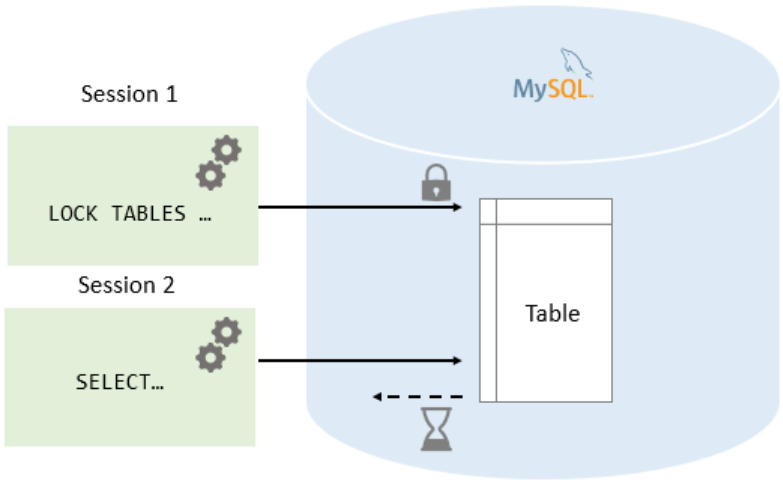
MySQL Table Locking

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 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

```
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 names of tables 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 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.

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

```
SELECT CONNECTION_ID();
```

CONNECTION_ID()
9

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

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

Next, query the data 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 update
```

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();
```

connection_id()
11

Next, query data from the `messages` table:

```
SELECT * FROM messages;
```

id	message
1	Hello

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

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

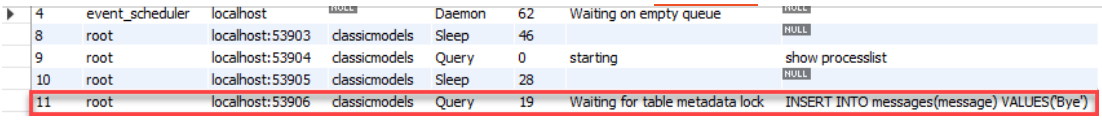
Here is the output:

#	Time	Action	Message
1	18:00:53	INSERT INTO messages(message) VALUES('Bye')	Running...

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 released yet.

From the first session, use the `SHOW PROCESSLIST` statement to show detailed information:

```
```



4	event_scheduler	localhost	NULL	Daemon	62	Waiting on empty queue	NULL
8	root	localhost:53903	classicmodels	Sleep	46		NULL
9	root	localhost:53904	classicmodels	Query	0	starting	show processlist
10	root	localhost:53905	classicmodels	Sleep	28		NULL
11	root	localhost:53906	classicmodels	Query	19	Waiting for table metadata lock	INSERT INTO messages(message) VALUES('Bye')

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 executed.

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

```
1 SELECT * FROM messages;
```

id	message
1	Hello
2	Bye

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.

```
1 LOCK TABLE messages WRITE;
```

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

```
1 INSERT INTO messages(message)
2 VALUES('Good Moring');
```

It worked.

Next, query data from the `messages` table.

```
1 SELECT * FROM messages;
```

id	message
1	Hello
2	Bye
3	Good Moring

It also works.

```
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;
```

Id	User	Host	db	Command	Time	State	Info
4	event_scheduler	localhost	NULL	Daemon	495	Waiting on empty queue	NULL
8	root	localhost:53903	classicmodels	Sleep	27		NULL
9	root	localhost:53904	classicmodels	Query	0	starting	show processlist
10	root	localhost:53905	classicmodels	Sleep	461		NULL
12	root	localhost:54000	classicmodels	Query	10	Waiting for table metadata lock	INSERT INTO messages(message...

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:

id	message
1	Hello
2	Bye
3	Good Moring
4	Good Night

Read vs. Write locks

> Read locks are “shared” locks which 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.

Related Tutorials

MySQL Transaction

« Previous Tutorial: MySQL Transaction

Next Tutorial: MySQL TRUNCATE TABLE »

Search this website

MYSQL QUICK START

What Is MySQL?

Install MySQL Database Server

Download MySQL Sample Database

Load Sample Database

MYSQL DATA MANIPULATION

SELECT

ORDER BY

WHERE

SELECT DISTINCT

AND

OR

IN

BETWEEN

LIKE

LIMIT

IS NULL

Table & Column Aliases

Joins

INNER JOIN

LEFT JOIN

RIGHT JOIN

Self Join

CROSS JOIN

GROUP BY

HAVING

ROLLUP

Subquery

Derived Tables

EXISTS

UNION

MYSQL DATA DEFINITION

MYSQL DATA TYPES

CHAR

DATE

DATETIME

DECIMAL

ENUM

INT

JSON

TIME

TIMESTAMP

VARCHAR

MYSQL GLOBALIZATION

MySQL Character Set

MySQL Collation

MYSQL IMPORT & EXPORT

Import a CSV File Into a Table

Export a Table to a CSV File

MYSQL PROGRAMMING INTERFACES

PHP MySQL Tutorial

Node.js MySQL Tutorial

Python MySQL Tutorial

Perl MySQL Tutorial

MySQL JDBC Tutorial

OTHER TUTORIALS

MySQL Administration

MySQL Full-Text Search

MySQL Cheat Sheet

MySQL Books and Video Training

MySQL Hosting

RECENT MYSQL TUTORIALS

MySQL SHOW GRANTS

Listing Stored Functions

MySQL DROP FUNCTION

MySQL REPEAT Loop

MySQL LEAVE

MySQL WHILE Loop

How To Call a Stored Procedure From a Trigger in MySQL

MySQL AFTER DELETE Trigger

MySQL BEFORE DELETE Trigger

MySQL AFTER INSERT Trigger

ABOUT MYSQL TUTORIAL WEBSITE

database administrators learn MySQL faster and more effectively.

All MySQL tutorials are practical and easy-to-follow, with SQL script and screenshots available. More About Us

SITE LINKS

About Us

Contact Us

Request a Tutorial

Privacy Policy