**ACID properties in sql**

ACID properties are a set of principles that ensure reliable processing of database transactions in SQL databases. They stand for Atomicity, Consistency, Isolation, and Durability. Each property contributes to the integrity and reliability of transactions. Here’s a detailed look at each property:

**1. Atomicity**

**Explanation:** Atomicity ensures that a transaction is all-or-nothing. If any part of the transaction fails, the entire transaction fails, and the database state is left unchanged. This means that either all operations within the transaction are completed successfully, or none are.

**Example:** Consider a banking system where you transfer $100 from Account A to Account B. This transaction involves two steps:

1. Deduct $100 from Account A.
2. Add $100 to Account B.

If the transaction fails after step 1 but before step 2, atomicity ensures that step 1 is rolled back, so the $100 is not deducted from Account A.

|  |
| --- |
| BEGIN TRANSACTION;  UPDATE accounts SET balance = balance - 100 WHERE account\_id = 'A';  UPDATE accounts SET balance = balance + 100 WHERE account\_id = 'B';  -- If any statement fails, ROLLBACK;  COMMIT; |

If any UPDATE statement fails, the transaction will be rolled back to its initial state.

**2. Consistency**

**Explanation:** Consistency ensures that a transaction takes the database from one valid state to another, maintaining database invariants and constraints. This means that only valid data according to all rules and constraints is written to the database.

**Example:** In a banking system, an invariant might be that the total amount of money in all accounts must remain constant. After transferring $100 from Account A to Account B, the total amount should still be the same.

|  |
| --- |
| BEGIN TRANSACTION;  -- Check initial invariant  SELECT SUM(balance) FROM accounts; -- Suppose the total is $2000  UPDATE accounts SET balance = balance - 100 WHERE account\_id = 'A';  UPDATE accounts SET balance = balance + 100 WHERE account\_id = 'B';  -- Check final invariant  SELECT SUM(balance) FROM accounts; -- Should also be $2000  COMMIT; |

If the total balance doesn't remain consistent, the transaction will be rolled back.

**3. Isolation**

**Explanation:** Isolation ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed sequentially. This means transactions are isolated from each other and intermediate states are not visible to each other.

**Example:** Consider two transactions:

* T1: Transfer $100 from Account A to Account B.
* T2: Transfer $200 from Account B to Account A.

Isolation ensures these transactions do not interfere with each other. Depending on the isolation level, the intermediate states of these transactions are not visible to each other.

|  |
| --- |
| -- Transaction T1  BEGIN TRANSACTION;  UPDATE accounts SET balance = balance - 100 WHERE account\_id = 'A';  UPDATE accounts SET balance = balance + 100 WHERE account\_id = 'B';  COMMIT;  -- Transaction T2  BEGIN TRANSACTION;  UPDATE accounts SET balance = balance - 200 WHERE account\_id = 'B';  UPDATE accounts SET balance = balance + 200 WHERE account\_id = 'A';  COMMIT; |

Without proper isolation, one transaction might see the intermediate state of the other, leading to inconsistent results.

**4. Durability**

**Explanation:** Durability ensures that once a transaction has been committed, it remains committed even in the case of a system failure. This means the changes made by the transaction are permanently recorded in the database.

**Example:** After transferring $100 from Account A to Account B and committing the transaction, the changes are permanently stored in the database. Even if the system crashes immediately after the commit, the changes will not be lost.

|  |
| --- |
| BEGIN TRANSACTION;  UPDATE accounts SET balance = balance - 100 WHERE account\_id = 'A';  UPDATE accounts SET balance = balance + 100 WHERE account\_id = 'B';  COMMIT; |

After the COMMIT statement, the changes are durable and will survive system crashes.