



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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

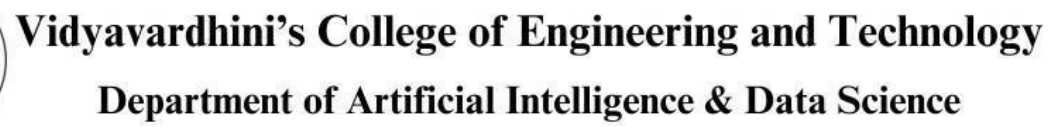
: LOCK:-

The screenshot shows a MySQL query editor with the following SQL code:

```
1 use social_media_db;
2 LOCK TABLES users WRITE;
3 select*from users;
4
5
6
7
```

The output window displays the following results:

#	Time	Action	Message
1	01:40:19	use social_media_db	0 row(s) affected
2	01:41:31	select*from users LIMIT 0, 1000	7 row(s) returned
3	01:41:54	LOCK TABLES users WRITE	0 row(s) affected



The screenshot shows a SQL editor interface. The top toolbar includes icons for saving, running, undo, redo, and other standard editing functions. The query window contains the following SQL code:

```
1 use social_media_db;  
2 • UNLOCK TABLES;  
3  
4  
5  
6
```

Below the query window is a horizontal separator line. The bottom section of the interface is the 'Output' window, which has a dropdown menu set to 'Action Output'. It displays a table of execution results:

#	Time	Action	Message
✓ 1	01:43:58	UNLOCK TABLES	0 row(s) affected
✓ 2	01:44:34	select from users LIMIT 0, 1000	7 row(s) returned
✓ 3	01:44:53	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.

1. Explain Transaction and Concurrency control techniques using locks.

here's a brief explanation of transactions and concurrency control techniques using locks:

1. Transaction:

- A transaction is a logical unit of work that consists of one or more database operations, such as INSERT, UPDATE, DELETE, or SELECT.
- Transactions ensure that all operations within them are executed atomically, meaning they either all succeed or all fail, maintaining data consistency.
- The ACID properties (Atomicity, Consistency, Isolation, Durability) define the characteristics of a transaction, ensuring reliability and data integrity.

2. Concurrency Control Techniques Using Locks:

- Locking: Locks are used to control access to shared resources (e.g., database records) to prevent conflicts and maintain data consistency in a multi-user environment.
- Types of Locks:
 - **Shared Locks (Read Locks): Allow multiple transactions to read data simultaneously but prevent write operations until all shared locks are released.
 - Exclusive Locks (Write Locks)**: Prevent other transactions from reading or writing to a resource until the exclusive lock is released.
- Concurrency Control Protocols:
 - Two-Phase Locking (2PL): Transactions acquire locks in two phases (growing phase and shrinking phase) and hold them until the end of the transaction. This ensures serializability but may lead to deadlocks.
 - Timestamp Ordering: Assigns a unique timestamp to each transaction and uses these timestamps to order conflicting operations, ensuring serializability without explicit locking. Conflicts are resolved by comparing transaction timestamps.
 - Optimistic Concurrency Control (OCC): Transactions operate without acquiring locks initially. Conflicts are detected at commit time, and if conflicts occur, the transaction is rolled back and retried.
 - Multi-Version Concurrency Control (MVCC): Maintains multiple versions of a data item to allow concurrent read and write operations without blocking. Each transaction sees a consistent snapshot of the database at its start time. These concurrency control techniques using locks ensure that transactions execute safely and efficiently in a multi-user database environment, preventing data inconsistencies and ACID properties. Different techniques may be suitable depending on the specific requirements and characteristics of the application.