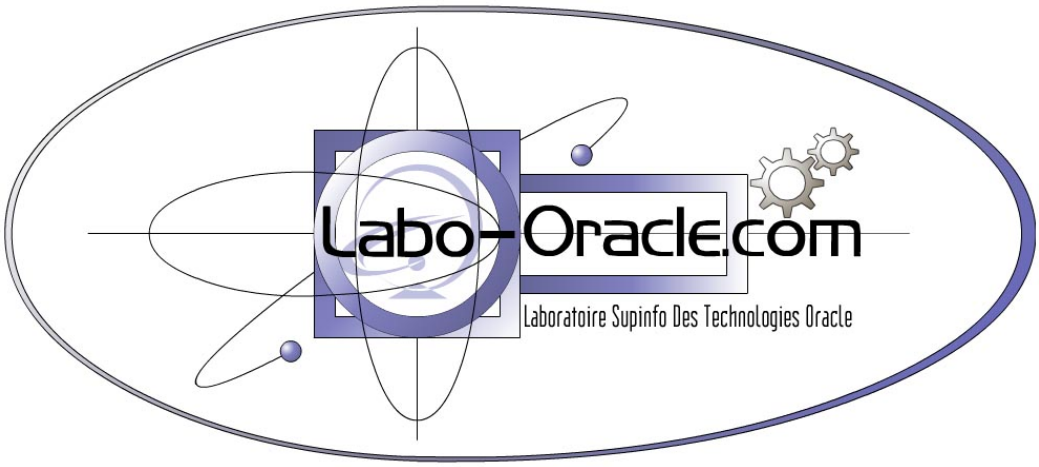

Module n°5

GESTION DES UTILISATEURS

1Z0-001

Auteur : Arnaud Bontemps
Version 1.3 – 7 août 2003
Nombre de pages : 29

Ecole Supérieure d'Informatique de Paris
23. rue Château Landon 75010 – PARIS
www.supinfo.com

Table des matières

1. Administration des utilisateurs

1.1. Création d'un utilisateur

1.1.1. Utilisateurs; Sécurité du domaine

La sécurité est une part intégrante de l'administration de la base de données. En tant qu'administrateur, vous serez responsable de la sécurité de la base de données. Vous pourrez effectuer ceci en définissant la sécurité du domaine pour chaque utilisateur. Chaque composant de la sécurité du domaine possède des fonctions distinctes.

- Le premier composant est le mécanisme d'authentification qui définit les différents privilèges d'un utilisateur. Un utilisateur pourra être authentifié par le serveur Oracle ou bien par le système d'exploitation.
  Le mécanisme d'authentification est spécifié lors de la création de l'utilisateur et pourra être modifié par la suite.
- Le second composant est le verrouillage des comptes. Les comptes utilisateurs peuvent être verrouillés afin d'empêcher un ou plusieurs utilisateurs de se connecter à la base de données. Ce verrouillage pourra se faire de manière automatique après un certain nombre d'échecs lors de l'authentification ou bien l'administrateur pourra verrouiller ou déverrouiller le compte manuellement.
- Le troisième composant est le tablespace par défaut. Il spécifie dans quel tablespace les objets de l'utilisateur seront stockés si celui-ci ne spécifie pas de manière explicite un tablespace lors de la création de nouveaux objets. Si aucun tablespace par défaut n'est spécifié lors de la création de l'utilisateur, Oracle lui assignera automatiquement le tablespace SYSTEM.
- Le quatrième composant est le tablespace temporaire. Celui-ci est défini par un nombre d'extents qui seront alloués par le serveur afin d'écrire des données temporaires comme des données de tri. Si ce tablespace n'est pas défini lors de la création de l'utilisateur, Oracle lui assignera automatiquement le tablespace SYSTEM.
- Le cinquième composant est les quotas pour les tablespaces. Ces quotas permettront de gérer la quantité d'espace disque allouée à un utilisateur dans les tablespaces.
- Le sixième composant correspond aux limitations de ressources, qui permettent de contrôler l'utilisation des ressources système. Ces ressources incluent le temps CPU, le nombre de connexions pour un utilisateur ainsi que le nombre d'entrées-sorties sur le disque.
- Le septième composant est représenté par les privilèges système et les privilèges objets qui permettent de contrôler les accès aux données pour chaque utilisateur. Ces privilèges permettent de définir les différentes actions que les utilisateurs pourront effectuer sur les données.
- Le huitième et dernier composant est représenté par les rôles qui permettent de contrôler les actions d'un utilisateur comme faisant parti d'un ensemble d'utilisateur. Les rôles peuvent être assimilés aux groupes d'utilisateurs sous *Windows NT*.

1.1.2. Création d'un utilisateur : Checklist

Vous pouvez contrôler l'accès à une base de données en créant, modifiant, supprimant et en surveillant des utilisateurs. Nous allons voir les différentes étapes de la création d'un utilisateur.

La première étape consiste à choisir un nom d'utilisateur et un mode d'authentification. Ces informations sont nécessaires afin de créer le schéma du nouvel utilisateur.
Lorsqu'un utilisateur est créé, un schéma portant le même nom est lui aussi créé pour l'utilisateur. Un schéma est un espace qui va contenir tous les objets de l'utilisateur comme des tables, des vues, des triggers, des procédures, des index. Un utilisateur ne peut être associé qu'à un seul schéma (portant le même nom). Quand un utilisateur se connectera, il aura alors accès à tous les objets du schéma correspondant.

La seconde étape consiste à choisir les tablespaces qui serviront à stocker les objets de son schéma. Il faudra ensuite choisir le quota d'espace disponible pour cet utilisateur sur ses différents tablespaces. Par défaut, le quota est de zéro octets pour chaque tablespace. Le quota correspond à l'espace disponible maximum pour les objets d'un utilisateur sur un tablespace. Il faut ensuite assigner un tablespace par défaut et un tablespace temporaire à ce nouvel utilisateur.

La troisième étape consiste à exécuter la commande CREATE USER qui va permettre de créer de manière proprement dite l'utilisateur. Il faudra ensuite lui assigner des privilèges ainsi que des rôles en fonction des différentes actions que pourra effectuer ce nouvel utilisateur.

La dernière étape consiste à former ce nouvel utilisateur pour qu'il puisse se connecter à la base de données et lui expliquer comment changer son mot de passe.

1.1.3. Création d'un utilisateur : Serveur d'authentification

Il existe deux situations distinctes pour créer un utilisateur :

- Vous créez un utilisateur quand cet utilisateur doit se connecter à une base de données et accéder à ses objets.
- Vous créez un utilisateur quand un utilisateur existant veut se connecter à une autre base de données. Dans ce cas, vous pourrez alors avoir deux ou trois comptes pour un même utilisateur.

Voici la syntaxe de la commande pour créer un utilisateur.

```
CREATE USER user IDENTIFIED {BY password | EXTERNALLY}
[DEFAULT TABLESPACE default_tablspace_name]
[TEMPORARY TABLESPACE temporary_tablespace_name]
[QUOTA {integer [K | M] | UNLIMITED } ON tablespace_name ]
[PASSWORD EXPIRED]
[ACCOUNT {LOCK | UNLOCK}]
[PROFILE {profile_name | DEFAULT}];
```

1.1.4. Création d'un utilisateur : Authentification Os

La méthode d'authentification est un choix obligatoire lors du processus de création d'un utilisateur. Dans cette partie, nous allons voir comment doit faire un DBA pour être sur que l'utilisateur sera authentifié par le système d'exploitation.

Quand vous utilisez l'authentification du système d'exploitation, le serveur Oracle conserve le compte utilisateur bien que ce soit le système d'exploitation qui authentifie l'utilisateur.

- La première étape consiste à utiliser les mots clés IDENTIFIED EXTERNALLY dans la commande CREATE USER et à initialiser le paramètre OS_AUTHENT_PREFIX qui est utilisé pour spécifier le format des noms d'utilisateur. Le serveur rajoutera le OS_AUTHENT_PREFIX au nom d'utilisateur du système d'exploitation. Il comparera ensuite cet utilisateur avec un utilisateur de la base de données. (La valeur par défaut du paramètre OS_AUTHENT_PREFIX est OPS$.)

- Il vous faudra aussi initialiser le paramètre REMOTE_OS_AUTHENT qui permet de définir si l'utilisateur est authentifié par le serveur Oracle (REMOTE_OS_AUTHENT=FALSE) ou par le système d'exploitation (REMOTE_OS_AUTHENT=TRUE).

1.1.5. Création d'un utilisateur : Conseils

- Il est préférable de choisir un mot de passe standard que l'utilisateur sera plus à même de se souvenir et de changer par la suite.

- Il est préférable d'utiliser l'authentification du système d'exploitation (mais ce choix reste discutable).
- Il peut être intéressant d'utiliser l'option EXPIRE lors de la création de l'utilisateur ce qui forcera l'utilisateur à changer son mot de passe lors de sa première connexion.
- Il est nécessaire de spécifier les tablespaces temporaire et par défaut pour tous les utilisateurs afin d'éviter que le tablespace SYSTEM soit mi par défaut et ne se fragmente trop rapidement.
- Il peut être judicieux d'utiliser des quotas car des quotas illimités peuvent créer des problèmes d'espace. De plus, les utilisateurs ne nécessitent aucuns quotas sur les tablespaces temporaires ou sur les tablespaces de rollback.

1.2.Gestion des utilisateurs

1.2.1. Gestion des mots de passe

Lors de la création d'un utilisateur, le DBA assigne un mot de passe à l'utilisateur. Ce mot de passe peut être modifié par l'utilisateur ou bien par le DBA. Le DBA peut quant à lui forcer l'expiration du mot de passe.

Voici la syntaxe SQL pour changer de mot de passe:

```
ALTER USER user_name [IDENTIFIED { BY password | EXTERNALLY }]
[PASSWORD EXPIRE];
```

Le changement de mot de passe ne sera effectif que pour les sessions suivantes et non pas pour celles en cours.

1.2.2. Compte bloqué

Il peut parfois être nécessaire de bloquer temporairement un compte utilisateur. Pour rendre indisponible un compte, le DBA va bloquer ce compte.

Voici la syntaxe pour bloquer ou non un compte utilisateur :

```
ALTER USER user_name
[ACCOUNT { LOCK | UNLOCK}];
```

1.2.3. Modification du quota d'un utilisateur

Voici la syntaxe SQL pour modifier le quota d'un utilisateur sur un tablespace :

```
ALTER USER user_name
[DEFAULT TABLESPACE default_tablespace_name]
[TEMPORARY TABLESPACE temporary_tablespace_name]
[QUOTA { integer [K | M] | UNLIMITED} ON tablespace_name];
```

1.2.4. Suppression d'un utilisateur

Il faut, pour pouvoir supprimer un utilisateur, que celui soit déconnecté.

Voici la syntaxe SQL pour supprimer un utilisateur :

```
DROP USER user_name [ CASCADE ];
```

Si l'option CASCADE est utilisée alors tous les objets de l'utilisateur seront d'abord supprimés avant de supprimer l'utilisateur. Si l'option CASCADE n'est pas utilisée alors que le schéma de l'utilisateur contient encore des objets, une erreur se produira lors de la suppression de l'utilisateur.

1.3.Monitoring des utilisateurs

1.3.1. Affichage des quotas utilisateurs

C'est grâce à la vue DBA_TS_QUOTAS que l'on pourra trouver toutes les informations sur les quotas des différents utilisateurs.

1.3.2. Affichage du statut des comptes utilisateurs

Vous pourrez avoir toutes les informations concernant un utilisateur grâce à la vue DBA_USERS.

Cette vue va vous permettre d'obtenir des informations concernant les utilisateurs. Voici quelques informations intéressantes :

- USERNAME : nom de l'utilisateur.
- USER_ID : Id de l'utilisateur.
- PASSWORD : Mots de passe de l'utilisateur (encrypté pour des raisons de sécurité).
- CREATED : Date de création de l'utilisateur.
- ACCOUNT_STATUS : Statut du compte (actif ou autre….).
- LOCK_DATE : Date à laquelle le compte a été bloqué.
- EXPIRY_DATE : Date d'expiration du compte.
- DEFAULT_TABLESPACE : Tablespace par défaut du compte.
- PROFILE : Profil du compte.
- EXTERNAL_NAME : Nom externe de l'utilisateur.

2. Administration des ressources et de la sécurité des mots de passe

2.1. Administrations des profils

2.1.1. Utilisation des profils

Un profil est un ensemble nommé de limitations sur les ressources systèmes, pouvant être protégé d'un mot de passe, et servant à optimiser les ressources et augmenter la sécurité de la base de données.

Voici les différentes ressources contrôlées par un profil :

CPU Time	Concurrent Sessions
I/O Opération	Password aging and expiration
Idle Time	Password History
Connect Time	Password Complexity Verification
Memory Space	Account Locking

Le DBA devra assigner un profil à chaque utilisateur. Si aucun profil n'est défini par défaut, l'utilisateur se verra assigner le profil par défaut DEFAULT, qui est automatiquement crée lors de la création de la base. Le profil par défaut ne comporte initialement aucunes limitations mais pourra ensuite être modifié par le DBA.

Les profils peuvent être d'un grand intérêt lorsque l'on souhaite restreindre les opérations qui vont nécessiter trop de ressources système.
Les profils permettent aussi de vérifier que l'utilisateur sera bien déconnecté de la base si sa session reste inactive un certain moment.
Ils peuvent aussi permettrent de regrouper les diverses utilisations de ressources systèmes et permettent finalement de gérer les mots de passe.

2.1.2. Création de profils

Pour créer un profil, vous devrez tout d'abord identifier les ressources qui devront être limitées par chaque profil. Vous devrez aussi savoir quelles seront ces limites. Pour vous aidez dans votre choix, vous pourrez utiliser les données historiques de la base ou bien les fichiers de trace utilisateur ou finalement en auditant la base de données afin d'avoir des données plus précises.

Voici la syntaxe qui va vous permettre de crée un profil :

```
CREATE PROFILE profile LIMIT
[SESSIONS_PER_USER {integer | UNLIMITED | DEFAULT}]
[CPU_PER_SESSION {integer | UNLIMITED | DEFAULT}]
[CPU_PER_CALL {integer | UNLIMITED | DEFAULT}]
[CONNECT_TIME {integer | UNLIMITED | DEFAULT}]
[IDLE_TIME {integer | UNLIMITED | DEFAULT}]
[LOGICAL_READS_PER_SESSION {integer | UNLIMITED | DEFAULT}]
[LOGICAL_READS_PER_CALL {integer | UNLIMITED | DEFAULT}]
[COMPOSITE_LIMIT {integer | UNLIMITED | DEFAULT}]
[PRIVATE_SGA {integer | UNLIMITED | DEFAULT}]
[FAILED_LOGIN_ATTEMPTS {integer | UNLIMITED | DEFAULT}]
[PASSWORD_LIFE_TIME {integer | UNLIMITED | DEFAULT}]
[PASSWORD_REUSE_TIME | PASSWORD_REUSE_MAX
     {integer | UNLIMITED | DEFAULT}]
[PASSWORD_LOCK_TIME {integer | UNLIMITED | DEFAULT}]
[PASSWORD_GRACE_TIME {integer | UNLIMITED | DEFAULT}]
```

```
  [PASSWORD_VERIFY_FUNCTION {function | NULL | DEFAULT}];
```

Lorsque vous créez un profil vous n'êtes pas obligé de spécifier tous les paramètres. Tous les paramètres qui n'auront pas été spécifiés se verront automatiquement assigner la valeur par défaut.

2.1.3. Assignation d'un profil à un utilisateur

Les profils ne peuvent être assignés qu'à des utilisateurs. Et chaque utilisateur ne peut avoir qu'un seul profil. Les utilisateurs peuvent, soit avoir le profil DEFAULT, soit celui assigné par le DBA. L'assignation d'un profil ne mettra pas en place les limitations de ressources sur la session courante d'un utilisateur.

Pour assigner un profil à un utilisateur lors de sa création, vous devrez utiliser le mot clé PROFILE. Vous pourrez aussi assigner un nouveau profil après la création de l'utilisateur avec la syntaxe suivante:

```
  ALTER USER user
  PROFILE profile;
```

2.2. Administration des profils

2.2.1. Modification d'un profil

Vous pouvez aussi modifier un profil mais celui-ci ne sera valable pour l'utilisateur que lors de sa prochaine ouverture de session.

Voici la syntaxe qui permet de modifier un profil :

```
  ALTER PROFILE profile LIMIT
  [SESSIONS_PER_USER {integer | UNLIMITED | DEFAULT}]
  [CPU_PER_SESSION {integer | UNLIMITED | DEFAULT}]
  [CPU_PER_CALL {integer | UNLIMITED | DEFAULT}]
  [CONNECT_TIME {integer | UNLIMITED | DEFAULT}]
  [IDLE_TIME {integer | UNLIMITED | DEFAULT}]
  [LOGICAL_READS_PER_SESSION {integer | UNLIMITED | DEFAULT}]
  [LOGICAL_READS_PER_CALL {integer | UNLIMITED | DEFAULT}]
  [COMPOSITE_LIMIT {integer | UNLIMITED | DEFAULT}]
  [PRIVATE_SGA {integer | UNLIMITED | DEFAULT}]
  [FAILED_LOGIN_ATTEMPTS {integer | UNLIMITED | DEFAULT}]
  [PASSWORD_LIFE_TIME {integer | UNLIMITED | DEFAULT}]
  [PASSWORD_REUSE_TIME | PASSWORD_REUSE_MAX
       {integer | UNLIMITED | DEFAULT}]
  [PASSWORD_LOCK_TIME {integer | UNLIMITED | DEFAULT}]
  [PASSWORD_GRACE_TIME {integer | UNLIMITED | DEFAULT}]
  [PASSWORD_VERIFY_FUNCTION {function | NULL | DEFAULT}];
```

2.2.2. Suppression d'un profil

Voici la syntaxe qui permet de supprimer un profil :

```
  DROP PROFILE profile [CASCADE];
```

L'option CASCADE permet de réassigner le profil DEFAULT aux utilisateurs dont le profil vient d'être supprimé.

2.2.3. Affichage des limitations de ressources

Pour avoir des informations sur les limitations de ressources, vous pouvez consulter les vues systèmes suivantes : DBA_USERS et DBA_PROFILES.

2.3.Administration des mots de passe

2.3.1. Gestion des mots de passe

Pour augmenter le contrôle de la sécurité, les profils permettent de gérer, avec certaines fonctionnalités, les mots de passe des utilisateurs.
La gestion des mots de passe grâce aux profils va permettre de gérer le blocage des comptes, le renouvellement et l'expiration des mots de passe, un historique des mots de passe et finalement permettre de gérer la complexité des mots de passe.

- Le blocage d'un compte utilisateur : Cette fonctionnalité va permettre de bloquer automatiquement un compte lorsqu'un utilisateur se sera trompé de mots de passe un certain nombre de fois.
  Les paramètres qui permettent de gérer cette fonctionnalité sont :
  FAILED_LOGIN_ATTEMPTS et PASSWORD_LOCK_TIME

- Le renouvellement et l'expiration d'un compte utilisateur : Cette fonctionnalité va permettre de déterminer la durée de vie du mot de passe. C'est à la fin de cette durée de vie qu'il faudra renouveler son mot de passe.
  Les paramètres qui permettent de gérer cette fonctionnalité sont :
  PASSWORD_LIFE_TIME et PASSWORD_GRACE_TIME.
  Si cet utilisateur se retrouvait avec son compte bloqué, voici la syntaxe qui vous permettrait de déverrouiller son compte.

```
ALTER USER <nom du user>
[ IDENTIFIED { BY <mot de passe> | EXTERNALLY } ]
[ PASSWORD EXPIRE ]
[ ACCOUNT { LOCK | UNLOCK } ];
```

  Dont voici un exemple d'utilisation

```
ALTER USER toto
IDENTIFIED BY tata
ACCOUNT UNLOCK;
```

  Cet exemple va permettre de déverrouiller le compte de toto en lui assignant tata comme nouveau mot de passe.

- L'historique des mots de passe : Cette fonctionnalité va permettre de comparer le nouveau mot de passe aux anciens mots de passe de l'utilisateur afin d'être sur qu'il ne puisse pas réutiliser un ancien mot de passe pendant un certain nombre de fois ou pendant un certain temps.
  Les paramètres qui permettent de gérer cette fonctionnalité sont :
  PASSWORD_REUSE_TIME et PASSWORD_REUSE_MAX.

- La vérification de la complexité des mots de passe : Cette fonctionnalité va permettre de vérifier la complexité du mot de passe afin d'éviter toutes possibilités de piratage.
  Le paramètre qui permet de gérer cette fonctionnalité est :
  PASSWORD_VERIFY_FUNCTION.
  Cette fonction devra être crée dans le schéma de SYS et devra avoir le prototype suivant :

```
<nom de la fonction> (
                <id du user (nom du paramètre)> IN VARCHAR2(30),
                <password du user(nom du paramètre) IN VARCHAR2(30),
                <ancien password(nom du paramètre)>IN VARCHAR2(30))
                RETURN BOOLEAN
```

Si le mot de passe convient, la fonction devra retourner la valeur TRUE et FALSE si le mot de passe ne convient pas.

Il est possible d'utiliser la fonction qui vous est fournie par Oracle dans le fichier utlpwdmg.sql qui devra être exécutée dans le schéma SYS.

2.3.2. Création de profil: Gestion des mots de passe

Voici les deux étapes qui vont permettre au DBA de pouvoir administrer les mots de passe des utilisateurs:

- Création d'un profil avec les options concernant les mots de passe et l'assigner ensuite aux utilisateurs.
- Blocage, déblocage, expiration des mots de passe avec les commandes CREATE USER ET ALTER USER.

Les options sur les mots de passe sont toujours actives même si le paramètre RESOURCE_LIMIT du fichier init.ora est à FALSE.

2.3.3. Définition d'une fonction de gestion des mots de passe

Pour vérifier la complexité d'un mot de passe, il est possible grâce aux profils de passer une fonction PL/SQL qui va permettre de vérifier la complexité du mot de passe. De plus il est possible de créer soit même cette fonction en respectant certaines restrictions. Pour utiliser la fonction par défaut fournie par Oracle il faudra lancer le script utlpwdmg.sql.

2.3.4. Afficher les informations sur les profils et les utilisateurs

Le dictionnaire de données contient un grand nombre d'informations utiles sur les mots de passes et les comptes utilisateurs.

Pour obtenir ces informations vous pouvez consulter les vues DBA_USERS et DBA_PROFILES.

3. Consommation de ressources utilisateurs

3.1. Contrôle des ressources

3.1.1. Limitations de ressources grâce aux profils

Les profils vous permettent d'appliquer des limitations de ressources aux utilisateurs. A chaque fois qu'un utilisateur se connecte une session est alors crée. Chaque session est consommatrice de ressources mémoire et processeur. Les restrictions liées à la session seront donc appliquées aux ressources utilisées durant la session.

Les profils permettent non seulement de gérer les ressources au niveau de la session mais aussi au niveau des appels de requêtes ou autres appels système.

Les limitations au niveau de la session sont les restrictions sur ressources lors de la durée de la session (par exemple, un utilisateur laisse sa session inactive, vous pouvez limiter le temps d'inactivité d'une session).

Les limitations au niveau des appels sont les restrictions qui seront appliquées lors de chaque appel système.

Voila comment vont se dérouler les étapes lors du dépassement d'une limite au niveau des appels système.

- Si une limite au niveau instruction est dépassée : Le processus traitant l'instruction SQL sera alors stoppé et l'instruction courante sera alors annulée. Un message d'erreur est ensuite affiché à l'utilisateur. Toutes les transactions précédentes restent validée et l'utilisateur reste connecté.
- Si une limite au niveau session est dépassée : Un message d'erreur est envoyé à l'utilisateur qui est ensuite déconnecté automatiquement.

3.1.2. Activer les limitations de ressources

Il existe trois étapes nécessaires pour mettre en place la gestion des ressources par les profils :
- Créer un profil.
- Assigner ce profil aux utilisateurs
- Mettre en place les limitations de ressources.

Par défaut, les règles de gestion des ressources sont inactives.

Il existe deux moyens de mettre en place ces règles de gestion :
- En modifiant le paramètre RESOURCE_LIMIT (TRUE ou FALSE) du fichier init.ora. Cette méthode n'est seulement possible que lorsque vous pouvez arrêter la base.
- En modifiant le paramètre RESOURCE_LIMIT grâce à la commande ALTER SYSTEM. Cette méthode permet de modifier de manière dynamique le paramètre mais ne permettra pas de garder la valeur de ce paramètre si la base de donnée est redémarrée et que vous n'avez pas modifié le fichier init.ora.

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE ;
```

Il est aussi possible de modifier ce paramètre grâce à Oracle Entreprise Manager et en allant dans Oracle Instance Manager.

3.1.3. Afficher les limitations de ressources

Il existe deux vues système qui permettent d'obtenir des informations sur les limitations de ressources.

La première vue est DBA_USERS. Grâce à cette vue vous pourrez savoir quel profil a été attribué à chaque utilisateurs grâce à la colonne PROFILE.

La deuxième vue est DBA_PROFILES. Grâce à cette vue vous pourrez savoir toutes les limitations qui ont été mises en place pour un profil donné.

3.1.4. Forcer l'utilisation des limitations de ressources

Nous allons maintenant voir quelques exemples de limitations.

Limitations au niveau de la session :

- CPU_PER_SESSION : Limite le temps d'utilisation du processeur pour une session. Le temps devra être saisi en 100ème de secondes.
- SESSION_PER_USER : Limite le nombre de sessions concurrentes pour un utilisateur.
- CONNECT_TIME : Limite le temps de connexion pour une session. Le temps devra être saisi en minutes.
- IDLE_TIME : Limite le temps d'inactivité d'une session. Le temps devra être saisi en minutes. Cette limitation ne concerne que le processus serveur seulement. Elle ne tient pas compte d'autres opérations comme les activités d'une application tierce.
- LOGICAL_READS_PER_SESSION : Limite le nombre total de lectures logique de blocs de données durant la session. Cette limitation permet d'éviter un trop grand nombre d'entrée/sortie qui monopoliseraient les ressources système.
- PRIVATE_SGA : Limite la mémoire utilisée dans la SGA par un utilisateur connecté par l'intermédiaire d'un serveur multi-thread.
- COMPOSITE_LIMIT : Limite le coût total en ressource pour une session. Le coût total en ressource correspond à la somme des paramètres CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION et PRIVATE_SGA. Pour connaître la valeur de ces différents paramètres vous pouvez interroger la vue RESOUCE_COST.

Limitations au niveau des appels système :

- CPU_PER_CALL : Limite le temps d'utilisation processeur par appel. Le temps devra être saisi en 100ème de secondes.
- LOGICAL_READS_PER_CALL : Limite le nombre de lecture de blocs pour chaque appel.

3.2. Utilisation du gestionnaire de ressources

3.2.1. Mise en place du gestionnaire de ressources

La gestion des ressources pourra permettre à différents types d'utilisateurs de pouvoir utiliser aux mieux les ressources disponibles (par exemple les utilisateurs travaillant avec une application de type DSS ou de type OLTP).

Le gestionnaire de ressource va vous permettre de définir des groupes d'utilisateurs et de spécifier les ressources qui pourront leur être alloué. Ces groupes sont appelés groupe de consommateur de ressources.

Il est possible qu'un utilisateur fasse parti de plusieurs groupes.

La liste des groupes de consommateurs de ressource et leurs allocations de ressources est appelée un "Resource Plan". Ces "Resource Plan" sont uniques et sont stockés dans le dictionnaire de données. Il

peut exister plusieurs "Resource Plan" mais il ne peut y avoir qu'un seul "Resource Plan" actif à la fois.

Il existe ensuite des directives de planification de ressource qui permettent de pouvoir assigner aux groupes de consommateur de ressource, les ressources dont ils ont besoin.

Pour mettre en place cette gestion des ressources, vous aurez besoin de 2 packages PL/SQL :
- DBMS_RESOURCE_MANAGER
- DBMS_RESOURCE_MANAGER_PRIVS

Avant de pouvoir définir ces "Resource Plan" ou tout les autres objets nécessaires à la gestion des ressources, il est nécessaire de créer une zone d'attente (ou Pending Area). Cette Pending Area est une zone de travail qui va permettre de créer et de modifier les "Resource Plan". Toutes les modifications seront effectuées dans cette zone de travail jusqu'à ce qu'elles soient validées.
Pour pouvoir crée cette zone, il vous faudra utiliser la procédure CREATE_PENDING_AREA du package DBMS_RESOURCE_MANAGER.

Ensuite il faudra décider du nombre de groupes de consommateurs nécessaires puis les nommer et les créer. Il est tout à fait possible d'utiliser les groupes prédéfinis comme SYS_GROUP, LOW_GROUP, DEFAULT_CONSUMER_GROUP et OTHER_GROUPS.

- SYS_GROUP : est attribué pour les utilisateurs SYS et SYSTEM et étant en relation avec le SYSTEM_PLAN.

- LOW_GROUP : est donné aux utilisateurs ayant une priorité plus faible. Il est aussi en relation avec le SYSTEM_PLAN.

- DEFAULT_CONSUMER_GROUP : est attribué à tous les utilisateurs comme groupe par défaut jusqu'à ce que vous changiez l'utilisateur de groupe de consommateurs de ressources.

- OTHER_GROUPS : Ce groupe devra être inclus obligatoirement dans chaque "Resource Plan".

Voici la commande PL/SQL à exécuter pour créer un groupe de consommateurs:

```
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP('<nom du groupe>',
                                            '<Description>');
```

Ensuite une fois le groupe crée, voici la commande PL/SQL qui permet de rajouter un utilisateur à un groupe :

```
DBMS_RESOURCE_MANAGER_PRIVS.CREATE_CONSUMER_GROUP(
     '<nom du user>',
     '<nom du groupe>',
     <option d'administration>);
```

L'option d'administration (valeur TRUE et FALSE) permet de définir si cet utilisateur pourra ou non ajouter d'autres utilisateurs à ce groupe.

Quand vous créez de nouveaux utilisateurs, ceux-ci sont automatiquement ajoutés au groupe DEFAULT_CONSUMER_GROUP jusqu'à ce que vous les en changiez.

Le groupe initial d'un utilisateur peut être modifié grâce à la commande suivante :

```
DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP (
     '<nom du user>',
```

```
      '<nom du groupe>');
```

Les utilisateurs faisant partis de plusieurs groupes peuvent changer leur groupe actuel. Ce changement pourra impliquer une seule des sessions de l'utilisateur ou toutes ses sessions.

Voici les deux syntaxes :

- Pour changer de groupe pour toutes les sessions de l'utilisateur :

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER (
     '<nom du user>',
     '<nom du groupe>');
```

- Pour changer de groupe pour une session particulière :

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS (
     <SID de la session>,
     <SERIAL# de la session>,
     '<nom du groupe>');
```

Le gestionnaire de ressource nécessite un ou plusieurs "Resource Plan".
Comme vu précédemment un "Resource Plan" est une liste nommée de groupes de consommateurs de ressources et de spécification sur les ressources associées à chaque groupe.

Voici la syntaxe qui permet de créer un "Resource Plan" :

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN (
     '<nom du resource plan>',
     '<description du resource plan>');
```

Chaque "Resource Plan" doit contenir une directive qui permettra d'être redirigé vers le groupe OTHER_GROUPS.

Voici la syntaxe qui permet de créer une directive :

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
     '<nom du resource plan>',
     '<nom du groupe de consommateur de ressource>',
     '<description de la directive>',
     <limitation de ressource>, […]);
```

Un seul "Resource Plan" ne pouvant être actif à la fois, on définira le "Resource Plan" actif grâce au paramètre RESOURCE_MANAGER_PLAN du fichier init.ora.

On pourra modifier la valeur de ce paramètre soit directement dans le fichier init.ora, soit de manière dynamique avec la syntaxe suivante :

```
SQL> ALTER SYSTEM SET
  2  RESOURCE_MANAGER_PLAN=<nom du plan>;
```

Pour obtenir toutes les informations relatives à tous les objets du gestionnaire de ressource, il suffit d'interroger les vues du dictionnaire de données :

- DBA_RSRC_PLANS
- DBA_RSRC_PLAN_DIRECTIVES
- DBA_RSRC_CONSUMER_GROUPS
- DBA_RSRC_CONSUMER_GROUPS_PRIVS
- DBA_USERS

4. Gestion des privilèges

4.1.Privilèges : Introduction

4.1.1. Privilèges : Types

Les privilèges sont une part importante de la sécurité d'une base de données Oracle.

Quand un utilisateur se connecte à une base de données celui-ci est alors authentifié. Le serveur Oracle effectue cette authentification en vérifiant la validité du nom et du mot de passe de l'utilisateur.
Cette authentification constitue la première étape vers la "sécurisation" de la base de données.
Les autres étapes permettant de sécuriser la base de données, est de donner des privilèges aux utilisateurs.

Il existe deux types de privilèges:

- Les privilèges systèmes : Ce type de privilège permet d'autoriser l'utilisateur d'effectuer un certain nombre d'opérations dans la base de données (comme se connecter ou bien créer modifier ou bien supprimer des objets de la base).Ces privilèges ne sont pas spécifiques à un objet de schéma.
- Les privilèges objets : Ce type de privilège permet d'autoriser l'utilisateur à effectuer des modifications et à accéder à certains objets de la base de données comme une table, une vue. Vous pouvez donner des droits sur des objets comme UPDATE ON, DELETE ON, EXECUTE ON, SELECT ON.

Attention : Il est dangereux d'attribuer les privilèges système comportant le mot clé ANY, car ceux-ci sont très puissants et peuvent dans certains cas rendre accessible tout les objets de la base de données.

4.1.2. Privilèges système

Nous allons maintenant voir de manière plus spécifique les privilèges systèmes.

Il faut savoir que tout les privilèges contenant le mot clé CREATE (tels que CREATE TABLE, CREATE PROCEDURE, etc…) incluent aussi les privilèges de suppression (tels que DROP TABLE, DROP PROCEDURE, etc…) car les objets de la base peuvent toujours être supprimés par leur propriétaire.

Les utilisateurs possédant le privilège CREATE TABLE ont donc aussi de manière implicite les droits de faire toutes les actions possible sur leur table comme de créer des index (CREATE INDEX mais ce privilège n'existe pas réellement mais il existe un privilège objet INDEX) ou de lancer des analyses statistiques (ANALYSE) sur leurs tables.

Les rôles sont des groupes nommés qui peuvent contenir un ou plusieurs privilèges systèmes et être ainsi donnés à des utilisateurs ou bien à d'autres rôles.

Attention : Le privilège système UNLIMITED TABLESPACE ne peut en aucun cas être donné à un rôle. Il faut aussi le donner avec précaution car ce rôle donne accès à tout l'espace disponible dans tous les tablespaces y compris le tablespace SYSTEM.

Voici des exemples de privilèges système :

Catégorie	Exemples
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX
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 UNLIMITED TABLESPACE

4.2.Gestion des privilèges système

4.2.1. Accorder des privilèges système

Un rôle est donc un ensemble nommé de privilèges qui peut ensuite être donné à des utilisateurs ou bien à d'autres rôles.

Voici la syntaxe qui permet de donner un rôle :

```
GRANT      {<nom du privilège> | <nom du rôle>}
     [, {<nom du privilège> | <nom du rôle>}] …
TO   {<nom d'un user> | <nom d'un rôle> | PUBLIC}
     [, {<nom d'un user> | <nom d'un rôle> | PUBLIC}] …
     [WITH ADMIN OPTION];
```

Le mot clé PUBLIC permet de donner un privilège système ou un rôle à l'ensemble des utilisateurs. Les mots clé WITH ADMIN OPTION permettent d'autoriser la personne ayant reçu ce rôle de le donner lui-même à d'autres utilisateurs ou rôles.

Il y a quelques points à savoir, dans l'environnement Oracle, pour donner un privilège système ou un rôle :

- Pour pouvoir donner un privilège système, il faut être DBA ou avoir reçu explicitement ce privilège avec l'option WITH ADMIN OPTION.
- La personne ayant l'option WITH ADMIN OPTION pourra à son tour donner ce privilège système à d'autres utilisateurs avec ou sans l'option WITH ADMIN OPTION.
- N'importe quel utilisateur ayant reçu le privilège système GRANT ANY ROLE peut donner n'importe quel rôle.
- L'utilisateur ayant l'option WITH ADMIN OPTION pourra donner ou retirer ce privilège système à n'importe quel utilisateur de la base de données sauf à celui lui ayant donné ce privilège.

4.2.2. Les privilèges SYSDBA et SYSOPER

Pour permettre à un utilisateur d'effectuer les nombreuses opérations de DBA, il faut lui donner des privilèges de DBA comme SYSDBA et SYSOPER. Le choix entre ces deux privilèges dépendant bien sur des actions que ce "DBA" devra effectuer.

Lorsque l'utilisateur possédant un des deux privilèges SYSDBA ou SYSOPER se connecte à la base, il est connecté de manière implicite au schéma de SYS.

Il faut savoir que le privilège SYSDBA fournit à l'utilisateur des privilèges DBA illimités qui lui permettent d'effectuer n'importe quelle opération.

Le privilège SYSOPER est un sous ensemble du privilège SYSDBA. Ce privilège permet de démarrer, arrêter, sauvegarder la base, mais ne permet pas de créer une base de données.

Voici un exemple des privilèges systèmes fournit par les deux différents privilèges.

Catégorie	Exemples
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE OPEN | MOUNT ALTER DATABASE BACKUP CONTROLFILE ALTER DATABSE BEGIN | END BACKUP RECOVER DATABASE ALTER DATABASE ARCHIVELOG RESTRICTED SESSION
SYSDBA	Tous les privilèges de SYSOPER mais avec l'option WITH ADMIN OPTION CREATE DATABASE RECOVER DATABASE UNTIL

Il est important de savoir que les privilèges SYSDBA et SYSOPER ne peuvent être attribués à un rôle.

4.2.3. L'authentification grâce au fichier de mots de passe

Il y a 4 étapes pour mettre en place le fichier de mots de passe servant à identifier les utilisateurs SYSDBA et SYSOPER:

- Création du fichier de mots de passe si celui-ci n'existe pas. Pour cela vous pouvez utiliser l'utilitaire de création de fichier de mot de passe ORAPWD.
- Vérifier que le paramètre REMOTE_LOGIN_PASSWORD_FILE du fichier init.ora possède la valeur EXCLUSIVE.
- Donner les privilèges SYSDBA et SYSOPER aux utilisateurs souhaités. Leur login sera alors automatiquement ajouté au fichier de mot de passe.
- Vérifier les utilisateurs présents dans le fichier de mot de passe grâce à la vue V$PWFILE_USERS. Cette vue contient 3 colonnes, l'une correspondant au nom du user, une autre correspondant au fait qu'il possède le privilège SYSDBA et une autre correspondant au fait qu'il possède le privilège SYSOPER.

4.2.4. Mécanisme de protection du dictionnaire de données

Les utilisateurs ont des accès restreint aux objets du dictionnaire de données sauf si ils ont été autorisés à accéder à des objets d'autres schémas.

Vous pouvez contrôler la protection du dictionnaire de données en modifiant le paramètre O7_DICTIONNARY_ACCESSIBILITY du fichier init.ora.
Si ce paramètre a pour valeur la valeur TRUE alors l'accès aux objets de SYS est autorisé, ce qui désactive le mécanisme de protection.

Si ce paramètre a pour valeur la valeur FALSE alors l'accès aux objets de SYS n'est plus autorisé; ce qui permet à des privilèges systèmes tels que EXECUTE ANY PROCEDURE d'exécuter n'importe quelle procédure de la base sauf celle du schéma de SYS.

4.2.5. Retrait de privilèges système

Il faut avoir la vision la plus précise possible des privilèges que vous allez fournir à un utilisateur car donner trop de droits à un utilisateur peut entraîner de nombreux problèmes de sécurité.

Vous devrez donc retirer des privilèges systèmes à un utilisateur ou bien à un rôle si ceux-ci ne sont plus utiles aux actions de l'utilisateur.
Voici la syntaxe :

```
REVOKE      {<nom du privilège> | <nom du role>}
     [,{<nom du privilège> | <nom du role>}] …
FROM        {<nom du user> | <nom du role> | PUBLIC}
     [,{<nom du user> | <nom du role> | PUBLIC} …;
```

4.2.6. Retrait de privilèges : WITH ADMIN OPTION

Nous avons vu précédemment qu'un utilisateur ayant reçu un privilège système avec l'option WITH ADMIN OPTION, pouvait à son tour donner des droits à d'autres utilisateurs.

Il faut être extrêmement prudent avec cette option car quand vous retirerez le privilège système à cet utilisateur, tout les autres utilisateurs, ayant reçu ce même privilège de cet utilisateur, conserverons ce privilège.

La révocation de ce privilège ne produit pas un effet en cascade.

4.3.Gestion des privilèges Objet

4.3.1. Accorder des privilèges objet

Les privilèges objets permettent aux utilisateurs d'effectuer des actions bien spécifiques sur des objets de la base de données, comme par exemple effacer des lignes d'une table en particulier.

Voici un tableau vous donnant une idée des privilèges objets que vous allez pouvoir utiliser sur certains objets:

Privilège Objet	Table	Vue	Séquence	Procédure
ALTER	✓		✓	
DELETE	✓	✓		
EXECUTE				✓
INDEX	✓			
INSERT	✓	✓		
REFERENCES	✓			
SELECT	✓	✓	✓	
UPDATE	✓	✓		

Il est tout à fait possible de donner ces privilèges objets à des rôles ou bien des utilisateurs.

Voici la syntaxe permettant de donner un ou plusieurs privilèges objet :

```
GRANT       {<nom du privilège> [<liste de colonne>]
     [,<nom du privilège> [<liste de colonne>]] … |
     ALL [PRIVILEGES]}
ON    <nom de l'objet>
TO    {<nom du user> | <nom du rôle> | PUBLIC}
     [,{<nom du user> | <nom du rôle> | PUBLIC}] …
     [WITH GRANT OPTION] ;
```

L'option WITH GRANT OPTION permet à l'utilisateur ayant reçu ce privilège objet de donner à son tour ce privilège.

Exemple :

```
GRANT EXECUTE ON dbms_pipe TO public;

GRANT UPDATE (empno, ename)
ON emp
TO Karen
WITH GRANT OPTION;
```

Il y a certaines règles à suivre avant de donner des privilèges objets.

- Pour donner un privilège objet à quelqu'un, il faut que cet objet vous appartienne ou bien que vous ayez reçu ce privilège avec l'option WITH GRANT OPTION.
- Vous possédez tous les privilèges objets ainsi que l'option WITH GRANT OPTION pour tous les objets qui vous appartiennent.
- Pour des raisons de sécurités, faites très attention lorsque vous donner des droits sur vos objets.
- L'option WITH GRANT OPTION ne peut être utilisée lorsque vous donnez un privilège objet à un rôle.

4.3.2. Retrait de privilèges objet

Il faut toujours se rappeler que donner trop de droits peut engendrer des problèmes de sécurité.

Nous allons donc voir comment retirer des droits à des utilisateurs ou à des rôles.

Voici la syntaxe :

```
REVOKE     {<nom du privilège> [, <nom du privilège>] …|
     ALL [PRIVILEGES]}
ON   <nom de l'objet>
FROM       {<nom du user> | <nom du rôle> | PUBLIC}
     [, {<nom du user> | <nom du rôle> | PUBLIC }]…
     [CASCADE CONSTRAINTS];
```

4.3.3. Retrait de privilèges : GRANT OPTION

A la différence de l'option WITH ADMIN OPTION, l'option WITH GRANT OPTION se retire en cascade. Si l'utilisateur à qui vous aviez donné un privilège avec l'option WITH GRANT OPTION l'avait donné à son tour à un autre utilisateur, et bien lorsque vous retirez ce privilège à cet utilisateur, alors tout ceux qui ont reçu ce même privilège de cet utilisateur se le verront retirer automatiquement.

4.4. Monitoring des privilèges

4.4.1. Affichage des privilèges système

Vous pourrez en tant que DBA aller chercher toutes les informations concernant les privilèges systèmes dans les vues système DBA_SYS_PRIVS, et SESSION_PRIVS.

4.4.2. Affichage des privilèges objet

Vous pourrez trouver les informations concernant les privilèges objets dans les vues DBA_TAB_PRIVS et DBA_COL_PRIVS.

5. Gestion des Rôles

5.1. Rôles : Introduction

5.1.1. Caractéristiques des rôles

Les rôles ont été mis en place afin de faciliter l'administration des privilèges de la base de données.

Ces rôles :

- Peuvent être composé de privilèges système et objet.
- Peuvent être donné à des utilisateurs et à des rôles (excepté lui-même).
- Les rôles peuvent être activés ou désactivés.
- Le nom d'un rôle ne pourra en aucun cas être celui d'un autre rôle ou celui d'un utilisateur.
- Les rôles n'appartiennent à aucun utilisateur et ne font partis d'aucun schéma.
- La description et les privilèges associés à un rôle sont stockés dans le dictionnaire de données.
- Pour donner ou retirer un droit, il suffit d'utiliser la même commande que pour donner un privilège.

5.1.2. Bénéfices des rôles

L'utilisation des rôles permet de :

- Réduire le besoin de donner des droits de manière individuel. Grâce aux rôles vous pouvez donner des droits à un rôle et donner ce rôle à des utilisateurs, ce qui vous évitera de donner tous ces droits à chaque utilisateur.
- Gérer de manière dynamique la gestion des privilèges. Si vous modifiez les privilèges associés à un rôle, alors tout les utilisateurs, possédant ce rôle, obtiendront les modifications de manière transparente et automatique.
- Gérer la sécurité en activant ou désactivant ces rôles, et en les sécurisant au moyen d'un mot de passe.
- Supprimer les effets de suppressions en cascade absent lors de la suppression de privilèges système ayant été fournit avec l'option WITH ADMIN OPTION.
- Diminuer le stockage de tous les ordres GRANT dans le dictionnaire de données.
- Réduire le nombre de privilèges à vérifier si le rôle est désactivé. Si le rôle est désactivé, il faudra moins de temps pour vérifier les droits de l'utilisateur lors de l'exécution d'une requête.

5.2. Implémentation des Rôles

5.2.1. Création d'un rôle

Voici la syntaxe qui permet de créer un rôle :

```
CREATE ROLE <nom du rôle>
[NOT IDENTIFIED | IDENTIFIED {BY <mot de passe> | EXTERNALLY}];
```

5.2.2. Identification des rôles prédéfinis

Il existe plusieurs rôles prédéfinis, dont voici un tableau récapitulatif :

Rôles prédéfinis	**Descriptions**
RESOURCE	Permet d'accéder aux ressources système. Attention ce rôle donne le privilège système UNLIMITED TABLESPACE. Il procure aussi une rétro-compatibilité avec la version 6 d'Oracle.

CONNECT	Permet à l'utilisateur de se connecter et de crée des objets à la condition que vous lui donniez un quota sur un tablespace.
DBA	Donne tout les privilèges système avec l'option WITH ADMIN OPTION
EXP_FULL_DATABASE	Permet de faire des exports complets de la base de données.
IMP_FULL_DATABASE	Permet de faire des imports complets de la base de données.
DELETE_CATALOG_ROLE	Permet de vider les tables d'audit.
EXECUTE_CATALOG_ROLE	Permet d'exécuter les packages PL/SQL système.
SELECT_CATALOG_ROLE	Permet d'avoir accès sur les tables et vues du dictionnaire de données.

Mis à part le rôle DBA, il n'est pas conseillé d'utiliser les rôles prédéfinis d'Oracle mais plutôt de créer et d'utiliser vos propres rôles.

5.2.3. Modifier un rôle

La seule chose qu'il est possible de modifier dans un rôle est son système de protection par mots de passe. Ceci peut être fait grâce à la syntaxe suivante:

```
ALTER ROLE <nom du rôle>
{ NOT IDENTIFIED | IDENTIFIED { BY <mot de passe> | EXTERNALLY }};
```

Le mot clé EXTERNALLY signifie que l'utilisateur devra être authentifié par le système d'exploitation avant de pouvoir activer ou désactiver ce rôle.

5.2.4. Assignation d'un rôle à un utilisateur

Voici la syntaxe pour assigner un rôle à un utilisateur ou à un autre rôle :

```
GRANT <nom du rôle> [,<nom du rôle>] …
TO {<nom du user> | <nom du rôle> | PUBLIC}
[, {<nom du user> | <nom du rôle> | PUBLIC}] …
[WITH ADMIN OPTION];
```

5.3.Contrôler les Rôles

5.3.1. Limitation du rôle par défaut

Le rôle par défaut des utilisateurs doit être défini en fonction de leur travail, ce qui permet de restreindre leurs activités sur les schémas.

Voici la syntaxe pour modifier le rôle par défaut d'un utilisateur:

```
ALTER USER <nom du user> DEFAUTL ROLE
{<nom du rôle> [, <nom du rôle>] … | ALL
[EXCEPT <nom du rôle> [, <nom du rôle>] …] | NONE};
```

5.3.2. Activation d'un rôle

Il est possible d'activer ou de désactiver un rôle temporairement pour une session. Quand un rôle est activé, l'utilisateur pourra alors utiliser les privilèges de ce rôle.

Voici la syntaxe qui permet d'activer ou de désactiver un rôle :

```
SET ROLE { <nom du rôle> [IDENTIFIED BY PASSWORD]
     [, <nom du rôle> [IDENTIFIED BY PASSWORD>]] … | ALL
     [EXCEPT <nom du rôle> [, <nom du rôle>] …] | NONE};
```

Cette commande désactive tout les rôles et réactive seulement ceux cités dans la commande.

5.3.3. Retrait d'un rôle à un utilisateur

Voici la syntaxe qui permet de retirer un rôle à un utilisateur ou à un autre rôle :

```
REVOKE <nom du rôle> [,<nom du rôle>] …
FROM {<nom du user> | <nom du rôle> | PUBLIC}
[,{<nom du user> | <nom du rôle> | PUBLIC}] …;
```

5.3.4. Suppression d'un rôle

Afin de s'assurer qu'un rôle ne peut plus être utilisé, il vaut mieux supprimer le rôle avec la syntaxe suivante :

```
DROP ROLE <nom du rôle>;
```

5.4. Rôles : Informations et Conseils

5.4.1. Conseils pour la création de rôles

Nous allons voir les différents points qu'il faut suivre pour mettre en place des rôles appropriés aux besoins des utilisateurs.

- Créer autant de rôle que de fonctions existantes.
- Assigner les privilèges nécessaires pour effectuer une fonction au rôle correspondant.
- Créer un rôle (rôle utilisateur) distinct pour chaque type d'utilisateur.
- Donner au rôle utilisateur, les rôles des différentes fonctions que ce type d'utilisateur pourra effectuer.
- Donner à chaque utilisateur son rôle utilisateur.
- Donner un mot de passe à chaque rôle.

5.4.2. Affichage des infos concernant les rôles

Vous pourrez trouver des informations sur les différents rôles dans les vues système suivantes :

- ROLE_SYS_PRIVS : vue concernant les privilèges système accordés à des rôles.
- ROLE_TAB_PRIVS : vue concernant les privilèges sur une table accordés à des rôles.
- ROLE_ROLE_PRIVS : vue concernant les rôles accorder à d'autres rôles.
- DBA_SYS_PRIVS : vue permettant de voir les privilèges systèmes accordés à des utilisateurs et à des rôles.
- DBA_ROLES : vue permettant d'accéder à tous les rôles présents dans la base de données.
- DBA_ROLE_PRIVS : vue permettant les rôles qui ont été accordés à d'autres rôles ou utilisateurs.

5.4.3. Affichage des rôles et privilèges actifs

Vous pouvez savoir quels sont les rôles actuellement actif pour un utilisateur en consultant la vue SESSION_ROLES.

La vue SESSION_PRIVS permet quand à elle de savoir quels sont les privilèges systèmes actuellement disponible pour une session d'un utilisateur.

6. Audit

6.1. Utilisation de l'audit de base de données

6.1.1. Catégories d'audit

Il existe trois catégories d'audit dans une base de données Oracle

- Audits des privilèges : Ces audits enregistrent les démarrages et arrêts de l'instance. Ils incluent le nom de l'utilisateur, le nom du terminal, la date et l'heure. Ils enregistrent aussi toutes les connexions effectuées par des utilisateurs SYSDBA ou SYSOPER.

- Audits de la base de données : Ces audits servent de moniteur et permettent d'enregistrer des informations sur les utilisateurs et sur les activités de la base de données. Toutes ces informations d'audit sont enregistrées dans un protocole d'audit qui est stocké dans la table AUD$ qui se trouve dans le dictionnaire de données. Ce protocole d'audit peut être utilisé pour détecter toutes les activités douteuses survenues dans la base. Ces audits permettent aussi d'enregistrer certaines activités de la base de données comme les entrées / sorties.

- Audits d'applications : Ces audits permettent de chercher toutes les modifications sur les colonnes et enregistrent ces modifications. Ce type d'audit doit être codé (par exemple sous forme de trigger) car il ne fait pas parti des outils standard d'audit.

6.1.2. Audit de base de données: Phases

Avant de faire un audit, il est important de définir une méthode d'audit qui va nous éviter de générer un trop grand nombre d'informations inutiles et de faire grossir de manière incontrôlée l'audit.

- La première phase dans un audit de base de données est d'activer l'audit au moyen du paramètre AUDIT_TRAIL du fichier init.ora. Puis de définir les actions d'audit au moyen de la commande AUDIT. Le paramètre AUDIT_TRAIL permet de définir où est ce que les informations d'audit seront enregistrées (dans la base ou dans le gestionnaire d'audit du système d'exploitation). Vous devrez ensuite grâce à la commande AUDIT définir des audits sur certains objets ou privilèges et définir si cet audit devra être enregistré pour chaque occurrence de cet évènement ou bien une seule fois par session.
- La seconde phase dans un audit de base de données est de générer des actions SQL et PL/SQL. Ce sont alors les options d'audit qui détermineront si oui ou non il faut générer des enregistrements d'audit.
- La troisième et dernière phase d'audit consiste à analyser les informations d'audit en allant examiner les vues du dictionnaire de données ou en utilisant un logiciel du système d'exploitation.

6.1.3. Valeur du paramètre AUDIT_TRAIL

Vous seul, en tant que DBA, pourrez activer ou désactiver l'audit grâce au paramètre AUDIT_TRAIL. Ce paramètre pourra avoir les valeurs suivantes :

- DB : Ce paramètre signifie que tous les enregistrements seront stockés dans la table AUD$.
- OS : Ce paramètre signifie que tous les enregistrements seront stockés dans le gestionnaire d'audit du système d'exploitation (par exemple dans le Gestionnaire des évènements de Windows NT).
- NONE : L'audit est désactivé. A noter qu'il est possible d'utiliser les commandes AUDIT et NOAUDIT lorsque le paramètre AUDIT_TRAIL est à la valeur NONE.

6.1.4. Requête d'audit

L'audit des ordres SQL se fera indépendamment du schéma de l'utilisateur. L'audit de requête se fera au moyen de la syntaxe suivante :

```
AUDIT <requête> [,<requête>] …
[BY <nom du user> [,<nom du user>] … ]
[BY {SESSION | ACCESS}]
[WHENEVER [NOT] SUCCESSFUL];
```

L'option BY permet de spécifier le ou les utilisateurs qui devront être audités (le paramètre par défaut est : tous les utilisateurs).
L'option BY SESSION permet d'enregistrer une seule fois un évènement par session alors que l'option BY ACCESS permet d'enregistrer le même événement autant de fois que celui-ci ce produit.
La valeur par défaut de l'option WHENEVER correspond au deux valeurs.

Voici un exemple d'audit de requête :

```
AUDIT DELETE ON scott.s_emp
BY ACCESS
WHENEVER NOT SUCCESSFUL;
```

Cet exemple enregistre toutes les tentatives de DELETE, sur la table s_emp de SCOTT, qui n'ont pas abouties.

6.1.5. Privilèges d'audit

L'audit des privilèges ne se base pas sur un objet particulier mais seulement sur le privilège utilisé.

Voici la syntaxe :

```
AUDIT < privilège système > [,<privilège système>] …
[BY <nom du user> [,<nom du user>] … ]
[BY {SESSION | ACCESS}]
[WHENEVER [NOT] SUCCESSFUL];
```

Oracle fournit un certain nombre de raccourci sur les privilèges système qui vous permettront d'auditer des privilèges multiple de manière simple.

Voici des exemples de raccourci :

- RESOURCE contient les privilèges suivants :
  - ALTER SESSION
  - CREATE CLUSTER
  - CREATE DATABASE LINK
  - CREATE PROCEDURE
  - CREATE ROLLBACK SEGMENT
  - CREATE SEQUENCE
  - CREATE SYNONYM
  - CREATE TABLE
  - CREATE TABLESPACE
  - CREATE VIEW
- CONNECT contient les privilèges suivants :
  - CREATE SESSION
- DBA contient les privilèges suivants :
  - AUDIT SYSTEM
  - CREATE PUBLIC DATABASE LINK
  - CREATE PUBLIC SYNONYM

        - CREATE ROLE
        - CREATE USER
- ALL PRIVILEGES contient les privilèges suivants :
    - Tous les privilèges système.

Exemple :

```
AUDIT ALTER ANY TABLE
BY scott
WHENEVER SUCCESSFUL;
```

6.1.6. Audit des objets de base de données

Il y a différents types d'objets qui peuvent être audité :

- Les tables
- Les vues
- Les séquences
- Les packages
- Les procédures
- Les fonctions

Voici la syntaxe permettant d'auditer un objet en particulier :

```
AUDIT      <operation devant être auditée>
     [,<operation devant être auditée>] …
ON {<nom de l'objet> | DEFAULT}
[BY {SESSION | ACCESS}]
[WHENERVER [NOT] SUCCESSFUL];
```

L'option DEFAULT spécifie que les options d'audit seront mises en place pour tout les objets nouvellement crées.

Ce type d'audit ne tiens pas compte de l'utilisateur ayant effectué l'opération auditée.

Exemple :

```
AUDIT DELETE ON scott.s_emp
BY SESSION
WHENEVER SUCCESSFUL;
```

6.1.7. Désactiver les options d'audit

Après avoir effectué tous vos audits, il est nécessaire de désactiver cette fonction afin d'éviter une augmentation inutile du référentiel d'audit. Pour cela, il existe deux possibilités :

- Soit donner la valeur NONE au paramètre AUDIT_TRAIL
- Soit désactiver chaque commande d'audit au moyen de la commande NOAUDIT

Voici la syntaxe de la commande NOAUDIT :

```
NOAUDIT { <requête> | <privilège système> }
[ BY <nom du user> ]
[ WHENERVER [ NOT ] SUCCESSFUL ];
```

Et voici la syntaxe de la commande NOAUDIT pour un audit base sur une requête :

```
NOAUDIT <requête>
ON { <nom de l'objet> | DEFAULT }
[ WHENERVER [ NOT ] SUCCESSFUL ];
```

Exemple :

```
NOAUDIT ALTER TABLE
BY scott
WHENEVER SUCCESSFUL;
```

6.2. Affichage des résultats d'audit

6.2.1. Vues concernant les options d'audit

Voici les vues ou vous pourrez allez chercher les informations concernant l'audit en général :

- ALL_DEF_AUDIT_OPTS : Cette vue contient toutes les options d'audit par défaut. Ces options correspondent à toutes les options d'audits dont héritera tout nouvel objet de la base.
- DBA_STMT_AUDIT_OPTS : Cette vue affichera toutes les options sur les audits de requête SQL.
- DBA_PRIV_AUDIT_OPTS : Cette vue contient toutes les options sur les audits de privilèges systèmes.
- DBA_OBJ_AUDIT_OPTS : Cette vue contient toutes les options sur les audits d'objets.

6.2.2. Afficher les résultats d'audit

Voici les vues contenant le résultat des audits :

- DBA_AUDIT_TRAIL : Cette vue contient absolument tous les résultats de tous les audits.
- DBA_AUDIT_SESSION : Cette vue contient toutes les connections et déconnections.
- DBA_AUDIT_STATEMENT : Cette vue contient les résultats des audits de requêtes SQL.
- DBA_AUDIT_OBJECT : Cette vue contient les résultats des audits d'objets.
- DBA_AUDIT_EXISTS : Cette vue contient les résultats sur les audits de type EXISTS/NOT EXISTS.

6.3. Vérification de la validité de l'audit

6.3.1. Conseils pour l'audit

Pour effectuer un audit intéressant, il est essentiel de suivre quelques règles :

- Identifier les nécessités à auditer, et définir les options minimales d'audit qui permettront d'obtenir le résultat escompté.
- Spécifier l'utilisateur qui devra être audité lors de l'audit des privilèges et des requêtes.
- Il est préférable d'utiliser l'option BY SESSION et non BY ACCESS. Cela permettra d'éviter d'enregistrer de nombreuses fois les mêmes opérations.
- Vous devez vérifier, lors d'audit d'objets, la vue DBA_AUDIT_OBJECT car les utilisateurs peuvent auditer leur propre objets.
- Vous devez faire attention lorsque vous donnez le privilège AUDIT ANY car cela donne le droit à un utilisateur d'auditer n'importe quel objet de son schéma.
- Ne pas oublier d'effacer le contenu de l'audit une fois les analyses terminées.
- Pour minimiser le nombre de lignes générées par l'audit, il est fortement conseillé d'utiliser l'option WHENEVER et de lui spécifier une valeur.
- Il est conseillé de déplacer la table AUD$, qui lors d'audit ne fait qu'augmenter et fragmenter le tablespace SYSTEM.
- Il est nécessaire de surveiller régulièrement la taille de la table AUD$.
- Il faut sécuriser les informations de l'audit. Pour cela seul le DBA devra posséder le rôle DELETE_CATALOG_ROLE.
- Vous ne devrez activer l'audit que lorsque cela est nécessaire.

6.3.2. Déplacer la table d'audit

Nous allons maintenant voir comment déplacer la table AUD$ sur un tablespace différent du tablespace SYSTEM.

Avant de déplacer cette table il est nécessaire de vérifier que l'audit est bien désactivé.

Ensuite il suffit de taper la commande suivante :

```
ALTER TABLE aud$
MOVE TABLESPACE <nom du nouveau tablespace>;
```

Nous allons ensuite créer un nouvel index pour la table AUD$ dans un tablespace différent du tablespace SYSTEM et du tablespace de la table AUD$ avec la syntaxe suivante :

```
CREATE INDEX <nom de l'index>
ON aud$(sessionid,ses$tid)
TABLESPACE <nom du tablespace>;
```