Experiment No.10

Implementation and demonstration of Transaction and Concurrency control techniques using locks

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;

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

2. Explain Transaction and Concurrency control techniques using locks.

➔ **Transaction and Concurrency Control Techniques Using Locks**:

- **Transaction**:
  - A transaction is a sequence of one or more SQL statements that are executed as a single unit of work.
  - It ensures that a set of operations are executed atomically (all or none).
  - Properties of a transaction (ACID properties):
    - **Atomicity**: Either all operations in a transaction are executed, or none of them are.

- **Consistency**: A transaction brings the database from one consistent state to another.
- **Isolation**: Transactions execute independently without interfering with each other.
- **Durability**: Once a transaction is committed, its effects are permanent.

- **Concurrency Control Techniques Using Locks**:
  - Concurrency control ensures that multiple transactions can execute concurrently without causing data inconsistencies.
  - Locks are used to prevent conflicts between concurrent transactions.
  - Types of locks:
    - **Shared Lock (S)**: Allows multiple transactions to read the same data simultaneously. No transaction can modify the data while a shared lock is held.
    - **Exclusive Lock (X)**: Prevents other transactions from reading or writing the data. Only one transaction can hold an exclusive lock at a time.
    - **Update Lock (U)**: A combination of shared and exclusive lock. It allows multiple transactions to read the data, but only one transaction can modify it.
    - **Intent Lock**: Indicates the intention of a transaction to acquire a shared or exclusive lock on a resource.
    - **Deadlock**: Occurs when two or more transactions are waiting indefinitely for each other to release locks.
    - **Two-Phase Locking Protocol (2PL)**: Ensures serializability by acquiring all locks before releasing any.
    - **Timestamp-Based Concurrency Control**: Assigns a timestamp to each transaction and uses it to determine the order of execution.
    - **Optimistic Concurrency Control**: Allows transactions to proceed without acquiring locks initially. Conflicts are detected during commit.
    - **Multi-Version Concurrency Control (MVCC)**: Creates multiple versions of data to allow concurrent reads and writes without blocking.
    - **Serializable Schedules**: Ensures that the execution of transactions is equivalent to some serial execution.
- **Benefits of Using Locks**:
  - Ensures data consistency and integrity.
  - Allows concurrent execution of transactions.
  - Prevents conflicts and race conditions.