**ACID Properties**

ACID is an acronym that stands for atomicity, consistency, isolation, and durability.

Together, these ACID properties ensure that a set of database operations (grouped together in a transaction) leave the database in a valid state even in the event of unexpected errors.

**Atomicity**

Atomicity guarantees that all of the commands that make up a transaction are treated as a single unit and either succeed or fail together. This is important as in the case of an unwanted event, like a crash or power outage, we can be sure of the state of the database. The transaction would have either completed successfully or been rollbacked if any part of the transaction failed.

example, money is deducted from the source and if any anomaly occurs, the changes are discarded and the transaction fails.

**Consistency**

Consistency guarantees that changes made within a transaction are consistent with database constraints. This includes all rules, constraints, and triggers. If the data gets into an illegal state, the whole transaction fails.

example, let’s say there is a constraint that the balance should be a positive integer. If we try to overdraw money, then the balance won’t meet the constraint. Because of that, the consistency of the ACID transaction will be violated and the transaction will fail.

**Isolation**

Isolation ensures that all transactions run in an isolated environment. That enables running transactions concurrently because transactions don’t interfere with each other.

For example, let’s say that our account balance is $200. Two transactions for a $100 withdrawal start at the same time. The transactions run in isolation which guarantees that when they both complete, we’ll have a balance of $0 instead of $100.

**Durability**

Durability guarantees that once the transaction completes and changes are written to the database, they are persisted. This ensures that data within the system will persist even in the case of system failures like crashes or power outages.

The ACID characteristics of transactions are what allow developers to perform complex, coordinated updates and sleep well at night knowing that their data is consistent and safely stored.

2) SQL statementsto simulate a transaction that includes locking

-- Start a transaction

BEGIN TRANSACTION;

-- Lock the table

LOCK TABLE accounts IN SHARE MODE;

-- Update the account balance

UPDATE accounts SET balance = balance - 100 WHERE account\_id = 1;

-- Commit the transaction

COMMIT;