Chapter 10 Notes on Transaction Management in MySQL

The file called MySQL Transaction Management in the MySQL Code for Examples directory illustrates the concepts discussed here. Lab 10.1 describes exercises using the same file.

*Autocommit*

There are two modes used for committing transactions in MySQL, Autocommit on, which is the default, or Autocommit off . When Autocommit is on, every individual SQL command is considered a transaction, and is committed immediately and ended. When Autocommit is off, a new transaction begins with a START TRANSACTION statement or with the first executable statement of the session and normally ends with a COMMIT or ROLLBACK. However the user can also explicitly begin a multi-statement transaction with a START TRANSACTION statement, and end it with a COMMIT or ROLLBACK in either mode. A START TRANSACTION statement or any DDL command also ends and commits any previous transaction. If the user is disconnected from a session during a transaction, that transaction is rolled back and ended. To turn Autocommit off , write

SET AUTOCOMMIT=0;

Setting the value back to 1 turns Autocommit on.

*Commit, Rollback, and Savepoints*

Like Oracle and standard SQL, MySQL uses START TRANSACTION to explicitly begin a transaction, and either COMMIT or ROLLBACK to explicitly end a transaction, with the usual meaning of those terms. It also permits the user to identify savepoints at desired points in a transaction, by writing a statement like

SAVEPOINT POINTA;

Later in the transaction, the user can roll back to the named savepoint by writing a statement like

ROLLBACK TO POINTA;

The savepoint can be released by writing

RELEASE SAVEPOINT POINTA;

*Concurrency Control in MySQL*

MySQL uses locks and multi-versioning for concurrency control. The isolation level for transactions can be specified using

SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED|READ COMMITTED|REPEATABLE READ|SERIALIZABLE};

The GLOBAL option sets the level for all sessions after the current one. The SESSION option refers to all later transactions in the current session. If neither is specified, the level is for the next transaction in the current session.

The READ UNCOMMITTED level uses no locks, but can result in reading dirty data, since it can read data from transactions that may later roll back. The READ COMMITTED makes uses of the multi-versioning feature, but uses no locks. Each read in a transaction gets a snapshot showing changes made by transactions committed before the read, plus its own changes, which it sees even though they are not yet committed. This can result in non-repeatable reads, since the snapshots may show changes from transactions that commit while the current transaction is in progress. The default level is REPEATABLE READ, which guarantees that if a transaction reads a value from the database more than once, it will always see the same value. This is done by using only a snapshot performed at the first read, so that all subsequent reads see data from that snapshot. Although these levels do not use locking, the user can force shared locking for a SELECT statement by using the a query of the form

SELECT \* FROM *tablename* LOCK IN SHARE MODE;

Or

SELECT \* FROM *tablename* FOR UPDATE;

For true serializability, the level should be set at serializable, which has the usual meaning. It uses shared locks for all reads, and exclusive locks for updates. Conflicting transactions must wait for locks to be released. Deadlock can result. MySQL uses a timeout to control how long a transaction waits for a lock. If the wait time is exceeded, the transaction is rolled back and ended.