**Qatar University**

**College of Engineering**

**Department of Computer Science and Engineering**

**CMPS 451 – Fall 2023**

**Assignment 3**

**SID**: 202005204 **STUDENT NAME:** Salma Eletreby **Email**: Se2005204@qu.edu.qa

**SID**: 202003532 **STUDENT NAME:** Aya Muhanad Hassan **Email**: ah2003532@qu.edu.qa

**SID**: 202001942 **STUDENT NAME:** Hissa Al Muhannadi **Email**: ha2001942@qu.edu.qa

**Submission date**: 6th December 2023

**Class section:** L51

Concurrency control mechanisms in MySQL

MySQL uses several different storage engines, one of them is InnoDB which is the default engine. It is an ACID-compliant storage engine that has mechanisms to commit, rollback, and recover crashed data. This engine becomes useful when more than one user edits info in a row as it prevents reads and writes from interfering with each other.

MySQL has two types of internal locking methods. One is row-level and the other is table-level. For the InnoDB tables, row-level locking is used which allows simultaneous write access by different users. The method acquires the locks of all the rows to be modified at the start of the transaction, this avoids deadlocks. In case the transaction modifies multiple rows from different tables, all locks are acquired for these tables at the start of the transaction and in the same order as in the transaction.The advantage of InnoDB is that it can detect deadlocks and rollback the transaction that caused it. This method has similarities to the conservative two-phase locking protocol as it also requires all items to be locked before the transaction begins and is a deadlock-free protocol.

Some of the types of locks used by InnoDB were covered in the lecture. For the standard row-level locking, InnoDB uses shared and exclusive locks.Another type is intention locks which can be of type intention-shared and intention-exclusive. Which follow similar rules to the multiple granularity level locking covered in the lecture. The difference is that no shared-intension-exclusive locks are used in InnoDB. A second difference is that the protocol in InnoDB does not allow transactions to unlock a node rather it waits until the existing lock is released.

The second internal locking method is table-level locking, used for MyISAM, MEMORY, and MERGE tables, which allows only one session to update tables. This method is suitable for databases that are used for read-mostly and single-user sessions. It prevents deadlock by acquiring all locks at the start of the transaction and updating the tables in the same order of the transaction. It is also similar to the two-phase locking protocol covered in the lecture. Write locks are granted only if no lock is on the table. Otherwise, the request will be added to the write lock queue. For a table to be granted read locks, it must not have any write locks. Otherwise, the request is added to the read lock queue. Table write locks are given higher priority than table read locks. This prevents the starvation of writing requests even if there are many read requests. Which is different from the solution to starvation covered in the lecture (FCFS).

Another protocol in MySQL is the InnoDB multi-versioning protocol which allows each user to have different views of the database depending on when the access begins.This method works by adding extra fields to each row in the database. One is the DB\_TRX\_ID field which contains the identifier of the last transaction that either updated, inserted, or deleted the row. Another is the DB\_ROLL\_PTR which contains the necessary information of a row before it is updated. This is similar to the multi-version concurrency control technique based on snapshot isolation discussed in the lecture. The similarity is that a transaction views data items based on committed values. This occurs because the update undo logs are only disregarded if the InnoDB has not assigned a snapshot to a transaction that uses the earlier version of the data item. Other similarities are that both keep track of older versions of the update item and read locks are not needed.

<https://dev.mysql.com/doc/refman/8.0/en/innodb-introduction.html>

<https://dev.mysql.com/doc/refman/8.0/en/internal-locking.html>

<https://dev.mysql.com/doc/refman/8.0/en/innodb-locking.html>

<https://dev.mysql.com/doc/refman/8.0/en/innodb-multi-versioning.html>

Recovery Mechanisms in MySQL

1. Transaction Logging (redo log in MySQL):

* **Similarities:**
* A record of every transaction is saved. If there is a system failure, the transactions from the log will be replayed.
* Modifications that did not finish updating data files before an unexpected shutdown are replayed automatically.[[1]](#footnote-1)
* **Differences:**
* If all changes are flushed from the buffer to the tablespaces at the time of the crash, the redo log application is skipped. Also, InnoDB skips the redo log application if redo log files are missing at startup.
* During recovery, InnoDB scans the redo log to collect counter value changes and applies the changes to the in-memory table object.
* When encountering index tree corruption, InnoDB writes a corruption flag to the redo log, which makes the corruption flag crash safe.[[2]](#footnote-2)

1. Checkpointing:

* **Similarities:**
* During crash recovery, InnoDB looks for a checkpoint label written on the log files. Then InnoDB scans the log files forward from the checkpoint, applying the logged modifications to the database.
* **Differences:**
* InnoDB writes checkpoint information to the first log file at each checkpoint.
* InnoDB creates checkpoints and this often involves flushing of modified database pages to disk.[[3]](#footnote-3)

1. Undo log:

* **Similarities:**
* The undo log is a collection of undo log records associated with a single read-write transaction.
* **Differences:**
* The undo log contains information about how to undo the latest change by a transaction to a clustered index record.[[4]](#footnote-4)

Security Mechanisms in MySQL

MySQL has three primary ways for applying security mechanisms: role-based access, authentication, and encryption. As discussed during the lecture, role-based access enforces policies and privileges based on organization roles. As discussed in the lecture, privileges are permissions to be granted to a user to allow them to perform certain functionalities. In MySQL, roles are created by grouping different privileges together [[5]](#footnote-5) and when a user is created by database administrator (DBA) they are assigned a role which immediately grants them all the privileges that this role has without having to grant each privilege individually[[6]](#footnote-6).

MySQL‘s connection allows for it to be encrypted using the Transport Layer Security (TLS) protocol. This allows MySQL to ensure that data received over a network is secure and trusted. The algorithms used by TLS allow MySQL to detect changes or losses in data or replay a transaction. [[7]](#footnote-7)

There are many ways to identify and authenticate a user. First is using electronic certificates granted by the Certificate Authority (CA). These certificates use a public key and a secret key so when a user presents their certificate as proof of their identity, if they are authentic, then using the secret corresponding key it can be decrypted [[8]](#footnote-8). The second is using the combination of access control and password to authenticate the user and ensure which transactions they can perform, which is called the privilege system. When a program attempts to connect to the MySQL server, MySQL will accept or reject based on verification of identity using the combination of correct username and password. If everything is correct, the connection will be accepted and any statement that is executed is double checked with the privileges that the client has as discussed in the lecture [[9]](#footnote-9). The security of the database can be enhanced further by using the validate\_password component that allows a DBA to specify the password’s length and other strength requirements to prevent attackers from guessing a password [[10]](#footnote-10). The third is using authentication plugins which allow for the creation of proxy users. Upon connection, the plugin will request that the connecting user is treated as a proxy user for checking privileges [[11]](#footnote-11). This proxy user allows for better security by hiding the database and easier auditing as it centralizes the connection.

In MySQL, DBAs have the authority to lock and unlock an account [[12]](#footnote-12). As discussed during the lecture, there might be login sessions or audits that record transactions, so if in any of these logs suspicious activity was found, a DBA has the authority to lock the account to investigate further and prevent the suspected user from preforming more transactions. Once the suspicion is cleared, the account can be unlocked. This helps in securing the database by preventing a suspected attacker from doing more harm.

Lastly, this option is available to MySQL enterprise which is data masking and de-identification. As discussed in the lecture, some sensitive data may exist that should be hidden in most cases, using those features we can hide the data yet still be able to use it in our transactions. For example, using the data masking feature we can transform the digits of a credit card number into an X or use random data for payment card numbers using the de-identification [[13]](#footnote-13). This would allow all non-authorized users to view the data, generate statistics or report without compromising the individual’s privacy.

1. <https://dev.mysql.com/doc/refman/8.0/en/innodb-redo-log.html#:~:text=The%20redo%20log%20is%20a,or%20low%2Dlevel%20API%20calls> [↑](#footnote-ref-1)
2. <https://dev.mysql.com/doc/refman/8.0/en/innodb-recovery.html> [↑](#footnote-ref-2)
3. <http://download.nust.na/pub6/mysql/doc/mysql-backup-excerpt/5.1/en/innodb-checkpoints.html> [↑](#footnote-ref-3)
4. <https://dev.mysql.com/doc/refman/8.0/en/innodb-undo-logs.html> [↑](#footnote-ref-4)
5. <https://dev.mysql.com/doc/refman/8.0/en/roles.html> [↑](#footnote-ref-5)
6. <https://dev.mysql.com/doc/refman/8.0/en/roles.html> [↑](#footnote-ref-6)
7. <https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/encrypted-connections.html> [↑](#footnote-ref-7)
8. <https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/encrypted-connections.html> [↑](#footnote-ref-8)
9. <https://dev.mysql.com/doc/refman/8.0/en/access-control.html> [↑](#footnote-ref-9)
10. <https://dev.mysql.com/doc/refman/8.0/en/validate-password.html> [↑](#footnote-ref-10)
11. <https://dev.mysql.com/doc/refman/8.0/en/proxy-users.html> [↑](#footnote-ref-11)
12. <https://dev.mysql.com/doc/refman/8.0/en/account-locking.html> [↑](#footnote-ref-12)
13. <https://dev.mysql.com/doc/refman/8.0/en/data-masking.html> [↑](#footnote-ref-13)