TencentDB for MySQL

Tencent Kernel TXSQL

Product Documentation

Contents

Tencent Kernel TXSQL Overview

Last updated：2024-07-22 11:03:04

TXSQL is a MySQL kernel branch maintained by the TencentDB team and is fully compatible with native MySQL. It provides various features similar to those in the MySQL Enterprise Edition, such as enterprise-grade transparent data encryption (TDE), auditing, dynamic thread pool, encryption function, backup and restoration, and parallel query. TXSQL not only deeply optimizes the InnoDB storage engine, query performance, and replication performance, but also improves the ease of use and maintainability of TencentDB for MySQL. While providing all the benefits of MySQL, it offers more enterprise-grade advanced features such as disaster recovery, restoration, monitoring, performance optimization, read/write separation, TDE, and auditing.

The following provides more information about TXSQL:

For details on the TencentDB for MySQL kernel version updates, see Kernel Version Release Notes.

The kernel minor versions of TencentDB for MySQL can be upgraded automatically or manually. For more information, see Upgrading Kernel Minor Version.

You can use a CVM instance to log in to a TencentDB for MySQL instance and check its kernel minor version. For more information, see Kernel Upgrade.

Kernel Version Release Notes
TXSQL Kernel Release Notes

Last updated：2024-06-20 10:26:07

This document describes the version updates of the TXSQL kernel.

Note:

For more information on how to upgrade the minor kernel version of a TencentDB for MySQL instance, see Upgrading Kernel Minor Version.

When you upgrade the minor version, some minor versions may be under maintenance and cannot be selected. The minor versions available in the console shall prevail.

MySQL 8.0 Kernel Version Release Notes

MySQL 5.7 Kernel Version Release Notes

MySQL 5.6 Kernel Version Release Notes

Minor Version	Description
20230630	**Note：** Starting from MySQL 8.0.29, the query results of tables in the Information Schema will use utf8mb3 instead of utf8. Versions of Connector/Net earlier than 8.0.28 do not support utf8mb3; encountering utf8mb3 will result in an error: Character Set 'utf8mb3' is not supported by .Net Framework. If the application uses Connector/Net, please upgrade Connector/Net to 8.0.28 or later before upgrading the TencentDB for MySQL version. For more details, see: MySQL 8.0.29 Character Set Support Changes in MySQL Connector/NET 8.0.28 New features Supported Nonblocking DDL feature. Supported xa commit to record the maximum gts instance TP/AP load statistics in relay log. Supported selecting Innodb temporary tables for parallel query of worker thread sharing. Supported using partition tables as parallel tables for parallel queries. Supported the flashback version query feature. Supports persistence for flashback query. Supported virtual indexes. Supported the range/list secondary partition feature. Supported the automatic relay log recovery feature. Supported the default algorithm for DDL, with options INPLACE/INSTANT. Supported Fast Query Cache. Supported the conversion of partition tables from MyISAM to InnoDB. Supported the correlated subquery cache feature. Performance Optimization Optimized the BINLOG LOCK_done lock conflict. Optimized thread pool performance.

Optimized the issue where disabling eq_ref cache in outer join leads to performance regression.
Enhanced parallel query:
Subquery \\derived table executed in parallel independently: Optimized and executed plan for subquery \\derived table in parallel, independent of main query execution.
Nested loop join inner table in parallel: When the NLJ outer table is small, the inner table can be selected as the parallel table for parallel execution with ROLL UP.
Hash join (in memory) executed in parallel: Work threads build complete hash tables separately, with parallel scanning on the probe end.
Parallel query supports global aggregation optimization.
Parallel query supports pushdown parallelism under having condition.
Optimized binlog submission for large transactions.
Optimized performance fluctuation caused by binlog purge.
Supported for hot updates, merge and optimization.
Bug Fixes
Fixed the issue of assertion failure when rolling back transactions of Parallel Copy DDL.
Fixed the issue where EXPLAIN FORMAT=TREE does not print subqueries in the condition of HashJoin.
Fixed the issue where redundant format causes instance running exception after instant add.
Fixed the issue of global transaction id rollback after upgrading from an older version.
Fixed the issue where index merge intersect causes incorrect query results.
Fixed the issue where cross-machine statistical information collection may block the shutdown process.
Fixed the issue where historical histogram versions might crash in a primary-secondary environment.
Fixed the issue of check index holding a large number of page locks.
Fixed the deadlock issue in cross-machine histograms under concurrent DDL operations.
Fixed the issue where the lock was not released when the cross-machine histogram task included too many columns.
Fixed several instant DDL issues.
Fixed the issue where update returning caused the client to disconnect.
Fixed the null pointer dereference vulnerability found by the vulnerability scan.
Fixed the issue where changes in the storage layer table structure under parallel execution may cause instance running exception.
Fixed the performance degradation issue caused by using WHERE column IN (list) in prepare statements.
Fixed the issue where statistical information might be empty when importing mysqldump logical backups.
Fixed the primary key conflict issue that occurs when using Parallel Copy DDL for table changes that include auto-increment columns.
Fixed the issue where the build branch of the hash join in parallel queries cannot be parallelized when the hash join is present.
Fixed the issue where the non-parallel branches of a parallel query join cannot be parallelized when there is a UNION.
Fixed two memory leak issues in parallel queries.
Fixed the partition exit issue for empty range in parallel queries.

	Fixed the error issue with Outline IN-list. Fixed the issue where partition_id overflow leads to truncate partition crash. Fixed the issue in parallel queries where related subqueries referencing worker table fields resulted in incorrect query results. Fixed the issue in parallel DDLs regarding obtaining an incorrect offset. Fixed the issue in parallel DDLs where adding a unique key to a column with duplicate data caused the instance to run abnormally. Fixed the issue of assertion in parallel hash join debug. Fixed the issue in parallel cost calculation where NDV was 0. Fixed the issue with JSON import accuracy. FORCE INDEX ORDER BY statement skips the index dive bug. Fixed the issue where the official subquery plan was displayed multiple times. Fixed the issue where the disk-based temporary table quantity does not increase. Fixed the deadlock issue caused by proxy change user. Fixed the issue where calling a stored procedure in a trigger due to permission verification optimization caused permission checks to be bypassed.
20221221	**Note：** Starting from MySQL 8.0.29, the query results of tables in the Information Schema will use utf8mb3 instead of utf8. Versions of Connector/Net earlier than 8.0.28 do not support utf8mb3; encountering utf8mb3 will result in an error: Character Set 'utf8mb3' is not supported by .Net Framework. If the application uses Connector/Net, please upgrade Connector/Net to 8.0.28 or later before upgrading the TencentDB for MySQL version. For more details, see: MySQL 8.0.29 Character Set Support Changes in MySQL Connector/NET 8.0.28 Bug Fixes Fixed the issue where after enabling log_slave_updates on a secondary node, the thread_id of the event written to the binlog on the secondary node changed.
20221220	**Note：** Starting from MySQL 8.0.29, the query results of tables in the Information Schema will use utf8mb3 instead of utf8. Versions of Connector/Net earlier than 8.0.28 do not support utf8mb3; encountering utf8mb3 will result in an error: Character Set 'utf8mb3' is not supported by .Net Framework. If the application uses Connector/Net, please upgrade Connector/Net to 8.0.28 or later before upgrading the TencentDB for MySQL version. For more details, see: MySQL 8.0.29 Character Set Support Changes in MySQL Connector/NET 8.0.28 Bug Fixes Fixed the instant DDL bug.
20221215	**Note：** Starting from MySQL 8.0.29, the query results of tables in the Information Schema will use utf8mb3 instead of utf8. Versions of Connector/Net earlier than 8.0.28 do not support utf8mb3; encountering utf8mb3 will result in an error: Character Set 'utf8mb3' is not supported by .Net Framework. If the application uses Connector/Net, please upgrade Connector/Net to 8.0.28 or later before upgrading the TencentDB for MySQL version. For more details, see:

	MySQL 8.0.29 Character Set Support Changes in MySQL Connector/NET 8.0.28 New Features Merges official changes from 8.0.23 to 8.0.30. Supported user connection status monitoring feature, which can be viewed through show detail processlist for connection monitoring. Supported the update wait N syntax. Supported nvl(), to_number(), to_char() function feature syntax. Supported cdb_kill_user_extra regular expression. Performance Optimization Optimized binlog rotate implementation method and improved binlog write speed. Optimized TencentDB for MySQL startup speed. Optimized binlog checksum calls, and reduced unnecessary CPU performance overhead. Optimized ha_innopart::external_lock lock hotspots, and reduced lock holding time. Optimized xa::Transaction_cache, and reduced lock conflicts. Reduced ha_innopart::clear_blob_heaps time consumption. Optimized purge threads lock hotspots, and reduced tasks_mutex and thread conflicts. Optimized Buffer Pool initialization, supporting parallel initialization, and accelerating initialization speed. Optimized read_only and select performance under high concurrency. Optimized permission validation for prepared statement and stored procedure. Optimized access to change buffer. Avoided unnecessary calls to fil_space_get, and reduced FAQs in extreme scenarios. Optimized lock conflict for GTID during transaction commit when binlog_order_commits is disabled. Applied Lock Free Hash to optimize trx_sys mutex conflict. Optimized the overhead of taking a snapshot in the transaction system. Optimized Writeset and improved performance. Replaced index drill-down with histogram. Supports Parallel DDL. Bug Fixes Fixed issues with abnormal statistical values such as innodb_row_lock_current_waits. Fixed the issue of excessively high memory usage with Group concat with group by. Fixed the issue of statistical information being severely underestimated in long records. Fixed the issue in parsing stored procedure syntax. Fixed the issue in FAST DDL optimization of flush list to release page concurrency.
20220831	New features Supported setting the MySQL version dynamically. Supported transparent column encryption. When creating a table, you can specify the encryption attribute for the `varchar` field, and the storage system will encrypt the column. This capability is expected to be commercialized in 2023. Fixed the exception of the third-party data subscription tool caused by subscription to the comparison SQL for internal data consistency during tool usage. **Note：**

After the database instance is migrated, upgraded, or recovered after failure, the system will compare the data consistency to ensure the consistency of data. When comparison SQL is in `statement` mode, exceptions are easy to occur in response of some third-party subscription tools to the SQL in `statement` mode. When the instance is upgraded to its kernel, the third-party data subscription tool can't subscribe the comparison SQL for internal data consistency.
Supported adding NO_WAIT | WAIT [n] for DDL operations. This enables such operations to be rolled back immediately if they cannot obtain the MDL lock and must wait or if they have waited the specified time for the MDL lock.
Supported the fast query cache feature, which is suitable for scenarios with more reads than writes. If there are more writes than reads, the data is updated very frequently, or the result set of the query is very large, we recommend that you not enable this feature.
Supported enhanced MTS deadlock detection.
Supported parallel query. After this feature is enabled, large queries can be automatically identified. The parallel query capability leverages multiple compute cores to greatly shorten the response time of large queries.
Performance optimizations
Optimized the overheads of the transaction system to take snapshots. The Copy Free Snapshot method is adopted, the transaction delay is deleted from the global active transaction hash, and the snapshot taking method is optimized to determine the logical timestamp of the snapshot event. As tested by sysbench, the extreme performance is increased by 11% in the read-write scenario.
Optimized permission check for prepared statements. A variable is used globally to indicate the permission version number, a prepared statement records the version number after being prepared, and the system checks whether the version number has changed during execution. If there is no permission change, the system will skip the permission check; otherwise, it will check the permission and record the version number again.
Optimized the accuracy of time acquisition in the thread pool.
Optimized record offset acquisition. A record offset is cached for each index. When the conditions are met, the cached offset will be directly used, saving the computing overheads of invoking the `rec_get_offsets()` function.
Optimized parallel DDL.
1. When the index field is small, the sampled memory size is reduced to lower the sampling frequency.
2. The K-way merge algorithm is used for sorting, which effectively reduces the number of rounds of merging and sorting to lower the number of IOs.
3.When records are read, the fixed-length offset is cached in order to avoid generating offsets for each record each time.
Optimized the undo log information recording logic to improve the INSERT performance.
Improved the performance after semi-sync was enabled.
Optimized the audit performance to reduce the system overheads.
Bug fixes
Fixed the issue where the displayed value of `Thread_memory` was abnormal sometimes.
Fixed the issue where the timestamp was inaccurate during batch statement audit.
Fixed issues related to column modification at the second level.

	Fixed the issue where the `CREATE TABLE t1 AS SELECT ST_POINTFROMGEOHASH("0123", 4326);` statement caused source-replica disconnection. Fixed the issue where the replica failed to retry during concurrent requests at the table level. Fixed the `Malformed packet` error reported when `show slave hosts` was executed. Fixed recycle bin issues. Fixed the issue where the jemalloc mechanism easily triggered OOM on ARM device models. Fixed the issue where `truncate pfs account table` caused the failure to collect statistics. Fixed the exception that occurred while restoring the child table first and then restoring the parent table when the recycle bin had a foreign key constraint. Fixed sql_mode log issues. Fixed the occasional issue where a procedure became abnormal when `CREATE DEFINER` was executed. Fixed Copy Free Snapshot issues. Fixed the performance fluctuation of the thread pool. Fixed the issue where the result of `hash join+union` might be empty. Fixed memory issues.
20220401	Bug fixes Fixed the issue where the stage variable error in Parallel DDL caused the stage null pointer to crash when creating FTS indexes. Fixed the possible crash when adding full-text indexes.
20220331	Bug fixes Fixed the crash caused by dereferencing wild pointers in the thread pool.
20220330	New features Enabled writeset parallel replication by default. Supported extended resource groups to control the I/O, memory utilization, and SQL timeout policy by user. Supported flashback query to query data at any time point within the UNDO time range. Supported `RETURNING` in a `DELETE`, `INSERT`, or `REPLACE` statement to retrieve the data rows modified by the statement. Supported the GTID replication feature extension in row mode. Supported transaction lock optimization. Enhanced the recycle bin to support TRUNCATE TABLE and automatic cleanup of tables in the recycle bin. Supported parallel DDL to speed up DDL operations for which to create indexes through three-phase parallel operations. Supported quick index column modification. Supported automatic statistics collection and cross-server statistics collection. Performance optimizations Optimized the GTID lock conflicts when transactions were committed if `binlog_order_commits` was disabled. Accelerated MySQL startup by changing the InnoDB startup phase from single-threaded creation of Rsegs to multi-threaded creation. Bug fixes

	Fixed the issue where the transaction did not end when the connection was closed after deadlock or lock wait. Fixed the issue where the `innodb_row_lock_current_waits` value was abnormal. Fixed the SQL type error in the audit plugin without USE DATABASE. Fixed the issue where tables smaller than `innodb_async_table_size` were also renamed during async drop of big tables. Fixed the issue with incorrect escape characters in the audit plugin. Fixed the issue of rollback after quick column modification. Fixed the issue where the transaction system (trx_sys) may crash if it contains XA transactions when it is closed. Fixed the crash when merging derived tables. Fixed the issue where `binlog_format` was modified after writeset was enabled. Fixed the error (error code: 1032) caused by hash scans with A->B->A->C update on the same row. Fixed the issue where the sort index might be invalid in prepared statement mode. Fixed the issue where the operator that consumed the materialized result might be merged into the returned value path of the materialized operator and result in incorrect comprehension and display of the execution plan. Fixed exceptions in extreme cases for async drop of big tables. Fixed the abnormal error message when setting a SQL filter. Fixed the syntax error reported during stored procedure parsing. Fixed the issue where historical histograms couldn't be applied. Fixed the role column display compatibility issue caused by `SHOW SLAVE HOSTS(show replicas)`. Fixed the crash of `Item_in_subselect::single_value_transformer` when the number of columns was incorrect. Fixed the crash caused by memory leaks during cascading update if a subtable contained virtual columns and foreign key columns.
20211202	New features Supported quick column modification. Supported histogram versioning. Supported SQL:2003 TABLESAMPLE (single table) sampling control syntax for obtaining random samples of physical tables. Added non-reserved keywords: TABLESAMPLE BERNOULLI. Added the `HISTOGRAM()` function to build a histogram for a given input field. Supported compressed histograms. Supported SQL throttling. Supported MySQL cluster role configuration (default role: CDB_ROLE_UNKNOWN). Added a new `Role` column to the `show replicas` command's display results to display roles. Supported proxy. Performance optimizations Optimized the hotspot update problem caused by `insert on duplicate key update`. Accelerated the application of hash scan by aggregating multiple identical binlog events. Greatly reduced the memory usage by the `PREPARE` statement in point queries in the thread pool mode when the plan cache was enabled.

	Bug fixes Fixed the error of unstable performance after hotspot update optimization was enabled. Fixed the issue where `select count(*)` parallel scans caused full-table scans in extreme cases. Fixed performance issues caused by execution plan changes due to reading zero statistics in various cases. Fixed the bug where queries were in the `query end` status for a long time. Fixed the bug where statistics were severely underestimated in long records. Fixed the bug where an error was reported when the Temptable engine was used and the number of aggregate functions in the selected column exceeded 255. Fixed the case sensitivity issue of column names in the `json_table` function. Fixed the bug that caused correctness issues in window functions because expressions returned early during `return true`. Fixed the correctness issue caused by the pushdown by `derived condition pushdown` when it contained user variables. Fixed the issue where SQL filters were prone to crash when no namespaces were added in a rule. Fixed the QPS jitters when the thread pool was enabled under high concurrency and high conflict. Fixed the issue where source-replica buffer pool sync leaked file handles in extreme cases (when host file systems were corrupted). Fixed the index mapping issue. Fixed the statistics cache sync issue. Fixed the crash when information was not cleared during execution of the `UPDATE` statement or stored procedures.
20210830	New features Supported limiting the number of preloaded rows. Supported optimizing plan cache point query. Supported extended ANALYZE syntax (UPDATE HISTOGRAM c USING DATA 'json') and direct writes to histograms. Performance optimizations Replaced index seek with histogram to reduce evaluation errors and I/O overheads (this capability is not enabled by default). Bug fixes Fixed the issue where there might be no statistics information during online DDL. Fixed the issue where generated columns on replica instances were not updated. Fixed the issue where the instance hung when binlog was compressed. Fixed the issue of missing GTID in the previous_gtids event of the newly generated binlog file. Fixed possible deadlocks when system variables were modified. Fixed the issue where the information of the SQL thread of the replica instance in SHOW PROCESSLIST was incorrectly displayed. Implemented the bug fix related to hash join provided in MySQL 8.0.23. Implemented the bug fix related to writeset provided in MySQL. Implemented the bug fix related to the query optimizer provided in MySQL 8.0.24. Fixed the concurrency bugs of optimizing flush list and releasing pages in FAST DDL.

	Optimized the memory usage during data dictionary update in instances with a large number of tables. Fixed the crash caused by new primary key creation after INSTANT ADD COLUMN. Fixed the OOM caused by memory growth in full-text index query. Fixed the issue where -1 was included in the TIME field in the result set returned by SHOW PROCESSLIST. Fixed the issue where tables might fail to be opened due to histogram compatibility. Fixed the floating point accumulation error when Singleton histograms were constructed. Fixed the replication interruption caused by using many Chinese characters in the table name of a row format log.
20210330	New features Supported source-replica buffer pool sync: After a high-availability (HA) source-replica switch occurs, it usually takes a long time to warm up the replica, that is, to load hotspot data into its buffer pool. To accelerate the replica's warmup, TXSQL now supports the buffer pool sync between the source and the replica. Supported sort-merge join. Supported FAST DDL operations. Supported querying the value of the `character_set_client_handshake` parameter. Performance optimizations Optimized the mechanism of scanning and flushing the dirty pages tracked in the flush list, so as to solve the performance fluctuation issue while creating indexes and thus improve the system stability. Bug fixes Fixed the deadlocks caused by the modification of the `offline_mode` and `cdb_working_mode` parameters. Fixed the persistent concurrency issue of the `max_trx_id` field in `trx_sys` table.
20201230	New features Supported the official updates of MySQL 8.0.19, 8.0.20, 8.0.21, and 8.0.22. Supported dynamic setting of thread pooling mode or connection pooling mode by using the `thread_handling` parameter. Performance optimizations Optimized the `BINLOG LOCK_done` conflict to improve write performance. Optimized the `trx_sys mutex` conflict by using lock-free hash to improve performance. Optimized redo log flushing. Optimized the buffer pool initialization time. Optimized the clearing of adaptive hash indexes (AHI) during the `drop table` operations on big tables. Optimized audit performance. Bug fixes Fixed performance fluctuation when cleaning InnoDB temporary tables. Fixed the read-only performance decrease when the instance has many cores. Fixed the error (error code: 1032) caused by hash scans. Fixed concurrency security issues caused by hotspot update.

20200630	New features Supported async drop of big tables. You can clear files asynchronously and slowly to avoid business performance fluctuation caused by dropping big tables. To apply for this feature, submit a ticket. Supported automatic killing of idle tasks to reduce resource conflicts. To apply for this feature, submit a ticket. Supported transparent data encryption (TDE). Bug fixes Fixed the issue where switch failed due to inconsistent positions between `relay_log_pos` and `master_log_pos`. Fixed the data file error caused by asynchronously storing data in the disk. Fixed the hard error when `fsync` returned `EIO` and retries were made repeatedly. Fixed the crash caused by phrase search under multi-byte character sets in full-text index.

Minor Version	Description
20230601	New Features Supports persistence for flashback query. Supported drop table force, enabling drop innodb metadata. Supported Parallel Copy DDL. Supported limit in subquery. Supported the conversion of partition tables from MyISAM to InnoDB. Bug Fixes Fixed the issue of index anomaly in primary-secondary BP synchronization feature. Fixed the issue where killing connections during large transactions caused anomalies. Fixed the issue of obtaining user-defined variable string errors in session track. Fixed the issue of failure to create index when parallel DDL is enabled and innodb_disable_sort_file_cache is set. Fixed some errors with instant modify column.
20230115	New Features Supported Nonblocking DDL feature. Supported validate password plugin. Supported for storing historical deadlock information. Performance Optimization Asynchronous deletion of large tables: Temporary tables also use the innodb_async_table_size filter table, and only tables exceeding innodb_async_table_size are deleted asynchronously, improving the processing efficiency. Bug Fixes Fixed the issue where creating a user with grant identified by failed, causing primary/standby interruption. Fixed the issue where GROUP_CONCAT did not correctly set USED_TABLES when the DERIVED_MERGE switch was enabled. Fixed the issue where the gtid_subset function failed to correctly handle null_value. Fixed the issue where dummy index cache failed to initialize system columns.

	Fixed the issue of instant add column in partition table exceeding the maximum number of columns. Fixed the issue where canal pulling binlog may cause OOM. Fixed the error in proxy when reusing connections with different users. Fixed the proxy's incorrect responses for row count, found rows, and db settings. Fixed the issue where the error messages during binlog sending and receiving were incomplete. Fixed anomalies in paging and pushdown calculations. Fixed potential invalidity of m_page after creating a subtree with Parallel DDL. Fixed the crash issue with instant modify under certain character sets.
20220716	New features Supported auto-increment column persistence for InnoDB. Supported precise memory statistics. Supported query-level memory monitoring. Supported recycle bin. Supported parallel DDL statements. Supported flashback query. Supported async rollback for internal XA transactions. Performance optimizations Optimized async drop of big tables.The original definition of big table is 50 GB, which can now be controlled by the `innodb_async_table_size` to make it more flexible. Bug fixes Fixed the issue where `ERROR 1878 (HY000): Temporary file write failure` was reported when `alter table` was executed to create indexes. Fixed the issue where buf/buf/pool couldn't be viewed in PFS memory monitoring data. Fixed the issue where the returning statement might cause exceptions in some scenarios due to permission checks. Fixed the issue where an error was reported because the parser did not correctly handle semicolons in statements. Fixed the issue where single quotation marks in audit statements were not escaped. Fixed the issue of sudden memory usage increase on the ARM platform. Fixed the issue of source-replica inconsistency caused by modifying `binlog_format` after writeset was enabled. Fixed the issue of high CPU usage caused by exiting a large number of threads at the same time. Fixed bugs related to `drop table partition force`. Fixed the issue where binlog dump got stuck and caused the instance restart to become slow. Fixed the issue where the source-replica sync failed because `create table like temporary table` did not inherit the character set in the binlog. Fixed the issue where `show detail processlist` displayed illegal characters. Fixed the issue where the `thread_group` lock was not released when the thread pool was closed in some cases. Fixed the issue where updating the parent table at the parallel table level caused the instance to run abnormally. Fixed the issue where virtual columns were calculated incorrectly on the replica.

	Fixed the issue where `gtid_subset` did not set `null_value` to `false` after executing a row.
20211230	New features Supported the official updates of MySQL 5.7.19–5.7.36. Supported source-replica buffer pool sync to speed up the performance recovery after HA switch (around 90 seconds faster than that in native mode). Added the backup lock feature to provide lightweight metadata locks to improve the service availability during backup. Performance optimizations Made functions related to `utf8/utf8mb4 my_charpos` inline to optimize the performance of UTF_8 functions in read_write scenarios. Upgraded jemalloc to v5.2.1. Optimized file number acquisition during binlog rotation. Optimized semi-sync replica I/O. Optimized hash scan aggregation. Accelerated the startup of crash recovery for large transactions.
20211102	New features Fixed the exception of the third-party data subscription tool caused by subscription to the comparison SQL for internal data consistency during tool usage. **Note：** After the database instance is migrated, upgraded, or recovered after failure, the system will compare the data to ensure data consistency. When comparison SQL is in `statement` mode, exceptions are prone to occur in response of some third-party subscription tools to the SQL in `statement` mode. When the instance is upgraded to its kernel, the third-party data subscription tool can't subscribe the comparison SQL for internal data consistency.
20211031	New features Supported writeset replication. Performance optimizations Optimized the checkpoint mechanism to increase the backup success rate. Optimized the hash scan index selection. Optimized the hotspot update performance to support `insert on duplicate key update`. Bug fixes Fixed the error of unstable performance after hotspot update was enabled. Fixed the crash caused by rolling back the UPDATE operation after an instant DDL. Fixed the issue where the `CREATE TABLE AS SELECT` statement didn't inherit the compression attribute after column compression was enabled. Fixed the instance crash caused by the `show variables like 'tencent_root%'` statement after the `skip-grant-table` option was enabled. Fixed the crash of the Query Rewriter plugin in read-only mode. Fixed the error (error code: 1032) caused by hash scans in partitioned tables. Fixed the issue where the first large transaction's SBM was 0 in MTS mode. Fixed the crash of `stop slave` caused by `slave_preserve_commit_order=ON, slave_transaction_retries=0`. Fixed several XA transaction bugs.

	Fixed the issue where SQL splicing went wrong during `show create` after a JSON field with a default value was created. Fixed the issue where disconnected transactions could not be rolled back after transactions were blocked. Fixed the issue where there might be no statistics information for long records in InnoDB persistent mode. Ported 8.0 to fix the issue where `ANALYZE TABLE` might cause query retention. Fixed the issue where the InnoDB statistics couldn't be synced to the server layer in time after change. Fixed the issue where statistical sampling might block writes for too long and cause a crash (bug# 31889883). Fixed the possibility of reading zero rows during the InnoDB statistics update process (bug# 105224). Fixed the possible O(N^2) behavior in MVCC (bug# 28825617). Fixed the crash caused by closing a temp table and triggering binlog rotation when a connection was released.
20210630	New features Added the new command SHOW SLAVE DETAIL [FOR CHANNEL channel] for displaying the binlog timestamp that the current replica has replayed. Supported transaction_read_only/transaction_isolation parameters. Performance optimizations Accelerated the application of hash scan on replicas by aggregating multiple identical binlog events. Bug fixes Fixed the issue where duplicate primary keys existed, columns couldn't be found, and columns were too long in temp tables caused by the `UPDATE` statement. Fixed the issue where there might be no statistics information during the DDL process. Fixed the inaccurate undo log size in connection status statistics. Fixed the instance crash caused by querying the metadata_locks table. Modified `of` as a non-reserved keyword. Fixed the issue where the dynamically modified version number was not invalidly displayed in new connections. Fixed the issue where the wild pointer was accessed during page_cache cleanning. Fixed the issue where the execution of ALTER TABLE might report the "Incorrect key file for table" error. Fixed the excessive memory usage by partitioned tables. Fixed the issue where -1 was included in the TIME field in the result set returned by SHOW PROCESSLIST. Fixed the lock wait of XA transaction replication on replica nodes. Fixed the incorrect lock of partitioned tables in equal range query.
20210331	New features Supported `RETURNING` clause in a `DELETE`, `INSERT`, or `REPLACE` statement to return information about the rows that were deleted or modified.by the statement. For `DELETE`, undo data is returned, while for `INSERT` or `UPDATE`, redo data is returned.

	Supported column compression: Row compression and data page compression are already supported, but if small fields in a table are read and written frequently while big fields are not, both of the compression methods waste a lot of computing resources. In contrast, column compression can compress big fields that are infrequently accessed and reduce the space for storing whole rows of fields, so as to improve read and write access efficiency. Supported querying the value of the `character_set_client_handshake` parameter. Supported the manual cleaning of page cache occupied by log files by using the `posix_fadvise()` function based on the sliding window technique, so as to lower the memory pressure on the operating system and improve instance stability. Performance optimizations Optimized the parallelism of CREATE INDEX: A merge sort is needed in a temp table in the process of creating indexes, which is time-consuming. The parallel temp-table merge sort algorithm is now supported to reduce the time by more than 50%. Optimized the mechanism of scanning and flushing the dirty pages tracked in the flush list, so as to solve the performance fluctuation issue while creating indexes and thus improve the system stability. Bug fixes Fixed the memory leak issue. Implemented the JSON bug fixes provided in MySQL 8.0 to improve the stability of using JSON. Fixed the error (error code: 1032) caused by hash scans. Fixed concurrency security issues caused by hotspot update. Implemented the gcol bug fixes provided by MySQL in batches. Fixed the failure to compare DateTime data with String data in some cases. Fixed the bug where file handles cannot be released if source-replica buffer pool sync is enabled. Fixed the deadlocks caused by setting the `offline_mode` parameter and creating connections at the same time. Fixed the crashes caused by the `m_end_range` parameter incorrectly set in concurrent range queries. Fixed the issue where it takes a long time to execute an `UPDATE` statement on a temp table if a JSON column appears in the `GROUP BY` clause.
20201231	New features Supported using `NOWAIT` and `SKIP LOCKED` in `SELECT FOR UPDATE/SHARE`. Supported dynamic setting of thread pooling mode or connection pooling mode by using the `thread_handling` parameter. Supported source-replica buffer pool sync. Supported monitoring of user connection status. Monitoring items include sync/async IO, memory, log size, CPU time, and lock duration. Performance optimizations Optimized the transaction subsystem to improve the high concurrency performance. Optimized the time to start crash recovery for large transactions. Optimized redo log flushing. Optimized the buffer pool initialization time. Optimized UTF8/UTF8MB4 string efficiency.

	Optimized audit performance. Revoked the restriction on the value of `gtid_purged` being empty. Optimized the backup lock. `LOCK TABLES FOR BACKUP`, `LOCK BINLOG FOR BACKUP`, and `UNLOCK BINLOG` are supported. `FLUSH TABLES WITH READ LOCK` is used to take a backup of the database, but it blocks the whole database from providing service. In contrast, the three statements above use a lightweight backup lock to ensure data consistency during physical/logical backup while allowing the database to providing service. Optimized the `drop table` operations on big tables. Bug fixes Fixed the hang issue when querying `performance_schema`. Fixed the overflow issue of the `digest_add_token` function. Fixed the crash caused by ibuf access when the `TRUNCATE TABLE` command was executed. Fixed the query correctness issue caused by const propagation when `LEFT JOIN` statement is used.
20200930	Performance optimizations Optimized the backup lock. `FLUSH TABLES WITH READ LOCK` is used to take a backup of the database, but it blocks the whole database from providing service. Therefore, a lightweight backup lock is provided in this version. Optimized the `drop table` operations on big tables. The `innodb_fast_ahi_cleanup_for_drop_table` parameter helps significantly reduce the time it takes to clean up adaptive hash indexes when dropping big tables. Bug fixes Fixed the crash caused by ibuf access when TRUNCATE TABLE was executed. Fixed cold backup failures when the quick column adding feature was enabled. Fixed performance degradation caused by frequently releasing InnoDB memory table objects. Fixed the query correctness issue caused by const propagation when `LEFT JOIN` statement is used. Fixed the core issue caused by rule class name conflict between SQL throttling and query rewrite. Fixed the concurrent update issue caused by the `INSERT ON DUPLICATE KEY UPDATE` statement in multiple sessions. Fixed the `duplicate key error` caused by concurrent INSERTs when `auto_increment_increment` is used. Fixed the crashes caused by evicting InnoDB memory objects. Fixed concurrency security issues caused by hotspot update. Fixed the coredump issue when enabling the thread pool after jemalloc was upgraded to v5.2.1. Fixed the incomplete audit log issue caused by fwrite error-free handling. Fixed the issue where `mysqld_safe` failed to print logs when it was started by a root user. Fixed the increase in the size of the DDL log file caused by `ALTER TABLE EXCHANGE PARTITION`.
20200701	Bug fixes Fixed the INNOBASE_SHARE index mapping error.

20200630	New features Supported using `NOWAIT` and `SKIP LOCKED` in `SELECT FOR UPDATE/SHARE` statements. Supported large transaction optimization, which can solve such problems as source-replica delay and backup failures caused by large transactions. Optimized audit performance to support async audit. Bug fixes Fixed the overflow of the `digest_add_token` function. Fixed the instance crash caused by `insert blob`. Fixed the source-replica replication interruption when a hash scan failed to find the record while updating the same row in an event. Fixed the hang issue when querying `performance_schema`.
20200331	New features Added the official MySQL 5.7.22 JSON series functions. Supported the hotspot update feature as described in Real-Time Session for ecommerce flash sale scenarios. Supported the SQL throttling feature as described in Real-Time Session. Supported encryption with custom KMS keys. Bug fixes Fixed the crash caused by phrase search under multi-byte character sets in full-text index. Fixed the crash of the CATS lock scheduling module in high-concurrency scenarios.
20190830	New features Supported skipping the corrupted data and continuing to parse when a binlog is corrupted. If the source instance and binlog are both damaged, this feature helps restore data from the replica database for use as much as possible. Supported syncing data from non-GTID to GTID mode. Supported querying the "user thread memory usage" by executing the `SHOW FULL PROCESSLIST` statement. Supported quick column adding for tables as described in Overview. This feature does not replicate the data or use disk capacity/IO, and can implement changes in real time during peak hours. Supported persistent auto-increment values. Bug fixes Fixed the issue where replication would be interrupted if the column name in a `GRANT` statement contained reserved words. Fixed the issue where SQL execution efficiency dropped when reverse scan was performed on a partitioned table. Fixed the issue where the query result had an exception due to data inconsistency when using virtual column index and primary key. Fixed the issue where data was missing due to InnoDB primary key range queries. Fixed the issue where the system crashed when a DDL statement was executed for a table with spatial indexes.

	Fixed the issue where source-replica disconnection occurred when the binlog size was too large and the file length in the heartbeat information exceeded the limit. Fixed the issue where other events could not be executed as scheduled when an event was deleted. Fixed the issue where the aggregate query result was incorrect.
20190615	New features Supported transparent data encryption (TDE).
20190430	Bug fixes Fixed the issue where null pointer reference occurred when the LONGTEXT feature was used in subqueries. Fixed the issue where source-replica disconnection occurred due to hash scan. Fixed the issue where the replica I/O thread was interrupted due to source binlog switch. Fixed the crash caused by the use of `NAME_CONST`. Fixed the illegal mix of collation error caused by character set.
20190203	New features Supported async drop of big tables. You can clear files asynchronously and slowly to avoid business performance fluctuation caused by dropping big tables. To apply for this feature, submit a ticket. Supported CATS lock scheduling. Supported creating and dropping temp tables and CTS syntax in transactions when GTID is enabled. To apply for this feature, submit a ticket. Supported implicit primary keys. To apply for this feature, submit a ticket. Supported users without super privileges to kill sessions of other users by configuring the `cdb_kill_user_extra` parameter (default value: `root@%`). Supported enterprise-grade encryption functions. To apply for this feature, submit a ticket. Bug fixes Fixed the issue where replication was interrupted when binlog cache file ran out of space. Fixed the hard error when `fsync` returned `EIO` and retries were made repeatedly. Fixed the issue where replication was interrupted and could not be recovered due to GTID holes.
20180918	New features Supported automatic killing of idle transactions to reduce resource conflicts. To apply for this feature, submit a ticket. Supported automatically changing the storage engine from MEMORY to InnoDB: If the global variable `cdb_convert_memory_to_innodb` is `ON`, the engine will be changed from MEMORY to InnoDB when a table is created or modified. Supported invisible indexes. Supported memory management with jemalloc, which can replace the jlibc memory management module to reduce memory usage and improve allocation efficiency. Performance optimizations Optimized binlog switch to reduce the `rotate` lock duration and improve system performance. Accelerated the crash recovery.

	Bug fixes Fixed the issue where an event became invalid due to source-replica switch. Fixed the crash caused by `REPLAY LOG RECORD`. Fixed the issue where the query result was incorrect due to loose index scans.
20180530	New features Supported SQL auditing. Supported table-level concurrent replication. To apply for this feature, submit a ticket. Performance optimizations Optimized replica instance locks to improve the sync performance of replica instances. Optimized the pushdown of the `SELECT ... LIMIT` statement. Bug fixes Fixed the issue where switch failed due to inconsistent positions between `relay_log_pos` and `master_log_pos`. Fixed the crash caused by `Crash on UPDATE ON DUPLICATE KEY`. Fixed the `Invalid escape character in string.` error when a JSON column was imported.
20171130	New features Supported the `information_schema.metadata_locks` view to query the MDL grant and wait status in the current instance.Supported the `ALTER TABLE NO_WAIT | TIMEOUT` syntax to grant DDL operations wait timeout. To apply for this feature, submit a ticket. Supported thread pool. To apply for this feature, submit a ticket. Bug fixes Fixed the error of `innodb_buffer_pool_pages_data` parameter overflow by calculating it based on `bytes_data`. Fixed the issue where speed limit plugin became unavailable in async mode.

Minor Version	Description
20220303	Bug fixes Fixed the abnormal release when the memory allocated by `mem_strdup` was used for `row_mysql_truncate_t::file_name` during async drop of big tables.
20220302	Bug fixes Fixed the memory leak issue in `sql_update.cc`.
20220301	New features Supported dynamically configuring the spin cycle (0–100) with the dynamic parameter `innodb_spin_wait_pause_multiplier`. This parameter is used for temporary adjustment and does not support fixing the change through the console. Supported printing deadlock loop information.After this feature is enabled through the parameter `innodb_print_dead_lock_loop_info`, when a deadlock occurs, you can run `show engine innodb status` to view the deadlock loop information. Bug fixes

	Fixed the issue where anonymous GTID transactions were generated in memory tables after replica restart. Fixed the issue where upgrade failed due to the missing `root@localhost` permission. Fixed the issue where the values of monitoring variables such as `innodb_row_lock_current_waits` were abnormal. Fixed the SQL type mapping error in the audit plugin.
20211030	New features Supported large transaction replication optimization. Performance optimizations Accelerated the application of hash scan. Bug fixes Fixed the OOM caused by a large number of table queries. Fixed the infinite loop error caused by setting `innodb_thread_concurrecy` to 0. Fixed the issue where there were no statistics information for long records. Fixed the SBM jump error. Fixed the `LOCK_binlog_end_pos hang` error.
20210630	New features Supported large transaction replication optimization. Bug fixes Fixed the incorrect copy when Index Merge was enabled. Fixed the issue where the replication would be interrupted if the execution of CREATE TABLE SELECT was interrupted when `cdb_more_gtid_feature_supported` was enabled in row mode. Fixed the bug that `max(id)` was greater than AUTO_INCREMENT in SHOW CREATE TABLE.
20201231	Bug fixes Fixed the error (error code: 1032) caused by hash scans. Fixed the issue where the source-replica auto-increment values were inconsistent due to the `REPLACE INTO` statement in `ROW` format. Fixed the memory leak caused by not freeing up the memory requested for parsing SQL statements. Fixed the issue where the sql_mode check is skipped when running `CREATE TABLE AS SELECT`. Fixed the issue where the `sql_mode` check was skipped when inserting default values. Fixed the issue where the `sql_mode` check was skipped when running UPDATE with bound parameters.
20200915	New features Supported the SQL throttling feature as described in Real-Time Session. Performance optimizations Optimized the initialization acceleration of buffer pool. Bug fixes Fixed the hang issue of `rename table` on both source and replica. Fixed the crash when `event_scheduler` was set to `disable` and `cdb_skip_event_scheduler` was changed from `on` to `off`.

	Fixed the `sync_wait_array` assertion failure when the maximum number of connections of `tencentroot` was not counted in `srv_max_n_threads`. Fixed the crash of source-replica parallel replication caused by the system table structure inconsistency between TencentDB for MySQL 5.6 and other cloud vendors' MySQL 5.6. Fixed the `INSERT ON DUPLICATE KEY UPDATE THE WRONG ROW` error. Fixed the `index_mapping` error. Fixed the MTR failure. Fixed the source-replica replication interruption when a hash scan failed to find the record while updating the same row in an event.
20190930	New features Supported querying the user thread memory usage by executing the `SHOW FULL PROCESSLIST` statement. Bug fixes Fixed GTID holes caused by the replication filter of the replica. Fixed the issue where source-replica disconnection occurred when the binlog size was too large and the file length in the heartbeat information exceeded the limit. Fixed the illegal mix of collation error caused by character set. Fixed the issue where the source-replica disconnection occurred due to hash scan. Fixed the crash caused by the use of `NAME_CONST`. Fixed the issue where the replica I/O thread was interrupted due to source binlog switch. Fixed the error of incompatible backups due to `innodb_log_checusum`.
20190530	Bug fixes Fixed the issue where dirty data might be read in RC mode. Fixed the issue where replica instance replay might fail due to the drop of temp table. Fixed the deadlock issue under high concurrency.
20190203	New features Supported async drop of big tables. You can clear files asynchronously and slowly to avoid business performance fluctuation caused by dropping big tables. To apply for this feature, submit a ticket. Supported users without super privileges to kill sessions of other users by configuring the `cdb_kill_user_extra` parameter (default value: `root@%`). Supported creating and dropping temp tables and CTS syntax in transactions when GTID is enabled. To apply for this feature, submit a ticket. Performance optimizations Optimized the replication and replay of partitioned tables to improve efficiency. Bug fixes Fixed the source-replica data inconsistency issue caused by insufficient temporary space. Fixed the issue of suspended hot record updates. Fixed the issue where the value of `Seconds_Behind_Master` was abnormal during concurrent replication.
20180915	New features

	Supported automatically changing the storage engine from MEMORY to InnoDB: If the global variable `cdb_convert_memory_to_innodb` is `ON`, the engine will be changed from MEMORY to InnoDB when a table is created or modified. Supported automatic killing of idle transactions to reduce resource conflicts. To apply for this feature, submit a ticket. Bug fixes Fixed the crash caused by `REPLAY LOG RECORD`. Fixed the error of time data inconsistency between source and replica due to decimal precision issues.
20180130	New features Supported thread pool. To apply for this feature, submit a ticket. Supported dynamically modifying replication filtering rules for replica nodes. Performance optimizations Reduced performance fluctuation caused by `DROP TABLE`. Bug fixes Fixed the database crash caused by authentication password strings.
20180122	New features Supported SQL auditing. Bug fixes Fixed the integer overflow issue. Fixed the error caused by queries using full-text index. Fixed the issue where the replica crashed during replication.
20170830	Bug fixes Fixed the issue where binlog speed limit became invalid in async mode. Fixed the issue where the `buffer_pool` status was abnormal. Fixed the issue where `SEQUENCE` and implicit primary key conflicted.
20170228	Bug fixes Fixed the character encoding bug in `DROP TABLE`. Fixed the issue where a table contained symbols like decimal points or `replicate-wild-do-table` couldn't be used to filter databases correctly. Fixed the issue where SQL threads exited too early after the replica had a `rotate` event.
20161130	Performance optimizations Split the `lock_log` lock to reduce the time used by lock logs and improve the concurrency performance. Separated the ACK thread of the source to reduce the response time. Prohibited the user thread from being killed while waiting for ACK in order to prevent phantom reads. Fixed the unnecessary `lock_sync` lock when `sync_binlog != 1`.

TXRocks Kernel Release Notes

Last updated：2023-09-13 15:41:31

This document describes the version updates of the TXRocks kernel.

Note

For more information on how to upgrade the minor kernel version of a TencentDB for MySQL instance, see Upgrading Kernel Minor Version.

When you upgrade the minor version, some minor versions may be under maintenance and cannot be selected. The minor versions available in the console shall prevail.

MySQL 8.0 Kernel Version Release Notes

MySQL 5.7 Kernel Version Release Notes

Minor Version	Note
20230401	New features: By default, the TokuDB engine in the table creation statement is converted to the RocksDB engine.

Minor Version	Note
20230401	New features: By default, the TokuDB engine in the table creation statement is converted to the RocksDB engine.

Database Proxy Kernel Release Notes

Last updated：2024-07-22 11:10:10

This document describes the kernel version updates of the TencentDB for MySQL database proxy.

Note:

If the MySQL kernel version requirements are not met, you can upgrade the kernel version of your database first as instructed in Upgrading Kernel Minor Version.

Database Proxy Version	MySQL Kernel Version Requirements	Description
1.3.7	MySQL 5.7 20211030 and later MySQL 8.0 20211202 and later	**Fixes** Fixed the routing error of the `select for update` statement in some cases. Modified the `select @@read_only` statement and made it possible to be routed to the source database. This prevents some frameworks that use read_only flags from misjudging the database proxy as unwritable. Fixed the database proxy node exceptions caused by a database instance HA in some scenarios.
1.3.4	MySQL 5.7 20211030 and later MySQL 8.0 20211202 and later	**Fixes** Fixed the issue where the 'show processlist' command returned incomplete data.
1.3.3	MySQL 5.7 20211030 and later MySQL 8.0 20211202 and later	**Fixes** Fixed the issue where an error was reported when the session connection pool reused connections to send `change_user` to the backend, and the issue where the PREPARE statement was not correctly handled by the database proxy after a new connection was established.
1.3.2	MySQL 5.7 20211030 and later MySQL 8.0 20211202 and later	**Fixes** Fixed the issue where `execute` statement didn't have a parameter type
1.3.1	MySQL 5.7 20211030 and later MySQL 8.0 20211202 and later	**Feature updates** Allowed instances with a weight of 0 to sustain read requests when the weight of all valid instances under the database proxy is 0.

		Supported the multi-AZ deployment architecture where read-only instances can be mounted across AZs. Provided the read-only mode. Supported transaction split. Supported momentary disconnection prevention, i.e., connection persistence, where the client will not be disconnected when a database instance HA switch occurs because of a scheduled task.
1.2.1	MySQL 5.7 20211030 and later MySQL 8.0 20211202 and later	**Feature updates** Supported the `db lower_case_table_names` parameter, indicating not to verify the letter case by default. Supported returning error messages in the query stage when errors occur during database proxy connection establishment.
1.1.3	MySQL 5.7 20211030 and later MySQL 8.0 20211202 and later	**Feature updates** Supported the use of hint routing information in the `COM_PREPARE` packet preprocessed by MySQL. After hint is used in `PREPARE` to specify the routing target, subsequent `execute` packets will be sent to the specified backend node. **Fixes** Fixed the issue where the frontend connection was reset immediately after source-replica switch of the source instance on the database proxy was performed. Fixed the issue where load balancing might fail when read-only instances exceeded the latency threshold. Routing will resume normally when the read-only instance latency falls below the threshold. Fixed the issue where MySQL 8.0 might return incorrect handshake information and cause connection failures.
1.1.2	MySQL 5.7 20211030 and later MySQL 8.0 20211130 and later	**Feature updates** Supported MySQL 8.0. Supported the connection pool feature at the connection level, which is useful in scenarios where non-persistent connections to the database are frequently established. The database proxy will save connections and reuse them during subsequent connection establishments. Supported the reconnection feature for read-only instances. In persistent connection scenarios, when a read-only instance is restarted or added, the database proxy will automatically establish a connection and restore routing to it. Updated the internal memory management mechanism to reduce the memory usage. **Fixes**

		Fixed the issue where the client connection persisted after the backend connection was closed due to timeout. Fixed the issue where the internal cache might cause excessively rapid increase of the memory utilization. Fixed the occasional issue where packets in an incorrect format were returned.
1.0.1	MySQL 5.7 20201230 and later	**Feature updates** Supported MySQL 5.7. Supported read/write separation. Supported read weight assignment in read/write separation. Supported the configuration of source-replica replication delay threshold. Routing to a read-only instance will be stopped if its delay exceeds the thresholds and will be recovered after it drops below the threshold. If the source-replica replication is interrupted, disconnected read-only instances will be removed directly. Supported the configuration of the least number of read-only instances. When read-only instances are removed, if the number is set to N, at least N instance(s) will be retained for routing. Supported the failover configuration, which is enabled by default. If it is disabled and the read weight of the source instance is 0, after all read-only instances are removed, an error will be reported for read requests. If failover is enabled and the read weight of the source instance is not 0, requests will be routed to the source instance. Supported using the HINT syntax to specify routing nodes.

Functionality Features
Killing Idle Transactions Automatically

Last updated：2024-07-22 11:12:05

Overview

This feature kills transactions that have been idle for the specified time period to release resources in time.

Supported Versions

Kernel version: MySQL 5.6 20180915 and above.
Kernel version: MySQL 5.7 20180918 and above.
Kernel version: MySQL 8.0 20200630 and above.

Use Cases

If a connection starts a transaction (explicitly using `begin` / `start transaction` or implicitly) but no new statement has been executed for the specified threshold period, the connection will be killed.

Instructions

Use the `cdb_kill_idle_trans_timeout` parameter to enable or disable the feature. If it is `0`, the feature is disabled; otherwise, a connection idle for `cdb_kill_idle_trans_timeout` or `wait_timeout` seconds, whichever is smaller, will be killed. (`wait_timeout` is a session parameter.)

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description
cdb_kill_idle_trans_timeout	Yes	ulong	0	[0, 31536000]	If it is `0`, the feature is dis otherwise, a transaction idl `cdb_kill_idle_trans` seconds will be killed.

Note:

Currently, you cannot directly modify the values of the above parameter. If needed, submit a ticket for assistance.

Parallel Replication

Last updated：2024-07-22 11:30:52

Overview

Prior to MySQL 5.6, the source node syncs binlogs and the replica node replays binlogs, both in the single-thread mode. MySQL 5.6 and later versions support the DATABASE/LOGICAL_CLOCK parallel replication scheme, but the granularity is too large to achieve expected parallel replication in many cases.
Tencent Cloud's TXSQL kernel team has optimized the parallel replication scheme. Table parallel replication is now supported, improving parallelism and reducing source-replica delay.

Supported Versions

Kernel version: MySQL 8.0 20201230 and later.
Kernel version: MySQL 5.7 20180530 and later.
Kernel version: MySQL 5.6 20170830 and later.

Use Cases

This feature is suitable for use cases where optimizing the parallelism of some loads can speed up the binlog replay at the replica node, thus reducing the source-replica delay.

Instructions

For MySQL 5.6 and 5.7, you can enable this feature by setting the `slave_parallel_type` parameter to the newly added value `TABLE`. MySQL 8.0 does not support the TABLE mode.
Additionally, the `cdb_slave_thread_status` table is added to the `information_schema` database to display the thread status of the replica node.
MySQL 5.6 parameter description
MySQL 5.7 parameter description
MySQL 8.0 parameter description

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description

slave_parallel_type	Yes	char*	SCHEMA	SCHEMA/TABLE	The level of parallel replication on the replica node. SCHEMA: Replication events of different schemas can be executed in parallel. TABLE: Replication events of different tables can be executed in parallel.

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range
slave_parallel_type	Yes	char*	LOGICAL_CLOCK	DATABASE/TABLE/LOGICAL_CLOCK

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Desc
slave_parallel_type	Yes	char*	LOGICAL_CLOCK	DATABASE/LOGICAL_CLOCK	The repli repli DAT Repl of di

					data exec LOG Repl of th clock can para

Dynamic Thread Pool

Last updated：2024-02-18 11:24:04

Overview

The thread pool (Thread_pool) uses a certain number of worker threads to process connection requests, which is typically used in scenarios with online transaction processing (OLTP) workloads. However, when many requests are slow queries, worker threads will be blocked by high-latency operations and fail to quickly respond to new requests. As a result, the system throughput of the thread pool mode is lower than that of the traditional one-thread-per-connection (Per_thread) mode.
The Per_thread and Thread_pool modes have their advantages and disadvantages, so the system needs to flexibly switch between them based on business types. Unfortunately, the mode switch must be completed by restarting the server (during peak hours in most cases), adversely affecting the business.
To allow users to flexibly switch between Per_thread and Thread_pool, TencentDB for MySQL has introduced the optimization of thread pool dynamic switch, that is, to enable or disable the thread pool without restarting the database service.

Supported Versions

Kernel version: MySQL 8.0 20201230 and above.
Kernel version: MySQL 5.7 20201230 and above.

Use Cases

This feature is suitable for the business which is sensitive to performance and needs to flexibly change the database working mode based on the business type.

Performance Impact

Switching from the thread pool mode to the one-thread-per-connection mode won't block queries or affect database performance.
Switching from the one-thread-per-connection mode to the thread pool mode under extremely high QPS and persistent high pressure may block requests because the thread pool is disabled before the switch.

Solution 1: you can increase the `thread_pool_oversubscribe` parameter and decrease the `thread_pool_stall_limit` parameter to quickly enable the thread pool. After the blocked SQL queries are processed, you can restore the parameters to their original values as needed.
Solution 2: if SQL queries start to be blocked, you can suspend or reduce service traffic for a few seconds, wait for thread pool enablement to complete, and then resume the continuous high-pressure service traffic.

Instructions

You can use the `thread_handling_switch_mode` parameter to control whether to dynamically change the thread working mode. Parameter values are described as follows:

Valid Value	Description
disabled	The mode cannot be changed dynamically.
stable	The mode can only be changed for new connections.
fast	(Default value) The mode can be changed for new connections and new requests.
sharp	Active connections will be killed in order to force the user to reconnect so that the mode can be changed quickly.

The `show threadpool status` command displays the following new status:
connections_moved_from_per_thread: the number of connections switched from Per_thread to Thread_pool.
connections_moved_to_per_thread: the number of connections switched from Thread_pool to Per_thread.
events_consumed: the total number of events consumed by the worker thread group in each thread pool. After the thread working mode is switched from Thread_pool to Per_thread, the total number of events won't increase any more.
average_wait_usecs_in_queue: the average time each event waits in the queue.
The `show full processlist` command displays the following new status:
Moved_to_per_thread: the number of times that the connection is switched to Per_thread.
Moved_to_thread_pool: the number of times that the connection is switched to Thread_pool.

Parameter Status Description

Thread pool parameters are described as follows:

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range

thread_pool_idle_timeout	Yes	uint	60	[1, UINT_MAX]
thread_pool_oversubscribe	Yes	uint	High Stability Parameter Template: 10 High Performance Parameter Template: 16	[3,32]
thread_pool_size	Yes	uint	The number of CPUs on the current machine	[1,1000]
thread_pool_stall_limit	Yes	uint	500	[10, UINT_MAX]
thread_pool_max_threads	Yes	uint	100000	[1,100000]
thread_pool_high_prio_mode	Yes, session	enum	transactions	transactions\\statement\\none

thread_pool_high_prio_tickets	Yes, session	uint	UINT_MAX	[0, UINT_MAX]
threadpool_workaround_epoll_bug	Yes	bool	false	true/false

The `show threadpool status` command displays the following status:

Status	Description
groupid	Thread group ID
connection_count	The number of user connections in the thread group
thread_count	The number of worker threads in the thread group
havelistener	Whether the thread group has a listener
active_thread_count	The number of active worker threads in the thread group
waiting_thread_count	The number of worker threads calling wait_begin in the thread group
waiting_threads_size	The number of sleeping worker threads waiting to be woken up in the thread group when there is no network event to handle (such worker threads will wait for `thread_pool_idle_timeout` seconds before being automatically killed)
queue_size	The length of the ordinary queue of the thread group
high_prio_queue_size	The length of the high priority queue of the thread group
get_high_prio_queue_num	The total number of times that events in the thread group are removed from the high priority queue
get_normal_queue_num	The total number of times that events in the thread group are removed from the ordinary queue
create_thread_num	The total number of worker threads created in the thread group
wake_thread_num	The total number of worker threads in the thread group awakened from the waiting_threads queue

oversubscribed_num	The number of times that worker threads are ready to go to sleep because the thread group is oversubscribed
mysql_cond_timedwait_num	The total number of times that worker threads in the thread group enter the waiting_threads queue
check_stall_nolistener	The total number of times that no listener is detected in the thread group in the stall check performed by the timer thread
check_stall_stall	The total number of times that the thread group is considered stalled in the stall check performed by the timer thread
max_req_latency_us	The maximum time in milliseconds for a user connection to wait in the queue in the thread group
conns_timeout_killed	The total number of times that user connections in the thread group are killed because there has been no new message on the client for the threshold period (net_wait_timeout)
connections_moved_in	The total number of connections migrated from other thread groups to this thread group
connections_moved_out	The total number of connections migrated from this thread group to other thread groups
connections_moved_from_per_thread	The total number of connections switched from the one-thread-per-connection mode to this thread group
connections_moved_to_per_thread	The total number of connections switched from this thread group to the one-thread-per-connection mode
events_consumed	The total number of events processed by the thread group
average_wait_usecs_in_queue	The average waiting time of all events in the queue in the thread group

NOWAIT

Last updated：2024-07-22 11:31:50

Overview

DDL statements support `NO_WAIT` and `WAIT` options. If a DDL statement with `WAIT` enabled fails to obtain an MDL lock, it will wait for `WAIT` seconds before it directly returns the query result. If a DDL statement with `NO_WAIT` enabled, it will directly return the query result without waiting for the MDL lock.
SELECT FOR UPDATE statements support `NOWAIT` and `SKIP LOCKED` options. If target rows are locked by another transaction, a SELECT FOR UPDATE statement is supposed to wait for the transaction to release the lock. But in some use cases like flash sales, you do not want to wait for a lock. You can use `SKIP LOCKED` to skip locked rows (as a result, the locked rows won't be returned in the query result set) or `NOWAIT` to return an error without waiting for the lock.
Note that `NO_WAIT` and `NOWAIT` are different keywords.

Supported Versions

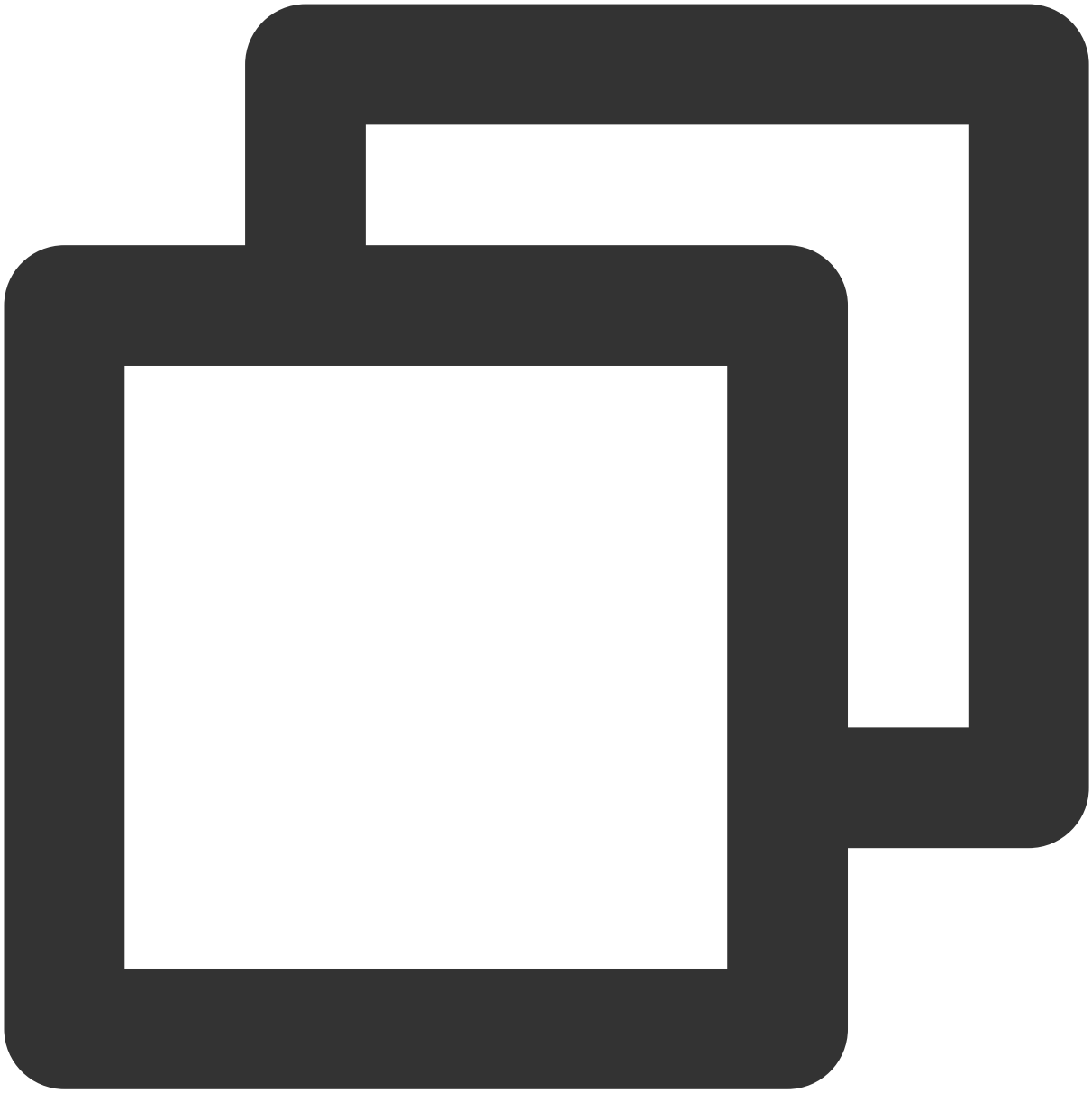
`NO_WAIT` and `WAIT` in DDL statements are supported in kernel version MySQL 5.7 20171130 and above.
`NOWAIT` and `SKIP LOCKED` in SELECT FOR UPDATE statements are supported in kernel version MySQL 5.7 20200630 and above (not just limited to MySQL 8.0 that natively supports the feature).

Use Cases

Currently, DevAPI/XPlugin does not support using `SKIP LOCKED` or `NOWAIT` in SELECT FOR UPDATE/SHARE statements. Note that `NO_WAIT` in DDL statements and `NOWAIT` in SELECT FOR UPDATE statements are different keywords for historical reasons.

Instructions

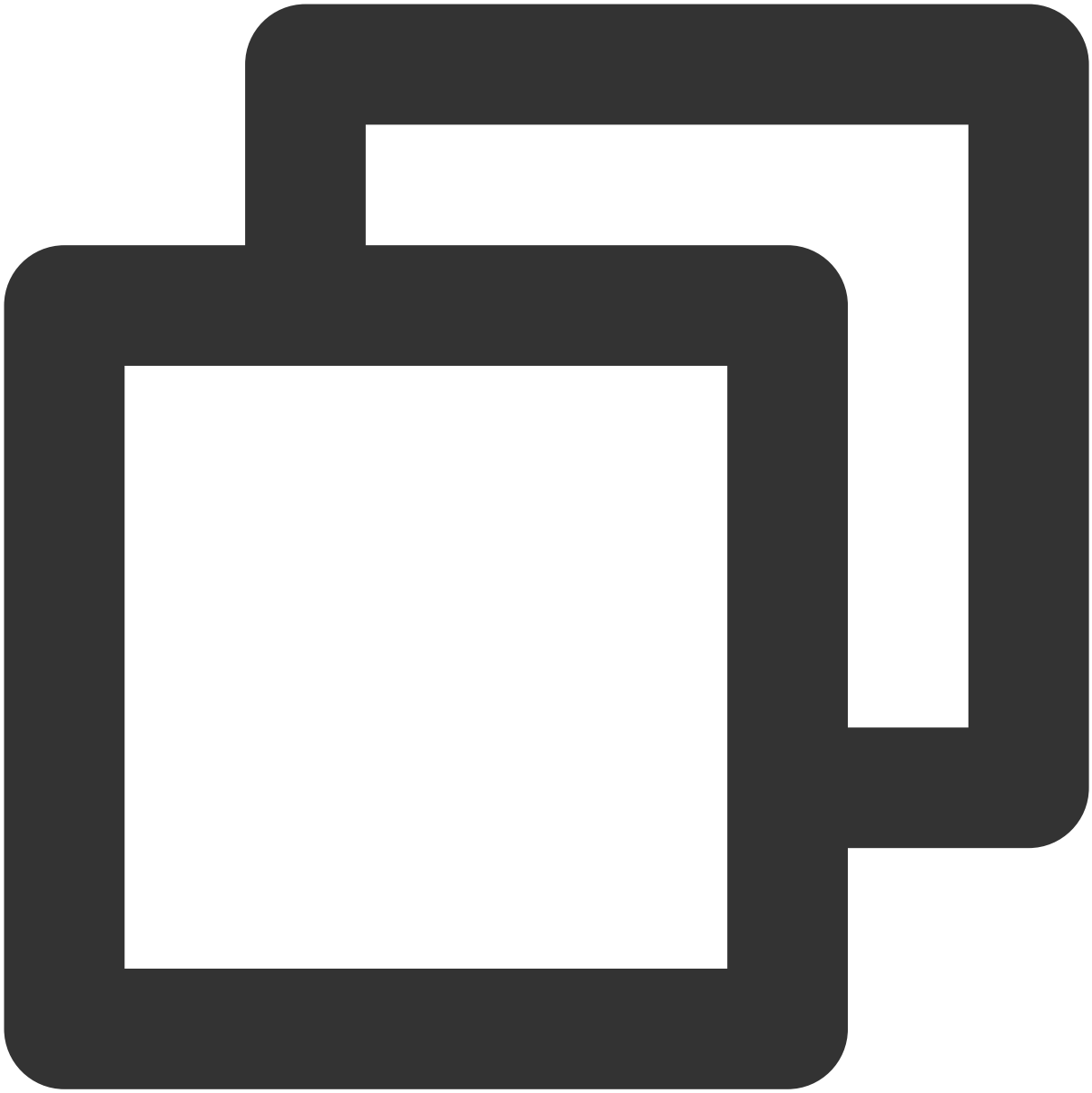
SELECT FOR UPDATE NOWAIT/SKIP LOCKED



```
################session 1################
MySQL [test]> create table t1(seat_id int, state int, primary key(seat_id)) engine=
Query OK, 0 rows affected (0.03 sec)

MySQL [test]> INSERT INTO t1 VALUES(1,0), (2,0), (3,0), (4,0);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

MySQL [test]> begin;
Query OK, 0 rows affected (0.01 sec)
```

```
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE;
+---------+-------+
| seat_id | state |
+---------+-------+
|       1 |     0 |
|       2 |     0 |
+---------+-------+
2 rows in set (0.00 sec)

###############session 2###############
MySQL [test]> SET SESSION innodb_lock_wait_timeout=1;
Query OK, 0 rows affected (0.00 sec)
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE NOWAIT;
ERROR 5010 (HY000): Do not wait for lock.
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE SKIP LOCKED;
+---------+-------+
| seat_id | state |
+---------+-------+
|       3 |     0 |
|       4 |     0 |
+---------+-------+
2 rows in set (0.00 sec)

MySQL [test]> SELECT * FROM t1 WHERE seat_id > 0 LIMIT 2 FOR UPDATE NOWAIT;
ERROR 5010 (HY000): Do not wait for lock.
MySQL [test]> SELECT * FROM t1 WHERE seat_id > 0 LIMIT 2 FOR UPDATE SKIP LOCKED;
+---------+-------+
| seat_id | state |
+---------+-------+
|       3 |     0 |
|       4 |     0 |
+---------+-------+
2 rows in set (0.00 sec)

MySQL [test]> commit;
Query OK, 0 rows affected (0.00 sec)
```

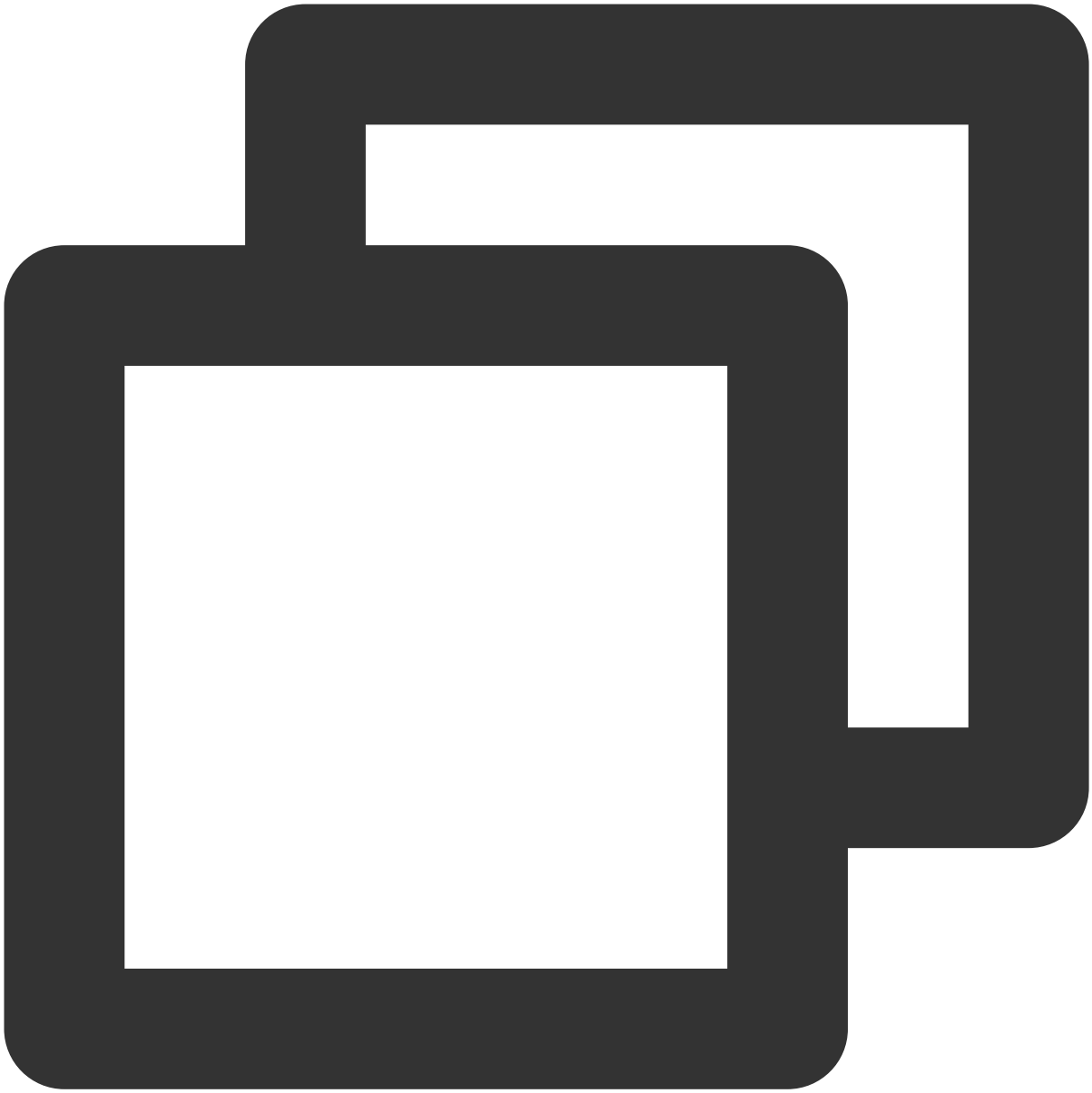
SELECT FOR SHARE NOWAIT/SKIP LOCKED



```
################session 1################
MySQL [test]> begin;
Query OK, 0 rows affected (0.01 sec)

MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE;
+---------+-------+
| seat_id | state |
+---------+-------+
|       1 |     0 |
|       2 |     0 |
+---------+-------+
```

```
2 rows in set (0.00 sec)

###############session 2###############
MySQL [test]> SET SESSION innodb_lock_wait_timeout=1;
Query OK, 0 rows affected (0.00 sec)

MySQL [test]> begin;
Query OK, 0 rows affected (0.00 sec)

MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 LOCK IN SHARE MODE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE NOWAIT;
ERROR 5010 (HY000): Do not wait for lock.
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE SKIP LOCKED;
+---------+-------+
| seat_id | state |
+---------+-------+
|       3 |     0 |
|       4 |     0 |
+---------+-------+
2 rows in set (0.00 sec)

MySQL [test]> commit;
Query OK, 0 rows affected (0.00 sec)
```

NO_WAIT and WAIT in DDL statements



```
ALTER TABLE `table` [NO_WAIT | WAIT [n]] `command`;
DROP TABLE `table` [NO_WAIT | WAIT [n]];
TRUNCATE TABLE `table` [NO_WAIT | WAIT [n]];
OPTIMIZE TABLE `table` [NO_WAIT | WAIT [n]];
RENAME TABLE `table_src` [NO_WAIT | WAIT [n]] TO `table_dst`;
CREATE INDEX `index` ON `table.columns` [NO_WAIT | WAIT [n]];
CREATE FULLTEXT INDEX `index` ON `table.columns` [NO_WAIT | WAIT [n]];
CREATE SPATIAL INDEX `index` ON `table.columns` [NO_WAIT | WAIT [n]];
DROP INDEX `index` ON `table` [NO_WAIT | WAIT [n]];
```

RETURNING

Last updated：2024-07-22 11:32:19

Overview

In some scenarios, you need to retrieve the rows manipulated by DML statements. There are generally two ways to do so:
Add a SELECT statement after the DML statement if the transaction is enabled.
Use a trigger or other complex operations.
However, running a SELECT statement increases query costs, and creating a trigger makes SQL implementation more complex and inflexible.
Therefore, TXSQL supports the RETURNING keyword to optimize such scenarios. The above requirements can be flexibly and efficiently met by appending RETURNING to a DML statement.

Supported Versions

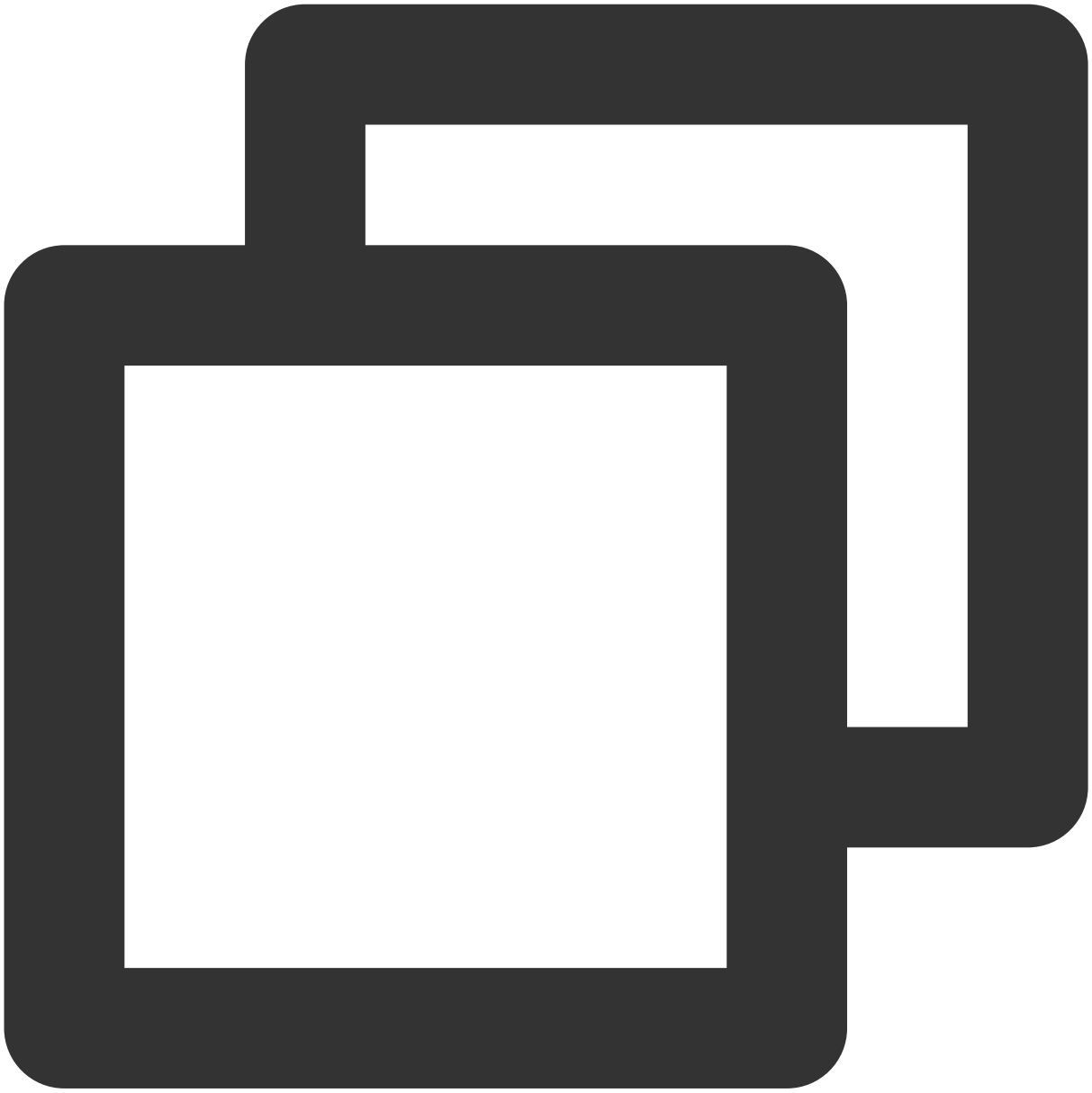
Kernel version: MySQL 5.7 20210330 and above.

Use Cases

MySQL 5.7 20210330 and above support INSERT ... RETURNING, REPLACE ... RETURNING, and DELETE ... RETURNING. The RETURNING keyword returns all rows that have been manipulated by an INSERT/REPLACE/DELETE statement. RETURNING can also be used in prepared statements and stored procedures.
Notes:
1. For DELETE ... RETURNING, the returned data rows are pre-images, while for INSERT/REPLACE ... RETURNING, they are post-images.
2. Currently, UPDATE ... RETURNING is not supported.
3. For INSERT/REPLACE ... RETURNING, columns in the outer table are currently invisible to the subquery in the RETURNING clause.
4. INSERT/REPLACE ... RETURNING only returns the value of `last_insert_id()` before the statement is executed successfully. To obtain the true value of `last_insert_id()` , you should use RETURNING to return the auto-increment column ID of the table.

Instructions

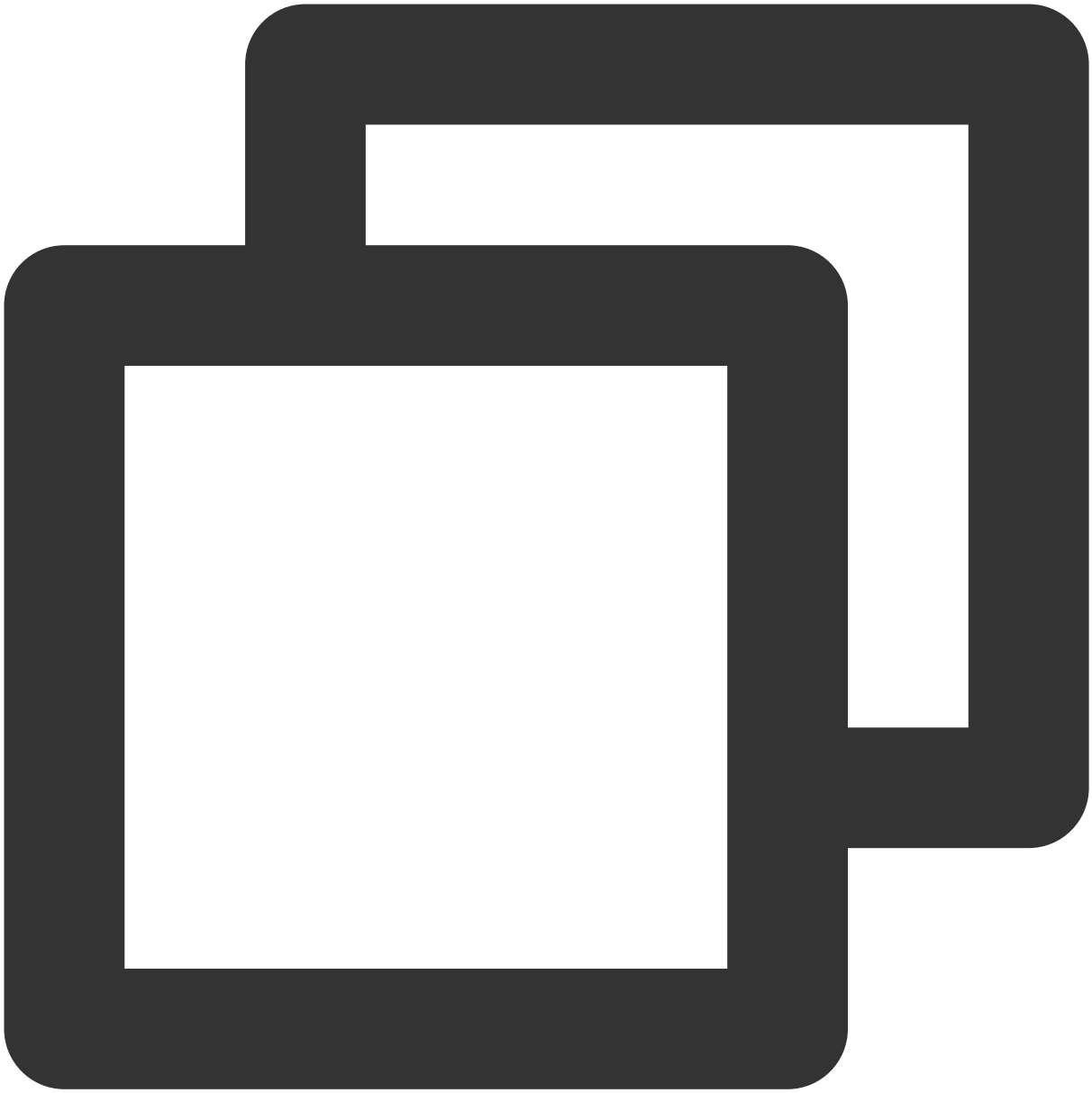
INSERT ... RETURNING



```
MySQL [test]> CREATE TABLE `t1` (id1 INT);
Query OK, 0 rows affected (0.04 sec)

MySQL [test]> CREATE TABLE `t2` (id2 INT);
Query OK, 0 rows affected (0.03 sec)
```

```
MySQL [test]> INSERT INTO  t2 (id2) values (1);
Query OK, 1 row affected (0.00 sec)

MySQL [test]> INSERT INTO t1 (id1) values (1) returning *, id1 * 2, id1 + 1, id1 *
+------+---------+---------+-------+--------------------+
| id1  | id1 * 2 | id1 + 1 | alias | (select * from t2) |
+------+---------+---------+-------+--------------------+
|    1 |       2 |       2 |     1 |                  1 |
+------+---------+---------+-------+--------------------+
1 row in set (0.01 sec)

MySQL [test]> INSERT INTO t1 (id1) SELECT id2 from t2 returning id1;
+------+
| id1  |
+------+
|    1 |
+------+
1 row in set (0.01 sec)
```

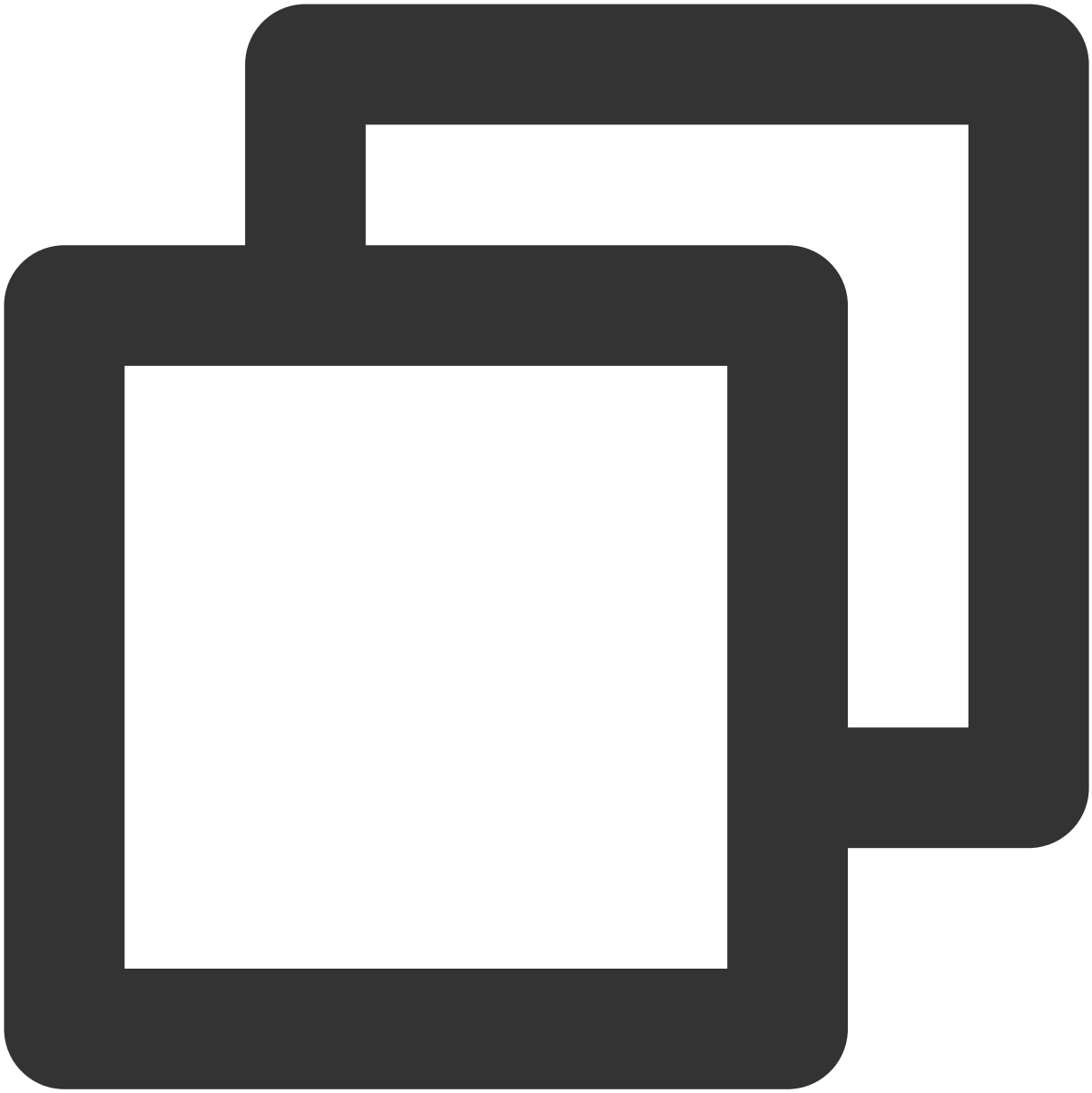
REPLACE ... RETURNING



```
MySQL [test]> CREATE TABLE t1(id1 INT PRIMARY KEY, val1 VARCHAR(1));
Query OK, 0 rows affected (0.04 sec)

MySQL [test]> CREATE TABLE t2(id2 INT PRIMARY KEY, val2 VARCHAR(1));
Query OK, 0 rows affected (0.03 sec)

MySQL [test]> INSERT INTO t2 VALUES (1,'a'),(2,'b'),(3,'c');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

MySQL [test]> REPLACE INTO t1 (id1, val1) VALUES (1, 'a');
```

```
Query OK, 1 row affected (0.00 sec)

MySQL [test]> REPLACE INTO t1 (id1, val1) VALUES (1, 'b') RETURNING *;
+-----+------+
| id1 | val1 |
+-----+------+
|   1 | b    |
+-----+------+
1 row in set (0.01 sec)
```

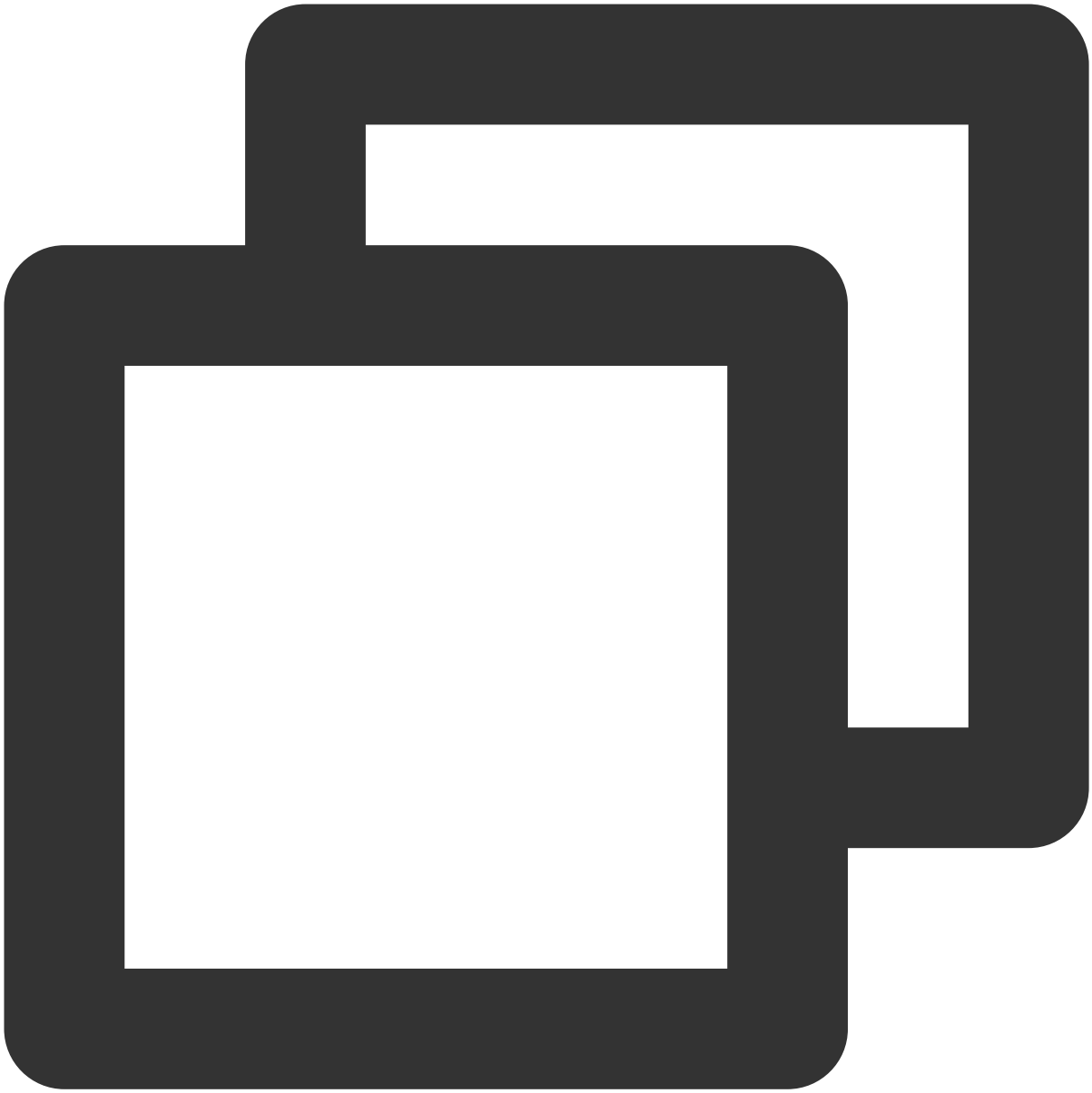
DELETE ... RETURNING



```
MySQL [test]> CREATE TABLE t1 (a int, b varchar(32));
Query OK, 0 rows affected (0.04 sec)

MySQL [test]> INSERT INTO t1 VALUES
    ->   (7,'ggggggg'), (1,'a'), (3,'ccc'),
    ->   (4,'dddd'), (1,'A'), (2,'BB'), (4,'DDDD'),
    ->   (5,'EEEEE'), (7,'GGGGGGG'), (2,'bb');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

MySQL [test]> DELETE FROM t1 WHERE a=2 RETURNING *;
```

```
+------+------+
| a    | b    |
+------+------+
|    2 | BB   |
|    2 | bb   |
+------+------+
2 rows in set (0.01 sec)

MySQL [test]> DELETE FROM t1 RETURNING *;
+------+---------+
| a    | b       |
+------+---------+
|    7 | ggggggg |
|    1 | a       |
|    3 | ccc     |
|    4 | dddd    |
|    1 | A       |
|    4 | DDDD    |
|    5 | EEEEE   |
|    7 | GGGGGGG |
+------+---------+
8 rows in set (0.01 sec)
```

Stored procedure



```
MySQL [test]> CREATE TABLE `t` (id INT);
Query OK, 0 rows affected (0.03 sec)

MySQL [test]> delimiter $$
MySQL [test]> CREATE PROCEDURE test(in param INT)
    -> BEGIN
    ->     INSERT INTO t (id) values (param) returning *;
    -> END$$
Query OK, 0 rows affected (0.00 sec)
MySQL [test]> delimiter ;
```

```
MySQL [test]> CALL test(100);
+------+
| id   |
+------+
|  100 |
+------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

Column Compression

Last updated：2024-07-22 11:34:21

Overview

Row compression and data page compression are already supported, but if small fields in a table are read and written frequently while big fields are not, both of the compression methods waste a lot of computing resources.
In contrast, column compression can compress big fields that are infrequently accessed while ignoring the frequently accessed small fields, which not only reduces the space for storing whole rows of fields but also improves the read and write access efficiency.
For example, in the employee table `create table employee(id int, age int, gender boolean, other varchar(1000) primary key (id))` , if access is frequent for small fields such as `id` , `age` , and `gender` but infrequent for the large field `other` , you can compress the `other` column. Generally, only read/write of the `other` column rather than other columns will trigger the compression and decompression of this column, which further reduces the size of the stored row data. In this way, frequently accessed small fields can be accessed more quickly, while infrequently accessed large fields can be compress to use less storage space.

Supported Versions

Kernel version: MySQL 5.7 20210330 and above.
Note:
Column compression for closed by default, if you want to use, please submit a ticket to open.

Use Cases

If a table has many frequently accessed small fields and infrequently accessed large fields, you can compress the large field columns.

Instructions

Supported data types

1. `BLOB` (including `TINYBLOB` , `MEDIUMBLOB` , and `LONGBLOB`)
2. `TEXT` (including `TINYTEXT` , `MEDIUMTEXT` , and `LONGTEXT`)

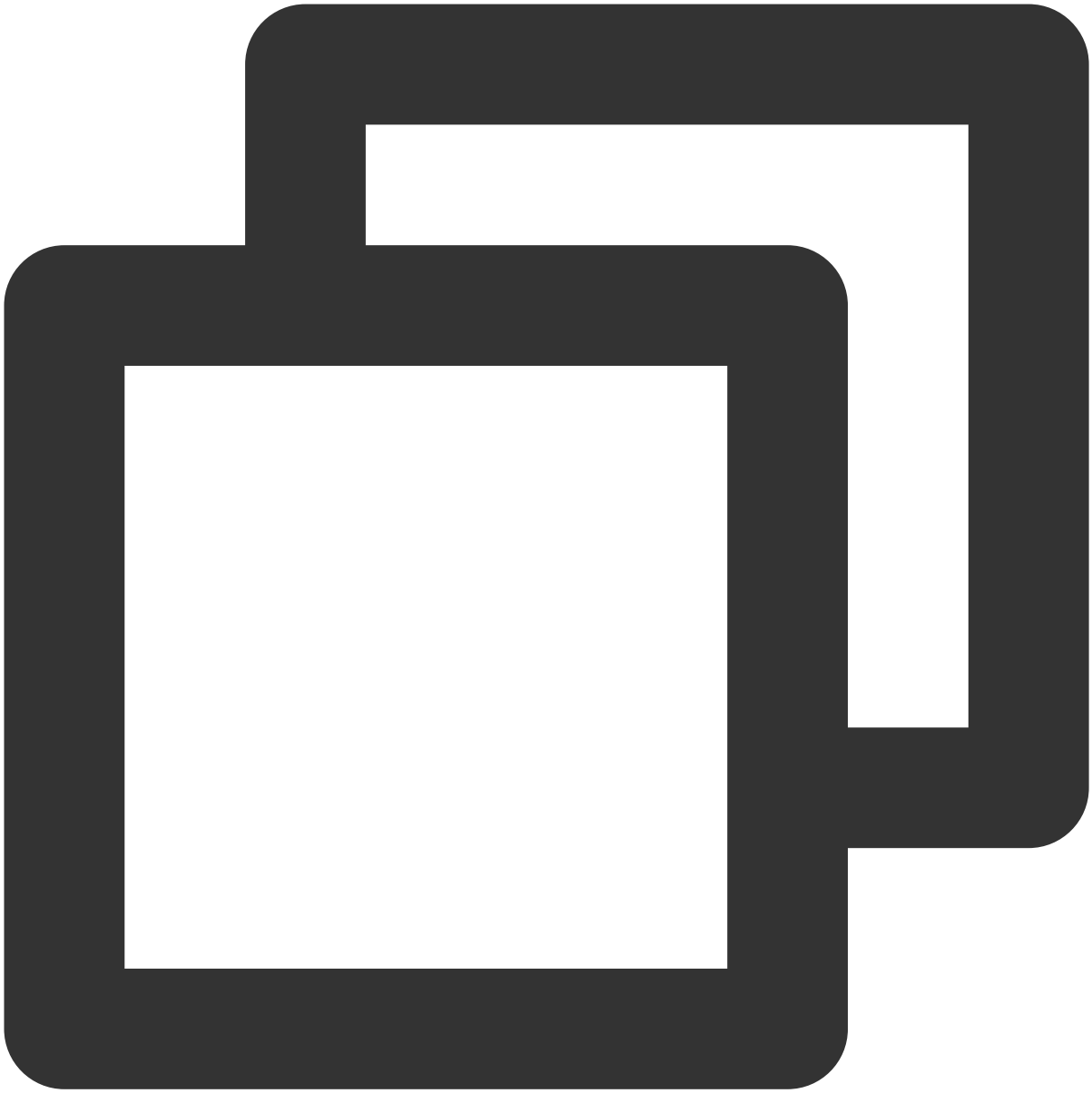
3. `VARCHAR`
4. `VARBINARY`

Note:

Here, the maximum length of `LONGBLOB` and `LONGTEXT` is $2^{32}-2$ bytes, which is one byte less than $2^{32}-1$ supported by native MySQL as described in String Type Storage Requirements.

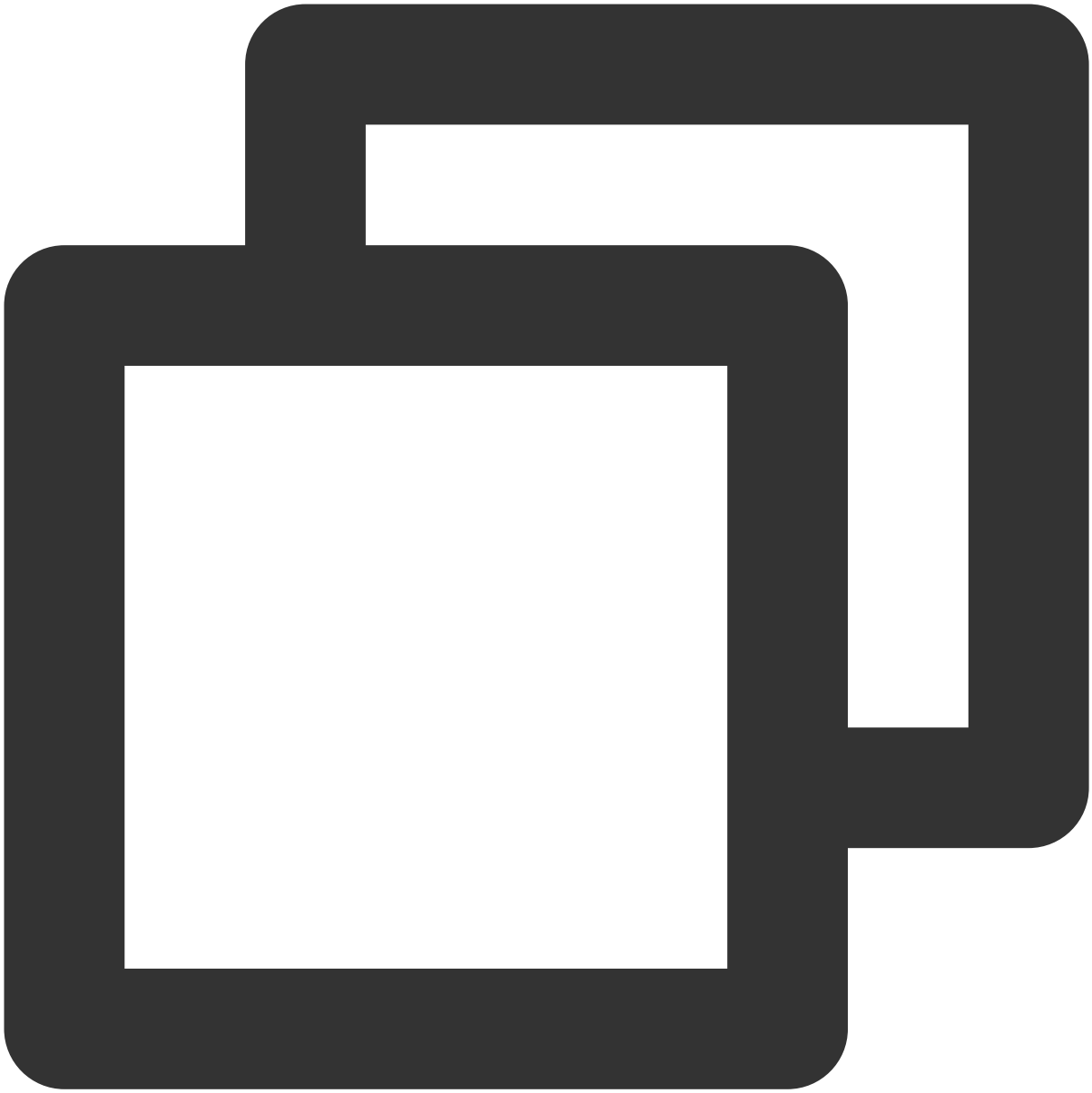
Supported DDL syntax types

Different from the table creation syntax of native MySQL, the definition of `COLUMN_FORMAT` in `column_definition` is changed in TencentDB for MySQL. In addition, column compression is supported only for tables with the InnoDB storage engine.



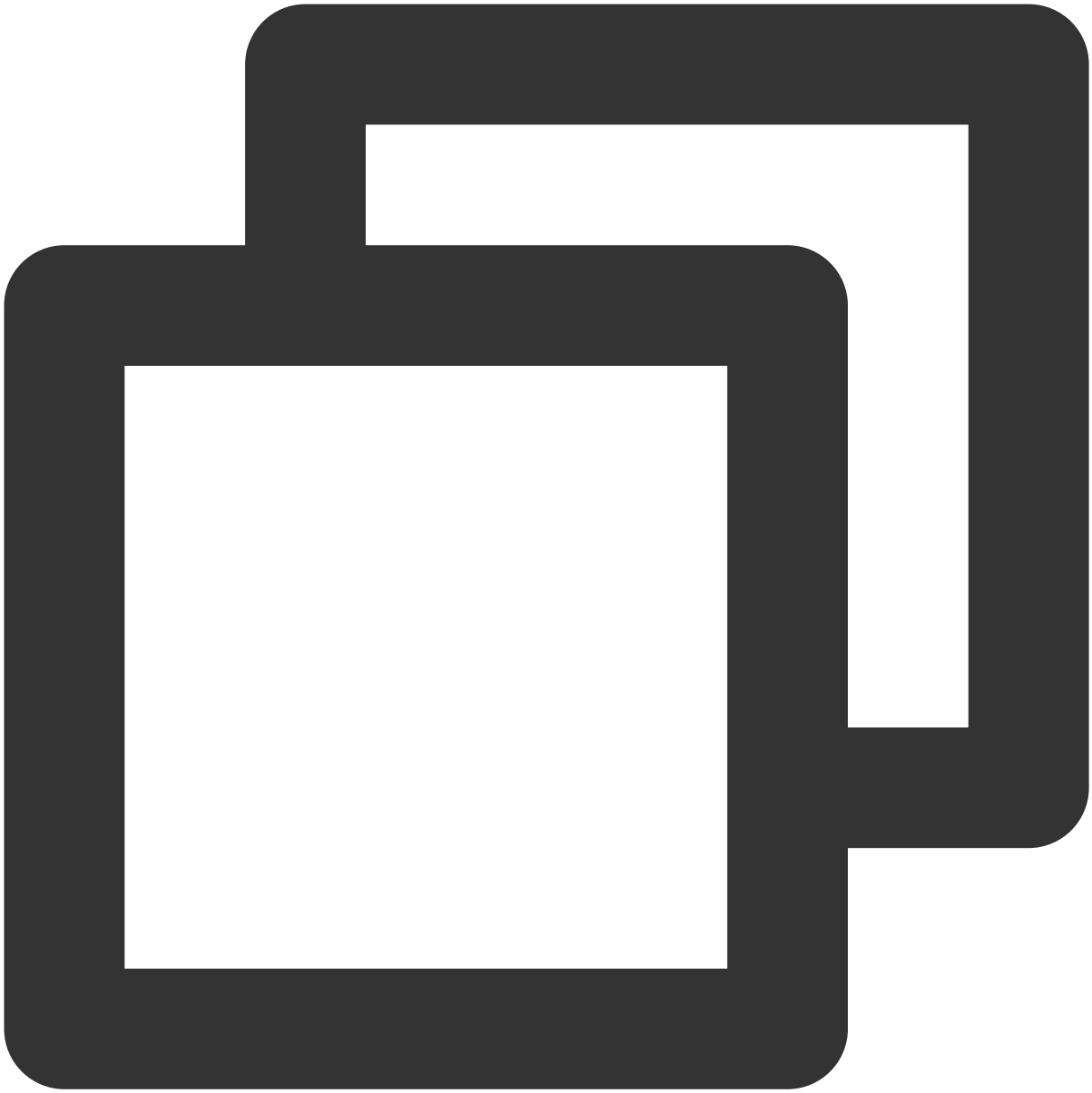
```
column_definition:
  data_type [NOT NULL | NULL] [DEFAULT default_value]
    [AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
    [COMMENT 'string']
    [COLLATE collation_name]
    [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}|COMPRESSED=[zlib]]  # `COMPRESSED`
    [STORAGE {DISK|MEMORY}]
    [reference_definition]
```

Below is a simple example:



```
CREATE TABLE t1(
  id INT PRIMARY KEY,
  b BLOB COMPRESSED
);
```

Here, as the compression algorithm is not specified, the `zlib` algorithm will be selected by default. You can also specify the compression algorithm keyword, but only `zlib` is supported currently.



```
CREATE TABLE t1(
  id INT PRIMARY KEY,
  b BLOB COMPRESSED=zlib
);
```

The following DDL syntaxes are supported:

CREATE TABLE:

DDL	Whether the Compression Attribute is Inherited
`CREATE TABLE t2 LIKE t1;`	Yes

`CREATE TABLE t2 SELECT * FROM t1;`	Yes
`CREATE TABLE t2(a BLOB) SELECT * FROM t1;`	No

ALTER TABLE:

DDL	Description
`ALTER TABLE t1 MODIFY COLUMN a BLOB;`	Alters a compressed column into a non-compressed one
`ALTER TABLE t1 MODIFY COLUMN a BLOB COMPRESSED;`	Alters a non-compressed column into a compressed one

New variable description

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Descripti
innodb_column_compression_zlib_wrap	Yes	bool	TRUE	TRUE/FALSE	If it is set data zlib be gener check wi
innodb_column_compression_zlib_strategy	Yes	Integer	0	[0, 4]	Column c policy. Va Z_DEFA 1: Z_FIL Z_HUFF Z_RLE; 4 Generall `Z_DEF` is the bes data, whi image da
innodb_column_compression_zlib_level	Yes	Integer	6	[0, 9]	Column c Value ran indicates The high smaller tl compres

					longer th duration.
innodb_column_compression_threshold	Yes	Integer	256	[0, 0xffffffff]	Column c threshold range: 1- whose le this thres compres original c unchang compres added.
innodb_column_compression_pct	Yes	Integer	100	[1, 100]	Column c in percer range: 1- compres **size afte compres before c** below thi the origir unchang compres added.

Note:

Currently, you cannot directly modify the values of the above parameters. If needed, submit a ticket for assistance.

New status description

Name	Type	Description
`Innodb_column_compressed`	Integer	Number of column compressions, including compressions for non-compressed data and compressed data.
`Innodb_column_decompressed`	Integer	Number of column decompressions, including decompressions for non-compressed data and compressed data.

New error description

Name	Scope	Description

`Compressed column '%-.192s' can't be used in key specification`	Name of the column specified for compression	The compression attribute cannot be specified for a column with an index.
`Unknown compression method: %s"`	Name of the compression algorithm specified in the DDL statement	An invalid compression algorithm other than `zlib` is specified in the `CREATE TABLE` or `ALTER TABLE` statement.
`Compressed column '%-.192s' can't be used in column format specification`	Name of the column specified for compression	If the `COLUMN_FORMAT` attribute has been specified for a column, other attributes cannot be specified, and `COLUMN_FORMAT` can be used only in NDB.
`Alter table ... discard/import tablespace not support column compression`	\\	The `ALTER TABLE ... DISCARD/IMPORT TABLESPACE` statement cannot be executed for tables with column compression enabled.

Performance

The performance varies by DDL and DML statements:

For DDL statements, sysbench is used for testing:

Column compression compromises much performance of DDL statements with the COPY algorithm, and the performance after compression is 7–8 times lower than before.

The impact of column compression on INPLACE DDL statements is subject to the data volume after compression. If the overall data size is reduced after compression, the DDL performance will be improved; otherwise, it will be compromised.

Column compression almost has no impact on INSTANT DDL statements.

For DML statements, in an 8-column table with the most common compression ratio of 1:1.8 (where the length of the inserted data varies randomly from 1 to 6,000, the inserted characters are random within 0–9 and a–b, a column contains a large volume of varchar data, and the data types of other columns are either char(60) or int), the performance of insertion, deletion, and query of non-compressed columns in this table is improved by below 10%, but the performance of update of non-compressed and compressed columns is reduced by below 10% and 15% respectively. This is because that TencentDB for MySQL first reads the value of a row and then writes the updated value, which triggers one decompression/compression process, while insertion and query only trigger one compression or decompression.

Notes

1. During logic export, CREATE TABLE statements will carry the `COMPRESSED` keyword. Therefore, TencentDB for MySQL supports such statements during import. Below are notes on official MySQL versions:
If the official MySQL version is below 5.7.18, data can be imported directly.
If the official MySQL version is 5.7.18 or above, the `COMPRESSED` keyword must be removed after logic export.
2. When DTS exports data from other cloud service providers or users, incompatibility may occur during binlog sync. In this case, you can skip DDL statements with the `COMPRESSED` keyword.

Flashback Query

Last updated：2024-07-22 11:35:35

Overview

Maloperations may occur in the process of database Ops and severely affect the business. Rollback and cloning are common recovery methods for maloperations, but they are error-prone and time-consuming in case of minor data changes and urgent troubleshooting, and are uncontrollable in recovery time when dealing with major data changes. The TXSQL team has developed and implemented the flashback query feature for the InnoDB engine. It allows you to query the historical data before a maloperation with a simple SQL statement and query the data at a specified time point through specific SQL syntax. This greatly saves the data query and recovery time and enables fast data recovery for better business continuity.

Supported Versions

Kernel version: MySQL 5.7 20220715 and later.
Kernel version: MySQL 8.0 20220331 and later.
For more information on how to view or upgrade the minor kernel version, see Upgrading Kernel Minor Version.

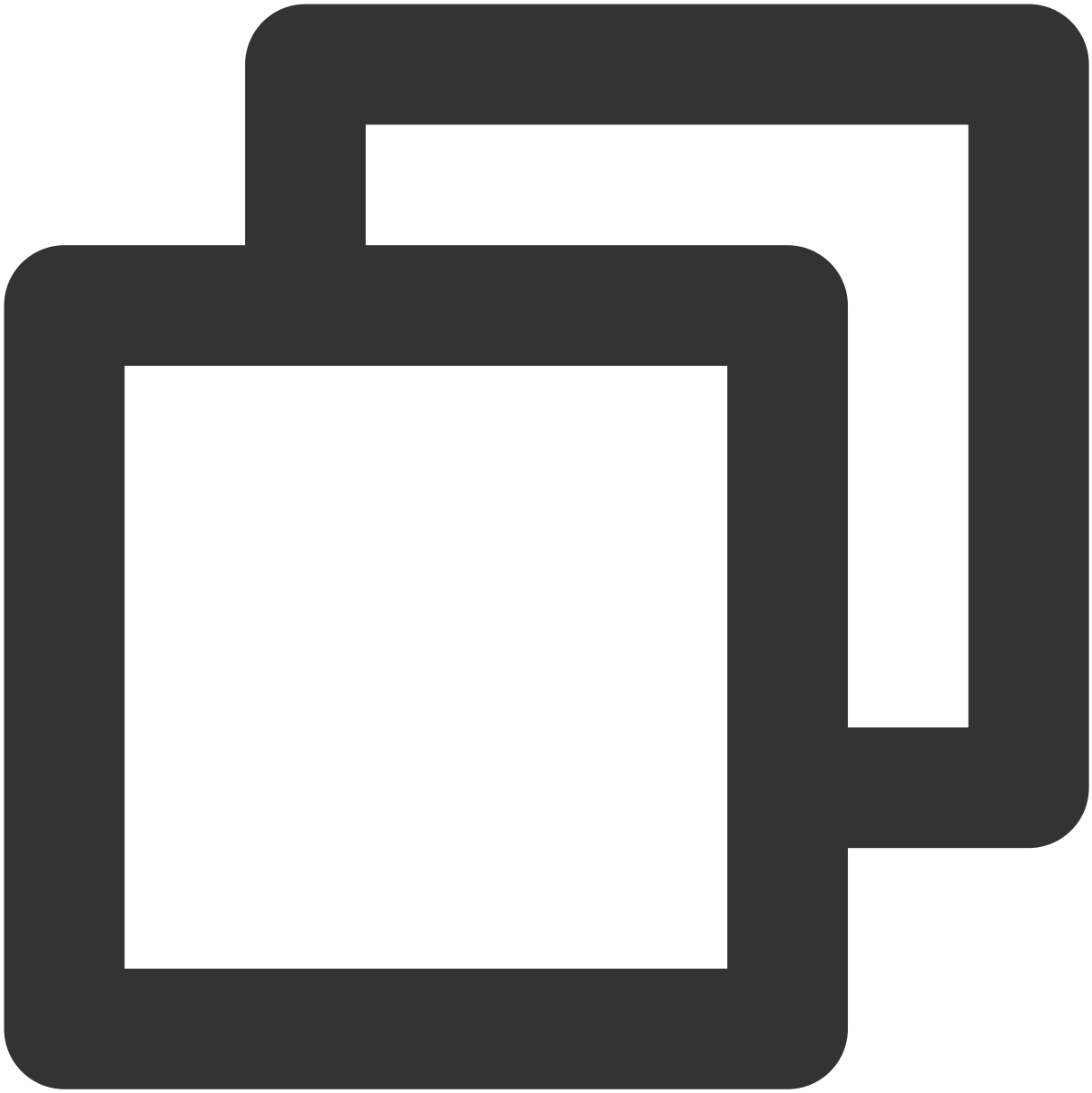
Use Cases

The flashback query feature is used to quickly query the historical data after a maloperation during database Ops.
Notes:
Flashback query is supported only for InnoDB physical tables but not views, other engines, or functions without actual columns such as `last_insert_id()` .
Only second-level flashback query is supported, and the accuracy cannot be fully guaranteed. If there are multiple changes within one second, any of them may be returned.
Flashback query is supported only for primary keys (or GEN_CLUST_INDEX).
Flashback query cannot be used in prepared statements or stored procedures.
Flashback query does not support DDL. If you perform DDL on a table (such as TRUNCATE TABLE, which should be recovered through the recycle bin), the results obtained by flashback query may not be as expected.
In the same statement, if multiple flashback query times are specified for the same table, the earliest time will be selected.

Due to the time difference between the source and replica instances, if you specify the same time for flashback query, the results obtained for the instances may be different.
Enabling the flashback query feature will delay undo log cleanup and increase the memory usage. We recommend you not set `Innodb_backquery_window` to a large value (preferably between 900 and 1,800), especially for instances with frequent business access requests.
If the database instance restarts or crashes, the historical information before the restart or crash cannot be queried. The specified time should be within the supported range (which can be viewed through the status variables `Innodb_backquery_up_time` and `Innodb_backquery_low_time` by running `show status like '%backquery%'`).

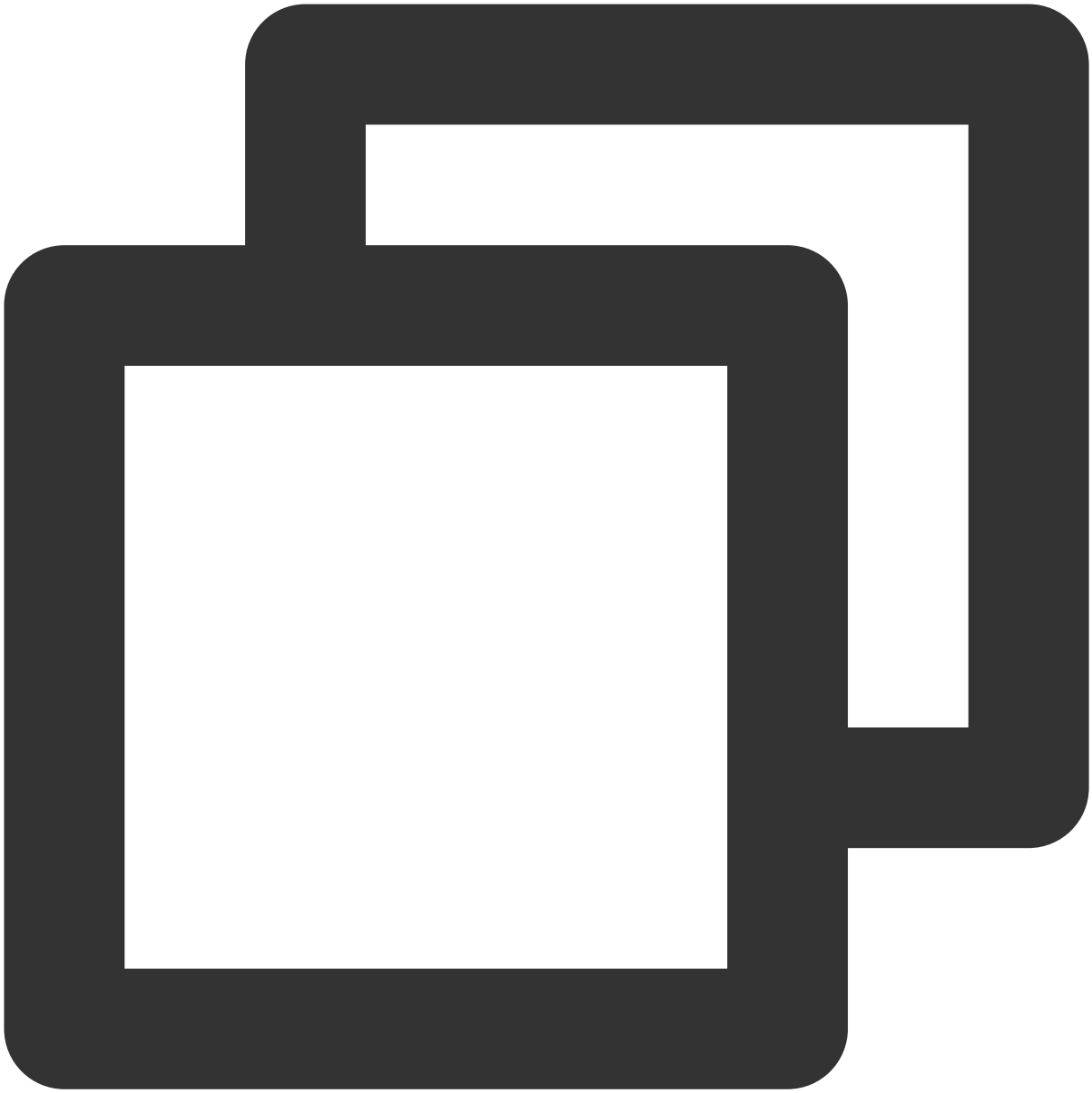
Instructions

Flashback query provides a new AS OF syntax. You can set the `Innodb_backquery_enable` parameter to `ON` to enable the flashback query feature and then query data at the specified time through the following syntax:



```
SELECT ... FROM <table name>
AS OF TIMESTAMP <time>;
```

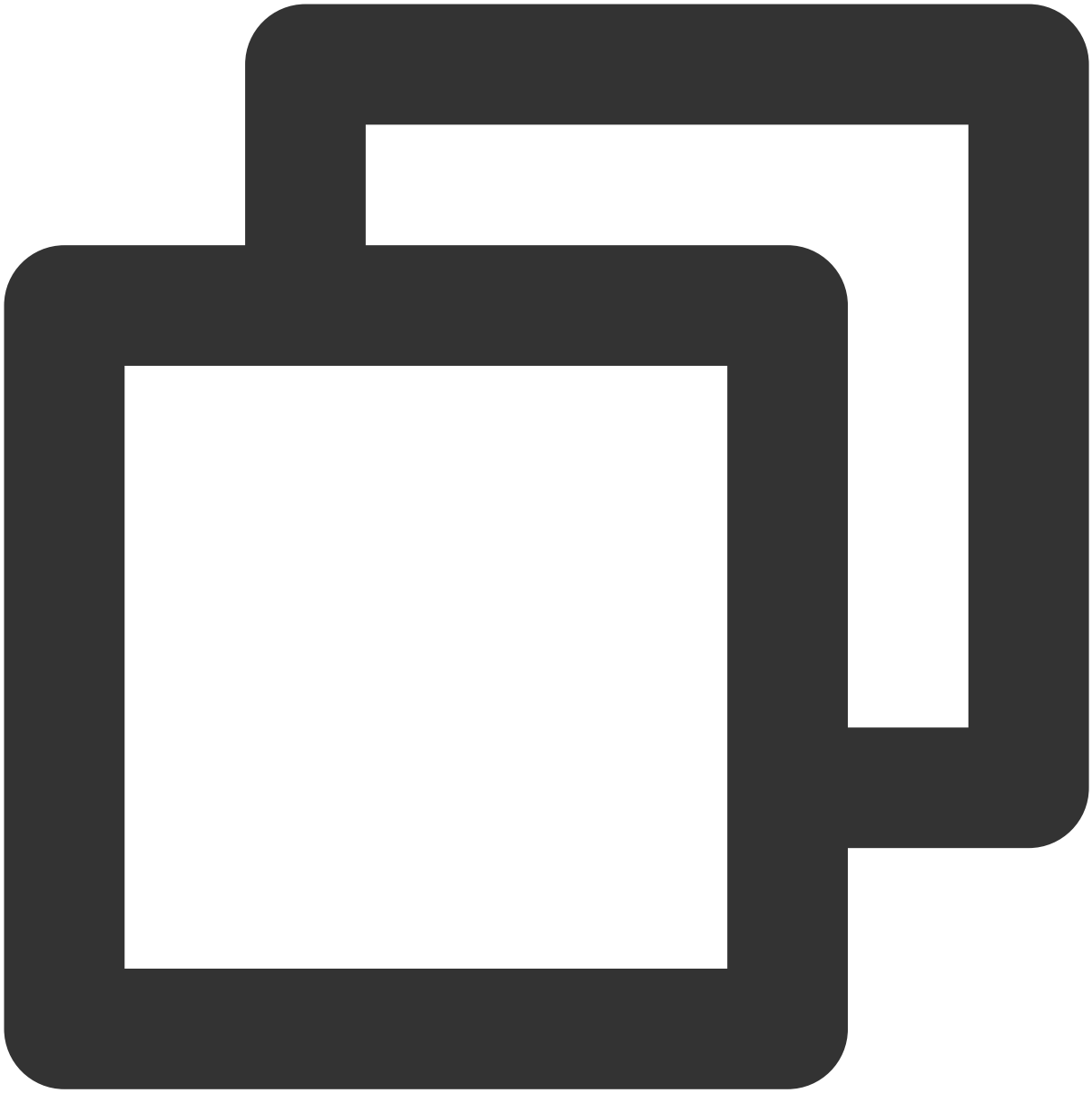
Example of querying data at the specified time



```
MySQL [test]> create table t1(id int,c1 int) engine=innodb;
Query OK, 0 rows affected (0.06 sec)

MySQL [test]> insert into t1 values(1,1),(2,2),(3,3),(4,4);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

MySQL [test]> select now();
+---------------------+
| now()               |
+---------------------+
```

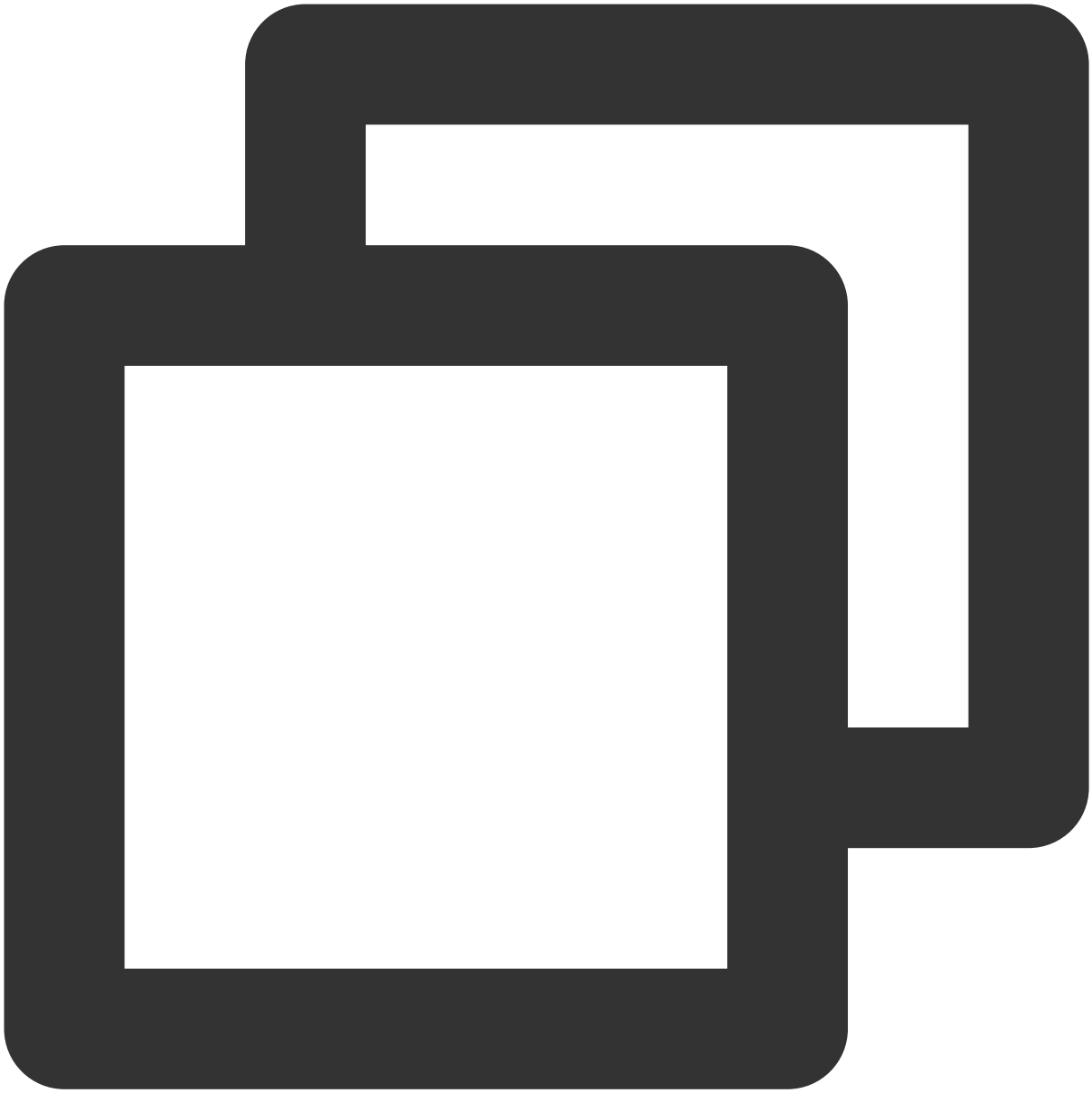
```
| 2022-02-17 16:01:01 |
+---------------------+
1 row in set (0.00 sec)

MySQL [test]> delete from t1 where id=4;
Query OK, 1 row affected (0.00 sec)

MySQL [test]> select * from t1;
+------+------+
| id   | c1   |
+------+------+
|    1 |    1 |
|    2 |    2 |
|    3 |    3 |
+------+------+
3 rows in set (0.00 sec)

MySQL [test]> select * from t1 as of timestamp '2022-02-17 16:01:01';
+------+------+
| id   | c1   |
+------+------+
|    1 |    1 |
|    2 |    2 |
|    3 |    3 |
|    4 |    4 |
+------+------+
4 rows in set (0.00 sec)
```

Example of creating a table from historical data



```
create table t3 select * from t1 as of timestamp '2022-02-17 16:01:01';
```

Example of inserting historical data into a table



```
insert into t4 select * from t1 as of timestamp '2022-02-17 16:01:01';
```

Parameters

The following table lists the configurable parameters of the flashback query feature.

Parameter	Scope	Type	Default Value	Value Range/Valid Values	Restart Required

Innodb_backquery_enable	Global	Boolean	OFF	ON/OFF	No
Innodb_backquery_window	Global	Integer	900	1–86400	No
Innodb_backquery_history_limit	Global	Integer	8000000	1–9223372036854476000	No

Performance Features
Parallel Query
Overview

Last updated：2024-07-22 12:34:17

TencentDB for MySQL supports parallel query. After this feature is enabled, large queries can be automatically identified. The parallel query capability leverages multiple compute cores to greatly shorten the response time of large queries.

Concept

Parallel query uses more computing resources to complete the query workload. The traditional query method is relatively friendly to small amounts of data (hundreds of gigabytes), but as the business grows, the data volume has reached the TB level in many cases, which exceeds the processing capacity of traditional databases. Parallel query is designed to solve this problem. During parallel query, the data is distributed to different threads at the storage layer, multiple threads on a single node process the data in parallel, the result pipelines are aggregated to the main thread, and the main thread performs a simple merge and returns the result. This greatly improves the query efficiency.

Feature background

TencentDB for MySQL goes beyond traditional MySQL databases in terms of computing, storage, disaster recovery, and elastic expansion; however, it still faces the following challenges:
As the internet develops, databases become more capable of storing data, and forms are carrying more and more data. When it comes to big table queries, SQL statements tend to be slow due to existing technical bottlenecks, which adversely affects the business process.
The current market environment sees an increasing number of report statistics and other analytical queries. Although not large in number, they involve a high data volume and are quite sensitive to query time. Gradually, data analysis capability, especially heterogeneous data processing, has become a must-have.
The above challenges are caused by the traditional technical implementation mode in the MySQL ecosystem. In particular, open-source releases support only the single-thread query mode, where only one thread (called user thread) is responsible for the parsing, optimization, and execution of a SQL statement. This mode cannot make full use of the hardware resources of modern multi-core CPUs and large memory devices, leading to a resource waste. Therefore, it is important to streamline analysis and enhance performance by using multi-core services in the query of a large amount of data, which is also the key to query acceleration, cost reduction, and efficiency improvement.

Strengths

Performance enhancement at no extra costs: You can upgrade the kernel capabilities at no extra costs, so that you can get the most out of the instance CPU computation for quicker statement response and higher computing performance.
Support for common statements: You can use most common SQL statements in virtually any business scenarios. This helps you accelerate your business smoothly.
Flexible parameter settings: You have many parameters at hand to control the conditions of enabling or disabling parallel query. This helps you make queries smarter and more adaptable to your business scenarios with no transformation needed.

Supported Statement Scenarios and Restricted Scenarios

Last updated：2024-07-22 12:34:38

This document describes the supported statement scenarios and restricted scenarios of the parallel query.

Supported statement scenarios

TencentDB for MySQL has implemented the parallel query feature for SQL statements with the following characteristics, with more to come.
Single-table scan: Full-table scan, index scan, index range scan, and index REF query in ascending or descending order are supported.
Multi-table join: The nested-loop join (NLJ) algorithm as well as semi join, anti join, and outer join are supported.
Subquery: Parallel query is supported for derived tables.
Data type: Different data types can be queried, such as integer, string, floating point, time, and overflow (with a runtime size limit).
There are no restrictions on common operators and functions.
COUNT, SUM, AVG, MIN, and MAX aggregate functions are supported.
UNION and UNION ALL queries are supported.
Traditional (default), JSON, and tree EXPLAIN formats are supported.

Restricted scenarios

The parallel query feature of TencentDB for MySQL is not supported in the following scenarios.

Restriction	Description
Statement compatibility restriction	Parallel query is not supported for non-query statements, including `INSERT ... SELECT` and `REPLACE ... SELECT`.
	Parallel query is not supported for statements in a stored program.
	Parallel query is not supported for prepared statements.
	Parallel query is not supported for statements in serial isolation-level transactions.
	Parallel query is not supported for locking reads, such as `SELECT FOR

	UPDATE` and `SELECT ... FOR SHARE`.
	Parallel query is not supported for CTEs.
Table/Index compatibility restriction	Parallel query is not supported for system, temp, and non-InnoDB tables.
	Parallel query is not supported for space index.
	Parallel query is not supported for full-text index.
	Parallel query is not supported for partitioned tables.
	Parallel query is not supported for tables in `index_merge` scan mode.
Expression/Field compatibility restriction	Parallel query is not supported for tables containing generated columns or BLOB, TEXT, JSON, BIT, and GEOMETRY fields.
	Parallel query is not supported for aggregate functions of the BIT_AND, BIT_OR, or BIT_XOR type.
	Parallel query is not supported for DISTINCT aggregations, such as SUM(DISTINCT) and COUNT(DISTINCT).
	Parallel query is not supported for GIS functions such as SP_WITHIN_FUNC and ST_DISTANCE.
	Parallel query is not supported for custom functions.
	Parallel query is not supported for JSON functions such as JSON_LENGTH, JSON_TYPE, and JSON_ARRAYAGG.
	Parallel query is not supported for XML functions such as XML_STR.
	Parallel query is not supported for user-lock functions such as IS_FREE_LOCK, IS_USED_LOCK, RELEASE_LOCK, RELEASE_ALL_LOCKS, and GET_LOCK.
	Parallel query is not supported for SLEEP, RANDOM, GROUP_CONCAT, SET_USER_VAR, and WEIGHT_STRING functions.
	Parallel query is not supported for certain statistical functions such as STD, STDDEV, STDDEV_POP, VARIANCE, VAR_POP, and VAR_SAMP.
	Parallel query is not supported for subqueries.
	Parallel query is not supported for window functions.
	Parallel query is not supported for ROLLUP.

Besides the above examples in Supported statement scenarios, you can also check the parallel query execution plan and thread working status to see whether a statement can be queried parallelly. For more information, see Viewing Parallel Query.

Enabling/Disabling Parallel Query

Last updated：2024-07-22 12:34:53

This document describes how to enable or disable the parallel query feature of TencentDB for MySQL via the console or command line.

Prerequisites

Database version: MySQL 8.0 on kernel version 20220831 or later.

Parameters

Note:
The parallel query feature can be enabled for both source and read-only instances, as long as their number of CPU cores is greater than or equal to 4.
You can enable the parallel query feature for the current instance by setting the `txsql_max_parallel_worker_threads` and `txsql_parallel_degree` parameters to a value other than `0` via the console or command line. Parameters and suggested settings are as follows:

Parameter information

Parameter	Variable Type	Scope	Default Value	Value Range
txsql_max_parallel_worker_threads	Integer	Global	{MIN(DBInitCpu,0)}	0–{MAX(DBInitCpu-2,2)}

txsql_parallel_degree	Integer	Global/session	4	0–64

Suggested settings

Parallelism limit: `txsql_parallel_degree` indicates the maximum number of threads for the parallel query of a single statement, i.e., the default parallelism. We recommend that you limit this value to half of the CPU core quantity of the instance. To ensure the stability, the parallel query feature is disabled for small clusters with fewer than four CPU cores, and you cannot adjust parallel query parameters via the console or command line.

During the parallel query of a SQL statement, the parallelism set by `txsql_parallel_degree` will be used by default, which can be adjusted through the HINT statement. For more information, see HINT Statement Control.

`txsql_max_parallel_worker_threads` indicates the number of threads of the instance that can be used for parallel query, and `txsql_max_parallel_worker_threads / txsql_parallel_degree` indicates the maximum number of SQL statements allowed in a parallel query.

`txsql_max_parallel_worker_threads` and `txsql_parallel_degree` control the status of the parallel query feature. When either of them is `0`, the feature is disabled.

TencentDB for MySQL offers various parameters for you to set the execution conditions of parallel query for business adaptation and stability. After conditions are set, the database will check whether each SQL statement can be executed against such conditions like execution cost, number of table rows, and memory usage for the parallel statement execution. Parameters are described as follows:

Parameter	Variable Type	Scope	Default Value
innodb_txsql_parallel_partitions_per_worker	Integer	Global/Session	13
txsql_optimizer_context_max_mem_size	Integer	Global/Session	{MIN(DBInitMemory*52429,8388(
txsql_parallel_cost_threshold	Integer	Global/Session	50000
txsql_parallel_exchange_buffer_size	Integer	Global/Session	1048576
txsql_parallel_table_record_threshold	Integer	Global/Session	5000

Note:

Parallel query parameters take effect immediately after being set, with no instance restart required.

If the scope of a parameter is session, it takes effect only for the statement.

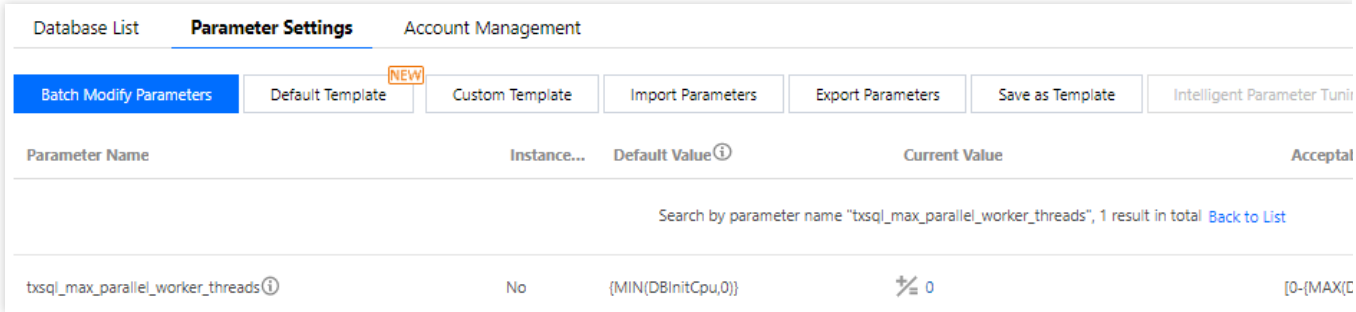
Enabling or disabling parallel query in the console

You can enable or disable the feature by setting parameters on the **Parameter Settings** page in the TencentDB for MySQL console.

Set `txsql_max_parallel_worker_threads` and `txsql_parallel_degree` to a value other than `0` to enable parallel query.

Set `txsql_max_parallel_worker_threads` or `txsql_parallel_degree` to `0` to disable parallel query.

You can also set execution conditions on the **Parameter Settings** page. For detailed directions, see Setting Instance Parameters.



Specifying the parallel execution mode of a SQL statement through the HINT statement

TencentDB for MySQL allows you to specify the parallel execution mode of a SQL statement through the HINT statement. For detailed directions, see HINT Statement Control.

References

Viewing Parallel Query

HINT Statement Control

HINT Statement Control

Last updated：2024-07-22 12:35:02

TencentDB for MySQL allows you to enable or disable the parallel query feature by adjusting parameters. Specifically, you can enable or disable the feature for all SQL statements, set execution conditions, or specify the execution mode of a specific SQL statement through the HINT statement in the console.

Note:

The HINT statement can specify whether to execute a SQL statement and apply session parameters to the statement. In addition, it also supports querying the specified parallel table.

HINT statement usage

Feature	Command Line	Description
Enable parallel query	`SELECT /*+PARALLEL(x)*/ ... FROM ...;`	`x` indicates the parallelism for the SQL statement, which should be greater than `0`.
Disable parallel query	`SELECT /*+PARALLEL(x)*/ ... FROM ...;`	If `x` is set to `0`, it indicates to disable parallel query.
Specify the parallel table	You can specify the table to be included in or excluded from the parallel query execution plan in either of the following ways: Specify the table to be included in the plan through `PARALLEL`. `SELECT /*+PARALLEL(t)*/ ... FROM ...;` Specify the table to be excluded from the plan through `NO_PARALLEL`. `SELECT /*+NO_PARALLEL(t)*/ ... FROM ...;`	`t` is the table name.
Specify both the parallel table and parallelism	`SELECT /*+PARALLEL(t x)*/ * ... FROM ...;`	`x` indicates the parallelism for the SQL statement, which should be greater than `0`. `t` is the table name.
Set the session parameter through	`SELECT /*+SET_VAR(var=n)*/ * ... FROM ...;`	`var` is the parallel query parameter in the

the HINT statement, which takes effect only for the specified SQL statement		`session` scope.

HINT statement use cases

Use case 1: `select /*+PARALLEL()*/ * FROM t1,t2;`
Set the parallelism to the value of `txsql_parallel_degree` (default) for the parallel query. If a statement does not meet the parallel query execution condition, serial query will be used.
Use case 2: `select /*+PARALLEL(4)*/ * FROM t1,t2;`
Set the parallelism of the statement to `4` regardless of the default value, i.e., `txsql_parallel_degree = 4`. If the statement does not meet the parallel query execution condition, serial query will be used.
Use case 3: `select /*+PARALLEL(t1)*/ * FROM t1,t2;`
Include the `t1` table in the parallel query and use the default parallelism. If `t1` is smaller than the value of `txsql_parallel_table_record_threshold`, serial query will be used.
Use case 4: `select /*+PARALLEL(t1 8)*/ * FROM t1,t2;`
Include the `t1` table in the parallel query and set the parallelism to `8`. If `t1` is smaller than the value of `txsql_parallel_table_record_threshold`, serial query will be used.
Use case 5: `select /*+NO_PARALLEL(t1)*/ * FROM t1,t2;`
Exclude the `t1` table from the parallel query. If `t1` is greater than the value of `txsql_parallel_table_record_threshold`, serial query will be used.
Use case 6: `select /*+SET_VAR(txsql_parallel_degree=8)*/ * FROM t1,t2;`
Set the parallelism of the statement to `8` regardless of the default value, i.e., `txsql_parallel_degree = 8`.
Use case 7: `select /*+SET_VAR(txsql_parallel_cost_threshold=1000)*/ * FROM t1,t2`
Set `txsql_parallel_cost_threshold=1000` for the statement. If its execution penalty is greater than `1000`, parallel query can be used.
Use case 8: `select /*+SET_VAR(txsql_optimizer_context_max_mem_size=500000)*/ * FROM t1,t2`
Set `txsql_optimizer_context_max_mem_size=500000` for a statement, which means to adjust the maximum memory size it can apply for in the parallel query plan environment to `500000`.

References

Enabling/Disabling Parallel Query

Viewing Parallel Query

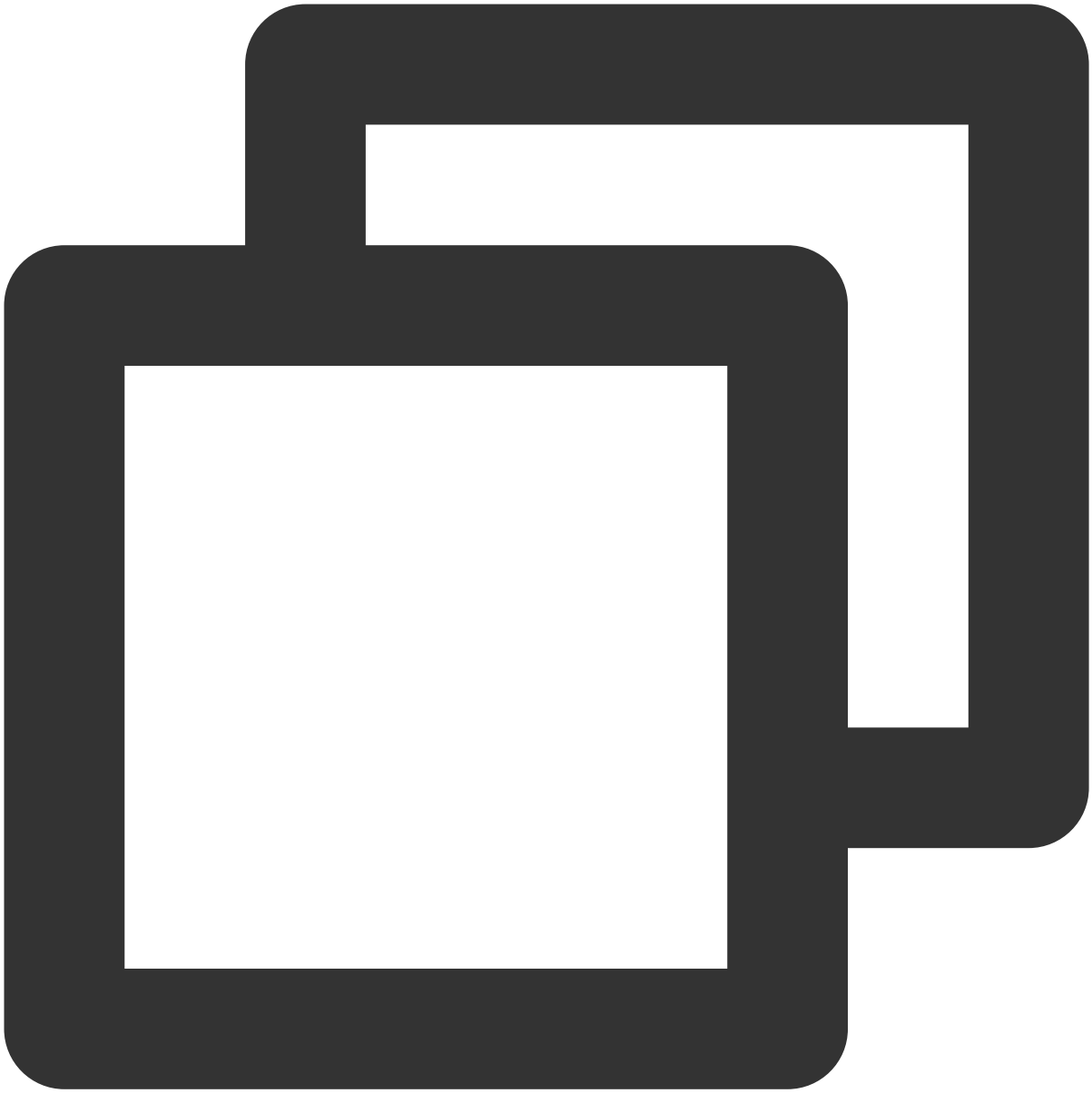
Viewing Parallel Query

Last updated：2024-07-22 12:35:17

TencentDB for MySQL allows you to view the parallel query execution plan and threads in the plan, so that you can clearly know how parallel query takes effect in a database and quickly troubleshoot issues.
This document describes two common methods for viewing parallel queries.

Option 1: Using the EXPLAIN statement

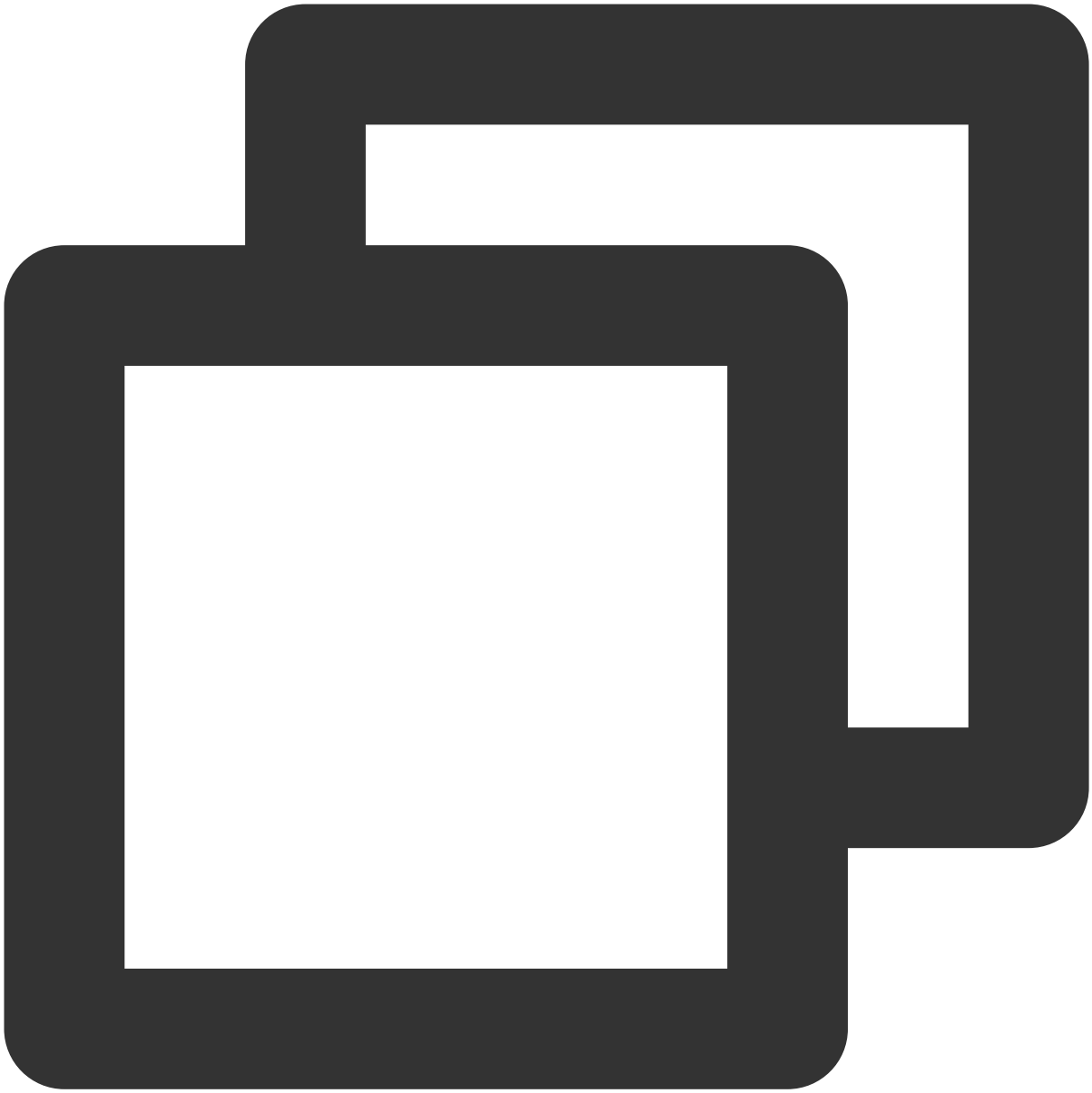
Sample SQL statement:



```
SELECT l_returnflag, l_linestatus, sum(l_quantity) as sum_qty
FROM lineitem
WHERE l_shipdate <= '1998-09-02'
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag, l_linestatus;
```

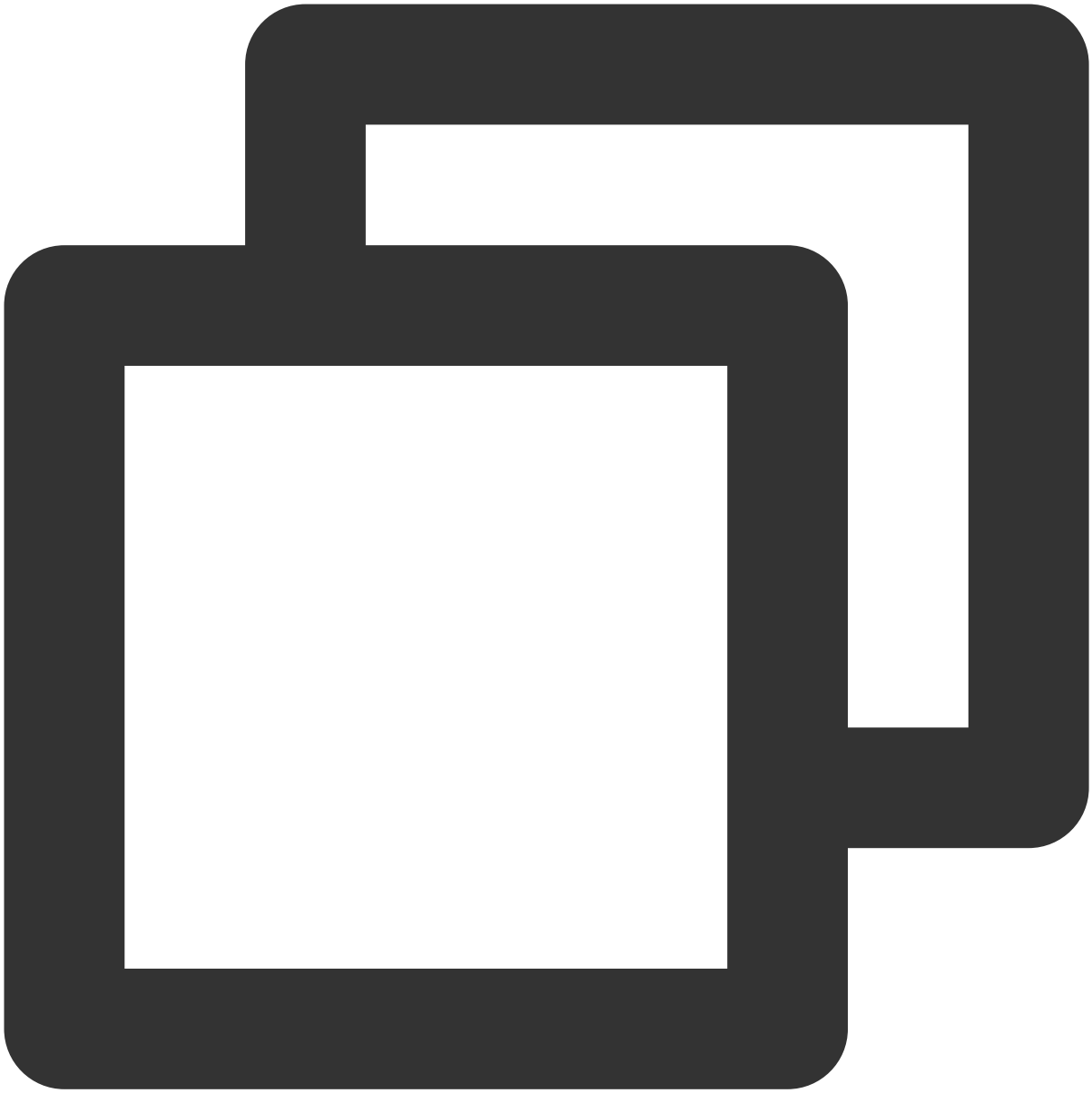
This sample is a simplified version of TPC-H Q1, a typical report operation.

EXPLAIN statement:



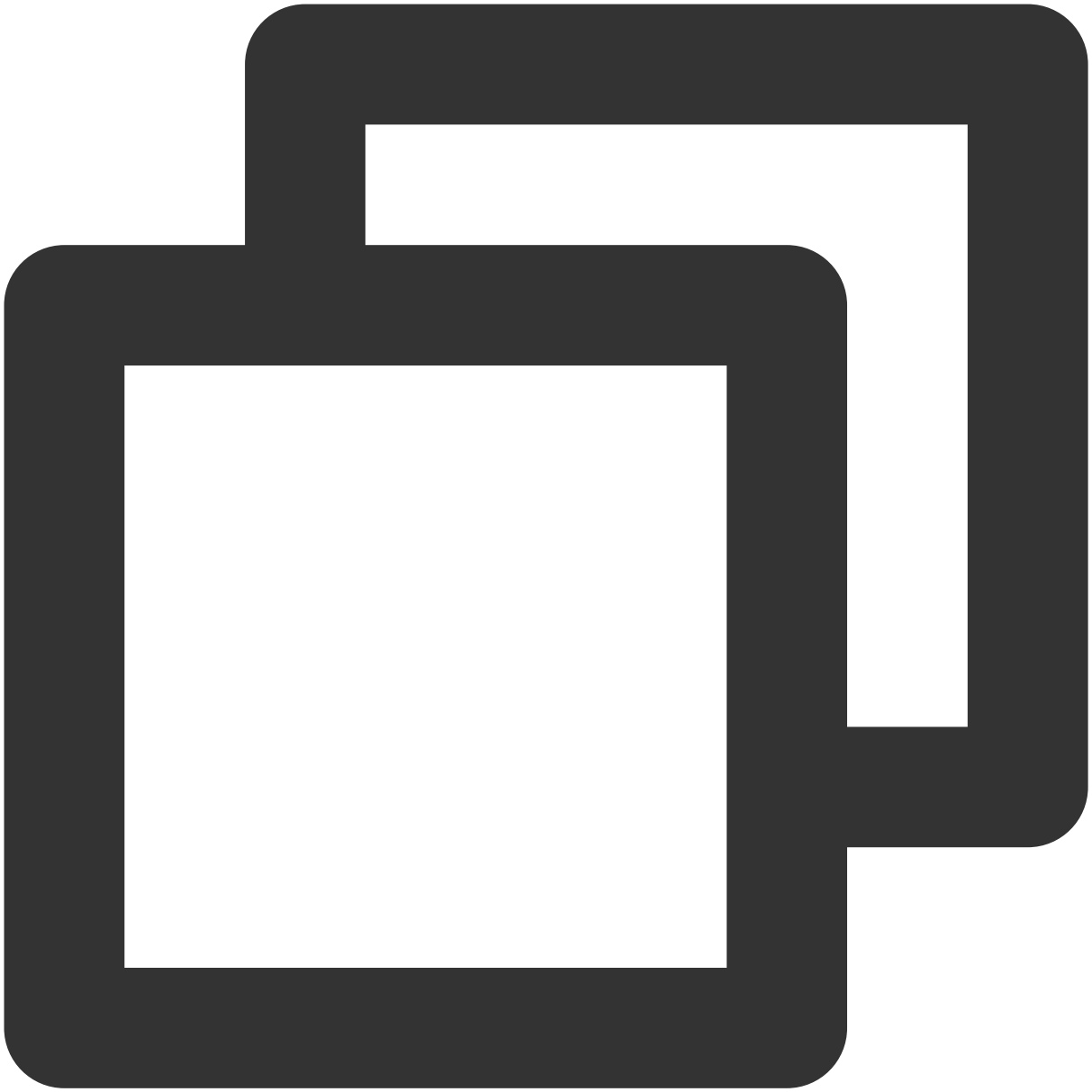
```
EXPLAIN  SELECT l_returnflag, l_linestatus, sum(l_quantity) as sum_qty
FROM lineitem
WHERE l_shipdate <= '1998-09-02'
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag, l_linestatus;
```

Query result:



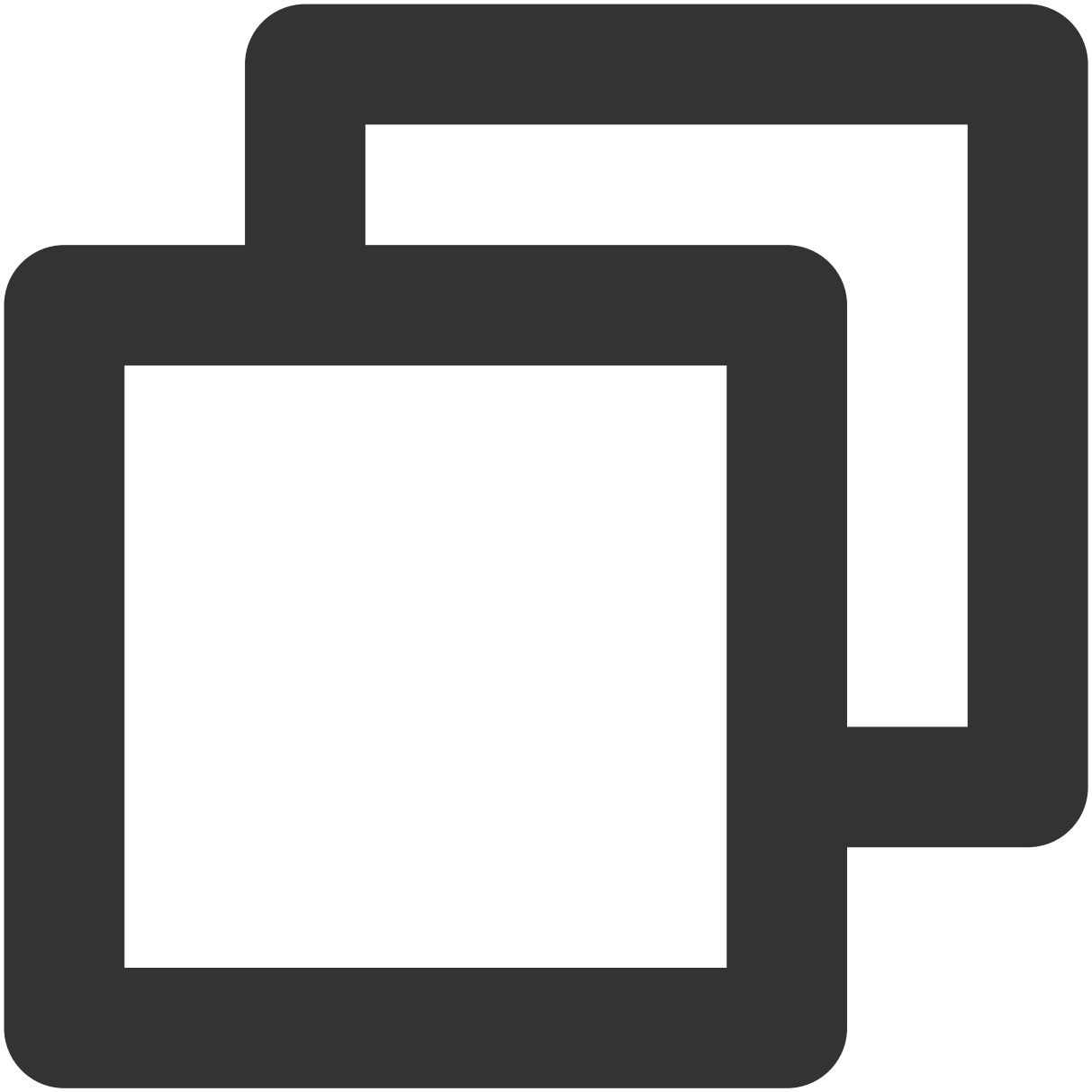
```
MySQL [tpch100g]> explain SELECT l_returnflag, l_linestatus, sum(l_quantity) as sum
+----+-------------+-------------+------------+------+---------------+------+------
| id | select_type | table       | partitions | type | possible_keys | key  | key_l
+----+-------------+-------------+------------+------+---------------+------+------
|  1 | SIMPLE      | lineitem    | NULL       | ALL  | i_l_shipdate  | NULL | NULL
|  1 | SIMPLE      | <sender1>   | NULL       | ALL  | NULL          | NULL | NULL
|  1 | SIMPLE      | <receiver1> | NULL       | ALL  | NULL          | NULL | NULL
+----+-------------+-------------+------------+------+---------------+------+------
3 rows in set, 1 warning (0.00 sec)
```

EXPLAIN format=tree:



```
EXPLAIN format=tree query  SELECT l_returnflag, l_linestatus, sum(l_quantity) as su
FROM lineitem
WHERE l_shipdate <= '1998-09-02'
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag, l_linestatus;
```

Query result:



```
MySQL [tpch100g]> explain format=tree SELECT l_returnflag, l_linestatus, sum(l_quan
*************************** 1. row ***************************
EXPLAIN: -> Sort: lineitem.L_RETURNFLAG, lineitem.L_LINESTATUS
    -> Table scan on <temporary>
        -> Final Aggregate using temporary table
            -> PX Receiver (slice: 0; workers: 1)
                -> PX Sender (slice: 1; workers: 4)
                    -> Table scan on <temporary>
                        -> Aggregate using temporary table
                            -> Filter: (lineitem.L_SHIPDATE <= DATE'1998-09-02')  (
                                -> Parallel table scan on lineitem  (cost=65449341.
```

```
1 row in set (0.00 sec)
```

As can be seen from the above result:

The parallel query plan assigns the statement to four worker threads for computing.

Aggregate operations are split into two segments that are executed by the user and parallel threads respectively.

The parallel scan operator is used for the `lineitem` table.

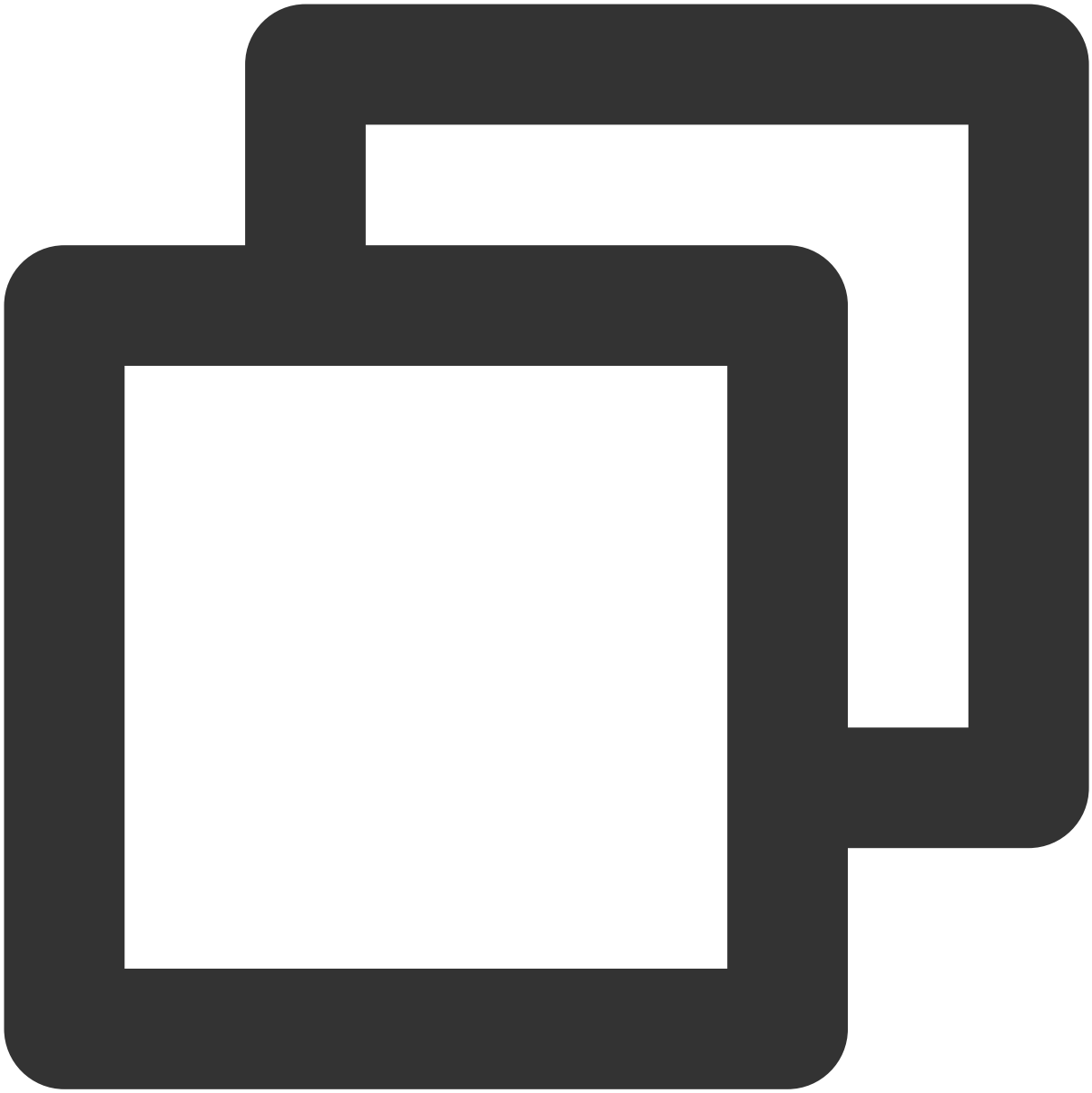
EXPLAIN format=tree query works better than the traditional EXPLAIN.

Option 2: Viewing in the thread list

The result of the `show processlist` command displays which threads are running. You can view not only the total number of current connections but also the connection status to identify abnormal query statements.

Based on the `show processlist` command, TencentDB for MySQL offers the proprietary `show parallel processlist` statement, which displays only the threads related to parallel query and filters out irrelevant threads.

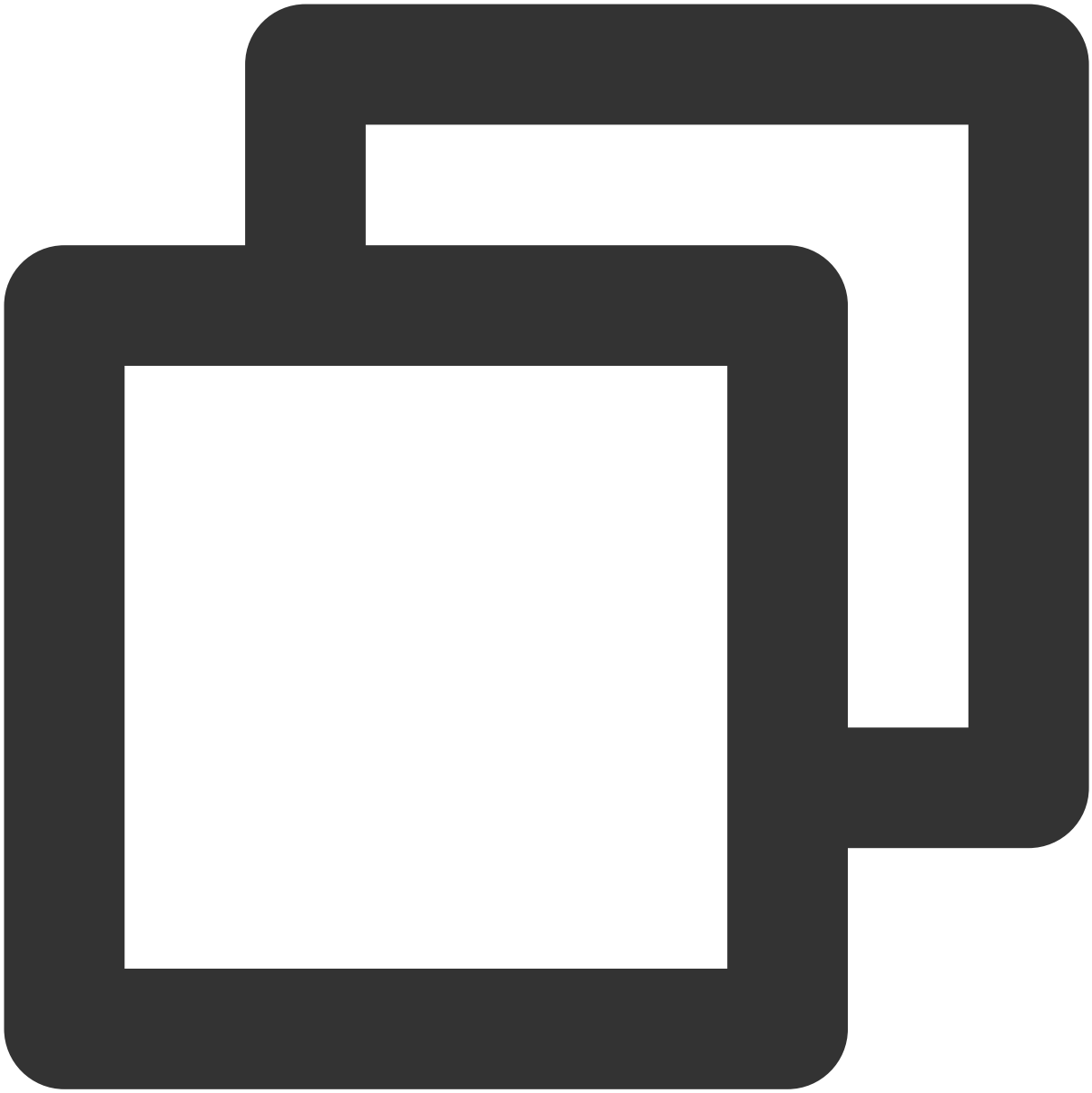
Sample SQL statement:



```
SELECT l_returnflag, l_linestatus, sum(l_quantity) as sum_qty
FROM lineitem
WHERE l_shipdate <= '1998-09-02'
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag, l_linestatus;
```

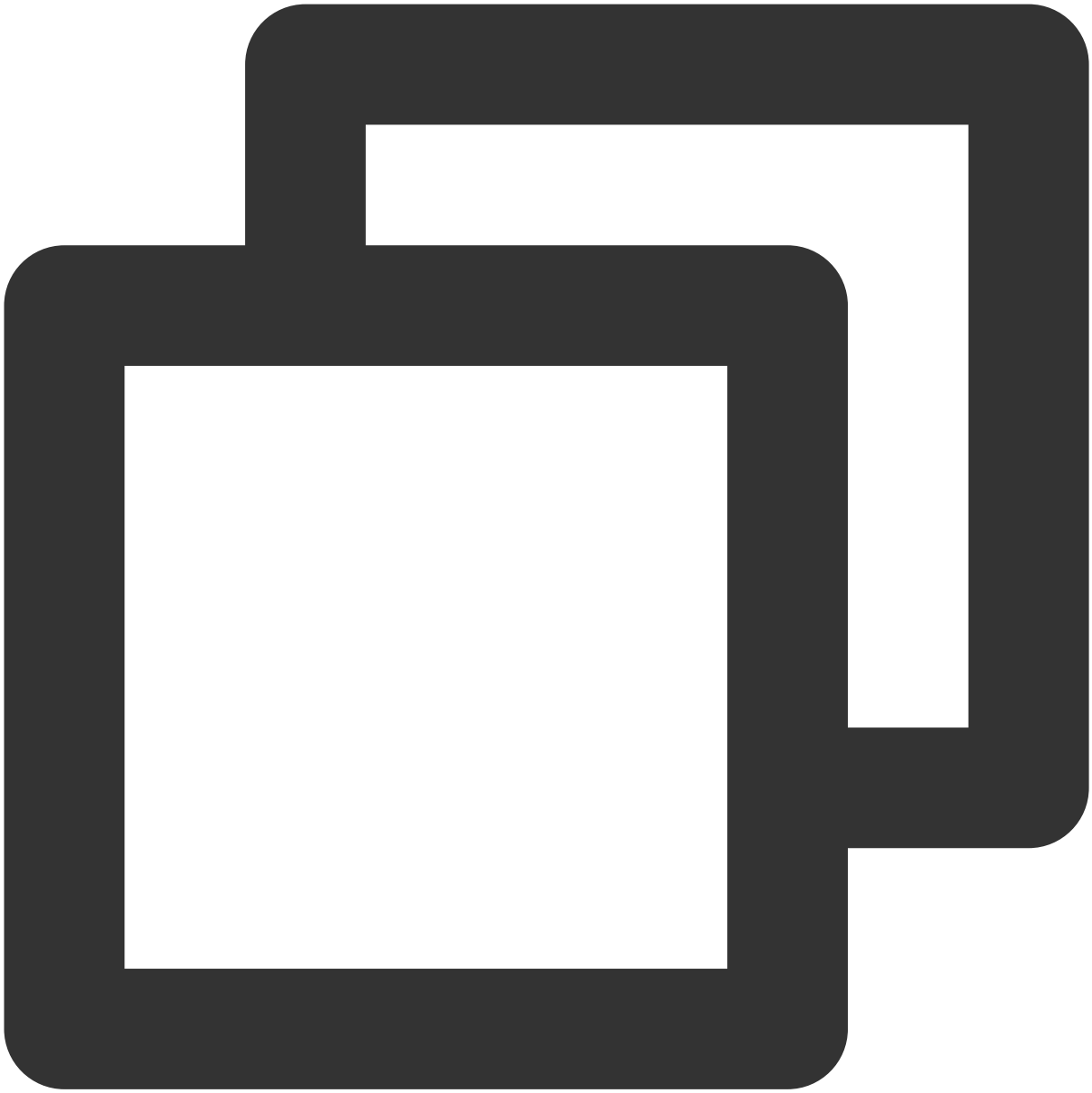
This sample is a simplified version of TPC-H Q1, a typical report operation.

`show processlist` query result:



```
mysql> show processlist;
+--------+-------------+-----------------+-----------+---------+-------+-----------
| Id     | User        | Host            | db        | Command | Time  | State
+--------+-------------+-----------------+-----------+---------+-------+-----------
|      7 | tencentroot | 127.0.0.1:49238 | NULL      | Sleep   |     0 |
|     11 | tencentroot | 127.0.0.1:49262 | NULL      | Sleep   |     0 |
|     13 | tencentroot | 127.0.0.1:49288 | NULL      | Sleep   |     1 |
| 237062 | tencentroot | localhost       | tpch100g  | Query   |    24 | Scheduling
| 237107 | tencentroot | localhost       | NULL      | Query   |     0 | init
+--------+-------------+-----------------+-----------+---------+-------+-----------
6 rows in set (0.00 sec)
```

`show parallel processlist` query result:



```
mysql> show parallel processlist;
+--------+-------------+-----------+----------+---------+------+-------------+-----
| Id     | User        | Host      | db       | Command | Time | State       | Info
+--------+-------------+-----------+----------+---------+------+-------------+-----
| 237062 | tencentroot | localhost | tpch100g | Query   |   18 | Scheduling  | SELE
| 237110 |             |           |          | Task    |   18 | Task runing | conn
| 237111 |             |           |          | Task    |   18 | Task runing | conn
| 237112 |             |           |          | Task    |   18 | Task runing | conn
| 237113 |             |           |          | Task    |   18 | Task runing | conn
```

```
+--------+-------------+-----------+----------+---------+------+-------------+-----
5 rows in set (0.00 sec)
```

As can be seen from the above result:

The parallel query plan assigns queries to four worker threads. There is only one data item in the user thread (ID: 237062). The SQL statement is pushed down to four worker threads. As indicated in `info`, all these four threads are executing `task 1`.

Each thread can be identified and located precisely.

Compared to `show processlist`, `show parallel processlist` can precisely find all running threads of parallel query and will not be affected by other threads.

References

Enabling/Disabling Parallel Query

HINT Statement Control

Large Transaction Replication

Last updated：2024-07-22 12:35:36

Overview

If multi-row large transaction is updated by a single statement in row format, an event will be generated for each row. As a result, a large number of binlogs are created, and APPLY operations in the replica database become slower during replication, causing replication delays.
After analyzing and optimizing the large transaction replication scenarios, the Tencent Cloud kernel team developed the large transaction replication optimization feature. With this feature, large transactions are automatically identified and binlogs are converted from row format into statement format, thus reducing the quantity of binlogs and increasing the replication performance.

Supported Versions

Kernel version: MySQL 5.6 20210630 and above.
Kernel version: MySQL 5.7 20200630 and above.
Kernel version: MySQL 8.0 20200830 and above.

Use Cases

This feature accelerates large transaction replay for tables without a primary key in row format. It can be enabled if you are sure that the delay is caused by slow replay due to the lack of primary key.
This feature aims to solve slow replication when there are large transactions in row format.

Performance Data

The replication time is reduced by 85% for UPDATE operations and about 30% for INSERT operations.

Instructions

The large transaction replication optimization feature judges whether a transaction is large based on the historical execution statistics of the SQL statement. If a transaction is identified as large and optimizable, its isolation level will

be automatically upgraded to repeatable read (RR), and the binlogs will be stored in statement format to reduce the time of executing the large transaction in the replica database. Here:

`cdb_optimize_large_trans_binlog` is the switch of this feature.

`cdb_sql_statistics` is the switch of SQL statement execution statistics collection.

`cdb_optimize_large_trans_binlog_last_affected_rows_threshold` and `cdb_optimize_large_trans_binlog_aver_affected_rows_threshold` are the thresholds for judging a large transaction.

`cdb_sql_statistics_info_threshold` is the number of legacy data entries retained in the memory.

To better monitor the transaction execution, the `CDB_SQL_STATISTICS` table is added in the `information_schema` database for you to query the statistics of the current transaction.

New parameters

Parameter	Status	Type	Default Value	Descripti
cdb_optimize_large_trans_binlog	true	bool	false	Switch of transacti optimizat
cdb_optimize_large_trans_binlog_last_affected_rows_threshold	true	ulonglong	10000	Large tra replicatic condition number c time
cdb_optimize_large_trans_binlog_aver_affected_rows_threshold	true	ulonglong	10000	Large tra replicatic condition average rows
cdb_sql_statistics	true	bool	false	Switch of executior collectior
cdb_sql_statistics_info_threshold	true	ulonglong	10000	Maximun statemer of `CDB_SQ`

Note:

Currently, you cannot directly modify the values of the above parameters. If needed, submit a ticket for assistance.

Newly added `information_schema.CDB_SQL_STATISTICS` table

Name	Type	Description
DIGEST_MD5	MYSQL_TYPE_STRING	MD5 value calculated from the digest of the SQL statement.
DIGEST_TEXT	MYSQL_TYPE_STRING	SQL statement digest text format.
SQL_COMMAND	MYSQL_TYPE_STRING	SQL command type.
FIRST_UPDATE_TIMESTAMP	MYSQL_TYPE_DATETIME	The time when the statistics information of the SQL statement is generated for the first time.
LAST_UPDATE_TIMESTAMP	MYSQL_TYPE_DATETIME	The time when the statistics information of the SQL statement is last updated.
LAST_ACCESS_TIMESTAMP	MYSQL_TYPE_DATETIME	The time when the statistics information of the SQL statement is last accessed.
EXECUTE_COUNT	MYSQL_TYPE_LONGLONG	The number of executions of this SQL statement.
TOTAL_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	Total number of affected rows.
AVER_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	Average number of affected rows.
LAST_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	The number of rows affected last time.
STMT_BINLOG_FORMAT_IF_POSSIBLE	MYSQL_TYPE_STRING	Whether binlogs for this SQL statement can be stored in statement format. Valid values: TRUE, FALSE.

Execution Plan Cache for Optimizing UK/PK Queries

Last updated：2024-07-22 12:35:46

Overview

In MySQL, SQL statement execution is divided into four stages: parsing, preparation, optimization, and execution. The execution plan cache feature is only available for prepared statements. After the feature is enabled, the first three stages will be skipped when executing a prepared statement, greatly boosting query performance.
In MySQL 8.0 20210830, the execution plan cache takes effect only for queries using unique keys (UKs) or primary keys (PKs). We will cover more types of queries in later versions.

Supported Versions

Kernel version: MySQL 8.0 20210830 and later.

Use Cases

This feature is mainly used to improve the query performance when executing short prepared statements with UKs or PKs on TencentDB instances. However, the extent to which performance may improved depends on your business.

Performance Impact

For UK and PK SQL statements, the delay is reduced by 20%-30% and the throughput is improved by 20%-30% after the execution plan cache is enabled (according to the sysbench test using the point_select.lua script).
Memory usage will increase after the execution plan cache is enabled.

Instructions

You can use the `cdb_plan_cache` parameter to enable or disable the execution plan cache and the `cdb_plan_cache_stats` parameter to query information about cache hits. Note that only accounts with the tencentroot permission can use the two parameters.

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description
cdb_plan_cache	Yes	bool	false	true/false	Whether to enable the feature. Only accounts with the feature permission can use the parameter.

Note:

Currently, you cannot directly modify the values of the above parameter. If needed, submit a ticket for assistance.

You can run the `show cdb_plan_cache` command to query information about execution plan cache hits. The command will return the following fields:

Field	Description
sql	A SQL statement with the question mark (?) which represents that the execution plan of this statement has been cached.
mode	SQL cache mode. Currently, only the `prepare` mode is supported.
hit	Number of hits for this session.

After `cdb_plan_cache_stats` is enabled, cache hit information will be recorded, affecting database performance.

SQL Execution Status

You can run `show profile` to show the status at each stage of SQL statement execution. But when the execution plan cache is hit, the status of `optimizing`, `statistics`, and `preparing` will be omitted.

fdatasync()

Last updated：2024-07-22 12:35:56

Overview

The `fsync()` system call flushes redo logs to disk, including metadata and data. But metadata contains unimportant information such as the last modified time. You can enable the `fdatasync()` system call to skip metadata when flushing redo logs in order to reduce costs.

Supported Versions

Kernel version: MySQL 5.7 20201230 and above.
Kernel version: MySQL 8.0 20201230 and above.

Use Cases

This feature is suitable for use cases with heavy write pressure.

Performance Data

TPS is improved by about 10%, according to the sysbench test in a high-concurrency continuous write scenario using the oltp_write_only.lua script.

Instructions

Use the `innodb_flush_redo_using_fdatasync` parameter to enable or disable `fdatasync()`. Valid values: `true` (enable), `false` (disable). Default value: `false`. If `fdatasync()` is enabled, metadata of redo logs won't be flushed to disk in real time.

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description
innodb_flush_redo_using_fdatasync	Yes	bool	false	true/false	Whether to call `fdatasync()`

					to flush redo logs

Auto-Increment Column Persistence

Last updated：2024-07-22 12:36:06

Overview

The auto-increment column persistence feature can persist an auto-increment column into a page to avoid duplicate auto-increment values.

Supported Versions

Kernel version: MySQL 5.7 20190830 and above.

Use Cases

This feature is suitable for scenarios where you don't want duplicate auto-increment values, such as legacy data archive.

Instructions

This feature is enabled in the kernel by default.

Buffer Pool Initialization

Last updated：2024-07-22 12:36:19

Overview

This feature speeds up the initialization of the buffer pool, reducing the startup time of the database instance.

Supported versions

Kernel version: MySQL 5.6 20200915 and above.
Kernel version: MySQL 5.7 20200630 and above.

Use Cases

This feature is used to speed up the startup of the database instance.

Performance Test Data

Performance test data collected from eight instances:

buffer_pool_size	Buffer Pool Initialization Time (Before Optimization)	Buffer Pool Initialization Time (After Optimization)	Increase (%)
50 GB	2.55 s	0.13 s	1,962%
200 GB	10.28 s	0.52 s	1,977%
500 GB	25.72 s	1.32 s	1,948%

Instructions

This feature is enabled in the kernel by default.

FAST DDL

Last updated：2024-07-22 12:36:36

Overview

This feature speeds up the creation of secondary index. After the feature is enabled, secondary indexes can be concurrently sorted in a temp table using multiple threads. The feature also optimizes the operation of locking the flush list when loading bulk data, effectively reducing the time consumed by CREATE INDEX and the impact on concurrent DML operations.

Supported Versions

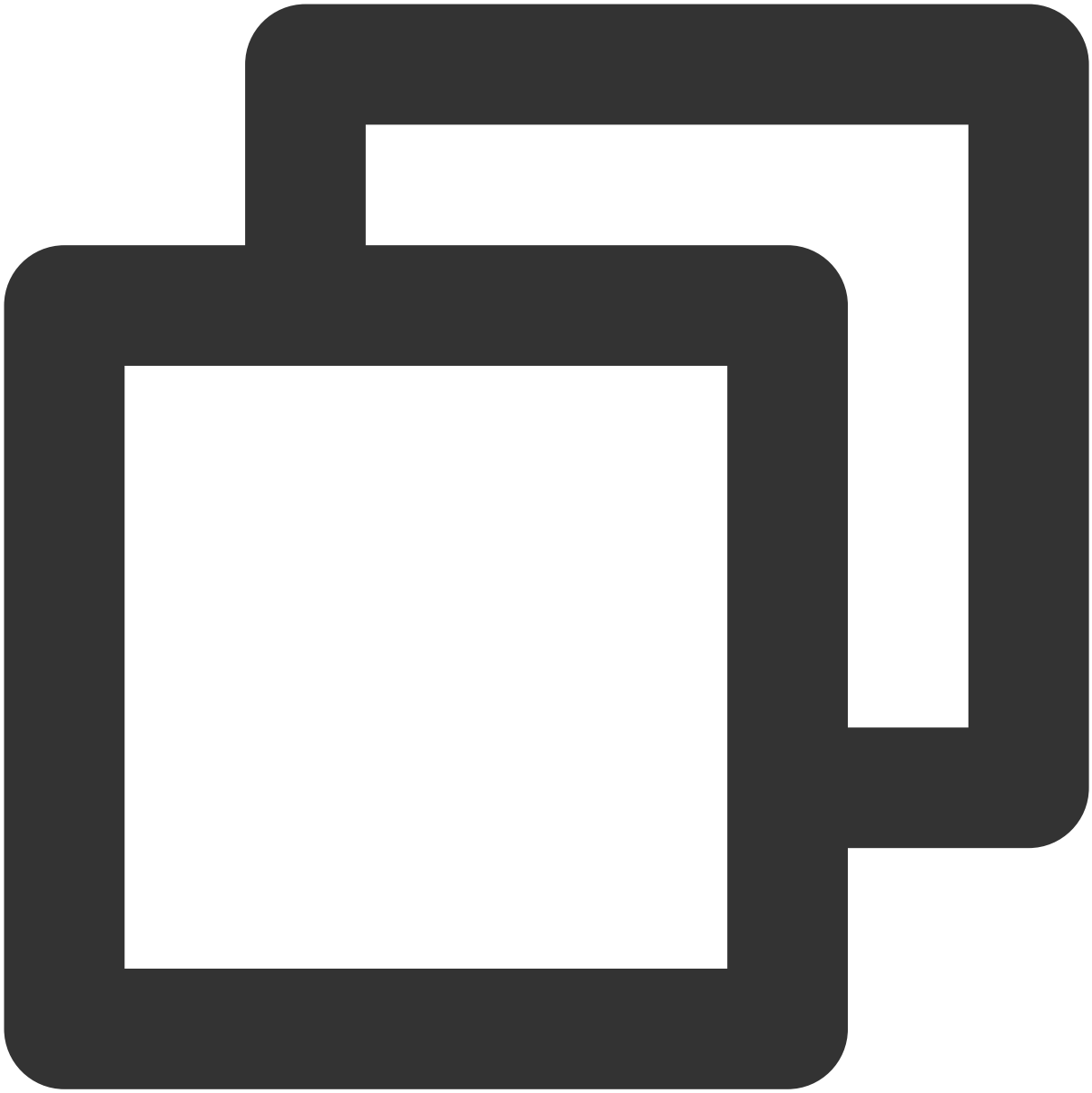
Kernel version: MySQL 8.0 20210330 and above.
Kernel version: MySQL 5.7 20210331 and above.

Use Cases

You need to perform DDL operations frequently on your database and may encounter the following DDL-related problems:
Why does database performance fluctuate when I add indexes, which even affects business writes and reads?
Why does it sometimes take more than 10 minutes to execute a DDL operation on a table less than one GB in size?
Why does database performance fluctuate when I exit a connection where a temp table is used?
To solve the above common problems, the TXSQL kernel team has optimized the operation of locking the flush list when loading bulk data, based on in-depth analysis and testing in multiple scenarios. The optimization effectively reduces the time consumed by CREATE INDEX, the impact on concurrent DML operations, and the impact caused by DDL operations.

Performance Data

Use sysbench to test database performance when importing two billion rows of data (about 453 GB) before and after FAST DDL is enabled.



```
mysql> set global innodb_fast_ddl=ON;
Query OK, 0 rows affected (0.00 sec)
```

When the feature is disabled, the operation takes 4,395 seconds; when the feature is enabled, the operation takes 2,455 seconds.

Instructions

Use the `innodb_fast_ddl` parameter to enable or disable this feature.

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description
innodb_fast_ddl	Yes	bool	OFF	{ON,OFF}	Enable or disable FAST DDL

Note:

Currently, you cannot directly modify the values of the above parameter. If needed, submit a ticket for assistance.

Invisible Index

Last updated：2024-07-22 12:36:47

Overview

Many users require the invisibility of an index to assess if it can be deleted. By making an index as invisible, you can test the impact of its deletion on query performance before deleting it. If the index is being used by any program or database user, an error will occur or be reported. This feature is now available to MySQL 5.7 and later versions, not just limited to MySQL 8.0.

Supported Versions

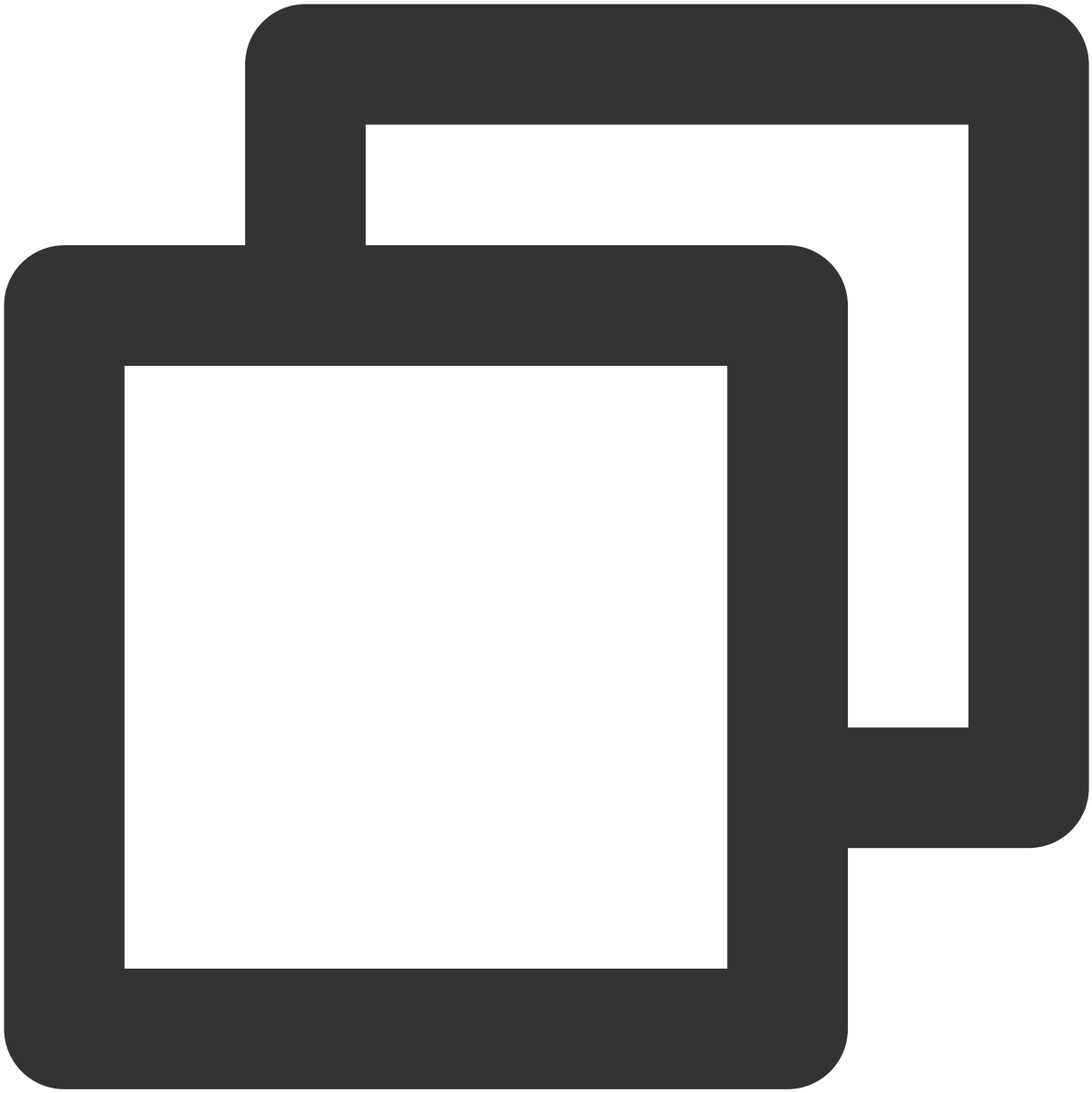
Kernel version: MySQL 5.7 20180918 and above.

Use Cases

Before deleting an index, you can make it invisible to see if it is still in use. If not, it can be securely deleted.

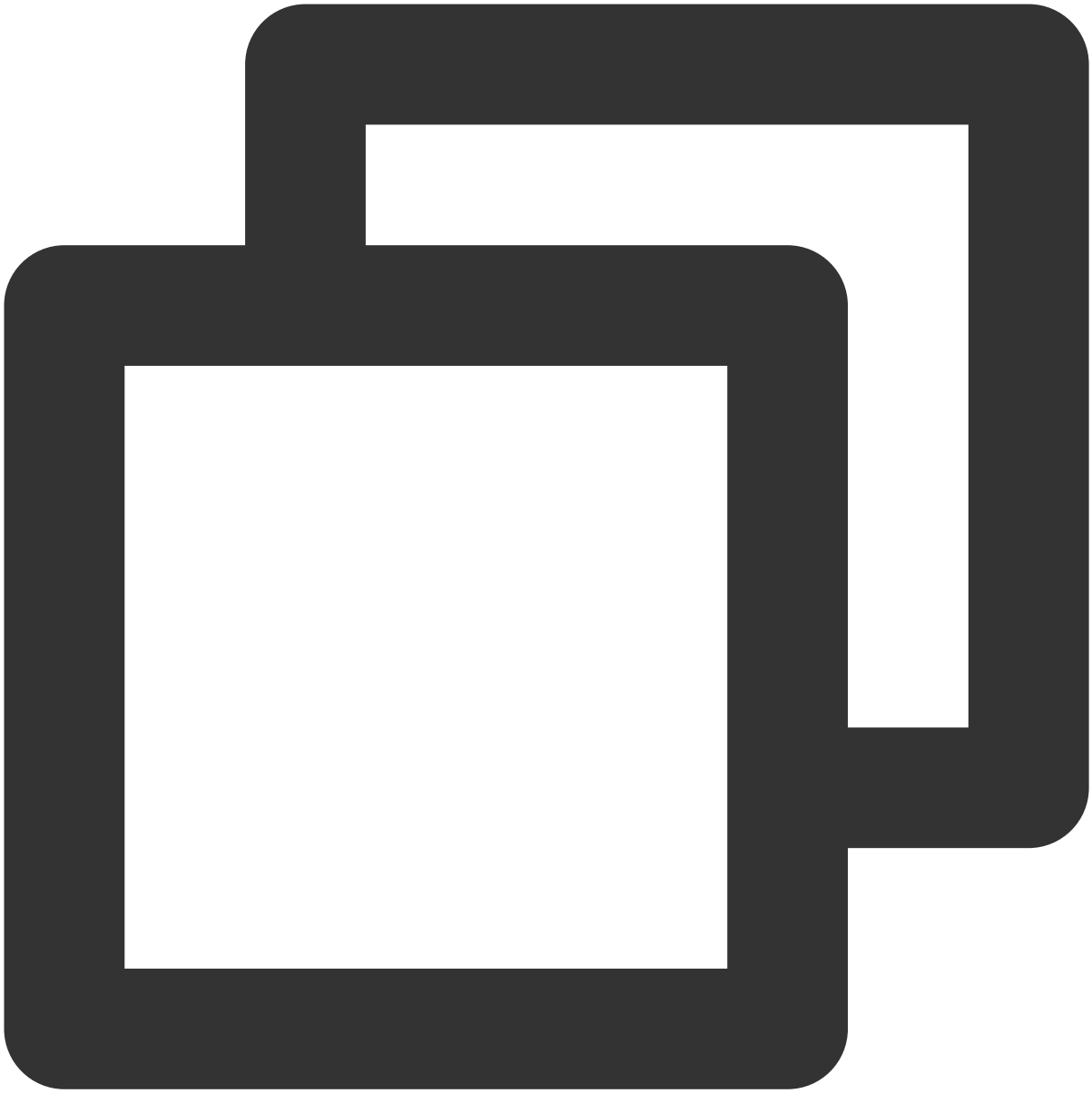
Instructions

Run the following statements to create an invisible index or make an index invisible:



```
CREATE TABLE t1 (
  i INT,
  j INT,
  k INT,
  INDEX i_idx (i) INVISIBLE
) ENGINE = InnoDB;
CREATE INDEX j_idx ON t1 (j) INVISIBLE;
ALTER TABLE t1 ADD INDEX k_idx (k) INVISIBLE;
```

Run the following statements to make an index visible:



```
ALTER TABLE t1 ALTER INDEX i_idx INVISIBLE;
ALTER TABLE t1 ALTER INDEX i_idx VISIBLE
```

CATS Transaction Scheduling Algorithm

Last updated：2024-07-22 12:37:00

Overview

TXSQL supports the Contention-Aware Transaction Scheduling (CATS) algorithm. This new algorithm automatically detects lock contention between transactions and schedules them based on their scheduling weights.
MySQL supports another transaction scheduling algorithm, aka First In First Out (FIFO), which was introduced earlier than CATS. When multiple transactions are waiting for the same lock, CATS prioritizes them by assigning a scheduling weight which is computed based on the number of transactions that a transaction blocks. The transaction with a higher scheduling weight will be executed sooner. Thus, transaction throughput is improved.

Supported Versions

Kernel version: MySQL 5.7 20190230 and above.
Kernel version: MySQL 8.0 20200630 and above.

Use Cases

This feature is suitable for use cases under high concurrency and heavy lock contention.

Performance Data

TPS is improved by more than 50% under high concurrency and heavy lock contention.
Test method: use the oltp_read_write.lua script of sysbench (pareto random type enabled) to test TPS on eight tables (10 MB data) at the REPEATABLE READ transaction isolation level
Test environment: TencentDB instance with 32 cores and 128 GB memory

Thread Count	FCFS (FIFO)	CATS	Performance Improvement
128	11,999	12,005	0%
256	6,609	10,137	53%
512	3,453	9,365	171%
1,024	2,196	7,015	219%

Instructions

In MySQL 5.7, you can use the global parameter `innodb_trx_schedule_algorithm` to specify the transaction scheduling algorithm. The default value is `auto` .

Valid values:

auto: Automatically adjust the transaction scheduling algorithm based on current system status. If the number of threads waiting for a lock exceeds 32, adopt CATS; otherwise, adopt First Come First Serve (FCFS), an algorithm similar to FIFO.

fcfs: Adopt the FCFS algorithm.

cats: Adopt the Contention-Aware Transaction Scheduling algorithm.

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description
innodb_trx_schedule_algorithm	Yes	string	auto	[auto,fcfs,cats]	Specify the transaction scheduling algorithm

Note:

Currently, you cannot directly modify the values of the above parameter. If needed, submit a ticket for assistance.

In MySQL 8.0, `auto` is the only valid value.

Computation Pushdown

Last updated：2024-07-22 12:37:15

Overview

This feature pushes LIMIT/OFFSET and SUM operations down to the storage engine InnoDB when querying single tables, effectively reducing query latency.
When LIMIT/OFFSET is executed using secondary indexes, this feature can avoid using the clustered index values as pointers to find the full table rows, effectively cutting the cost of scanning table data.
This feature pushes SUM operations down to InnoDB. In other words, instead of sending rows to the MySQL server, InnoDB calculates data itself and returns the final result to the MySQL server.

Supported Versions

LIMIT/OFFSET optimization applies to kernel version MySQL 5.7 20180530.
SUM optimization applies to kernel version MySQL 5.7 20180918.

Use Cases

This feature is mainly used to optimize single-table queries with LIMIT/OFFSET or SUM clauses, such as `Select *from tbl Limit 10”, “Select* from tbl Limit 10,2` , and `Select sum(c1) from tbl` .
This feature cannot optimize the following queries:
Queries with DISTINCT, GROUP BY, or HAVING clauses
Nested subqueries
Queries with FULLTEXT indexes
Queries with ORDER BY clauses, where the optimizer fails to use indexes to implement ORDER BY
Queries with multi-range read (MRR)
Queries with SQL_CALC_FOUND_ROWS.

Performance Data

Import one million rows of data and test query performance in sysbench:
The execution time of `select * from sbtest1 limit 1000000,1;` decreases from 6.3 to 2.8 seconds.
The execution time of `select sum(k) from sbtest1;` decreases from 5.4 to 1.5 seconds.

Instructions

During the execution of an SQL statement, the optimizer automatically modifies the query execution plan to implement computation pushdown according to the following parameters.

Parameters are as follows:

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description
cdb_enable_offset_pushdown	Yes	bool	ON	{ON,OFF}	Enable or disable LIMIT/OFFSET pushdown. It is enabled by default.
cdb_enable_sumagg_pushdown	Yes	bool	OFF	{ON,OFF}	Enable or disable SUM pushdown. It is disabled by default.

Note:

Currently, you cannot directly modify the values of the above parameters. If needed, submit a ticket for assistance.

Security Features
Transparent Data Encryption

Last updated：2024-07-22 12:38:10

Overview

TXSQL inherits the transparent data encryption mechanism of MySQL and provides another implementation of the keyring plugin: keyring KMS, which integrates keyring with Tencent Cloud's enterprise-grade Key Management Service (KMS) service.
KMS is a data and key security protection service of Tencent Cloud, where all involved processes use high-security communication protocols to guarantee high service security. In addition, it provides distributed cluster management and hot backup capabilities to ensure high service reliability and availability.
KMS uses a two-layer key system, which involves two types of keys: customer master key (CMK) and data encryption key (DEK). A CMK is used to encrypt small packet data (up to 4 KB in size), such as DEK, password, certificate, and configuration file. A DEK is used to encrypt massive amounts of business data in symmetric encryption method during storage or communication and is encrypted and protected in asymmetric encryption method with a CMK. In this way, data can be encrypted both in the memory and files.

Supported Versions

Kernel version: MySQL 5.7 20171130 and later.
Kernel version: MySQL 8.0 20200630 and later.

Use Cases

Transparent data encryption means that data encryption/decryption operations are imperceptible to users. It supports real-time I/O encryption/decryption of data files; that is, data will be encrypted before being written to the disk and decrypted when being read from the disk into the memory. This helps meet the compliance requirements for static data encryption.

Instructions

For more information, see Enabling Transparent Data Encryption.

Audit

Last updated：2024-07-22 12:38:22

Overview

Tencent Cloud offers database auditing for TencentDB MySQL instances. With this feature, database access and SQL statement execution information, including the start time of statement execution, the number of scanned rows, lock wait time, CPU time, client IP, username, and SQL statement, will be audited, assisting enterprises in risk management and data protection.

Use Cases

This feature is suitable for the use cases where risky database behaviors (such as SQL injection and abnormal operation) need to be recorded and alarmed.

Performance Impact

There are two audit modes: sync and async. Sync audit synchronously records all audit logs with an average impact of less than 6% on instance performance. But async audit has almost no impact (less than 3%, to be precise), which is industry-leading.

Instructions

For more information on how to enable TencentDB for MySQL audit, see Enabling TencentDB for MySQL Audit.

Stability Features
Second-Level Column Addition

Last updated：2024-07-22 12:42:01

Overview

The quick column addition feature allows you to quickly add columns to a big table by only modifying the data dictionary, which eliminates the need of data replication during column adding and greatly reduces the column adding time for big tables and the impact on the system.

Supported Versions

Kernel version: MySQL 5.7 20190830 and later.
Kernel version: MySQL 8.0 20200630 and later.

Use Cases

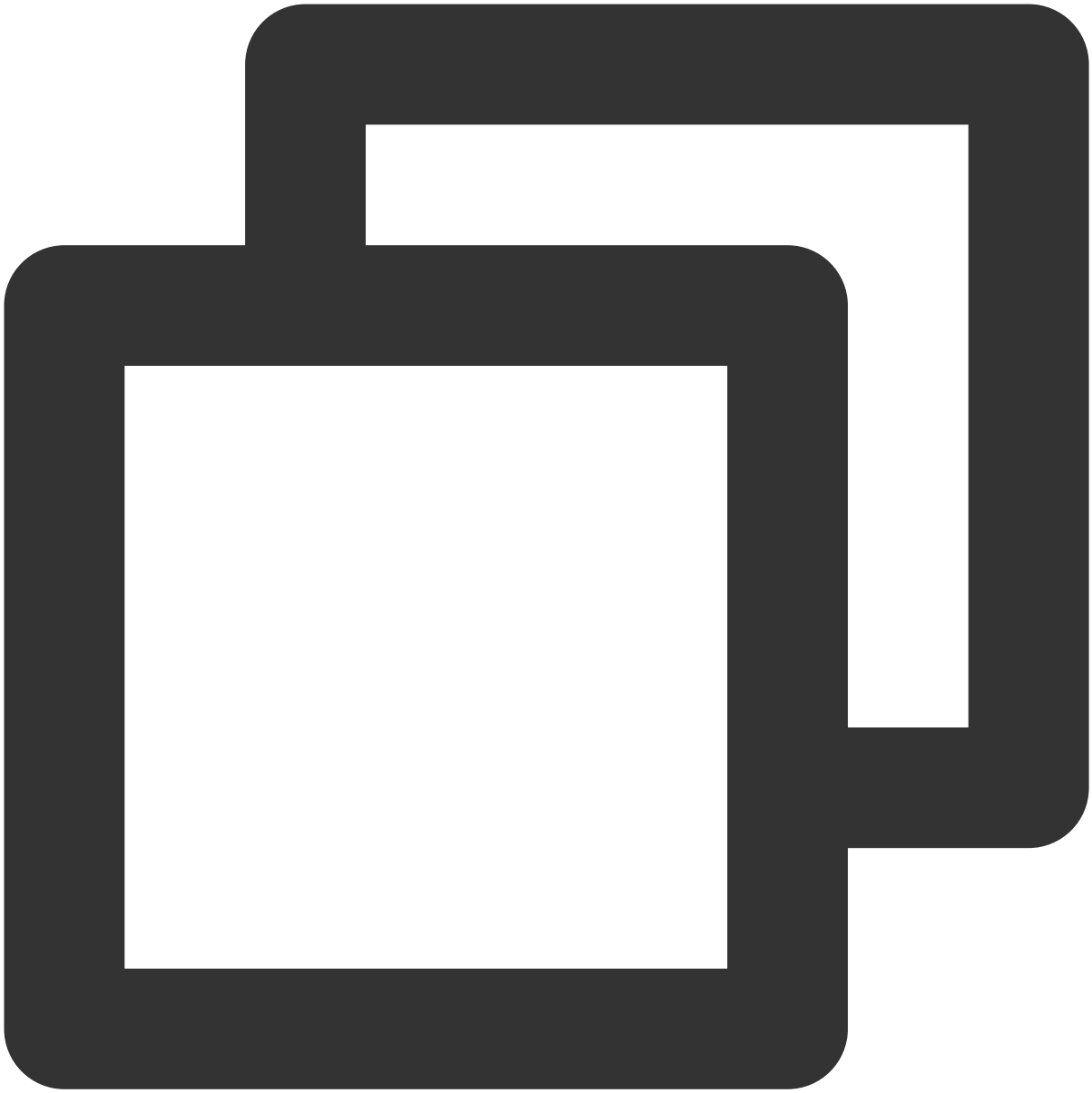
This feature is suitable for adding columns to a table with a high volume of data.

Performance Data

In tests with a table of 5 GB data, the time for adding a column is reduced from 40 seconds to below 1 second.

Instructions

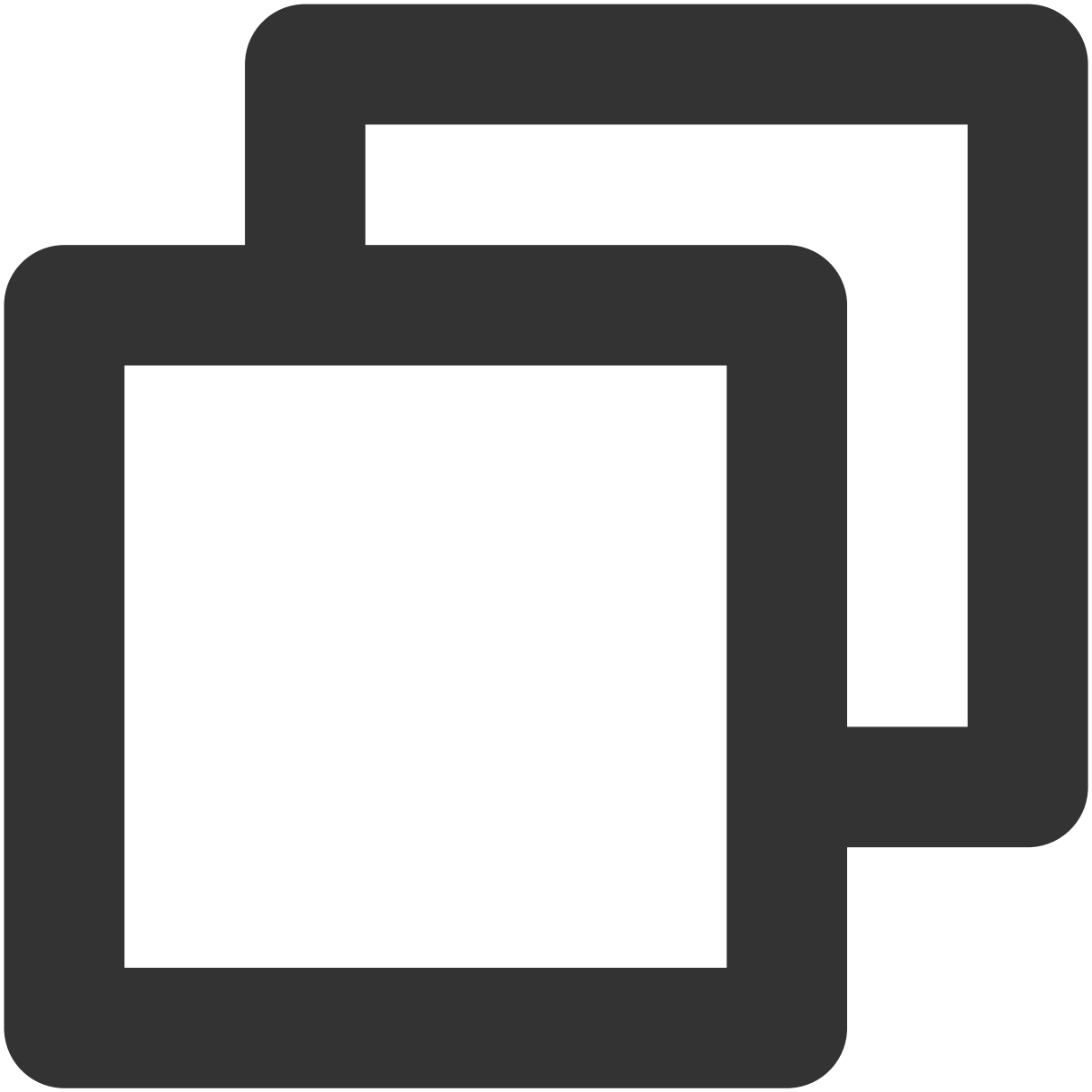
INSTANT ADD COLUMN syntax
Add the `algorithm=instant` clause to `ALTER TABLE` to add a column as follows:



```
ALTER TABLE t1 ADD COLUMN c INT, ADD COLUMN d INT DEFAULT 1000, ALGORITHM=INSTANT;
```

The `innodb_alter_table_default_algorithm` parameter is added, which can be set to `inplace` or `instant`.

This parameter is `inplace` by default and can be configured to adjust the default algorithm of `ALTER TABLE` as follows:



```
SET @@global.innodb_alter_table_default_algorithm=instant;
```

If no algorithm is specified, the default algorithm configured by this parameter will be used for `ALTER TABLE` operations.

Restrictions on INSTANT ADD COLUMN

A statement can contain only column addition operations.

A new column will be added to the end, and column order cannot be changed.

INSTANT ADD COLUMN is not supported in tables with the COMPRESSED row format.

INSTANT ADD COLUMN is not supported in tables with a full-text index.

INSTANT ADD COLUMN is not supported for temp tables.

Second-Level Column Modification

Last updated：2024-06-18 14:39:37

Description of the Feature

Second-level Column Modification operations are only recorded in the Data Dictionary Table to log Column Modification Information, avoiding the data copying that was previously necessary for column modification operations. This significantly reduces the time required for Large Table Column Modifications, minimizing the impact on Application Systems and resource consumption.

Supported Versions

Kernel version MySQL 8.0 20230630 and above.

Applicable Scenario

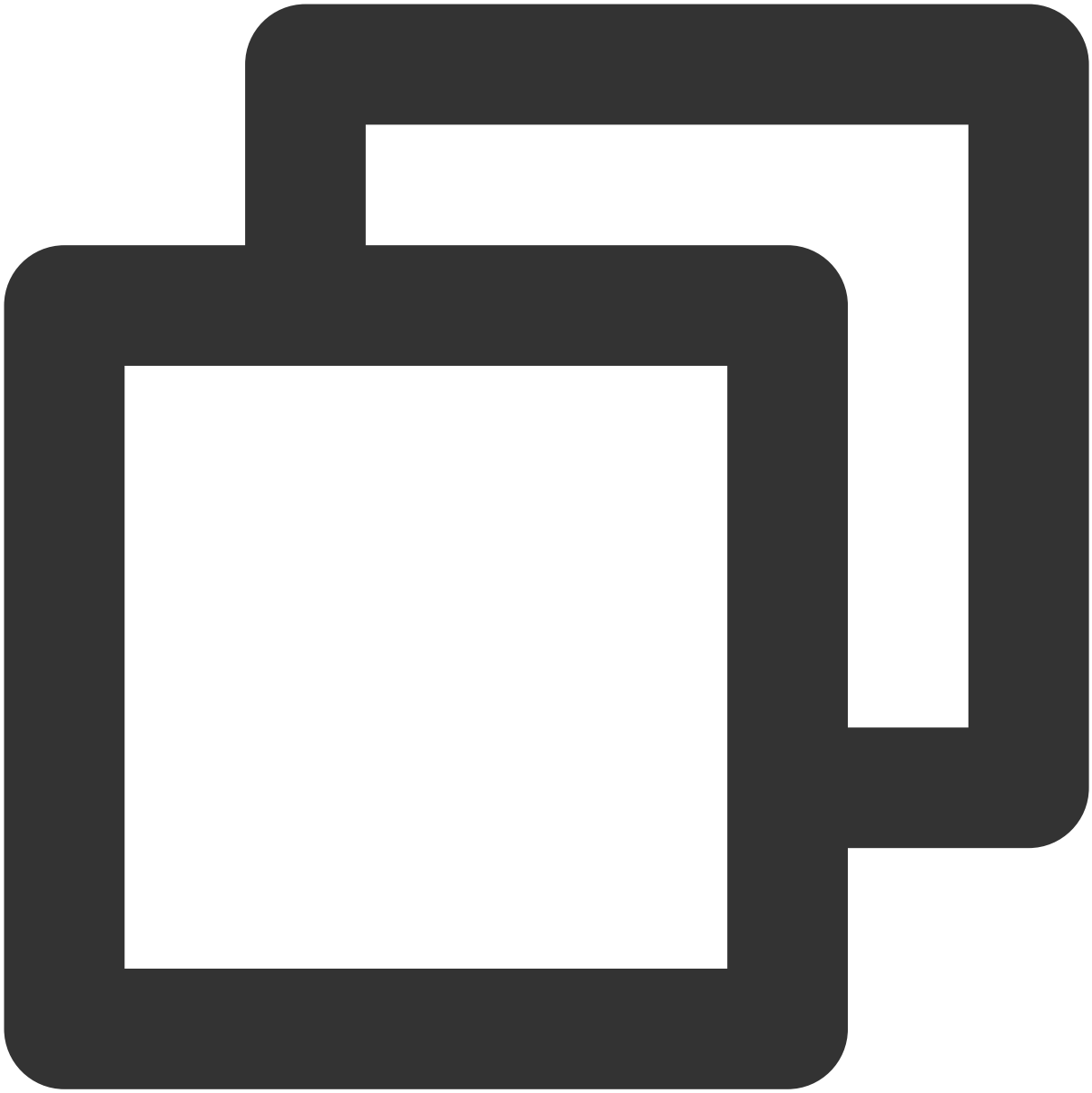
This feature is suitable for modifying columns in Tables with Large Data Volumes.

Test Results

Number of table rows	Modification Duration without using Second-level Column Modification Feature	Modification Duration with Second-level Column Modification Feature
1 million	22.9 seconds	0.01 seconds
10 Million	13 minutes and 39.72 seconds	0.01 seconds
100 million	3 hours, 51 minutes, 16.40 seconds	0.01 seconds

Use Instructions

Second-level Column Modification Syntax ALTER TABLE add algorithm = instant Clause, column modification can be performed with the following statement:



```
ALTER TABLE modify_tab_col MODIFY COLUMN c1 BIGINT,ALGORITHM=INSTANT;
```

Add the parameter cdb_instant_modify_column_enabled to control the Second-level Column Modification feature, which can be set to ON/OFF.

Parameter Name	Status	Type	Default Value	Valid Values/Value Range	Description
cdb_instant_modify_column_enabled	yes	bool	OFF	ON/OFF	Feature Toggle, determining whether

					to enable the Second-level Column Modification feature.

Note:

Users cannot directly modify the parameter values mentioned above. To make changes, submit a ticket.

Second-level Column Modification Restrictions

Support is limited to modifying column types only; changes to the field's nullable, unsigned/signed, charset are not supported, but modifying the default property is allowed.

Only certain type modifications are supported, and only length can be increased. Currently, conversions are only supported between char and varchar, binary and varbinary, and amongst tinyint/smallint/mediumint/int/bigint.

For example:

char(10) → char(100)

char(10) → varchar(100)

varchar(10) → char(100)

varchar(10) → varchar(100)

binary(10) → binary(100)

binary(10) → varbinary(100)

varbinary(10) → binary(100)

varbinary(10) → varbinary(100)

tinyint/smallint/mediumint/int → bigint

tinyint/smallint/mediumint → int

tinyint/smallint → mediumint

tinyint → smallint

Note:

Modifications between char/varchar and binary/varbinary, as well as integer types, are not supported. For example, changing from char to binary, binary to varchar, or reducing the range of integer types, such as from bigint to int or int to smallint, is not allowed.

A single column can only be modified using INSTANT MODIFY COLUMN once, but multiple columns can be modified with it simultaneously.

After a single column is added/modified using INSTANT ADD/MODIFY COLUMN for the first time, any subsequent modifications to that column must be done in a non-instant manner.

INSTANT ADD COLUMNS and INSTANT MODIFY COLUMNS operations must be executed separately. You may first execute INSTANT ADD COLUMNS, then INSTANT MODIFY COLUMNS, or vice versa. You cannot perform

INSTANT MODIFY COLUMN on a column that has been added with INSTANT ADD COLUMN.

You cannot modify the column name and column type at the same time. Instead, you can modify the column name first and then column type.

Import/export is not supported.

Encryption and compression are not supported.

Async Deletion of Big Tables

Last updated：2024-07-22 12:42:20

Overview

This feature is used to drop tables with large data files to avoid I/O fluctuation.
`DROP TABLE` renames the original database file (.ibd) to make a new temp file and returns a success. The temp file is stored in the directory specified by the `innodb_async_drop_tmp_dir` parameter and is truncated in batches on the backend. The size of the file to be truncated each time is specified by the `innodb_async_truncate_size` parameter. The status of the async table drop feature is controlled by the `innodb_async_truncate_work_enabled` parameter.
This feature does not require user operations and is automatically completed by the kernel. The principle is to create a hard link in another directory for the data file of the table when the table is dropped, so when `DROP TABLE` is executed, only the hard link to the file is deleted. After that, the backend thread will scan the files that need to be deleted in the hard-linked directory and automatically truncate the data file of the dropped table.

Supported Versions

Kernel version: MySQL 5.6 20200303 and later.
Kernel version: MySQL 5.7 20220715 and later.
Kernel version: MySQL 8.0 20200630 and later.

Use Cases

This feature is used to drop tables with large data files.

Instructions

For MySQL 5.6, you can set the `innodb_async_truncate_work_enabled` parameter to `ON` to enable the async mode of `DROP TABLE` . The default value is `OFF` .
For MySQL 5.7 and 8.0, you can set the `innodb_table_drop_mode` parameter to `ASYNC_DROP` to enable the async mode of `DROP TABLE` . The default value is `SYNC_DROP` .

The size of the file to be truncated each time is specified by the `innodb_async_truncate_size` parameter.
This is not supported for MySQL 5.6.
You can make the async drop of big tables more efficient by enabling the `innodb_fast_ddl` parameter as instructed in FAST DDL.
MySQL 5.6 parameter description
MySQL 5.7 parameter description
MySQL 8.0 parameter description

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	Description
innodb_async_truncate_work_enabled	Yes	string	OFF	ON/OFF	Whether to drop big tables asynchronously

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	D
innodb_table_drop_mode	Yes	string	SYNC_DROP	SYNC_DROP/ASYNC_DROP	W b a
innodb_async_truncate_size	Yes	int	128	128–168	T fi tr ti b th o `T` e M

Parameter	Effective Immediately	Type	Default Value	Valid Values/Value Range	D
innodb_table_drop_mode	Yes	string	SYNC_DROP	SYNC_DROP/ASYNC_DROP	W b a
innodb_async_truncate_size	Yes	int	128	128–168	T fi

					tr ti b th o `T` e M

Hotspot Update

Last updated：2024-07-22 12:42:32

Overview

For businesses with frequent updates or flash sales, the hotspot update feature greatly optimizes the performance of the UPDATE operation on frequently updated rows. If automatic hotspot update detection is enabled, the system will automatically detect whether there is a single row of hotspot update, and if so, it will queue the large number of concurrent UPDATE operations and execute them in sequence, so as to reduce the risk of concurrency performance being compromised by many row locks.

Supported Versions

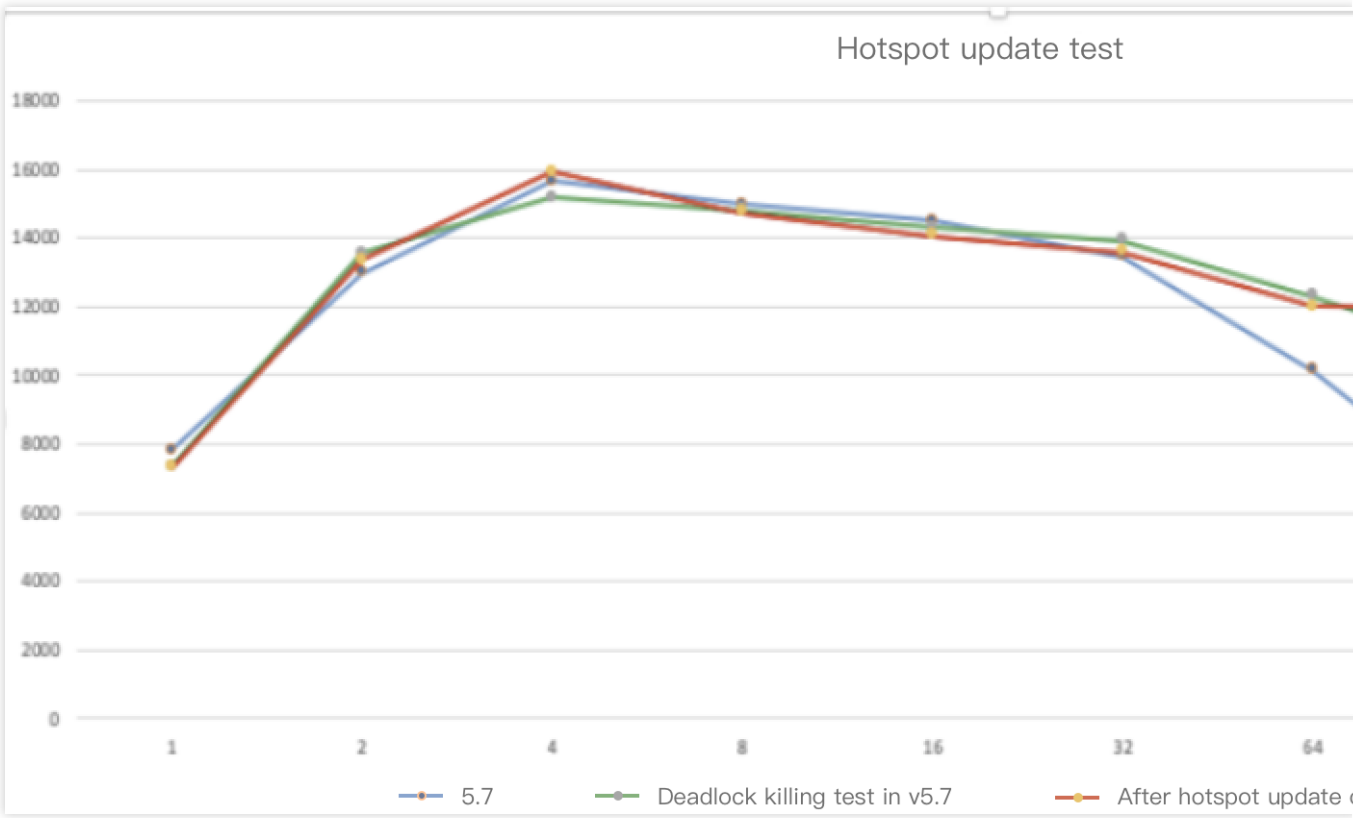
Kernel version: MySQL 5.7 20200630 and above.
Kernel version: MySQL 8.0 20200830 and above.

Use Cases

This feature is suitable for scenarios where the pressure of updating a single row or multiple rows with the primary key specified is very high, such as flash sales.

Performance Data

For high-concurrency UPDATE operations on a single row with the primary key specified, the performance is improved by over 10 times.



Instructions

For more information, see Hotspot Update Protection.

SQL Throttling

Last updated：2024-06-18 17:24:18

Overview

The SQL throttling feature enables you to set a keyword to limit the number of concurrent executions of the specified SQL statement.

Supported Versions

Kernel Version MySQL 8.0 20211202 and Later
Kernel Version MySQL 5.7 20200331 and Later
Kernel Version MySQL 5.6 20200915 and Later

Use Cases

This feature is suitable for SQL statements with high concurrency and resource usage that compromise system performance.

Instructions

For more information, see SQL Throttling.

Statement Outline

Last updated：2024-07-22 12:42:52

Overview

SQL optimization is a crucial step in improving database performance. To avoid the impact when the optimizer fails to select an appropriate execution plan, TXSQL provides the outline feature for you to bind execution plans. MySQL allows you to use hints to manually bind execution plans. The hint information contains the optimization rule for SQL statements, algorithm to be used, and index for data scan, and an outline relies on hints to specify execution plans. Tencent Cloud provides the `mysql.outline` system table for you to add plan binding rules and the `cdb_opt_outline_enabled` switch for you to enable/disable the outline feature.

Supported Versions

Kernel version: MySQL 8.0 20201230 and above.

Use Cases

This feature is suitable for scenarios where an execution plan in the production environment has poor performance (for example, the index in the execution plan is incorrect), but you don't want to modify SQL statements and release a new version to fix this problem.

Performance Impact

If `cdb_opt_outline_enabled` is enabled, the execution efficiency of SQL statements missing the outline will not be affected.
The execution efficiency of SQL statements hitting the outline will be lower than that of general execution, but the performance after outline binding is generally improved by several times.
To use `cdb_opt_outline_enabled`, you should consult the OPS or kernel engineers to avoid faulty binding and consequential performance compromise.

Instructions

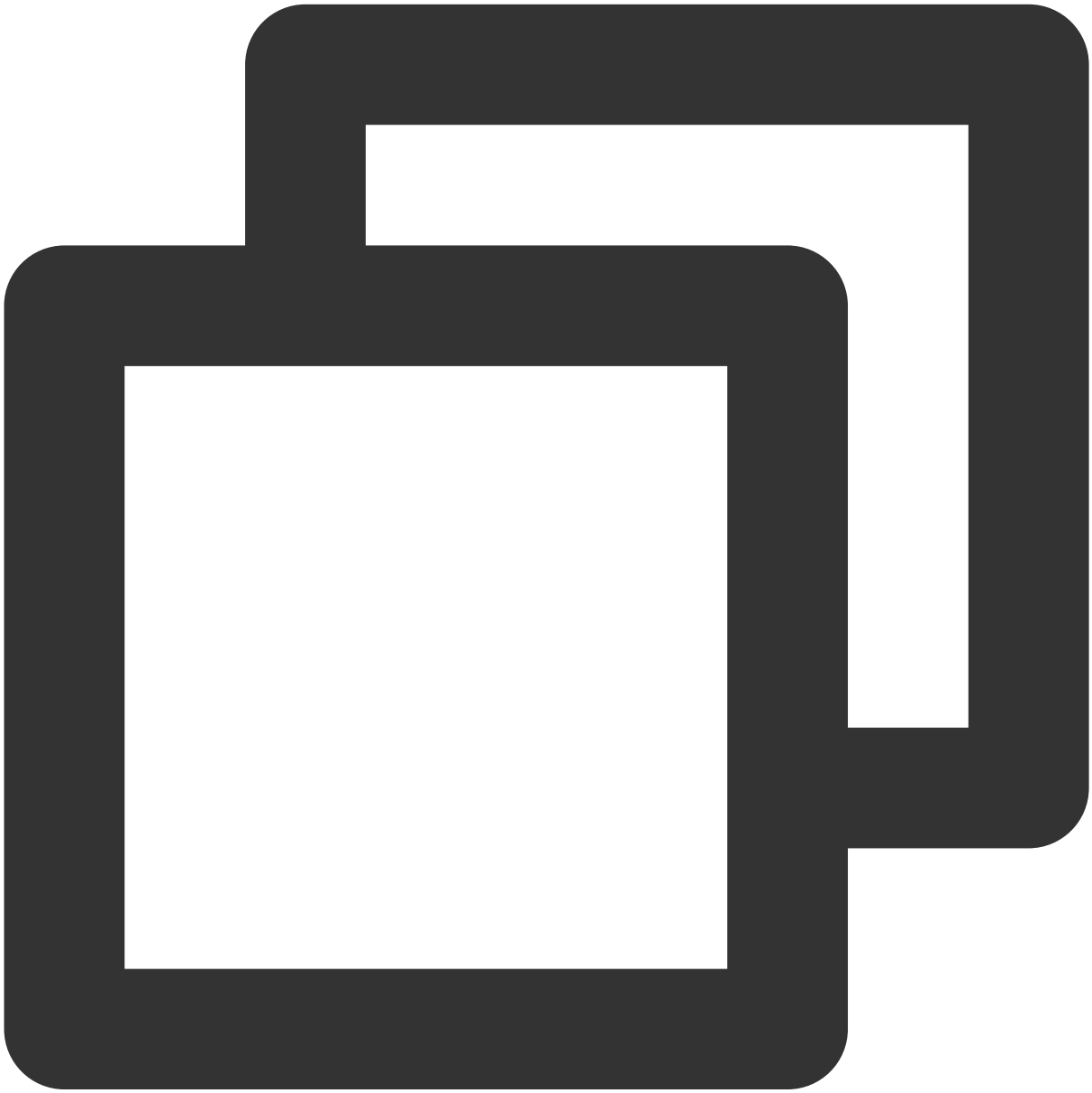
The outline syntax uses a new syntax form:

Configure outline information: `outline "sql" set outline_info "outline";`

Clear outline information: `outline reset ""; outline reset all;`

Refresh outline information: `outline flush;`

Below are the outline use methods with the following schemas as examples:



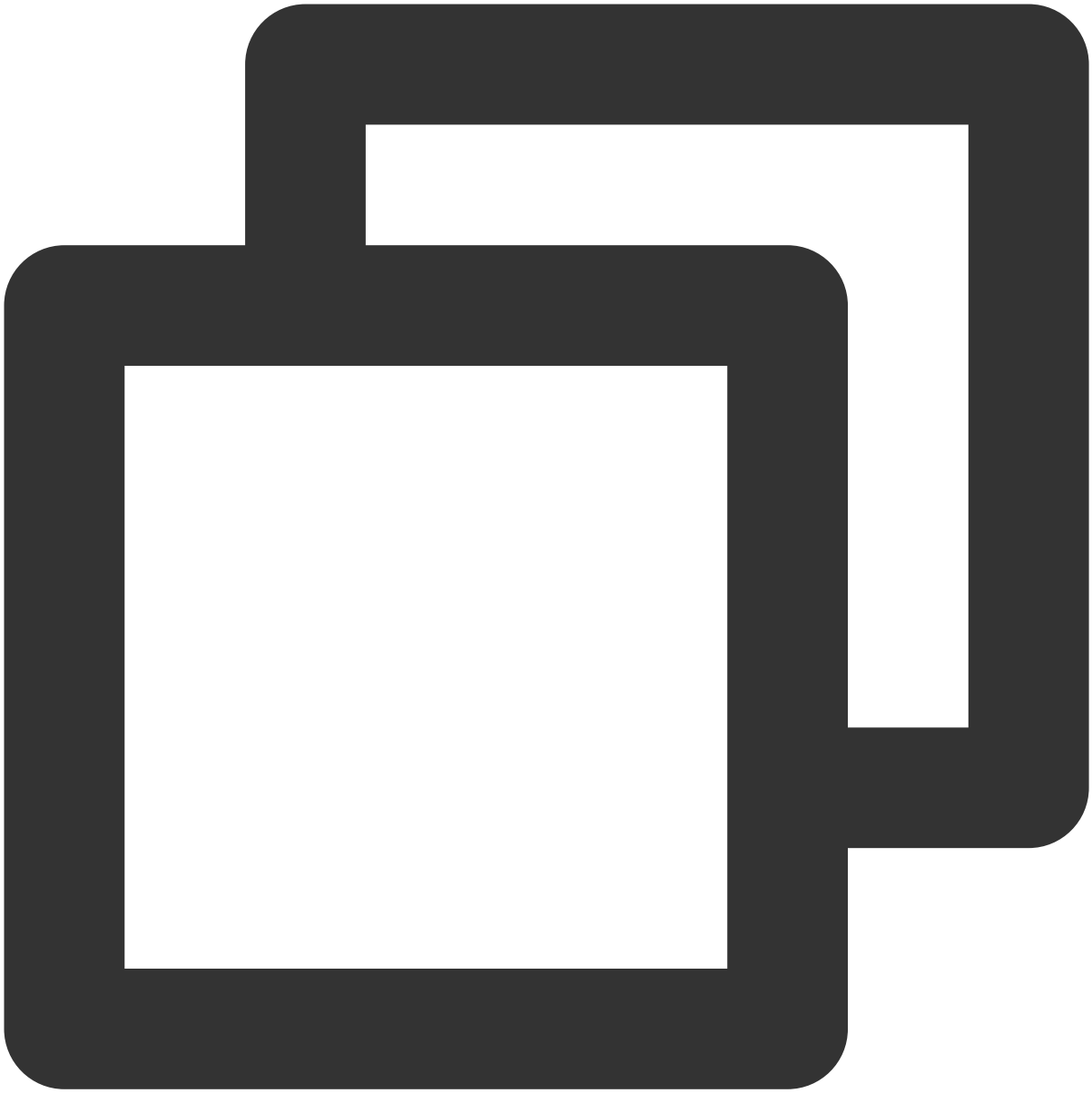
```
create table t1(a int, b int, c int, primary key(a));
create table t2(a int, b int, c int, unique key idx2(a));
create table t3(a int, b int, c int, unique key idx3(a));
```

Parameter	Effective Immediately	Type	Default Value	Valid Values	Description
cdb_opt_outline_enabled	Yes	bool	false	true/false	Whether to enable the outline feature.

Note:
Currently, you cannot directly modify the values of the above parameter. If needed, submit a ticket for assistance.

Binding outline

To bind an outline, replace the SQL statement with another where the SQL syntax is not changed but some hint information is added to tell the optimizer how to execute the statement.
The syntax is in the format of `outline "sql" set outline_info "OUTLINE";` . Note that the string after `outline_info` must start with `"OUTLINE:"` , which is followed by the SQL statement with the hint information added. For example, you can add the index in column `a` to table `t2` in the SQL statement `select *from t1, t2 where t1.a = t2.a` as follows:



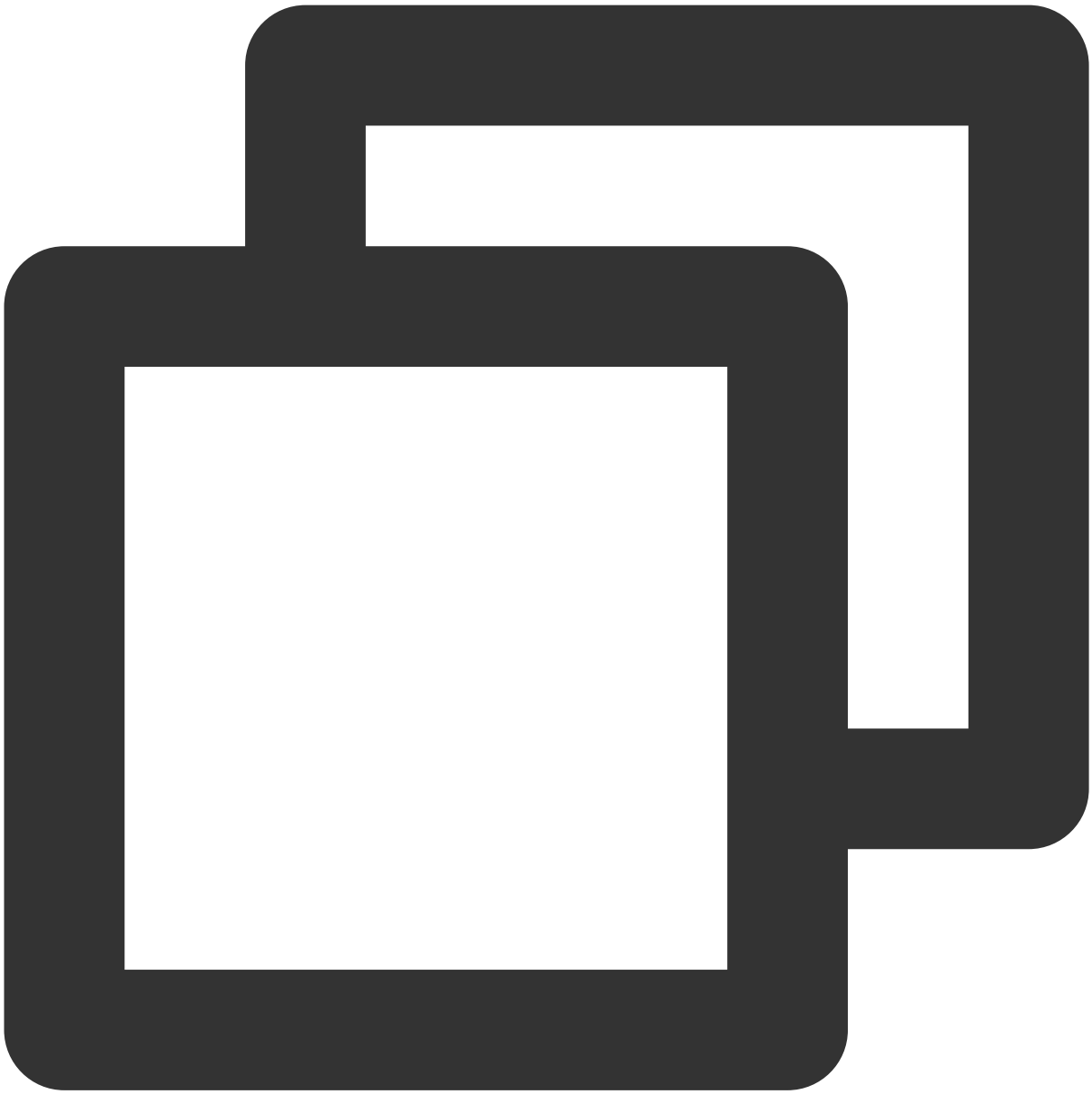
```
outline "select* from t1, t2 where t1.a = t2.a" set outline_info "OUTLINE:select *
```

Binding optimizer hint

To make the feature more flexible, TXSQL allows you to add optimizer hints incrementally to SQL statements. You can also implement the same feature by directly binding an outline.

The syntax is in the format of `outline "sql" set outline_info "outline";` . Note that the string after `outline_info` must start with `"OPT:"` , which is followed by the optimizer hint information to be added. For

example, you can specify `SEMIJOIN` of `MATERIALIZATION/DUPSWEEDOUT` for the SQL statement `select *from t1 where t1.a in (select b from t2)` as follows:



```
outline "select* from t1 where t1.a in (select b from t2)" set outline_info "OPT:2#
outline "select * from t1 where t1.a in (select b from t2)" set outline_info "OPT:1
```

You can add only one optimizer hint to the original SQL statement at a time and must comply with the following rules:

The `OPT` keyword must follow ".

':' must be placed before the new statement to be bound.

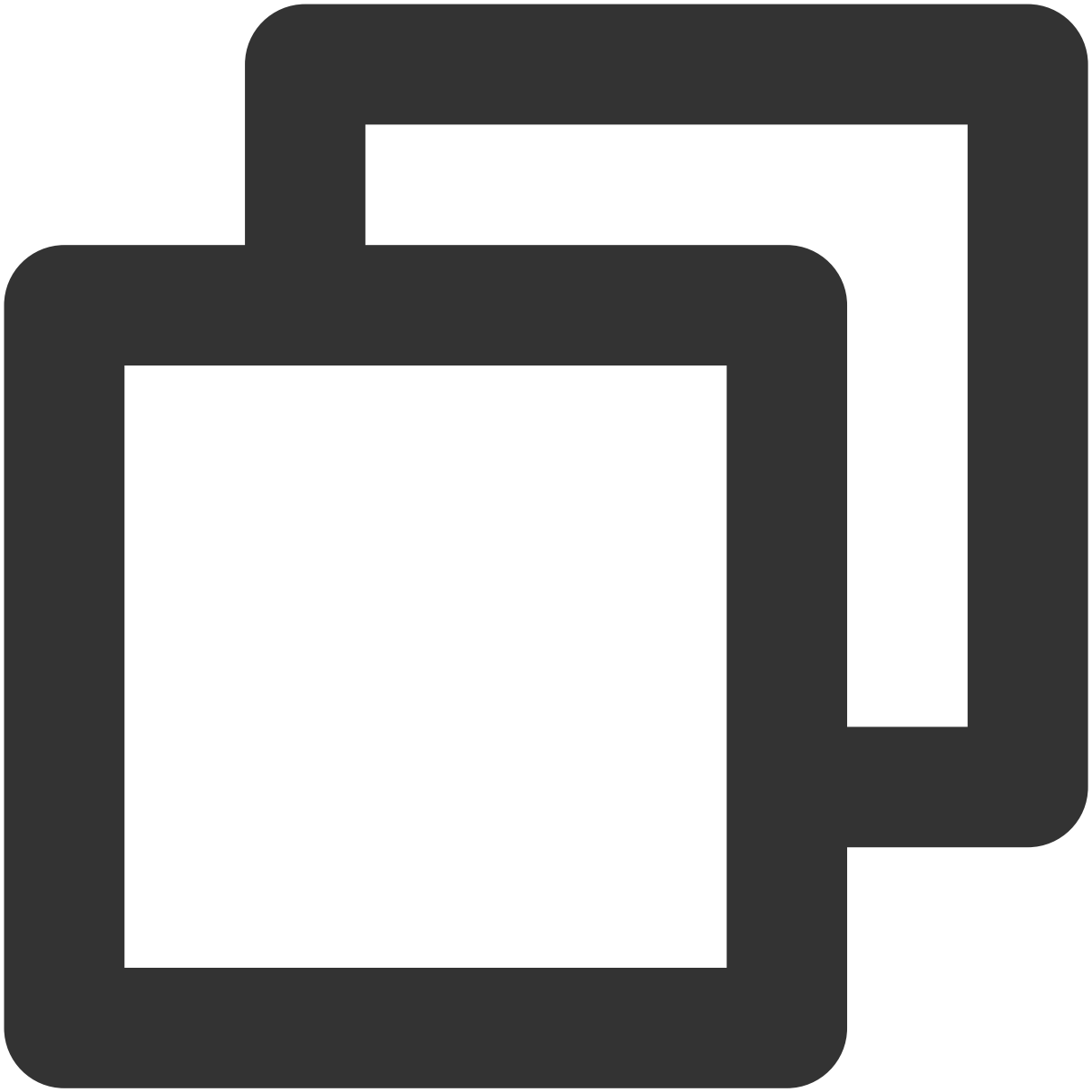
You must add two fields (query block number#optimizer hint string), which must be separated with "#" (e.g., `"OPT:1#max_execution_time(1000)"`).

Binding index hint

To make the feature more flexible, TXSQL allows you to add index hints incrementally to SQL statements. You can also implement the same feature by directly binding an outline.

The syntax is in the format of `outline "sql" set outline_info "outline";` . Note that the string after `outline_info` must start with `"INDEX:"` , which is followed by the index hint information to be added.

For example, you can add the index `idx1` of `USE INDEX` in `FOR JOIN` type to the table `t1` in the database `test` in query block 3 for the SQL statement `select *from t1 where t1.a in (select t1.a from t1 where t1.b in (select t1.a from t1 left join t2 on t1.a = t2.a))` as follows:



```
outline "select* from t1 where t1.a in (select t1.a from t1 where t1.b in (select t
```

You can add only one index hint to the original SQL statement at a time and must comply with the following rules:

The `INDEX` keyword must follow ".

':' must be placed before the new statement to be bound.

You must add five fields (query block number#db_name#table_name#index_name#index_type#clause).

Here, `index_type` has three valid values (0: INDEX_HINT_IGNORE; 1: INDEX_HINT_USE; 2: INDEX_HINT_FORCE), and `clause` also has three valid values (1: FOR JOIN; 2: FOR ORDER BY; 3: FOR

GROUP BY), which must be separated by "#" (e.g., `"INDEX:2#test#t2#idx2#1#0"`, indicating to bind the index `idx1` in `USE INDEX FOR JOIN` type to the table `test.t2` in the second query block).

Deleting the outline information of SQL statement

TXSQL allows you to delete the outline binding information from an SQL statement.
The syntax is in the format of `outline reset "sql";`. For example, to delete the outline information from `select *from t1, t2 where t1.a = t2.a`, run the following statement: `outline reset "select* from t1, t2 where t1.a = t2.a";`.

Clearing all outline information

TXSQL allows you to clear all outline binding information in the kernel. The syntax is `outline reset all`, and the execution statement is `outline reset all;`.
There may be some specific problems in the production environment where you must bind an index. In this case, you can directly configure an outline for binding.
You should analyze the possible performance compromise after configuring an outline and bind an outline only if the compromised performance is acceptable. You can consult kernel engineers if necessary.

Parameter Status Description

TXSQL provides multiple methods to view the outline binding of your SQL statements. You can use the `mysql.outline` table to view the information of configured outlines. You can also use the `show cdb_outline_info` and `select * from information_schema.cdb_outline_info` APIs to view the outline information in the memory. Whether the entered SQL statement is bound to the specified outline is subject to whether the outline information is in the memory. Therefore, you can use the two APIs for debugging.
The `mysql.outline` system table is added to store the records of configured outline information, which has the following fields:

Field Name	Description
Id	Outline number.
Digest	Hash value of the original SQL statement.
Digest_text	Fingerprint information text of the original SQL statement.
Outline_text	Fingerprint information text of the SQL statement after the outline is bound.

You can use `show cdb_outline_info` or `select * from information_schema.cdb_outline_info` to view the outline records in the memory, and execution of the

corresponding SQL statement will hit the bound plan in the outline. The parameters are as follows:

Field Name	Description
origin	Original SQL statement fingerprint.
outline	SQL statement fingerprint after the outline is bound.

TXRocks Engine Overview

Last updated：2024-07-22 12:48:36

TXRocks Overview

TXRocks is a transactional storage engine developed by Tencent's TXSQL team based on RocksDB, a very popular high-performance persistent key-value (KV) store.

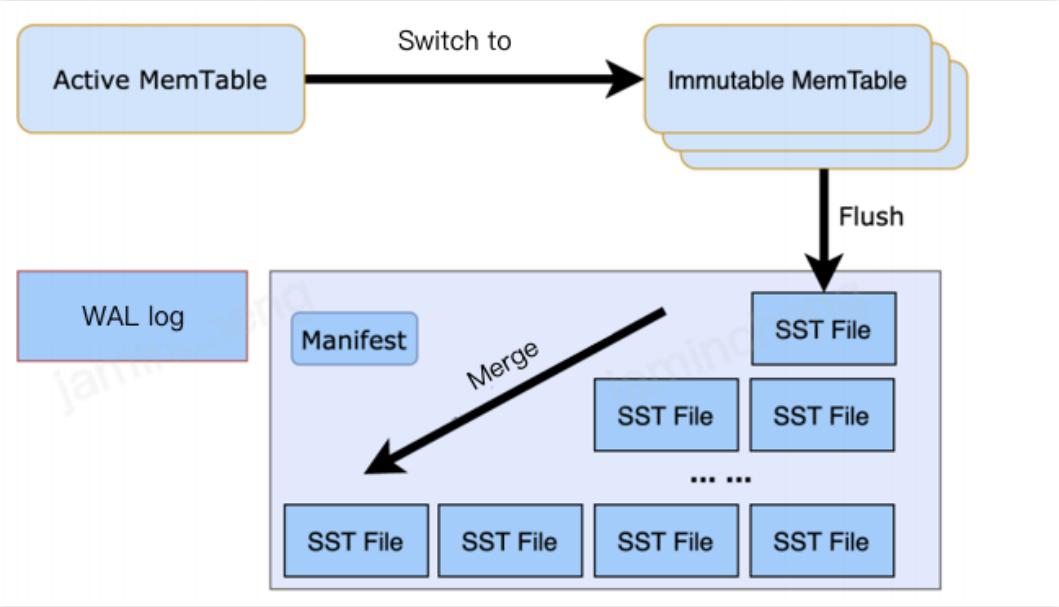
Why TXRocks?

By leveraging the LSM tree storage structure of RocksDB, the TXRocks transactional storage engine not only reduces wastes caused by InnoDB's half-full pages and fragments, but also uses the compact storage format. Therefore, it has a performance comparable to that of InnoDB but requires only a half or even smaller storage space. It is more suitable for businesses with a large data volume and high requirements for the transactional read/write performance.

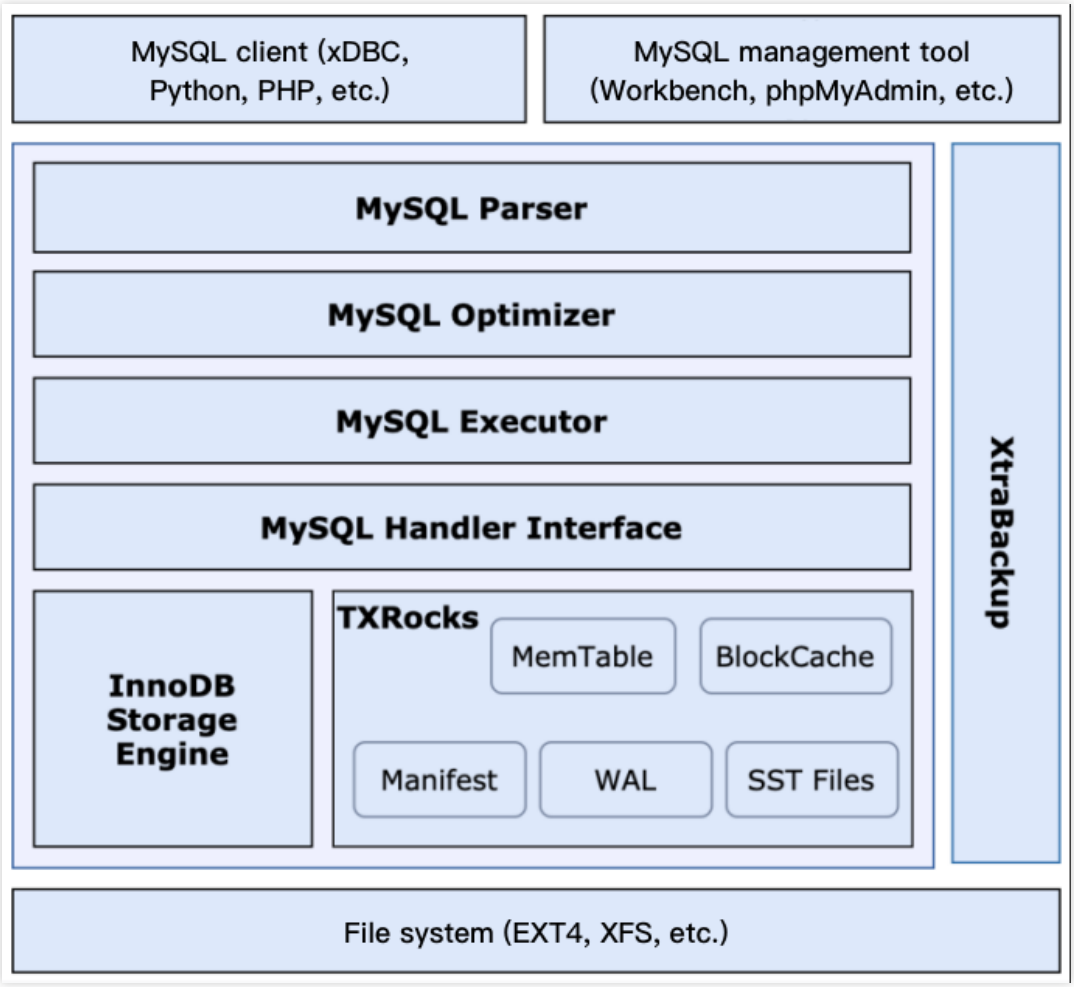
RocksDB's LSM Tree Architecture

RocksDB uses the LSM tree storage structure, where data is organized as a set of MemTables in the memory and multi-level SST files on the disk.
For a write request, the new version of the record is first written to an active MemTable, and then WAL is written for data durability. After the MemTable and WAL file are written for the request, a response can be returned.
When the volume of data written into an active MemTable reaches a certain threshold, the MemTable will be switched to a frozen immutable MemTable. The backend thread flushes the immutable MemTable to the disk to generate an SST file. SST files are sorted at L0 to L6 levels in the flush order. At each level, the records of different SST files are sequential and don't overlap. However, to release the memory space occupied by immutable MemTables promptly, such records are allowed to overlap at L0.
When a record is read, it will be searched for in the active MemTable, immutable MemTable, SST files at L0, and SST files at L1–6 in sequence. Once the record is found in any component, the latest version of the record is found and will be returned immediately.
When a range scan is performed, an iterator will be generated at each level of data containing MemTables. The iterators perform a merge sort to find the next record. As shown in the read process, if the LSM tree has too many levels, the read performance, especially the performance of range scan will drop significantly. Therefore, to maintain a

better LSM tree shape, the backend continuously performs compaction operations to merge low-level data to a higher level and thus reduce the number of levels.



TXRocks Architecture



TXRocks Strengths

Less storage space

Compared with the B+tree structure used by InnoDB, the LSM tree can save a considerable amount of storage space. InnoDB's B+tree split often results in half-full pages, idle pages, and space waste; therefore, InnoDB has a lower effective page utilization.

The size of TXRocks SST files is generally set to dozens or hundreds of MB or a greater value. Therefore, TXRocks has much fewer wastes caused by 4K alignment. Although an SST file is divided into blocks, those blocks don't need to be aligned.

In addition, TXRocks SST files use prefix compression, so that only one record will be generated for data records with the same prefix. SST files at different levels can adopt different compression algorithms, further reducing the storage space overheads. For transaction overheads, InnoDB records must contain `trx id` and `roll_ptr` fields. By contrast, other transaction overheads don't need to be stored for SST files at the lowest level of TXRocks (containing most data); for example, the version number in a record can be erased after a long enough period of time.

Lower write amplification

InnoDB uses the in-place change mode, where the entire page may be flushed to the disk even when only one row of data is changed, causing a high write amplification and more random writes.
TXRocks uses the append-only change mode, which has a lower write amplification; therefore, it is more friendly to devices such as SSD with a limited number of write cycles.

Use Cases

TXRocks is very suitable for businesses that are sensitive to the storage costs, have much more writes than reads and a large data volume, and require a high transaction read/write performance.

How to Use TXRocks

For more information, see Instructions.

Optimization and Subsequent Development

The TXSQL team has been optimizing TXRocks based on business needs; for example, they have improved the SUM query performance by over 30 times. They are also actively exploring the integration with new hardware to uses AEP as the L2 cache for higher performance and cost-effectiveness.
As the storage engine of TencentDB for MySQL, TXRocks will be continuously optimized and improved with regard to problems encountered during use. The TXSQL team will also make technical explorations based on new hardware and release TXRocks in more key services as an important supplement to InnoDB.

Instructions

Last updated：2024-07-22 12:50:43

TXRocks is a transactional storage engine developed by Tencent's TXSQL team based on RocksDB. It saves more storage space and has a lower write amplification.

Product Overview

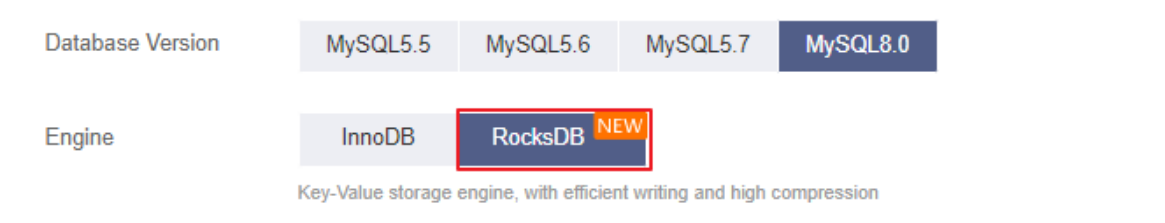
By leveraging the LSM tree storage structure of RocksDB, the TXRocks transactional storage engine not only reduces wastes caused by InnoDB's half-full pages and fragments, but also uses the compact storage format. Therefore, it has a performance comparable to that of InnoDB but requires only a half or even smaller storage space. It is more suitable for businesses with a large data volume and high requirements for the transactional read/write performance.

Prerequisites

The database version must be MySQL 5.7 or 8.0 on a two-node architecture.

Purchasing TencentDB for MySQL Instance (with RocksDB Engine)

You can select RocksDB as the engine when purchasing an instance on the TencentDB for MySQL purchase page. For more information on other parameters, see Creating MySQL Instance.

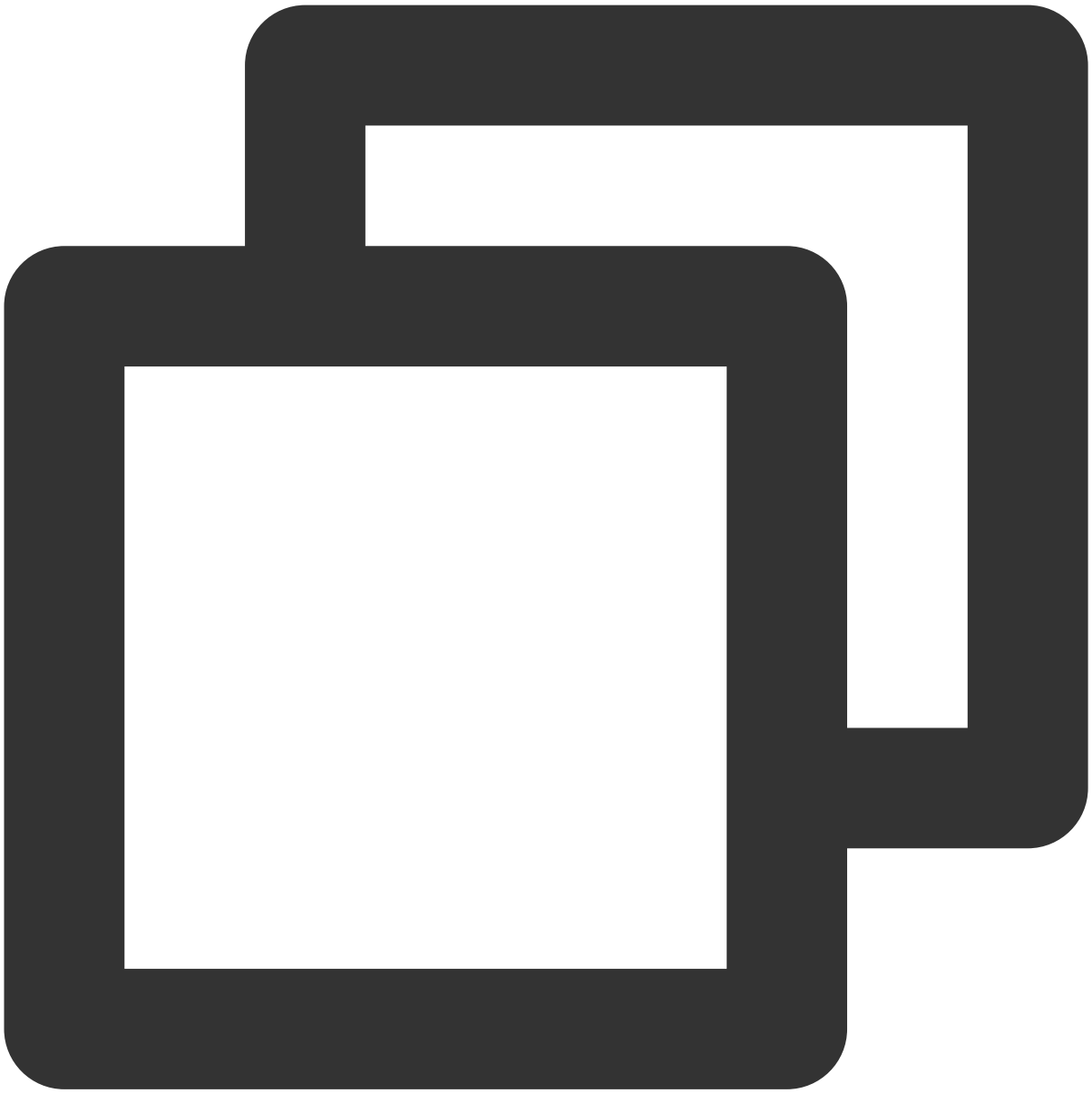


Note:

RocksDB is a key-value storage engine, with efficient writing and high compression. Currently, only TencentDB for MySQL 5.7 and 8.0 instances can use the RocksDB engine.

Creating RocksDB Table

If RocksDB is selected as the default engine during instance creation, it will be the default engine used for table creation. You can run the following command to view the default engine:



```
show variables like '%default_storage_engine%';
```

```
mysql> show variables like '%default_storage_engine%';
+------------------------+---------+
| Variable_name          | Value   |
+------------------------+---------+
| default_storage_engine | RocksDB |
+------------------------+---------+
1 row in set (0.00 sec)
```

If the default engine is RocksDB, you cannot specify a storage engine in table creation statements:

```
mysql> create table tencent_db (id int primary key,c1 varchar(30),c2 varcha
Query OK, 0 rows affected (0.00 sec)

mysql> show create table tencent_db;
+------------+-----------------------------------------------------------------
-----------------+
| Table      | Create Table
                |
+------------+-----------------------------------------------------------------
-----------------+
| tencent_db | CREATE TABLE `tencent_db` (
  `id` int(11) NOT NULL,
  `c1` varchar(30) DEFAULT NULL,
  `c2` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=RocksDB DEFAULT CHARSET=utf8 |
+------------+-----------------------------------------------------------------
-----------------+
1 row in set (0.00 sec)
```

After a table is created, its data will be stored in RocksDB and can be used in the same way as in InnoDB.

Engine Feature Limits

TXRocks has certain limits on engine features as detailed below:

Category	Feature	TXRocks Limit
DDL	Online DDL	Not supported. For example, `ALTER TABLE ... ALOGRITHM=INSTANT` is not supported. Only the `COPY` algorithm is supported for partition management operations.
SQL	Foreign key	Not supported.
	Partitioned table	Not supported.
	Generated column	Not supported.
	Explicit DEFAULT	Not supported. For example, `CREATE TABLE t1(c1

	expression	FLOAT DEFAULT(RAND()))ENGINE=ROCKSDB` will fail, with the error `'Specited storage engine' is not supported for default value expressions` reported.
	Encrypted table	Not supported.
Index	Spatial index	The spatial index and spatial data types such as `GEOMETRY` and `POINT` are not supported.
	Full-text index	Not supported.
	Multi-valued index	Not supported.
Replication	Group replication	Not supported.
	Binlog format	Only the `ROW` format is supported, while `STMT` and `MIXED` formats are not.
	Clone plugin	Not supported.
	Transportable tablespace	Not supported.
Transaction and lock	LOCK NOWAIT and SKIP LOCKED	Not supported.
	Gap lock	Not supported.
	Savepoint	Not supported.
	Partial LOB field update	Not supported.
	XA transaction	Not recommended.

Parameter Description

Note:

When creating a TencentDB for MySQL instance, you can select RocksDB as the default storage engine. You can also customize the parameter template to suit your needs by following the parameter descriptions below.

MySQL 5.7 parameter list

Parameter	Restart Required	Default Value	Value Range/Valid Values	Description
rocksdb_use_direct_io_for_flush_and_compaction	Yes	ON	ON/OFF	Whether to use D

rocksdb_flush_log_at_trx_commit	No	1	0/1/2	Controls when to It is similar to `innodb_flush` and indicates whe be synced when t 0: Transactions a committed. 1: Transactions a committed. 2: Transactions a second.
rocksdb_lock_wait_timeout	No	1	1–1073741824	Lock wait timeout
rocksdb_deadlock_detect	No	ON	ON/OFF	Whether to enable it is enabled, all d recorded in `mys`
rocksdb_manual_wal_flush	Yes	ON	ON/OFF	If the total size of `rocksdb_max_` RocksDB will forc to the disk to ensu can be deleted.

MySQL 8.0 parameter list

Parameter	Restart Required	Default Value	Value Range/Valid Values	Descriptic
rocksdb_flush_log_at_trx_commit	No	1	0/1/2	Controls v It is simila `innodb_` and indica be synced 0: Transa committed 1: Transa committed 2: Transa second.
rocksdb_lock_wait_timeout	No	1	1–1073741824	Lock wait
rocksdb_merge_buf_size	No	524288(=512K)	100–	Size of the

			18446744073709551615	secondary
rocksdb_merge_combine_read_size	No	8388608 (=8M)	524288(=512K)–18446744073709551615	Size of the during sec
rocksdb_deadlock_detect	No	ON	ON/OFF	Whether t
rocksdb_manual_wal_flush	Yes	ON	ON/OFF	If the total `rocksd` RocksDB to the disk can be de

RocksDB Monitoring Metrics

RocksDB monitoring metrics are as listed below:

Metric	Description
rocksdb_bytes_read	Data read from disk
rocksdb_bytes_written	Data written to disk
rocksdb_block_cache_bytes_read	Blocks read
rocksdb_block_cache_bytes_write	Blocks written
rocksdb_wal_log_capacity	Data written to WAL log

Cost Performance

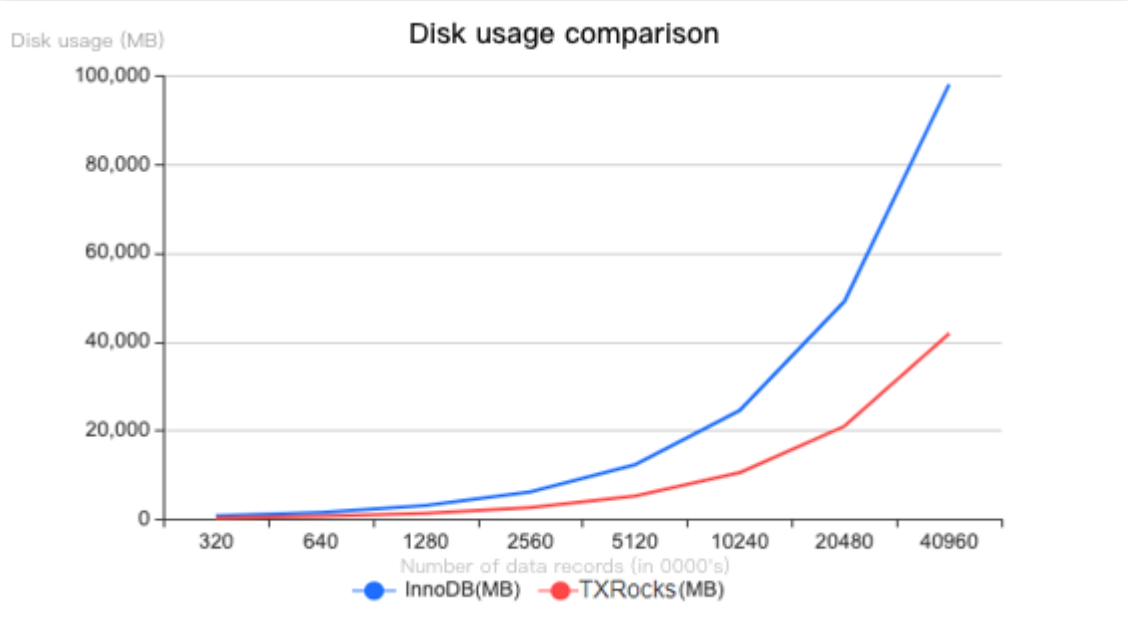
Last updated：2024-07-22 12:50:55

TXRocks has a performance comparable to that of InnoDB; however, its LSM tree structure reduces wastes caused by InnoDB half-full pages and fragments, saving more storage space and delivering an ultra high cost performance.

Background

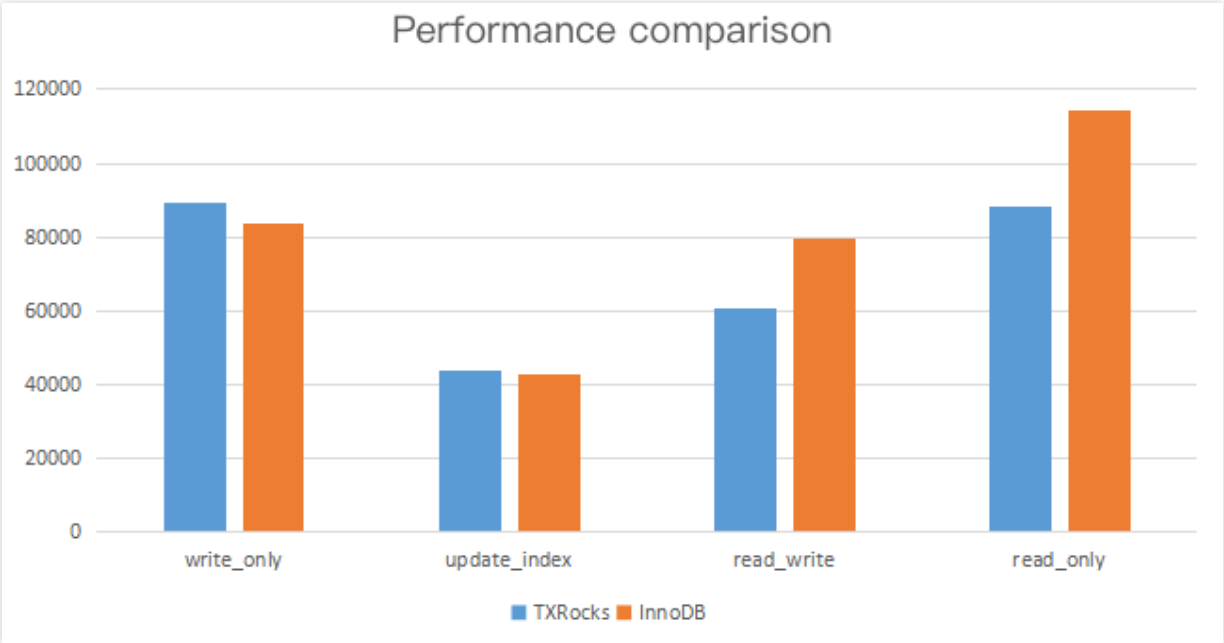
TXRocks is used in TencentDB products as an important supplement to InnoDB. With a similar performance, it is further optimized and improved to save more storage space. Below is the comparison between the two engines in terms of space usage and performance.

TXRocks Uses Less Space than InnoDB



Test scenario: Both storage engines use the default configuration and the default table structure of sysbench. Each table contains 800,000 records, and the total number of tables gradually increases from 4 to 512.
The space usage of TXRocks and InnoDB under the specified test conditions is as shown above. The disk usage is displayed on the Y axis.
As shown in the test data, the greater the data volume, the slower the increase of disk usage by TXRocks, and the higher the storage space utilization of TXRocks (it uses only 42.71% of the space used by InnoDB in the best case). For data records with highly repetitive prefixes, TXRocks has a higher compression rate and storage cost performance.

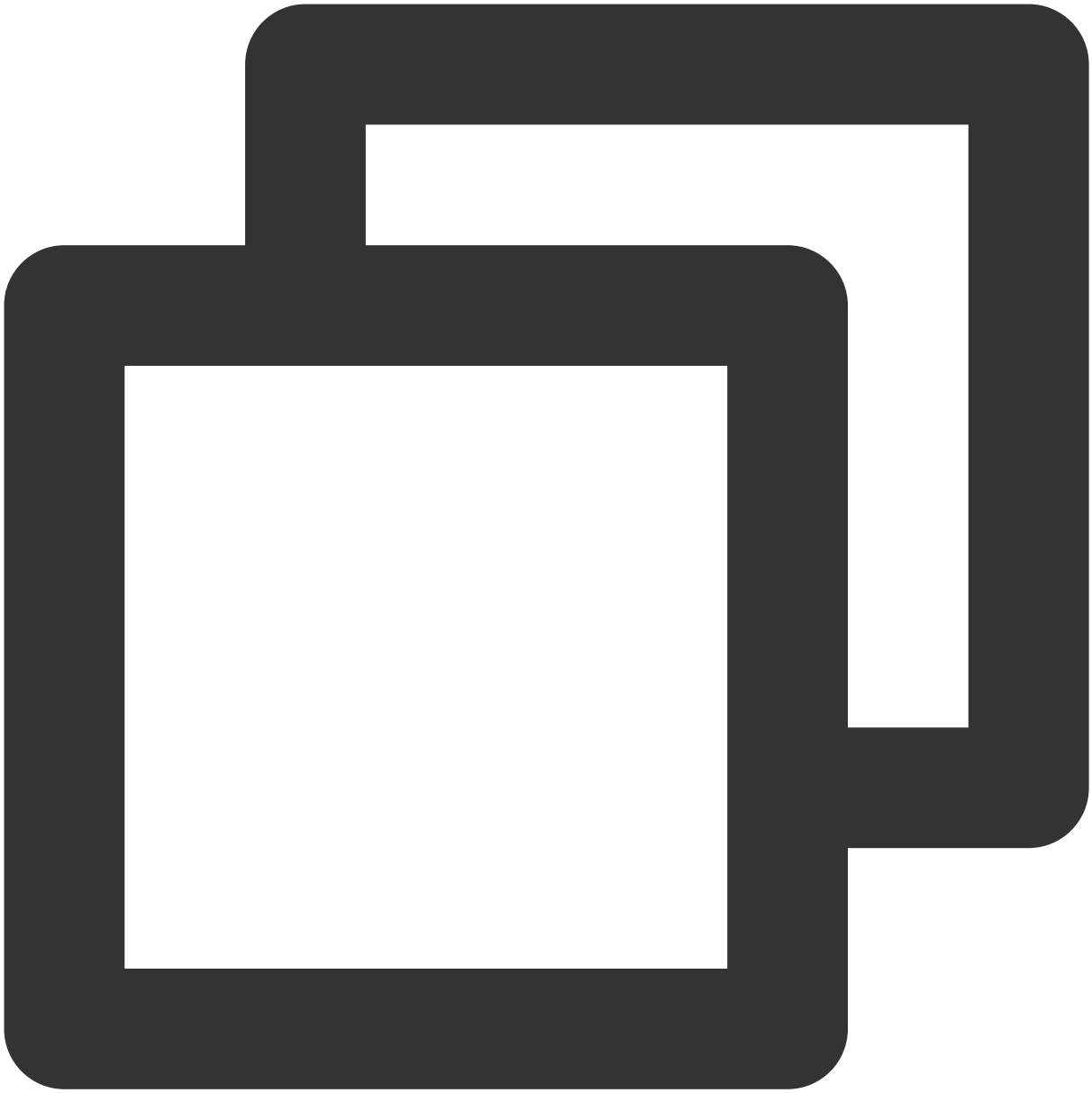
TXRocks and InnoDB Have a Similar Performance



Test scenario: An 8-core 32 GB MEM instance and six tables containing five million rows of data each are used for testing. Each test case is performed after cold instance restart and runs for 1,200 seconds.

The performance comparison between TXRocks and InnoDB under the specified test conditions is as shown above. You can see that TXRocks and InnoDB have a similar performance.

Key parameters in the sysbench command:



```
sysbench --table-size=5000000 --tables=6 --threads=32 --time=1200
```

Summary

TXRocks is a TencentDB for MySQL storage engine that has a performance comparable to that of InnoDB but uses less space. It not only guarantees the business performance, but also reduces the storage costs. For more information, see Overview.

Practical Tutorial of TXRocks

Last updated：2024-07-31 10:00:32

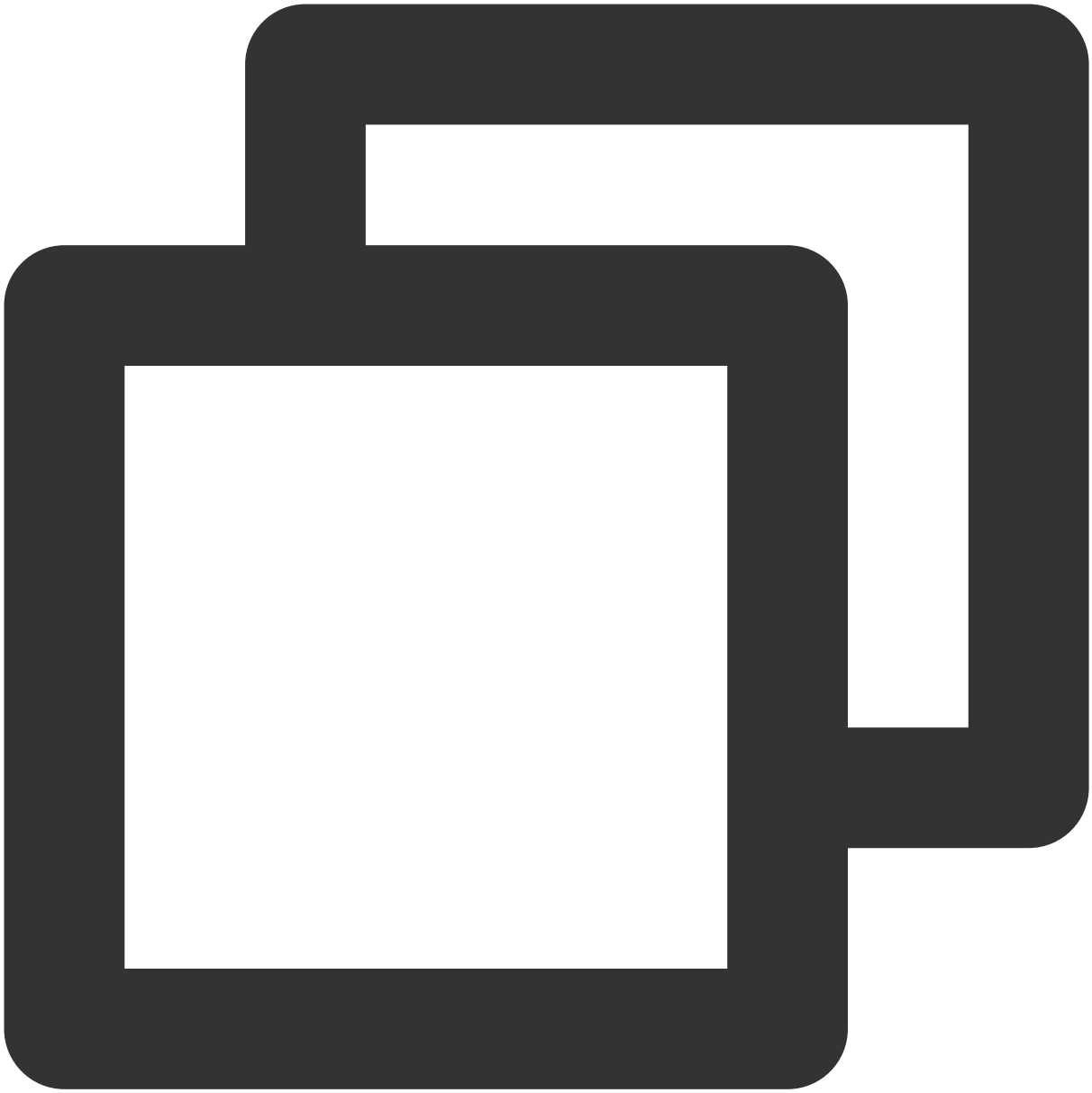
This document describes how to accelerate the import of massive amounts of data to a database with the TXRocks practical tutorial.

Background

Scenario: The import of massive amounts of data to a database with the TXRocks engine needs to be accelerated.
Impact: The `Rows inserted during bulk load must not overlap existing rows` error may be reported when massive amounts of data are imported.

Option 1

1. Delete secondary indexes and retain only the primary key index.
2. Adjust memory parameters based on the specification and data volume.
Note:
Appropriately increase the values of `rocksdb_merge_buf_size` and `rocksdb_merge_combine_read_size` parameters based on the specification and data volume.
`rocksdb_merge_buf_size` indicates the data volume of each way in k-way merge during index creation.
`rocksdb_merge_combine_read_size` indicates the total memory used in k-way merge.
`rocksdb_block_cache_size` indicates the size of `rocksdb_block_cache`. We recommend you decrease its value temporarily during k-way merge.
3. Use `bulk load` to import the data.



```
SET session rocksdb_bulk_load_allow_unsorted=1;
SET session rocksdb_bulk_load=1;
...
Import the data
...
SET session rocksdb_bulk_load=0;
SET session rocksdb_bulk_load_allow_unsorted=0;
```

Note:

If the imported data is sorted, you don't need to configure `rocksdb_bulk_load_allow_unsorted`.

4. Recreate secondary indexes one by one after all data is imported.

Note:

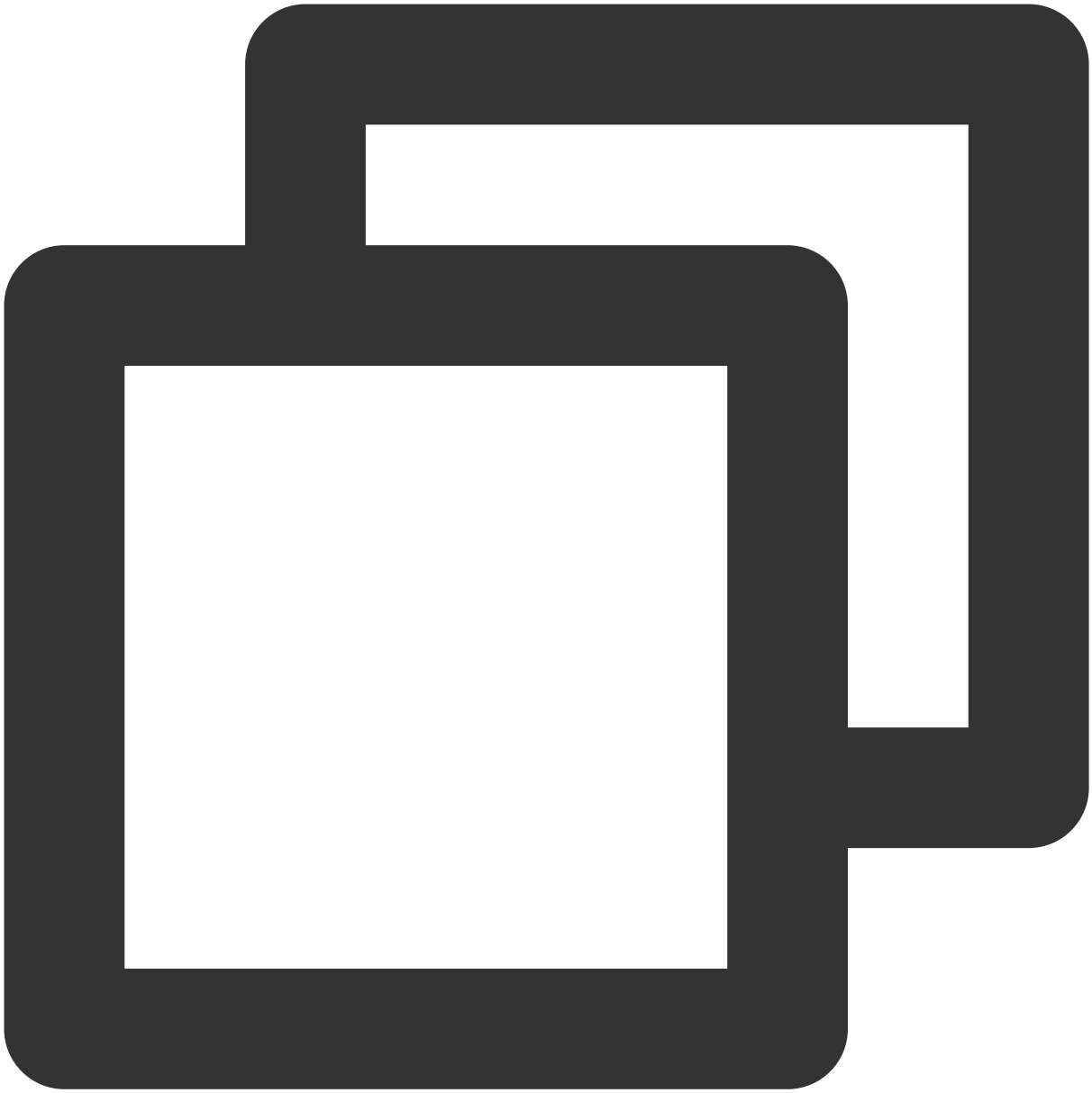
Secondary index creation involves k-way merge. `rocksdb_merge_buf_size` indicates the data volume of each way, and `rocksdb_merge_combine_read_size` indicates the total memory used in k-way merge.

For example, we recommend you set `rocksdb_merge_buf_size` to 64 MB or higher and set `rocksdb_merge_combine_read_size` to 1 GB or higher to avoid OOM. After all data is imported, you must modify the parameters to their original values.

As a lot of memory is used during the creation of each secondary index, we recommend you not create many of them at the same time.

Option 2

You can disable `unique_check` during data import to improve the import performance.



```
SET unique_checks=OFF;
...
Import the data.
...
SET unique_checks=ON;
```

Note:

After the operation is completed, you must set `unique_checks` back to `ON` ; otherwise, the uniqueness of INSERT operations in subsequent normal transaction writes will not be checked.