



Prairie Postgres

Making Access Control Management Easy: Introducing pg_acm

Hettie Dombrovskaya

Prairie Postgres

PG Conf EU Valencia
Oct 2026

Who am I

Hettie Dombrovskaya

**Optimizing everything in Postgres, my life, and
the rest of the world!**

**Founder & President of Prairie Postgres NFP
LPI Board of Directors
ACM Chicago Chapter Communications Chair**

PG DATA 2027 Organizer



I am from Chicago!





2027.pg-data.org



PG DATA 2027

June 11-12 2027
UIC Student Center East

- Training Sessions
 - Labs
 - Talks
- Childcare
- DEI grants

CfP is open!

Registration is open!

Call for sponsors is open!

One more QR Code:
Please leave a feedback on my talk!



And now to the talk: What will be covered

- Identifying common security challenges
- What can we do about them?
- Security framework overview
- Ongoing issues and prospects

Why this talk?

- ... when securing data is a high priority
- ... and PostgreSQL has everything to provide it
- Still...

USER: POSTGRES

SCHEMA: public

WHY?!

#1 No need in roles and schemas

And all examples in documentation create objects in **PUBLIC** schema!

Table of Contents

Preface

1. What Is PostgreSQL?
2. A Brief History of PostgreSQL
3. Conventions
4. Further Information
5. Bug Reporting Guidelines

I. Tutorial

4. SQL Syntax

5. Data Definition

6. Data Manipulation

7. Queries

8. Data Types

9. Functions and Operators

10. Type Conversion

11. Indexes

12. Full Text Search

13. Concurrency Control

14. Performance Tips

15. Parallel Query

III. Server Administration

16. Installation from Binaries

17. Installation from Source Code

18. Installation from Source Code on Windows

19. Server Setup and Operation

20. Server Configuration

5.4. System Catalogs and Constraints

5.5. System Columns

5.6. Modifying Tables

5.6.1. Adding a Column

5.6.2. Removing a Column

5.6.3. Adding a Constraint

5.6.4. Removing a Constraint

5.6.5. Changing a Column's Default Value

5.6.6. Changing a Column's Data Type

5.6.7. Renaming a Column

5.6.8. Renaming a Table

5.7. Privileges

5.8. Row Security Policies

5.9. Schemas

5.9.1. Creating a Schema

5.9.2. The Public Schema

5.9.3. The Schema Search Path

5.9.4. Schemas and Privileges

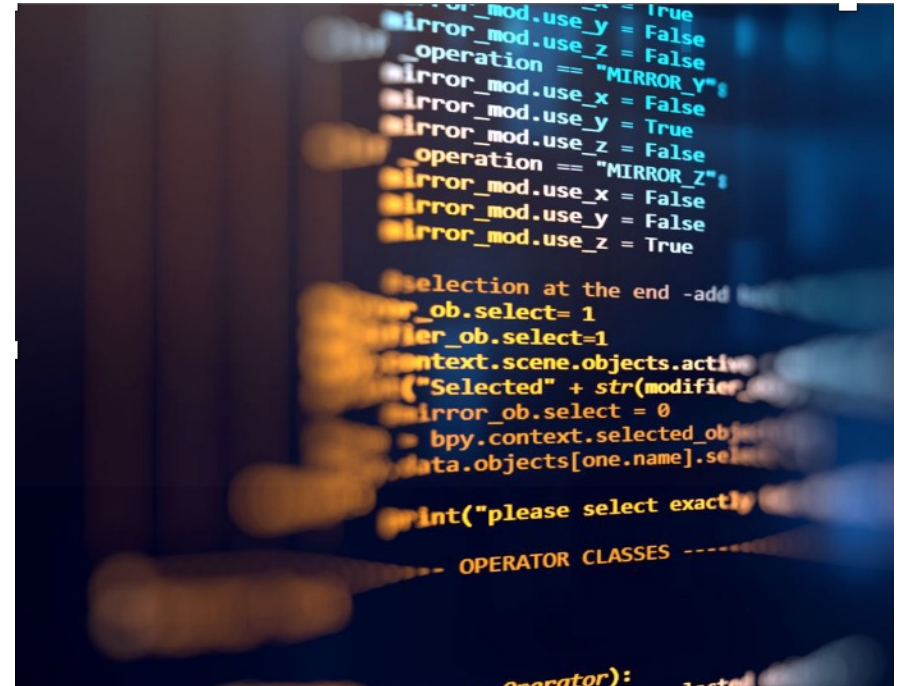
You do not need it – why bother?!!

Applications are developed using
postgres user

Move to production (ship to customer)
still using superuser

*(because we do not know what will break
otherwise)*

And since we use connections pools,
everyone is a superuser



#2: The wonders of inheritance: grant anything to anyone!

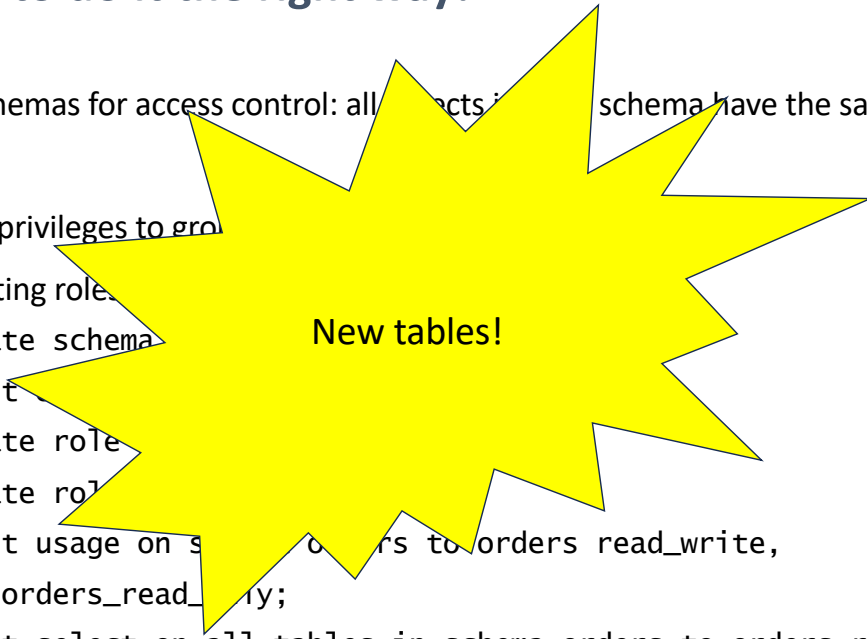
```
CREATE ROLE sales_reporter;  
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO sales_reporter;  
GRANT sales_reporter TO megan;  
GRANT INSERT, DELETE ON ALL TABLES IN SCHEMA sales TO anna;  
GRANT anna TO megan;  
CREATE ROLE payroll_manager;  
GRANT ALL ON ALL TABLES IN SCHEMA payroll TO payroll_manager;  
GRANT payroll_manager TO anna;
```

What about
megan???

What about other
databases?!

Let's try to do it the right way!

- Using schemas for access control: all objects in a schema have the same set of privileges
- Granting privileges to groups
- And granting roles



`create schema orders;
grant usage on schema orders to app_user;
create role orders_read_only;
create role orders_read_write;
grant usage on schema orders to orders_read_write,
orders_read_only;
grant select on all tables in schema orders to orders_read_only;
grant select, insert, update, delete on all tables in schema
orders to orders_read_write;
grant orders_read_write to app_user;`

What is not going to work?

#3: Default privileges

```
alter default privileges in schema orders
    grant select on tables to orders_read_only;
alter default privileges in schema orders
    grant select, insert, update, delete on tables to orders_read_write;
```

Now:

```
create table orders.customer (
    customer_id int primary key,
    customer_name text);
```

- Why were default permissions not applied?!

```
alter default privileges in schema orders for role orders_owner grant
select, insert, update, delete on tables to orders_read_write;
```

#4: The wonders of ownership!

```
create schema orders owner orders_owner;  
set role orders_owner;  
create table orders (order_id int);  
$  
orders_owner;  
orders;  
orders;
```

Permission denied!

What will happen?

Challenges #5, #6, #7... Lots of weird things!

```
grant select orders.sales_points to role_ro;  
grant insert, update, delete on orders.sales_points to role_app;  
grant role_ro to your_user;  
grant role_app to your_user;  
revoke delete on orders.sales_points from your_user;
```

Will this work?

- It won't, and moreover, errors won't be reported:
 - REVOKE of permissions which are not granted
 - GRANT permissions which are already granted except for roles
- You can't drop user that has any privileges
- You can't drop role cascade
- **And there is no easy way to see what permissions a given user has!**

I have a dream...

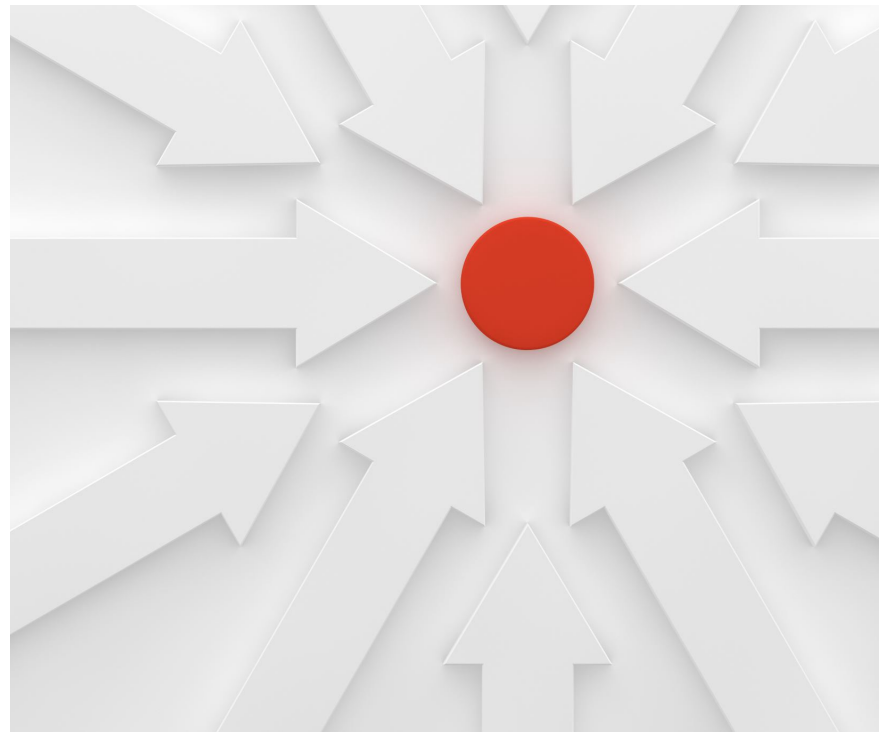
One schema

One owner

Ownership inheritance

Permission inheritance

Easy ownership change



Security Model Overview



**Enabling “strict
ownership”**

Basic principles



Principle of least privilege

- A user is given the minimum levels of access needed to perform their job functions.

Durability

- Non-superuser users do not have a way to bypass security settings

Encapsulation

- Object permissions are granted to group roles only



Key features

Permissions granularity

A schema is the least security unit
(all objects within a schema have
the same permissions)

Ownership inheritance

All objects in the schema are owned by
the schema owner, no matter who created
them

Owner-driven default privileges

Default schema privileges are always applied,
regardless of who created an object

Security-definer functions

Roles creation/deletion can be performed by non
superuser using security definer functions

Event triggers

The described behavior is event triggers-
driven

Enabling security model

- Deploy the package:

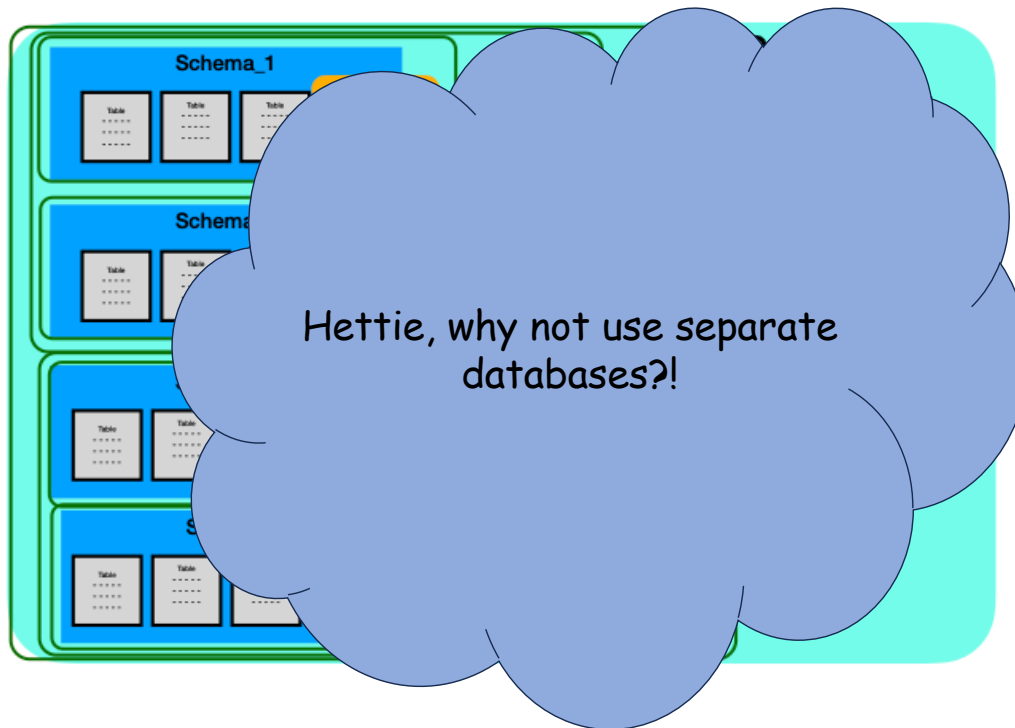
`https://github.com/hettie-d/pg\_acm`

- Enable security by running

```
select * from acm.enable_security();
```

- This will
 - enable event triggers
 - grant execute on all security-definer functions to the database owner role

Roles and ownership



Creating account

```
select * from  
    acm_tools.create_cust_account('acct_name');  
select * from  
    acm_tools.assign_account_user(  
        p_account_name => 'acct_name',  
        p_acct_user => 'user_name',  
        p_acct_user_password => 'password');
```

create schema my_schema

who can create schemas:

account owner

Actions:

- creates schema
- creates owner role
- creates read_write role
- creates read_only role
- sets up default schema privileges
- grants schema owner to account owner

Behavior

- all objects will be owned by schema owner
- permissions granted to read_only and read_write_role

drop schema my_schema

who can drop:

- Account user that created it

Actions:

- revokes read_only role from all users
- revokes read_write from all users
- revokes custom roles
- revokes owner role
- drops all associated roles
- drops schema

alter schema my_schema

who can alter:

- Account user that created it

what can be altered:

- only name

Actions:

- renames schema
- renames all associated roles

Security definer functions

Assign role:

- assign_schema_owner_role
- assign_schema_app_role
- assign_schema_ro_role

Revoke role:

- revoke_schema_owner_role
- revoke_schema_app_role
- revoke_schema_ro_role

assign_schema_owner_user

Input parameters:

- schema_name
- ddl_user_name
- ddl_user_password (opt)
- p_update_search_path boolean

Actions:

- creates user ddl_user_name if it does not exist
- changes password if user exists & password provided
- grants schema owner role to ddl_user_name
- changes seach_path if the flag is true

assign_schema_app_user

Input parameters:

- schema_name
- app_user_name
- app_user_password (opt)
- p_update_search_path boolean

Actions:

- creates user app_user_name if it does not exist
- changes password if user exists & password provided
- grants schema read_write role to app_user_name
- changes search_path if the flag is true

assign_schema_ro_user

Input parameters:

- schema_name
- ro_user_name
- ro_user_password (opt)
- p_update_search_path boolean

Actions:

- creates user ro_user_name if it does not exist
- changes password if user exists & password provided
- grants schema read_only role to ro_user_name
- changes seach_path if the flag true

Can you drop an account?

Yes!

```
select * from  
  acm_tools.drop_cust_account(  
    p_acct_name=>'acct_name',  
    p_force =>true);
```

What if you need to create an extension?

Event triggers on create/drop extension disable/enable
pg_acm triggers on CREATE SCHEMA and CREATE any object
within schema

What if you need more roles?

```
select * from  
acm_tools.create_role_for_schema(  
  p_schema_name=>'my_schema',  
  p_select =>true,  
  p_insert boolean default false,  
  p_update boolean =>true,  
  p_delete boolean default false,  
  p_truncate boolean default false)
```

```
select * from  
acm_tools.assign_role(  
  p_role_name =>'my_schema_su',  
  p_user =>'my_schema_update_user',  
  p_password =>'xxx')
```

How can I see all the roles and users?

```
select * from  
  acm_tools.list_acct_schemas_roles_flat()
```

account text	account_owner text	account_users acm_tools.users[]	schema_name text	schema_owner text	owner_users acm_tools.users[]	read_only_role text	read_users acm_tools.users[]	read_write_role text	app_users acm_tools.users[]
acct2	acct2_owner	{"acct_user2,-1"}	schema21	schema21_own...	{"acct_user2,-1"}	schema21_read_o...	{"ro_user21,50"}	schema21_read_wr...	[null]
acct2	acct2_owner	{"acct_user2,-1"}	schema22	schema22_own...	{"acct_user2,-1"}	schema22_read_o...	[null]	schema22_read_wr...	{"app_user22,50"}

Superpower!

- Public to private
- Reset schema owner

What is there for the users?

No need for a new
database when
you start a new
project

Creating users
without superuser
privileges

Assign and revoke
users' privileges

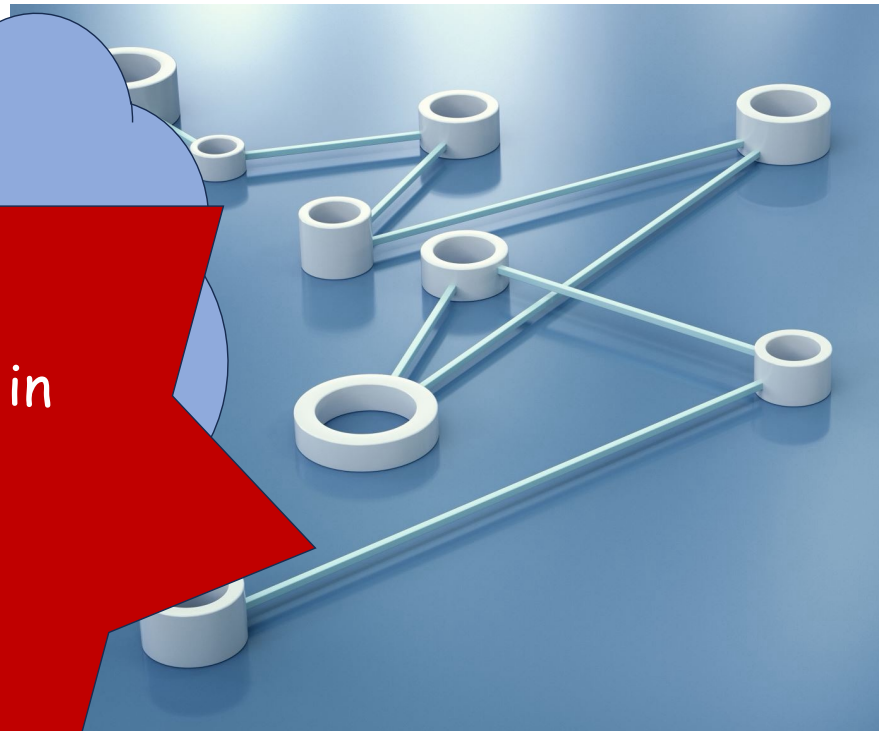
Change users'
passwords

You are in control!

Future work

- rework rese
- generic dra
- converti

But I want it in
Postgres!



Thank you!

Josh Davis

Greg Clough

Andrew Jackson

Boris Novikov

Where to find me

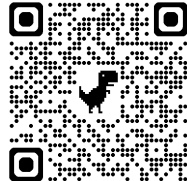
LinkedIn:

<https://www.linkedin.com/in/henrietta-dombrovskaya-367b26/>



GitHub:

<https://github.com/hettie-d>



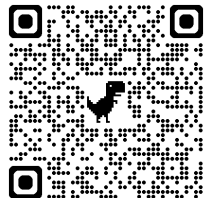
Prairie Postgres

prairiepostgres.org



Prairie PostgreSQL User Group

<https://www.meetup.com/illinois-prairie-postgresql-user-group>





2027.pg-data.org



And don't forget about PG DATA 2027!!!

<https://sessionize.com/pg-data-2027/>

<https://2027.pg-data.org/sponsors/>

Feedback!



Q&A Time!