Oracle

- Rôles
- Bases de données privées virtuelles(VPD)

Patricia Serrano Alvarado
Université de Nantes - LINA

Les rôles dans Oracle

- Privilège
  - Droit d'exécuter une sentence SQL ou d'accéder les objets (tables, vues, objets, etc.) d'un autre utilisateur
  - Un privilège représente le couple (objet, privilège)
  - 173 privilèges dans Oracle 10g (SYSTEM_PRIVILEGE_MAP)
- Rôle
  - Ensemble de privilèges qui peut être attribué aux utilisateurs ou aux rôles

Types de privilèges

- Privilèges objets
  - Privilèges permettant la gestion des objets crées par les utilisateurs
  - CREATE, SELECT, INSERT, etc.
- Privilèges système
  - Privilèges permettant la gestion du système de bases de données
  - Ne sont pas liés à un objet ou schéma
  - CREATE SESSION, CREATE ROLE, ALTER ROLE, etc.
- Rôles
  - Peuvent contenir de privilèges objets et système

Privilèges objets

Objet	Privilège
Tables	select, insert, update, delete, alter, debug, flashback, on commit refresh, query rewrite, references, all, index
Views	select, insert, update, delete, under, references, flashback, debug
Sequence	alter, select
Packeges, Procedures, Functions (Java classes, sources...)	execute, debug
Materialized Views	delete, flashback, insert, select, update
Directories	read, write
Libraries	execute
User defined types	execute, debug, under
Operators	execute
Indextypes	execute

Privilèges sur les objets

Object Privilege	Table	View	Sequence	Procedures, Functions, Packages	Materialized View	Directory	Library	User-defined Type	Operator	Index-type
ALTER	X		X							
DELETE	X	X			X[b]					
EXECUTE				X[c]			X[c]	X[c]	X[c]	X[c]
DEBUG	X	X		X				X		
FLASHBACK	X	X			X					
INDEX	X									
INSERT	X	X			X[b]					
ON COMMIT REFRESH	X									
QUERY REWRITE	X									
READ						X				
REFERENCES	X	X								
SELECT	X	X	X		X					
UNDER		X						X		
UPDATE	X	X			X[b]					
WRITE						X				

Privilèges système

- Définis par Oracle et pas modifiables par les utilisateurs
- Peuvent être attribués uniquement par les administrateurs du système ayant comme privilèges
  - Le privilège à attribuer avec l'option ADMIN OPTION
  - Le privilège général GRANT ANY PRIVILEGE
- Les privilèges sur les objets ne peuvent pas être attribués avec des privilèges systèmes ou rôles dans la même sentence GRANT

L'option ADMIN OPTION

- Utilisable avec les rôles et les privilèges système
- Semblable à WITH GRANT OPTION des privilèges objets
- Le receveur peut
  - Attribuer ou révoquer le rôle ou le privilège système
  - Attribuer le rôle ou le privilège système avec l'option ADMIN OPTION
  - Modifier (ALTER) ou supprimer le rôle
- Pour annuler l'option, il faut révoquer le privilège
- Exemples
  - GRANT myRole TO alice WITH ADMIN OPTION;
  - REVOKE myRole FROM alice;

Avantages de l'utilisation de rôles

- Gestion de privilèges réduite
  - Au lieu d'attribuer un ensemble de privilèges aux utilisateurs, un par un, les privilèges sont attribués à un rôle et uniquement le rôle est attribué aux utilisateurs
- Gestion dynamique des privilèges
  - Si les privilèges d'un groupe change, uniquement le rôle sera modifié
- Surveillance plus simple
  - Plus facile de vérifier les privilèges des utilisateurs

Utilisation des rôles

- Pour la gestion des privilèges d'un group d'utilisateurs
- Pour la gestion des privilèges d'une application de base de données
  - Attribution à un rôle des privilèges nécessaires à l'exécution d'une application
  - Le rôle d'application crée sera attribué aux utilisateurs de l'application

Rôles prédéfinis d'Oracle

- Définis automatiquement lors de l'installation d'Oracle
- Aident à la gestion de la base de données
- Peuvent être gérés comme les rôles définis par un utilisateur
- Exemples
  - CONNECT : CREATE VIEW, CREATE TABLE, ALTER SESSION, CREATE CLUSTER, CREATE SESSION, CREATE SYNONYM, CREATE SEQUENCE, CREATE DATABASE LINK
  - RESOURCE : CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE

Gestion de rôles

- Création/suppression
  - Un rôle peut être crée et supprimé
- Sécurisation
  - Les rôles peuvent être protégés par un mot de passe pour éviter que n'importe qui les ajoute/supprime/attribue de privilèges
- Attribution
  - Un rôle peut être attribué aux utilisateurs ou aux rôles
  - On peut attribuer des privilèges aux rôles

Création/suppression de rôles

- Création de rôles
  - Uniquement par des utilisateurs
    - Ayant le privilège CREATE ROLE
- Modification de rôles
  - Par les utilisateurs
    - Ayant eu attribué le rôle avec l'option ADMIN OPTION
    - Ayant le privilège ALTER ANY ROLE
- Suppression de rôles
  - Par les utilisateurs
    - Ayant créé le rôle
    - Ayant eu attribué le rôle avec l'option ADMIN OPTION
    - Ayant le privilège DROP ANY ROLE

Création/suppression de rôles

- Création et sécurisation
  - Les rôles peuvent être crées avec un mot de passe
    - CREATE ROLE myRole;
    - CREATE ROLE myRole IDENTIFIED BY my_pwd;
  - Un mot de passe peut également être attribué à un rôle existant
    - ALTER ROLE myRole IDENTIFIED BY my_pwd;
- Suppression
  - DROP ROLE myRole;

Attribution de privilèges à un rôle

- L'attribution peut être fait par les utilisateurs
  - Ayant créé le rôle
  - **Ayant eu le rôle attribué (avec ou sans l'option ADMIN OPTION)**
  - Ayant le privilège système GRANT ANY OBJECT PRIVILEGE (pour les privilèges objets)
  - Ayant le privilège système GRANT ANY PRIVILEGE (pour les privilèges système)
  - Exemples
    - GRANT SELECT ON myTable TO myRole;
    - GRANT SELECT ON myTable TO myRole1, myRole2;
    - GRANT SELECT, UPDATE ON myTable TO myRole;

Suppression de privilèges à un rôle

- La suppression peut être faite par des utilisateurs
  - Ayant préalablement attribué le privilège au rôle
  - Ayant le privilège système GRANT ANY OBJECT PRIVILEGE
  - Exemples
    - REVOKE SELECT ON myTable FROM myRole;
    - REVOKE ALL ON myTable FROM myRole;

Attribution de rôles

- Les rôles peuvent être attribués uniquement par les utilisateurs
  - Ayant eu le rôle attribué avec l'option ADMIN OPTION
  - Le privilège général GRANT ANY ROLE
  - Exemples
    - Attribution de rôle à rôle
      - GRANT myRole TO myRole2;
      - GRANT myRole, myRole3 TO myRole2, myRole4;
      - Interdiction de cycles ! -> ~~GRANT myRole2 TO myRole;~~
    - Attribution de rôle aux utilisateurs
      - GRANT myRole TO bob, alice;
      - REVOKE myrole TO bob;

Révocation de rôles

- Un rôle peut être révoqué par les utilisateurs
  - Ayant attribué le rôle
  - Ayant eu le rôle attribué avec l'option ADMIN OPTION
  - Ayant le privilège système GRANT ANY ROLE
  - Exemples
    - REVOKE myRole TO bob;
    - REVOKE myRole TO medecins;

Le privilège GRANT ANY OBJECT PRIVILEGE

- Permet d'attribuer/révoquer des privilèges à la place d'autres utilisateurs
  - Trois utilisateurs : A, B, C
  - A est un administrateur de la BD qui possède le privilège GRANT ANY OBJECT PRIVILEGE
  - B a créé la table table_b
  - L'utilisateur A exécute

    GRANT SELECT ON B.table_b TO C;

  - Cela est comme si B a attribué le droit directement à C
  - La même opération peut être faite avec REVOKE

Où trouver l'information sur les privilèges ?

- Dans les dictionnaires d'Oracle
  - Un dictionnaire est une vue gérée automatiquement par Oracle
- Les dictionnaires portent sur tout type d'information liée à la BD
  - Utilisateurs
  - Sessions
  - Les objets
  - Les privilèges
  - Les rôles
  - Etc.

Quelques dictionnaires

- Privilèges
  - USER_COL_PRIVS : montre les colonnes des objets sur lesquelles l'utilisateur actuel est le owner/grantor/grantee
  - USER_TAB_PRIVS : montre les privilèges sur les objets où l'utilisateur est le grantee
- Rôles
  - DBA_ROLES : montre tous les rôles du système
  - USER_ROLE_PRIVS : montre les rôles attribués à l'utilisateur
- Système
  - USER_SYS_PRIVS : montre les privilèges système attribués à l'utilisateur
  - SYSTEM_PRIVILEGE_MAP : montre tous les privilèges du système

Table 4–7 Views That Display Grant Information about Privileges and Roles

View	Description
ALL_COL_PRIVS	Describes all column object grants for which the current user or PUBLIC is the object owner, grantor, or grantee
ALL_COL_PRIVS_MADE	Lists column object grants for which the current user is object owner or grantor.
ALL_COL_PRIVS_RECD	Describes column object grants for which the current user or PUBLIC is the grantee
ALL_TAB_PRIVS	Lists the grants on objects where the user or PUBLIC is the grantee
ALL_TAB_PRIVS_MADE	Lists the all object grants made by the current user or made on the objects owned by the current user.
ALL_TAB_PRIVS_RECD	Lists object grants for which the user or PUBLIC is the grantee
DBA_COL_PRIVS	Describes all column object grants in the database
DBA_TAB_PRIVS	Lists all grants on all objects in the database
DBA_ROLES	This view lists all roles that exist in the database, including secure application roles
DBA_ROLE_PRIVS	Lists roles granted to users and roles
DBA_SYS_PRIVS	Lists system privileges granted to users and roles
ROLE_ROLE_PRIVS	This view describes roles granted to other roles. Information is provided only about roles to which the user has access.

View	Description
ROLE_SYS_PRIVS	This view contains information about system privileges granted to roles. Information is provided only about roles to which the user has access.
ROLE_TAB_PRIVS	This view contains information about object privileges granted to roles. Information is provided only about roles to which the user has access.
USER_COL_PRIVS	Describes column object grants for which the current user is the object owner, grantor, or grantee
USER_COL_PRIVS_MADE	Describes column object grants for which the current user is the grantor
USER_COL_PRIVS_RECD	Describes column object grants for which the current user is the grantee
USER_ROLE_PRIVS	Lists roles granted to the current user
USER_TAB_PRIVS	Lists grants on all objects where the current user is the grantee
USER_SYS_PRIVS	Lists system privileges granted to the current user
USER_TAB_PRIVS_MADE	Lists grants on all objects owned by the current user
USER_TAB_PRIVS_RECD	Lists object grants for which the current user is the grantee
SESSION_PRIVS	Lists the privileges that are currently enabled for the user
SESSION_ROLES	Lists the roles that are currently enabled to the user

D'autres dictionnaires

- SESSION_ROLES
  - Tous les rôles actifs
- USER_SOURCE
  - Le code des procédures appartenant à l'utilisateur
- ALL_SOURCE
  - Le code des procédures appartenant à l'utilisateur ou à ceux auxquels il a accès
- DBA_SOURCE
  - Toutes les procédures de la BD
- USER_CATALOG
  - Information sur les tables, vues, séquences et synonymes de l'utilisateur
- USER_OBJECTS
  - Tout type d'objet Oracle (*clusters, database links, directories, functions, indexes, libraries, packages, java classes, abstract datatypes, resource plans, sequences, synonyms, tables, triggers, materialized views, LOBs, and views*)

VPD
Virtual Private Databases

Désavantages des rôles et des vues

- Les vues ne permettent pas un contrôle efficace par utilisateur
  - Supposez une table nommé Clients
  - Control d'accès aux tuples ou colonnes par client
  - Nécessité de faire une vue par client !
- Ces techniques n'empêchent pas les attaques par injection de code
  - Supposez l'utilisation d'une application client/serveur
  - L'interface client est écrite en PHP, JavaScript ou VisualBasic

```
Sql = "SELECT solde FROM Compte
WHERE id_client = ' " & client  & "
'AND id_compte = " & compte & " ; "
```

Le WHERE peut être remplacé par :
id_client = 'Jean' AND
id_compte = 111111 OR 1 = 1 ;
Où Jean est l'utilisateur client malveillant

Bases de données virtuelles d'Oracle

- Propose d'attacher directement aux objets de la base de données des « politiques de sécurité »
- Utilise de techniques de réécriture pour ajouter dynamiquement dans le WHERE des requêtes un prédicat basé sur de politiques de sécurité
- Permet d'utiliser les contextes d'application pour avoir de l'information sécurisé sur l'utilisateur

Exemple d'utilisation 1

- Accès restreint en base à une valeur
- Supposez
  - Une table employés d'une grande enseigne créée par le DBA où uniquement les données liées au depto 'achats' doivent être utilisées

    employes(emp_id, nom, prenom, magasin, depto)
  - Un utilisateur quelconque qui exécute

    SELECT * FROM employes

    WHERE magasin= 'Casino' ;
  - Oracle VPD automatiquement ajoute une clause dans le WHERE

    SELECT * FROM employes

    WHERE magasin= 'Casino'

    AND depto= 'achats' ;

Exemple d'utilisation 2

- Accès restreint par utilisateur **avec contexte d'application**
- Supposez
  - Une table clients créée par le DBA
    clients(client_id, adresse, compteBanquaire)
    où client_id est le nom d'utilisateur de la BD
  - Que l'utilisateur Bob exécute

    SELECT * FROM dba.clients;

  - Oracle VPD dynamiquement ajoute une clause WHERE

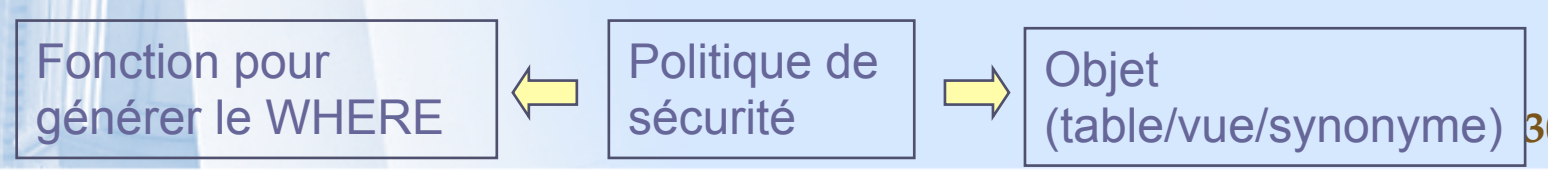
    SELECT * FROM dba.clients
    WHERE dba.client_id=SYS_CONTEXT('userenv' , 'session_user');

Avantages

- Peu importe l'application utilisée par l'utilisateur, le contrôle sera le même
- Pas besoin de faire des vues

La clause à ajouter

- Générée automatiquement selon une fonction définie par l'administrateur système (DBA)
- La fonction doit avoir
  - Paramètres : (schéma, objet concerné)
  - Résultat : clause WHERE valide de type VARCHAR2
- Les politiques de sécurité font le lien entre les fonctions et les objets



Politique de sécurité

- Permet de spécifier
  - Un nom de politique de sécurité
  - Les objets à contrôler (tables/vues/synonymes)
  - Le type de sentence à utiliser avec ces objets (SELECT, INSERT, DELETE, UPDATE, CREATE INDEX, ALTER INDEX)
    - Par défaut : SELECT, INSERT, DELETE, UPDATE
  - La fonction qui crée le where
  - Etc.
- Granularité possible au niveau de lignes et de colonnes

Exemple de fonction pour générer la clause WHERE

- Reprenez l'exemple 1
- Exemple de fonction créée par le DBA

```
CREATE OR REPLACE FUNCTION auth_emps(
    schema_var IN VARCHAR2,
    table_var IN VARCHAR2
)
RETURN VARCHAR2
IS
    return_val VARCHAR2 (400);
BEGIN
    return_val := 'DEPTO = 'achats'';
    RETURN return_val;
END auth_emps;
/
```

Paramètres

Résultat

Exemple de politique de sécurité

- Toujours avec l'exemple 1
- Exemple de sentence SQL pour définir une politique de sécurité

```
BEGIN
    DBMS_RLS.ADD_POLICY (
      object_schema => 'dba',
      object_name => 'employes',
      policy_name => 'emps_policy',
      function_schema => 'sys',
      policy_function => 'auth_emps',
      statement_types => 'select, insert, update, delete'
    );
END;
/
```

Exemple d'utilisation

- Toujours avec l'exemple 1

emp_id	nom	prenom	magasin	depto
001	scott	martin	geant	compta
002	poulet	antoine	geant	achats
003	le blanc	brian	casino	achats
004	martin	andré	casino	compta

- Si un utilisateur exécute

  SELECT * FROM employes;

  – il aura comme résultat uniquement le contenu des employés 002 et 003
- Oracle VPD ajoute la clause

  WHERE DEPTO= 'achats'

Procédures du package DBMS_RLS

Table 7–1 DBMS_RLS Procedures

Procedure	Description
For Handling Individual Policies	
DBMS_RLS.ADD_POLICY	Adds a policy to a table, view, or synonym
DBMS_RLS.ENABLE_POLICY	Enables (or disables) a policy you previously added to a table, view, or synonym
DBMS_RLS.REFRESH_POLICY	Invalidates cursors associated with nonstatic policies
DBMS_RLS.DROP_POLICY	To drop a policy from a table, view, or synonym
For Handling Grouped Policies	
DBMS_RLS.CREATE_POLICY_GROUP	Creates a policy group
DBMS_RLS.DELETE_POLICY_GROUP	Drops a policy group
DBMS_RLS.ADD_GROUPED_POLICY	Adds a policy to the specified policy group
DBMS_RLS.ENABLE_GROUPED_POLICY	Enables a policy within a group
DBMS_RLS.REFRESH_GROUPED_POLICY	Parses again the SQL statements associated with a refreshed policy
DBMS_RLS.DISABLE_GROUPED_POLICY	Disables a policy within a group
DBMS_RLS.DROP_GROUPED_POLICY	Drops a policy that is a member of the specified group
For Handling Application Contexts	
DBMS_RLS.ADD_POLICY_CONTEXT	Adds the context for the active application
DBMS_RLS.DROP_POLICY_CONTEXT	Drops the context for the application

Avant de continuer qqs concepts

- Schéma
  - Collection d'objets
  - Chaque utilisateur a un schéma et il a le même nom que le nom de l'utilisateur
  - Les objets du schéma sont des structures logiques crées par les utilisateurs contenant ou référençant leur données
- Session
  - Connexion à travers laquelle un utilisateur peut interroger la BD
  - Les sessions sont ouvertes par un programme externe à Oracle
  - Un utilisateur peut avoir plusieurs sessions et chacune est indépendante
- Package
  - Unité pouvant contenir des fonctions, procédures, variables et sentences SQL
  - Possibilité d'avoir une partie publique (appelée avec nom_package.nom_fonction) et une partie privée (utilisable uniquement à l'intérieur du package)

Contexte d'application SYS_CONTEXT

- Ensemble de couples (nom-valeur) enregistré en mémoire
- C'est comme une variable globale (array) avec de l'information accédée pendant une session
  - Permet par exemple de récupérer de manière sécurisée de l'information sur l'utilisateur (e.g., USERENV)
    - id, nom, poste, schéma, session, base de données, etc.
  - Cette information est précieuse pour identifier de manière réelle l'utilisateur
    - Elle est sécurisé car enregistrée de manière sécurisée et pas modifiable/falsifiable par l'utilisateur
    - Les applications peuvent s'en servir pour le contrôle d'accès (comparaisons dans des requêtes, vérifications, etc.)

Les variables de USERENV

Parameter	Explanation	Return Length
AUDITED_CURSORID	Returns the cursor ID of the SQL that triggered the audit	N/A
AUTHENTICATION_DATA	Authentication data	256
AUTHENTICATION_TYPE	Describes how the user was authenticated. Can be one of the following values: Database, OS, Network, or Proxy	30
BG_JOB_ID	If the session was established by an Oracle background process, this parameter will return the Job ID. Otherwise, it will return NULL.	30
CLIENT_IDENTIFIER	Returns the client identifier (global context)	64
CLIENT_INFO	User session information	64
CURRENT_SCHEMA	Returns the default schema used in the current schema	30
CURRENT_SCHEMAID	Returns the identifier of the default schema used in the current schema	30
CURRENT_SQL	Returns the SQL that triggered the audit event	64
CURRENT_USER	Name of the current user	30
CURRENT_USERID	Userid of the current user	30
DB_DOMAIN	Domain of the database from the DB_DOMAIN initialization parameter	256
DB_NAME	Name of the database from the DB_NAME initialization parameter	30
ENTRYID	Available auditing entry identifier	30
EXTERNAL_NAME	External of the database user	256
FG_JOB_ID	If the session was established by a client foreground process, this parameter will return the Job ID. Otherwise, it will return NULL.	30
GLOBAL_CONTEXT_MEMORY	The number used in the System Global Area by the globally accessed context	N/A
HOST	Name of the host machine from which the client has connected	54
INSTANCE	The identifier number of the current instance	30

Les variables de USERENV

Parameter	Explanation	Return Length
IP_ADDRESS	IP address of the machine from which the client has connected	30
ISDBA	Returns TRUE if the user has DBA privileges. Otherwise, it will return FALSE.	30
LANG	The ISO abbreviate for the language	62
LANGUAGE	The language, territory, and character of the session. In the following format: language_territory.characterset	52
NETWORK_PROTOCOL	Network protocol used	256
NLS_CALENDAR	The calendar of the current session	62
NLS_CURRENCY	The currency of the current session	62
NLS_DATE_FORMAT	The date format for the current session	62
NLS_DATE_LANGUAGE	The language used for dates	62
NLS_SORT	BINARY or the linguistic sort basis	62
NLS_TERRITORY	The territory of the current session	62
OS_USER	The OS username for the user logged in	30
PROXY_USER	The name of the user who opened the current session on behalf of SESSION_USER	30
PROXY_USERID	The identifier of the user who opened the current session on behalf of SESSION_USER	30
SESSION_USER	The database user name of the user logged in	30
SESSION_USERID	The database identifier of the user logged in	30
SESSIONID	The identifier of the auditing session	30
TERMINAL	The OS identifier of the current session	10

Gestion de contextes

- Privilèges pour créer un contexte
  - CREATE ANY CONTEXT
  - DROP ANY CONTEXT
- Le schéma d'un contexte sera toujours SYS (pas celui de l'administrateur système créant le contexte)
- Création d'un contexte

  CREATE CONTEXT namespace USING package-name;
  - Le namespace est le nom du contexte
  - Le package doit contenir une procédure pour l'initialisation des variable
  - Le package/procédure peut être déclaré après la création
- Suppression d'un contexte

  DROP CONTEXT namespace

Initialiser les variables du contexte

- Utilisation de la fonction SET_CONTEXT du package DBMS_SESSION
- Besoin d'avoir le privilège EXECUTE sur le package DBMS_SESSION
- Fonction à utiliser dans la procédure d'initialisation des variables du contexte

```
DBMS_SESSION.SET_CONTEXT(
    Namespace VARCHAR2,
    Attribut VARCHAR2,
    Value VARCHAR2
 );
```

- Besoin d'un trigger de type LOGON pour lancer l'initialisation lors de la connexion de chaque utilisateur

Interroger les variables du contexte

- La fonction SYS_CONTEXT permet de récupérer l'information du contexte

  SYS_CONTEXT(

  Namespace,

  Paramètres

)

- Par défaut le namespace USERENV est un contexte initialisé avec l'information de l'utilisateur
  - SYS_CONTEXT('userenv', 'session_user');

Exemple de contexte -création

- Exemple simple de création, initialisation et interrogation d'un contexte
- Supposez
  - La table employes créée sous l'utilisateur scott
  - Le besoin de récupérer le nom et le depto de chaque utilisateur
  - L'utilisateur système sysadmin crée le contexte qui enregistrera ces informations

1. CREATE OR REPLACE CONTEXT emp_ctx USING set_emp_ctx_pkg;

Exemple de contexte -initialisation

```
1.  CREATE OR REPLACE PACKAGE set_emp_ctx_pkg IS
2.   PROCEDURE set_emp;
3.  END;
4.  /

5.  CREATE OR REPLACE PACKAGE BODY set_emp_ctx_pkg IS
6.   PROCEDURE set_emp
7.  IS
8.   nom_ctx VARCHAR2(20);
9.   depto_ctx VARCHAR2(20);
10. BEGIN
11. nom_ctx:=SYS_CONTEXT('USERENV', 'SESSION_USER');
12. DBMS_SESSION.SET_CONTEXT('emp_ctx', 'nom', nom_ctx);
13.
14. SELECT depto INTO DEPTO_CTX
15. FROM scott.employes
16. WHERE UPPER(NOM)=nom_ctx;
17. DBMS_SESSION.SET_CONTEXT('emp_ctx', 'depto', depto_ctx);
18. END;
19. /
```

Trigger de type LOGON

```
1. CREATE OR REPLACE TRIGGER set_emp_trigger
   AFTER LOGON ON DATABASE
2.     BEGIN
3.         sysadmin.set_emp_ctx_pkg.set_emp;
4.     END;
5. /
```

Exemple de contexte -interrogation

1. CONNECT scott/scott
2. SELECT SYS_CONTEXT('emp_ctx','nom') FROM DUAL;

Cela donne : 'SCOTT'

1. SELECT SYS_CONTEXT('emp_ctx','depto') FROM DUAL;

Cela donne : 'compta'

Contexte d'application dans les VPD

- La fonction qui crée la clause WHERE peut utiliser un contexte d'application
- Ainsi
  - Lorsqu'un utilisateur se connecte à la base le contexte d'application enregistre plusieurs informations
  - Ensuite une politique de sécurité restreint l'accès à la BD en base aux informations du contexte

Exemple de fonction pour générer la clause WHERE avec utilisation de contexte d'application

- Utilisation du contexte **emp_ctx** dans **auth_emps**

```
CREATE OR REPLACE FUNCTION auth_emps(
    schema_var IN VARCHAR2,
    table_var IN VARCHAR2)
RETURN VARCHAR2
IS
    return_val VARCHAR2 (400);
BEGIN
    return_val := 'DEPTO = SYS_CONTEXT("emp_ctx", "depto") ' ;
    RETURN return_val;
END auth_emps;
/
```

Exemple d'application

emp_id	nom	prenom	magasin	depto
001	scott	martin	geant	compta
002	poulet	antoine	geant	achats
003	le blanc	brian	casino	achats
004	martin	andré	casino	compta

- Si l'utilisateur scott exécute
  - select * from employes;
  - Il aura comme résultat les tuples 001 et 004
  - Oracle VPD ajoute la clause WHERE DEPTO= 'compta'
- Si l'utilisateur poulet exécute
  - select * from employes;
  - Il aura comme résultat les tuples 002 et 003
  - Oracle VPD ajoute la clause WHERE DEPTO= 'achats'

Multiples politiques

- Plusieurs politiques de sécurité peuvent être attachées à un même objet
- Tous les prédicats générés sont inclus dans la nouvelle requête séparés par AND
- Attention aux politiques sémantiquement contradictoires
- Possibilité de faire des groupes de politiques

VPD par colonne

- Jusqu'ici on a vu comment restreindre les tuples qui résultent d'une requête
  - Row-level security
- Possibilité de restreindre, en plus, les colonnes
  - Column-level security

VPD par colonne

- Action complémentaire à la sécurité par tuples
- Dans la définition de la politique de sécurité, utiliser les paramètres

1. sec_relevant_cols
   - La fonction qui crée la clause WHERE sera appelée uniquement si les colonnes introduites ici sont référencées dans une requête
   - **Les tuples** qui ne correspondent pas à la politique de sécurité ne sont pas montrées dans le résultat
2. sec_relevant_cols_opt
   - Pareil que sec_relevant_cols sauf que les tuples qui ne correspondent pas à la politique de sécurité sont montrées et uniquement les valeurs des colonnes sensibles sont données comme NULL

Exemple 1 de politique avec VPD par colonne

```
BEGIN
   DBMS_RLS.ADD_POLICY (
     object_schema => ‘dba',
     object_name => ‘employes',
     policy_name => ‘emps_policy',
     function_schema => 'sys',
     policy_function => 'auth_emps',
     statement_types => 'select, insert, update, delete',
     sec_relevant_cols=> 'magasin'
   );
END;
/
```

Exemple d'utilisation 1

emp_id	nom	prenom	magasin	depto
001	scott	martin	geant	compta
002	poulet	antoine	geant	achats
003	le blanc	brian	casino	achats
004	martin	andré	casino	compta

- Supposez la fonction du slide 48
- Supposez que l'utilisateur scott exécute
  - Select emp_id, nom, prenom from employes
  - Il aura comme résultat tous les tuples de la table employes
  - La clause WHERE n'est pas ajoutée car la colonne « magasin » n'est pas référencée dans la requête
- Si scott exécute
  - Select * from employes
  - Il aura comme résultat les tuples 001 et 002 car la clause WHERE DEPTO='compta' est ajoutée du fait que la colonne magasin est référencée dans la requête

Exemple 2 de politique avec VPD par colonne

- On peut en plus initialiser le paramètre sec_relevant_cols_opt afin d'avoir dans le résulta des requêtes tous les tuples mais avec de valeurs nulles dans les colonnes sensibles

```
BEGIN
    DBMS_RLS.ADD_POLICY (
      object_schema => 'dba',
      object_name => 'employes',
      policy_name => 'emps_policy',
      function_schema => 'sys',
      policy_function => 'auth_emps',
      statement_types => 'select, insert, update, delete',
      sec_relevant_cols=> 'magasin ',
      sec_relevant_cols_opt=>dbms_rls.ALL_ROWS
    );
END;
/
```

Exemple d'utilisation 2

- Si l'utilisateur scott exécute
  - Select * from employes
  - Il aura comme résultat tous les tuples mais avec de valeurs nulles sur les tuples des employés qui n'appartiennent pas à son département

emp_id	nom	prenom	magasin	depto
001	scott	martin	geant	compta
002	poulet	antoine	NULL	achats
003	le blanc	brian	NULL	achats
004	martin	andré	casino	compta

- Quel est le résultat si l'utilisateur poulet exécute la même requête ?

Debug de VPD

- Type de problème
  - Erreurs dans les fonctions créant le WHERE
    - *ORA-28110: policy function or package has error*
    - Une solution-> vérifier que les tables accédées dans la fonction existent
  - SQL incorrect
    - *ORA-28113: policy predicate has error*
    - La clause WHERE une fois ajoutée à la requête principale rend la requête invalide
    - Une solution-> vérifier la fonction qui génère la clause WHERE (quottes, types, nom des colonnes, etc.)
    - Une autre solution-> regarder les « trace file ». Besoin de droit de lecture sur le répertoire où les trace file d'Oracle sont enregistrés ☹

Debug de VPD

- Type de problème
  - Contexte d'application vide
    - Le contexte d'application et une politique de sécurité utilisent une même table
    - Une solution-> utiliser une table dédiée au contexte d'application qui n'ait pas de politique de sécurité

Dictionnaires utiles

- ALL_POLICIES
- USER_POLICIES
- ALL_POLICY_GROUPS
- USER_POLICY_GROUPS
- ALL_POLICY_CONTEXTS
- USER_POLICY_CONTEXTS
- **V$VPD_POLICY**

Conclusion

- VPD permet un contrôle d'accès au niveau de tuples grâce aux politiques de sécurité
- Les politiques de sécurité peuvent être appliquées au niveau des
  - Tables/vues
  - Colonnes (rec_relevant_cols)
- L'utilisation des contexte d'application contribuent à améliorer les performances

Bibliographie

- Oracle Database Security Guide 10g Release. Décembre 2003.
  - http://www.stanford.edu/dept/itss/docs/oracle/10g/network.101/b10773.pdf
- Oracle 10g The Complete Reference, 2004.
  - http://www.sciences.univ-nantes.fr/info/perso/permanents/Patricia.Serrano-Alvarado/Oracle10gTheReference.pdf
- Chapitre du livre Effective Oracle Database 10g Security by Design
  - http://www.devshed.com/c/a/Oracle/RowLevel-Security-with-Virtual-Private-Database/
- Cours sur le web
  - http://www.sagecomputing.com.au/papers_presentations/vpd.pdf
  - http://www.nyoug.org/Presentations/2003/fgananda.pdf

Cont.

- Site web pour RBAC
  - http://csrc.nist.gov/groups/SNS/rbac/