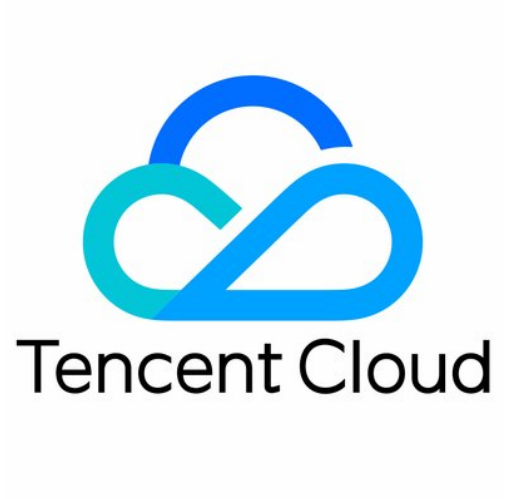
TencentDB for TcaplusDB

FAQs

Product Documentation



Contents

FAQs
Database Features

Last updated : 2021-02-07 10:45:17

Does TcaplusDB support data removal?

TcaplusDB supports table-level data removal, where the data is removed according to the last time it is written. TcaplusDB also supports key-level data removal, and to use this feature, please submit a ticket and select **Other** on the **Select the related product** page.

What are the data structures of TcaplusDB?

TcaplusDB supports data structures such as list array, queries by part of keys (indexes), key-value, key-object (that is, the value of a single key can be arbitrary data structures, for example, game server can serialize `lua table` into the value field).

What is the single-instance memory capacity and CPU utilization of the TcaplusDB SDK?

The maximum single-instance memory consumption is 73 MB, and the maximum CPU utilization is 30%.

How many tables are in a table group in TcaplusDB?

In TcaplusDB, a table group can have up to 256 tables. If there are more than 256 tables in a table group, you can add a new table group or merge the tables. If you need technical support, please submit a ticket and select **Other** on the **Select the related product** page.

What are the restrictions on the key and value fields in TcaplusDB?

The number of key fields of the generic table is 4, the number of key fields of the list table is 3, and the size of a single key field is 1,024 B. The number of value fields of the generic table is 128, the number of value fields of the list table is 127, the size of a single value field is 256 KB, and the maximum size of a record is 1 MB.

How long is the TcaplusDB backup file retained?

The engine files backed up by TcaplusDB are retained for 7 days, and the Ulog is saved for 7 days. The retention periods vary with TcaplusDB environment. For the retention period of a specific TcaplusDB environment, you can contact customer service.

Is the game server connected to all tcaproxy (access layer)?

In order to reduce the cost of maintaining the TCP connections between the game server and tcaproxy (access layer), the game server supports selecting some tcaproxy (access layer) to establish connections.

How do I perform data analysis of TcaplusDB data?

TcaplusDB data can be exported in any format, including json, pb and other formats, and can be imported into data analysis systems such as TDW. TcaplusDB supports the import of real-time data into MySQL databases and so on.

What is the maximum size of a single table in TcaplusDB? What is the limit on the number of records?

A single table can be subdivided into 10,000 data shards, each data shard is 256 GB, that is, the total size of a single table is 10,000 * 256 GB. A single table has no limit on the number of records, and the number of records in a single table is related to the size of a single record.

Does TcaplusDB support multi-table transaction operations and batch write operations?

TcaplusDB supports neither multi-table transaction operations nor batch write operations. To perform multi-table transaction operations, changes have to be made at your business side. In TcaplusDB, operations have to be done in sequence. For example, when multiple operations are submitted at the same time, all completed operations are rolled back if one of the subsequent operation fails. After the rollback, you can submit these operations again. For important operations, we recommend keeping the logs.

Is the TcaplusDB API upgrade backward compatible?

TcaplusDB API upgrade is backward compatible, and existing APIs, command words, and features will not be modified.

Database Use

Last updated : 2021-02-07 10:54:44

How should I get the definition of the error codes in a response packet?

We recommend calling the functions of `TcapErrCode::TcapErrCodeInit` and `TcapErrCode::GetErrStr` in the game server to get error codes, or searching the header files of the TcaplusDB API locally.

How does TcaplusDB's optimistic locking work and how to use it?

Let's use the purchase of train tickets as an example:

1. 100 people want to snap up the same train ticket. The recorded version number of the train ticket is 10, and the 100 people use the same recorded version number to snap up the train ticket.
2. 100 people conduct write operations to this ticket. After the write operation is completed, the recorded version number will increase. Therefore, for the operation of a single key, the tcapsvr worker threads are queued. When the first person successfully purchased the ticket, the recorded version number of the train ticket changed to 11, and the recorded version number of the remaining 99 people is still 10.
3. When tcapsvr handles the write requests of the 99 people, an error will occur because the recorded version number at tcapsvr end is inconsistent with the version number in the request. The concurrent principles are as follows:
   - N requests. If only the first request succeeds and the remaining N-1 requests fail, the version protection rules are used.
   - N requests. If all N requests are required to be executed, no version protection rules are required. TcaplusDB operates on the same key.

Queue and call the `SetCheckDataVersionPolicy` function, whose values include:

- `CHECKDATAVERSION_AUTOINCREASE` : the recorded version number is detected, which will only automatically increase if it is the same as the server-side version number.
- `NOCHECKDATAVERSION_OVERWRITE` : the recorded version number is not detected, and the recorded version number of the client is forcibly written to the server.
- `NOCHECKDATAVERSION_AUTOINCREASE` : the recorded version number is not detected, and the recorded version number of the server will automatically increase.
- You are recommended to use the default type `CHECKDATAVERSION_AUTOINCREASE` .

What are the use cases and precautions for the LIST table?

Where there is a 1:N use case, when N < 1024, the LIST table is prioritized, such as storing the player's most recent 100 emails, the most recent 100 battle records, and so on. The LIST table supports insertion at the head of the queue and removal at the end of the queue, insertion at the end of the queue and removal at the head of the queue, as well as the Top N operations sorted by insertion time. The number of units under a single key can be increased by modifying the table, because the old data needs to be compatible and cannot be modified to become smaller. You can obtain the total number of records under a single key by using `listgetall` . It is recommended to obtain data according to the offset and limit of a certain threshold that you set. For `listreplace` , `listdelete` , and `listdeletebatch` , you need to specify the correct index. For `listaddafter` , you need to specify the elimination rule when the number of element units under a single key reaches the upper limit. If the `SetListShiftFlag` function are called, the LIST table has two increasing and two obtaining directions. There are 4 possibilities:

1. Query A, B, C, D, E as offset = positive number and limit = 2, and the result is A, B; C, D; E.
2. Query A, B, C, D, E as offset = negative number and limit = 2, and the result is D, E; B, C; A.
3. Query E, D, C, B, A as offset = positive number and limit = 2, and the result is E, D; C, B; A.
4. Query E, D, C, B, A as offset = negative number and limit = 2, and the result is B, A; D, C; E.

In addition, calling `GetRecordMatchCount` can get the total number of records.

What is the difference between INSERT, UPDATE and REPLACE?

For INSERT operation, when the key does not exist, an INSERT operation is performed; when the key exists, an error code is returned: `TcapErrCode::SVR_ERR_FAIL_RECORD_EXIST` .
For REPLACE operation, when the key does not exist, an INSERT operation is performed; when the key exists, if the optimistic lock is used, different operations are performed based on the result of the optimistic lock. If the operation is successful, the REPLACE operation is performed, and if it fails, an error code is returned: `TcapErrCode::SVR_ERR_FAIL_INVALID_VERSION;` . If optimistic locking is not used, the REPLACE operation is performed.
For UPDATE operation, when the key exists, if the optimistic lock is used, different operations are performed based on the result of the optimistic lock. If the operation is successful, the UPDATE operation is performed, and if it fails, the error code is returned: `TcapErrCode::SVR_ERR_FAIL_INVALID_VERSION` . If optimistic locking is not used, the UPDATE operation is performed. When the key does not exist, an error code is returned: `TcapErrCode::TXHDB_ERR_RECORD_NOT_EXIST` .

How do I get the number of records in a table?

There is a count command word in the TcaplusDB API. If `tcaplus_client` is used, you can use the `count` table name command to get the number of records in the table.

Does TcaplusDB support traversal operations?

TcaplusDB supports traversal operations, including traversal operations for GENERIC and LIST tables. You can use the `SetOnlyReadFromSlave(bool flag)` API to traverse the data from the secondary tcapsvr, which will not affect the services provided by the primary tcapsvr.

Does TcaplusDB support updating and obtaining partial fields?

TcaplusDB support updating partial fields. When updating and obtaining records, it is recommended to explicitly call API `SetFieldNames(IN const char* field_name[], IN const unsigned field_count)` to determine the fields of this read and write operation, and reduce the network traffic overhead caused by invalid fields.

Is TcaplusDB order-preserving for the continuous operations of a single primary key?

For the same game server, the operations of the same primary key are order-preserving, while the operations of different primary keys are not order-preserving. For different game servers, the order is not preserved.

Does TcaplusDB support table definition changes?

TcaplusDB supports table definition changes. To add non-primary key fields and change macros, you can just change the table. For more complex changes, you can dynamically change the table structure using data migration and logs. To use this feature, please submit a ticket and select **Other** on the **Select the related product** page.

How do I know whether the packeting of response packet has ended?

For traversal, check whether the traversal ends according to state, that is, the `GetState` API. For the rest of packeting scenarios, check whether the packeting ends according to the `HaveMoreResPkgs` function.

What is the difference between GetRecordCount and GetRecordMatchCount?

A request may have N response packets. If there are multiple packets, `GetRecordCount` refers to the number of records in the response packet, and `GetRecordMatchCount` refers to the data records stored at the tcapsvr (storage layer) (total number of records for a single key).

Does TcaplusDB have a pass-through field?

The CS protocol of TcaplusDB is divided into two parts: head and body. `UserBuff` (maximum size is 1 KB), `AsyncID` and `Sequence` in head are all pass-through fields. You can use them accordingly.

What is the role of SetResultFlag?

When performing write operations, the response packet supports returning records. When performing read operations, calling this function is invalid. The specific values of `result_flag` are as follows:
0: only return whether the operation is successful or not without returning the value field.
1: return data consistent with the request field.
2: return the latest data for all fields of the changed record.
3: return the old data for all fields of the changed record.

The `SetResultFlagForSuccess` API can be used to return the data if the operation succeeds; the `SetResultFlagForFail` API can be used to return the data if the operation fails.

Can I perform increase operations on multiple non-primary key fields at a time? What if the primary key does not exist?

To increase multiple non-primary key fields, these fields need to be assigned with values by the request from game server. If one of keys does not exist, you can use the `SetAddableIncreaseFlag` function to perform increase operations. If no key exists, insert a key and then perform increase operations, where the non-increased fields of the key will not be stored and they will take the default value when the record is read. If the key exists, the increase operation can be performed immediately.

How do I reduce traffic costs when TcaplusDB reads records?

When TcaplusDB reads a record, it can be set so it does not return the value field if the record has no change in a fixed period of time, neither does it return the value field if the record version number does not change. For more information, see the `SetFlags` function.

Does TcaplusDB support rollbacks? What is the supported rollback granularity?

TcaplusDB supports rollbacks, including all-server/all-region rollback, single-table rollback, and can roll back N records from 100 billion records. It also supports cold standby time rollback (most recent 01:05:00 am), precise time rollback (to the second), and fuzzy rollback (you can specify the rollback rules). For speed reference, precise time rollback takes about 2 hours for a rollback of 300 GB data and 200 GB Ulog. The principle of the cold standby time rollback is to replace the engine file. The principle of precise time rollback is to roll back the cold standby engine file + Ulog to the needed time point. A key-based rollback requires you to block these keys first and then unblock them after TcaplusDB has finished the rollback.

How efficient are queries by partial keys (indexes) with TcaplusDB?

It is recommended to use queries by partial keys (indexes) in 1:N (N > 1024) app scenarios. The number of primary keys under a single index key is equal to 10 GB/the size of primary key of a single record. A single read-write index operation takes about 100 ms (there are more than 100,000 pieces of data records under a single index key).

What is the timeout mechanism for the TcaplusDB API? What does "it is timeout" error mean?

The TcaplusDB API assigns an ID to each request. After it is successfully sent, the ID is pushed into the timeout judgment structure. If the response packet for the request returns, the ID is removed from the structure. If the response packet of the request has not been processed by the application layer within 3 seconds, an error log with the keyword "it is timeout" is printed. You will then need to check whether the game server is blocked. It may be that tcaproxy (access layer) dropped the packets when returning packets to the game server, or the response packet has arrived at the game server, but the game server did not process it in time.

TcaplusDB recommends that you implement the timeout mechanism yourself, which allows you to perform retry and other processes to timeout requests.

What are the roles of SendRequest, OnUpdate, and ReceiveResponse functions of TcaplusDB API?

The role of `SendRequest` is to send requests. This request may have been sent to the network, or it may be blocked in the sending channel of the game server and a tcaproxy (access layer). The role of `ReceiveResponse` is to get the response packet from the local receive queue. `OnUpdate` is responsible for sending the requests of the send queue to the network and receiving response packets from the network to the receive queue. In message-driven programming mode, the `OnUpdate` is recommended to be called once every millisecond.

How does TcaplusDB achieve global auto increase of fields?

You need to define a single table, and set the field type of a single key and a single value to int64_t (multiple value fields can implement counter arrays). Multiple game servers can increase a single key concurrently (without setting the version protection rules), and get the increase result returned for this operation, then the result of the increase will be a global auto increase.

What rules are defined for queries by partial keys (indexes)?

The index key must be a part of the primary key, the index key must contain the shardkey, and the index key cannot be the primary key.

How do I achieve two-way query of the ID and name of a single table?

The third key field x is used as the shardkey, and the index is built on ID and x, name and x to achieve this feature. For example, when storing player information, the player's district can be added, that is, the primary key is uin, name, district, and the index key is uin and district, and name and district.

Does tcaplus_client support the display of fields at the second nesting level or above?

Yes. You may execute `help select` to see how `select * into a.xml` is used.

What should be noted when changing the table in TcaplusDB?

1. You can only add new fields. You cannot modify the type and name of an existing field, or delete an existing field. To modify these, you need to dynamically modify the table structure.
2. The length of an array or string in a non-primary key field can be increased but not reduced. To modify these, you need to dynamically modify the table structure.

3. The version number should be incremented by 1 for each new field, and the version number defined at the top of the XML file should also be incremented by 1 (that is, the version number of the XML file must be the same as that of the new field). The table in TcaplusDB needs to be changed before the table on game server is changed. TcaplusDB supports dynamic table structure modification, and to use this feature, please submit a ticket and select **Other** on the **Select the related product** page.

What should be noted when defining the table in TcaplusDB?

1. The `refer` field needs to be added to the `count` field.
2. The table shardkey should be highly discrete.
3. The index key cannot be the same as the primary key.

Database Principles

Last updated : 2021-02-07 10:59:23

How does the game server kick out an invalid tcaproxy (access layer) node?

TcaplusDB API supports disaster recovery in case of any tcaproxy exceptions. There are two main ways in which the API kicks out invalid tcaproxy processes:

1. The API physically considers that a tcaproxy is not available. The API sends heartbeat detection packets to all connected tcaproxy every second. If a game server does not receive the corresponding heartbeat return packets from tcaproxy within 10 seconds, the API will actively disconnect the TCP connection to the tcaproxy and actively connect the tcaproxy at the next onupdate.
2. The API logically considers that a tcaproxy is not available. The API calculates the request and response ratio of a tcaproxy every 10 seconds as a basis for judgment. The timeout threshold for a request packet is 3 seconds. If a tcaproxy times out more than 3 times, the tcaproxy is considered unavailable and no request will be sent to it. A `getmetdata` request is sent to the tcaproxy 60 seconds later. If the tcaproxy can correctly handle the `getmetadata` request, the API considers the tcaproxy available and requests will be sent to it again.

If the game server finds that a tcaproxy is not available within 10s, it will not send data to the tcaproxy node.

How does the game server choose the tcaproxy (access layer) node?

The game server maintains a consistent Hash ring locally. Once a tcaproxy (access layer) node has been verified, it will be added to the Hash ring. If a tcaproxy (access layer) node reduces capacity or the TCP link between game server and tcaproxy (access layer) has been disconnected due to machine abnormalities, the game server will remove the tcaproxy (access layer) node from the Hash

ring. The game server calculates hash values based on the primary key in the request (if it is a `batchget` request, it randomly selects a single tcaproxy (access layer) node), and then selects a single tcaproxy (access layer) node from the consistent Hash ring.

Does TcaplusDB have the compression feature?

TcaplusDB has a compression feature based on the Google snappy compression algorithm, including protocol compression (i.e., compression of the request/response packet between game server and tcaproxy (access layer) and data compression (i.e., compression of the data that needs to be stored by tcapsvr (storage layer)). If you want to reduce the network traffic costs between game server and tcaproxy (access layer), we recommend that you enable the protocol compression. You can also call the `SetCompressSwitch` function of TcaplusDB API to enable the tcapsvr (storage layer) compression, which can save disk space, improve IO disk performance, and reduce the CPU resources used for compression and decompression.

Is the TcaplusDB API thread-safe?

TcaplusDB API is non-thread-safe, mainly because components such as tlog and tdr are not thread-safe. It is recommended that a single thread uses a single API object and a single game region uses a single API object. If you need to interact across game regions, it is recommended to maintain multiple API objects with a single game server.

How does tcaproxy (access layer) achieve disaster recovery?

Tcaproxy (access layer) adopts a peer-to-peer design scheme, that is, all tcaproxy (access layer) nodes under a single game region contain routing information of all tables under a single game region. If a tcaproxy (access layer) fails, as long as the remaining tcaproxy (access layer) nodes are not overloaded, the game server will kick out the abnormal tcaproxy (access layer) node, which will not affect the use of game server. There is no single point of failure risk for the tcaproxy (access layer).

How does tcapsvr (storage layer) achieve disaster recovery?

The tcapsvr (storage layer) runs in a primary-secondary mode (primary tcapsvr and secondary tcapsvr). Primary and secondary tcapsvr synchronize data in real time, and are deployed at the different IDCs in the same city, ensuring that primary-secondary synchronization latency is less than 10 ms. If the secondary tcapsvr is abnormal, it will not affect the use of game server (if the read request offloading is not enabled, the requests of game server are processed by the primary tcapsvr. If the read request offloading is enabled, the secondary tcapsvr will assist in processing part of the read requests), and the DBA will rebuild the secondary tcapsvr; if the primary tcapsvr is abnormal, the secondary tcapsvr will perform failure recovery and the DBA will apply for a new machine to rebuild the secondary tcapsvr. There is no single point of failure risk for the tcapsvr (storage layer).

Does TcaplusDB have overload protection?

Both the access layer and the storage layer have process-level overload protection measures to ensure that services will not collapse during peak hours.

What is the principle of cold and hot data exchange for TcaplusDB?

TcaplusDB uses memory + SSD disk storage. The first GB data of a single engine file is mapped in memory. The hot data is placed in the memory and the cold data is placed on the disk. The LRU algorithm is used for hot and cold data exchange. The `get` operation of game server triggers the LRU swap-in operation, and the LRU thread of tcapsvr (storage layer) is responsible for the LRU swap-out operation. Try to ensure that the hot data is stored in the memory, thus ensuring high cache hit rate and low single read and write latency.

How does tcaproxy (access layer) choose tcapsvr (storage layer)?

Each table defines a shardkey. If no shardkey is defined, the shardkey defaults to the primary key. Tcaproxy (access layer) selects the corresponding tcapsvr (storage layer) according to `hash (shardkey) % 10000` (where % is the remainder operator), so the shardkey should be highly discrete.

What is the level of the lock in TcaplusDB?

The granularity of the lock in TcaplusDB is logging level.