Base de données

Cours 14 - Gestion et sécurité des usagers

Table des matières

Pourquoi la gestion des utilisateurs est cruciale ?

- **Protection des données sensibles** : Empêche l'accès non autorisé aux informations confidentielles
- **Conformité légale** : Respect des réglementations (RGPD, HIPAA, PCI-DSS)
- **Principe du moindre privilège** : Chaque utilisateur ne reçoit que les permissions nécessaires à sa fonction
- **Audit et traçabilité** : Permet de savoir qui a fait quoi et quand
- **Prévention des erreurs humaines** : Limite les dégâts causés par des opérations accidentelles

Conséquences d'une mauvaise gestion

Les conséquences d'une mauvaise gestion des utilisateurs peuvent être graves :

- Fuites de données sensibles
- Corruption ou perte de données
- Non-conformité aux réglementations
- Perte de réputation et sanctions financières
- Arrêt de service (DoS accidentel ou intentionnel)

Qu'est-ce qu'un rôle utilisateur ?

Un rôle est une entité qui peut posséder des privilèges et des permissions pour accéder et manipuler les objets de la base de données.

Un rôle peut représenter un utilisateur individuel ou un groupe d'utilisateurs.

Rôles vs Utilisateurs

Dans PostgreSQL, la distinction entre `ROLE` et `USER` a été supprimée depuis la version 8.1.

Donc : `CREATE USER nom` = `CREATE ROLE nom WITH LOGIN`

La seule différence est qu'un `USER` est un `ROLE` avec la capacité de se connecter à la base de données par défaut.

Création de rôle utilisateur

La commande pour créer un utilisateur (rôle avec capacité de connexion) est la suivante :

```
1  -- Creation of a basic user
2  CREATE USER employee_dev WITH
3      PASSWORD 'mot_de_passe_securise';
4
5  -- Equivalent using CREATE ROLE
6  CREATE ROLE employee_dev WITH
7      LOGIN
8      PASSWORD 'mot_de_passe_securise';
```

Listing: https://www.postgresql.org/docs/current/sql-createrole.html

Création de rôle de groupe

Il est souvent pratique de regrouper les utilisateurs pour faciliter la gestion des privilèges : ainsi, les privilèges peuvent être accordés ou révoqués à un groupe dans son ensemble.

Dans PostgreSQL, cela se fait en créant un rôle qui représente le groupe, puis en accordant l'appartenance à ce rôle de groupe à des rôles utilisateur individuels.

Création de rôle de groupe

La commande pour créer un rôle de groupe est la suivante :

```
1  -- Creation of a group role
2  CREATE ROLE developers;
3  CREATE ROLE administrators;
4
5  -- Add a user to a group
6  GRANT developers TO employee_dev;
```

Listing: https://www.postgresql.org/docs/current/sql-createrole.html

Attributs des rôles

Les rôles peuvent avoir plusieurs attributs qui définissent leurs permissions et capacités :

- **SUPERUSER** : Accès complet
- **CREATEDB** : Peut créer des bases de données
- **CREATEROLE** : Peut créer des rôles
- **LOGIN** : Peut se connecter
- **CONNECTION LIMIT** : Limite de connexions
- **PASSWORD** : Mot de passe
- **VALID UNTIL** : Expiration
- **INHERIT** : Héritage des privilèges

Création de rôle avec attributs

Voici un exemple complet de création d'un rôle avec plusieurs attributs :

```
1  CREATE ROLE admin_application WITH
2      LOGIN       -- can connect to the database
3      PASSWORD 'MotDePasseComplexe123!'
4      CREATEDB    -- can create databases
5      CREATEROLE -- can create roles
6      CONNECTION LIMIT 10   -- limit to 10 concurrent connections
7      VALID UNTIL '2025-12-31'; -- account expiration date
```

Listing: https://www.postgresql.org/docs/current/sql-createrole.html

Suppression des rôles

Comme les rôles peuvent posséder des objets de la base de données et détenir des privilèges pour accéder à d'autres objets, supprimer un rôle n'est souvent pas qu'une simple question d'exécuter `DROP ROLE`.

Il est important de s'assurer de réaffecter ou supprimer les objets et privilèges associés avant de supprimer un rôle.

Suppression des rôles

Voici comment supprimer un rôle en toute sécurité :

```
1  -- Reaffecting ownership of objects to another role
2  REASSIGN OWNED BY old_dev TO new_dev;
3
4  -- Revoke all privileges granted to the old role
5  DROP OWNED BY old_dev;
6
7  -- Finally, drop the old role
8  DROP ROLE old_dev;
```

Listing: https://www.postgresql.org/docs/current/sql-createrole.html

Qu'est-ce qu'un privilège ?

Un privilège est une permission accordée à un rôle et/ou un utilisateur pour effectuer des actions spécifiques sur des objets de la base de données (tables, vues, fonctions, etc.).

Les privilèges permettent de contrôler l'accès et les opérations que les utilisateurs peuvent effectuer.

Hiérarchie des privilèges

Les privilèges sont organisés hierarchiquement :

1. Privilèges au niveau de la base de données
2. Privilèges au niveau du schéma
3. Privilèges au niveau des objets (tables, vues, fonctions)
4. Privilèges au niveau des colonnes

L'héritage des privilèges se fait du niveau supérieur vers l'inférieur, 1 étant le plus élevé dans ce cas-ci.

Types de privilèges principaux

Voici les principaux types de privilèges que l'on peut accorder ou révoquer:

- **SELECT** : Lire des données
- **INSERT** : Ajouter des données
- **UPDATE** : Modifier des données
- **DELETE** : Supprimer des données
- **TRUNCATE** : Vider une table
- **REFERENCES** : Créer des clés étrangères
- **EXECUTE** : Exécuter des fonctions

GRANT - Syntaxe de base

La commande pour attribuer des privilèges est la suivante :

```
1 GRANT { { SELECT | INSERT | UPDATE | DELETE | ... | ALL } [, ...] }
2     ON { [ TABLE ] table_name [, ...] }
3     TO { [ GROUP ] role_name | PUBLIC } [, ...]
4     [ WITH GRANT OPTION ];
```

Listing: https://www.postgresql.org/docs/current/sql-grant.html

GRANT - Exemples

Exemples d'attribution de privilèges :

```sql
-- Only grants SELECT privilege on a table to a user
GRANT SELECT ON TABLE clients TO employee_dev;

-- Grants many privileges on a table to a role
GRANT SELECT, INSERT, UPDATE ON orders TO developers;

-- Grants all privileges on a table to a role
GRANT ALL ON TABLE products TO administrators;

-- Privileges on all tables in a schema to a user
GRANT SELECT ON ALL TABLES IN SCHEMA public TO employee_dev;
```

Listing: https://www.postgresql.org/docs/current/sql-grant.html

REVOKE - Syntaxe

La commande pour révoquer des privilèges est la suivante :

```
1  REVOKE [ GRANT OPTION FOR ]
2      { { SELECT | INSERT | ... | ALL } [, ...] }
3      ON { [ TABLE ] table_name [, ...] }
4      FROM { [ GROUP ] role_name | PUBLIC } [, ...]
5      [ CASCADE | RESTRICT ];
```

Listing: https://www.postgresql.org/docs/current/sql-revoke.html

REVOKE - Exemples

Exemples de révocation de privilèges :

```sql
-- Revoke specific privilege from a user
REVOKE INSERT ON orders FROM employee_dev;

-- Revoke all privileges from a role
REVOKE ALL ON products FROM developers;

-- Revoke with CASCADE (also revokes delegated privileges)
REVOKE SELECT ON clients FROM admin_application CASCADE;
```

Listing: https://www.postgresql.org/docs/current/sql-revoke.html

Meilleures pratiques

1. **Principe du moindre privilège** : Donner uniquement les permissions nécessaires
2. **Utiliser des groupes** : Centraliser la gestion des permissions
3. **Revue régulière** : Auditer périodiquement les privilèges
4. **Mots de passe forts** : Politique de mots de passe robustes
5. **Expiration des comptes** : Pour les comptes temporaires
6. **Séparation des environnements** : Différents utilisateurs pour dev/test/prod

Bibliographie

- https://www.postgresql.org/docs/current/user-manag.html
- https://www.postgresql.org/docs/current/sql-createrole.html
- https://www.postgresql.org/docs/current/role-attributes.html