SE Comp - B	Roll number :
Experiment no. : 10	Date of Implementation :

Aim: Simple Transaction implementation

Tool Used : MySql/PostgreSQL
https://www.javatpoint.com/mysql-transaction

Related Course outcome : At the end of the course, Students will be able to Use and Apply the concept of transaction, concurrency and recovery

Indicator	Poor	Average	Good
Timeliness • **Maintains assignment deadline (3)**	**Assignment not done (0)**	**One or More than One week late (1-2)**	**Maintains deadline (3)**
Implementation of concepts (3)	**N/A**	**< 80% complete (1-2)**	**100% complete (3)**
Originality • **Extent of plagiarism(2)**	**Copied it from someone else(0)**	**At least few parts of it have been done without copying(1)**	**Experiment has been solved completely without copying (2)**
Knowledge • **In depth knowledge of the assignment(2)**	**Unable to answer 2 questions(0)**	**Unable to answer 1 question (1)**	**Able to answer 2 questions (2)**

Rubrics for assessment of Experiment:

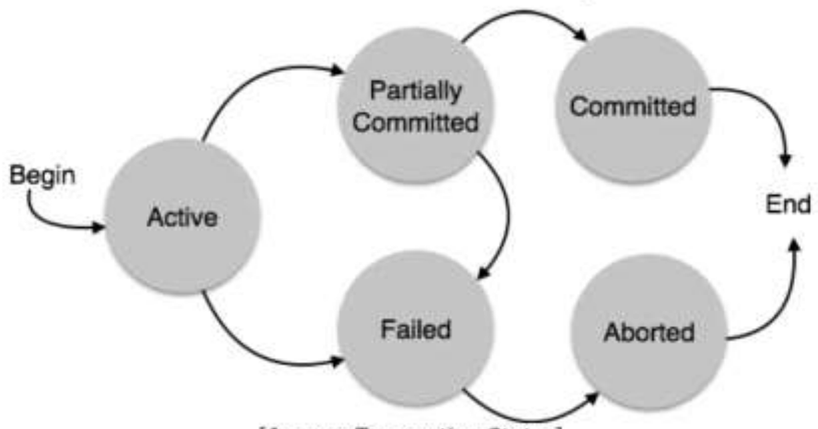
Assessment Marks :

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Total : (Out of 10)

Teacher's Sign :

EXPERIMENT 10	***Transaction concept***
Aim	To implement Simple Transaction concept
Tools	Mysql/PostgreSQL
Theory	A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further. *Transactions* are a fundamental concept of all database systems. The essential point of a transaction is that it bundles multiple steps into a single, all-or-nothing operation. The intermediate states between the steps are not visible to other concurrent transactions, and if some failure occurs that prevents the transaction from completing, then none of the steps affect the database at all. **Properties of Transactions** Transactions have the following four standard properties, usually referred to by the acronym ACID − • **Atomicity** − Ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure and previous operations are rolled back to their former state. • **Consistency** − Ensures that the database properly changes states upon a successfully committed transaction. • **Isolation** − Enables transactions to operate independently of and transparent to each other. • **Durability** − Ensures that the result or effect of a committed transaction persists in case of a system failure. In PostgreSQL, a transaction is set up by surrounding the SQL commands of the transaction with BEGIN and COMMIT commands. So our banking transaction would actually look like: BEGIN; UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice'; -- etc etc COMMIT; End;

Theory	**State Diagram :** A transaction in a database can be in one of the following states:  Begin Active Partially Committed Committed Failed Aborted End *[Image: Transaction States]* For example, consider a bank database that contains balances for various customer accounts, as well as total deposit balances for branches. Suppose that we want to record a payment of $100.00 from Alice's account to Bob's account. BEGIN; --sql SAVEPOINT my_savepoint; UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice'; UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Bob'; ROLLBACK TO my_savepoint; or commit; --UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Wally'; COMMIT;

Theory

Transaction Control (TCL)

The following commands are used to control transactions −

- **BEGIN TRANSACTION** − To start a transaction.
- **COMMIT** − To save the changes, alternatively you can use **END TRANSACTION** command.
- **ROLLBACK** − To rollback the changes.

Transactional control commands are only used with the DML commands INSERT, UPDATE and DELETE only. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

The BEGIN TRANSACTION Command

Transactions can be started using BEGIN TRANSACTION or simply BEGIN command. Such transactions usually persist until the next COMMIT or ROLLBACK command is encountered. But a transaction will also ROLLBACK if the database is closed or if an error occurs.

The following is the simple syntax to start a transaction –
BEGIN;

or

BEGIN TRANSACTION;

The COMMIT Command
The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.
The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.
The syntax for COMMIT command is as follows –
COMMIT;

or

END TRANSACTION;

Theory	**The ROLLBACK Command** The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued. The syntax for ROLLBACK command is as follows − ROLLBACK;
Task	**Task1: Perform following task** create table student with column (id, name) start transaction; Insert following records (1, ‘Amita’) (2, ‘Sheena’) (3, ‘Lavina’) (4, ‘Rex’) (5, “Rahul’) Update name of id 5 form ‘Rahul’ to ‘Abhijit’ Create a save point A; Insert new record (6, ‘chris’) Create a save point B; Insert new record (7, ‘Bravo’) Create a save point C; Display all rows of the students table (select * from students) Observe the output **Task 2: Rollback to save point B and observe the output** Perform task 2 and observe the output and explain the output **Task 3: Rollback to save point A and observe the output** Perform task 3 and observe the output and explain the output **Task 4: Now delete record of ‘Rex’, before delete create a save point , and rollback to this save point to undo this delete operation** Perform task 4 and observe the output and explain the output **Task 5: Now Perform commit** Perform task 5 and observe the output and explain the output
Links	**https://www.studytonight.com/dbms/tcl-command.php** **https://www.splessons.com/lesson/mysql-tcl/** **https://www.tutorialspoint.com/sql/sql-transactions.htm**

Post Lab Questions:	1. Explain set transaction command in SQL 2. Explain how do you remove a savepoint (checkpoint) that you have created?