Bases de Données

Confidentialité et Droits

Arnaud Delhay-Lorrain

Département Informatique
IUT de Lannion
Université de Rennes

BUT Informatique

Plan du cours

1. Motivations
2. Mécanismes d'octroi et d'annulation des privilèges

Motivations

Mécanismes de sécutité et de protection :

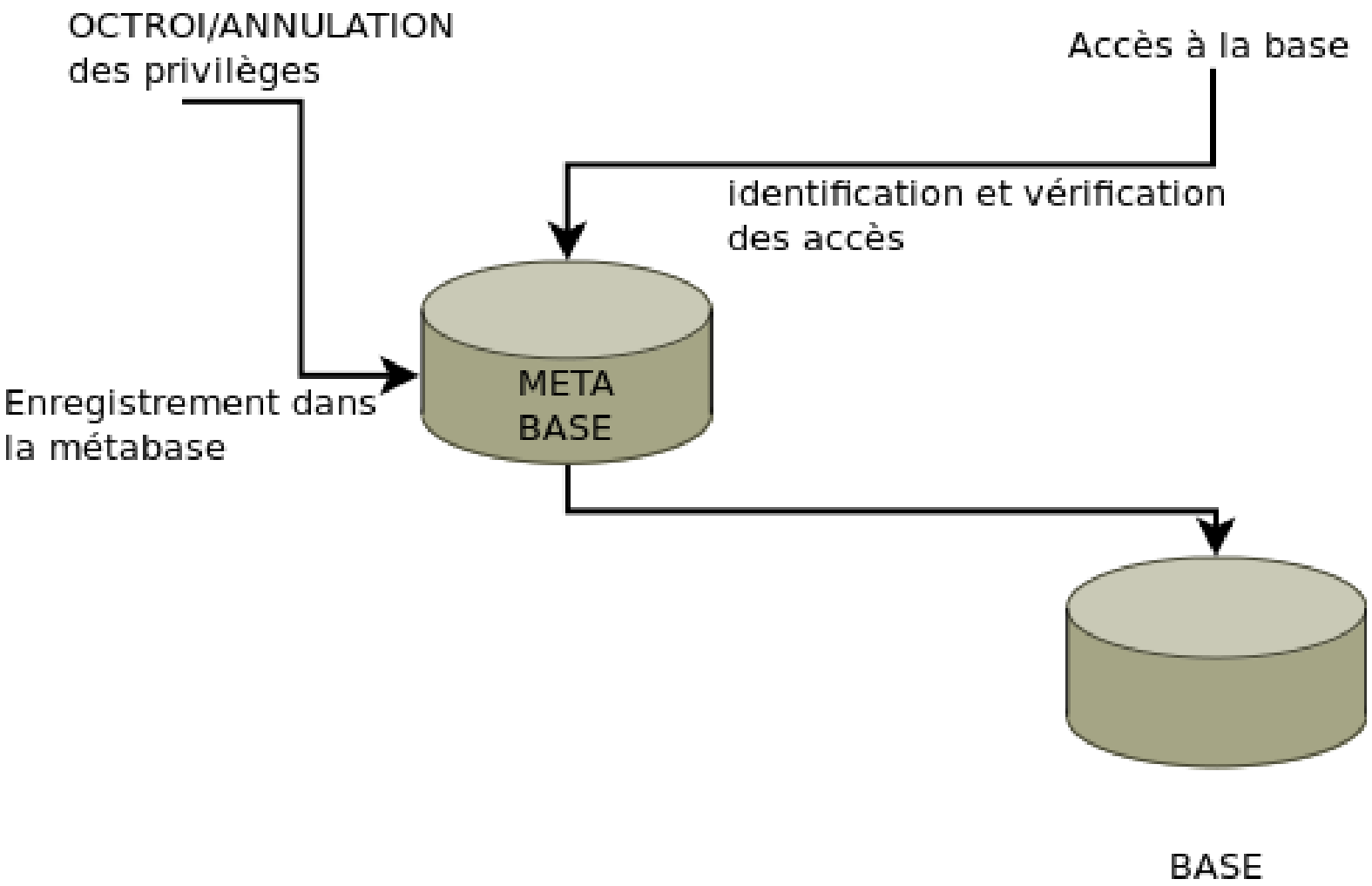
- contrôle d'accès au système par des utilisateurs identifiés
- contrôle d'accès aux données ou confidentialité
- contrôle de la modification valide des données ou intégrité
  - contraintes d'intégrité et triggers
  - transactions

Motivations

Trois grands types de contrôles :

- dépendant du contenant exemple : L'utilisateur peut lire un partie de la table mais ne peut rien modifier.
- dépendant du contexte exemple : l'utilisateur peut lire un seul n-uplet de la table (celui qui le concerne) et peut modifier le contenu de l'attribut "adresse".
- dépendant du contenu exemple : l'utilisateur peut lire le contenu des attributs "numero" et "performance" et peut modifier le contenu de "salaire" si celui-ci est inférieur à 12500 Euro.

Mécanismes d'octroi et d'annulation des privilèges



Les rôles

- PostgreSQL gère les droits d'accès aux bases de données en utilisant le concept de rôles.
- Un rôle peut être vu soit comme un utilisateur de la base de données, soit comme un groupe d'utilisateurs de la base de données, suivant la façon dont le rôle est configuré.
- Les rôles peuvent posséder des objets de la base de données (par exemple des tables) et peuvent affecter des droits sur ces objets à d'autres rôles pour contrôler qui a accès à ces objets.
- Il est possible de donner l'appartenance d'un rôle à un autre rôle, l'autorisant du coup à utiliser les droits affectés au rôle dont il est membre.

Le concept des rôles comprend les concepts des *utilisateurs* et des *groupes*. Tout rôle peut agir comme un utilisateur, un groupe ou les deux.

Les rôles : définition et utilisation

- `CREATE ROLE nom_utilisateur ;`
- `DROP ROLE nom_utilisateur ;`
- programmes hors connexion au SGBD :

```
createuser nom_utilisateur
dropuser nom_utilisateur
```

- `SELECT rolname FROM pg_roles;`

Les rôles : définition et utilisation

- droit de connexion :

```
CREATE ROLE nom LOGIN;
CREATE USER nom;
```

- droit de création de base de données :

```
CREATE ROLE nom_utilisateur CREATEDB ;
```

- mot de passe :

```
CREATE ROLE nom_utilisateur PASSWORD
            'le_mot_de_passe' ;
```

- modifier un rôle : `ALTER ROLE`

Les rôles : définition et utilisation

```
CREATE ROLE nom [ [ WITH ] option [ ... ] ]
où option peut être :
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | CREATEUSER | NOCREATEUSER
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | CONNECTION LIMIT limite_connexion
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'motdepasse'
    | VALID UNTIL 'dateheure'
    | IN ROLE nomrole [, ...]
    | IN GROUP nomrole [, ...]
    | ROLE nomrole [, ...]
    | ADMIN nomrole [, ...]
    | USER nomrole [, ...]
    | SYSID uid
```

Le contrôle des autorisations

2 commandes SQL :

- `GRANT`
- `REVOKE`

Les privilèges varient d'un SGBD à l'autre mais on retrouve :

- `SELECT`, `INSERT`, `DELETE`
- `UPDATE [OF <liste d'attributs>]`
- `REFERENCES`

Le contrôle des autorisations

Principe : Seul le créateur de l'objet possède tous les privilèges sur cet objet.

Pour accéder à un objet (qu'il n'a pas créé) doit avoir au préalable reçu explicitement les autorisation nécessaires en fonction de règles bien définies.

L'ocroi des privilèges

```
GRANT { { SELECT | INSERT | UPDATE | DELETE
          | RULE | REFERENCES | TRIGGER }
[,...] | ALL [ PRIVILEGES ] }
ON [ TABLE ] nomtable [, ...]
TO { nomutilisateur | GROUP nomgroupe | PUBLIC }
[, ...]
[ WITH GRANT OPTION ]
```

Les privilèges peuvent donc être accordés par héritage (WITH GRANT OPTION).

L'ocroi des privilèges sur les tables

De même on peut utiliser cette commande pour les bases (DATABASE) ou un schéma (SCHEMA).

```
GRANT { { CREATE | CONNECT | TEMPORARY | TEMP }
    [, ...] | ALL [ PRIVILEGES ] }
        ON DATABASE database_name [, ...]
        TO { [ GROUP ] role_name | PUBLIC } [, ...]
        [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ]
```

L'annulation des privilèges

```
REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | DELETE
        | RULE | REFERENCES | TRIGGER }
[,...] | ALL [ PRIVILEGES ] }
ON [ TABLE ] nom_table [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC
[, ...]
[ CASCADE | RESTRICT ]
```

Le mode `RESTRICT` est le mode par défaut de PostgreSQL (`CASCADE` pour ORACLE).

Appartenance à un rôle

- Il est souvent intéressant de grouper les utilisateurs pour faciliter la gestion des droits : de cette façon, les droits peuvent être donnés ou supprimés pour tout un groupe.
- Une fois que ce rôle existe, vous pouvez lui ajouter et lui supprimer des membres en utilisant les commandes GRANT et REVOKE :

```
GRANT role_groupe TO role1, ... ;
REVOKE role_groupe FROM role1, ... ;
```

- Typiquement, un rôle utilisé en tant que groupe n'aura pas l'attribut LOGIN bien que vous puissiez le faire si vous le souhaitez.