Department of Information Technology

Academic Year: 2022-23
Semester: III
Class / Branch: SE (IT)
Subject: SQL Lab
Name of Instructor: Prof. Charul Singh

Name of Student: Harsh Joshi
Student ID: 22204012
Date of Performance: 19/12/2022
Date of Submission: 19/12/2022

Experiment No:7

Aim: To implement TCL commands and concurrency control techniques using locks

Software used: MySQL

Theory :

Transaction control language (TCL) commands are used to manage transactions in database. These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions**.**

Commit command

Commit command is used to permanently save any transaction into database. Following is Commit command's
Syntax: *commit***;**

Rollback command

This command restores the database to last commited state. It is also use with savepoint command to jump to a savepoint in a transaction.
Following is Rollback command's syntax:
rollback **to *savepoint-name*;**

Savepoint command

Savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.
Following is savepoint command's syntax:
savepoint *savepoint-name*;

Example of Savepoint and Rollback

ID	NAME

1	abhi
2	adam
4	alex

Lets use some SQL queries on the above table and see the results.

```
INSERT into class
values(5,'Rahul'); commit;

UPDATE class set name='abhijit' where
id='5'; savepoint A;

INSERT into class
values(6,'Chris'); savepoint B;

INSERT into class
values(7,'Bravo'); savepoint C;

SELECT * from class;
```

The resultant table will look like,

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit
6	chris
7	bravo

Now **rollback** to **savepoint B**

```
rollback to B;
SELECT * from
class;
```

The resultant table will look like

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit

6	chris

Now **rollback** to **savepoint A**

rollback to A;
SELECT * from
class;

The result table will look like

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit

Transaction

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

Open_Account(A)

Old_Balance = A.balance

New_Balance = Old_Balance - 500

A.balance = New_Balance

Close_Account(A)

B's Account

Open_Account(B)

Old_Balance = B.balance

New_Balance = Old_Balance + 500

B.balance = New_Balance

Close_Account(B)

Concurrency Control

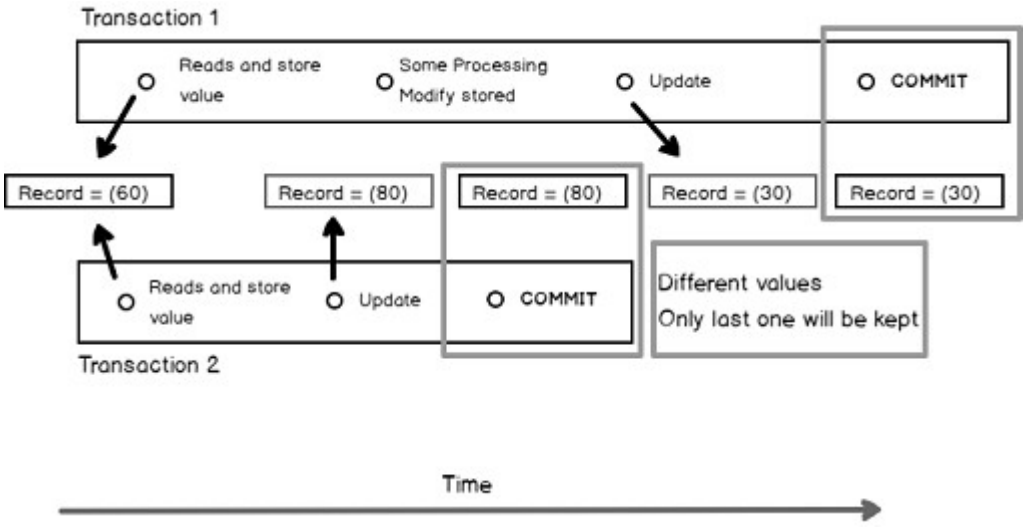
Before diving into transaction levels details, it's important to get used to typical concurrency problems and how we call them.

Lost update and dirty write

This phenomenon happens when two transactions access the same record and both updates this record. The following figure summarizes what could happen in a simple example.

In this example, we have 2 concurrent transactions that access a record with a (60) modifiable value. This record is identified either by its rowId or by a primary key column that won't be presented here for simplicity.

The first transaction reads this record, does some processing then updates this record and finally commits its work. The second transaction reads the record then updates it immediately and commits. Both transactions do not update this record to the same value. This leads to a loss for the update statement performed by second transaction.



As Transaction 1 **overwrites a value that** Transaction 2 **already modified. We could have said that** Transaction 1 **did a « dirty write » if** Transaction 2 **didn't commit its work.**

Output:

```
mysql> Commit;
Query OK, 0 rows affected (0.04 sec)

mysql> Update student set s_name="Maithili" where s_id=1;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> insert into student values(6,"Kashish");
Query OK, 1 row affected (0.04 sec)

mysql> select*from student;
+------+----------+
| s_id | s_name   |
+------+----------+
|    1 | Maithili |
|    2 | Tanvi    |
|    3 | Shreya   |
|    4 | Sanjana  |
|    5 | Aditya   |
|    6 | Kashish  |
+------+----------+
6 rows in set (0.00 sec)
```

```
mysql> Start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> Update student set s_name="Rahul" where s_id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> insert into student values(7,"Sahil");
Query OK, 1 row affected (0.00 sec)

mysql> rollback
    -> ;
Query OK, 0 rows affected (0.04 sec)

mysql> select*from student;
+------+----------+
| s_id | s_name   |
+------+----------+
|    1 | Maithili |
|    2 | Tanvi    |
|    3 | Shreya   |
|    4 | Sanjana  |
|    5 | Aditya   |
|    6 | Kashish  |
+------+----------+
6 rows in set (0.00 sec)
```

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select*from student;
+------+---------+
| s_id | s_name  |
+------+---------+
|    2 | Tanvi   |
|    3 | Shreya  |
|    4 | Sanjana |
|    5 | Aditya  |
|    6 | Kashish |
|    7 | Sahil   |
+------+---------+
6 rows in set (0.01 sec)
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> savepoint initial;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into student values(8,"Aniket");
Query OK, 1 row affected (0.00 sec)

mysql> savepoint ins;
Query OK, 0 rows affected (0.00 sec)

mysql> update student set s_name="Sanskruti" where s_id=5;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> savepoint upd;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from student where s_id=7;
Query OK, 1 row affected (0.00 sec)

mysql> savepoint del;
Query OK, 0 rows affected (0.00 sec)

mysql> select*from student;
+------+-----------+
| s_id | s_name    |
+------+-----------+
|    2 | Tanvi     |
|    3 | Shreya    |
|    4 | Sanjana   |
|    5 | Sanskruti |
|    6 | Kashish   |
|    8 | Aniket    |
+------+-----------+
6 rows in set (0.00 sec)
```

```
mysql> rollback to upd;
Query OK, 0 rows affected (0.00 sec)

mysql> select*from student;
+------+------------+
| s_id | s_name     |
+------+------------+
|    2 | Tanvi      |
|    3 | Shreya     |
|    4 | Sanjana    |
|    5 | Sanskruti  |
|    6 | Kashish    |
|    7 | Sahil      |
|    8 | Aniket     |
+------+------------+
7 rows in set (0.00 sec)

mysql> rollback to ins;
Query OK, 0 rows affected (0.00 sec)

mysql> select*from student;
+------+----------+
| s_id | s_name   |
+------+----------+
|    2 | Tanvi    |
|    3 | Shreya   |
|    4 | Sanjana  |
|    5 | Aditya   |
|    6 | Kashish  |
|    7 | Sahil    |
|    8 | Aniket   |
+------+----------+
7 rows in set (0.00 sec)

mysql>
```

Conclusion: In this experiment we studied about TCL commands. TCL stands for Transaction Control Language. The TCL commands are used to perform transactions on database. COMMIT command is used to save the transactions made, ROLLBACK command is used to skip the DML commands and come back to the start of the database & SAVEPOINT command is used to save the transaction to rollback to the point where necessary.