SGBD : Exploitation d'une base de données [R206]

Hocine ABIR

March 29, 2022

IUT Villetaneuse
E-mail: abir@iutv.univ-paris13.fr

CONTENTS

CHAPTER

QUATRE

Droits d'accès et Privilèges

4.1 Introduction

4.1.1 Blabla

Dans une organisation, des rôles sont créés pour différentes tâches. Les permissions pour effectuer certaines opérations sont attribuées à des rôles spécifiques. Par exemple dans le département d'informatique, les *directeurs des études* ont un rôle particulier, et par ce rôle ils acquièrent le droit (permission) d'assurer (ou d'éffectuer) certaines fonctions (tâches) dans l'organisation (*département*). Les droits ne sont pas affectés directement á chaque personne, mais seulement les acquièrent par le biais de leur(s) rôle(s). La gestion des droits et permissions de chaque personne (ou individu) de l'organisation est rendu plus simple en affectant les rôles appropriés á chaque individu.

Des règles de restrictions (ou contraintes) sur les droits potentiellement hérités de rôles opposés peuvent être nécessaires. Par exemple, les rôles *enseignant* et *etudiant* sont exclusifs.

4.1.2 Introduction

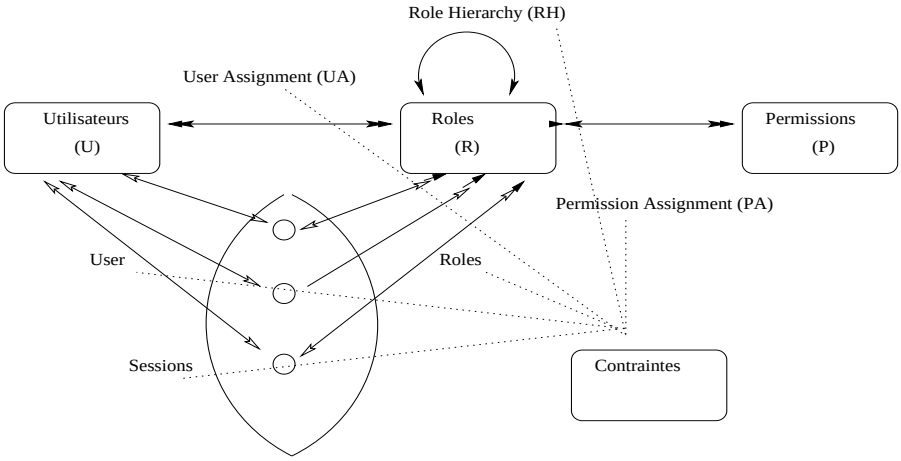
Le Contrôle d'accès est la méthode (ou approche) utilisée par un Système de Gestion de base de données pour restreindre les accès (opérations sur les objets et/ou les données) aux utilisateurs autorisés.

Dans un Système de Gestion de Base de Données, cette approche de la gestion des tâches (permissions ou droits) et des individus (utilisateurs) est appelée RBAC (Role Based Access Control).

Une liste de permissions est attachée à chaque objet (Access Control List ou ACL). Chaque élément de la liste est un triplet qui spécifie :

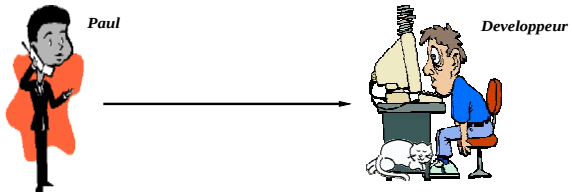
1. Qui est autorisé ?
2. Quel mode d'accès ?
3. Comment cette autorisation est obtenue (propriétaire ou ayant droit) ?

4.2 Modèle RBAC (Role Based Acces Control)

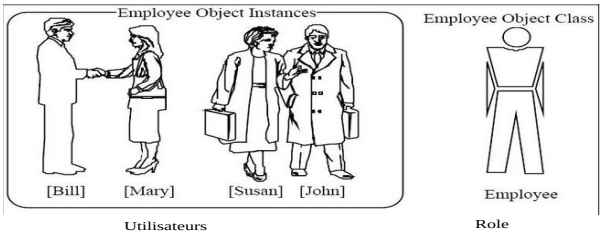


4.2.1 Utilisateur et Role

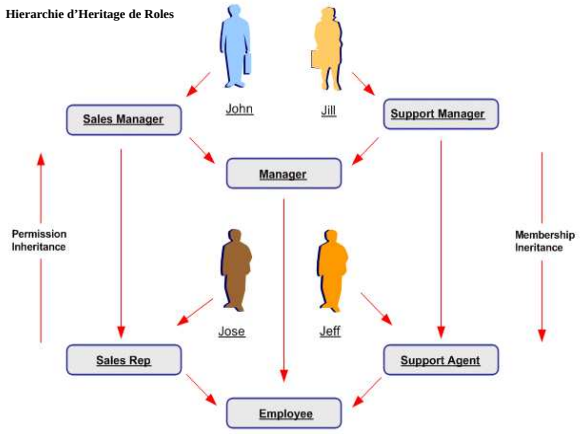
- Utilisateur : par exemple *Paul*, *Jean*
- Role : par exemple *etudiant*, *enseignant*



Un utilisateur peut être membre de plusieurs rôles et un rôle peut être attribué à plusieurs utilisateurs.



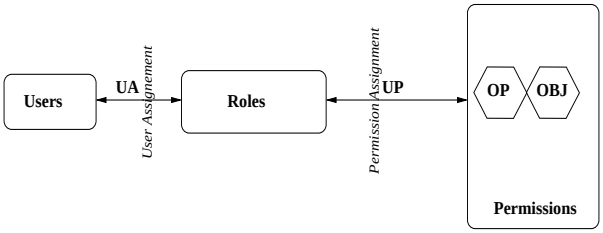
Un rôle peut être membre de plusieurs rôles : hiérachie. Un rôle peut hériter des rôles dont il est membre (union des permissions) ou pas (limité aux permissions de son rôle actuel).



Les rôles inférieurs s'appellent 'junior' et les supérieurs 'senior'.

4.2.2 Permission

Une permission est un mode d'accès particulier accordé (autorisé) sur un ou plusieurs objets à un ou plusieurs rôles. Permission est synonyme d'autorisation, droit d'accès ou privilège. Une permission est donc un couple [objet,methode] ou [classe,methode].



4.2.3 Session

Une session est une association entre un utilisateur et un rôle. Un utilisateur peut activer un ou plusieurs rôles durant une même session. Un utilisateur peut ouvrir plusieurs sessions mais une session est associée à un et un seul utilisateur.

4.2.4 Contrainte

Une contrainte est régle sur l'ensemble ou une partie des associations décrite ci-dessus. Nombre d'utilisateur d'un rôle, nombre de session, rôle prérequis, rôles mutuellement exclusif, contrainte sur la hierachie etc.

Les relations d'exclusion mutuel sont définies par la séparation des contraintes duty. les contraintes SOD (Separation Of Duty - SOD) peuvent être divisées en

- Contrainte statique ou SSD (Static Separation of Duty) : qui spécifie que deux rôles (ou permissions) mutuellement exclusifs ne peuvent être attribué simultanément à un rôle
- Contrainte dynamique ou DSD (Dynamic Separation of Duty) : qui spécifie que deux rôles (ou permissions) mutuellement exclusifs ne peuvent être activés simultanément par un même rôle. Par contre, deux rôles mutuellement exclusifs peuvent être attribué simultanément à un rôle.

Exemple de Types de contraintes :

- Séparation de tâches (separation of duties) Exemple: Celui qui approuve un cheque ne peut pas être celui qui le signe
- Contraintes temporelles Un docteur ne peut être admis en salle d'opération que entre 8:00 et 18:00
- Autres: vaste éventail de possibilités Aucun usager ne peut avoir plus de 5 roles Aucun groupe de moins de 3 usagers ne peut couvrir toutes les permissions existantes dans un département

4.3 RBAC sous Postgres

Attribution des rôles

1. Un rôle peut ajouter un membre utilisateur
2. Une session peut avoir plusieurs rôles actifs en même temps
3. On peut spécifier le rôle par défaut d'un utilisateur

Relations entre rôles et contraintes

4. on peut construire une hiérachie des rôles
5. on ne peut pas spécifier une séparation statique des contraintes sur les rôles et permissions (duties)
6. on ne peut pas spécifier une séparation dynamique des contraintes sur les rôles et permissions (duties)
7. on peut spécifier une cardinalité maximum ou un minimum pour un rôle

Attribution des privilèges

8. on peut attribuer des privilèges système à un rôle,
9. on peut attribuer des privilèges sur les objets à un rôle

4.3.1 Objets

PostgreSQL identifie plusieurs classes d'objets :

1. TABLE
2. DATABASE
3. FUNCTION
4. LANGUAGE
5. SCHEMA
6. TABLESPACE

4.3.2 Privilèges

Un Privilge est une autorisation (droit) d'exécution d'une commande SQL particulière ou un mode d'accès à un objet par un utilisateur.

- SQL identifie un ensemble trés détaillé de privilges sur les objets (relations, etc.).
- Onze privilèges en tout sur différents objets.

Exemple : Objets de classe TABLE

Mode d'Accès	Permission
SELECT	interroger la table
INSERT	Insérer dans la table
UPDATE	Modifier les tuples de la table
DELETE	Supprimer des tuples de la table
RULE	Définir des règles sur la table
REFERENCES	Référencier la table dans une clé étrangère
TRIGGER	Définir des déclencheurs (trigger) sur la table

4.3.3 Utilisateurs et rôles

Un rôle est une entité qui regroupe des privilèges et d'autres rôles. L'objectif des rôles est de faciliter la gestion des droits d'accès et des utilisateurs en regroupant différents priviligèges nécessaires pour effectuer un ensemble d'opérations. Sous PostgreSQL, le concept de rôle subsume le concept d'utilisateur et de groupe c'est à dire qu'un :

- utilisateur : est un rôle ayant une option particulière (LOGIN).
- groupe : est un rôle dont tous les membres sont des utilisateurs.

Sous PostgreSQL :

- un rôle (et/ou utilisateur) est désigné par un identifiant (par exemple le nom de l'utilisateur ou de la fonction),
- Il y a un identifiant (rôle) PUBLIC. Les privilèges attribués à l'identifiant PUBLIC sont hérité par tout identifiant (utilisateur ou rôle).

4.3.4 Caractéristiques d'un rôle

Privilèges système

- SUPERUSER: Un rôle ayant cet attribut tous les privilèges possibles (Super-Utilisateur). Initialement, seul le rôle système `postgres` possède cet attribut. Ce rôle a tous les privilèges possibles du système. Par défaut NOSUPERUSER.

```
CREATE ROLE nom_role SUPERUSER;
```

- LOGIN : seuls les rôles ayant cet attribut peuvent être utilisé pour se connecter à une base de donées. cet attribut transforme un rôle en utilisateur. Par défaut NOLOGIN.

```
CREATE ROLE nom_role LOGIN;
```

- CREATEDB : Cet attribut autorise un rôle à créer des bases de données. Par défaut NOCREATEDB

```
CREATE ROLE nom_rolen CREATEDB.
```

- CREATEROLE : Cet attribut autorise un rôle à créer de nouveaux rôles , modifier la composition d'un rôle (privilèges et membres). Par défaut NOCREATEROLE

```
CREATE ROLE nom_role CREATEROLE
```

- INHERIT : Cet attribut permet à un rôle d'hériter tous les privilèges des rôles dont il est membre direct ou indirect. Par défaut INHERIT. Si cet attribut est positionné à NOINHERIT alors l'utilisateur est autorisé à spécifier son rôle (ou changer de rôle) par la commande SET ROLE :

```
SET [ SESSION | LOCAL ] ROLE nom_role
```

  - SESSION : session (connexion)
  - LOCAL : transaction

Ces caractéristiques peuvent être modifiées par la commande :

```
ALTER ROLE nom_role [ [ WITH ] option [ ... ] ]
```

Caractéristiques valuées

- CONNECTION LIMIT max_con : permet de sécifier le nombre maximum de connexions concurrentes max_con que le rôle peut effectuer. Cette attribut n'a de sens que si le rôle a l'attribut LOGIN. Par défaut (-1) aucune limite.

```
CREATE ROLE nom_role CONNECTION LIMIT  max_con;
```

- `VALID UNTIL 'estampille'` : permet de sécifier une date (et heure) limite de validiter du rôle. Par défaut, aucune limite.

```
CREATE ROLE nom_role VALID UNTIL estampille';
```

- `PASSWORD mot_de_passe`: permet d'associer un mot de passe à un rôle. Cet attribut n'a de sens que si :
  - le rôle a l'attribut `LOGIN`
  - la méthode d'authentification exige un mot de passe.

```
CREATE ROLE nom_role PASSWORD 'mot_de_passe'.
```

Ces paramètres peuvent être modifiées par la commande :

```
ALTER ROLE nom_role SET parametre { TO | = } { valeur | DEFAULT }
```

4.3.5 Hiéarchies des rôles

Motivation

- Les privilèges associés aux rôles peuvent se recouvrir : c'est à dire que des utilisateurs appartenant à différents rôles peuvent effectuer des opérations communes,
- Certains opérations peuvent être effectuées par tous les utilisateurs,

Si l'on spécifie répétivement ces opérations pour chaque rôle, la situation deviendrai difficile à gérer et inefficace.

Une Hiéarchie des rôles peut être établie pour représenter la structure natuel de ces rôles. Une Hiéarchie des rôles définit des rôles qui :

- ont des attributs uniques,
- peuvent contenir d'autres rôles : c'est à dire inclure implicitement des opérations qui sont associées à d'autres rôles.

Construction de la Hiéarchie

- `IN ROLE role_parent`: cette clause permet de définir un nouveau rôle `nom_role` comme membre du rôle `role_parent`.

```
CREATE ROLE nom_role IN ROLE role_parent;
```

- `ROLE role_membre` : cette clause permet de définir un nouveau rôle `nom_role` ayant comme membre `role_membre`.

```
CREATE ROLE nom_role ROLE role_membre;
```

- `ADMIN role_membre` : cette clause permet de définir un nouveau rôle `nom_role` ayant comme membre `role_membre`. De plus, le rôle `role_membre` peut modifier les membres de `nom_role` (ajout, suppression).

```
CREATE ROLE nom_role ADMIN role_membre;
```

Les membres d'un rôle peuvent être modifiés avec les commandes `GRANT ROLE` et `REVOKE ROLE`.

Ajout

```
GRANT role [, ...]
TO username [, ...]
[ WITH ADMIN OPTION ]
```

- ajoute le rôle `username` comme membre du rôle `role`
- si `WITH ADMIN OPTION` est spécifié alors le membre `username` peut modifier la liste des membre du rôle`role`.

Retrait

```
REVOKE [ ADMIN OPTION FOR ]
    role [, ...] FROM username [, ...]
    [ CASCADE | RESTRICT ]
```

- retirer le rôle `username` du rôle `role` (`username` n'est plus membre de `role`)
- si `CASCADE` est spécifié alors tous les membres que `username` a ajouté dans le rôle `role` sont aussi retirés récursivement.
- si `ADMIN OPTION FOR` est spécifié alors le rôle `username` peut retirer des rôles qui lui sont membres.

Exemple de Hiérachie

4.3.6 Acquisition de Privilèges

- un utilisateur a tout les privilèges possibles sur un objet qu'il a créé.

- un utilisateur peut attribuer des privilèges à d'autres utilisateurs (y compris PUBLIC).
- Avec l'option WITH GRANT OPTION un utilisateur permet à un autre utilisateur ayant acquis des privilèges de pouvoir lui aussi les attribuer à d'autres utilisateurs.

La commande GRANT permet d'attribuer des priviléges. Sa structure comporte :

```
GRANT  <liste de privileges>
       ON  <relation ou autre objet>
       TO  <liste de utilisateurs>
       [WITH GRANT OPTION ];
```

L'option WITH GRANT OPTION permet d'autoriser le(s) utilisateur(s) à attribuer leurs droits à leur tour.

Objet : TABLE nomtable

Privilége	Description
SELECT	permission d'interroger (SELECT) une table, une vue ou une séquence (fonction currval)
INSERT	permission d'insérer (INSERT) dans une table ou vue
UPDATE	permission de modifier (UPDATE) une ou plusieurs colonnes des tuples d'une table. SELECT ... FOR UPDATE et SELECT ... FOR SHARE nécessite ce privilège
DELETE	permission de supprimer (DELETE) des tuples d'une table ou vue
RULE	permission de définir des règles sur une table ou vue (CREATE RULE)
REFERENCES	permission de référencier la table dans une clé étrangère (clause REFERENCES).
TRIGGER	permission de définir des déclencheurs (trigger) sur la table (CREATE TRIGGER)

Objet : DATABASE nombd

Privilége	Description
CREATE	permission de créer un nouveau schéma dans la base de données (CREATE SCHEME)
TEMPORARY	permission de créer une table temporaire (CREATE TEMPORARY TABLE)

Objet : SCHEMA nomschema

Privilége	Description
CREATE	permet de créer de nouveaux objets dans le schéma
USAGE	permet l'acces aux objets du schéma (d'autres privilèges sont nécessaires)

Objet : `TABLESPACE` nomtablespace

Privilége	Description
`CREATE`	permet de créer des index et des tables dans le tablespace

Objet : `FUNCTION` nomfonction

Privilége	Description
`EXECUTE`	permet d'utiliser une fonction et les opérateurs implémentés en utilisant cette fonction

Objet : `LANGUAGE` nomlangage

Privilége	Description
`USAGE`	autorise l'utilisation du langage pour la création de fonctions

Exemple de `GRANT`

4.3.7 Restriction de Privilèges

La commande `REVOKE` permet de retirer des priviléges à des rôles. Sa description est analogue à `GRANT` :

```
REVOKE  [ GRANT OPTION FOR ]
        <liste de privileges>
        ON    <relation ou autre objet>
        FROM  <liste de utilisateurs>
        [CASCADE | RESTRICT ];
```

Exemple de `REVOKE`

4.4 Diagramme des Privilèges (ou `GRANT`s)

4.4.1 Définition

Un Diagramme des Privilges est un graphe orienté construit en utilisant les règles suivantes :

- Noeuds : Un noeud est un triplet (rôle, privilege [,option et/ou proprietaire]) où le troisième terme désigne :
  - l'option : indique si le rôle a l'attribut `GRANT OPTION`,
  - le proprietaire : indique si le rôle est proprietaire de l'objet,
  - sinon rien.

  Un noeud représentant un rôle R ayant un privilège P est noté RP

- Si le privilège P a l'option `GRANT OPTION`, le noeud est noté RP^*
- Si A est le propriétaire de l'objet désigné par le privilège P le noeud est noté RP^{**}.

  Remarque :
  * $RP^{**} \Rightarrow RP^*$
  * RP^{**}, RP^*, RP sont trois noeuds distincts

Notation des modes d'accès:

- r = SELECT ("read")
- w = UPDATE ("write")
- a = INSERT ("append")
- d = DELETE
- R = RULE
- x = REFERENCES
- t = TRIGGER
- X = EXECUTE
- U = USAGE
- C = CREATE
- T = TEMPORARY

- Arcs :

1. Quand un rôle R^i attribut (grant) un privilège P à un rôle R^j :
   un des arcs suivant est construit :
   - $R^iP^* \longrightarrow R^jP$
   - ou $R^iP^{**} \longrightarrow R^jP$
   - ou $R^iP^* \longrightarrow R^jP^*$ (`GRANT OPTION`)
   - ou $R^iP^{**} \longrightarrow R^jP^*$ (`GRANT OPTION`)

2. Quand un rôle R^i retire (`REVOKE`) un privilège P à un rôle R^j avec l'option cascade (`CASCADE`) alors supprimer l'arc $R^iP \longrightarrow R^jP$

3. Quand un rôle R^i retire (`REVOKE`) un privilège P à un rôle R^j avec l'option restreinte (`RESTRICT`) et il existe au moins un arc de la forme $R^jP \longrightarrow R^kP$ alors ne rien faire (graphe inchangé) sinon supprimer l'arc $R^iP \longrightarrow R^jP$

4. Supprimer tous les noeuds Y qui ne sont sur un chemin ayant pour origine un noeud X^{**} (noeud propriétaire) : leurs privilèges sont révokés.

4.4.2 Règle d'autorisation

Un rôle R a un privilège P si et seulement si il y a un chemin du noeud OP^{**} (l'origine du privilège, O est proprétaire) au noeud RP ou RP^* ou RP^{**}.

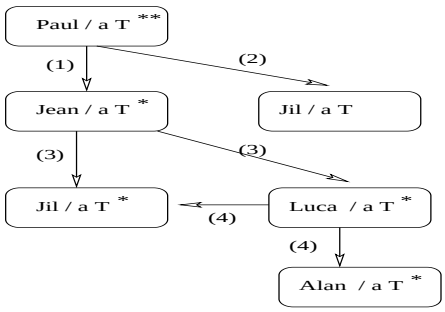
4.4.3 Exemple

Supposons que "*Paul*" est le proprétaire de la table "*T*", et que "*Jean*", "*Jil*", "*Luca*" et "*Alan*" sont quatre autres utilisateurs. On considère la séquence de commandes SQL suivantes :

1. Paul# GRANT INSERT ON T TO Jean WITH GRANT OPTION;
2. Paul# GRANT INSERT ON T TO Jil;
3. Jean# GRANT INSERT ON T TO Jil, Luca WITH GRANT OPTION;
4. Luca# GRANT INSERT ON T TO Jean, Jil, Alan WITH GRANT OPTION;

   qui engendre le diagramme de privilèges suivant :

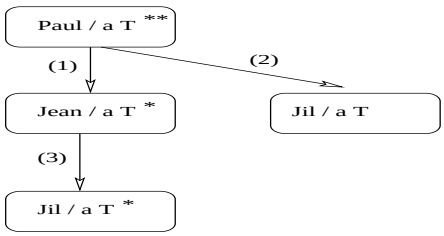
```
# \z t
          Access privileges for database "droit"
 Schema | Name | Type  |           Access privileges
--------+------+-------+-------------------------------
 public | t    | table | {paul=arwdRxt/paul,
                          jean=a*/paul,jil=a/paul,
                          jil=a*/jean,luca=a*/jean,
                          jil=a*/luca,alan=a*/luca}
(1 row)
```



5. Jean# REVOKE INSERT ON T FROM Luca CASCADE;

   qui engendre le diagramme de privilèges suivant :

```
# \z t
          Access privileges for database "droit"
 Schema | Name | Type  |           Access privileges
--------+------+-------+-------------------------------
 public | t    | table | {paul=arwdRxt/paul,
                          jean=a*/paul,jil=a/paul,
                          jil=a*/jean}
(1 row)
```



6. Paul# REVOKE INSERT ON T FROM Jil CASCADE;

   qui engendre le diagramme de privilèges suivant :

```
# \z t
          Access privileges for database "droit"
 Schema | Name | Type  |           Access privileges
--------+------+-------+------------------------------
 public | t    | table | {paul=arwdRxt/paul,
                          jean=a*/paul,
                          jil=a*/jean}
(1 row)
```