For Your Eyes Only: Roles, Privileges, and Security in PostgreSQL

Ryan Booz
FOSDEM Postgres Devroom 2024
github.com/ryanbooz/presentations
Agenda

01 The Building Blocks
02 Roles
03 Special Roles
04 Privileges
05 Inheritance
06 Object Ownership
07 Predefined Roles
Disclaimer(s)
We won't cover everything
Content is applicable to currently supported versions of PostgreSQL (12+)

(and most of it for 9.6+)
01/07
The Building Blocks
Cluster

Server/Host (Firewall, Ports)

- Cluster
  Port: 5432
  `pg_hba.conf`

- Cluster
  Port: 5433
  `pg_hba.conf`

- Cluster
  Port: 5434
  `pg_hba.conf`
Cluster

ROLES

Databases
First layer of authentication

Similar to a firewall ruleset for PostgreSQL

Cloud vendors largely manage this for you

Which hosts & roles, can connect to what databases, using what authentication method?
# Allow any user on the local system to connect to any database with any database user name using Unix-domain sockets (the default for local connections).

# TYPE  DATABASE  USER  ADDRESS  METHOD
local   all       all       all       trust

# The same using local loopback TCP/IP connections.

# TYPE  DATABASE  USER  ADDRESS  METHOD
host    all       all       127.0.0.1/32 trust

# Allow any user from host 192.168.12.10 to connect to database "postgres" if the user's password is correctly supplied.

# TYPE  DATABASE  USER  ADDRESS  METHOD
host    postgres  all       192.168.12.10/32 scram-sha-256

https://www.postgresql.org/docs/current/auth-pg-hba-conf.html
Avoid using 'TRUST' method at all costs!*

*(can be useful for local development machines... but still...)*
Use scram-sha-256 for password authentication
02/07
Roles
Roles

- Own databases, schemas, and objects
  - Tables, Functions, Views, Etc.
- Have cluster-level privileges (attributes)
- Granted privileges to databases, schemas, and objects
- Can possibly grant privileges to other roles
Users and Groups

- Semantically the same as roles
- By Convention:
  - User = LOGIN
  - Group = NOLOGIN
- PostgreSQL 8.2+ `CREATE (USER|GROUP)` is an alias
CREATE USER user1 WITH PASSWORD 'abc123' INHERIT;

CREATE GROUP group1 WITH INHERIT;

CREATE ROLE user1 WITH LOGIN PASSWORD 'abc123' INHERIT;
Role Attributes

- Predefined settings that can be enabled/disabled for a given role
- Essentially cluster-level (non-database) privileges
- Map to columns in `pg_catalog.pg_roles`
PostgreSQL15 Attributes

LOGIN
SUPERUSER
CREATEROLE
CREATEDB
REPLICATION LOGIN

PASSWORD
INHERIT
BYPASSRLS
CONNECTION LIMIT
Unless otherwise set, new roles can **INHERIT** privileges from other roles and have unlimited connections.
Role Specific Session Settings

• Roles can set role-specific defaults for run-time configuration at connection time

• Any settings that can be set via SET command can be altered for a ROLE

```
ALTER ROLE user1 SET jit TO off;

ALTER ROLE user1 RESET jit;
```
03/07 Special Roles
PostgreSQL Superuser
PostgreSQL Superuser
PostgreSQL Superuser

• is created by default when the cluster is initialized
• Typically named `postgres` because the system process user initiates a `initdb`
• Bypasses all security checks except `LOGIN`
• Full privilege to do "anything"
• Treat superuser with care (like `root` on Linux)
Most cloud providers do not provide superuser access
Superuser-like
Superuser-like

- Create a role with the right level of control
- Recommend adding `CREATEROLE` and `CREATEDB`
- Allows user management and database ownership
- May still limit some actions (e.g. installing extensions limited to superuser)
04/07
Privileges
Privileges

- The set of access rights to databases, schemas, and objects
- Can be granted (GRANT) or revoked (REVOKE) by a role with authority
- Explicit GRANT or REVOKE only impacts existing objects
PostgreSQL 15 Privileges

<table>
<thead>
<tr>
<th>Action</th>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>CREATE</td>
</tr>
<tr>
<td>INSERT</td>
<td>CONNECT</td>
</tr>
<tr>
<td>UPDATE</td>
<td>TEMPORARY</td>
</tr>
<tr>
<td>DELETE</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>TRUNCATE</td>
<td>USAGE</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>SET</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>ALTER SYSTEM</td>
</tr>
</tbody>
</table>
PUBLIC Role

• All roles are granted implicit membership to PUBLIC
• The public role cannot be deleted
• Granted CONNECT, USAGE, TEMPORARY, and EXECUTE by default
• <=PG14: CREATE on the public schema by default
• >=PG15: No CREATE on public schema by default
Security Best Practice for PUBLIC

- Revoke all privileges on the public schema from the PUBLIC role
- Revoke all database privileges from the PUBLIC role (maybe)

```sql
REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON DATABASE db_name FROM PUBLIC;
```
Granting Privileges

-- grant the ability to create a schema
GRANT CREATE ON DATABASE app_db TO admin1;

-- see and create objects in schema
GRANT USAGE,CREATE IN SCHEMA demo_app TO dev1;

-- allow some roles only some privileges
GRANT SELECT, INSERT, UPDATE
ON ALL TABLES IN SCHEMA demo_app TO jr_dev;
Granting Privileges

- Remember, explicit grants only effect existing database objects!

```sql
-- This will only grant to existing objects
GRANT ALL TO ALL TABLES IN SCHEMA public TO dev1;
```
More Detail on GRANT and REVOKE

What the privileges mean:
https://www.postgresql.org/docs/current/ddl-priv.html

How to GRANT privileges:
https://www.postgresql.org/docs/current/sql-grant.html

How to REVOKE privileges:
https://www.postgresql.org/docs/current/sql-revoke.html
05/07
Inheritance
Privilege Inheritance

• Roles can be granted membership into another role
• If a role has INHERIT set, they automatically have usage of privileges from member roles
• The preferred method for managing group privileges
CREATE ROLE sr_dev WITH LOGIN password='abc' INHERIT;
CREATE ROLE rptusr WITH LOGIN password='123' INHERIT;
CREATE ROLE admin WITH NOLOGIN NOINHERIT;
CREATE ROLE ropriv WITH NOLOGIN NOINHERIT;

GRANT INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA app TO admin;
GRANT SELECT ON ALL TABLES IN SCHEMA app TO ropriv;

GRANT admin, ropriv TO sr_dev;
GRANT ropriv TO rptusr;
Table access on 'app' schema

- ropriv
  - rptusr
    - SELECT
  - sr_dev
    - SELECT
Table access on 'app' schema

- **ropriv**
  - rptusr
    - SELECT

- **admin**
  - sr_dev
    - SELECT, INSERT, UPDATE, DELETE
Object Ownership

• Object creator = owner
• Owner is a "superuser" of the objects they own
• Initial object access = Principle of Least Privilege
  • Unless specifically granted ahead of time, objects are owned and "accessible" by the creator/superuser only
• Roles can specify default privileges to Grant for each object type that they create
Default Privileges

```sql
ALTER DEFAULT PRIVILEGES
GRANT SELECT ON TABLES TO public;

cituscon=> \ddp

<table>
<thead>
<tr>
<th>Owner</th>
<th>Schema</th>
<th>Type</th>
<th>Access privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>postgres</td>
<td></td>
<td>table</td>
<td>=r/postgres</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>postgres=arwdDxt/postgres</td>
</tr>
</tbody>
</table>
```

Providing Object Access

Option 1: (owner)
Explicitly GRANT access after object creation

Option 2: (owner)
ALTER DEFAULT PRIVILEGES

Option 3: 
SET ROLE to app role before creation with correct default privileges

Option 4: (PG14+)
Use pg_read_all_data or pg_write_all_data predefined roles
Object Ownership Security

- CREATE OR REPLACE doesn't change ownership
- Security issue with users that have create permissions (particularly the public schema)
- PostgreSQL 15 removes default create permissions from PUBLIC on the public schema
DEMO
07/07
Predefined Roles
Predefined Roles

• Cluster-level roles that can be granted
• Work starting in PostgreSQL 14+ to simplify privilege management
  • `pg_read_all_data` (for example)
    • If a role that has `CONNECT` to a database, they can `SELECT` from all tables
### Table 22.1. Predefined Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Allowed Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>pg_read_all_data</td>
<td>Read all data (tables, views, sequences), as if having SELECT rights on those objects, and USAGE rights on all schemas, even without having it explicitly. This role does not have the role attribute BYPASSRLS set. If RLS is being used, an administrator may wish to set BYPASSRLS on roles which this role is GRANTED to.</td>
</tr>
<tr>
<td>pg_write_all_data</td>
<td>Write all data (tables, views, sequences), as if having INSERT, UPDATE, and DELETE rights on those objects, and USAGE rights on all schemas, even without having it explicitly. This role does not have the role attribute BYPASSRLS set. If RLS is being used, an administrator may wish to set BYPASSRLS on roles which this role is GRANTED to.</td>
</tr>
<tr>
<td>pg_read_all_settings</td>
<td>Read all configuration variables, even those normally visible only to superusers.</td>
</tr>
<tr>
<td>pg_read_all_stats</td>
<td>Read all pg_stat_* views and use various statistics related extensions, even those normally visible only to superusers.</td>
</tr>
<tr>
<td>pg_stat_scan_tables</td>
<td>Execute monitoring functions that may take ACCESS SHARE locks on tables, potentially for a long time.</td>
</tr>
<tr>
<td>pg_monitor</td>
<td>Read/execute various monitoring views and functions. This role is a member of pg_read_all_settings, pg_read_all_stats and pg_stat_scan_tables.</td>
</tr>
<tr>
<td>pg_database_owner</td>
<td>None. Membership consists, implicitly, of the current database owner.</td>
</tr>
<tr>
<td>pg_signal_backend</td>
<td>Signal another backend to cancel a query or terminate its session.</td>
</tr>
<tr>
<td>pg_read_server_files</td>
<td>Allow reading files from any location the database can access on the server with COPY and other file-access functions.</td>
</tr>
<tr>
<td>pg_write_server_files</td>
<td>Allow writing to files in any location the database can access on the server with COPY and other file-access functions.</td>
</tr>
<tr>
<td>pg_execute_server_program</td>
<td>Allow executing programs on the database server as the user the database runs as with COPY and other functions which allow executing a server-side program.</td>
</tr>
<tr>
<td>pg_checkpoint</td>
<td>Allow executing the CHECKPOINT command.</td>
</tr>
<tr>
<td>pg_use_reserved_connections</td>
<td>Allow use of connection slots reserved via reserved_connections.</td>
</tr>
<tr>
<td>pg_create_subscription</td>
<td>Allow users with CREATE permission on the database to issue CREATE SUBSCRIPTION.</td>
</tr>
</tbody>
</table>

[https://www.postgresql.org/docs/current/predefined-roles.html](https://www.postgresql.org/docs/current/predefined-roles.html)
What Questions do you have?
THANK YOU! 🎉

github.com/ryanbooz/presentations