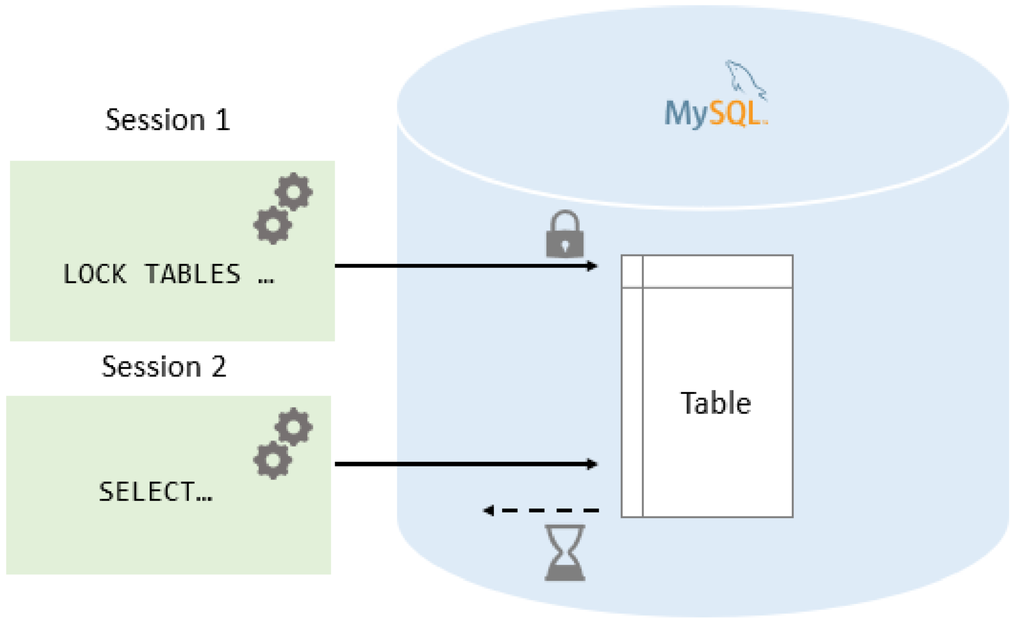
MySQLTUTORIAL

MySQL Table Locking

Summary: in this tutorial, you will learn how to use MySQL locking for cooperating table accesses between sessions.

A lock is a flag associated with a table. MySQL allows a client session to explicitly acquire a table lock for preventing other sessions from accessing the same table during a specific period.

A client session can acquire or release table locks only for itself. And a client session cannot acquire or release table locks for other client sessions.



Before we move on, let's create a table (https://www.mysqltutorial.org/mysql-create-table/) named `messages` for practicing with the table locking statements.

```
CREATE TABLE messages (
    id INT NOT NULL AUTO_INCREMENT,
    message VARCHAR(100) NOT NULL,
    PRIMARY KEY (id)
);
```

MySQL LOCK TABLES statement

The following `LOCK TABLES` statement explicitly acquires a table lock:

```
LOCK TABLES table_name [READ | WRITE]
```

In this syntax, you specify the name of the table that you want to lock after the `LOCK TABLES` keywords. In addition, you specify the type of lock, either `READ` or `WRITE`.

MySQL allows you to lock multiple tables by specifying a list of comma-separated table names with lock types that you want to lock after the `LOCK TABLES` keywords:

```
LOCK TABLES table_name1 [READ | WRITE],
            table_name2 [READ | WRITE],
             ... ;
```

MySQL UNLOCK TABLES statement

To release a lock for a table, you use the following `UNLOCK TABLES` statement:

```
UNLOCK TABLES;
```

READ Locks

A `READ` lock has the following features:

- A `READ` lock for a table can be acquired by multiple sessions at the same time. In addition, other sessions can read data from the table without acquiring the lock.
- The session that holds the `READ` lock can only read data from the table, but cannot write. And other sessions cannot write data to the table until the `READ` lock is released. The write operations from another session will be put into the waiting states until the `READ` lock is released.
- If the session is terminated, either normally or abnormally, MySQL will release all the locks implicitly. This feature is also relevant for the `WRITE` lock.

Let’s take a look at how the `READ` lock works in the following scenario.

First, connect to the database in the first session and use the `CONNECTION_ID()` function to get the current connection id as follows:

```
SELECT CONNECTION_ID();
```

Then, insert a new row (https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the `messages` table.

```
INSERT INTO messages(message)
VALUES('Hello');
```

Next, query the data (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the `messages` table.

```
SELECT * FROM messages;
```

After that, acquire a lock using the `LOCK TABLE` statement.

```
LOCK TABLE messages READ;
```

Finally, try to insert a new row into the `messages` table:

```
INSERT INTO messages(message)
VALUES('Hi');
```

MySQL issued the following error:

```
Error Code: 1099. Table 'messages' was locked with a READ lock and can't be updated.
```

So once the `READ` lock is acquired, you cannot write data to the table within the same session.

Let's check the `READ` lock from a different session.

First, connect to the database and check the connection id:

```
SELECT CONNECTION_ID();
```

Next, query data from the `messages` table:

```
SELECT * FROM messages;
```

Then, insert a new row (https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the `messages` table:

```
INSERT INTO messages(message)
VALUES('Bye');
```

Here is the output:

The insert operation from the second session is in the waiting state because a READ lock is already acquired on the messages table by the first session and it has not been released yet.

From the first session, use the `SHOW PROCESSLIST (https://www.mysqltutorial.org/mysql-show-processlist/)` statement to show detailed information:

```
SHOW PROCESSLIST;
```

After that, go back to the first session and release the lock by using the `UNLOCK TABLES` statement. After you release the `READ` lock from the first session, the `INSERT` operation in the second session is executed.

Finally, check the data of the `messages` table to see if the `INSERT` operation from the second session really executed.

```
SELECT * FROM messages;
```

Write Locks

A `WRITE` lock has the following features:

- The only session that holds the lock of a table can read and write data from the table.
- Other sessions cannot read data from and write data to the table until the `WRITE` lock is released.

Let's go into detail to see how the `WRITE` lock works.

First, acquire a `WRITE` lock from the first session.

```
LOCK TABLE messages WRITE;
```

Then, insert a new row into the `messages` table.

```
INSERT INTO messages(message)
VALUES('Good Morning');
```

It worked.

Next, query data from the `messages` table.

```
SELECT * FROM messages;
```

It also works.

After that, from the second session, attempt to write and read data:

```
INSERT INTO messages(message)
VALUES('Bye Bye');

SELECT * FROM messages;
```

MySQL puts these operations into a waiting state. You can check it using the `SHOW PROCESSLIST` statement:

```
SHOW PROCESSLIST;
```

Finally, release the lock from the first session.

```
UNLOCK TABLES;
```

You will see all pending operations from the second session executed and the following picture illustrates the result:

Read vs. Write locks

- Read locks are “shared” locks that prevent a write lock is being acquired but not other read locks.
- Write locks are “exclusive ” locks that prevent any other lock of any kind.

In this tutorial, you have learned how to lock and unlock tables for cooperating with the table accesses between sessions.