PSQL DB Security

Subject: PostgreSQL
Students: BOURBAI Ismail

CREATE ROLE:

PostgreSQL uses the roles concept to manage database access permissions. A role can be a user or a group, depending on how you setup the role. A role that has login right is called user. A role may be a member of other roles, which are known as groups.
To create a new role, you use the `CREATE ROLE` statement as follows:

```
postgres=# CREATE ROLE ismail LOGIN password 'secret'; -- Create simple role name it ismail with paswword secret
CREATE ROLE
postgres=# CREATE ROLE admin WITH CREATEDB CREATEROLE; -- Create and admin how can create databases and create roles
CREATE ROLE
postgres=# select rolname from pg_roles;
          rolname
--------------------------
 pg_monitor
 pg_read_all_settings
 pg_read_all_stats
 pg_stat_scan_tables
 pg_read_server_files
 pg_write_server_files
 pg_execute_server_program
 pg_signal_backend
 postgres
 tom
 ismail
 admin
(12 rows)
```

Figure 1 - Create new Role

Make **ismail** the **owner** of test_base database:

```
ALTER DATABASE test_base OWNER TO ismail;
```

Now the user ismail can connect to test_base:

```
C:\
λ psql -U ismail test_base
Password for user ismail:
psql (11.1)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.
```

Figure 2 - Connect With User

Privileges:

When an object is created, it is assigned an owner. The owner is normally the role that executed the creation statement. For most kinds of objects, the initial state is that only the owner (or a superuser) can do anything with the object. To allow other roles to use it, *privileges* must be granted.
There are different kinds of
privileges: SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE, and USAGE. The privileges applicable to a particular object vary depending on the object's type (table, function, etc). For complete information on the different types of privileges supported by PostgreSQL.

To assign privileges, the GRANT command is used. For example, if “ismail” is an existing role, and “test_table” is an existing table, the privilege to update the table can be granted with:

```
C:\
λ psqlauth
Password for user postgres:
psql (11.1)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \c test_base
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
You are now connected to database "test_base" as user "postgres".
test_base=# GRANT UPDATE ON test_table TO ismail;
GRANT
test_base=#
```

Figure 3 - GRANT example

To revoke a privilege, use `REVOKE` command:

- Revoke all privilege:
  `REVOKE ALL PRIVILEGES ON test_table FROM ismail;`
- Revoke specific privilege from the public (for ex: INSERT):
  `REVOKE INSERT ON test_table FROM PUBLIC;`