In a database transaction, there are typically two main types of locks:

1. shared lock

2. Exclusive lock

The duration of these locks depends on the specific operation and the

isolation level used by the transaction.

Shared Lock (S-Lock):

Duration: Shared locks are typically short-lived. They are acquired

when a transaction wants to read a resource (e.g., a row or a table)

and doesn't intend to modify it. Shared locks allow multiple

transactions to read the same resource simultaneously,

ensuring data consistency.

Compatibility: Multiple transactions can hold shared locks

on the same resource concurrently.

Example: When multiple users read the same row of a table

simultaneously, each transaction acquires a shared lock.

Exclusive Lock (X-Lock):

Duration: Exclusive locks are generally held for a shorter duration

as well. They are acquired when a transaction intends to modify a

resource

and needs to ensure that no other transactions can access or

modify it concurrently.

Compatibility: Exclusive locks are exclusive, meaning that only

one transaction can hold an exclusive lock on a resource at a time.

This ensures data integrity during updates.

Example: When a transaction wants to update a row in a table,

it acquires an exclusive lock on that row until the update is completed.

The actual duration of locks can vary depending on several factors:

Transaction Isolation Level: The isolation level of the transaction

(e.g., READ COMMITTED, REPEATABLE READ, SERIALIZABLE) affects

how locks are acquired and released. Higher isolation levels tend

to hold locks for longer periods to ensure data consistency.

Transaction Commit or Rollback: Locks acquired by a transaction are

typically released when the transaction is committed (successfully)

or rolled back (aborted). Therefore, the duration of locks depends on

when the transaction reaches one of these states.

Lock Timeout: Some database systems allow you to specify a lock timeout,

which determines how long a transaction can wait for a lock.

If a lock cannot be acquired within the specified timeout,

the transaction may fail or be rolled back.

Database System Implementation: The behavior of locks may also be

influenced by the specific database system you are using,

as different database systems may have slightly different locking

mechanisms and behaviors.

It's important to note that while shared and exclusive locks are common,

databases may have other types of locks as well,

depending on their design and features. For example, row-level locks,

page-level locks, and table-level locks are variations that

some databases support.

The specific behavior and duration of locks should be documented

in the database system's documentation and can vary between

database systems and versions.

############################################################################

The two primary internal tables that store information about

transaction locks in InnoDB are:

1. performance\_schema.data\_locks

2. performance\_schema.metadata\_locks

The performance\_schema.data\_locks table provides information

about locks on data pages, while the performance\_schema.metadata\_locks table

provides information about locks on metadata pages.

> SELECT \* FROM performance\_schema.data\_locks;

> SELECT \* FROM performance\_schema.metadata\_locks;

> SELECT \* FROM performance\_schema.data\_locks WHERE object\_name = 'my\_table';

You can also use the SHOW ENGINE INNODB STATUS statement to get

information about all locks held by InnoDB

> SHOW ENGINE INNODB STATUS;

############################################################################

To get all the tables in use and the tables which are locked we use

following command

It shows db name, table name, In\_use, Name\_locked

So here In\_use col shows table is locked if value is greater than 0

> show open tables;

############################################################################

> show engines;

Gives available storage engine on mysql server

############################################################################

> show open tables where In\_use > 1;

Gives all tables which are open in mysql table cache and gives locks on

tables by In\_use column whose value shows number of locks on table

############################################################################

> show processlist;

statement displays a list of all of the threads that are currently

running on your MySQL server. The output includes the following

information for each thread:

ID: The identifier for the thread.

USER: The user who started the thread.

HOST: The host from which the thread connected.

db: The database that the thread is using.

Command: The command that the thread is executing (if any).

Time: The amount of time that the thread has been executing.

State: The current state of the thread.

############################################################################

Below sql query will return a list of all locks that are currently

being held by InnoDB. The following is an explanation of each column

in the output:

Trx\_Id: The identifier of the transaction that is holding the lock.

Table: The name of the table that is locked.

Index: The name of the index that is locked (if any).

Data: The specific data that is locked (e.g., a row, a page).

Mode: The mode of the lock (e.g., X, S).

Status: The status of the lock (e.g., GRANTED, WAITING).

Type: The type of the lock (e.g., TABLE, ROW).

> SELECT ENGINE\_TRANSACTION\_ID as Trx\_Id,

OBJECT\_NAME as `Table`,

INDEX\_NAME as `Index`,

LOCK\_DATA as Data,

LOCK\_MODE as Mode,

LOCK\_STATUS as Status,

LOCK\_TYPE as Type

FROM performance\_schema.data\_locks;

############################################################################

> SELECT \* FROM information\_schema.SCHEMATA;

To get all the db available on your mysql server

############################################################################

> SELECT \* FROM information\_schema.INNODB\_TRX;

This statement will return a list of all of the transactions that are currently

executing inside InnoDB. The trx\_started column in this table shows the

time at which the transaction started

############################################################################

> SELECT

trx\_id,

trx\_state,

trx\_started,

NOW() - trx\_started AS lock\_time

FROM information\_schema.INNODB\_TRX

WHERE trx\_state IN ('ACTIVE', 'WAITING');

Usung this we can get total amount of time from which the transaction

is active or waiting

############################################################################

> SELECT \* FROM information\_schema.PROCESSLIST;

This statement will return a list of all of the threads that are currently

running on your MySQL server. The output includes the following

information for each thread:

ID: The identifier for the thread.

USER: The user who started the thread.

HOST: The host from which the thread connected.

db: The database that the thread is using.

Command: The command that the thread is executing (if any).

Time: The amount of time that the thread has been executing.

State: The current state of the thread.

############################################################################

This statement can be used to monitor the time required to complete various

sql statements, like in below example we are monitoring insert statements

SELECT

EVENT\_ID,

SQL\_TEXT,

TIMER\_START,

TIMER\_END,

TIMER\_WAIT

FROM performance\_schema.events\_statements\_history

WHERE SQL\_TEXT LIKE '%insert%';

############################################################################