# POSTGRESQL - TRANSACTIONS

 

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

A transaction is the propagation of one or more changes to the database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing transaction on the table. It is important to control transactions to ensure data integrity and to handle database errors.

Practically, you will club many PostgreSQL queries into a group and you will execute all of them together as a part of a transaction.

## Properties of Transactions:

Transactions have the following four standard properties, usually referred to by the acronym ACID:

- **Atomicity:** ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure and previous operations are rolled back to their former state.
- **Consistency:** ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation:** enables transactions to operate independently of and transparent to each other.
- **Durability:** ensures that the result or effect of a committed transaction persists in case of a system failure.

## Transaction Control:

There are following commands used to control transactions:

- **BEGIN TRANSACTION:** to start a transaction.
- **COMMIT:** to save the changes, alternatively you can use **END TRANSACTION** command.
- **ROLLBACK:** to rollback the changes.

Transactional control commands are only used with the DML commands INSERT, UPDATE and DELETE only. They can not be used while creating tables or dropping them because these operations are automatically committed in the database.

## The BEGIN TRANSACTION Command:

Transactions can be started using BEGIN TRANSACTION or simply BEGIN command. Such transactions usually persist until the next COMMIT or ROLLBACK command is encountered. But a transaction will also ROLLBACK if the database is closed or if an error occurs.

Following is the simple syntax to start a transaction:

```
BEGIN;

or

BEGIN TRANSACTION;
```

## The COMMIT Command:

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

The syntax for COMMIT command is as follows:

```
COMMIT;

or

END TRANSACTION;
```

## The ROLLBACK Command:

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.

The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for ROLLBACK command is as follows:

```
ROLLBACK;
```

## Example:

Consider COMPANY table is having the following records:

```
 id | name   | age | address    | salary
----+--------+-----+------------+--------
  1 | Paul   |  32 | California|  20000
  2 | Allen |  25 | Texas      |  15000
  3 | Teddy |  23 | Norway     |  20000
  4 | Mark   |  25 | Rich-Mond |  65000
  5 | David |  27 | Texas      |  85000
  6 | Kim    |  22 | South-Hall|  45000
  7 | James |  24 | Houston    |  10000
```

Now, let's start a transaction and delete records from the table having age = 25 and finally we use ROLLBACK command to undo all the changes.

```
testdb=# BEGIN;
DELETE FROM COMPANY WHERE AGE = 25;
ROLLBACK;
```

If you will check COMPANY table is still having the following records:

```
 id | name   | age | address    | salary
----+--------+-----+------------+--------
  1 | Paul   |  32 | California|  20000
  2 | Allen |  25 | Texas      |  15000
  3 | Teddy |  23 | Norway     |  20000
  4 | Mark   |  25 | Rich-Mond |  65000
  5 | David |  27 | Texas      |  85000
  6 | Kim    |  22 | South-Hall|  45000
  7 | James |  24 | Houston    |  10000
```

Now, let's start another transaction and delete records from the table having age = 25 and finally we use COMMIT command to commit all the changes.

```
testdb=# BEGIN;
DELETE FROM COMPANY WHERE AGE = 25;
COMMIT;
```

If you will check COMPANY table is still having the following records:

```
 id | name  | age | address    | salary
----+-------+-----+------------+--------
  1 | Paul  |  32 | California |  20000
  3 | Teddy |  23 | Norway     |  20000
  5 | David |  27 | Texas      |  85000
  6 | Kim   |  22 | South-Hall |  45000
  7 | James |  24 | Houston    |  10000
(5 rows)
```