**Assignment 3**

**Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.**

**ACID Properties of a Transaction**

**Atomicity:** This property ensures that a transaction is treated as a single unit, which either completes in its entirety or does not complete at all. If any part of the transaction fails, the entire transaction is rolled back, leaving the system in its previous state.

**Consistency:** This guarantees that a transaction will bring the database from one valid state to another valid state, maintaining database invariants. For instance, if a database rule states that a specific field must always be positive, a transaction must ensure this rule is respected even after its execution.

**Isolation:** Isolation ensures that concurrently executed transactions do not affect each other. The intermediate state of a transaction is invisible to other transactions until the transaction is complete, preventing concurrent transactions from reading uncommitted changes.

**Durability:** This property guarantees that once a transaction has been committed, it will remain so, even in the event of a system failure. This is typically achieved through logging mechanisms that ensure committed changes are recorded permanently.

**SQL Statements to Simulate a Transaction with Locking and Isolation Levels**

Here, we will simulate a transaction in a database environment using SQL. Let's assume we have a simple table account with the following schema:

CREATE TABLE accounts (

account\_id INT PRIMARY KEY,

balance DECIMAL (10, 2)

);

**Setting up initial Data:**

INSERT INTO accounts (account\_id, balance) VALUES (1, 1000.00),

(2, 1500.00);

**Transaction Example with Locking**

In this example, we'll perform a transfer of **$100** from account\_id 1 to account\_id 2.

BEGIN TRANSACTION;

SELECT balance FROM accounts WHERE account\_id = 1 FOR UPDATE;

SELECT balance FROM accounts WHERE account\_id = 2 FOR UPDATE;

UPDATE accounts SET balance = balance - 100.00 WHERE account\_id = 1;

UPDATE accounts SET balance = balance + 100.00 WHERE account\_id = 2;

COMMIT;

**Demonstrating Different Isolation Levels**

1. **READ UNCOMMITTED:** Transactions can read data that has been modified but not yet committed by other transactions.

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

BEGIN TRANSACTION;

SELECT balance FROM accounts WHERE account\_id = 1;

-- Data read might be dirty (uncommitted changes from other transactions), COMMIT;

1. **READ COMMITTED**: A transaction can only read data that has been committed. This prevents dirty reads.

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

BEGIN TRANSACTION;

SELECT balance FROM accounts WHERE account\_id = 1;

-- Only committed data can be read

COMMIT;

1. **REPEATABLE READ:** Ensures that if a transaction reads a row, it will see the same data throughout the transaction, preventing non-repeatable reads or phantom reads.

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

BEGIN TRANSACTION;

SELECT balance FROM accounts WHERE account\_id = 1;

-- Subsequent reads will see the same data even if other transactions modify it

COMMIT;

1. **SERIALIZABLE:** The highest isolation level, which ensures complete isolation from other transactions. This level can be thought of as transactions being executed serially rather than concurrently.

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

BEGIN TRANSACTION;

SELECT balance FROM accounts WHERE account\_id = 1;

-- Ensures no other transaction can modify the data being read until this transaction completes

COMMIT;

**Summary**

* Atomicity ensures all or nothing execution.
* Consistency maintains database rules.
* Isolation prevents interference between transactions.
* Durability ensures committed changes persist.

Using different isolation levels, you can control how transactions interact with each other and manage concurrency to avoid issues like dirty reads, non-repeatable reads, and phantom reads.