SET TRANSACTION

Syntax

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_property [, transaction_property] ...

transaction_property:
    ISOLATION LEVEL level
  | READ WRITE
  | READ ONLY

level:
     REPEATABLE READ
   | READ COMMITTED
   | READ UNCOMMITTED
   | SERIALIZABLE
```

Description

This statement sets the transaction isolation level or the transaction access mode globally, for the current session, or for the next transaction:

- With the `GLOBAL` keyword, the statement sets the default transaction level globally for all subsequent sessions. Existing sessions are unaffected.
- With the `SESSION` keyword, the statement sets the default transaction level for all subsequent transactions performed within the current session.
- Without any `SESSION` or `GLOBAL` keyword, the statement sets the isolation level for the next (not started) transaction performed within the current session.

A change to the global default isolation level requires the `SUPER` privilege. Any session is free to change its session isolation level (even in the middle of a transaction), or the level for its next transaction.

Isolation level

To set the global default isolation level at server startup, use the `--transaction-isolation=level` option on the command line or in an option file. Values of level for this o use dashes rather than spaces, so the allowable values are `READ-UNCOMMITTED`, `READ-COMMITTED`, `REPEATABLE-READ`, or `SERIALIZABLE`. For example, to set the defau isolation level to `REPEATABLE READ`, use these lines in the `[mysqld]` section of an option file:

```
[mysqld]
transaction-isolation = REPEATABLE-READ
```

To determine the global and session transaction isolation levels at runtime, check the value of the `tx_isolation` system variable:

```
SELECT @@GLOBAL.tx_isolation, @@tx_isolation;
```

InnoDB supports each of the translation isolation levels described here using different locking strategies. The default level is `REPEATABLE READ`. For additional information a InnoDB record-level locks and how it uses them to execute various types of statements, see XtraDB/InnoDB Lock Modes, and http://dev.mysql.com/doc/refman/en/innodb-loc set.html.

Isolation Levels

The following sections describe how MariaDB supports the different transaction levels.

READ UNCOMMITTED

`SELECT` statements are performed in a non-locking fashion, but a possible earlier version of a row might be used. Thus, using this isolation level, such reads are not consiste is also called a "dirty read." Otherwise, this isolation level works like `READ COMMITTED`.

READ COMMITTED

A somewhat Oracle-like isolation level with respect to consistent (non-locking) reads: Each consistent read, even within the same transaction, sets and reads its own fresh sn See http://dev.mysql.com/doc/refman/en/innodb-consistent-read.html.

For locking reads (`SELECT` with `FOR UPDATE` or `LOCK IN SHARE MODE`), InnoDB locks only index records, not the gaps before them, and thus allows the free insertion of n records next to locked records. For `UPDATE` and `DELETE` statements, locking depends on whether the statement uses a unique index with a unique search condition (such a `id = 100`), or a range-type search condition (such as `WHERE id > 100`). For a unique index with a unique search condition, InnoDB locks only the index record found, not before it. For range-type searches, InnoDB locks the index range scanned, using gap locks or next-key (gap plus index-record) locks to block insertions by other sessions into covered by the range. This is necessary because "phantom rows" must be blocked for MySQL replication and recovery to work.

Note: Since MariaDB 5.1, if the `READ COMMITTED` isolation level is used or the `innodb_locks_unsafe_for_binlog` system variable is enabled, there is no InnoDB gap lo except for foreign-key constraint checking and duplicate-key checking. Also, record locks for non-matching rows are released after MariaDB has evaluated the `WHERE` conditi MariaDB/MySQL 5.1, if you use `READ COMMITTED` or enable innodb_locks_unsafe_for_binlog, you must use row-based binary logging.

REPEATABLE READ

This is the default isolation level for InnoDB. For consistent reads, there is an important difference from the `READ COMMITTED` isolation level: All consistent reads within the sa transaction read the snapshot established by the first read. This convention means that if you issue several plain (non-locking) `SELECT` statements within the same transactio `SELECT` statements are consistent also with respect to each other. See http://dev.mysql.com/doc/refman/en/innodb-consistent-read.html.

For locking reads (SELECT with FOR UPDATE or LOCK IN SHARE MODE), UPDATE, and DELETE statements, locking depends on whether the statement uses a unique in a unique search condition, or a range-type search condition. For a unique index with a unique search condition, InnoDB locks only the index record found, not the gap before other search conditions, InnoDB locks the index range scanned, using gap locks or next-key (gap plus index-record) locks to block insertions by other sessions into the gaps c by the range.

This is the only transaction level that can be used with the row based binary logging.

This is the minimum isolation level for non-distributed XA transactions.

SERIALIZABLE

This level is like REPEATABLE READ, but InnoDB implicitly converts all plain SELECT statements to `SELECT ... LOCK IN SHARE MODE` if `autocommit` is disabled. If aut is enabled, the SELECT is its own transaction. It therefore is known to be read only and can be serialized if performed as a consistent (non-locking) read and need not block f transactions. (This means that to force a plain SELECT to block if other transactions have modified the selected rows, you should disable autocommit.)

Distributed XA transactions should always use this isolation level.

Access mode

These clauses appeared in MariaDB 10.0.

The access mode specifies whether the transaction is allowed to write data or not. By default, transactions are in `READ WRITE` mode (see the `tx_read_only` system variab `ONLY` mode allows the storage engine to apply optimizations that cannot be used for transactions which write data. The only exception to this rule is that read only transaction perform DDL statements on temporary tables.

It is not permitted to specify both `READ WRITE` and `READ ONLY` in the same statement.

`READ WRITE` and `READ ONLY` can also be specified in the `START TRANSACTION` statement, in which case the specified mode is only valid for one transaction.

Examples

```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Attempting to set the isolation level within an existing transaction without specifying `GLOBAL` or `SESSION`.

```
START TRANSACTION;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
ERROR 1568 (25001): Transaction characteristics can't be changed while a transaction is in progress
```

← ROLLBACK ↑ Transactions ↑ LOCK TABLES and UNLOCK TABLES →

Comments

Re: SET TRANSACTION ISOLATION LEVEL *4 years, 9 months ago* ric

there is a note about use READ COMMITTED isolation level on Maria51, what about maria55? I use Maria55.30, can i use Replication statement based with isolation READ COMMITTED and dont have any 'out of sync' on the slaves?

Re: SET TRANSACTION ISOLATION LEVEL *4 years, 6 months ago* Federicc

That note is also written in all versions of Oracle MySQL manual since 5.1. Each version says: "In MySQL 5.x". I changed the note in this page to "Since MariaDB 5.1".