K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch: A3 Roll No.: 1911034

Experiment / assignment / tutorial No. 10

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Implementation of Concurrency Control Protocols

Objective: To understand Transaction, Transaction Control Protocols and its implementation.
Implement Lock based protocol.

Expected Outcome of Experiment:
CO 5: Formulate and demonstrate the transaction, concurrency and recovery techniques

Books/ Journals/ Websites referred:

1. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
2. www.db-book.com
3. Korth, Slberchatz, Sudarshan : "Database Systems Concept", 5th Edition , McGraw Hill
4. Elmasri and Navathe,"Fundamentals of database Systems", 4th Edition,PEARSON Education.
5. https://dev.mysql.com/doc/refman/8.0/en/innodb-transaction-isolation-levels.html

Resources used:

Theory

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions.

Concurrency control is provided in a database to:

i. enforce isolation among transactions.

ii. preserve database consistency through consistency preserving execution of transactions.
iii. resolve read-write and write-read conflicts.

Concurrency control protocols can be broadly divided into two categories −

1. Lock based protocols
2. Time stamp based protocols

Lock-based Protocols

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds −

1. Binary Locks − A lock on a data item can be in two states; it is either locked or unlocked.
2. Shared/exclusive − This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

Timestamp Ordering Protocol

The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

The timestamp of transaction Ti is denoted as TS(Ti).
Read time-stamp of data-item X is denoted by R-timestamp(X).
Write time-stamp of data-item X is denoted by W-timestamp(X).

Timestamp ordering protocol works as follows −

```
If a transaction Ti issues a read(X) operation −
    If TS(Ti) < W-timestamp(X)
        Operation rejected.
    If TS(Ti) >= W-timestamp(X)
        Operation executed.
    All data-item timestamps updated.
```

```
If a transaction Ti issues a write(X) operation −
    If TS(Ti) < R-timestamp(X)
        Operation rejected.
    If TS(Ti) < W-timestamp(X)
        Operation rejected and Ti rolled back.
    Otherwise, operation executed.
```

Implementation of Lock Protocol:

First we create a schema ‘example’ and create a table users as follows:

```
use example;

create table user_data

(

id int not null auto_increment,

user_name varchar (40),

primary key (id)

);
```

Then we insert three values into the table:

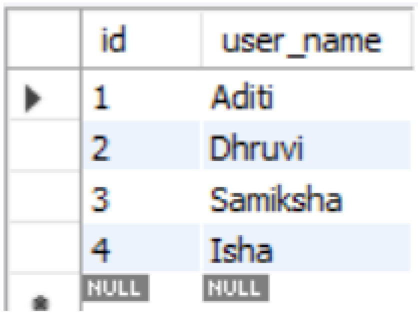
```
insert into user_data values (1,"Aditi");

insert into user_data values (2,"Dhruvi");

insert into user_data values (3,"Samiksha");

insert into user_data values (4,"Isha");
```

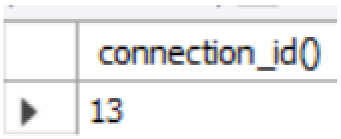
Showing the data of the table after inserting 4 values



id	user_name
1	Aditi
2	Dhruvi
3	Samiksha
4	Isha
NULL	NULL

Showing the current connection id

select connection_id()



connection_id()
13

Now we issue a ‘READ’ lock on the table which will not allow any connection to write the table

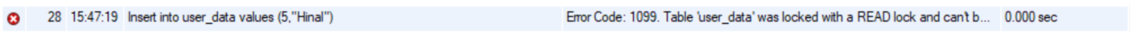
lock table user_data READ;

After acquiring a READ lock we try to insert a value into the table

use example;

Insert into user_data values (5,"Hinal");

Error Message gets displayed as follows:

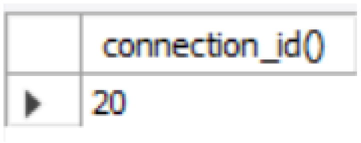


```
28  15:47:19  Insert into user_data values (5,"Hinal")    Error Code: 1099. Table 'user_data' was locked with a READ lock and can't b...  0.000 sec
```

Full text of Error message :

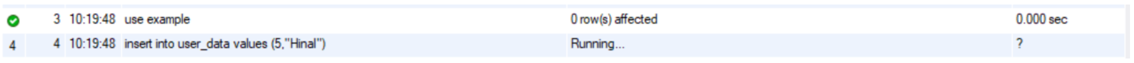
Error Code: 1099. Table 'user_data' was locked with a READ lock and can't be updated

Now we try to login through a different connection and show the connection id:



	connection_id()
▶	20

Now we try to insert (WRITE) a value into the same table when a READ lock is acquired by another connection;



```
✓    3  10:19:48  use example                              0 row(s) affected    0.000 sec
4    4  10:19:48  insert into user_data values (5,"Hinal")  Running...           ?
```

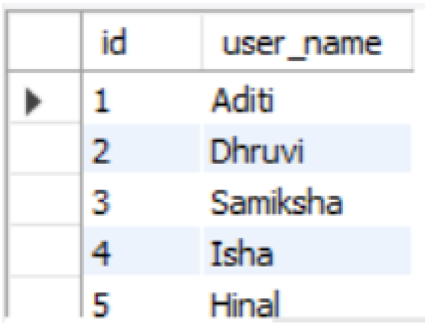
The connection shows a 'Running' State .

Now we unlock the table from the first connection

unlock tables;

commit;

Displaying the values of the first table after unlocking:



	id	user_name
▶	1	Aditi
	2	Dhruvi
	3	Samiksha
	4	Isha
	5	Hinal

The 5th value has been inserted as the tables are unlocked.

WRITE LOCKS ON TABLES

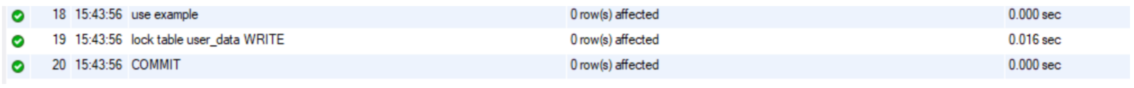
First we issue a WRITE lock on a table from one connection

use example;

lock table user_data WRITE;

COMMIT;

Output



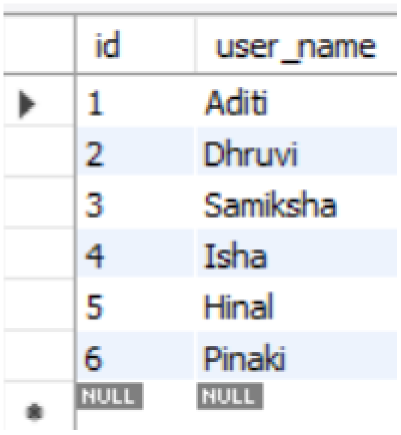
```
18  15:43:56  use example                   0 row(s) affected    0.000 sec
19  15:43:56  lock table user_data WRITE    0 row(s) affected    0.016 sec
20  15:43:56  COMMIT                        0 row(s) affected    0.000 sec
```

Inserting data into the table from the same connection:

use example;

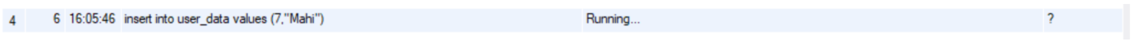
Insert into user_data values (6,"Pinaki")

This value has been successfully inserted as can be seen below:



id	user_name
1	Aditi
2	Dhruvi
3	Samiksha
4	Isha
5	Hinal
6	Pinaki
NULL	NULL

Now we try to insert data into the same table from a different connection when WRITE lock has been issued-

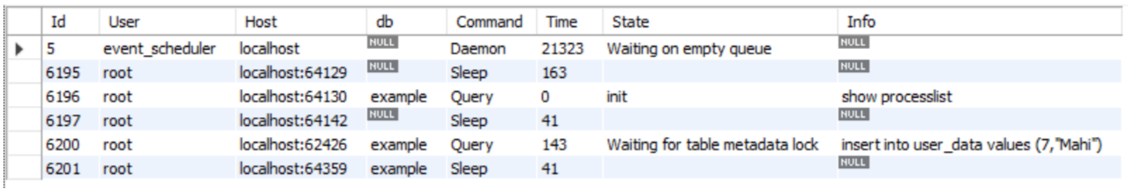


```
4    6  16:05:46  insert into user_data values (7,"Mahi")    Running...    ?
```

Our request has been put into a waiting state.

We can see the process list from the first connection:

Using show processlist;

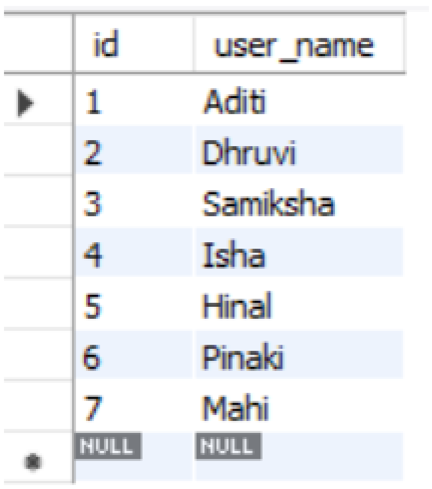


Id	User	Host	db	Command	Time	State	Info
5	event_scheduler	localhost	NULL	Daemon	21323	Waiting on empty queue	NULL
6195	root	localhost:64129	NULL	Sleep	163		NULL
6196	root	localhost:64130	example	Query	0	init	show processlist
6197	root	localhost:64142	NULL	Sleep	41		NULL
6200	root	localhost:62426	example	Query	143	Waiting for table metadata lock	insert into user_data values (7,"Mahi")
6201	root	localhost:64359	example	Sleep	41		NULL

Now we unlock the table from the first session using unlock tables;

```
use example;

select * from user_data;
```



id	user_name
1	Aditi
2	Dhruvi
3	Samiksha
4	Isha
5	Hinal
6	Pinaki
7	Mahi
NULL	NULL

As we unlocked the tables the 7th entry was inserted

Post Lab Questions:

1. **Explain pitfalls of 2PL (Two Phase Locking) Protocol**

1.Two-phase locking may also limit the amount of concurrency that occur in a schedule because a Transaction may not be able to release an item after it has used it.

2. This may be because of the protocols and other restrictions we may put on the schedule to ensure serializability, deadlock freedom and other factors.

3.This is the price we have to pay to ensure serializability and other factors, hence it can be considered as a bargain between concurrency and maintaining the ACID properties**.**

Conclusion (In your own words): In this program we were successfully able to understand the Lock based protocols and implement it on our database using different connections.