Guarding Data Integrity: Transactional Behaviour in MySQL

Mikaël Francoeur, 23 Feb 24
Confoo

Contents

- What’s a transaction
- What’s a lock
- 2 types of reads
- Multi-Version Concurrency Control (MVCC)
- 3 types of read phenomena
- Transaction Isolation Levels
    - Repeatable Read
    - Read Committed
    - Read Uncommitted
    - Serializable

What’s a transaction

- Sequence of statements that is atomic WRT persistence and can be isolated from other transactions.
- Lifecycle
  - BEGIN, COMMIT, ROLLBACK, autocommit

What’s a lock

- An access right
- Resource types
  - Table
  - Row
  - Row + preceding gap
- Are kept until the end of a transaction
- Prevents concurrent modification of a resource
- all rows scanned in UPDATE/DELETE, SELECT ... FOR SHARE, SELECT ... FOR UPDATE, INSERT INTO ... SELECT (in REPEATABLE READ)

2 types of reads

- Consistent Nonlocking Reads
  - Plain SELECT
  - Read data from a single snapshot/timestamp
  - See results of earlier INSERT/UPDATE/DELETE statements in the same transaction
- Locking Reads
  - Take locks
  - Prevent concurrent modification
  - Always see the latest committed version of a row.

Multi-Version Concurrency Control (MVCC)

- Every row contains a pointer to part of an “undo log” that contains information on how to rebuild previous versions of the row.
- Prevents certain read phenomena without locking/contention.

3 Read Phenomena

Defined in SQL-92

- Non-Repeatable Reads
  - A transaction reads a row, then reads it again but the row has changed or was deleted.
- Phantom Reads
  - A transaction reads a set of rows, then reads it again but new rows are returned.
- Dirty Reads
  - A transaction reads uncommitted data from another transaction.

Transaction Isolation Levels

Prevent or allow read phenomena as a performance tradeoff.

Can be set globally, by session, or by transaction

Implemented with

- Locking
- MVCC

Four levels

- SERIALIZABLE
- REPEATABLE READ
- READ COMMITTED
- READ UNCOMMITTED

REPEATABLE READ

MVCC

- The first SELECT in a transaction establishes a snapshot that all other queries in the same transaction will read from.
  - Does not apply to locking reads
  - More resource-intensive

Locking

- Does gap locking
- Able to lock the absence of a row

Non-repeatable read	Phantom read	Dirty read	Read consistency
No*	No*	No	Transaction

READ COMMITTED

MVCC

- Every consistent nonlocking read gets its own snapshot.
- Less use of undo logs.

Locking

- No gap locking
- For UPDATE/DELETE statements, locks for evaluated rows that won't be updated are released immediately.

Non-repeatable read	Phantom read	Dirty read	Read consistency
Yes	Yes	No	Statement

READ UNCOMMITTED

Read Phenomena

- Transaction sees uncommitted data from other ongoing transactions.
- Surprising/undocumented effects.
- Do not use in production!

Useful to debug integration tests

- SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

Non-repeatable read	Phantom read	Dirty read	Read consistency
Yes	Yes	Yes	Statement

SERIALIZABLE

- Similar to REPEATABLE READ
- Every SELECT is transformed into a SELECT ... FOR SHARE, except if autocommit=ON and not in an explicit transaction.

Non-repeatable read	Phantom read	Dirty read	Read consistency
No	No	No	Transaction

Transaction isolation levels

Isolation	Non-repeatable read	Phantom read	Dirty read	Read consistency
SERIALIZABLE	No	No	No	Transaction
REPEATABLE READ	No*	No*	No	Transaction
READ COMMITTED	Yes	Yes	No	Statement
READ UNCOMMITTED	Yes	Yes	Yes	Statement

* Except for locking reads

Tips

- If you need to SELECT data, then inspect it, and then store something based on that data, consider
  - using a locking read.
  - merging the SELECT with the INSERT/UPDATE/DELETE
- Use gap locking to prevent new data from appearing while your transaction is ongoing.
- To test transaction isolation or prevent regressions
  1. Race two transactions multiple times (ex. 10 000 times)
  2. Count the number of deadlocks occurring
  3. Stop whenever you reach 10 deadlocks, or you find an unwanted read phenomena

Conclusion

- SERIALIZABLE
  - no read phenomena
  - more contention
- REPEATABLE READ
  - can be hard to reason about
  - use it to lock the absence of something
- READ COMMITTED
  - no gap locking
  - less usage of undo logs, use for longer transactions
- READ UNCOMMITTED
  - not for production

Further Reading

(arranged in descending order of recommendation)

Blog series on InnoDB locking

- Łopuszański, Kuba. “InnoDB Data Locking” (blog). https://dev.mysql.com/blog-archive/innodb-data-locking-part-1-introduction/.

On MySQL/InnoDB

- https://dev.mysql.com/doc/refman/8.0/en/innodb-transaction-isolation-levels.html
- https://dev.mysql.com/doc/refman/8.0/en/innodb-locks-set.html
- Karwin, Bill. 2022. *SQL Antipatterns, Volume 1: Avoiding the Pitfalls of Database Programming*. Raleigh, NC : The Pragmatic Bookshelf.
- Jepsen. “MySQL 8.0.34”. https://jepsen.io/analyses/mysql-8.0.34.

On transactions:

- Kleppman, Martin. “Hermitage: Testing Transaction Isolation Levels”. https://github.com/ept/hermitage.
- *SQL-92*. http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt.