SQL DCL : Data Control Language

Fabien Coelho, Claire Medrala

Mines Paris – PSL

Décembre 2023

Authentification et Autorisations

Authentification lors de la connexion — *Système*

- pour un utilisateur (un groupe)
- authentification souvent par mot de passe...
- selon la machine d'origine (numéro ip)
- selon la base de données

Autorisations des manipulations — *BD*

- accès aux objets, aux données (tables, colonnes... lignes)
- consultation vs modification
- délégation des droits à d'autres utilisateurs

Authentification — *système*

`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 scram-sha-256
hostssl    all    all     0.0.0.0/0    scram-sha-256
host       comics +eleves 127.0.0.1/32 scram-sha-256
```


Authentification – Modes — *système*

trust	pas d'authentification !	*déblocage...*
scram-sha-256	mot de passe géré par pg	
md5 password	idem, versions antérieures	
ident	RFC 1413 (tcp 113) – connexions réseau ou locales	
LDAP	RFC 4510 *Lightweight Directory Access Protocol*	
GSSAPI	RFC 2743 – single sign on	
SSPI	MS Windows – single sign on	
PAM	Pluggable Authentication Modules (Linux, Solaris)	
Kerberos	protocole d'authentification réseau	

Autorisation – Qui — ROLE / USER GROUP

titulaires de droits dans la base de données
- identifié par **nom** (et numéro) pour le *cluster*
- **propriétaire** d'objets (base, schéma, relation)
- droits spéciaux : **super-user**...

rôle abstraction du standard SQL pour **utilisateur** ou **groupe**

utilisateur rôle individuel, peut se **connecter**
- mot de passe associé
- nombre maximal de connexions
- date d'expiration...

groupe rôles contenant des utilisateurs, pas de connection
- organisation des droits
- restrictif (acquisition explicite) ou hérité (automatique)

Autorisation – Création — ROLE

Utilisateur — LOGIN

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

Groupe — NOLOGIN

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


Autorisation – Création — USER GROUP

Utilisateur

```
CREATE USER hobbes
  WITH ENCRYPTED PASSWORD 'grr...'
  CONNECTION LIMIT 3
  VALID UNTIL '2038-01-19 03:14:07 UTC'
  IN GROUP family;
```

Groupe

```
CREATE GROUP family;

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


Autorisation – Héritage — INHERIT vs NOINHERIT

INHERIT droits des groupes **cumulés par défaut**
- comportement par défaut

```
DROP TABLE Eleves;
-- ok
```

NOINHERIT droits des groupes **ignorés par défaut**
- `SET ROLE` explicite pour y accéder
- permet de passer admin temporairement

```
DROP TABLE Eleves;
-- permission denied
SET ROLE admin_oasis;
DROP TABLE Eleves;
-- ok
RESET ROLE;
```

logiquement user `NOINHERIT`, groupe `INHERIT`

Autorisation – Philosophie

- Système de droits SQL très complet...
  - sujets administrateur, propriétaire, utilisateurs, groupes ou `PUBLIC`
  - objets `DATABASE SCHEMA TABLE LANGUAGE FUNCTION`...
  - droits selon les objets, plus **droits d'attribution** des droits
- Deux commandes : `GRANT REVOKE`...
- Permissif par défaut (consultation/utilisation pour `PUBLIC`)
  en évolution selon les versions...

Droits — `DATABASE` / `SCHEMA`

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 — `TABLE`

`SELECT`	autorise (éventuellement sur une colonne)
`INSERT`	idem
`UPDATE`	idem
`DELETE`	idem
`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

Droits — `FUNCTION` / `LANGUAGE`

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)

Autorisation – Gestion des droits GRANT REVOKE

- syntaxe un peu verbeuse, mais compréhensible...
- **permission** sur **objet** pour **sujet** (avec **délégation**)

```
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}[, ...] } ;
```


Manipulation de tables ?

cluster	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...

Exercice

Lecture d'une table

- Comment donner à l'utilisateur *Hobbes*
- le droit en lecture et écriture
- de la table *Models*
- dans la base de données *phones* ?

Droits spéciaux

Droit d'admistration

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)*

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
                        Droits d'accès
+--------+--------+-------+--------------------------+
| Schéma |  Nom   | Type  |      Droits d'accès      |
+--------+--------+-------+--------------------------+
| public | auteur | table | fabien=arwdDxt/fabien    |
|        |        |       | corrector=arwdDxt/fabien |
|        |        |       | eleves=r/fabien          |
|        |        |       | =r/fabien                |
+--------+--------+-------+--------------------------+
```

pg_catalog consultation des tables systèmes

- type interne `aclitem` dans description des objets

Conclusion

modèle très complet et quasi standard

- concerne aussi tous les types d'objets
- existe aussi des droits sur les *lignes* (RLS)

mais droits souvent gérés au niveau **application**

- permissions souvent complexes liées à la logique applicative
- gestion déclarative ou procédurale dans le code des apps

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
TYPE	ajout de nouveaux types (base, composites)
DOMAIN	type + contraintes
LANGUAGE	nouveau langage côté serveur
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. . .)

RLS : *Row-Level Security* 1/2

- 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);
```

RLS : *Row-Level Security* 2/2

- 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 ...
```