Capgemini

PostgreSQL

Lesson 6: PostgreSQL - Transactions, Locks and Security

Lesson Objectives

In this lesson, you will learn about:

- Transactions
- Locks
- Security

Transactions

- Transaction is a unit of work that is performed against a database
- Transactions are sequence of steps in logical order to accomplish some work
- Transaction propagates one or more changes to the database
- Example:
  - if you are creating a record or updating a record from the table, then you are performing transaction on the table
  - It is important to control the transactions to ensure data integrity and to handle database errors
- We group many PostgreSQL queries to be executed together as a part of transaction

Transaction Properties

- Transactions have following standard ACID properties:
  - Atomicity: ensures 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 original state
  - Consistency: ensures that the database properly changes states, when a transaction is committed successfully
  - Isolation: enables transactions to operate independently and are transparent to each other
  - Durability: ensures that the result or effect of committed transaction persists in case of a system failure

Transaction Control

- Commands used to control Transactions:
  - BEGIN TRANSACTION: to start a transaction
  - COMMIT/END TRANSACTION: to save the changes
  - ROLLBACK: to rollback the changes
- Transaction control statements are only used with DML statements – INSERT, UPDATE and DELETE
- They cannot be used while creating or dropping tables as they are auto commit operations

BEGIN TRANSACTION

- Transactions can be started with BEGIN or BEGIN TRANSACTION command
- The transaction persists till next COMMIT or ROLLBACK command
- The transaction will rollback if the database is closed or some error occurs

```
BEGIN;
Or
BEGIN TRANSACTION;
```

COMMIT/ END TRANSACTION

- COMMIT/END TRANSACTION Command
  - 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.

```
COMMIT;
Or
END TRANSACTION;
```

- ROLLBACK Command
  - 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.

```
ROLLBACK;
```

TRANSACTION

- Example:

```
BEGIN;
Delete from emp where deptno=10;
ROLLBACK;
```

- Transactions starts, deletes records for deptno 10, and then ROLLBACK will undo all the ch

```
BEGIN;
Delete from emp where deptno=10;
COMMIT;
```

- Transactions starts, deletes records for deptno 10, and then COMMIT will make changes permanent in the database

Locks

- Locks are used to prevent multiple users from performing concurrent operations on single table
- Rows modified by UPDATE and DELETE are exclusively locked automatically for the duration of transaction
- This prevents other users from changing the row until the transaction is either committed or rolled back
- Other user will have to wait till the transaction is completed to update the same row.
- If they modify different rows, no waiting is necessary.
- SELECT queries never have to wait

Lock command

- Database performs locking automatically
- In certain cases, locking must be controlled manually
- Manual locking can be done by using the LOCK command.
- It allows specification of a transaction's lock type and scope

```
LOCK TABLE table_name IN lock_mode
```

- lock_mode:
  - lock mode specifies which locks this lock conflicts with
  - If no lock mode is specified then it is ACCESS EXCLUSIVE – most restrictive
  - Lock mode values: ACCESS SHARE, ROW SHARE , ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, ACCESS EXCLUSIVE
  - Lock will be held till the transaction completes and are automatically released after the transaction is completed

Dead locks

- Deadlocks can occur when two transactions are waiting for each other to finish their operations
- PostgreSQL can detect them and end them with a ROLLBACK
- Still some times your application may run into deadlock
- To prevent deadlock make sure to design your application in such a way that they will lock objects in the same order

- Advisory Locks
  - PostgreSQL provides a means for creating locks that have application-defined meanings. These are called *advisory locks*
  - a common use of advisory locks is to emulate pessimistic locking strategies typical of so called "flat file" data management systems
  - Advisory locks are faster, and are automatically cleaned up by the server at the end of the session

Locks

- You can lock emp table until transaction ends

```
BEGIN;
LOCK TABLE emp IN ACCESS EXCLUSIVE MODE;
```

- table is locked until the transaction ends and to finish the transaction you will have to either rollback or commit the transaction

Security and Privileges

- PostgreSQL manages database access permissions using users and groups.
- Users own database objects (for example, tables) and can assign privileges on those objects to other users to control who has access
- When you create a database object, you become its owner.
- By default, only the owner of an object has got all permissions on the objects

Security and Privileges

- In order to allow other users to use it, privileges must be granted using GRANT

```
GRANT select, insert ON emp TO user1;
```

- You can remove permissions from user by using REVOKE

```
REVOKE insert ON emp FROM user1;
```

Managing Users

- For creating database users:

```
CREATE user testuser;
```

- For viewing database users:

```
select * from pg_user;
```

- Delete users:

```
Drop user testuser;
```

Managing Users

- For group:

```
CREATE GROUP grp1;
```

- Adding user to group:

```
ALTER GROUP grp1 ADD USER testuser;
```

- To remove user from a group:

```
ALTER GROUP grp1 DROP USER testuser;
```

Demo

Create Transaction

Lab

Lab 6

Summary

In this lesson, you have learn about:

- Transaction is a piece of work that begin with begin BEGIN TRANSACTION STATEMENT
- Transaction completed with COMMIT or Rollback transaction
- Lock prevent multiple users from concurrent access to same record

Review Question

Question 1:Transaction completes when

- END Transaction
- COMMIT
- Rollback
- All of the above

Question 2: Most restrictive lock is:

- ACCESS SHARE,
- ROW SHARE
- ACCESS EXCLISIVE

People matter, results count.

About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2016 global revenues of EUR 12.5 billion.

Visit us at

www.capgemini.com