



Administration base de données ORACLE



Gestion des utilisateurs

- 
- Tout utilisateur doit posséder **un nom d'utilisateur et un mot de passe pour pouvoir accéder à la base**. C'est ce nom d'utilisateur qui sera le lien avec les droits d'accès qui lui sont accordés

Types d'utilisateurs

Un compte utilisateur = Un schéma de BD

Au niveau le plus élémentaire, un schéma de base de données indique quelles tables ou relations constituent la base de données, ainsi que les champs inclus dans chaque table



Création des utilisateurs

Voici les différentes étapes qui seront nécessaire à la création d'un utilisateur Oracle :

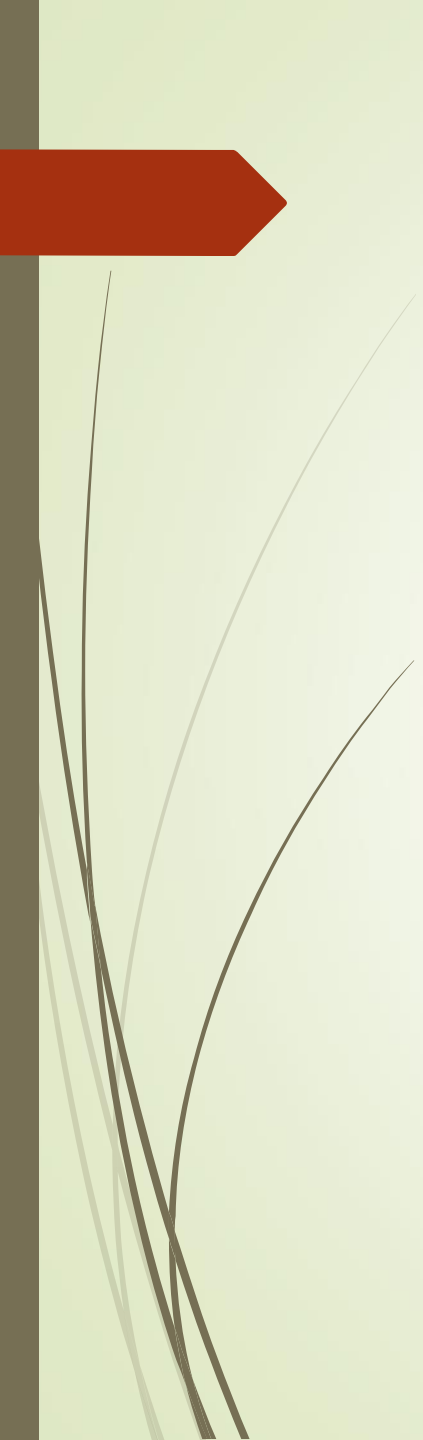
- Choisir un nom d'utilisateur
- Choisir une méthode d'authentification
- Choisir les TABLESPACES que l'utilisateur pourra utiliser
- Définir les quotas sur chaque TABLESPACE
- Définir les TABLESPACES par défaut de l'utilisateur
- Créer l'utilisateur
- Assigner les rôles et privilèges à l'utilisateur



Introduction



- Un **schéma** est une collection (ou un ensemble) nommé d'objets tels que des tables, vues, procédure et packages associés à un utilisateur précis.
- Quand un utilisateur de base de données est crée, son schéma est automatiquement crée. Un utilisateur ne pourra alors être associé qu'à un seul schéma et réciproquement.

- 
- Un utilisateur de base de données va correspondre à un login qui aura reçu certains privilèges.
 - Cet utilisateur sera stocké dans le dictionnaire de données et disposera d'un espace de stockage pour ses objets qui seront alors stockés dans son schéma.

En Oracle on pourra assimiler un utilisateur avec son schéma.



Choix du nom de l'utilisateur

- La première chose à faire pour créer un nouvel utilisateur va être de définir un login.
- Afin d'éviter d'avoir trop de problèmes lors de l'ajout de nouveaux utilisateurs, il est fortement recommandé de mettre une stratégie de nommage en place.
- Par exemple tout les noms d'utilisateur devront être composé des 6 premières lettres de leur nom, d'un "_" et de la première lettre de leur prénom.



Choix du nom de l'utilisateur

Il convient ensuite de connaître les limitations et règles de nommage à respecter:

- Taille maximale 30 caractères.
- Ne devra contenir que des lettres de [a-z] et des chiffres [0-9]. Tous les caractères accentués ou autres sont à éviter.
- Vous pourrez également utiliser les symboles #, \$, _.
- Le login devra commencer par une lettre. Si vous désirez utiliser des logins composé uniquement de chiffres vous devrez alors entourer votre login entre "".



Choisir la méthode d'authentification de l'utilisateur

Afin d'authentifier un utilisateur et de définir les actions que celui-ci sera en mesure d'effectuer sur la base de données, le serveur Oracle doit pouvoir vérifier les accès de l'utilisateur lorsque celui-ci se connecte.

Il existe différents type d'authentification :

- Authentification par la base de données.
- Authentification par le système d'exploitation.

Création d'un utilisateur(authentication par BD)

- Ordre SQL simple pour créer un utilisateur:

```
CREATE USER login IDENTIFIED BY password;
```

- Cette instruction va créer un utilisateur dont le nom est **login**, et mot de passe et **password**.



Exemple

CREATE USER **polysb**

IDENTIFIED BY **polysb**

DEFAULT TABLESPACE USERS

QUOTA 10M ON USERS

TEMPORARY TABLESPACE TEMP



Explication

- Cet ordre crée un utilisateur nommé '**polysb**', et dont le mot de passe est '**polysb**'.
- **polysb** créera par défaut ses objets dans la partition 'users' (jusqu'à concurrence de '10Mo').
- Son espace de travail temporaire (utilisé en interne par Oracle) sera la partition 'temp'.

Création d'un utilisateur(authentication par OS)

- Ordre SQL simple pour créer un utilisateur:

```
CREATE USER login IDENTIFIED externally;
```

Gestion des users: paramètres de création

```
CREATE USER user
IDENTIFIED { BY password | EXTERNALLY }
[ DEFAULT TABLESPACE tablespace ]
[ TEMPORARY TABLESPACE tablespace ]
[ QUOTA { integer [ K | M ] | UNLIMITED } ON tablespace ] ...
[ PASSWORD EXPIRE ]
[ ACCOUNT { LOCK | UNLOCK } ]
```

Modification d'un utilisateur

- Pour changer le mot de passe d'un utilisateur:

ALTER USER < login de l'utilisateur > IDENTIFIED **BY** < nouveau mot de passe >

- Pour modifier le statut d'un utilisateur:

-- Verrouillage du compte

ALTER USER scott **ACCOUNT LOCK**;

-- Activation du compte

ALTER USER scott **ACCOUNT UNLOCK**;

- Modifier les quota

➤ **ALTER** USER scott

Quota 15 M on tablespace1

Quota 5 M on tablespace2;

Suppression d'un utilisateur

- Pour supprimer un utilisateur:

DROP USER login [**CASCADE**];

Si l'utilisateur a des objets dans son tablespace comme des tables, des view,...

On ne peut pas supprimer son compte grâce à drop user login

Il faut le supprimer en utilisant : **drop user login cascade**

- Option **CASCADE** : supprime aussi les objets
 - Impossible de supprimer un utilisateur connecté



Gestion des users: informations sur les users

- Descriptif de la vue **DBA_USERS**
- Descriptif de la vue **DBA_TS_QUOTAS**



Les Privilèges

Privilège

Nous venons de voir comment créer un compte utilisateur pour **polysb**. Pour pouvoir créer un compte utilisateur, il faut disposer du privilège **CREATE USER**.

Il est préférable que cette commande ne soit accessible qu'au DBA.

En ce qui concerne notre utilisateur **polysb**, malgré la création de son compte, il ne peut toujours pas se connecter à la base de données. **Il faut pour cela que le DBA lui accorde le privilège CREATE SESSION**

Privilège

Un privilège est le **droit d'exécuter un type d'instruction SQL** spécifique. Quelques exemples de privilèges :

- **le droit de se connecter à une base de données (autrement dit ouvrir une session),**
- **le droit de créer une table,**
- **le droit de sélectionner des lignes dans une table.**

Les privilèges d'une base de données Oracle peuvent être répartis en deux catégories distinctes :

- **les privilèges système,**
- **les privilèges objets.**



Assigner des privilèges système à un utilisateur

- Lorsqu'un utilisateur est créé avec l'instruction CREATE USER, il ne dispose encore d'aucun droit car aucun privilège ne lui a encore été assigné.
- Il faut donc lui assigner les privilèges nécessaires .
- Il doit pouvoir se connecter, créer des tables, des vues, des séquences.

Les privilèges système

Catégorie	Exemples
TABLE	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE SELECT ANY TABLE UPDATE ANY TABLE DELETE ANY TABLE
SESSION	CREATE SESSION ALTER SESSION RESTRICTED SESSION
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX
USER	CREATE USER ALTER USER DROP USER

- Le mot-clé **ANY** signifie que les utilisateurs disposent du privilège de gestion d'objets dans n'importe quel schéma.

Assigner des privilèges système à un utilisateur

Pour lui assigner ces privilèges de niveau système il faut utiliser l'instruction **GRANT** dont voici la syntaxe:

```
GRANT [système_privilege | | rôle | | ALL PRIVILEGES ] to  
[user | | PUBLIC | | rôle ]
```

- **système_privilege** représente un privilège système
- **rôle** représente un rôle préalablement créé
- **ALL PRIVILEGES** représente tous les privilèges
- **user** représente le nom de l'utilisateur qui doit bénéficier du privilège
- **PUBLIC** assigne le privilège à tous les utilisateurs
- **WITH ADMIN OPTION** assigne à l'utilisateur le droit d'assigner, de retirer, de modifier et de supprimer à son tour les privilèges du rôle reçus

Assigner des privilèges système à un utilisateur

Pour que l'utilisateur puisse simplement se connecter à la base, il doit bénéficier du privilège système **CREATE SESSION**, Ensuite il faut lui assigner des droits de création de table, puis les droits de création de vues,

```
GRANT CREATE SESSION ,  
CREATE TABLE ,  
CREATE VIEW TO nom_utilisateur ;
```


Assigner des privilèges objet à un utilisateur

- On utilise aussi la commande GRANT:

GRANT [**object_privilege** | | **ALL PRIVILEGES**] (**column**)
ON **schema** .**Object** **TO** [**user** | | **role** | | **PUBLIC**]

- **object_privilege** représente un privilège objet
- **role** représente un rôle préalablement créé
- **ALL PRIVILEGES** représente tous les privilèges assignés à l'exécuteur de l'instruction
- **column** représente le nom de colonne d'une table
- **schema** représente le nom d'un schéma
- **object** représente le nom d'un objet du schéma
- **WITH GRANT OPTION** assigne à l'utilisateur de droit d'assigner à son tour le privilège reçu à un autre utilisateur
 - (WITH GRANT OPTION s'applique à un utilisateur ou à PUBLIC, mais pas à un rôle)

Assigner des privilèges objet à un utilisateur

- attribution de privilège par le **propriétaire** de l'objet (toto) :
 - `GRANT SELECT, INSERT, UPDATE (nom, prenom) ON Client TO tata`
 - L'utilisateur peut modifier la table `Client` mais uniquement les colonnes `nom` et `prenom`.
- attribution de privilège par un utilisateur , **non propriétaire** de l'objet ayant reçu le privilège avec l'option **WITH GRANT OPTION** de l'objet :
 - `GRANT SELECT ON toto.client TO tata WITH GRANT OPTION;`



Assigner des privilèges objet à un utilisateur

Attention:

- Pour pouvoir mettre à jour ou supprimer des lignes d'une table, les privilèges **UPDATE** et **DELETE** ne suffisent pas. Le privilège **SELECT** est nécessaire.
- Un utilisateur munis des droits DBA ne pourra pas accorder de privilèges sur un objet qui ne lui appartient pas



Assigner des privilèges objet à un utilisateur

➤ Principes généraux appliqués aux privilèges

- Un utilisateur possède automatiquement tous les privilèges sur un objet qui lui appartient
- Un utilisateur ne peut pas donner plus de privilèges qu'il n'en a reçu
- s'il n'a pas reçu le privilège avec l'option **WITH GRANT OPTION**, un utilisateur ne peut pas assigner à son tour ce même privilège

Les privilèges objet

- DELETE
- EXECUTE
- INSERT
- SELECT
- UPDATE
- REFERENCES (création d'une contrainte d'intégrité: clé étrangère)

L'instruction **GRANT** permet d'assigner un ou plusieurs privilèges système ou objet. Cependant, lorsque la liste des privilèges est importante, cette manière de procéder s'avère rapidement fastidieuse et répétitive



Les rôles



Les Rôles

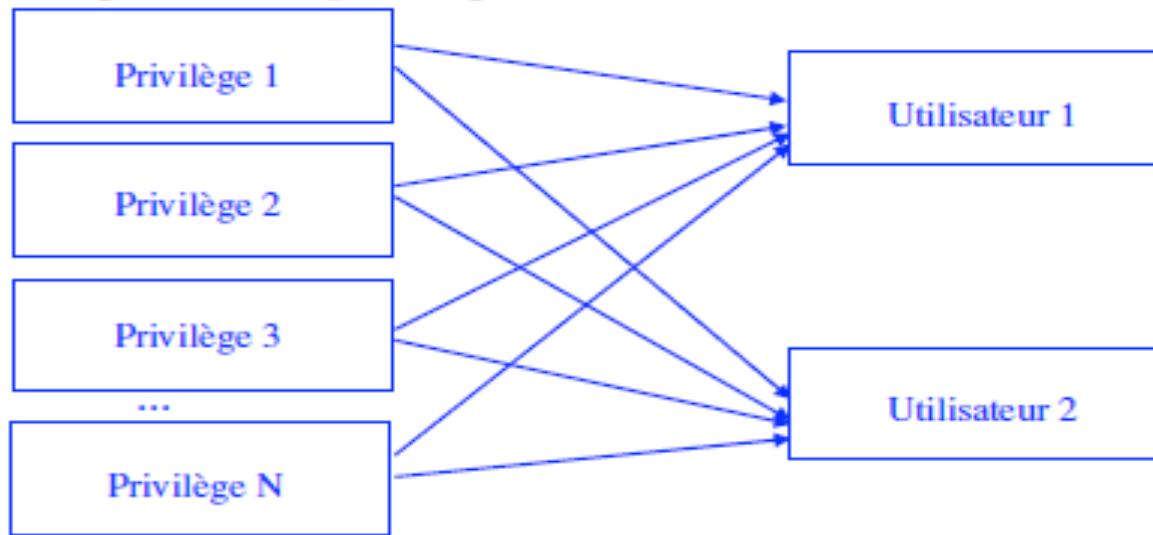
- Regroupement de privilèges pour des familles d'utilisateurs
- Facilitent la gestion des privilèges objet en évitant les ordres GRANT
- Un rôle par défaut est donné à un utilisateur
- Un utilisateur peut posséder plusieurs rôles mais n'est connecté qu'avec un seul à la fois
- Un rôle facilite la gestion des privilèges
- pour des raisons de sécurité, un **mot de passe peut être** assigné à un rôle



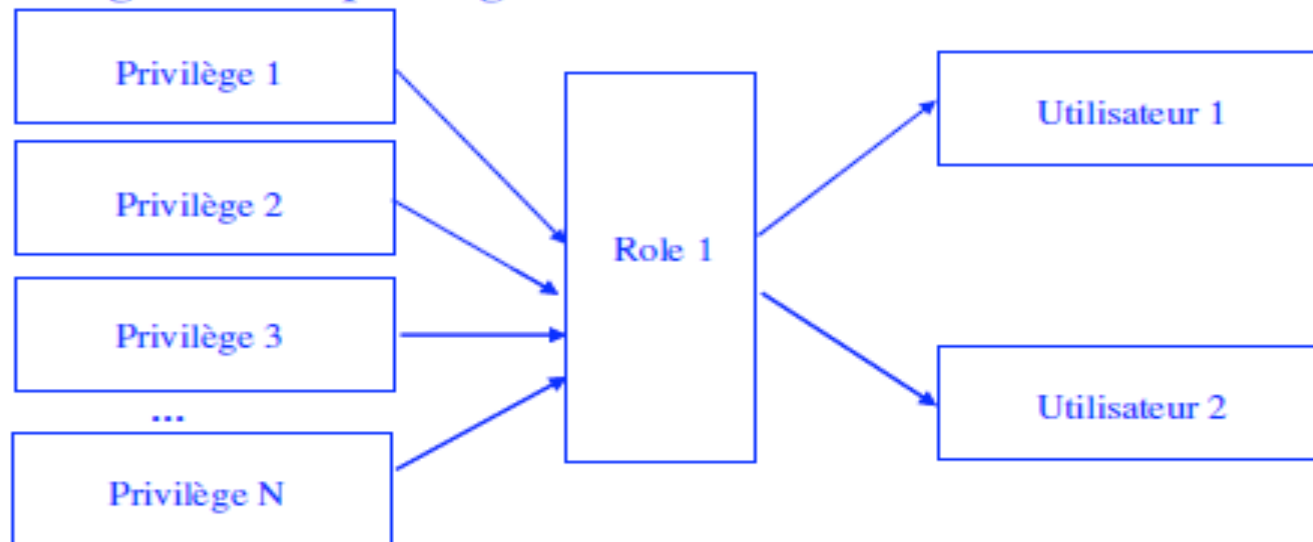
Introduction

- Les Rôles et privilèges sont définis pour **sécuriser** l'accès aux données de la base .
- Ces concepts sont mis en œuvre pour protéger les données en **accordant** (ou **retirant**) des privilèges à un utilisateur ou un groupe d'utilisateurs
- **Un rôle** est un **regroupement de privilèges**. Une fois créé il peut être assigné à **un utilisateur** ou à un **autre rôle**

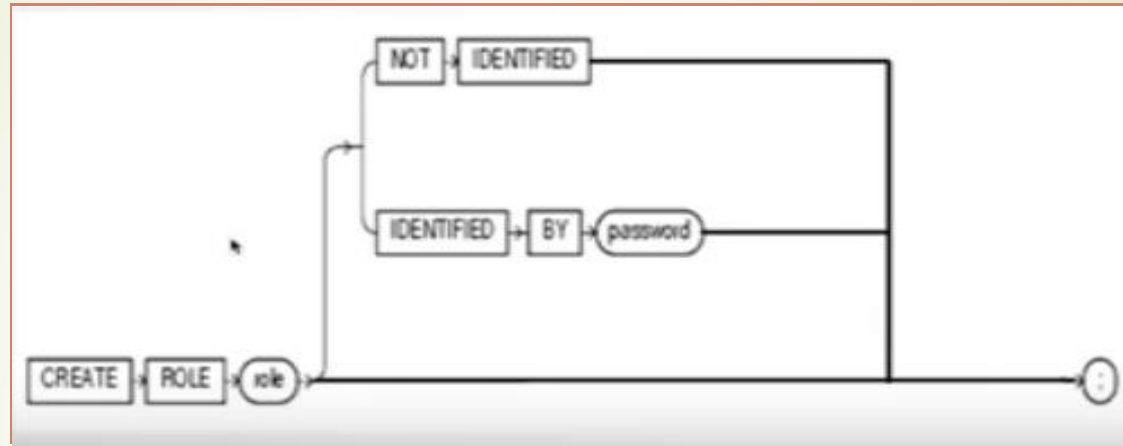
Assignation de privilèges aux utilisateurs : SANS ROLES



Assignation de privilèges aux utilisateurs : VIA UN ROLE



Création des rôles



- Se crée avec l'instruction **CREATE ROLE**

CREATE ROLE nomRôle
[IDENTIFIED {By mot_de_passe} | NOT IDENTIFIED]

- **nomRôle** représente le nom du rôle
- **NOT IDENTIFIED** (défaut) indique qu'aucun mot de passe n'est nécessaire pour activer le rôle
- **IDENTIFIED BY mot_de_passe** indique qu'un mot de passe est nécessaire pour activer le rôle

Manipulation des rôles :

Exemples

- non identifié :
CREATE ROLE rr;
- identifié par mot de passe
CREATE ROLE rr **IDENTIFIED BY** pswdrr;
- Pour créer un rôle il faut avoir le privilège : **CREATE ROLE**

Manipulation des rôles : Exemples

Lorsque le rôle est créé, il ne contient rien et il faut l'alimenter à l'aide d'instructions **GRANT**

```
CREATE ROLE accounting;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON CPT.FACTURE TO accounting;  
GRANT SELECT, INSERT, UPDATE, DELETE ON CPT.LIG_FAC TO accounting;  
GRANT SELECT, INSERT, UPDATE, DELETE ON CPT.JOURNAL TO accounting;
```

Une fois le rôle créé, il peut être assigné à un utilisateur ou à un autre rôle

```
GRANT accounting TO nom_utilisateur ;
```

Les rôles

- Trois rôles existent en standard:

- **CONNECT**
- **RESOURCE**
- **DBA**

➤ **Pour voir** les privilèges système assignés au rôle:

```
select * from DBA_SYS_PRIVS where grantee='CONNECT';  
select * from DBA_SYS_PRIVS where grantee='RESOURCE';  
select * from DBA_SYS_PRIVS where grantee='DBA' order by PRIVILEGE;  
select * from DBA_ROLES ;
```

Les rôles

- La liste des rôles assignés à l'utilisateur au cours de sa session est visible via la vue **SESSION_ROLES**:

```
select * from SESSION_ROLES;
```

- La liste des privilèges assignés à l'utilisateur au cours de sa session est visible via la vue **SESSION_PRIVS**:

```
select * from SESSION_PRIVS;
```

Modifier un rôle

- Syntaxe :
ALTER ROLE nom
[**IDENTIFIED** {**By** mot_de_passe} | **NOT IDENTIFIED**]
- La seule chose qu'il est possible de modifier dans un rôle est son système de protection par mots de passe
- vous devez disposer du rôle approprié avec l'option **ADMIN** ou du privilège système **ALTER ANY ROLE**



Les rôles: suppression

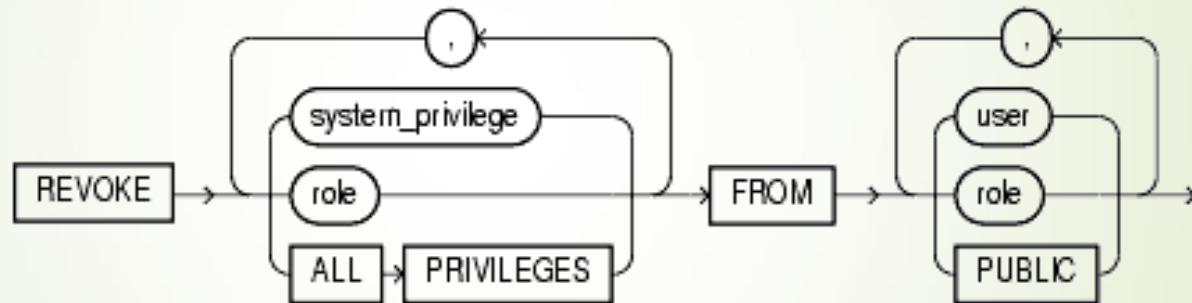
- Un rôle peut être supprimé en utilisant l'instruction **DROP ROLE**
- `DROP ROLE nom_role ;`
- Le rôle spécifié ainsi que tous les privilèges qui lui sont associés sont supprimés de la base et également retiré à tous les utilisateurs qui en bénéficiaient



Retirer des privilèges

Retirer des privilèges système

- Les privilèges système qui ont été assignés à des utilisateurs ou à des rôles peuvent être retirés avec l'instruction **REVOKE**:



```
REVOKE [ system_privilege || rôle || ALL PRIVILEGES ] FROM  
[ user || PUBLIC || rôle ].
```

- Retirer des privilèges à un utilisateur ne supprime pas son schéma ni les objets qu'il contient

Retirer des privilèges objet

- Les privilèges objet qui ont été assignés à des utilisateurs ou à des rôles peuvent être retirés avec l'instruction **REVOKE**:

```
REVOKE [object_privilege | | ALL PRIVILEGES] (column ) ON  
schema . Object FROM[user | | role | | PUBLIC]
```



Les types de commandes SQL

DML (DATA MANIPULATION LANGUAGE)

- SELECT
- INSERT
- UPDATE
- DELETE
- MERGE

DDL (DATA DEFINITION LANGUAGE)

- CREATE
- ALTER
- DROP
- RENAME
- TRUNCATE
- COMMENT

TCL (TRANSACTION CONTROL LANGUAGE)

- COMMIT
- ROLLBACK
- SAVEPOINT

DCL (DATA CONTROL LANGUAGE)

- GRANT
- REVOKE

Exercice

