**Understanding PostgreSQL Roles & Privileges**

**Introduction**

PostgreSQL uses a powerful and flexible system for managing access to database objects. This system is based on **roles**, which can own database objects and have privileges assigned to them. Understanding how roles and privileges work is crucial for setting up secure and efficient access controls in a PostgreSQL database environment.

**1. Roles in PostgreSQL**

**1.1 What is a Role?**

In PostgreSQL, a **role** is an entity that can own database objects and have database privileges. A role can be a **user** or a **group**, depending on how it is used.

* A **user** is a role with the LOGIN attribute.
* A **group** is a role without LOGIN, used to manage privileges across multiple users.

**1.2 Creating Roles**

-- Create a role that can log in (user)

CREATE ROLE analyst LOGIN PASSWORD 'securepassword';

-- Create a role without login (group)

CREATE ROLE data\_team;

**1.3 Altering Roles**

-- Grant login ability

ALTER ROLE data\_team WITH LOGIN;

-- Add a password

ALTER ROLE analyst WITH PASSWORD 'newpassword';

**2. Role Attributes**

PostgreSQL roles can be assigned various attributes:

| **Attribute** | **Description** |
| --- | --- |
| LOGIN | Allows the role to connect to the database |
| SUPERUSER | Grants all privileges (bypasses checks) |
| CREATEDB | Allows creating new databases |
| CREATEROLE | Allows creating, altering, or dropping roles |
| INHERIT | Inherits privileges of roles it is a member of |
| REPLICATION | Allows streaming replication |
| BYPASSRLS | Bypass Row-Level Security policies |

CREATE ROLE admin WITH LOGIN CREATEDB CREATEROLE;

**3. Role Membership**

Roles can be members of other roles, allowing group-based access control.

-- Make analyst a member of data\_team

GRANT data\_team TO analyst;

This allows analyst to inherit privileges granted to data\_team (if INHERIT is enabled).

**4. Privileges in PostgreSQL**

**4.1 Object Types and Applicable Privileges**

| **Object Type** | **Privileges** |
| --- | --- |
| Database | CONNECT, CREATE, TEMPORARY |
| Table | SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER |
| Column | SELECT, INSERT, UPDATE |
| Sequence | USAGE, SELECT, UPDATE |
| Schema | CREATE, USAGE |
| Function | EXECUTE |
| Language | USAGE |

**4.2 Granting Privileges**

-- Grant SELECT on a table to a role

GRANT SELECT ON customers TO analyst;

-- Grant all privileges on a table

GRANT ALL PRIVILEGES ON orders TO admin;

**4.3 Revoking Privileges**

-- Revoke a privilege

REVOKE INSERT ON customers FROM analyst;

**5. Default Privileges**

You can set default privileges for future objects created by a role:

-- Set default privileges in a schema

ALTER DEFAULT PRIVILEGES IN SCHEMA sales

GRANT SELECT, INSERT ON TABLES TO data\_team;

**7. Role Management Best Practices**

**7.1 Principle of Least Privilege**

Assign only the minimum privileges required for users to perform their tasks.

**7.2 Use Group Roles**

Create group roles to manage privileges for multiple users efficiently.

**7.3 Avoid SUPERUSER**

Restrict use of SUPERUSER; instead, assign only specific privileges.

**7.4 Audit Privileges Regularly**

Review and audit roles and privileges periodically.

**8. Practical Example**

-- Create roles

CREATE ROLE readonly;

CREATE ROLE readwrite;

-- Grant privileges to roles

GRANT CONNECT ON DATABASE mydb TO readonly, readwrite;

GRANT USAGE ON SCHEMA public TO readonly, readwrite;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonly;

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO readwrite;

-- Create users

CREATE ROLE alice LOGIN PASSWORD 'pass123';

CREATE ROLE bob LOGIN PASSWORD 'pass456';

-- Assign roles

GRANT readonly TO alice;

GRANT readwrite TO bob;

**9. Viewing Roles and Privileges**

**9.1 List Roles**

\du

**9.2 List Role Attributes**

SELECT rolname, rolsuper, rolcreaterole, rolcreatedb, rolcanlogin

FROM pg\_roles;

**9.3 Check Table Privileges**

\z table\_name

**Conclusion**

PostgreSQL roles and privileges offer a robust access control mechanism suitable for both small-scale and enterprise-grade applications. Proper use of roles, role attributes, and privilege management can significantly enhance database security, maintainability, and scalability.