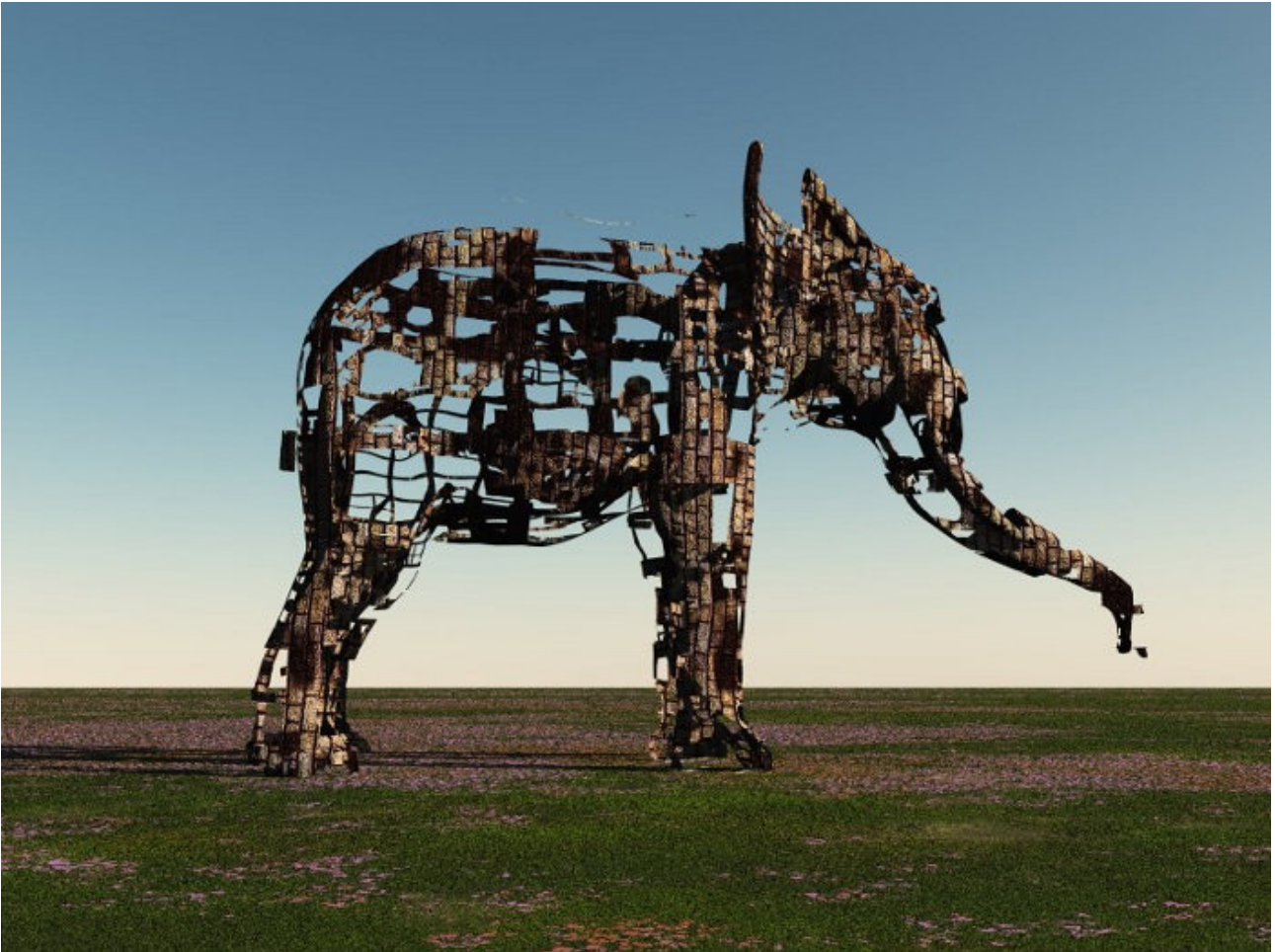


A large, light blue, stylized monogram of the letters 'FD' is centered on the page. The letters are thick and have a fluid, calligraphic feel. The 'F' has a long, sweeping tail that curves downwards and to the left. The 'D' is rounded and sits to the right of the 'F'.

# Managing rights in PostgreSQL

## Table des matières

Managing rights in PostgreSQL.....	3
1 The author.....	3
2 Introduction.....	4
3 Users, groups and roles.....	4
3.1 Users and groups.....	5
3.2 Modifying a role.....	5
4 Special roles and role attributes.....	5
4.1 Superusers.....	6
4.2 The PUBLIC role.....	6
4.3 Attributes.....	6
4.4 Inheritance.....	6
4.5 Inheritance example.....	7
5 Default rights.....	7
6 How access is granted or denied.....	8
6.1 Host Based Access.....	8
6.2 Database connection attribute.....	8
6.3 The object hierarchy.....	9
6.4 Going through to a relation.....	9
6.5 Ownership.....	9
6.6 Special cases.....	10
6.7 Viewing rights.....	10
6.8 Granting and Revoking rights.....	10
6.9 Securing the default installation.....	11
7 Default privileges.....	11
7.1 How default privileges work.....	11
7.2 The read only user.....	12
7.3 Other use cases.....	12
8 SE-PostgreSQL?.....	13
8.1 Prerequisites.....	13
8.2 Installation.....	13
8.3 Creating your policy.....	14
8.4 Current limitations.....	14
9 Conclusion.....	14



# Managing rights in PostgreSQL

## 1 The author



- Auteur : Nicolas Thuvin
- Company : Dalibo
- Date : December 2011
- URL : <https://support.dalibo.com/kb/conferences/bla>

## 2 Introduction

In this talk :



- How rights works in PostgreSQL from connection to SQL statement execution
- How to manage roles and rights
- Defaults privileges
- SE-PostgreSQL?

I will try to show real world example whenever possible.

## 3 Users, groups and roles



- Users are used to identify people accessing the db
- Groups allow to share rights between users
- Since 8.1, users and groups are roles
- A user is a role that can log in
- A group is a role that cannot log in

## 3.1 Users and groups

- To create a user:

```
CREATE ROLE user_name LOGIN <ATTRIBUTES>;
```



- To create a group:

```
CREATE ROLE group_name NOLOGIN <ATTRIBUTES>;
```

- To add a rôle to another:

```
GRANT ROLE group_name TO user_name;
```

## 3.2 Modifying a role

- ALTER ROLE



- For example, to set a password:

```
ALTER ROLE postgres WITH PASSWORD 'new_password';
```

## 4 Special roles and role attributes

- Superusers
- The PUBLIC role
- Global modification attributes
- Inheritance

## 4.1 Superusers

- By default postgres, without a password (!)
- Can be given to any role using the SUPERUSER attribute:



```
ALTER ROLE ROLE role_name SUPERUSER;  
ALTER ROLE role_name NOSUPERUSER;
```

- Superusers are god on the cluster, but:
  - They must pass through Host Based Access (pg\_hba.conf)
  - They cannot connect to a database with data\_lowconn set to false

## 4.2 The PUBLIC role



- An implicit group everybody belongs to
- Has some default rights granted

## 4.3 Attributes



- A set of global rights
- Superuser
- Inheritance
- Login, connection limit and validity
- Database, Role creation
- ⇒ Columns of pg\_roles

## 4.4 Inheritance



- Allow a role to get the rights of other roles granted to it directly or not
- Use of SET ROLE to obtain rights from other roles
- Protect the role from having too many rights all the time

## 4.5 Inheritance example

How to delegate superuser privileges without giving the password of postgres to others:

- Create a admins group with inheritance :

```
CREATE ROLE admins NOLOGIN NOINHERIT;
```

- Create a admin account with no superuser rights:



```
CREATE ROLE one_admin LOGIN PASSWORD 'foobar';
```

- Put one\_admin into admins group:

```
GRANT admins TO one_admin;
```

- Put admins into postgres:

```
GRANT postgres TO admins;
```

## 5 Default rights

After initdb:

- Local access only (listen\_addresses, pg\_hba.conf)
  - Right to connect to any database but template0
    - CONNECT : connect to the database
    - TEMP : create temporary tables
  - Rights on the public schema
    - USAGE : access the objects
    - CREATE : create new objects
- ⇒ Those default rights are granted to PUBLIC

## 6 How access is granted or denied



- Host Based Access
- The object hierarchy
- Going through to a relation
- Ownership

### 6.1 Host Based Access



- Configuration done in `pg_hba.conf`
- Define what authentication method will be asked for :
  - A user (role with the LOGIN attribute)
  - Who wants to connect to a database
  - From a particular host (or the local Unix Domain socket)
- Access is granted when :
  - A line matches
  - AND the method is NOT reject
  - AND the client correctly answer to authentication method
- Superusers cannot bypass this check
- The `pg_hba.conf` file is walked from top to bottom, the server stops when a line matches or at the bottom

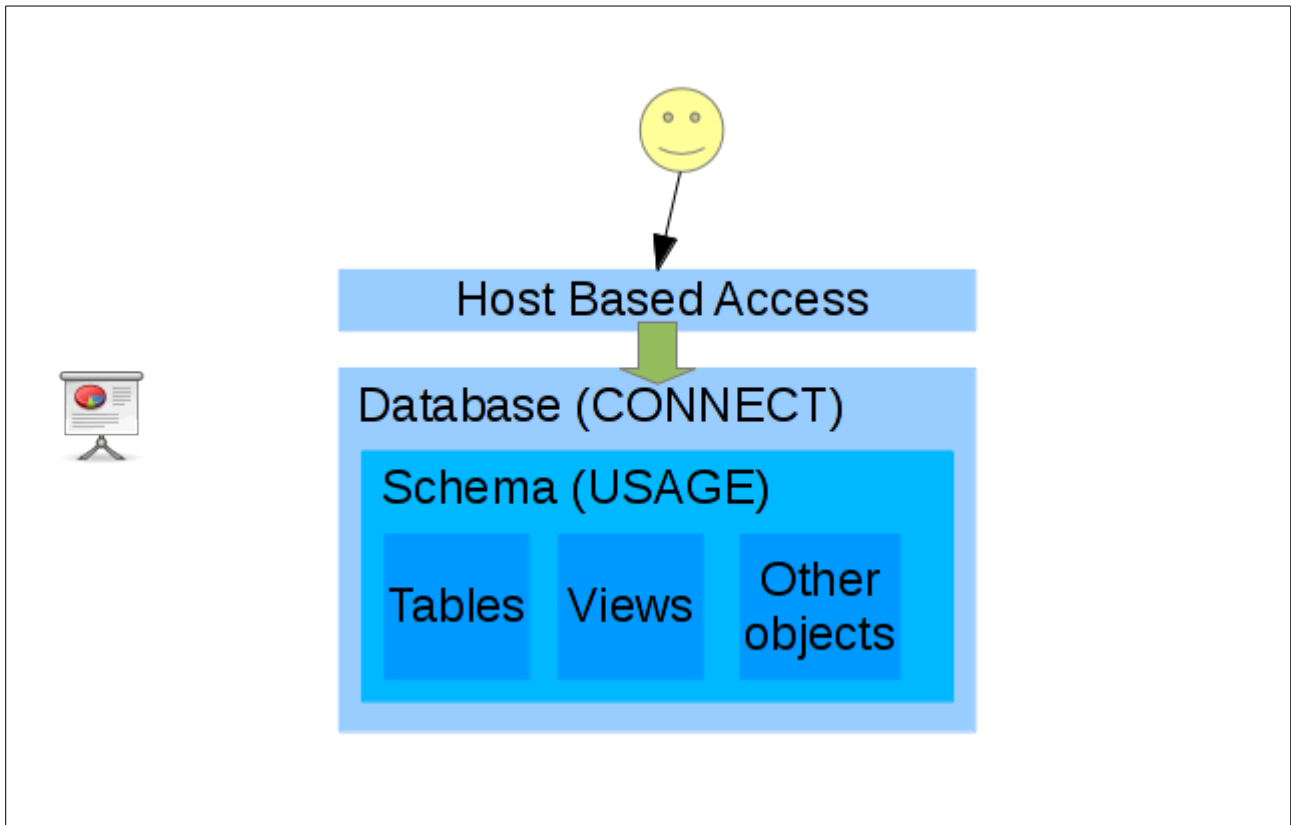
### 6.2 Database connection attribute




- The database must allow connections
- `dataallowconn` set to true in `pg_database`
- Superusers cannot bypass this
- Exemple `template0`



## 6.3 The object hierarchy



## 6.4 Going through to a relation



Provides HBA says ok and the database allows connections, the role:

1. Must have the CONNECT right to the database
2. Must have the USAGE on the schema containing the object
3. Must have the ownership or right to access or modify the contents of the relation

## 6.5 Ownership



- The owner of an object can:
  - Access and modify its contents
  - Modify its structure
  - Drop it provided it has the right to modify the parent object
- Someone who does not own an object:
  - Cannot access/modify the contents unless a right is granted
  - Cannot modify its definition (no rights exist for that)
  - Can drop it if he/she owns the schema /!\

## 6.6 Special cases



- Views:
  - Rights needed to access them like any other relation
  - The underlying query is executed with the rights of their owner
- Functions:
  - Rights needed to execute them
  - Can be executed with the privileges of their owner (SECURITY DEFINER)

## 6.7 Viewing rights



- \*acl columns in tables of the system catalog, mainly:
  - pg\_database: dataacl ⇒ Database rights (\l)
  - pg\_namespace: nspacl ⇒ Schema rights (\dn+)
  - pg\_class: relacl ⇒ Tables, Views and Sequences (\dp)
  - pg\_proc: proacl ⇒ Functions
- If empty, then default rights
- Format documented on the documentation of GRANT

## 6.8 Granting and Revoking rights



- Use GRANT to give a right
- Use REVOKE to remove it
- The name of privileges depends on the target
- WITH GRANT OPTION allows the target role to give the right
- ALL keyword to give all rights ( $\geq 9.0$ )
- Give a role to another (manage membership)
- « \h GRANT » in psql remembers the details for you

## 6.9 Securing the default installation



1. Set a password for postgres:
2. Configure pg\_hba.conf to use the md5 method and reload
3. Give ownership of databases to a non applicative role
4. Revoke rights from the PUBLIC role:

```
ALTER ROLE postgres WITH PASSWORD 'new_password';
```

```
REVOKE ALL ON DATABASE db_name FROM PUBLIC;  
REVOKE ALL ON SCHEMA public FROM PUBLIC;
```

Then one can:

- Grant rights to applicative roles
- Setup default privileges to ease the management of rights

## 7 Default privileges



- A way to automatically give rights at object creation
- Best used when included in the design
- Very powerful and can be life saving

## 7.1 How default privileges work



- ALTER DEFAULT PRIVILEGES FOR role IN SCHEMA nsp GRANT right ON objects TO other\_role
- When role:
  - creates an object of the “objects” (table, sequence...)
  - inside the nsp schema
- then right is automatically granted to other\_role on the new object
- use \ddp in psql to view default privileges

## 7.2 The read only user



- There is always a boss who wants a read-only access
- How to solve that:

```
CREATE ROLE readonly LOGIN PASSWORD 'some_pass';
-- Existing objects
GRANT CONNECT ON DATABASE the_db TO readonly;
GRANT USAGE ON SCHEMA public TO readonly;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonly;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO readonly;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO readonly;
-- New objects
ALTER DEFAULT PRIVILEGES FOR ddl_user IN SCHEMA public GRANT SELECT ON TABLES TO
readonly;
ALTER DEFAULT PRIVILEGES FOR ddl_user IN SCHEMA public GRANT SELECT ON SEQUENCES
TO readonly;
ALTER DEFAULT PRIVILEGES FOR ddl_user IN SCHEMA public GRANT EXECUTE ON FUNCTIONS
TO readonly;
```

## 7.3 Other use cases



- The best is to use different roles for managing the structure and the content:
  - The owner takes care of the structure
  - The owner has default privileges to let an application role modify the data
- Default privileges can be used to clean rights before going to production:
  - Setup the default privileges
  - Restore dumps with `pg_restore -U ddl_user -X -o`

## 8 SE-PostgreSQL?



- Allow to enhance security by asking SELinux if access can be granted to an object
- SELinux context is checked after regular privileges (like on the system)
- Can enforce the external policy up to the column (like regular privileges)

### 8.1 Prerequisites



- A SELinux enabled system, e.g. Linux only
- PostgreSQL  $\geq$  9.1
- The `sepgsql` module (`--with-selinux`)
- The Reference Policy module for PostgreSQL loaded
- IPsec or some way to label what comes from the network
- Knowledge on SELinux policy development

## 8.2 Installation

- Confine the PostgreSQL server on the Linux side:
  - load the `postgresql.pp` SELinux Policy module
  - (re)label the files of the PostgreSQL installation
- Load `sepgsql` at the cluster startup:



```
shared_preload_libraries = 'sepgsql'
```

- Create the SE-PostgreSQL functions inside the database:

```
\i /path/to/contrib/sepgsql.sql;  
SELECT sepgsql-restorecon(NULL);
```

## 8.3 Creating your policy



- The reference policy gives some interfaces for SELinux roles (see `postgresql.if`)
- The reference policy gives examples on possible rights
- Use `SECURITY LABEL` statements to label the objects

## 8.4 Current limitations



With SE-PostgreSQL in 9.1:

- No labels for database
- No row level labels
- No Data Definition Language rights
- Unable to hide object existence, only the contents

## 9 Conclusion



- PostgreSQL features on privileges are rich
- The default installation is not so bad on the security side, and easily hardened
- Default privileges ease the management of rights, when properly used
- And SE-PostgreSQL adds promising security features to PostgreSQL