# SQL DCL : Data Control Language

Fabien Coelho
MINES ParisTech

Composé avec LaTeX, révision 3581





## Droits système vs base de données

**système** autorisation de connexion
- utilisateurs réunis en groupes
- authentification par mot de passe
- machines autorisées (numéro ip)

**base** de donnée
- accès aux objets, aux données (tables, colonnes… lignes)
- consultation vs modification
- délégation des droits à d'autres utilisateurs





## Concept de rôle

**titulaire de droits** dans la base de données
- identifié par nom (et numéro) pour le *cluster*
  approximation par catalogue : `calvin@dbname`
- propriétaire d'objets (base, schéma, relation)
- super-user, création de bases, d'utilisateurs…

**utilisateur** rôle spécial qui peut se **connecter**
- mot de passe, nombre de connexion max, expiration…

**groupe** rôles donnés à des utilisateurs, pas de connection
- organisation, regroupement des droits
- restrictif (acquisition explicite) ou hérité (automatique)





## Création d'un rôle utilisateur : `LOGIN`

```
CREATE ROLE "calvin" WITH
  LOGIN
  NOINHERIT
  ENCRYPTED PASSWORD 'Hobbes!'
  IN ROLE "etudiant";
```

## Création d'un rôle de regroupement : `NOLOGIN`

```
CREATE ROLE "maitre"
  NOLOGIN -- par défaut
  ADMIN "calvin"
  ROLE "hobbes", "suzy";
```





## Création d'un utilisateur

```
CREATE USER hobbes
  WITH ENCRYPTED PASSWORD 'grr'
  IN GROUP family;

CREATE USER calvin
  WITH ENCRYPTED PASSWORD 'h02bes'
  CREATEDB CREATEUSER
  IN GROUP family;
```

## Création d'un groupe

```
CREATE GROUP family;

CREATE GROUP herge WITH tintin, milou, haddock;
```





## Acquisition explicite d'un rôle

**SET ROLE …** droits effectifs de ce rôle
- doit être membre du rôle (groupe)
- `CURRENT_USER` vs `SESSION_USER`
- les droits du rôle s'appliquent strictement

**RESET ROLE** repasse au rôle (user) initialement connecté (session).

**INHERIT vs NOINHERIT** contrôle des droits hérités automatiquement





## ROLE INHERIT vs NOINHERIT

**NOINHERIT** droits des groupes, ignorés par défaut

`SET ROLE` explicite pour y accéder

```
CREATE TABLE t1();
-- permission denied
SET ROLE plusfort;
CREATE TABLE t2();
-- ok
```

**INHERIT** droits des groupes cumulés par défaut

c'est valeur par défaut (hélas)

```
CREATE TABLE t1();
-- ok
```

**logiquement** user `NOINHERIT`, groupe `INHERIT`





## Contrôles des accès

- sujet important pour une base de donnée partagée
- système de droit très complet…

  **sujets** administrateur, propriétaire, utilisateurs, groupes ou `PUBLIC`

  **objets** `DATABASE SCHEMA TABLE LANGUAGE FUNCTION`
  et connexion à une base particulière

  **droits** selon les objets, plus **droits d'attribution** des droits

- par défaut permissif (consultation/utilisation pour `PUBLIC`)

## Droits de connexion et méthode d'authentification

**pg_hba.conf** fichier de configuration dans la base

modifiable par l'administrateur du système

**syntaxe** `method bases users hosts auth...`

**method** `local` ou réseau `host hostssl hostnossl`

**bases** `all sameuser samegroup...`

**users** `all calvin +eleves` (membres du groupe)

**hosts** numéro IP et masque de sous-réseau

**auth** `trust password md5 ident...`

```
local      all    calvin              ident
hostnossl  all    hobbes  10.2.14.0/24 md5
hostssl    all    all     0.0.0.0/0    md5
host       comics +eleves 127.0.0.1/32 md5
```





## Modes d'authentifications des connexions clients

**trust** pas d'authentification ! *déblocage...*

**password** mot de passe géré par pg, circule en clair

**md5** idem, mais pas de circulation en clair

**scram-sha-256** idem, mais mot de passe mieux protégé

**crypt** idem, ancienne version (pg avant 7.2)

**GSSAPI** RFC 2743 – single sign on

**SSPI** MS Windows – single sign on

**Kerberos** environnement chiffré...

**ident** RFC 1413 (tcp 113) – connexions réseau ou locales

**LDAP** RFC 4510 *Lightweight Directory Access Protocol*

**PAM** Pluggable Authentication Modules (Linux, Solaris)





## Allocation ou retrait des droits sur les objets

— syntaxe un peu verbeuse, mais compréhensible...

```
GRANT { ALL | { USAGE | ... }[, ...] } (col[, ...] )
ON { DATABASE | SCHEMA | TABLE | ... } objectname
TO { { user | GROUP group | PUBLIC}[, ...] }
[ WITH GRANT OPTIONS ] ;

REVOKE [ GRANT OPTIONS FOR ]
  { ALL | { USAGE | SELECT | ... }[, ...] }
ON { DATABASE | SCHEMA | TABLE | ... } objectname
FROM { { user | GROUP group | PUBLIC}[, ...] } ;
```





## Droits sur `DATABASE`

**`CONNECT`** connection à un catalogue

**`CREATE`** création de nouveaux `SCHEMA`

**`TEMPORARY`** création de tables temporaires

## Droits sur `SCHEMA`

**`CREATE`** créations de relations `TABLE SEQUENCE VIEW`

et objets attachés `CONSTRAINT INDEX`...

**`USAGE`** utilisation des relations





## Droits sur `TABLE`

**`SELECT INSERT UPDATE DELETE`** autorise (colonne)

**`TRUNCATE`** suppression de toutes les lignes d'une table

**`REFERENCES`** clefs étrangères (origine ou destination) (colonne)

**`RULE`** créations de règles de transformations...

**`TRIGGER`** actions automatiques avant ou après évènements

## Droit sur `FUNCTION`

**`EXECUTE`** exécution d'une fonctions

concerne indirectement `OPERATOR AGGREGATE`...





## Droit sur `LANGUAGE`

**`USAGE`** nouvelles fonctions dans ce langage

exemples : C, SQL, PL/pgSQL, PL/Tcl, PL/Perl, PL/Python...

versions *trusted* (limitée) ou *untrusted* (complète)





## Manipulation de tables ?

**système** droit connexion dans `pg_hba.conf`

**DATABASE** droit `CONNECT`

**SCHEMA** droit `USAGE`

**TABLE** droit `SELECT INSERT UPDATE DELETE TRUNCATE`...

aussi : `ON ALL TABLES IN SCHEMA ...`

**autres** éventuellement sur les séquences, vues...





## Droits d'administrations

**createrole** création utilisateurs et groupes

**createdb** création de databases

**replication** connection de réplication

**bypassrls** sauvegardes complètes

**superuser** tout !

### Droit de modification des droits ?

**administrateur** de la base, évidemment !

**propriétaire** de l'objet concerné !

**utilisateurs** à qui on l'a donné

```
GRANT ... WITH GRANT OPTIONS;
```





### Affichage des droits *(privileges)*

**pg_catalog** consultation des tables systèmes

type interne `aclitem` dans description des objets

**psql** commande interactive `\dp`

grantee=perms/grantor, `*` si attribuables

**r** SELECT *read*, **w** UPDATE *write*, **a** INSERT *append*, **d** DELETE,
**x** REFERENCES, **t** TRIGGER, **X** EXECUTE, **U** USAGE, **C** CREATE,
**c** CONNECT, **T** TEMPORARY

```
comics=# \dp auteur
            Access privileges for database "comics"
 Schema |  Name  | Type  |           Access privileges
--------+--------+-------+---------------------------------------
 public | auteur | table | {coelho=arwdxt/coelho,eleves=r/coelho}
```





### RLS : *Row-Level Security*

— contrôle des accès par *lignes* `CREATE POLICY`

— ajouté au niveau *table* et *colonne*

— comptes admin *calvin* et utilisateur *hobbes* :

```
CREATE USER calvin SUPERUSER;
CREATE USER hobbes;
```

— création d'une table par admin *calvin* :

```
CREATE TABLE AppPassword(
  id SERIAL PRIMARY KEY,
  login TEXT UNIQUE NOT NULL,
  pass TEXT);
INSERT INTO AppPassword(login, pass)
  VALUES ('calvin', NULL), ('hobbes', NULL);
```





— permissions par admin *calvin* :

```
GRANT SELECT (id, login) ON TABLE AppPassword TO PUBLIC;
GRANT UPDATE ON TABLE AppPassword TO PUBLIC;
CREATE POLICY PassAccess ON AppPassword
  USING (TRUE) -- read
  WITH CHECK (login = CURRENT_ROLE); -- write
ALTER TABLE AppPassword ENABLE ROW LEVEL SECURITY;
```

— accès restreints pour *hobbes* :

```
SELECT id, login FROM AppPassword;
-- OK
SELECT * FROM AppPassword;
-- ERROR: permission denied for relation apppassword
UPDATE AppPassword SET pass = 'yin' WHERE login = 'hobbes';
-- OK
UPDATE AppPassword SET pass = 'yang' WHERE login = 'calvin';
-- ERROR: new row violates row-level security policy ...
```





### Conclusion

— modèle très complet et quasi standard

— extension RLS très puissante, mais complexe

— droits souvent gérés directement au niveau **application**





### Compléments à SQL, pas tous standards

**FUNCTION** ajout de nouvelles fonctions

**TRIGGER** actions sur évènements (appel une fonction)

**RULE** réécriture des requêtes

traduction de `INSERT UPDATE DELETE` sur `VIEW`

**INDEX** indexation des tables pour optimisation





### Extensions des capacités

**TYPE** ajout de nouveaux types (base, composites)

**DOMAIN** type + contraintes

**LANGUAGE** nouveau langage côté serveur

**OPERATOR CLASS** classe d'opérateurs (pour indexation)

**OPERATOR** un opérateur (appel une fonction)

**AGGREGATE** un aggrégat (appel une fonction)

**CAST** conversions entre types

**CONVERSION** encodages des chaînes de caractères

**EXTENSION** groupe d'objets (fonctions, opérateurs...)



## List of Slides