TRANSACTIONS

Transaction in SQL:

- Sequence of one or more SQL statements that are executed as a single unit of work.
- Ensure data integrity and follow ACID (Atomicity, Consistency, Isolation, Durability) properties.
- *Goal*: To ensure that all operations within a transaction are either completed successfully or rolled back entirely if an error occurs.

Commands:

BEGIN TRANSACTION

The BEGIN TRANSACTION command marks the beginning of a new transaction. All SQL statements that follow this command will be part of the same transaction until a COMMIT or ROLLBACK is encountered.

COMMIT

The COMMIT command is used to save all changes made during the current transaction to the database.

ROLLBACK

The ROLLBACK command is used to undo all changes made in the current transaction. It is used when an error occurs or when the desired changes cannot be completed.

SAVEPOINT

A SAVEPOINT is used to create a checkpoint within a transaction. We can roll back to a specific SAVEPOINT instead of rolling back the entire transaction. This allows us to undo part of the transaction rather than the entire transaction.

1. If a transaction encounters an error in the third statement and a ROLLBACK is issued, will the prior updates be retained?

No, the prior updates will be undone. Since no SAVEPOINT was created, the ROLLBACK will revert all changes made within the transaction.

2. Can another transaction view Alice's updated balance before she commits her transaction in the READ COMMITTED isolation level?

No, only committed changes are visible. Until Alice's transaction is committed, her balance update remains hidden from other transactions.

3. If two transactions simultaneously reduce Alice's balance by 100, will one overwrite the other?

It depends on the isolation level. In READ COMMITTED, the second transaction could modify the balance based on outdated data. In SERIALIZABLE, one transaction would be aborted to prevent conflicts.

4. If ROLLBACK TO SAVEPOINT after_alice; is executed, will all changes be undone or only those made after the savepoint?

Only the changes made after the savepoint will be reverted, while earlier modifications remain intact.

5. Which isolation level ensures that phantom reads do not occur?

SERIALIZABLE prevents phantom reads by ensuring transactions operate as if executed sequentially.

6. Does PostgreSQL allow dirty reads (reading uncommitted data from another transaction)?

No, PostgreSQL does not permit dirty reads, as even READ COMMITTED ensures only committed data is visible.

7. If autocommit is enabled in PostgreSQL, does an UPDATE statement get immediately saved?

Yes, with autocommit enabled, every statement is automatically committed upon execution.

8.

If I do this:

BEGIN;

UPDATE accounts SET balance = balance - 500 WHERE id = 1;

-- (No COMMIT yet)

And from another session, I run:

SELECT balance FROM accounts WHERE id = 1;

Will the second session see the deducted balance?

No, the second transaction will view the original balance. PostgreSQL's MVCC ensures that uncommitted changes remain invisible to other transactions.