Introduction to access control

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

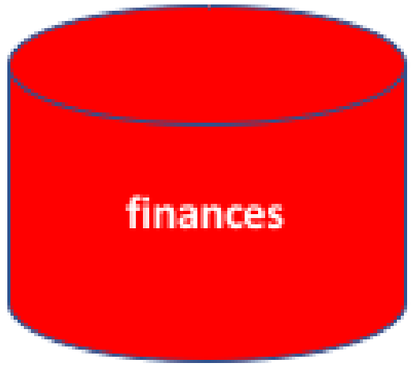
datacamp

The default superuser

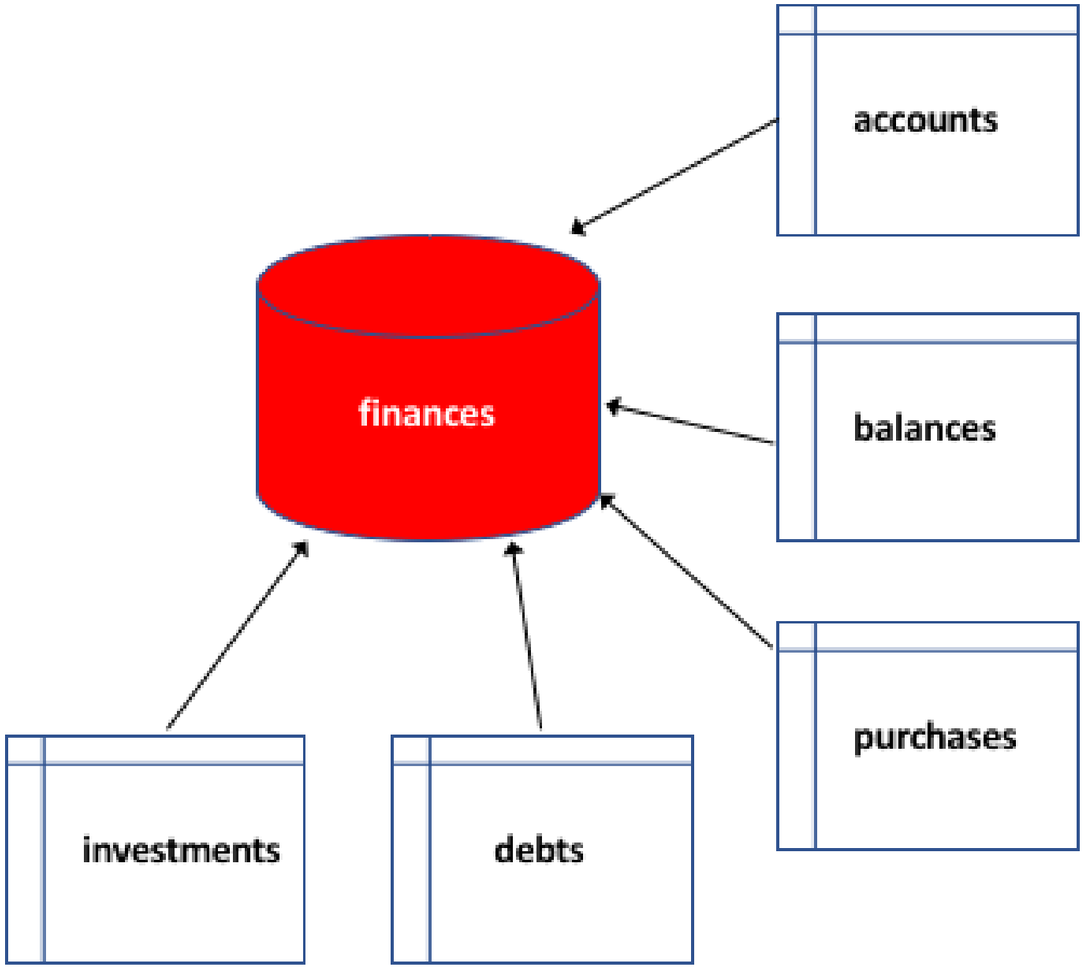
- `postgres` "superuser" role
- Administers database
- `postgres` privileges
  - Creating databases
  - Dropping databases
  - Inserting records
  - Deleting records
  - Dropping tables
- `postgres` user should be used with care

Example: a personal finance database

- Creation of `finances` database



Example: a personal finance database



Example: a personal finance database

- Database is personal and not publicly accessible
- User with restricted access should be created
- User abilities:
  - Adding records
  - Querying records
  - Editing records

Creating new users

- `CREATE USER`
  - Used to generate a new account

```
CREATE USER newuser;
```

- `newuser` can create tables in database
- No access to tables created by other users

Setting user password

- Passwords enhance security
- No passwords by default

```
CREATE USER newuser WITH PASSWORD 'secret';
```

```
ALTER USER newuser WITH PASSWORD 'new_password';
```

Let's practice!

CREATING POSTGRESQL DATABASES

PostgreSQL access privileges

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

PostgreSQL roles and privileges

- Users are a type of role
- Group roles can also be defined
- Database object access given to roles

The GRANT command

- Privileges are "granted" to roles by owner
- The `GRANT` command bestows privileges
- Many privileges can be granted including:
  - `SELECT`
  - `DELETE`
  - `UPDATE`

```
GRANT p ON obj TO grantee;
```

Example: personal finance database

```
CREATE TABLE account (
    id SERIAL PRIMARY KEY,
    short_name VARCHAR(25),
    provider_id INTEGER REFERENCES provider(id),
    balance DECIMAL
);
```

```
CREATE USER fin WITH PASSWORD '38\5)uk1+3&}*Y';
```

Example: personal finance database

- `fin` user needs access to `account` table
- `fin` access
  - Add new accounts
  - Update accounts
  - Query accounts
- Superuser grants privileges

```
GRANT INSERT ON account TO fin;
```

```
GRANT UPDATE ON account TO fin;
```

```
GRANT SELECT ON account TO fin;
```

Table modification privileges

- Some privileges cannot be granted
- Modifying table requires ownership

```
ALTER TABLE account ADD COLUMN date_opened DATE;
```

```
ALTER TABLE account RENAME COLUMN short_name
TO nickname;
```

```
ALTER TABLE account OWNER TO fin;
```

Let's practice!

CREATING POSTGRESQL DATABASES

Hierarchical access control

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Access control with schemas

- Schema - named container for db objects
- Schemas can be used for access control

Example: schema use in finances database

- Spouse access to `finances` database
- `public` schema used by default
- Two new schemas: `me` and `spouse`

```
CREATE SCHEMA me;
```

```
CREATE SCHEMA spouse;
```

```
CREATE TABLE me.account (...);
```

```
CREATE TABLE spouse.account (...);
```

Granting schema privileges

```
CREATE USER better_half WITH PASSWORD 'changeme';
```

```
GRANT USAGE ON SCHEMA spouse TO better_half;
```

```
GRANT USAGE ON SCHEMA public TO better_half;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA spouse;
TO better_half;
```

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

Schema-based access control implemented

Using groups

- Group - a type of role that identifies one or more users
- Access control can be applied at group level

```
CREATE GROUP family;
```

```
GRANT USAGE ON SCHEMA public TO family;
```

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

```
ALTER GROUP family ADD USER fin
ALTER GROUP family ADD USER better_half;
```

Shared and individual data access

- Shared schema access enabled to `public` schema
- Individual schemas control data access

Let's practice!

CREATING POSTGRESQL DATABASES

Removing access

CREATING POSTGRESQL DATABASES

Darryl Reeves
Assistant Professor, Long Island University - Brooklyn

Example: rolling back privileges

- Cousin interested in databases
- Superuser access mistakenly provided
- Good backup strategy saves the day
- New user account added

```
CREATE USER cousin;
```

```
ALTER GROUP family ADD USER cousin;
```

```
GRANT ALL PRIVILEGES ON finances.* TO cousin;
```

`finances` data deleted again

Example: rolling back privileges

- Privileges removed using `REVOKE` command
- `REVOKE` follows similar format to `GRANT`

```
REVOKE DELETE, TRUNCATE ON finances.* FROM cousin;
```

Example: rolling back privileges

- Privileges can be reset

```
REVOKE ALL PRIVILEGES ON finances.* FROM cousin;
```

```
GRANT SELECT ON finances.* FROM cousin;
```

- `REVOKE` can remove users from groups

```
REVOKE family FROM cousin;
```

Let's practice!

CREATING POSTGRESQL DATABASES

Course wrap-up

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Course content

Chapter 1: Structure of PostgreSQL Databases

Chapter 2: PostgreSQL Data Types

Chapter 3: Database Normalization

Chapter 4: Access Control in PostgreSQL

Next steps

- Database objects (e.g. views and functions)
- Data types (e.g. geometric and array-based)
- Normalization (e.g 4NF)
- Access control

Congratulations!

CREATING POSTGRESQL DATABASES