



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David



POSTGRESQL SECURITY & ADMINISTRATION

Week:
24-Nov-2023

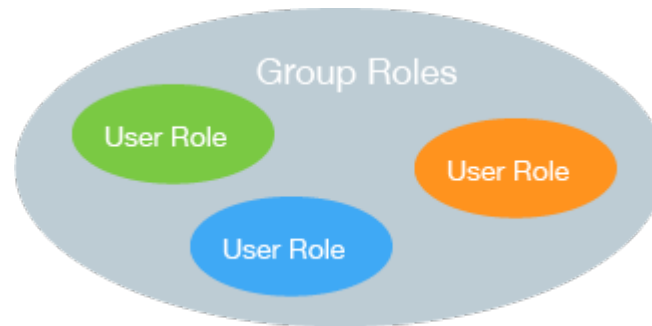
Security and Administration: Database Systems (DBS)
Dr Suresh Shanmugasundaram

PostgreSQL Roles

- ❑ PostgreSQL uses roles to represent user accounts.
- ❑ It doesn't use the user concept like other database systems.
- ❑ Typically, roles can log in are called login roles.
- ❑ They are equivalent to users in other database systems.

Group Roles

When roles contain other roles,
they are call group roles.



Create Roles

- ❑ To create a new role, you use the CREATE ROLE statement as follows:
 - ❑ CREATE ROLE role_name;
- ❑ When you create a role, it is valid in all databases in the database server (or cluster).
- ❑ The following statement uses the CREATE ROLE statement to create a new role called stephen:
 - ❑ CREATE ROLE stephen;

Check System Catalog for Roles

- ❑ To get all roles in the current PostgreSQL database server, you can query them from the pg_roles system catalog as follows:
 - ❑ `SELECT rolname FROM pg_roles;`
 - ❑ Notice that the roles that start with `pg_` are system roles.

Data output		Messages	Notifications
       			
	rolname name		
1	pg_database_owner		
2	pg_read_all_data		
3	pg_write_all_data		
4	pg_monitor		
5	pg_read_all_settings		
6	pg_read_all_stats		
7	pg_stat_scan_tables		
8	pg_read_server_files		
9	pg_write_server_files		
10	pg_execute_server_program		
11	pg_signal_backend		
12	postgres		
13	stephen		

List All Existing Roles

- ❑ If you use the psql tool, you can use the \du command to list all existing roles in the current PostgreSQL database server.

- ▣ \du

```
postgres=# psql -U hazel -W postgres
postgres=# \du
```

List of roles
Attributes

Role name		Member of
hazel		{}
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	{}
stephen	Cannot login	{}

Role Attributes

- ❑ The attributes of a role define privileges for that role including login, superuser, database creation, role creation, password, etc:
 - ▣ CREATE ROLE name WITH option;
 - ▣ Code language: PostgreSQL SQL dialect and PL/pgSQL (pgsql)
- ❑ In this syntax, the WITH keyword is optional. And the option can be one or more attributes including SUPER, CREATEDB, CREATEROLE, etc.

Create Login Roles

- ❑ The following statement creates a role called hazel that has the login privilege and an initial password:
 - ❑ CREATE ROLE hazel
 - ❑ LOGIN
 - ❑ PASSWORD 'my123abc';

Create SuperUser Roles

- ❑ The following statement creates a role called alyson that has the superuser attribute.
 - ❑ CREATE ROLE alyson
 - ❑ SUPERUSER
 - ❑ LOGIN
 - ❑ PASSWORD 'my123abc';
 - ❑ Code language: SQL (Structured Query Language) (sql)
- ❑ The superuser can override all access restrictions within the database therefore you should create this role only when needed.
- ❑ Notice that you must be a superuser in order to create another superuser role.

Role Login

- ❑ Now, you can use the role `hazel` to log in to the PostgreSQL database server using the `psql` client tool:
 - ▣ `psql -U hazel -W postgres`
 - ▣ Code language: Shell Session (shell)
- ❑ It will prompt you for a password.
- ❑ You need to enter the password that you entered in the `CREATE ROLE` statement.

Create Roles that can Create Databases

- If you want to create roles that have the database creation privilege, you use the CREATEDB attribute:
 - ▣ CREATE ROLE mydba
 - ▣ CREATEDB
 - ▣ LOGIN
 - ▣ PASSWORD 'town1234';

Create Roles with Validity Period

- To set a date and time after which the role's password is no longer valid, you use the valid until attribute:
 - ▣ For example, the following statement creates a dev_api role with password valid until the end of 2029:
 - ▣ CREATE ROLE java_dev_api WITH
 - ▣ LOGIN
 - ▣ PASSWORD 'town123'
 - ▣ VALID UNTIL '2024-01-01';
 - ▣ After one second tick in 2024, the password of java_dev_api is no longer valid.

Create Roles with Connection Limit

- ❑ The following creates a new role called api that can make 1000 concurrent connections:
- ❑ `CREATE ROLE java_api`
- ❑ `LOGIN`
- ❑ `PASSWORD 'town1234'`
- ❑ `CONNECTION LIMIT 10;`

List All Roles

- ❑ The following psql command shows all the roles that we have created so far:
 - ❑ \du

```
postgres=# \du
```

List of roles
Attributes

Role name		Member of
alyson	Superuser	{}
hazel		{}
java_api	10 connections	{}
mydba	Create DB	{}
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	{}
stephen	Cannot login	{}

Role Summary

- ❑ PostgreSQL uses roles to represent user accounts.
- ❑ A role that can login is equivalent to a user account in other database systems.
- ❑ Use role attributes to specify the privileges of the roles e.g., LOGIN allows the role to login, CREATEDB allows the role to create a new database, SUPERUSER allows the role to have all privileges.

PostgreSQL Granting Rights to Roles

- ❑ After creating a role with the LOGIN attribute, the role can log in to the PostgreSQL database server.
- ❑ However, it cannot do anything to the database objects like tables, views, functions, etc.
 - ▣ For example, the user role cannot select data from a table or execute a specific function.
 - ▣ To allow the user role to interact with database objects, you need to grant privileges on the database objects to the user role by using the GRANT statement.

Grant Statement

- ❑ The following shows the simple form of the GRANT statement that grants one or more privileges on a table to a role:
 - ▣ GRANT privilege_list | ALL
 - ▣ ON table_name
 - ▣ TO role_name;
- ❑ Specify the privilege_list that can be SELECT, INSERT, UPDATE, DELETE, TRUNCATE, etc.
- ❑ You use the ALL option to grant all privileges on a table to the role.

Sample Grant Statements

- ❑ Create a table named student:
 - ❑ create table student (
 - ❑ student_id int generated always as identity,
 - ❑ first_name varchar(100) not null,
 - ❑ last_name varchar(100) not null,
 - ❑ email varchar(255) not null unique,
 - ❑ phone varchar(25) not null,
 - ❑ primary key(student_id)
 - ❑);

Sample Grant Statements

- ❑ Use the role `hazel` to log in to the PostgreSQL database server in a separate session.
- ❑ Next, attempt to select data from the `candidates` table from the `hazel`'s session:
 - ❑ `SELECT * FROM student;`
 - ❑ PostgreSQL issued an error:
 - ❑ `ERROR: permission denied for table student.`

Sample Grant Statements

- ❑ To grant the SELECT privilege on the candidates table to the role hazel, you execute the following GRANT statement in the postgres' session:
 - ❑ GRANT SELECT
 - ❑ ON student
 - ❑ TO hazel;
- ❑ Execute the SELECT statement from the hazel's session:
 - ❑ SELECT * FROM candidates;
 - ❑ PostgreSQL returns an empty result set instead of an error.

Insert a Student Record

- ❑ Next
 - ❑ `INSERT INTO student(first_name, last_name, email, phone)`
 - ❑ `VALUES('Amy','Morgan','amy.morgan@myprovider.com','01792-123456');`
- ❑ PostgreSQL issued the following error because hazel does not have the INSERT privilege on the candidates table:
 - ❑ `ERROR: permission denied for table candidates`

Grant Insert, Update & Delete Rights

- ❑ Next, grant INSERT, UPDATE, and DELETE privileges on the candidates table to the role hazel:
 - ❑ GRANT INSERT, UPDATE, DELETE
 - ❑ ON student
 - ❑ TO hazel;

Grant Insert, Update & Delete Rights

□ Next:

- INSERT INTO student(first_name, last_name, email, phone)
- VALUES('Amy','Morgan','amy.morgan@myprovider.com','01792-123456');

□ Finally:

- SELECT * FROM student;

Data output Messages Notifications					
       					
	student_id [PK] integer 	first_name character varying (100) 	last_name character varying (100) 	email character varying (255) 	phone character varying (25) 
1	1	Amy	Morgan	amy.morgan@myprovi...	01792-123456

Grant All Privileges on a Table to a Role

- ❑ The following statement grants all privileges on the candidates table to the role hazel:
 - ❑ GRANT ALL
 - ❑ ON student
 - ❑ TO hazel;

Grant All Privileges on all Tables in a Schema to a Role

- ❑ The following statement grants all privileges on all tables in the public schema of the dvdrental sample database to the role hazel:
 - ❑ GRANT ALL
 - ❑ ON ALL TABLES
 - ❑ IN SCHEMA "public"
 - ❑ TO hazel;

Grant SELECT on All Tables

- ❑ In order to do that, you can grant SELECT privilege on all tables in the public schema like this:
 - ❑ GRANT SELECT
 - ❑ ON ALL TABLES
 - ❑ IN SCHEMA "public"
 - ❑ TO java_api;

Revoke Privileges

- ❑ The REVOKE statement revokes previously granted privileges on database objects from a role.
- ❑ The following shows the syntax of the REVOKE statement that revokes privileges on one or more tables from a role:

```
-- Revoking Privileges  
REVOKE privilege | ALL  
ON TABLE table_name | ALL TABLES IN SCHEMA schema_name  
FROM role_name;
```

Revoke Privileges

- ❑ First, specify the one or more privileges that you want to revoke. You use the ALL option to revoke all privileges.
- ❑ Second, specify the name of the table after the ON keyword. You use the ALL TABLES to revoke specified privileges from all tables in a schema.
- ❑ Third, specify the name of the role from which you want to revoke privileges.

Revoke Privileges

- Assume the Role and Grants Statements as below:

```
-- creates a role called hazel that has the login  
-- privilege and an initial password:  
CREATE ROLE hazel  
LOGIN  
PASSWORD 'my123abc';
```

```
-- Grant All Privileges on a Table to a Role  
GRANT ALL  
ON student  
TO hazel;
```

Revoke Privileges

- ❑ Revoking Hazel's Update Permissions From the student table

```
-- Revoking Hazel's Update Permissions  
-- From the student table  
REVOKE UPDATE  
ON student  
FROM hazel
```

Summary Privileges List

Privileges	Description
SELECT	Ability to execute SELECT operators for a table.
INSERT	Ability to execute INSERT operators for a table.
UPDATE	Ability to execute UPDATE operators for a table.
DELETE	Ability to execute DELETE operators for a table.
TRUNCATE	Ability to execute TRUNCATE operators on a table.
REFERENCES	Ability to create external keys (requires privileges for both parent and child tables).
TRIGGER	Ability to create triggers for a table.
CREATE	Ability to perform CREATE TABLE operators.
ALL	Provides all privileges.