TDSQL for MySQL

Product Introduction

Product Documentation

Contents

Product Introduction

Overview

Last updated：2024-01-06 17:31:28

Overview

TDSQL for MySQL is a distributed database service deployed in Tencent Cloud that supports automatic sharding (horizontal splitting) and the Shared Nothing architecture. With a distributed database, your business obtains a complete logical database table which is split and distributed evenly across multiple physical shard nodes on the backend. TDSQL deploys the master-slave architecture by default and provides a full set of solutions for disaster recovery, backup, restoration, monitoring, and migration, making it ideal for storing terabytes to petabytes of data.

Features

OLTP

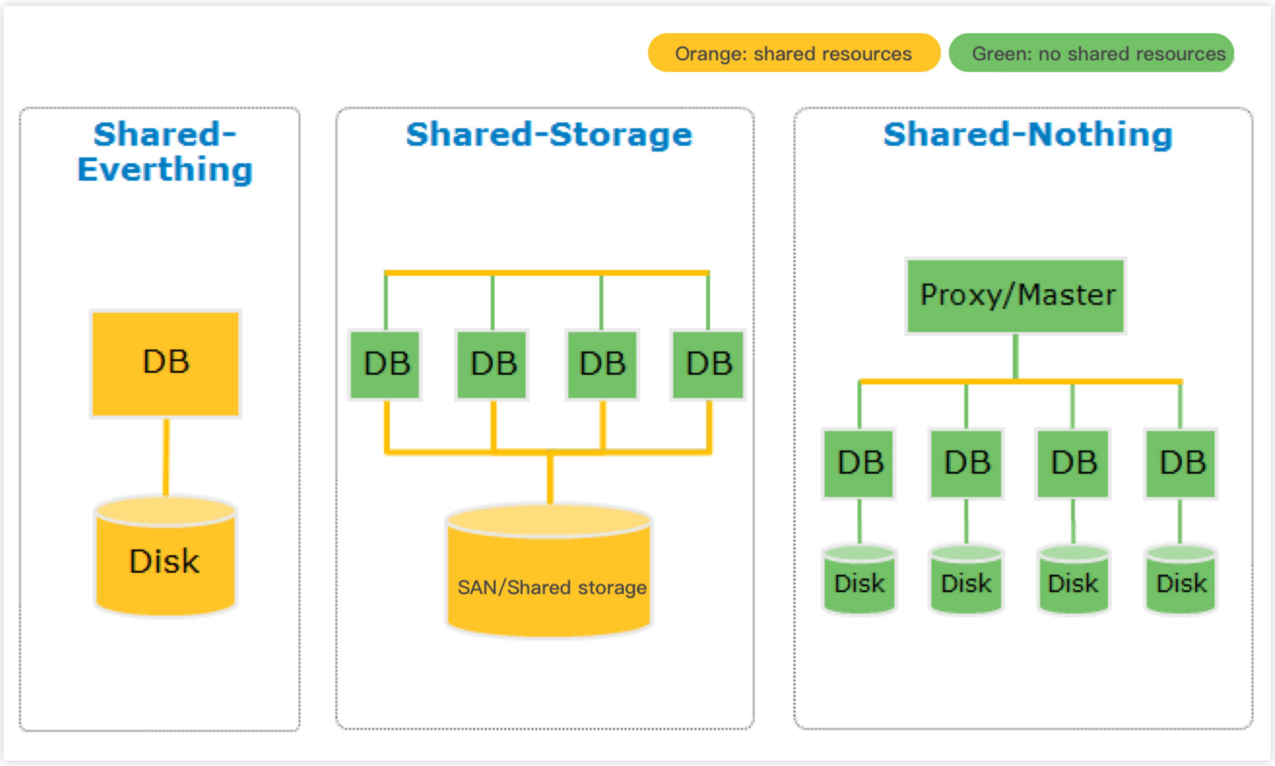
TDSQL is a distributed database for OLTP businesses.

Sharding

TDSQL is a distributed database that supports sharding.
Sharding is to spread the data of a table into multiple independent physical database servers according to a defined rule to form an "independent" database "shard". Multiple shards together form a logically complete database instance.

Shared Nothing architecture

The traditional solution uses minicomputers and shared storage, which is more expensive and prone to capacity and performance bottlenecks. TDSQL adopts the Shared Nothing architecture, with each node computing and storing a portion of data. Therefore, no matter how quickly your business grows, you just need to keep adding servers in the distributed cluster to meet the growing computing and storage needs.



Data splitting methods (sharding rules)

In principle, TDSQL uses a sharding scheme based on automatic horizontal splitting. Specifically, a modulo operation is executed on the shardkey, and then data is distributed into different databases through TProxy according to the specific range of values after modulo operation.

A relational database is a two-dimensional model. To shard data, it is usually necessary to find a **shardkey field** to determine the sharding dimension. Then, a rule needs to be defined to actually shard the database.

Below are some common shardkey options:

Based on date order, such as sharding by year (one shard for 2015 and another for 2016).

Advantages: Simple and easy to find.

Disadvantages: The server performance for the current (e.g., 2016) hot data may be insufficient, while the storage performance for cold data is idle.

Based on user ID modulo, where fields in the specific range after modulo operation are spread across different databases.

Advantages: The performance is relatively balanced and all data of the same user is in the same database.

Disadvantages: This may lead to data skew (for example, when a merchant system is designed, one merchant's data in an ecommerce mall may be more than that of thousands of small merchants).

Based on primary key modulo, where fields in the specific range after modulo operation are spread across different databases.

Advantages: The performance is relatively balanced, data skew seldom occurs, and all data of the same primary key is in the same database.

Disadvantages: Data is randomly distributed, and some business logics may require cross-shard join that is not supported directly.
Before sharding multiple tables, the following options are available:
Noshard: No sharding.
-tableshard: When each table is sharded, select shardkeys arbitrarily for sharding based on the actual needs regardless of inter-table relationships.
groupshard: A few correlated tables are designed based on the same shardkey, so that the related data can be aggregated into one physical node.
In terms of sharded data source management, there are currently two modes:
Client mode: The data sources of multiple shards are managed by the configuration in the business program module, and the reading, writing, and data integration of the shards are performed within the business program.
Middleware proxy mode: A middleware proxy is built on the frontend of the sharding databases which are imperceptible to the frontend application.

Problems TDSQL Can Help You Solve

Performance bottlenecks in standalone databases

Faced with millions of users of the internet-based business, a standalone database will reach bottlenecks in data storage capacity, access capacity, and disaster recovery due to hardware and software limitations as the business grows.

Heavy workload for application-layer sharding development

Application-layer sharding highly couples business logic with database logic, which incurs heavy development workload over rapid iteration of the current business. Based on the imperceptible sharding scheme of TDSQL, your developers only need to modify the code during initial access without having to care much about the database logic during subsequent iterations, which can greatly reduce the development workload.

Problems with open-source solutions or NoSQL

Choosing open-source or NoSQL solutions can also break through database bottlenecks at no or relatively low costs. However, you need to pay attention to the following issues with such solutions:
1. Bug fixing of a product depends on the progress in the community. Can you wait if you encounter a serious bug?
2. Are there members in your team who are familiar with the product and can continuously maintain it without affecting the project in case of staffing changes?
3. Is the associated system ready?
4. What business do you focus on? Does your business metric system involve inputting resources to ensure the open-source product's resource and lifecycle management, distributed logic, high-availability deployment and switchover, disaster recovery and backup, self-service OPS, and troubleshooting?

Strengths

Last updated：2024-01-06 17:31:28

Ultra high performance

A single shard can sustain up to 240,000 QPS, and the performance of an entire instance can be linearly improved as the number of shards increases.
Computing/Storage resources can be independently and imperceptibly scaled, and a single instance can offer exabytes of storage capacity.
Each node at the computing layer can be read and written, and a single instance can easily sustain tens of millions of QPS.
Ultra high compression ratio (up to 20:1) is supported for storage, which is especially suitable for business scenarios with a high number of writes and a low number of reads.

Professional and reliable services

TDSQL has been massively verified by various types of Tencent's core products for over 10 years, such as social networking, ecommerce, payment, audio and video services.
It has comprehensive features of data backup, disaster recovery, and quick upgrade.
It boasts a complete monitoring and alarming system, where most of failures can be recovered through automated program processing.
It supports data encryption by AES and Chinese SM4 algorithms to meet the compliance requirements of static data encryption.
It supports various cutting-edge features in the distributed database world, such as distributed multi-table join, small table broadcast, distributed transaction, and SQL passthrough.
It achieves a high availability of up to 99.95% and a high data reliability of up to 99.99999% (seven nines).

Ease of use

Except for a small number of syntax differences from native MySQL and MariaDB, TDSQL can be used just like a standalone database, and the sharding process is imperceptible to the business and requires no manual intervention.
It is compatible with MySQL protocols (i.e., supporting kernels such as MySQL and MariaDB).
It provides a web-based console, read/write separation, and dedicated Ops commands.

Use Cases

Last updated：2024-01-06 17:31:28

Note:
TDSQL for MySQL is currently only applicable to OLTP business scenarios, such as transaction systems and frontend systems. It is not applicable to OLAP-based systems such as ERP and BI.

Large Applications (Real-time Transaction with Ultra-high Concurrency)

Service providers from ecommerce, finance, O2O, social networking, retail, and SaaS industries generally have the following problems restricting their business development: a large user base (million-level or higher), frequent marketing campaigns, and slower response from core trading system databases.
TDSQL for MySQL features linearly horizontal scalability that can increase the processing power of databases in real time for improved access efficiency. It can sustain over 15 million QPS at the peak, enabling you to cope with high-concurrence real-time transaction scenarios. For example, WeChat Pay, Tenpay, and Tencent Top-up are all supported by databases using the TDSQL for MySQL architecture.

IoT Data (Storage and Access of Petabytes of Data)

In typical IoT scenarios such as industrial monitoring and remote control, smart city, smart home, and IoV, a large number of sensing and monitoring devices, high sample rate, and massive storage capacity are required. Generally, petabytes to exabytes of data may be generated for storage per year. Traditional x86 server architecture and open-source database solutions cannot store or use such a huge amount of data at all.
TDSQL for MySQL features horizontal capacity expansion, helping you store massive amounts of data at lower costs than shared storage solutions.

File Index (Instant Access to Trillions of Data Rows)

Generally, a cloud service platform contains hundreds of millions to trillions of rows of image, document, and video data. It usually needs to store the indexes of such files to databases and perform real-time CRUD operations at the index level.
As the service platform sustains access requests from customers, extremely high service quality and performance are required. Traditional databases cannot support such access and use. TDSQL for MySQL boasts ultra-high

performance, scalability, and strong sync capabilities, effectively ensuring the service quality and data consistency on the platform.

Cost-effective Commercial Database Solution

In order to support massive data storage and high-concurrence database access, large enterprises, and banks are extremely dependent on minicomputers and high-end storage. Internet companies can achieve the same or even higher capabilities of commercial databases with low-cost x86 servers and open-source software.
TDSQL for MySQL is suitable for various scenarios such as national or provincial business system aggregation, large ecommerce and channel platforms, internet banking services, and trading systems.

Basic Principles

Sharding

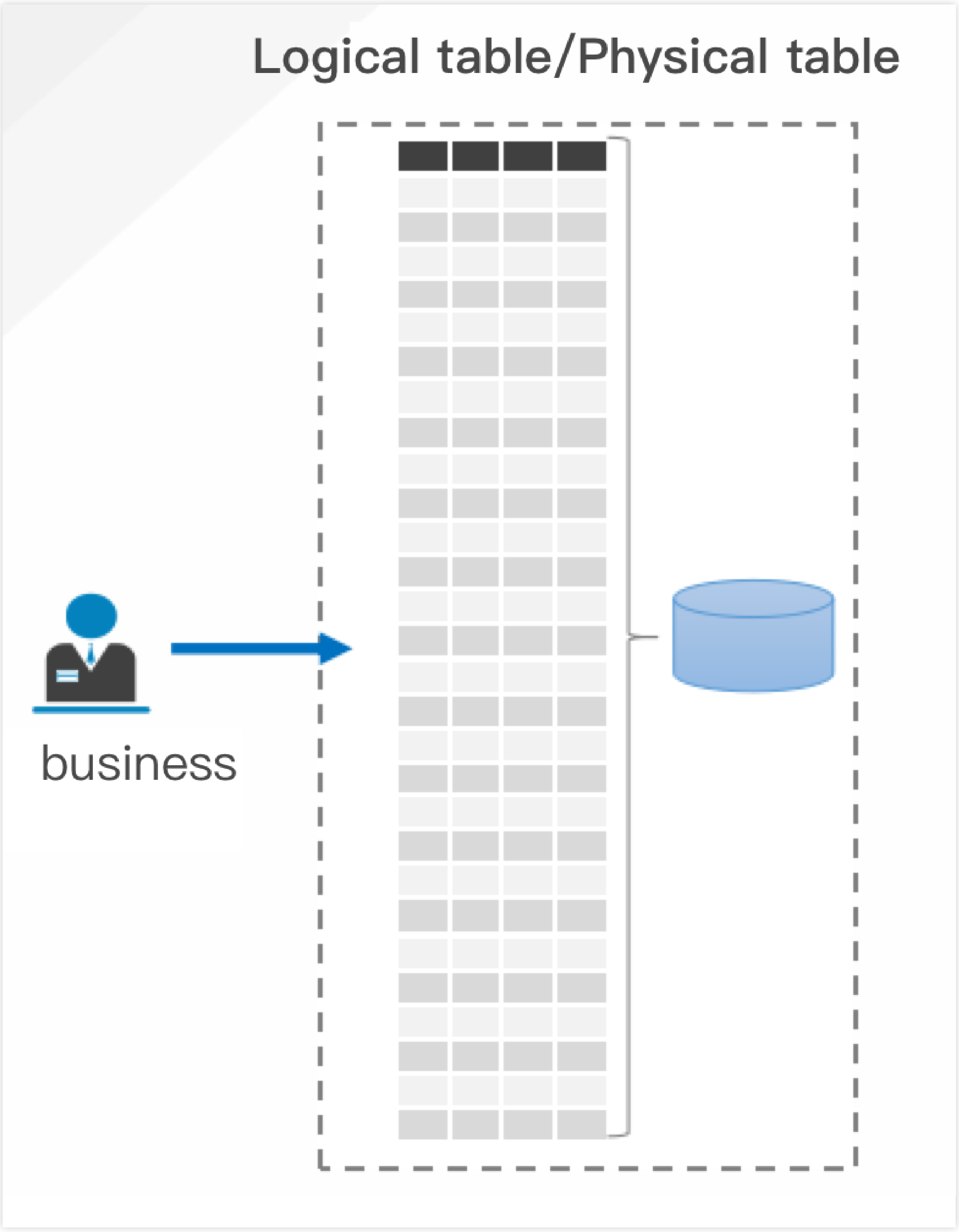
Last updated：2024-01-06 17:31:28

Overview

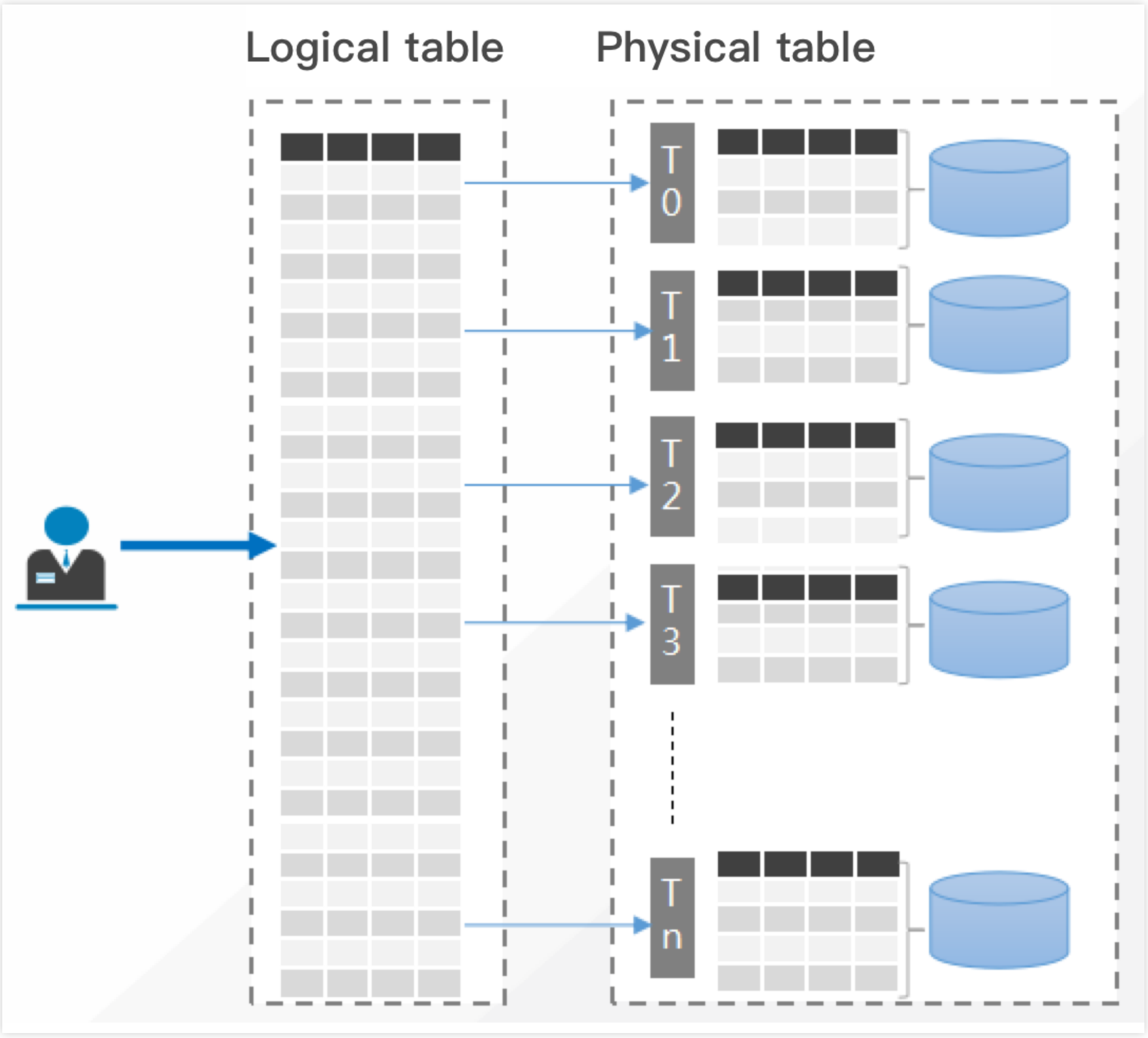
Horizontal partitioning (sharding) is actually how TDSQL for MySQL works. Each node takes part in computing and storing data, but only for a part of data. Therefore, no matter how your business grows, you can simply keep adding devices to the distributed cluster to meet the growing computing and storage needs.

Horizontal Sharding

Sharding means spreading the data of a table into multiple independent physical database servers according to a defined rule to form an "independent" database "shard". Multiple shards together form a logically complete database instance.
In a conventional standalone database, reads and writes of a complete table are done on only one physical storage device.



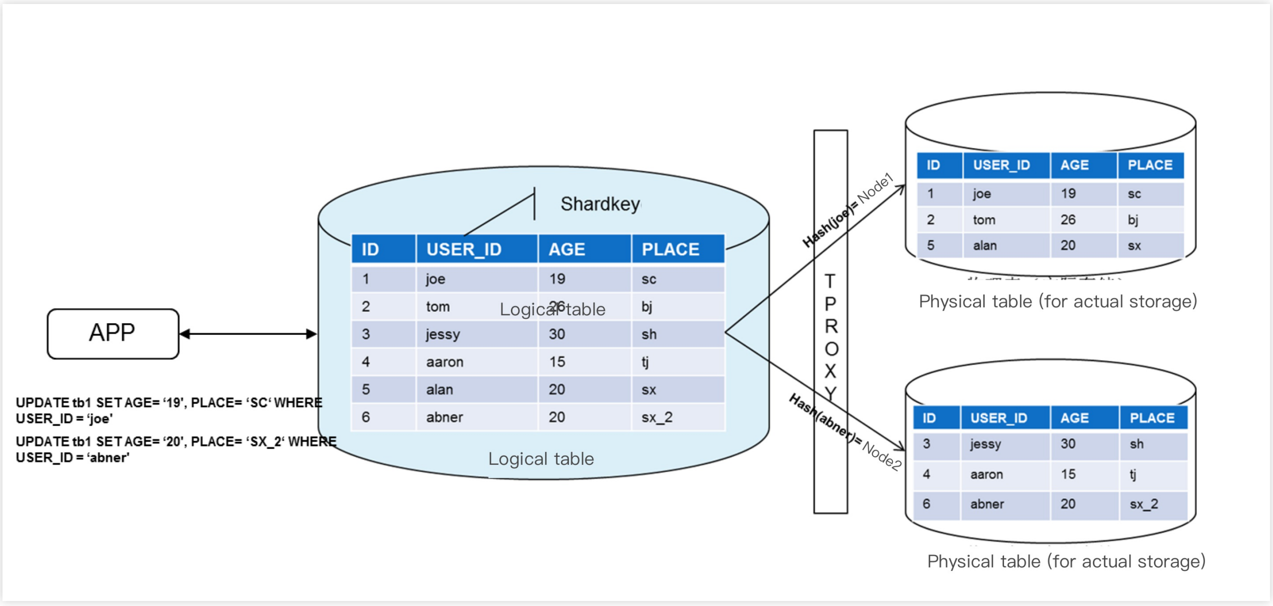
In a distributed database, based on the shardkey set during table creation, the system automatically distributes data to different physical shards, but the table is still logically complete.



In TDSQL for MySQL, a shardkey is usually required to determine the sharding dimension, and then sharding is performed by using a field modulo operation (HASH) scheme. The field used for HASH calculation is the shardkey. The HASH algorithm can virtually guarantee relatively even distribution of data in different physical devices.

Data writes (shardkey included in SQL statement)

1. The business writes a row of data.
2. The gateway hashes the shardkey to get its hash value.
3. Different hash value ranges correspond to different shards (determined by the pre-sharding algorithm of the scheduling system).
4. Data is stored in the corresponding shard according to the sharding algorithm.


APP
Shardkey
Logical table
T P R O X Y
Hash(joe)= Node1
Hash(abner)= Node2
Physical table (for actual storage)
Physical table (for actual storage)

```
UPDATE tb1 SET AGE= '19', PLACE= 'SC' WHERE USER_ID = 'joe'
UPDATE tb1 SET AGE= '20', PLACE= 'SX_2' WHERE USER_ID = 'abner'
```

ID	USER_ID	AGE	PLACE
1	joe	19	sc
2	tom	26	bj
3	jessy	30	sh
4	aaron	15	tj
5	alan	20	sx
6	abner	20	sx_2

ID	USER_ID	AGE	PLACE
1	joe	19	sc
2	tom	26	bj
5	alan	20	sx

ID	USER_ID	AGE	PLACE
3	jessy	30	sh
4	aaron	15	tj
6	abner	20	sx_2

Data Aggregation

Data aggregation: If the data of a query SQL statement involves multiple sharded tables, the statement will be routed to all these tables for execution. TDSQL for MySQL will aggregate the data returned by each sharded table based on the original SQL semantics and return the final result.

Note:

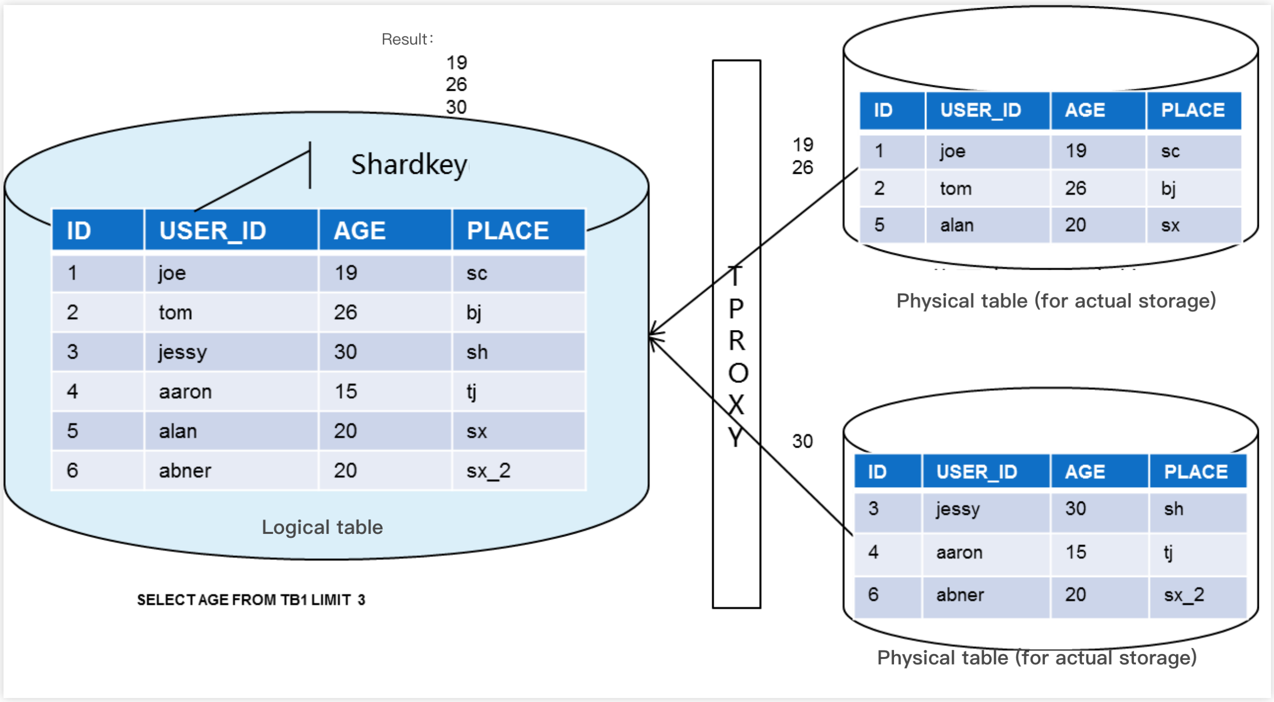
When you execute a SELECT statement, we recommend that you include the `shardKey` field; otherwise, the gateway may aggregate the execution results after a full-table scan which is slow and affects the performance significantly.

Data reads (specific shardkey included):

1. When the business sends a SELECT request which includes the shardkey, the gateway will perform a hash calculation on the shardkey.
2. Different hash value ranges correspond to different shards.
3. Data is obtained from the corresponding shard according to the sharding algorithm.

Data reads (specific shardkey excluded)

1. If the business sends a SELECT request without the shardkey, the request will be sent to all shards.
2. The shards send back data to the proxy after querying their own content.
3. The proxy aggregates the data according to SQL rules and then responds to the gateway.


Result：
19
26
30
Shardkey
Logical table
SELECT AGE FROM TB1 LIMIT 3
PROXY
19
26
30
Physical table (for actual storage)
Physical table (for actual storage)

ID	USER_ID	AGE	PLACE
1	joe	19	sc
2	tom	26	bj
3	jessy	30	sh
4	aaron	15	tj
5	alan	20	sx
6	abner	20	sx_2

ID	USER_ID	AGE	PLACE
1	joe	19	sc
2	tom	26	bj
5	alan	20	sx

ID	USER_ID	AGE	PLACE
3	jessy	30	sh
4	aaron	15	tj
6	abner	20	sx_2

Read/Write Separation

Last updated：2024-01-06 17:31:28

Overview

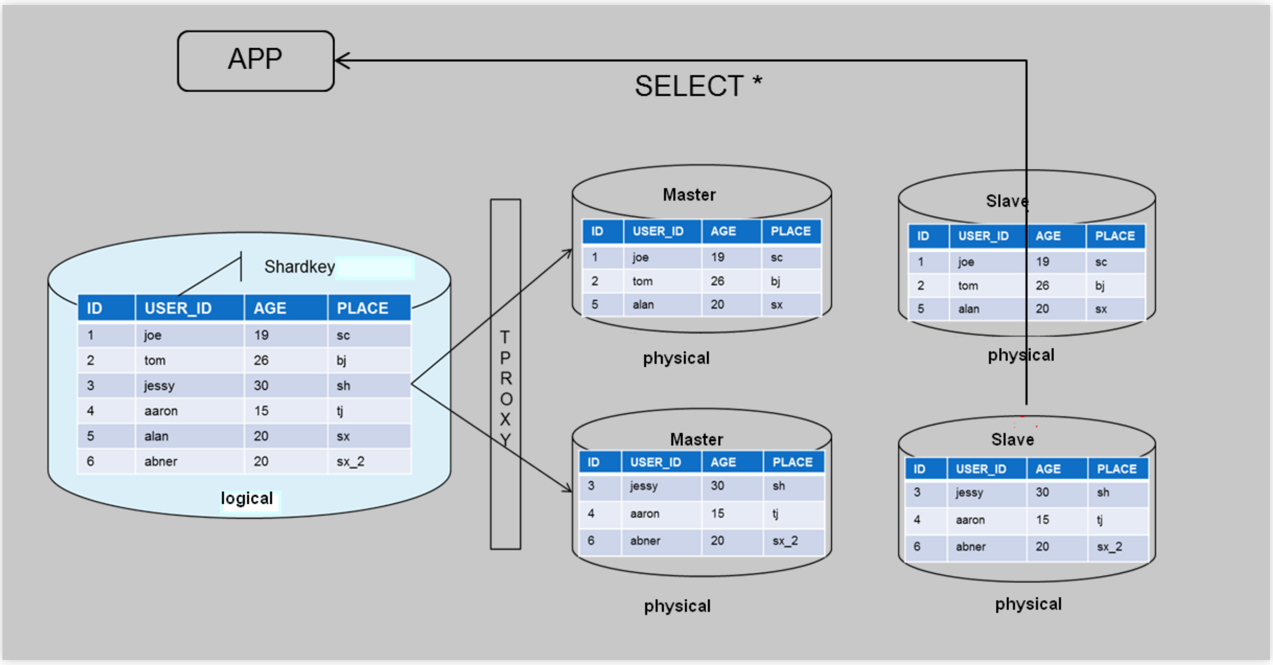
You can use the read/write separation feature to distribute the read pressure to replica nodes if there is a lot of pressure to process a large volume of read requests.
TDSQL for MySQL supports read/write separation by default. Each replica server in the architecture has the read-only permission. If multiple replica servers are configured, the gateway cluster (TProxy) will automatically assign read requests to the replica servers with lower loads to sustain the read traffic of large applications.

How it works

To achieve read/write separation, transactional operations like INSERT, UPDATE, and DELETE are handled by the source node while query operations like SELECT are handled by the replica node.

Read-only account

Note:
If your instance architecture is 1-source-1-replica, the read-only separation feature can only be used for low-load read-only tasks. Avoid high-load tasks such as large transactions, as they affect the backup tasks and availability of the replica server.
A read-only account only has read permission and reads data from the replica server (or read-only instance) in a database cluster by default.
Read requests can be automatically sent to the replica server through the read-only account, with results returned.



ID	USER_ID	AGE	PLACE
1	joe	19	sc
2	tom	26	bj
3	jessy	30	sh
4	aaron	15	tj
5	alan	20	sx
6	abner	20	sx_2

ID	USER_ID	AGE	PLACE
1	joe	19	sc
2	tom	26	bj
5	alan	20	sx

ID	USER_ID	AGE	PLACE
1	joe	19	sc
2	tom	26	bj
5	alan	20	sx

ID	USER_ID	AGE	PLACE
3	jessy	30	sh
4	aaron	15	tj
6	abner	20	sx_2

ID	USER_ID	AGE	PLACE
3	jessy	30	sh
4	aaron	15	tj
6	abner	20	sx_2

Elastic Expansion

Last updated：2024-01-06 17:31:28

Overview

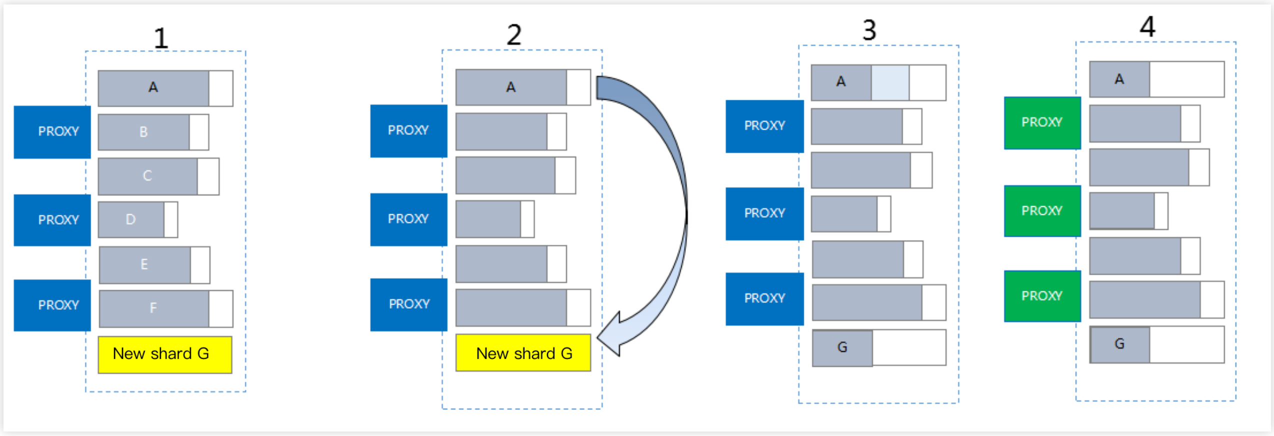
TDSQL for MySQL supports real-time expansion in the console. There are two capacity expansion methods: adding shards and expanding the existing shards. The entire expansion process is completely imperceptible to your business without downtime. Only a fraction of shards become read-only for seconds (or are interrupted) during the expansion, while the entire cluster will not be affected.

Capacity Expansion Process

TDSQL for MySQL achieves automatic expansion and ensures stability with Tencent's automatic rebalancing technology.

Adding a new shard

1. Scale node A in the TDSQL for MySQL console.
2. According to the configuration of new node G, some data of node A will be migrated (from secondary) to node G.
3. After the data is completely synchronized, TDSQL will verify the data in node A and node G (read-only for one to tens of seconds), but the entire service is not suspended.
4. The scheduling system notifies proxy to switch routing.



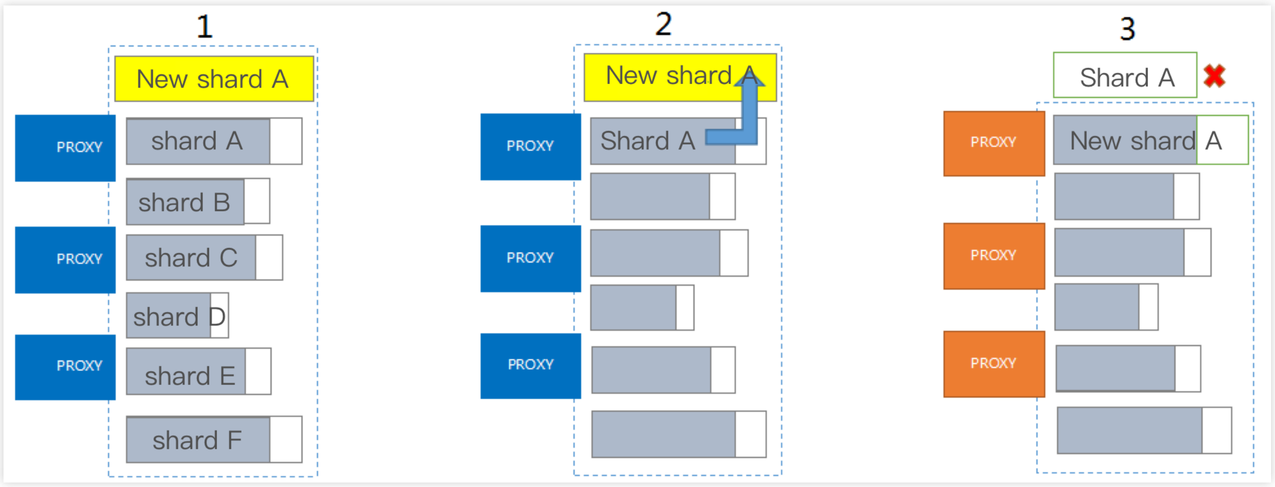
Expanding an existing shard

The expansion based on existing shards is actually to replace the old one with a larger physical shard.

Note:

The expansion based on existing shards does not add any shard and will not change the logic rules of sharding or the number of shards.

1. Assign a new physical shard (hereinafter referred to as "new shard") based on required configuration.
2. Synchronize the data and configuration of the physical shard to be upgraded (hereinafter referred to as "old shard") to the new shard.
3. After the data sync is completed, switch the route in the Tencent Cloud gateway to the new shard, and then you can use the new shard.



References

A distributed database contains multiple shards. To upgrade the specification of an existing TDSQL for MySQL instance, please see Purchase and Upgrade.

Strong Sync

Last updated：2024-01-06 17:31:28

Background

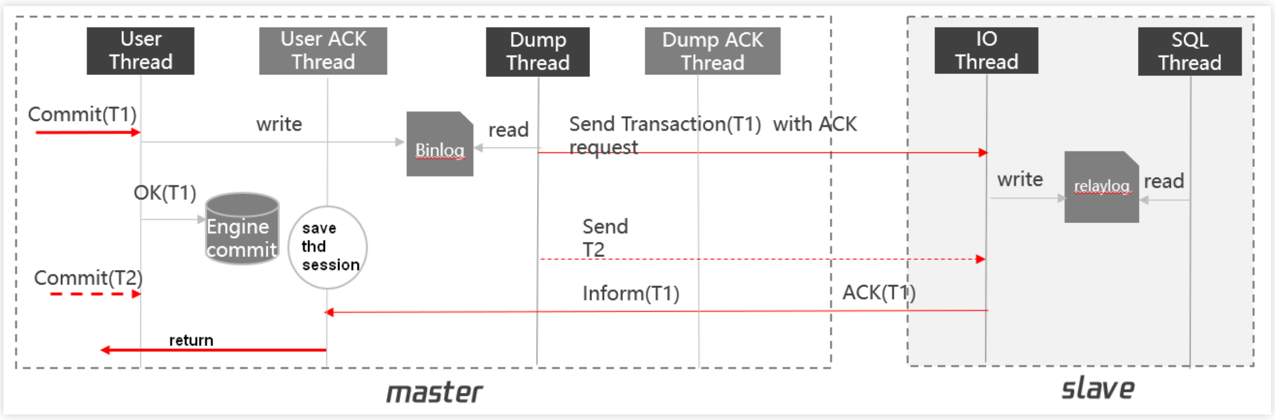
Traditionally, there are three methods of data replication:
Async replication: An application initiates an update request. After completing the corresponding operation, the source node (source) responds to the application immediately and replicates data to the replica node (replica) asynchronously.
Strong sync replication: An application initiates an update request. After completing the operation, the source replicates data to the replica immediately. After receiving the data, the replica returns a success message to the source. Only after receiving the message from the replica will the source respond to the application. The data is replicated synchronously from the source to the replica.
Semi-sync data replication: when an update request is initiated by an application, the primary node replicates data to the secondary node as soon as the update operation is executed on the primary node. After receiving the data and writing into the relay log (which is not necessary to execute), the secondary node returns a message to the primary node. Only after the message is successfully returned can the primary node respond to the application with a request success.

Known Issues

When the source or replica is unavailable, there is a chance of data inconsistency for the above three methods.
As the core of system data storage and service, a database should be highly available. In production systems, high availability solutions are often required to ensure uninterrupted system operations, and the data sync technology serves as the foundation of such solutions.

Solution

In Tencent Cloud's proprietary parallel multi-thread asynchronous replication (MAR, aka strong sync) scheme based on the MySQL protocol, only after a secondary node successfully synchronizes data (logs) can the primary node respond to the application layer, ensuring that the primary and the secondary nodes have completely the same data.
Below is how it works:



In the MAR scheme, when a request is initiated at the application layer, only after a secondary node successfully returns a message can the primary node respond to the application layer with a request success, ensuring that the primary and the secondary nodes have completely the same data.

MAR strong sync scheme outperforms other mainstream sync schemes. For more information, see Performance Comparison Data for Strong Sync. It has the following features:

Consistent synchronous replication ensures strong consistency of data between nodes.

Complete imperceptibility to the business means that read-write separation and sync enhancement are not required at the business side.

Asynchronization of serial sync threads and introduction of thread pool boost a substantial increase in performance.

Cluster architecture is supported.

Automatic member control is supported and faulty nodes are automatically removed from the cluster.

Automatic node join is supported without human intervention required.

Each node contains a complete replica of the data and can be switched at any time.

There is no need to share storage devices.

Instance Architecture

Last updated：2024-01-06 17:31:28

InnoDB engine

Note:

As the database audit feature is being refactored and upgraded, it is not available for newly purchased instances during this period.

Instance Architecture	Definition	Node	Feature
Standard Edition (one source and one replica)	Each shard provides a high-availability architecture based on source/replica active-active deployment.	Two nodes: One source node and one replica node	Supports read-only replicas. In the 1-source-1-replica architecture, the read-only replica feature can only be used for low-load read-only tasks. Avoid high-load tasks such as large transactions, as they affect the backup tasks and availability of the replica server. Automatic node failover. Default sampling granularity for monitoring: Once every 5 minutes. Maximum backup time: 7 days. Operation log backup time: 7 days. Supports database audit where audit logs can be retained for 15 days and an unlimited number of rules which can be configured.
Standard Edition (one source and two replicas)	Each shard provides a high-availability architecture based on source/replica multi-site active-active deployment.	Three nodes: One source node and two replica nodes	
Finance Edition (one source and one replica)	Each shard provides a high-availability architecture based on source/replica active-active deployment.	Two nodes: One source node and one replica node	Supports other deployment schemes. To use other schemes, contact your Tencent Cloud sale rep. Supports read-only replicas and implements intelligent load scheduling for read-only replicas. Automatic node failover. Default sampling granularity for monitoring: Once every 1 minute. Maximum backup time: 3,650 days. You need to submit a ticket for application. Operation log backup time: 60 days by default (1 year for archive storage). Supports database audit where audit logs can be retained for 15 days.
Finance Edition (one source and	Each shard provides a high-availability architecture based on source/replica multi-site	Three nodes: One source	

two replicas)	active-active deployment.	node and two replica nodes	Provides assistance for regulatory compliance.

TDStore engine

In the beta test phase, the TDStore engine adopts a high-availability architecture of three nodes: one source node and two replica nodes.

Regions and AZs

Last updated：2024-01-06 17:31:28

Public Cloud

Tencent Cloud currently offers multiple regions for your choice. The regions and availability zones supported by TDSQL for MySQL can be viewed on the Purchase Page.