



Collectif francophone pour l'enseignement libre de l'informatique

SQL

Rôles et contrôle d'accès

CoFELI:MCED_SQL_12

Luc LAVOIE (luc.lavoie@usherbrooke.ca)

CoFELI/Scriptorum/SQL_12-Controle-d-acces, version 1.0.1.b, en date du 2025-11-12

— document de travail, ne pas citer —

Sommaire

Ce module présente les mécanismes et les éléments de langages propres à SQL relativement au contrôle d'accès.

Mise en garde

Le présent document est en cours d'élaboration ; en conséquence, il est incomplet et peut contenir des erreurs.

Historique

diffusion	resp.	description
2025-11-12	LL	Revue.
2025-05-06	LL	Ménage.
2023-11-01	LL	Ébauche initiale selon les formats contenus dans /Users/Shared/DGS/CoLOED/AsciiDoc/format.

Table des matières

Introduction.....	4
1. Le dialecte PostgreSQL.....	4
1.1. Éléments de langage	4
1.2. Authentification.....	7

Introduction

La norme ISO reprend pour l'essentiel le modèle présenté dans le module SGBD_12. Malheureusement plusieurs dialectes s'en écartent, ce qui rend difficilement transportable la mise en oeuvre d'une politique de sécurité.

Les éléments de langages et les dispositifs du standard ISO sont donc présentés dans le but de faciliter la comparaison entre les différents dialectes. Nous documenterons les autres langages en les comparant chacun avec la norme ISO, plutôt qu'entre eux.

Pour le moment, seul le dialecte PostgreSQL v18 est présenté, d'autres suivront.

1. Le dialecte PostgreSQL

PostgreSQL a fusionné les concepts ISO de USER, GROUP et ROLE.

Un USER est un ROLE doté des privilèges et défini par des paramètres de connexion.

Un GROUP est un ROLE doté de privilèges et défini par une liste de ROLE, la récursivité n'étant pas permise (directement ou indirectement).

Les autres ROLE sont... des ROLE (au sens ISO).

La source première du contenu de la présente section est [PostgreSQL18a], plus précisément le chapitre I de la partie IV (manuel de référence) pour la syntaxe (<https://docs.postgresql.fr/18/sql-commands.html>), et les chapitres 20 et 21 de la partie III (manuel d'administration du serveur) pour les fondements et les règles de pratique recommandées (<https://docs.postgresql.fr/18/client-authentication.html> et <https://docs.postgresql.fr/18/user-manag.html>).

1.1. Éléments de langage

1.1.1. ROLE (USER, GROUP)

Définir (indéfinir) un rôle et lui associer des droits d'accès.

- CREATE ROLE nom [[WITH] Option [...]]
- DROP ROLE [IF EXISTS] nom [, ...]

Les options sont divisées en trois groupes (type, connexion, comportement).

- Option — de type (en relation avec le SGBD)
 - [NO]SUPERUSER
 - [NO]CREATEDB
 - [NO]CREATEROLE
 - [NO]LOGIN
 - [NO]REPLICATION
 - [NO]BYPASSRLS
 - Option — de connexion
 - [ENCRYPTED] PASSWORD 'motdepasse' | PASSWORD NULL
 - CONNECTION LIMIT limite_connexion
 - VALID UNTIL 'heuredate'
 - Option — de comportement
 - [NO]INHERIT
- Quand l'instruction GRANT est utilisée pour définir l'appartenance d'un rôle à un autre, l'instruction GRANT peut utiliser la clause WITH INHERIT pour indiquer si les droits du rôle

bénéficiaire doivent être « hérités » par le nouveau membre. Si l'instruction GRANT n'indique pas explicitement le comportement de l'héritage, l'instruction GRANT sera créée avec WITH INHERIT TRUE si le rôle membre est configuré avec INHERIT et avec WITH INHERIT FALSE s'il est configuré avec NOINHERIT.

Avant PostgreSQL v16, l'instruction GRANT n'acceptait pas WITH INHERIT. Du coup, modifier la propriété au niveau rôle changeait aussi le comportement des droits déjà existants. Ce n'est plus le cas.

- IN ROLE nom_role [, ...]

La clause IN ROLE fait que le nouveau rôle se voit ajouté automatiquement comme membre des rôles existants cités.

Il n'existe pas d'option pour ajouter le nouveau rôle en tant qu'administrateur; il faut utiliser une commande GRANT séparée pour cela.

- ROLE nom_role [, ...]

La clause ROLE fait que les rôles existants cités sont ajoutés automatiquement comme membre du nouveau rôle. Ceci a pour effet de transformer le nouveau rôle en « groupe ».

- ADMIN nom_role [, ...]

Cette clause est équivalente à la clause précédente (ROLE), à la différence que les rôles nommés sont ajoutés au nouveau rôle avec le privilège de pouvoirs promouvoir d'autres rôles au sein de celui-ci.

Liste des rôles (psql)

La commande suivante \du permet d'obtenir la liste courante de tous les rôles; elle ne peut être exécutée que depuis une session ouverte par un administrateur du SGBD ayant tous les priviléges, typiquement l'utilisateur postgres.

1.1.2. ALTER

Modifier un rôle, en modifier les droits d'accès associés.

Les possibilités sont les suivantes:

- ALTER ROLE nom RENAME TO nouveau_nom
- ALTER ROLE spécification_rôle [WITH] Option [...]
- ALTER ROLE spécification_rôle_complet SET paramètre_configuration { TO | = } { value | DEFAULT }
- ALTER ROLE spécification_rôle_complet SET paramètre_configuration FROM CURRENT
- ALTER ROLE spécification_rôle_complet RESET paramètre_configuration
- ALTER ROLE spécification_rôle_complet RESET ALL

où

```
spécification_rôle_complet ::=  
  { spécification_rôle | ALL } [ IN DATABASE nom_base ]  
  
spécification_rôle ::=  
  nom_rôle  
  | CURRENT_ROLE  
  | CURRENT_USER  
  | SESSION_USER
```

Ces quatre rôles se distinguent ainsi:

- nom_role : un des rôles spécifiquement défini dans la BD ;
- SESSION_USER : le rôle (de type USER) attaché à la session par laquelle la transaction courante a été soumise ;

- CURRENT_USER : le rôle (de type USER) sous lequel la transaction courante a été soumise ;
- CURRENT_ROLE : le rôle (pas forcément de type USER) sous lequel la transaction courante est couramment exécutée ;

Les rôles SESSION_USER et CURRENT_USER sont donc dotés de paramètres de connexion et ont été préalablement authentifiés.

À un instant donné les trois derniers rôles (CURRENT_ROLE, CURRENT_USER, SESSION_USER) peuvent être différents :

- SESSION_USER est un rôle détenu par une application qui ouvre les sessions pour ses clients (qui ne le peuvent peut-être pas). On s'assure ainsi que la session est ouverte par une application légitime.
- CURRENT_USER est un rôle détenu par un utilisateur au nom duquel la transaction est soumise par l'application via une session.
- CURRENT_ROLE est un rôle pris par le CURRENT_USER en cours de transaction, soit directement (SET ROLE) soit indirectement (via une routine définie comme SECURITY DEFINER).

1.1.3. GRANT, REVOKE

Attribution (retrait) d'un rôle à un autre.

```
GRANT nom_role [, ...] TO spécification_rôle [, ...] +
[ WITH { ADMIN | INHERIT | SET } ] +
[ GRANTED BY spécification_rôle ]
```

où

```
spécification_rôle ::=  
nom_rôle  
| PUBLIC  
| CURRENT_ROLE  
| CURRENT_USER  
| SESSION_USER
```

Attribution (retrait) d'un droit d'accès à un rôle

```
GRANT droit-accès +
ON type-d'objet objet [, ...] +
TO spécification_rôle +
[ WITH GRANT OPTION ] +
[ GRANTED BY spécification_rôle ]
```

où

```
droit-accès ::=  
SELECT  
| INSERT  
| UPDATE  
| DELETE  
| TRUNCATE  
| REFERENCES  
| TRIGGER  
| CREATE  
| CONNECT  
| TEMPORARY  
| EXECUTE
```

```
| USAGE  
| SET  
| ALTER SYSTEM  
| ...  
  
type-d'objet ::=  
    TABLE  
    | TYPE  
    | ROUTINE  
    | ...
```



Voir la documentation de PostgreSQL pour déterminer les paires (droit-accès, type-objet) légitimes et le sens précis du droit en regard du type d'objet.

1.1.4. SET ROLE

Changement de rôle en cours de session.



Voir la documentation de PostgreSQL pour déterminer les modalités du changement de rôle.

1.1.5. REASSIGN

Tout objet se voit attribuer un propriétaire au moment de sa création, cette instruction permet d'en changer. Le propriétaire d'un objet hérite automatiquement de certains droits.



Voir la documentation de PostgreSQL pour déterminer les modalités du changement de propriétaire.

1.1.6. Les rôles prédéfinis

Certains rôles sont prédéfinis au niveau du SGBD. Ceux-ci varient d'un dialecte à un autre.



Voir la documentation de PostgreSQL pour connaitre les rôles prédéfinis.

1.1.7. La définition des routines (INVOKER et DEFINER)

[EXTERNAL] SECURITY INVOKER

Spécifie que la fonction est exécutée avec les droits de l'utilisateur qui l'appelle. C'est la valeur par défaut.

[EXTERNAL] SECURITY DEFINER

Spécifie que la fonction est exécutée avec les droits de l'utilisateur qui en est le propriétaire (OWNER).

Le mot clé EXTERNAL est autorisé pour la conformité SQL, mais il est optionnel, car, contrairement à SQL, cette fonctionnalité s'applique à toutes les fonctions, pas seulement celles externes.

— PG, Partie VI - Référence - CREATE FUNCTION

1.2. Authentification

PostgreSQL offre quantité de méthodes d'authentification différentes. La méthode utilisée pour authentifier une connexion client particulière peut être sélectionnée d'après l'adresse (du client), la base de données et l'utilisateur.

Les noms d'utilisateur de bases de données sont séparés de façon logique des noms d'utilisateur du système d'exploitation sur lequel tourne le serveur. Si tous les utilisateurs d'un serveur donné ont aussi des comptes sur la machine serveur, il peut être pertinent d'attribuer aux utilisateurs de bases de données des noms qui correspondent à ceux des utilisateurs du système d'exploitation. Cependant, un serveur qui accepte des connexions distantes peut avoir des utilisateurs de bases de données dépourvus de compte correspondant sur le système d'exploitation. Dans ce cas, aucune correspondance entre les noms n'est nécessaire.

— PG, chapitre 21

PostgreSQL offre quantité de méthodes d'authentification différentes. La méthode utilisée pour authentifier **le client associé à une connexion** particulière peut être sélectionnée d'après l'adresse **d'origine de la connexion**, la base de données et le **nom d'utilisateur associé à la connexion**.

Pour plus de détails, voir les modules MCED:SGBD_13-Autentification et MCED:SQL_13-Authetification.

— LL, 2024-04-01

Produit le 2025-11-13 06:18:24 -0500



Collectif francophone pour l'enseignement libre de l'informatique