Database Management System – cs422 DE

Assignment 7 – Week 10 & 11

This assignment is based on lecture 9 (chapter 22).

○ Submit your *own work* on time. No credit will be given if the assignment is submitted after the due date.
○ Note that the completed assignment should be submitted in .pdf format only.
○ In MCQs, if you think that your answer needs more explanation to get credit then please write it down.

---

(1) _____ ensures that once transaction changes are done, they cannot be undone or lost, even in the event of a system failure.

A. Atomicity
B. Consistency
C. Durability
D. Isolation

ANS: C

(2) Deadlocks are possible only when one of the transactions wants to obtain a(n) _____ lock on a data item.

A. Binary
B. Shared
C. Exclusive
D. Complete

ANS: C

(3) If several concurrent transactions are executed over the same data set and the second transaction updates the database before the first transaction is finished, the _____ property is violated and the database is no longer consistent.

A. Atomicity
B. Consistency
C. Durability
D. Isolation

ANS: D

(4) When a program is abnormally terminated, the equivalent of a ___ command occurs.

A. COMMIT
B. ROLLBACK
C. QUIT
D. EXIT

ANS:

(5) The deadlock state can be changed back to stable state by using _____ statement.

A. COMMIT
B. ROLLBACK
C. SAVEPOINT
D. DEADLOCK

ANS: B

(6) Explain what is meant by a transaction. Why are transactions important units of operation in a DBMS?

ANS: A transaction is a sequence of database operations (read/write) that represents a single logical unit of work. A transaction must either complete entirely (COMMIT) or have no effect at all (ROLLBACK).

Transactions are important because they enforce the ACID properties: Atomicity, Consistency, Isolation, Durability

Without transactions, partial updates, data corruption, and inconsistencies could occur, especially in multi-user systems.

(7) Describe, with examples, the types of problem that can occur in a multi-user environment when concurrent access to the database is allowed.

ANS:

When multiple users access the database at the same time, the following problems can occur:

Lost Update: Two transactions update the same data, and one update overwrites the other.
Example: T1 and T2 read balance = 100. T1 updates to 150, then T2 updates to 120 → T1's update is lost.

Dirty Read: A transaction reads data written by another transaction that has not yet committed.
Example: T2 reads a value updated by T1, but T1 later aborts.

Non-Repeatable Read: A transaction reads the same data twice and gets different values.
Example: T1 reads balance, T2 updates and commits, T1 reads again and sees a new value.

Phantom Read: A transaction re-runs a query and finds new rows added by another transaction.
Example: T1 counts rows, T2 inserts a new row and commits, T1 counts again and gets a different result.

(8) Give full details of a mechanism for concurrency control that can be used to ensure the types of problems discussed in the above question cannot occur. Show how the mechanism prevents the problems illustrated from occurring. Discuss how the concurrency control mechanism interacts with the transaction mechanism.

ANS:

A commonly used concurrency control mechanism is Strict Two-Phase Locking (Strict 2PL).

How it works:

- A transaction must obtain a Shared (S) lock before reading data.
- A transaction must obtain an Exclusive (X) lock before writing data.
- All locks are held until the transaction commits or aborts.
- Locks are released only at COMMIT or ROLLBACK.

How it prevents concurrency problems:

- Lost update: Exclusive locks prevent two transactions from writing the same data at the same time.
- Dirty read: Transactions cannot read uncommitted data because locks are held until commit.
- Non-repeatable read: Read locks prevent other transactions from modifying data being read.
- Phantom read: Locks on data items (or ranges) prevent unexpected new records.

Interaction with the transaction mechanism:
- Locks are acquired during transaction execution.
- If a transaction commits, all changes become permanent and locks are released.
- If a transaction aborts, all changes are rolled back and locks are released.
- This ensures serializability, recoverability, and consistency of the database.

(9) Explain the concepts of serial, non-serial, and serializable schedules. State the rules for equivalence of schedules.
ANS:
- Serial schedule: Transactions execute one after another with no interleaving.
- Non-serial schedule: Operations of multiple transactions are interleaved.
- Serializable schedule: A non-serial schedule that produces the same result as some serial schedule.

Rules for equivalence of schedules:
- Conflict equivalence: Conflicting operations occur in the same order.
- View equivalence: Same read-from relationships and same final writes.

(10) What is a timestamp? How do timestamp-based protocols for concurrency control differ from locking based protocols?
ANS:
A timestamp is a unique value assigned to a transaction when it starts, representing its start time.

Timestamp-based protocols: Use timestamps to order transactions, do not use locks, and avoid deadlocks but may cause transaction aborts.

Locking-based protocols: Use shared and exclusive locks, may cause waiting and deadlocks, but usually cause fewer aborts.