Managing PostgreSQL on Windows/Ubuntu

Outline

1. The pgAdmin
2. Practices – Create a new database
3. Access to database objects: Login/Group Roles
4. Backup and restore

- localhost
- Port: 5432
- Account: postgres
- Password: admin

3. Access to database objects: Login/Group Roles

Roles

- Actually, we use postgres account to connect to PostgreSQL system: super role having all privileges.
- PostgreSQL uses the **roles concept** to manage database access permissions. A role can be
  - **a user**: a role that has login right is called user or login role
  - or a group: a role may be a member of other roles, which are known as groups
- **Each database user** should have **an individual account** for logging into the PostgreSQL system
- **pgAdmin** allows you to create Roles and to grant Roles access to database objects

Group Roles

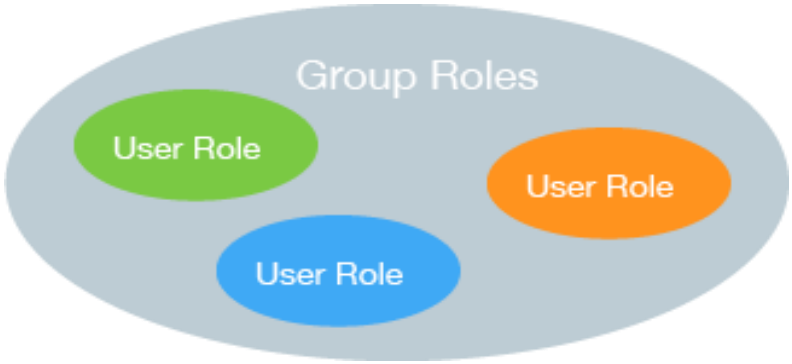
- Create access **permissions for groups of users**
- While you can grant an individual user account access directly to a database object, the preferred method is to use Group Roles
- allow you to **easily change access for database objects** without having to touch hundreds (or even thousands) of individual user.

- Default, **public** group role:
  - applies to all users on the PostgreSQL system
  - NOT able to remove any user account from the public Group Role
  - does not appear in the pgAdmin Group Roles listing

Login Roles (or user accounts)

- Are roles that are allowed to log into the PostgreSQL server

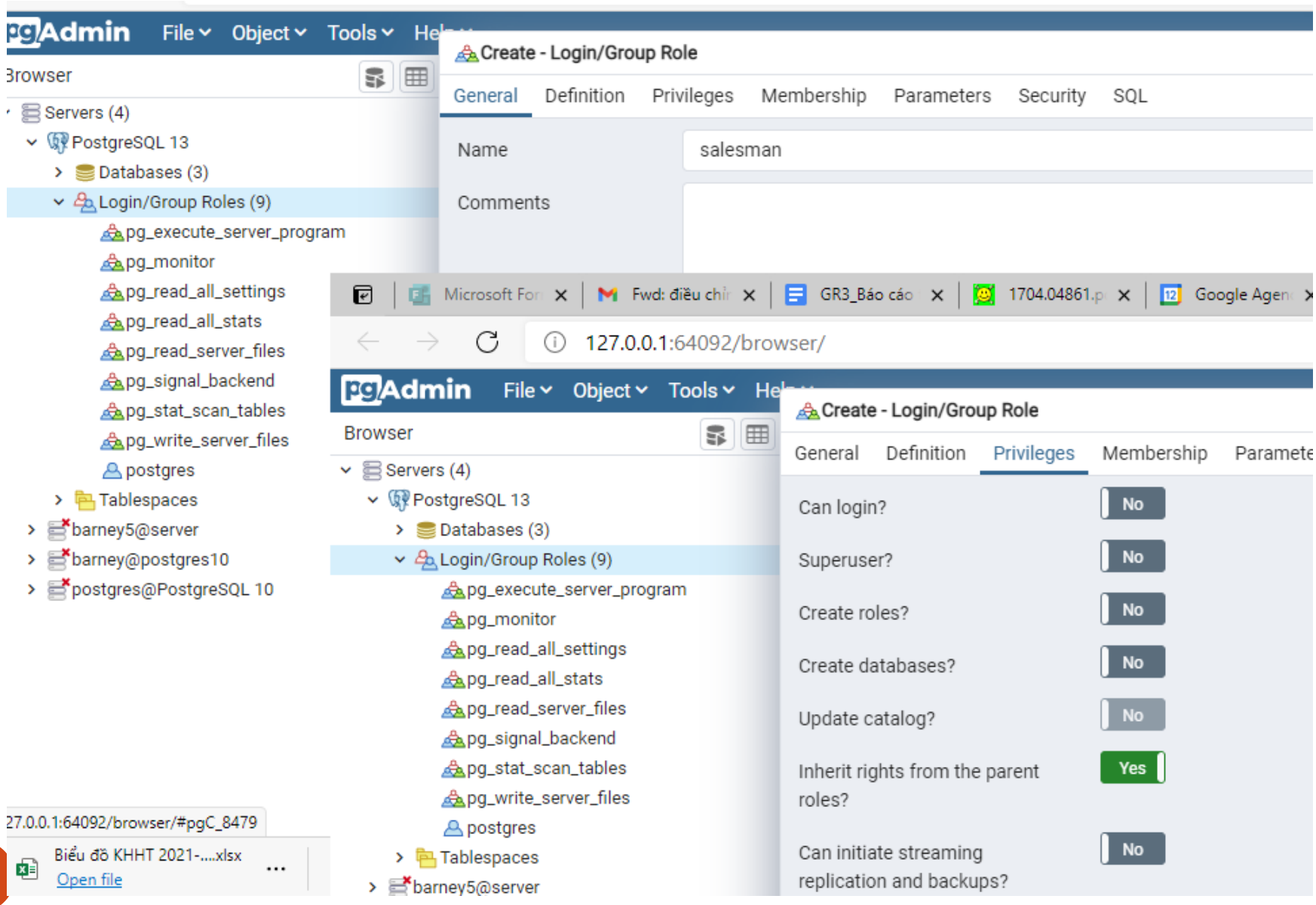
- That account is then **assigned as a member** of the appropriate **Group Roles that grant privileges** to the database objects required



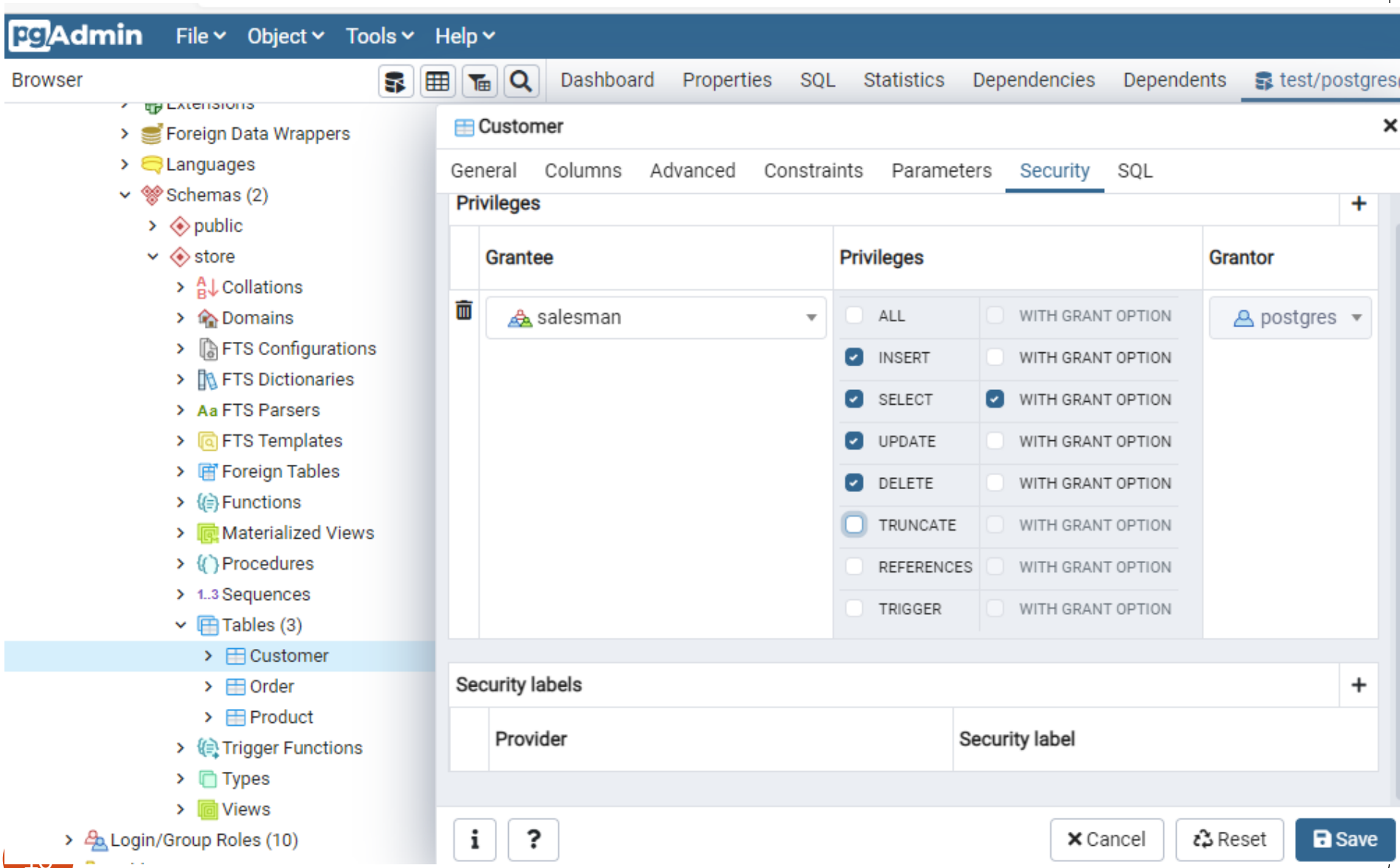
Practices – (continue…)

- Create a database ***test***
  - Customer
  - Product
  - Order
- Create two Group Roles
  - *Salesman* Group Role: read, write permission on the Customer and Order, only read permission on the Product
  - *Accountant* Group Role: read, write permission on the Product and Order, read permission on the Customer
- Create two Login Roles
  - salesman - Barney
  - accountant - Fred

Create a group role



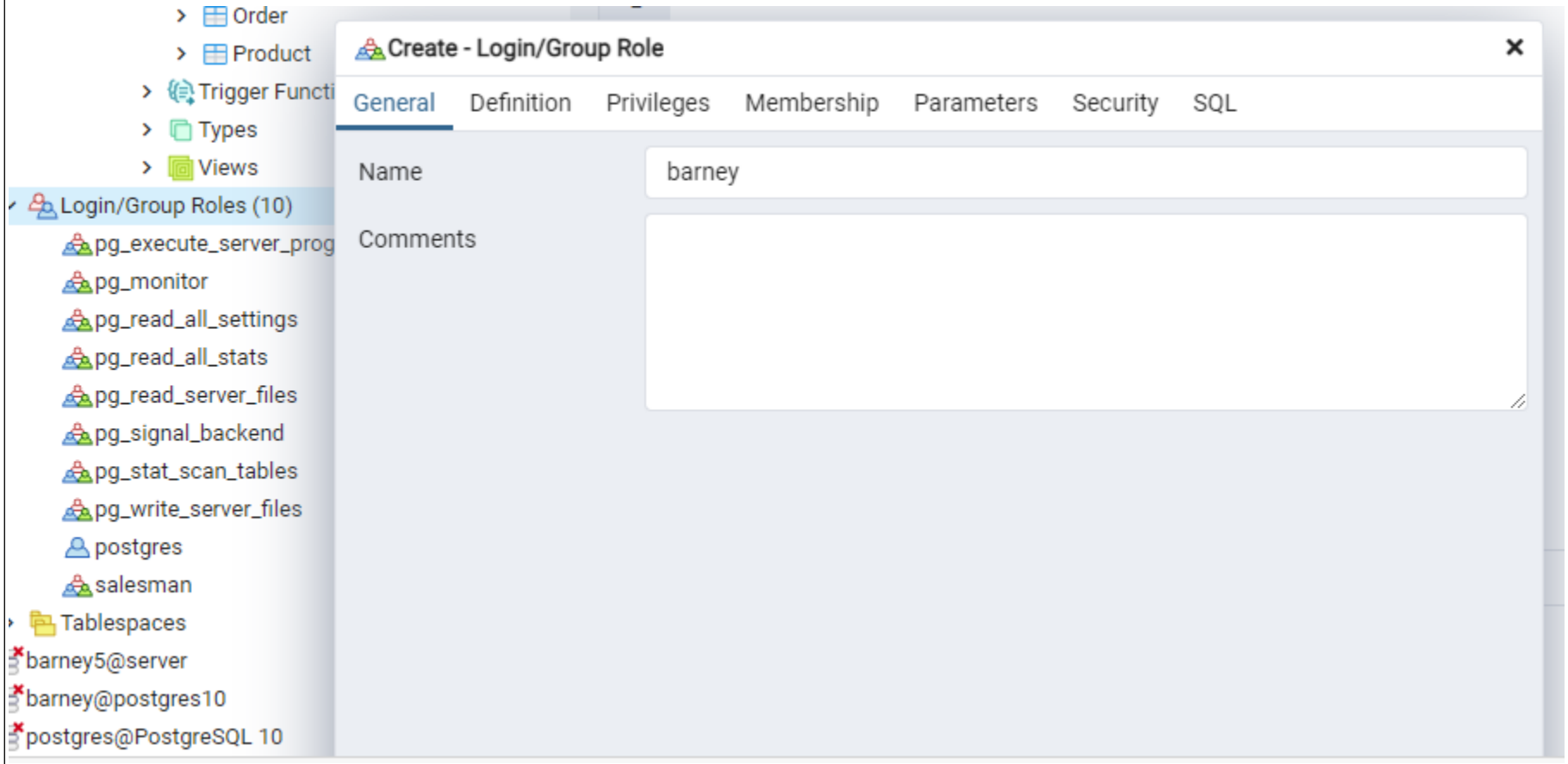
Grant privileges on the Customer table



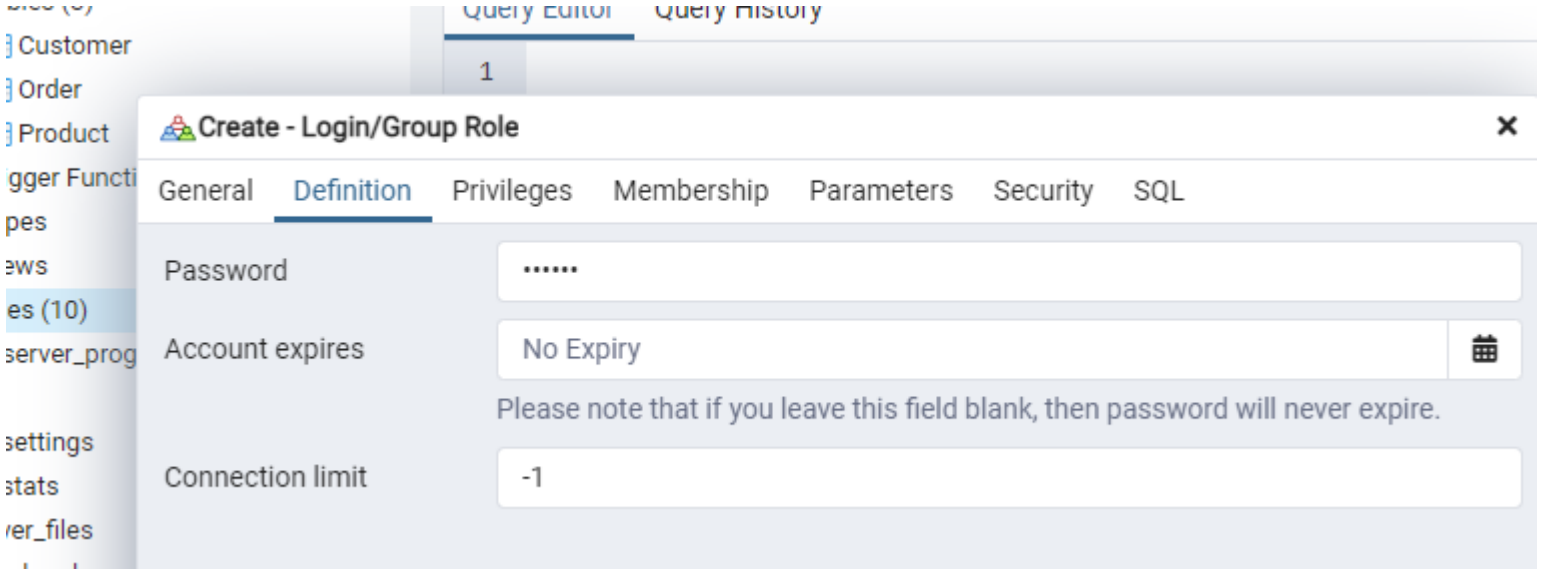
pgAdmin Object Privilege Codes

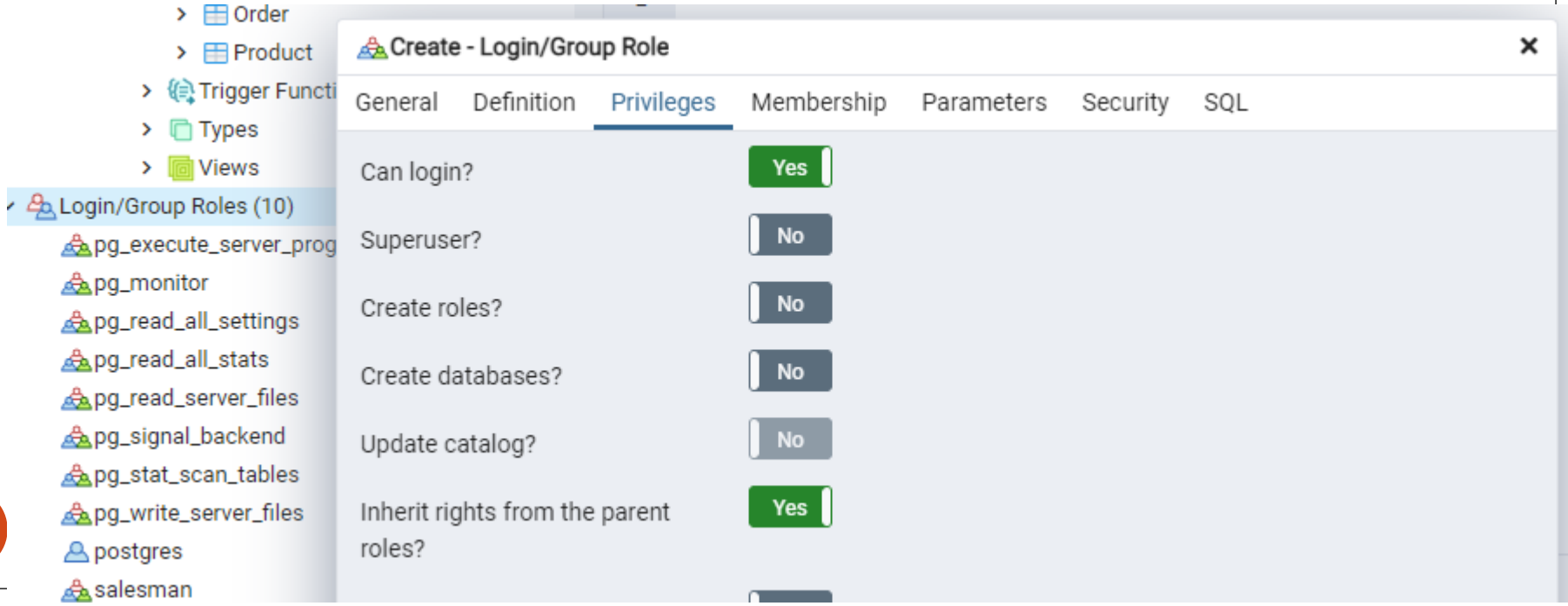
Code	Privilege
a	INSERT (append)
r	SELECT (read)
w	UPDATE (write)
d	DELETE
R	RULE
x	REFERENCES
t	TRIGGER
X	EXECUTE
U	USAGE
C	CREATE
T	TEMPORARY

Creating Login Roles

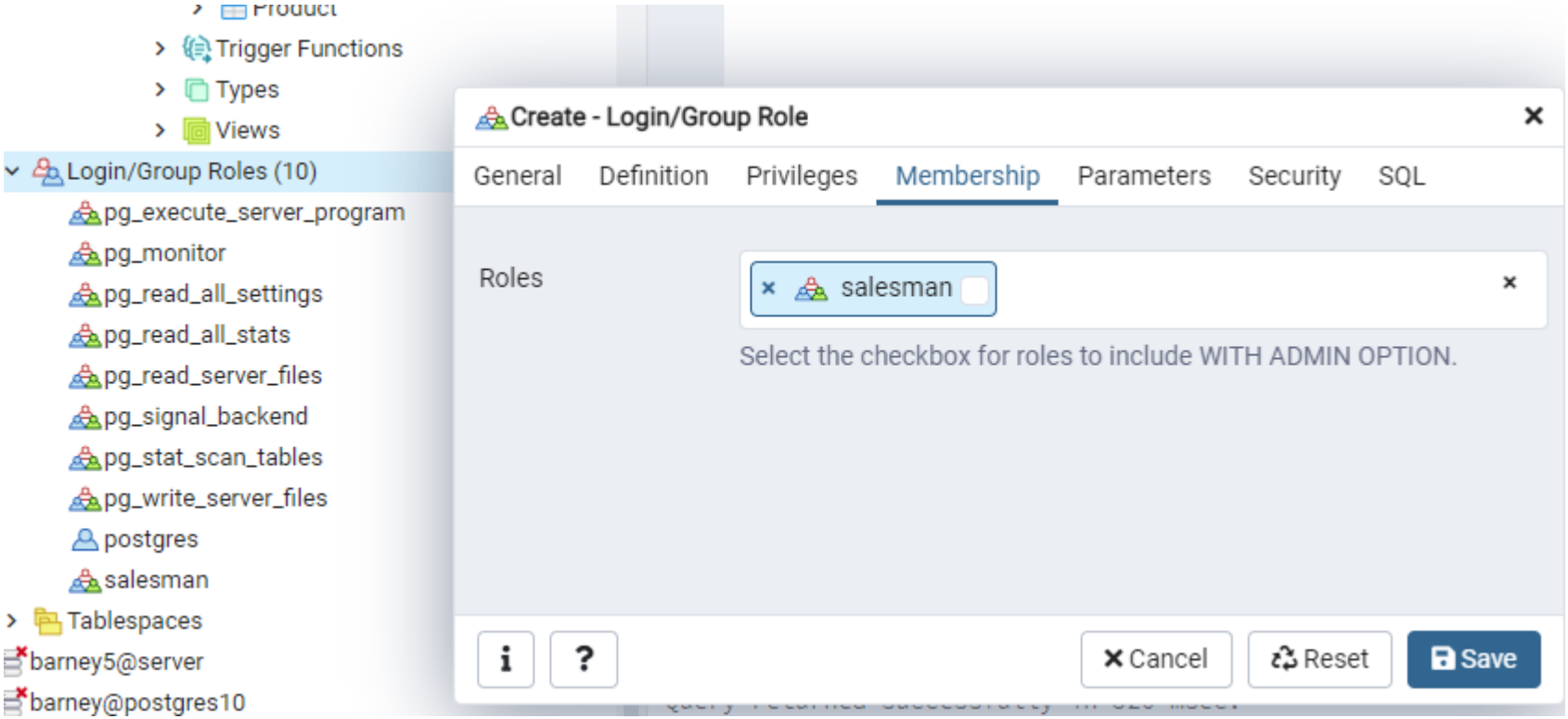


Enter a password and grant the "can login" privilege





Assign the login role/user (barney) to a group role (salesman)



Testing

- Create a new "Customer" in public schema
  - Same structure and same privileges as "Customer" in store schema

  (i.e: saleman can read, write on public."Customer")

Testing

- Login to the test database and the ***barney*** Login Role using pgAdmin 4

- Try to view and edit data on public."Customer"

➔ any problem ?

- Try to view and edit data on store."Customer"

➔problem? Why?

Testing

- Re-login with ***postgres*** account
  - Grant "usage" privileges on store schema

- Re-login with barney account
  - Try to view and edit data on store."Customer"
    ➔problem? Why?

  - Try to view and edit data on store."Product"
    ➔problem? Why?

Testing access permission using SQL statements

- Login to the test database and the ***fred*** Login Role with pgAdmin 4
- Open Query Tool, run following commands and see what will be happen

  - SELECT * from store."Product";
  - INSERT into store."Product" VALUES ('LAP001', 'Laptop', 'TakeAlong', 'Acme', '500.00', 100);
  - INSERT into store. "Customer"("CustomerID", "LastName", "FirstName") VALUES ('Cus001', 'Thi Oanh', 'Nguyen');

Practices – (continue…)

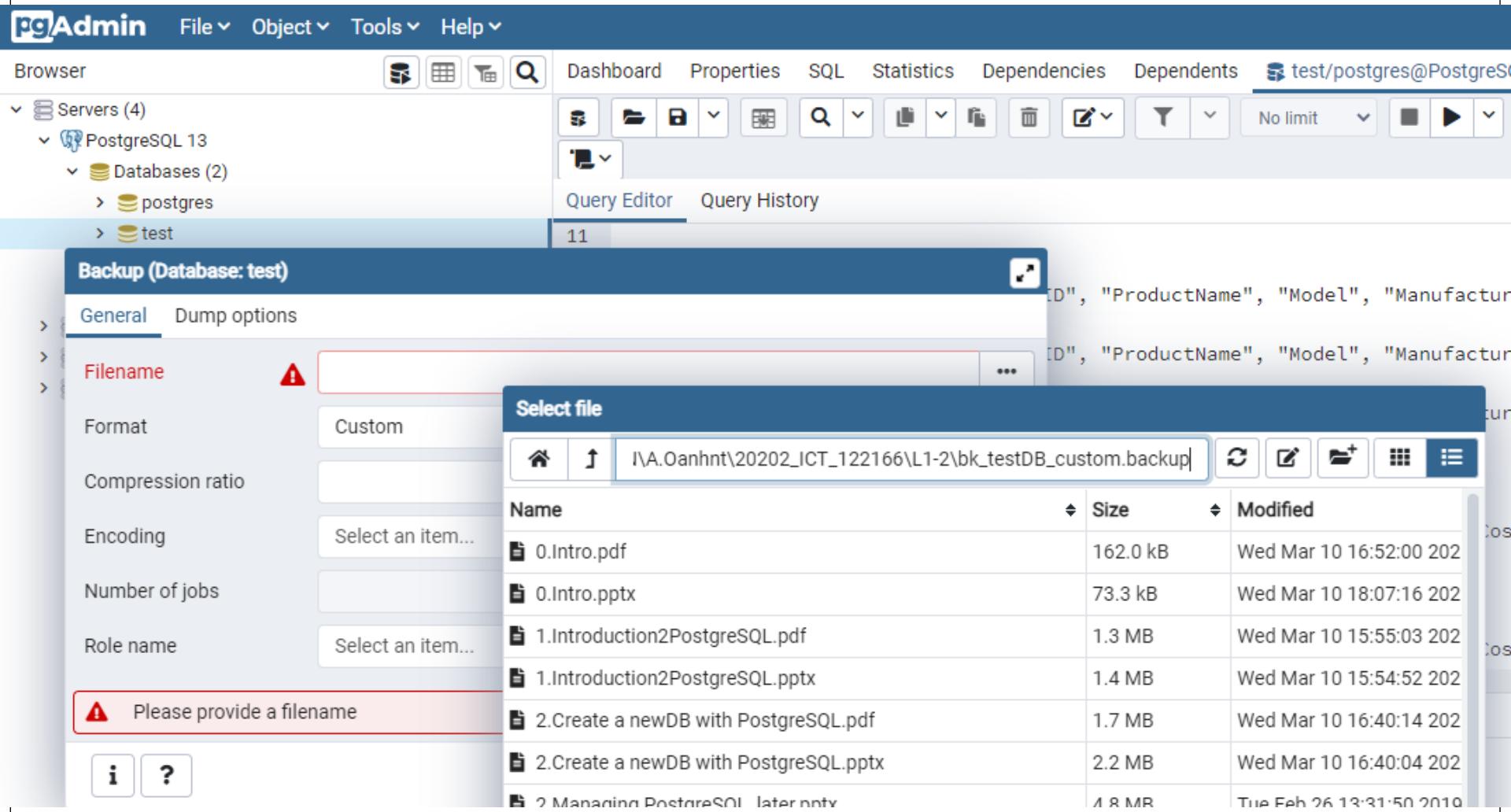
- Create two Group Roles
  - *Salesman* Group Role: read, write permission on the Customer and Order, only read permission on the Product
  - *Accountant* Group Role: read, write permission on the Product and Order, read permission on the Customer
- Create two Login Roles
  - salesman - Barney
  - accountant - Fred

And then verify whether the roles and their permissions are well defined.

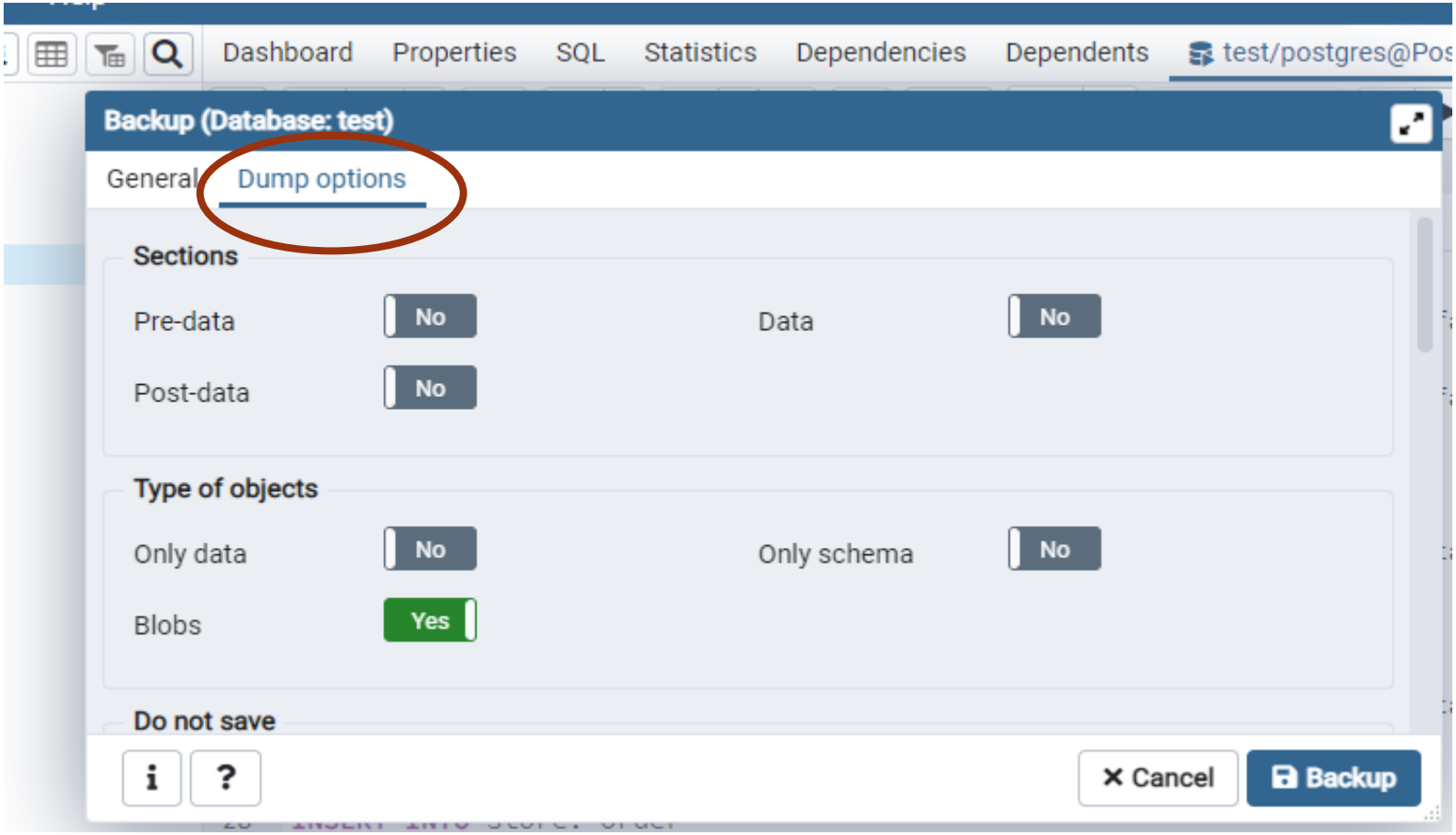
4. BACKUP and RESTORE

- Using pgAdmin 4

BACKUP

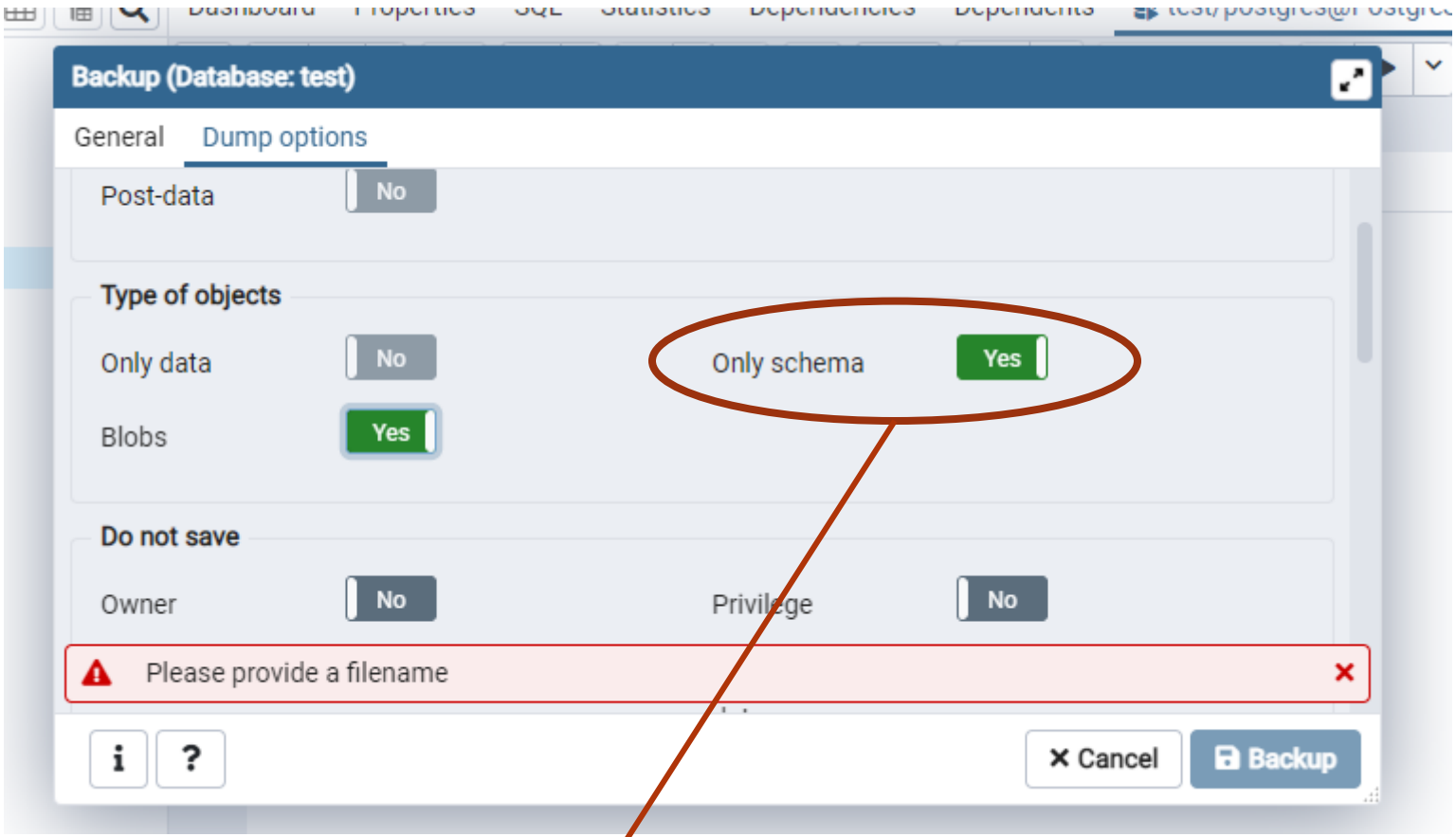

pgAdmin
File
Object
Tools
Help
Browser
Servers (4)
PostgreSQL 13
Databases (2)
postgres
test
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
test/postgres@PostgreS
No limit
Query Editor
Query History
11
Backup (Database: test)
General
Dump options
Filename
Format
Custom
Compression ratio
Encoding
Select an item...
Number of jobs
Role name
Select an item...
Please provide a filename
Select file
I\A.Oanhnt\20202_ICT_122166\L1-2\bk_testDB_custom.backup
Name
Size
Modified
0.Intro.pdf
162.0 kB
Wed Mar 10 16:52:00 202
0.Intro.pptx
73.3 kB
Wed Mar 10 18:07:16 202
1.Introduction2PostgreSQL.pdf
1.3 MB
Wed Mar 10 15:55:03 202
1.Introduction2PostgreSQL.pptx
1.4 MB
Wed Mar 10 15:54:52 202
2.Create a newDB with PostgreSQL.pdf
1.7 MB
Wed Mar 10 16:40:14 202
2.Create a newDB with PostgreSQL.pptx
2.2 MB
Wed Mar 10 16:40:04 202

BACKUP



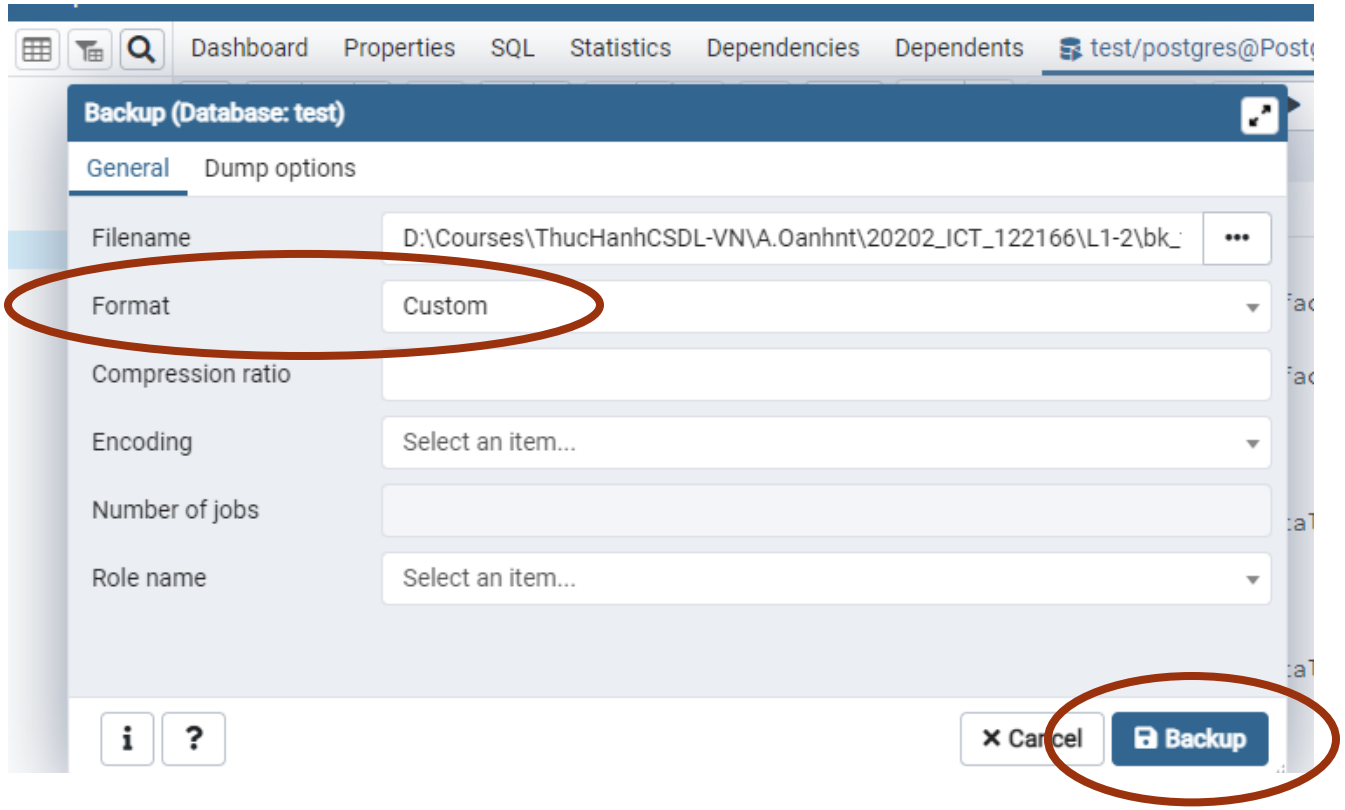
Use options by default, all schema, data and access permission will be saved

BACKUP



Backup only schema (database structure)

BACKUP


Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
test/postgres@Post
Backup (Database: test)
General
Dump options
Filename
D:\Courses\ThucHanhCSDL-VN\A.Oanhnt\20202_ICT_122166\L1-2\bk_
Format
Custom
Compression ratio
Encoding
Select an item...
Number of jobs
Role name
Select an item...
Cancel
Backup

BACKUP

Format field to select the format that is best suited for your application:

- Select ***Custom*** to create a custom archive file that you can use with pg_restore to create a copy of a database. Custom archive file formats must be restored with ***pg_restore***. This format offers the opportunity to select which database objects to restore from the backup file. Custom archive format is recommended for medium to large databases as it is compressed by default.

- Select **Tar** to generate a tar archive file that you can restore with pg_restore. The tar format does not support compression.

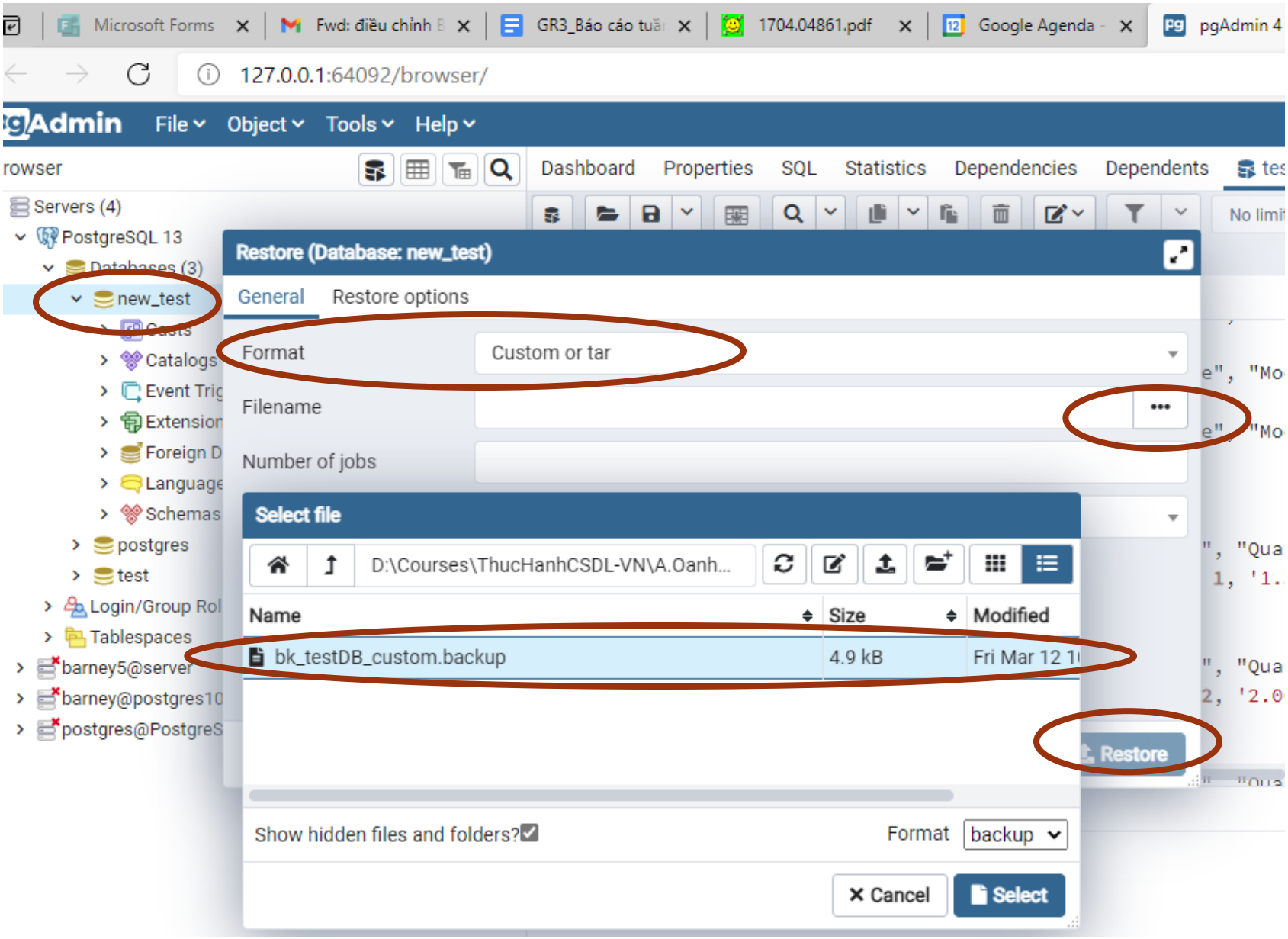
BACKUP

- Select **Plain** to create a plain-text script file.
  - It contains SQL statements and commands
  - you can be edited in a text editor, if desired, before using the **psql** program to restore database objects. Plain format is normally recommended for smaller databases;

- Select **Directory** to generate a directory-format archive suitable for use with ***pg_restore***
  - This file format creates a directory with one file for each table and blob being dumped, plus a Table of Contents file describing the dumped objects in a machine-readable format **that *pg_restore*** can read.
  - This format is compressed by default.

Restore

Create a new DB then restore a database from a saved file under custom, tar or directory format



4. BACKUP and RESTORE

- Using command-line tools

(PostgreSQL client applications) –> next file