Transaction Management

Principles

What is Transaction Managment?

Transaction: sequence of one or more SQL operations executed as a single unit. It ensures that the database remains in a consistent state.

A transaction is either fully completed (**committed**) or undone (**rolled back**) if any part fails.

Importance of Transactions

- Ensure data integrity when multiple users are accessing or modifying the database simultaneously.
- Prevents inconsistent data during concurrent operations.
- Key for managing concurrency and avoiding conflicts between users.

ACID Principles

ACID properties: set of key principles that ensure reliable processing of database transactions. The properties contribute to the consistency, integrity and robustness of database operations.

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

Why ACID Properties

- **Data Integrity**: ensure the database remains in a valid, consistent state, even in the face of failures or concurrent access.
- **Concurrency Control**: by providing **isolation**, multiple users can interact with the database at the same time without corrupting the data.

- **Error Handling**: if a transaction fails due to an error, **atomicity** ensures no partial changes are applied, keeping the database clean.
- **Data Reliability**: **durability** ensures that once a transaction is committed, the changes won't be lost, providing reliability and confidence in data storage.

1. Atomicity

Atomicity ensures that each transaction is treated as a single "unit of work". Either all operations within a transaction are completed successfully, or one of them are applied to the database.

2. Consistency

Consistency ensures that a transaction brings the database from one valid state to another, preserving any defined rules, constraints and data integrity.

3. Isolation

Isolation ensures that transactions executed concurrently are isolated from each other, meaning the operations of one transaction are not visible to other transactions until they are completed. This prevents conflicts and ensures consistency even when multiple transactions are running at the same time.

4. Durability

Durability guarantees that once a transaction is committed, its changes are permanently recorded in the database, even in the case of a system crash or power failure.

Concurrency Anomalies and Isolation Levels

Dirty Read

A **dirty read** occurs when a transaction reads data that has been modified by another transaction but not yet committed. If the other transaction rolls back, the data read by the first transaction becomes invalid or inconsistent.

This issue can arise in low isolation levels like **Read Uncommitted**.

Non-Repeatable Read

A **non-repeatable read** occurs when a transaction reads the same data twice but gets different results due to changes made by another transaction in between the reads.

Lost Updates

A **lost update** occurs when two transactions simultaneously read and update the same data.

The second transaction overwrites the changes made by the first transaction, without either being aware of the conflict. This results in the first update being lost.

Phantom Read

A **phatom read** occurs when a transaction reads a set of rows that match a condition, but another transaction inserts or deletes rows that affect