CREATE ROLE to Create Groups

by Sophia

WHAT'S COVERED

This lesson explores using the CREATE ROLE to create groups in a database, in two parts. Specifically, this lesson will cover:

1. Introduction

As you learned in the previous lesson, the CREATE ROLE statement can be used to create a user or a group, depending on the parameters you use. This is something unique to PostgreSQL, as other databases typically have different statements for creating users and groups. The previous lesson explained how to create a login role for a user, with the LOGIN attribute and a password. In contrast, group roles do not have a LOGIN attribute or a password. That's because group roles are not for logging in (that is, they are not for authentication). Instead, they are for accessing resources; users assigned to the group will inherit any privileges that the group has.

2. Exploring the CREATE ROLE Attributes

You have already seen the CREATE ROLE statement at work and learned about a few attributes, but let's take a deeper look at it now, including some additional attributes. The structure of the CREATE ROLE command looks like this:

```
CREATE ROLE <rolename>
<WITH>
[SUPERUSER]
[CREATEDB]
[CREATEROLE]
[INHERIT]
```

```
[LOGIN]
[CONNECTION LIMIT] [VALID UNTIL]
[IN ROLE];
```

As you learned in the previous section, adding GROUP after the role name makes it a group role to which you can assign users.

Notice the various attributes in square brackets. As you learned in the previous lesson, this statement offers a variety of special-purpose options you can include. They are all optional.

HINT

The angle brackets and square brackets in the syntax shown do not appear in actual usage.

For each of the attributes, there is also a NO version that negates that attribute's use. For example, SUPERUSER gives a user account the ability to override all access restrictions, and NOSUPERUSER denies the user that ability.

Perhaps you are wondering why you would need to deny an ability; why wouldn't you just not assign it in the first place? It becomes needed when one of those attributes is applied to a group. All the users in that group automatically have that attribute too. If you want someone to be in the group but not have that attribute, you can assign the "NO" version of the attribute to the specific user account.

We looked at the SUPERUSER, CREATEDB, VALID UNTIL, and LOGIN options in the prior lesson. As a reminder, the SUPERUSER is one that has the ability to override all access restrictions. It can be dangerous, as the role can drop any objects or access any data. The account must be a SUPERUSER to create a role using the SUPERUSER attribute.

The CREATEDB attribute just defines if the role is able to create databases. Typically, a database administrator role would benefit from this attribute.

The CREATEROLE attribute allows the user to create new, ALTER, and DROP other roles.

The INHERIT attribute enables the role to inherit the privileges of group roles that it is a member of. INHERIT is the default, so you would not typically use it. What you might use, however, is NOINHERIT, which will prevent inheritance

The CONNECTION LIMIT determines how many concurrent connections the role can make. The default is set to -1, meaning that there is no limit.

IN ROLE will list one or more existing roles in which the new role would be immediately added as a member. This could be a login role or another group role. For example, you could have an executive role that is set up to be part of a management role so that the executive role will get all of the permissions that the management role has.

There are other attributes that can be used with the CREATE ROLE that are very specific to a given scenario that we will not cover.

We can use the different attributes together in a single CREATE ROLE statement. For example, you could have created an admin role that can create databases and roles:

```
CREATE ROLE adminrole
CREATEDB
CREATEROLE;
```

This would create a group role called adminrole.

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

In this lesson's **introduction**, you learned that the CREATE ROLE command is a powerful tool for managing user access and permissions within the database system. This command allows administrators to define roles with specific privileges and attributes, effectively grouping permissions based on user roles or responsibilities. A role can represent an individual user, a group, or even an application, enabling efficient and organized access control. With the CREATE ROLE command in PostgreSQL, administrators can specify login options, password settings, and membership in other roles. You **explored the CREATE ROLE attributes** that admins can use for different scenarios, whether they want to create read-only roles, grant access to specific tables, or manage users with special privileges. Once roles are defined, users can grant them, nest them within other roles, or use them to enforce security policies in combination with other access control mechanisms. PostgreSQL's CREATE ROLE command enables you to manage access rights versatilely and robustly. Streamlining permission management and enhancing data security are both resulting benefits.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND FAITHE WEMPEN (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.