud console: Web-based UIs.
Region: A physical IDC. In general, a TDSQL-C for MySQL instance and a CVM instance should be in the same region to achieve the best access performance.
Availability zone (AZ): A physical location with independent power supply and network resources within a region. There are no substantial differences between different AZs in the same region.
Multi-AZ: A physical location created by combining multiple AZs in the same region.
Read-write instance: A TDSQL-C for MySQL database resource in Tencent Cloud. A cluster can have only one read-write instance.
Read-only instance: A compute node that can only be read from. A cluster can have 0–15 read-only instances.
Billing mode: The billing mode of an instance resource, which can be monthly subscription, pay-as-you-go, or serverless.
Pay-as-you-go: A postpaid billing mode, where you can apply for resources for on-demand use and will be charged based on the actual usage upon settlement.
Monthly subscription: A prepaid billing mode, where you need to pay the fees for one or multiple months or even years based on your need for cloud resources.
Serverless: TDSQL-C for MySQL Serverless adopts Tencent Cloud's proprietary serverless architecture for next-gen cloud-native relational database services. It is billed based on the actual computing and storage resource usage, so you only need to pay for what you use.
Instance type: General or dedicated.
Compatible database version: MySQL 5.7 and 8.0 currently.
Instance specification: The specification configuration of a compute instance, such as 2-core 16 GB MEM.
Project: Used to categorize and manage instance resources.
Tag: A cloud resource management tool that allows you to use different standards to categorize, search for, and aggregate cloud resources with the same attributes.
Maintenance time: To ensure the stability of your TencentDB instance, the backend system performs maintenance operations on the instance during the maintenance window from time to time. We highly recommend you set an acceptable maintenance time for your business instance, usually during off-peak hours, so as to minimize the potential impact on your business.
Security group: Security access control to instances by specifying IP, protocol, and port rules for instance access.

Network: A network made up of several nodes and linkages that connect them. It represents many objects and their interconnections. For performance and security considerations, only VPC network is supported currently.
Private read-write address: The IP and port assigned to a database for both read and write requests within your VPC network.
Private read-only address: The IP and port assigned to a database for read requests only within your VPC network.
Public read-write address: The IP and port that provide public network access and support both read and write requests.
Public read-only address: The IP and port that provide public network access and support read requests only.
Port: A port in a computer, switch, or router.
Database: A set of organized, shared, and centrally managed data that is stored on a computer for a long period.
Database account: A username used to log in to and manage a database.
Character set: A mapping relationship or encoding rule, including a coded character set and character encoding. The code points corresponding to a character set are mapped into binary sequences, so that they can be stored and processed by a computer.
Cloud Virtual Machine (CVM): A scalable computing service provided by Tencent Cloud.
Monitoring: To make it easier for you to view and stay up to date with instance conditions, TDSQL-C for MySQL provides a wide variety of performance monitoring metrics and convenient monitoring features (such as custom view, time comparison, and merged monitoring metrics).
Alarm policy: You can create alarms to stay informed of the status changes of certain metrics. The specific metrics will be monitored for a certain period of time, and alarm notifications will be sent by SMS, email, and phone at specified intervals based on the given threshold.
Recycle bin: A place where terminated instances are stored before elimination. Such instances can be restored.
Backup: Data is stored separately or as a file copy to tackle possible unexpected situations such as file or data loss or corruption.
Automatic backup: Currently, snapshot backup is supported. You can set the backup time for the system to automatically save data.
Manual backup: You can manually create backup files at any time. Manual backup is supported only for full backup.
Database audit: It can record accesses to databases and executions of SQL statements to help you manage risks and improve the database security.

Use Limits

Last updated：2023-07-13 11:05:59

This document describes the TDSQL-C for MySQL use limits that you must follow to ensure the stable and secure operation of your cluster.

Engine Limits

TDSQL-C for MySQL only supports the InnoDB engine.

Naming Limits

Limit	Description
Cluster name	The length must be less than 60 characters. It can contain letters, digits, hyphens, underscores, and dots.
Read-write/read-only instance name	The length must be less than 60 characters. It can contain letters, digits, hyphens, underscores, and dots.
Account name	The length ranges from 1 to 16 characters. It must be a combination of letters, digits and special characters, which must start with a letter and end with a letter or digit. The special symbols are underscores and dots. It cannot be the same as an existing account name.
Database name	The length can be up to 60 characters. It must be a combination of lowercase letters, digits, hyphens, and underscores. It must start with a letter and end with a letter or digit. It cannot be the same as an existing database name and cannot be changed after creation.

Quota Limits

Quota	Limit

Read-only instance	A cluster can have 0–15 read-only instances.
Tag	Tag keys must be unique. There can be up to 20 tags. A tag can be bound to up to 50 instances at a time.
Free tier of backup space	TDSQL-C for MySQL backup space currently is free of charge. In the future, a free tier of backup space will be set based on the storage space selected during cluster purchase, and any excessive usage will incur fees.
Backup retention period	Backups can be retained for 7 (default) to 1,830 days.
Log retention period	Logs can be retained for 7 (default) to 1,830 days.
Project	A project is defined on a cluster basis, and multiple instances in the same cluster belong to the same project.

Operation Limits

Limit	Description
Kernel version upgrade	Cluster switch will be required after version upgrade is completed (that is, the database may be disconnected for seconds). We recommend that you use applications configured with automatic reconnection feature and conduct the switch during the instance maintenance time. If the number of tables in a single instance exceeds one million, upgrade may fail and database monitoring may be affected. Make sure that the number of tables in a single instance is below one million. The kernel minor version cannot be downgraded once upgraded.
Failover	When the source node fails, TDSQL-C for MySQL will switch to a replica node. As a disconnection for less than 30 seconds will occur during the switch, make sure that your business has an automatic reconnection mechanism.
Network switch	Switching the network may cause all of the instance's private IPs to change. The system will automatically assign new IPs, so you need to modify the instance IPs on the client promptly. The original IPs will expire after 24 hours by default. The expiration time can be customized during the network switch. If it is set to zero, the original IPs will be repossessed immediately after the network switch. When switching the network, you can only select a new VPC and subnet in the same region and AZ where the instance resides.

Storage space	There is a limit on the storage space of each pay-as-you-go or serverless compute instance. For more information, see Product Specifications and Compute Unit. The storage space in the monthly subscription billing mode is as purchased. The compute instance specification has a storage upper limit. If you need more storage space, upgrade the specification.
Data recovery	We recommend that you back up important data to avoid data loss before restoring data. You can do so through rollback or cluster cloning.
Configuration adjustment	TDSQL-C for MySQL supports fast in-place configuration upgrade or downgrade, and a momentary disconnection may occur in special cases. Make sure that your business has an automatic reconnection mechanism. We recommend that you perform this operation during off-peak hours.

Keywords/Reserved Words Limits

A **keyword** refers to a word that has a meaning in a SQL statement. A **reserved word** refers to a certain word in a keyword (such as SELECT, DELETE, and BIGINT) that is reserved in the corresponding version of the database. These **reserved words/keywords** require special processing (such as by adding quotation marks) before they can be used as identifiers like table name and column name; otherwise, errors will occur. In contrast, non-reserved words/keywords can be directly used as identifiers.

The keywords and reserved words in TDSQL-C for MySQL are basically the same as those in MySQL as listed in Keywords and Reserved Words.

In addition, TDSQL-C for MySQL has the following additional reserved words and keywords:

CLUSTER

THREADPOOL_SYM

Suggestions on Usage Specifications
SQL Usage Specifications

Last updated：2023-11-01 16:59:49

This document describes the SQL usage specifications and suggestions after a TDSQL-C for MySQL cluster is created.

Basic database design specifications

All characters are stored and represented in UTF-8 or utf8mb4 encoding. Tables and fields must have comments.
Avoid using large transactions.
Note：
For example, if you run multiple SELECT or UPDATE statements in a frequently executed transaction, the concurrency capabilities of MySQL will be compromised severely, as the resources such as locks held by the transaction can be released only when the transaction is rolled back or committed. In addition, the consistency of data writes also needs to be assessed.

Database SQL query specifications

When using `ORDER BY .. LIMIT` queries, optimize the query statements through index to improve the efficiency.
Keep the number of rows in the result set filtered by a WHERE condition in ORDER BY, GROUP BY, and DISTINCT queries below 1,000; otherwise, the query will be inefficient.
Use an index to directly retrieve the sorted data for ORDER BY, GROUP BY, and DISTINCT statements. For example, you can use `key(a,b)` for `where a=1 order by b`.
When using JOIN queries, use indexes in the same table in the WHERE condition as much as possible.
Note：
For example, in the statement `select t1.a, t2.b from t1,t2 where t1.a=t2.a and t1.b=123 and t2.c= 4`:
If the `t1.c` and `t2.c` fields have the same value, only `b` in the index `(b,c)` in `t1` is used. If you change `t2.c=4` in the WHERE condition to `t1.c=4`, you can use the complete index. Such cases may occur in field redundancy design (denormalization).
We recommend that you use UNION ALL to reduce the use of UNION. You need to consider whether to deduplicate the data.

As UNION ALL doesn't deduplicate and sort the data, it runs faster than UNION. If your business doesn't need deduplication, use UNION ALL preferentially.
When you implement the paginated query logic in your code, the result should be returned directly if COUNT is 0. This avoids executing subsequent pagination statements.
Avoid frequently performing the COUNT operation on tables, as it takes a long time (generally seconds) on big tables. If you need to frequently perform COUNT, use a dedicated counter table.
If you are sure that there is only one returned result, use `LIMIT 1` . If you can determine the number of results provided that the data is correct, use LIMIT queries as much as possible to return results more quickly.
You can change the DELETE and UPDATE statements to SELECT, and perform EXPLAIN to evaluate their performance. EXPLAIN command can help you analyze the execution plan and the performance bottleneck of SQL query statements. However, it should be noted that frequent execution of SELECT statements may result in slow database performance. Therefore, minimize the number of SELECT statement executions when using the EXPLAIN command to analyze SQL queries. When analyzing SQL query statements, you need to consider the query efficiency and database performance comprehensively, and choose the best solution.
TRUNCATE TABLE runs faster and uses fewer system and log resources than DELETE. It is recommended if you need to delete an entire table with no triggers.
Note：
TRUNCATE TABLE doesn't write the deleted data to the log file.
TRUNCATE TABLE acts the same as DELETE with no WHERE clauses.
TRUNCATE TABLE cannot be written in the same transaction as other DML statements.
Avoid using negative queries and scanning full tables.
Note：
A negative query refers to a query using negative operators such as NOT, !=, <>, NOT EXISTS, NOT IN, and NOT LIKE. Negative queries cannot perform binary searches by using the index structure but can scan full tables.
Avoid using JOIN to join more than three tables. The fields to be joined must have the same data type.
In multi-table join queries, make sure that joined fields have an index. Select the table with the smallest result set as the driving table to join other tables in a multi-table join. You need to keep an eye on the table index and SQL performance even for dual-table join.

Database SQL development specifications

Split simple SQL statements first
Note：
For example, in the OR condition `f_phone='10000' or f_mobile='10000'` , the two fields have their respective index, but only one will be used. In this case, you can split it into two SQL statements or use UNION ALL.
If you need to implement complex computing or business logic in SQL, consider implementing it at the business layer.
Use an appropriate pagination method to improve the pagination efficiency. Do not use skip paging for large pages.

Note：

For example, if the pagination statement is as follows:

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

A high number of I/O operations will be generated, as MySQL uses the read-ahead policy.

We recommend that you pass in the limit value of the last pagination for the current pagination.

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

Write update statements based on the primary key or unique key in transactions; otherwise, gap locks will be generated, and the lock scope will be expanded internally, which will compromise the system performance and generate deadlocks.

Do not use foreign keys and cascading. Implement the foreign key logic at the application layer.

Note：

For example, as `student_id` is the primary key in the student table, it is a foreign key in the score table. If the update of `student_id` in the student table triggers the update of `student_id` in the score table, then it is a cascading update.

Foreign keys and cascading updates are suitable for a standalone instance with a low concurrency but not a distributed cluster with a high concurrency.

Cascading updates have risks of database update storms, and foreign keys slow down insertions into the database.

Reduce IN operations. Keep the number of elements in the set after IN below 500.

To reduce interactions with the database, use batch SQL statements, like `INSERT INTO … VALUES (XX), (XX),(XX)....(XX);` . Here, we recommend that you keep the number of `XX` items below 100.

Avoid using stored procedures, as they are hard to debug and extend and have no portability.

Avoid using triggers, event schedulers, and views to implement the business logic, which should be processed at the business layer to prevent logic dependency on the database.

Avoid using implicit type conversion.

Note：

Below are the specific conversion rules:

1. When at least one parameter between the two is NULL, the comparison result will also be NULL. A special case is that when <=> is used to compare two NULL values, `1` will be returned. However, in both cases, such type conversion is not needed.
2. If both parameters are strings, they will be compared as strings without type conversion.
3. If both parameters are integers, they will be compared as integers without type conversion.
4. When a hexadecimal value is compared with a non-numeric value, it will be treated as a binary string.
5. If a parameter is of TIMESTAMP or DATETIME type, and the other is a constant, the constant will be converted into a TIMESTAMP value.
6. If a parameter is a decimal, and the other is an integer, the integer will be converted into a decimal for comparison. If the other parameter is a floating-point number, the decimal will be converted into a floating-point number for comparison.

7. In other cases, both parameters will be converted into floating-point numbers for comparison.
8. If an index is created for a string field, and the field is compared with an INT value, the rule 7 will apply.

If the type defined by `f_phone` is VARCHAR but `f_phone in (098890)` is used in the WHERE clause, both parameters will be converted into FLOAT values. In this case, FLOAT values converted from strings will make MySQL unable to use indexes and compromise the performance.

If `f_user_id ='1234567'` is used, the rule 2 will apply, and the number will be directly treated as a string for comparison.

Use as few SQL statements in a transaction as possible if the business permits. Keep the number of statements below 5, as a long transaction will cause problems such as long data lock, internal MySQL cache, and excessive connection consumption.

Avoid using natural joins.

Note：

Natural joins don't display or define the joined columns but imply them, so they may be hard to understand or cannot be ported.

Database index design specifications

Reduce the use of ORDER BY query statements that cannot be optimized through an index based on the actual business needs, as the ORDER BY, GROUP BY, and DISTINCT statements consumes a lot of CPU resources.

If complex SQL statements are involved, design them by referring to existing indexes, execute EXPLAIN to view execution plans, and use indexes to add more query restrictions.

When using a new SELECT, UPDATE, or DELETE statement, you need to use EXPLAIN to view the index usage in the execution plan. Try preventing the `Extra` column from displaying `Using File Sort` and `Using Temporary`. If the number of rows scanned in the execution plan exceeds 1,000, assess whether to allow launching the code. You also need to collect and analyze the statistics of slow logs daily to handle slow log statements promptly.

Note：

Notes on EXPLAIN:

types: ALL, index, range, ref, eq_ref, const, system, NULL (the farther to the left, the higher the performance).

possible_keys: It refers to which index can be used by MySQL to find the record in the table. If a field involved in the query has an index, the index will be listed but won't necessarily be used for query.

key: It indicates the key (index) that MySQL actually decides to use. If no indexes are selected, the key will be NULL. To force MySQL to use or ignore indexes in the `possible_keys` column, use `FORCE INDEX`, `USE INDEX`, or `IGNORE INDEX` in the query.

ref: It indicates which columns or variables are used to query the values in the index column.

rows: It indicates the number of rows to be read to find the required records estimated based on the table statistics and index usage.

Extra:

Using temporary: It indicates that MySQL needs to use a temp table to store the result set, which is often seen in sorting and paginated queries.
Using filesort: Filesort refers to a sorting operation that cannot be completed by using an index in MySQL.
Using index: It indicates that an index is used. If only `Using index` is displayed, no data tables are queried, and only an index table is used to complete the query. This is called covering index. If `Using where` is also displayed, an index is also used to search for and read the records, in which case data tables need to be queried.
Using where: It indicates a conditional query. If the entire table is not read, or not all required data can be obtained through indexes, `Using where` will be displayed. If the `type` column is `ALL` or `index`, and this prompt is not displayed, you may be executing an incorrect query, and all data will be returned.
If a function is used in the WHERE condition column, indexes will become invalid.

Note：

For example, in `WHERE left(name, 5) = 'zhang'`, the `left` function will invalidate the indexes for `name`. You can modify the condition in your business to stop using the function. If the returned result set is small, you can also filter eligible rows in your business.

Database Permissions and Index Specifications

Last updated：2023-11-01 17:00:37

This document describes the usage specifications and suggestions after a TDSQL-C for MySQL cluster is created.

Database permission specifications

All DDL operations (such as creating tables and modifying table structures) can only be performed by DBAs through Database Management Center (DMC) during off-peak hours after approval.
Permissions should be managed in a fine-grained manner by separating read, write, Ops, and development permissions.
DDL operations logs should be retained.

Database and table specifications

InnoDB is a transactional storage engine in MySQL. It is required for table creation and transaction feature in MySQL.Other MySQL engines .don't support transaction.
The decimal type must be DECIMAL. FLOAT or DOUBLE cannot be used.
Note：
FLOAT and DOUBLE values may lose their precision and cause rounding errors when stored. If a value to be stored is out of the range of DECIMAL, split the value into INTEGER and DECIMAL parts and store them separately.
The following reserved words cannot be used: DESC, RANGE, MATCH, and DELAYED. For more information, see Keywords and Reserved Words.
A data table must have a primary key, which can be either a business-relevant ordered unique field or a business-irrelevant auto-increment field.
Note：
The lack of the primary key can easily cause slow source database execution and replication delay.
Define table fields as NOT NULL and set default values when creating a table to avoid inserting null or missing values. We recommend that you set the default value to `0` for the field of numeric types, and empty string like '' for that of character types like varchar.
We recommend that you make each table contain two DATETIME fields: `create_time` and `update_time`.
Note：
You can get the required data from a data warehouse based on these two fields without consulting the business team. When an exception occurs in the database, you can use these two fields to determine the time when the data is

inserted and updated or determine whether to restore data in extreme cases.

Keep the number of fields in a single table below 50.

If the lengths of stored strings are almost the same, use fixed-length CHAR strings.

Provided that the data consistency is ensured, cross-table redundant fields are allowed to avoid correlated subqueries and improve the query performance.

Note：

Redundant fields must comply with the following rules:

The fields are not frequently modified.

The fields are not large VARCHAR or TEXT.

Data types with a proper storage length can accelerate search while saving the database tablespace and index storage space. LONG TEXT and BLOB are not recommended.

Index specifications

Use the same field type to prevent implicit conversion from causing invalid indexes.

We recommend that you create a unique index for all minimum sets of fields with uniqueness in your business, even if they are field combinations.

For example, if a table contains fields `a`, `b`, `c`, `d`, `e`, and `f`, and field combinations `ab` and `ef` have uniqueness, then we recommend you create unique indexes for `ab` and `ef` respectively.

Note：

Even if complete verification control is implemented at the application layer, dirty data may be generated as long as there is no unique index.

Before creating a unique index, consider whether it is indeed helpful to the query. Useless indexes can be deleted.

Assess the impact of extra indexes on the INSERT operation performance. Determine whether to create unique indexes based on the requirements for the correctness and performance of data with uniqueness.

Create indexes on fixed-length fields such as INT fields. When creating an index on a VARCHAR field, you must specify the index length, but you don't need to create an index on the entire field; instead, determine the index length based on the actual text distinction.

Note:

The index length and distinction are a pair of contradictions. Generally, for strings, the distinction of an index with a length of 20 bytes will be higher than 90%. The distinction formula is `count(distinct left(column name, index length))/count(*)`. Place the column names with a high distinction on the left.

If possible, do not use left fuzzy searches (such as `SELECT * FROM users WHERE u_name LIKE '%hk'`) or full fuzzy searches on pages; otherwise, index scan may downgrade to full-table scan.

Note:

An index file has the leftmost prefix match feature of B-tree. If the value on the left is not determined, the index cannot be used.

Use a covering index to query data and avoid returning to the table. However, do not add too many fields to the covering index; otherwise, the write performance will be compromised.

Note:

Types of indexes that can be created include primary key, unique, and normal indexes. A covering index indicates that if you execute an EXPLAIN statement for query, `using index` will be displayed in the `Extra` column.

Optimize the SQL performance as follows: `range` (minimum level), `ref` (basic level), and `consts` (maximum level).

When creating a composite index, place the column with the highest distinction on the left.

Keep the number of indexes in a single table below 5 or 20% of the number of table fields.

Avoid the following misunderstandings when creating indexes:

Indexes should be frequently used. An index needs to be created for a query.

Indexes should be as few as possible. Indexes take up the space and slow down updates and insertions.

Unique indexes are not needed. Business uniqueness must be implemented at the application layer in the "query first and insert later" method.