**# ACID Properties in Databases**

ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability. These properties are essential for ensuring reliable transaction processing in a database. Here's a human-friendly explanation of each property, along with a simple example to illustrate how they work together.

**1. Atomicity**

- Definition: Atomicity ensures that a series of operations within a transaction are treated as a single unit. This means that either all the operations are completed successfully, or none of them are. If any part of the transaction fails, the entire transaction is rolled back, and the database remains unchanged.

**2. Consistency**

- Definition: Consistency ensures that a transaction takes the database from one valid state to another, maintaining all predefined rules, such as constraints, cascades, and triggers. After the transaction is completed, the database must remain in a valid state.

- Example: Suppose a database constraint requires that the total balance across all accounts must always be positive. If a transaction results in a negative total balance, consistency ensures that the transaction will be rolled back, maintaining the rule.

**3. Isolation**

- Definition: Isolation ensures that the operations of a transaction are hidden from other transactions until the transaction is completed. This prevents transactions from interfering with each other and ensures data integrity.

- Example: Imagine two people are simultaneously transferring money into the same account. Isolation ensures that one transfer does not interfere with the other, and each transaction will see a consistent state of the account balance.

**4. Durability**

- Definition: Durability guarantees that once a transaction is committed, it will remain so, even in the event of a system failure (like a crash or power outage). This means that completed transactions are permanently recorded in the database.

- Example: If you transfer money and receive a confirmation, durability ensures that even if the system crashes immediately after, the transferred amount will still be reflected in the account balance when the system is back online.

--------------------------------------------------------------------------------------------------------------------------

1. Atomicity:

Atomicity ensures that all operations within a transaction are completed successfully. If any operation fails, the transaction is rolled back.

---------------------------

-- Create the accounts table

CREATE TABLE accounts (

account\_id INT PRIMARY KEY,

account\_name VARCHAR(100),

balance DECIMAL(10, 2)

);

-- Insert sample data

INSERT INTO accounts (account\_id, account\_name, balance) VALUES (1, 'Alice', 1000.00);

INSERT INTO accounts (account\_id, account\_name, balance) VALUES (2, 'Bob', 500.00);

-- Begin transaction

BEGIN;

-- Subtract $100 from Alice's account

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

-- Add $100 to Bob's account

UPDATE accounts SET balance = balance + 100 WHERE account\_id = 2;

-- Commit the transaction

COMMIT;

2. Consistency:

Consistency ensures that a transaction takes the database from one valid state to another, maintaining all predefined rules.

---------------------------

-- Add a constraint that ensures balance is never negative

ALTER TABLE accounts ADD CONSTRAINT balance\_non\_negative CHECK (balance >= 0);

-- Begin transaction

BEGIN;

-- Try to subtract $1100 from Alice's account (should fail)

UPDATE accounts SET balance = balance - 1100 WHERE account\_id = 1;

-- Add $1100 to Bob's account (won't execute because of the above failure)

UPDATE accounts SET balance = balance + 1100 WHERE account\_id = 2;

-- Commit the transaction

COMMIT;

3. Isolation:

Isolation ensures that the operations of one transaction are isolated from those of other transactions.

---------------------------

-- Session 1: Begin transaction

BEGIN;

-- Session 1: Select Alice's balance (initially 1000)

SELECT balance FROM accounts WHERE account\_id = 1;

-- Session 2: Begin transaction

BEGIN;

-- Session 2: Transfer $100 from Alice to Bob

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

UPDATE accounts SET balance = balance + 100 WHERE account\_id = 2;

-- Session 2: Commit the transaction

COMMIT;

-- Session 1: Try to select Alice's balance again

SELECT balance FROM accounts WHERE account\_id = 1;

-- Session 1: Commit the transaction

COMMIT;

4. Durability:

Durability ensures that once a transaction is committed, its changes are permanent, even in the case of a system failure.

---------------------------

-- Begin transaction

BEGIN;

-- Transfer $200 from Alice to Bob

UPDATE accounts SET balance = balance - 200 WHERE account\_id = 1;

UPDATE accounts SET balance = balance + 200 WHERE account\_id = 2;

-- Commit the transaction

COMMIT;

-- Simulate a system crash

-- (In practice, the DBMS handles this with transaction logs)

-- After system recovery, verify the balances

SELECT \* FROM accounts;

--------------------------------------------------------------------------------------------------------------------------

**Example Scenario: Bank Transfer**

Let's walk through a bank transfer example to see how the ACID properties work together:

**Scenario**: Alice wants to transfer $100 from her savings account to her checking account.

1. **Atomicity**:
   * The transaction includes two steps: subtracting $100 from Alice's savings account and adding $100 to her checking account.
   * If the subtraction step succeeds but the addition step fails, the entire transaction is rolled back, and Alice's accounts remain unchanged.
2. **Consistency**:
   * The database ensures that Alice’s total balance (savings + checking) remains the same before and after the transaction.
   * If an error occurs that violates any database constraints (e.g., a rule that accounts cannot have a negative balance), the transaction will not be completed.
3. **Isolation**:
   * If Bob is also transferring money to Alice’s checking account at the same time, isolation ensures that Alice’s checking account balance is updated correctly for each transaction, as if they were processed one after the other, not simultaneously.
4. **Durability**:
   * Once Alice receives confirmation that the transfer is complete, the changes are permanent. Even if there is a power outage or system crash immediately after, the $100 will still be in her checking account once the system recovers.