Experiment No.10
Implementation and demonstration of Transaction and Concurrency control techniques using locks
Date of Performance:
Date of Submission:

Aim :- Write a query to lock and unlock a table for transaction and concurrency control.

Objective :- To learn locking of tables for transaction processing and concurrency control.

Theory:

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 unauthorized modification into the same table during a specific period.

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 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. The

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

TABLES table_name [READ | WRITE];

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

UNLOCK TABLES;

Implementation:

Code

farm_database

Limit to 1000 rows

```
133 •  SAVEPOINT my_savepoint;
134
135 •  ROLLBACK TO my_savepoint;
136
137 •  CREATE USER 'ankit'@'localhost';
138
139 •  DROP USER 'ankit'@'localhost';
140
141 •  GRANT SELECT, INSERT, UPDATE, DELETE ON FMS TO 'ankit'@'localhost';
142
143 •  GRANT SELECT, INSERT, UPDATE, DELETE ON FMS TO 'ankit'@'localhost';
144
145 •  CREATE VIEW Farmer_Farm_View AS
146    SELECT FMS.F_ID, FMS.F_NAME, FMS.L_NAME, FMS.AGE, FMS.MOB_NO, FMS.PLACE, FARM.farm_id, FARM.farm_size, FARM.farm_type, FARM.farm_location
147    FROM FMS
148    INNER JOIN FARM ON FMS.F_ID = FARM.farm_id;
149
150 •  LOCK TABLES FMS WRITE;
151
152 •  LOCK TABLES SUPPLIERS READ;
153
154 •  UNLOCK TABLES;
155
```

Output:

✔	40	10:48:24	LOCK TABLES FMS WRITE	0 row(s) affected
✔	41	10:48:30	LOCK TABLES SUPPLIERS READ	0 row(s) affected
✔	42	10:48:34	UNLOCK TABLES	0 row(s) affected

Conclusion: Locking and unlocking of tables is achieved and verified using insert command in the same table of a database system.

Explain Transaction and Concurrency control techniques using locks.

Ans.

Transaction:

A transaction is a logical unit of work that consists of one or more database operations (e.g., reads, writes, updates) executed as a single indivisible unit. Transactions ensure that database operations are either all completed successfully or none are completed at all, maintaining the consistency and integrity of the database.

The ACID properties (Atomicity, Consistency, Isolation, Durability) govern the behavior of transactions:

Atomicity: Ensures that all operations within a transaction are executed as a single indivisible unit. If any operation fails, the entire transaction is rolled back, leaving the database in its original state.

Consistency: Guarantees that the database remains in a consistent state before and after a transaction. It ensures that only valid data modifications are allowed, preserving data integrity and enforcing integrity constraints.

Isolation: Ensures that the concurrent execution of multiple transactions does not interfere with each other, providing each transaction with the illusion that it is executing alone on the database. Isolation levels control the degree of isolation between transactions.

Durability: Ensures that the changes made by a committed transaction persist even in the event of system failures. Committed transactions are permanently stored in the database and cannot be undone.

1. **Concurrency Control Techniques using Locks:**

Concurrency control ensures that multiple transactions can execute concurrently without interfering with each other, while still preserving the ACID properties of transactions. Locking is a common technique used for concurrency control, where transactions acquire and release locks on data items to control access and ensure consistency.

Types of Locks:

Shared (Read) Locks: Allow multiple transactions to read a data item simultaneously but prevent any transaction from writing to it until all shared locks are released.

Exclusive (Write) Locks: Restrict access to a data item to a single transaction for writing, preventing other transactions from reading or writing to it until the exclusive lock is released.

Locking Protocols:

Two-Phase Locking (2PL): Transactions acquire all the locks they need before starting execution (growing phase) and release all locks at once when they complete (shrinking phase). This protocol ensures serializability but may lead to deadlock if not managed properly.

Timestamp Ordering: Each transaction is assigned a unique timestamp, and locks are acquired and released based on these timestamps to ensure serializability and prevent conflicts. Older transactions are given priority over newer transactions.

Deadlock Detection and Prevention: Deadlocks occur when two or more transactions are waiting indefinitely for resources held by each other, resulting in a cyclic dependency. Techniques such as deadlock detection algorithms and deadlock prevention strategies (e.g., timeout mechanisms, deadlock avoidance) are employed to detect and resolve deadlocks in the database system.

Isolation Levels: Isolation levels define the degree of isolation between concurrent transactions, determining the visibility of intermediate transaction states and the phenomena they can exhibit (e.g., dirty reads, non-repeatable reads, phantom reads). Common isolation levels include Read Uncommitted, Read Committed, Repeatable Read, and Serializable.