Sécurité des Bases de Données (M2 INIS)

Contrôle d'accès

Benjamin Nguyen

Objectif : Ce TP est une introduction aux méthodes de base du contrôle d'accès pour les BD relationnelles, sur Oracle XE. Dans une première partie nous abordons la création d'utilisateurs, de privilèges, de rôles, et de profiles, et de droits contextuels.

Matériel nécessaire: le répertoire en ligne pour la séance est:
http://www.benjamin-nguyen.fr/SBD/

Différents éléments de syntaxe vous sont donnés dans les extraits de la documentation dans le répertoire en ligne du cours. Vous consulterez aussi la doc. en ligne:
http://docs.oracle.com/database/121/index.htm

Pour lancer la version en ligne : ouvrez une fenêtre de commande puis tapez sqlplus
Vous entrerez ensuite le login / mdp de votre utilisateur admin ou système. Si vous perdez le mot de passe vous pouvez le réinitialiser en faisant :

```
sqlplus / as sysdba
alter user <username> identified by <password>;
```

Utilisateurs, privilèges, rôles, profiles.

Q1 : **Création des utilisateurs, affectation des droits minimaux, vérification des droits**

1.1 Connectez vous avec un utilisateur ayant des droits administrateur sur la BD (ex: utilisateur «system»), et créez un utilisateurs « user_test ». Pour cet utilisateur, vous limiterez le quota de création de structures de données à 500Ko.

La syntaxe de création d'utilisateurs est :

CREATE USER <username> IDENTIFIED BY <password> [options];
avec [options]: DEFAULT TABLESPACE <tablespace> -- espaces de travail par défaut (donnez le nom **users**)
TEMPORARY TABLESPACE <tablespace> (donnez le nom **temp**)
QUOTA int {K | M} ON <tablespace> -- quotas sur les TABLESPACES
PROFILE <profile_name>
PASSWORD EXPIRE
ACCOUNT {LOCK|UNLOCK}

1.2 Quels Tablespaces ont été attribués par défaut à l'utilisateur ? Vérifiez que le quota établis à la création du compte est correct. Vous devrez consulter les tables systèmes suivantes : **dba_tablespacesg, dba_users** et **dba_ts_quotas**.

1.3 Essayez de vous connecter en tant que « user_test » pour vérifier que ça fonctionne.

1.4 Donnez à « user_test » les privilèges systèmes nécessaires pour se connecter et pour pouvoir créer des objets de type tables, vue, et procédures stockées.

1.5 Consulter les tables systèmes adéquates (**dba_sys_privs, dba_tab_privs, dba_role_privs**) pour vérifier les droits attribués à « user_test ». Vous vérifierez au moins 1) les droits systèmes, 2) les droits sur les objets, et 3) les rôles qui sont attribués à l'utilisateur.

Q2 : **Droits implicites et explicites sur des objets**

2.1 Créez un deuxième utilisateur « user2_test » et donnez lui les mêmes droits qu'à « user_test ». L'utilisateur « user_test » crée une table nommée TEST_PUBLIC contenant 1 seule colonne DATA de type VARCHAR(100). Peut-il lui-même insérer dans cette table ?

2.2 L'utilisateur « user2_test », s'il connait le nom de cette table et le nom de l'utilisateur qui l'a créée (schéma), peut-il voir que cette table existe? Peut-il accéder à cette table ? Même question pour un administrateur (utilisateur « system »).

2.3 Faire en sorte que l'utilisateur « user_test » autorise « user2_test » à lire (SELECT) et écrire (INSERT, UPDATE) dans la table TEST_PUBLIC. L'utilisateur « user2_test » devra pouvoir accéder à la table TEST_PUBLIC en utilisant un nom d'objet utilisé comme "synonyme" pour cette table (instruction `CREATE PUBLIC SYNONYM`).

Q3 : **Droits sur des procédures stockées**

3.1 L'utilisateur « user_test » crée une procédure TEST_PROC qui insert un tuple passé en paramètre dans la table TEST_PUBLIC. Pour créer la procédure, vous pourrez vous aider de la syntaxe suivante (voir la documentation PL/SQL fournie ou en ligne pour les détails):

```
CREATE [OR REPLACE] PROCEDURE procedure_name [ (parameter [,parameter]) ]
IS
      [declaration_section]
BEGIN
      executable_section
[EXCEPTION exception_section]
END [procedure_name];
```

3.2 L'utilisateur « user_test » autorise « user2_test » à exécuter cette procédure. Vérifier que « user2_test » puisse insérer dans la table TEST_PUBLIC en utilisant cette procédure.

3.3 L'utilisateur « user_test » retire maintenant à « user2_test » tous ses droits sur la table TEST_PUBLIC. L'utilisateur «user2_test » peut-il toujours insérer dans la table avec la procédure ? Qu'en concluez vous sur les "droits" par défaut d'une procédure stockée ?

3.4 On peut redéfinir la procédure pour qu'elle ait les mêmes droits que celui qui l'invoque en précisant `AUTHID CURRENT_USER` avant le `IS`.

Q4 : **Administration des droits: notion de rôle et de profile utilisateur**

La notion de rôle permet à l'administrateur d'accorder plus simplement des privilèges aux utilisateurs. Une fois les privilèges associés à un rôle, chaque fois qu'un nouveau compte utilisateur sera créé, il suffira de l'associer à son (ou ses) rôle(s) pour lui accorder les droits.

4.1 Retirer les privilèges "système" accordés à « user_test » et « user2_test ». Vous pourrez lancer la requête suivante depuis le compte administrateur pour vous aider à identifier les requêtes à exécuter:

```
SELECT 'REVOKE ' || PRIVILEGE || ' FROM ' || GRANTEE || ';' from dba_sys_privs where grantee LIKE 'USER%';
```

4.2 Créez un rôle «VENDOR_ROLE » et affecter à ce rôle les privilèges "système" initialement accordés aux deux utilisateurs «user_test» et «user2_test». Associez ce rôle à « user_test » et « user2_test », puis vérifiez leurs privilèges.

4.3 La gestion du mot-de-passe est implantée par les « profils » utilisateurs. Comme pour les rôles, les profils sont définis et puis ensuite assignés aux utilisateurs. Supposons que « user_test » essaie d'attaquer le compte « user2_test » en utilisant une méthode exhaustive (essaie de tous les mots de passes) pour s'y connecter. Créez un profil «MDP_USERS» (`CREATE PROFILE`) qui protège les comptes des utilisateurs contre ce type d'attaque en limitant à trois le nombre de tentatives de connexion échouées. Testez-le sur un le compte « user2_test ». Vous pouvez vérifier la bonne création du profile en utilisant la table **dba_profiles**.

4.4 Connectez-vous en tant qu'administrateur (utilisateur « system »), puis lancez le script VENDOR.sql. Des tables sont créées et peuplées, représentant des vendeurs, les commandes qu'ils ont passées pour les clients, les détails de ces commandes, les produits commandés et leurs fournisseurs.

4.5 Créer un rôle VENDOR_ROLE, et un profile VENDOR_PROF limitant le nombre de tentatives de connexion de chaque vendeur.

4.6 (**Pour ceux qui connaissent PL/SQL**) Créer un script PL/SQL qui crée un compte pour chaque vendeur tel que: les login / mot de passe du vendeur soient son nom (colonne NOMVEN) et son mot de passe (colonne PWD), son rôle soit VENDOR_ROLE, et son profile soit VENDOR_PROF. Vérifier que chaque compte ait été crée avec le rôle voulu en interrogeant la base.

4.7 Permettre à chaque vendeur de lire et d'écrire les tables VEN, COM, DET (sans donner à chaque vendeur individuellement).

Q5 : **Autorisations basées sur le contenu.**

Modifier le rôle VENDOR_ROLE pour ne permettre à chaque vendeur que :
- de visualiser uniquement ses propres commandes/détails,
- d'insérer et de mettre à jour uniquement les commandes/détails le concernant,
- de visualiser la ligne de la table VEN le concernant, et de ne modifier que le Pays et le Pwd de cette ligne.

Vous serez amené à utiliser la table **sys_context**.

Q6 : **Autorisations contextuelles.**
On souhaite que les vendeurs ne puissent pas passer de commande après 18h. Mettez en place ce contrôle et tester le.