#PostgreSQL#

Section 10. Transactions
PostgreSQL Transactions – show you how to handle transactions in PostgreSQL using BEGIN, COMMIT, and ROLLBACK statements.

Can You Teach Me above all the topics with Realtime example (create a simple table and insert some values) and show output in Table Format and Also Explain Each topic why this needed, what happen, like syntax and so on.

PostgreSQL Transactions

A **transaction** is a sequence of operations performed as a single logical unit of work. Transactions ensure data integrity and provide control over data manipulation.

Why Transactions Are Needed:

1. **Atomicity**: All operations in a transaction are completed; if not, none are applied.
2. **Consistency**: Ensures the database remains consistent before and after the transaction.
3. **Isolation**: Transactions occur independently.
4. **Durability**: Once a transaction is committed, changes persist even after a crash.

Key Commands:

1. **BEGIN**: Starts a new transaction.
2. **COMMIT**: Saves changes made in the transaction.
3. **ROLLBACK**: Undoes changes made during the transaction.

Real-Time Example

Step 1: Create a Simple Table

```sql


CREATE TABLE accounts (
    account_id SERIAL PRIMARY KEY,
    account_name VARCHAR(50),
    balance NUMERIC(10, 2)
);
```

Step 2: Insert Initial Data

```sql
INSERT INTO accounts (account_name, balance) VALUES
('Alice', 1000.00),
('Bob', 500.00);
```

Step 3: Display Table

```sql
SELECT * FROM accounts;
```

account_id	account_name	balance
1	Alice	1000.00
2	Bob	500.00

Example 1: BEGIN and COMMIT

Scenario: Transferring $200 from Alice to Bob

```sql
BEGIN;

UPDATE accounts SET balance = balance - 200 WHERE account_name = 'Alice';
UPDATE accounts SET balance = balance + 200 WHERE account_name = 'Bob';

COMMIT;
```

Explanation:

1. **BEGIN** starts a transaction.
2. Alice's balance is reduced by $200.
3. Bob's balance is increased by $200.
4. **COMMIT** saves these changes to the database.

Output:

```sql
SELECT * FROM accounts;
```

account_id	account_name	balance
1	Alice	800.00
2	Bob	700.00

Example 2: ROLLBACK

Scenario: Canceling a Mistaken Update

```sql
BEGIN;
UPDATE accounts SET balance = balance - 500 WHERE account_name = 'Alice';
ROLLBACK;
```

Explanation:

1. **BEGIN** starts a transaction.
2. Alice's balance is mistakenly reduced by $500.
3. **ROLLBACK** undoes this change, reverting the balance.

Output:

```sql
SELECT * FROM accounts;
```

account_id	account_name	balance
1	Alice	800.00
2	Bob	700.00

Syntax Overview

1. **BEGIN**:

   ```sql
   BEGIN;
   ```

2. **COMMIT**:

   ```sql
   COMMIT;
   ```

3. **ROLLBACK**:

```sql
ROLLBACK;
```

Why Transactions Matter:

- **Prevent Partial Updates**: In complex operations, only complete and valid changes are applied.
- **Error Recovery**: Mistakes can be undone without affecting the database state.
- **Multi-User Environments**: Ensure operations don't interfere with each other.

Would you like to explore advanced transaction features like savepoints or isolation levels?