Class: System Integration
Exercise name: 04b Database Granular Access
Student name: Daniel Causevic

1. Database Overview

The database used for this assignment is PostgreSQL. This section describes the configuration of the database schema, users, and access controls.

1.1 Sample Database

The PostgreSQL server contains a sample database called "EmployeeData", with the table "Employee". The table contains several columns, including:

- emp_id (Primary Key, Auto Increment)
- name (VARCHAR(100), not nullable)
- department (VARCHAR(50), not nullable)
- salary (NUMERIC, not nullable)
- email (VARCHAR(100))

This database setup is designed to allow fine-grained access control for various users based on roles.

1.2 SQL Configuration Overview

```
CREATE DATABASE EmployeeData;

CREATE TABLE EmployeeData.Employee (
    emp_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    department VARCHAR(50) NOT NULL,
    salary NUMERIC NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

1.3 User List

The PostgreSQL server has several users with different permissions on the EmployeeData database:

Username	Password (Redacted)	Permissions
read_only	DLSHomies	Can only read (SELECT)
write_only	DLSHomies	Can only write (INSERT, UPDATE, DELETE)

read_write	DLSHomies	Can read and write (SELECT, INSERT, UPDATE, DELETE)
no_access	DLSHomies	No access (Cannot connect to the database)

---

2. How To

This section provides a step-by-step guide to testing the granular access of PostgreSQL for different users.

Step 1: Install PostgreSQL Client

To interact with the database, ensure you have installed a PostgreSQL client like pgAdmin or psql. Follow the installation instructions for your platform:

- pgAdmin: pgAdmin Downloads
- psql: Install PostgreSQL from the official website PostgreSQL Downloads.

Step 2: Connect to PostgreSQL Server

Using pgAdmin or psql, connect to the PostgreSQL server with the provided server details:

- Host: documentationforsi.postgres.database.azure.com
- Port: 5432
- Database: documentationforsi
- Username and Password: danielcausevic - DLSHomies123

Step 3: Execute Read and Write Operations

After connecting, test the permissions by performing SELECT and INSERT operations for each user role. Execute the following SQL statements:

Insert Operation:

```
INSERT INTO EmployeeData.Employee (name, department, salary, email)
VALUES ('John Doe', 'Engineering', 85000, 'john.doe@example.com');
```

Select Operation:

```
SELECT * FROM EmployeeData.Employee;
```

For each user role, observe the expected results:

- For `read_only`:
  - SELECT: Success (can view data).
  - INSERT: Fail (cannot insert data).
- For `write_only`:
  - SELECT: Fail (cannot view data).

    - INSERT: Success (can insert data).
- For `read_write`:
    - SELECT: Success (can view data).
    - INSERT: Success (can insert data).
- For `no_access`:
    - SELECT: Fail (cannot view data).
    - INSERT: Fail (cannot insert data).

Important: Each SQL statement must be executed separately. If the first one fails (e.g., due to a permission issue), the second one will not execute.

Step 4: Repeat Steps for All Users

Repeat the tests from Step 3 for each of the user roles: read_only, write_only, read_write, and no_access. Ensure that each role behaves as expected, and verify that the permissions are applied correctly.

---

3. Conclusion

This documentation outlines the process for connecting to the PostgreSQL server, testing user roles, and verifying the granular access control in the EmployeeData database. The goal is to ensure that data access is restricted at the appropriate level (read-only, write-only, or both) for each user, with the smallest unit of data access (row, column, or cell) enforced through role-based permissions.

By following these steps, the Integrator should be able to verify the proper functioning of the access control setup. The Exposee is responsible for ensuring that the necessary configuration and documentation are provided, making it possible for the Integrator to complete the assignment without additional assistance.