MySQL Table Locking

A lock is a mechanism associated with a table used to restrict the unauthorized access of the data in a table. **MySQL allows a client session to acquire a table lock explicitly to cooperate with other sessions to access the table's data**. MySQL also allows table locking to prevent it from unauthorized modification into the same table during a specific period.

A session in MySQL can acquire or release locks on the table only for itself. Therefore, one session cannot acquire or release table locks for other sessions. It is to note that we must have a TABLE LOCK and SELECT privileges for table locking.

Table Locking in MySQL is mainly **used to solve concurrency problems**. It will be used while running a transaction, i.e., first read a value from a table (database) and then write it into the table (database).

[MySQL](https://www.javatpoint.com/mysql-tutorial) provides **two types of locks** onto the table, which are:

**READ LOCK:** This lock allows a user to only read the data from a table.

**WRITE LOCK:** This lock allows a user to do both reading and writing into a table.

It is to note that the default storage engine used in MySQL is InnoDB. The InnoDB storage engine does not require table locking manually because MySQL automatically uses row-level locking for InnoDB tables. Therefore, we can do multiple transactions on the same table simultaneously to read and write operations without making each other wait. All other storage engines use table locking in MySQL.

Before understanding the table locking concept, first, we will create a new table named "**info\_table**" using the statement as follows:

**CREATE** **TABLE** info\_table (

    Id **INT** NOT NULL AUTO\_INCREMENT,

**Name** **VARCHAR**(50) NOT NULL,

    Message **VARCHAR**(80) NOT NULL,

**PRIMARY** **KEY** (Id)

);

MySQL LOCK TABLES Statement

The following is the syntax that allows us to acquire a table lock explicitly:

1. LOCK TABLES table\_name [**READ** | WRITE];

In the above syntax, we have specified the **table name** on which we want to acquire a lock after the **LOCK TABLES** keywords. We can specify the **lock type**, either READ or WRITE.

We can also lock more than one table in MySQL by using a list of comma-separated table's names with lock types. See the below syntax:

1. LOCK TABLES tab\_name1 [**READ** | WRITE],
2. tab\_name2 [**READ** | WRITE],...... ;

MySQL UNLOCK TABLES Statement

The following is the syntax that allows us **to release a lock** for a table in MySQL:

1. mysql> UNLOCK TABLES;

LOCK TYPES

Let us understand the lock types in detail.

READ Locks

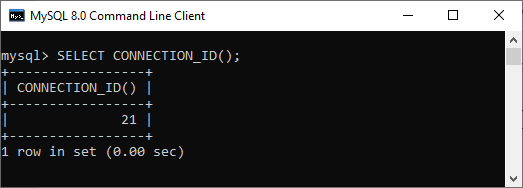
The following are the features of the READ lock:

* At the same time, MySQL allows multiple sessions to acquire a READ lock for a table. And all other sessions can read the table without acquiring the lock.
* If the session holds the READ lock on a table, they cannot perform a write operation on it. It is because the READ lock can only read data from the table. All other sessions that do not acquire a READ lock are not able to write data into the table without releasing the READ lock. The write operations go into the waiting states until we have not released the READ lock.
* When the session is terminated normally or abnormally, MySQL implicitly releases all types of locks on to the table. This feature is also relevant for the WRITE lock.

Let us take an example to see how READ locks work in MySQL with the given scenario. We will first connect to the database and use the **CONNECTION\_ID()** function that gives the current connection id in the first session as follows:

1. mysql> **SELECT** CONNECTION\_ID();

See the below output:



Next, we will insert few rows into the **info\_table** using the below statement:

mysql> **INSERT** **INTO** info\_table (**name**, message)

**VALUES**('Peter', 'Hi'),

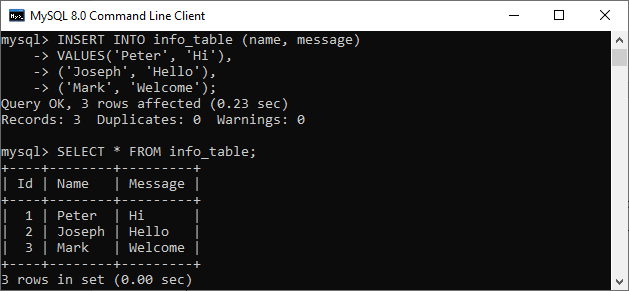
('Joseph', 'Hello'),

('Mark', 'Welcome');

Now, verify the data into the table using the below statement:

1. mysql> **SELECT** \* **FROM** info\_table;

We should see the output as follows:



Now, we will execute the LOCK TABLE statement to acquire a lock onto the table:

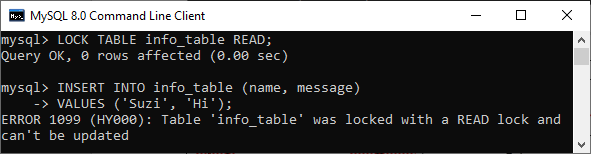
1. mysql> LOCK **TABLE** info\_table **READ**;

After that, we will try to insert a new record into the info\_table as follows:

mysql> **INSERT** **INTO** info\_table (**name**, message)

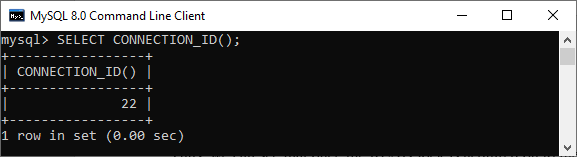
**VALUES** ('Suzi', 'Hi');

We will get the below output where MySQL issues the following message **"Table 'info\_table' was locked with a READ lock and can't be updated"**.

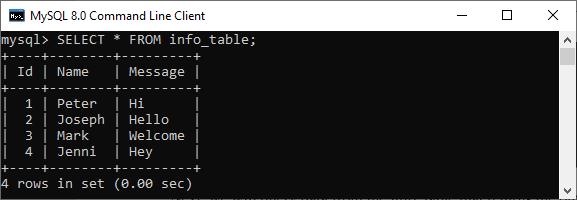


Thus, we can see that once the READ lock is acquired on to the table, we cannot write data to the table in the same session.

Now, we will check how the READ lock work from a different session. First, we will connect to the database and see the connection id:



Next, we will query data from the info\_table that returns the output as follows:



Then, insert some rows into this table as follows:

mysql> **INSERT** **INTO** info\_table (**name**, message)

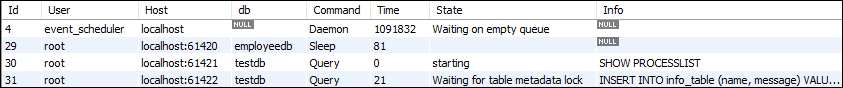
**VALUES** ('Stephen', 'Hello');

We should see the output as follows:

MySQL Table Locking

In the above output, we can see that the insert operation from the second session is in the **waiting state**. It is due to the READ lock, which is already acquired on the table by the first session and has not been released yet.

We can see the detailed information about them using the **SHOW PROCESSLIST** statement in the first session. See the below output:



Finally, we need to release the lock by using the **UNLOCK TABLES** statement in the first session. Now, we are able to execute the INSERT operation in the second session.

Write Locks

The following are the features of a WRITE lock:

* It is the session that holds the lock of a table and can read and write data both from the table.
* It is the only session that accesses the table by holding a lock. And all other sessions cannot access the data of the table until the WRITE lock is released.

Let us take an example to see how WRITE locks works in MySQL with the given scenario. In the first session, we will acquire a WRITE lock using the below statement:

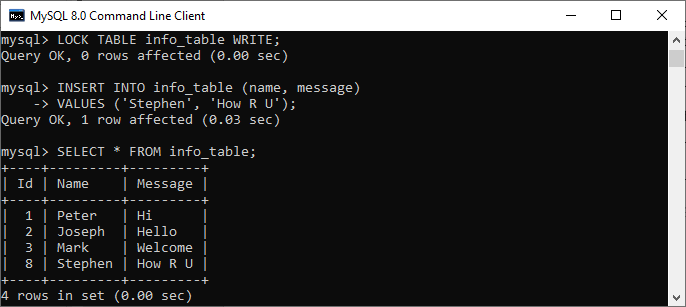
1. mysql> LOCK **TABLE** info\_table WRITE;

Then, we will insert a new record into the info\_table as follows:

mysql> **INSERT** **INTO** info\_table (**name**, message)

**VALUES** ('Stephen', 'How R U');

The above statement worked. Now, we can verify the output using the SELECT statement:



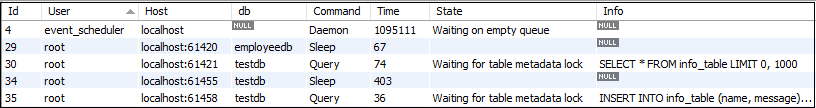
Again, we will attempt to access (read/write) the table from the second session:

**INSERT** **INTO** info\_table (**name**, message)

**VALUES** ('George', 'Welcome');

2. **SELECT** \* **FROM** info\_table;

We can see that these operations are put into a waiting state. See the detailed information about them using the SHOW PROCESSLIST statement:



Finally, we will release the lock from the first session. Now, we can execute the pending operations.

Read vs. Write Lock

* Read lock is similar to "**shared**" locks because multiple threads can acquire it at the same time.
* Write lock is an "**exclusive**" locks because another thread cannot read it.
* We cannot provide read and write locks both on the table at the same time.
* Read lock has a **low priority** than Write lock, which ensures that updates are made as soon as possible.