



# *Administración de Bases de Datos*

*(Ingeniería Informática)*

## Tema 2. CONTROL DE LA SEGURIDAD EN UNA BASE DE DATOS

# Seguridad para las BD

- ▶ Las 4 As: Autenticación, Autorización, Acceso y Auditoría
- ▶ Autenticación
  - ▶ Passwords
  - ▶ Multi-factor authentication (smartcards can be used).
- ▶ Autorización y Acceso
  - ▶ Privilegios
  - ▶ Oracle Label Security, Oracle Data Vault
- ▶ Auditoría (lo veremos más adelante en el curso)
  - ▶ Capturar quien accede a qué, de manera que podamos validar las políticas de seguridad impuestas.

# Tareas de seguridad

- ▶ Asegurarse de que la instalación y configuración de la BD es segura
- ▶ Gestionar los aspectos de seguridad de las cuentas de usuario:
  - ▶ desarrollo de políticas de contraseña segura,
  - ▶ creación y asignación de roles, privilegios de administración, de sistema, etc.
  - ▶ restricción del acceso a los datos sólo a los usuarios apropiados,
  - ▶ etc.
- ▶ Asegurarse de que las conexiones de red son seguras
- ▶ Cifrado de datos sensibles
- ▶ Asegurar que la base de datos no tiene vulnerabilidades de seguridad y está protegida contra intrusos
- ▶ Decidir qué componentes de base de datos auditar y cómo
- ▶ Descarga e instalación de parches de seguridad

# Parámetros de seguridad

```
ALTER SYSTEM SET var=valor  
ALTER SYSTEM RESET
```

**Table 2-1** *Default Security Settings for Initialization and Profile Parameters*

Setting	10g Default	18c Default
AUDIT_TRAIL	NONE	DB
O7_DICTIONARY_ACCESSIBILITY	FALSE	FALSE
PASSWORD_GRACE_TIME	UNLIMITED	7
PASSWORD_LOCK_TIME	UNLIMITED	1
FAILED_LOGIN_ATTEMPTS	10	10
PASSWORD_LIFE_TIME	UNLIMITED	180
PASSWORD_REUSE_MAX	UNLIMITED	UNLIMITED
PASSWORD_REUSE_TIME	UNLIMITED	UNLIMITED
REMOTE_OS_ROLES	FALSE	FALSE

# Gestión de Usuarios. El SO

- ▶ Para conectarse a Oracle es necesario un usuario y un modo de identificación
- ▶ Cada persona/proceso que se conecte al servidor debe identificarse
- ▶ La identificación puede ser gestionada por Oracle o por el SO
- ▶ Por el SO

- ▶ En Windows grupos creados por defecto
- ▶ Nos conectamos sin necesidad de proporcionar credenciales: `sqlplus /`
- ▶ Para ello, crear usuario con opción external. Ejm:

Create user "OPS\$HOST\USUARIO" identified externally;

- ▶ OPS\$ es un prefijo que Oracle precisa para esos usuarios. Puede modificarse modificando el parámetro `OS_AUTHENT_PREFIX`
- ▶ usuario debe existir en el SO.
- ▶ WARNING: Cada SO impone sus reglas de case-sensitive.
- ▶ Tiene sus ventajas y desventajas.

# Gestión de Usuarios. Oracle

- ▶ Para conectarse a Oracle es necesario un usuario y un modo de identificación
- ▶ Cada persona/proceso que se conecte al servidor debe identificarse
- ▶ Los Perfiles de Usuario y los Roles facilitan la creación de usuarios
- ▶ Perfiles de Usuario:
  - ▶ Simplifican la gestión de usuarios. Ej.:
  - ▶ Se pueden fijar restricciones de password y de recursos.

`Resource_limit=true`

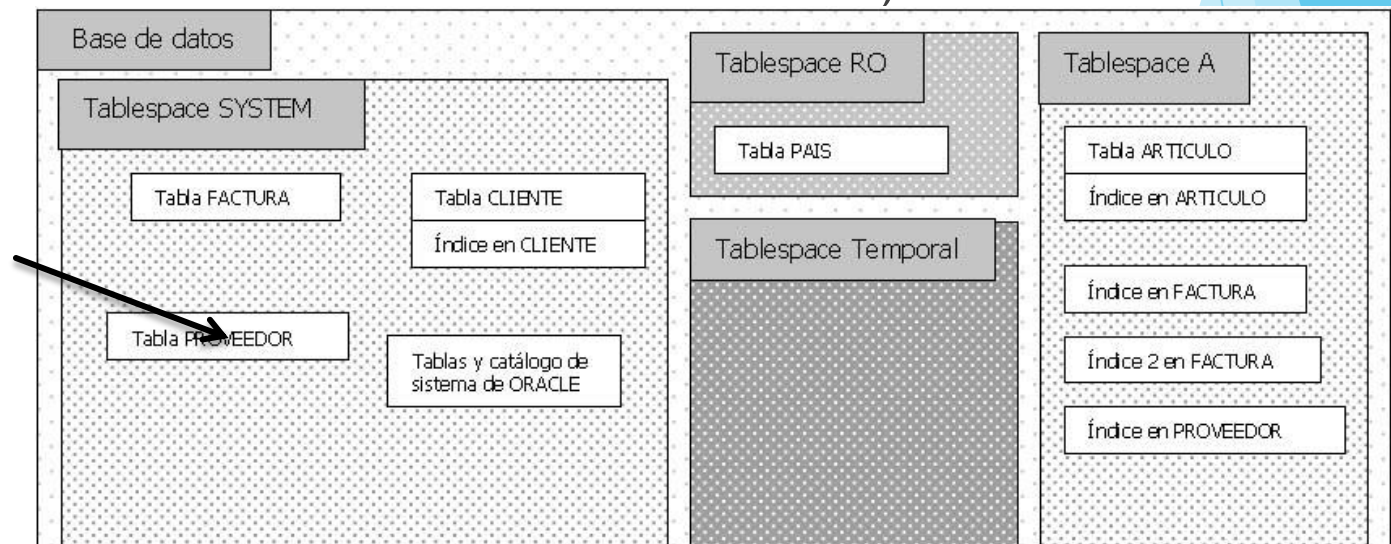
```
CREATE PROFILE Perfil_1 LIMIT
SESSIONS_PER_USER 3      -- Máximo núm. de sesiones para ese usuario.
CONNECT_TIME UNLIMITED  -- Duración máxima de la conexión.
IDLE_TIME 10             -- Minutos de tiempo muerto en una sesión.
FAILED_LOGIN_ATTEMPTS 4  -- n° máximo de intentos para bloquear cuenta.
PASSWORD_LIFE_TIME 90    -- N° de días de expiración de la password.
PASSWORD_GRACE_TIME 3;   -- Periodo de gracia después de los 90 días.
```

# Gestión de Usuarios

## ► Antes de crear Usuarios...

- Oracle hace una división lógica de los datos de una base de datos en TABLESPACES.
- Cuando se crea la base de datos se define un TABLESPACE por defecto para todos los usuarios (USERS)
- Cuando se crea el usuario, se puede cambiar el TABLESPACE por defecto para ese usuario
- Cuando se crea una tabla, se puede cambiar el TABLESPACE donde se crea (si el usuario tiene cuota en ese TABLESPACE)

No se aconseja  
usar el tablespace  
SYSTEM para  
usuarios, ya que  
es poco seguro



# Tablespaces

Nombre	Tamaño	Espacio Libre	Usado (%)	Ampliació...	Tamaño Má...	Estado	Tipo
SYSAUX	600MB	38MB	93,7	✓	Ilimitado	●	📁
SYSTEM	840MB	640KB	99,9	✓	Ilimitado	●	📁
TEMP	129MB	128MB	,8	✓	Ilimitado	●	📁
UNDOTBS1	65MB	37MB	43,1	✓	Ilimitado	●	📁
USERS	5MB	4MB	21,3	✓	Ilimitado	●	📁

```
SELECT * from DBA_TABLESPACES; -- Comprueba los ts existentes
create tablespace nombre datafile 'nombrefichero.dbf' size
10M autoextend on; /*Crea un ts con un fichero de tamaño 10M
autoextensible. Para que un usuario pueda usarlo debe asignársele
una cuota de uso. */
```



# Gestión de Usuarios

## ▶ Crear Usuarios: La única cláusula obligatoria es IDENTIFIED:

- ▶ Las cuotas máximas se establecen por TABLESPACE y se pueden especificar en Megabytes (M), en Gigabytes (G), etc. o indicar que no hay límite (**UNLIMITED**).

```
CREATE USER Pepe IDENTIFIED BY Clave_Pepe
  TEMPORARY TABLESPACE temp_ts -- Tablespace para los seg. temporales.
  DEFAULT TABLESPACE data_ts   -- Tabl. por defecto para sus objetos.
  QUOTA 100M ON test_ts         -- Máximo espacio en ese Tablespace.
  QUOTA 500K ON data_ts
  PROFILE Perfil_1;             -- Asigna el Perfil_1 al usuario.
```

- ▶ Un usuario recién creado no tiene ningún privilegio. Se deben conceder algunos permisos. Como mínimo los necesarios para conectarse a la BD y crear objetos.
- ▶ **Modificar un usuario**: ALTER USER (con similar formato).
- ▶ **Borrar un usuario**: DROP USER Onieva CASCADE;
  - ▶ Si el esquema del usuario no está vacío, se debe poner CASCADE: para borrar todos sus objetos.

Users vs Schemas

# Roles

```
CREATE ROLE nombre_role
[NOT IDENTIFIED
| IDENTIFIED {BY password |
              USING [schema.]package |
              EXTERNALLY |
              GLOBALLY}}];
```

- ▶ Primero se crea el rol vacío y luego se le asignan privilegios y/o otros roles.
- ▶ Por defecto es no identificado (lo más normal) aunque se puede imponer identificación adicional que ha de aportar el usuario cuando lo vaya a usar
- ▶ Para poder utilizar un rol identificado por contraseña se debe primero utilizar la instrucción `SET role nombre_role identified by password`

Ejemplo: **CREATE ROLE *Gestor* IDENTIFIED BY 123;**  
**CREATE ROLE *USUARIO\_NORMAL*;**

# Gestión de Usuarios

- ▶ Oracle tiene por defecto tres tipos usuarios que conviene conocer:
  - ▶ Usuarios administradores: **SYS, SYSTEM.**
  - ▶ Usuarios generales: Son los usuarios creados por los administradores. E.g. ORACLE crea algunos (cuyas cuentas están bloqueadas inicialmente) por defecto de ejemplo.
  - ▶ Usuarios internos: Utilizados por procesos externos y en background de ORACLE. También tienen sus cuentas bloqueadas (es decir, el login manual con estas cuentas no es posible) y no deben ser NUNCA eliminadas del sistema
- ▶ Usuarios administradores principales
  - ▶ **SYS:** Tiene todos los privilegios y no debería utilizarse CASI nunca, ni mucho menos crearse objetos ni modificarse en su esquema. Contiene todas las tablas y vistas del diccionario de datos.
  - ▶ **SYSTEM:** Permite realizar todas las tareas administrativas excepto *backup y recovery y upgrade* de la base de datos. Tampoco puede hacer un shutdown y startup de la BD.

# Permisos: GRANT y REVOKE

- ▶ POLITICA LP (LEAST PRIVILEGES)
- ▶ Permisos: Concederlos (GRANT) y Revocarlos (REVOKE).
  - ▶ Pueden usar estas órdenes los propietarios (*owner*) de los objetos (tablas, vistas...) o el DBA (administrador de la BD) o aquellos usuarios que han recibido el privilegio con *GRANTABLE* activado.
  - ▶ Permisos sobre Objetos: Hay que especificar el Tipo de Permiso y el Objeto al que se aplicará:
    - ▶ Tipos de Permisos: **SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES, INDEX** (para crear índices) y **EXECUTE** (para procedimientos, funciones, etc.).
      - ▶ **UPDATE** se puede dar sobre un atributo o varios (entre paréntesis). `GRANT UPDATE ON Cuentas(Saldo) TO Cajero`
      - ▶ **ALL**: Especifica TODOS los Permisos posibles sobre el Objeto.
        - ▶ Por defecto, el propietario de un objeto tiene todos los permisos posibles sobre él.
    - ▶ Tipos de Objetos: Tablas, Vistas, Secuencias, Procedimientos, Funciones, Vistas Materializadas...

# Permisos: GRANT y REVOKE

- ▶ **Permisos del Sistema**: Son permisos que no se definen para un objeto concreto.
  - ▶ Ejemplos:
    - ▶ **CREATE TABLE**: Para crear tablas propias.
    - ▶ **CREATE ANY TABLE**: Para crear tablas en otros esquemas o usuarios.
    - ▶ **ALTER ANY TABLE**: Para modificar tablas. Sin **ANY** no existe porque es absurdo.
    - ▶ **DROP ANY TABLE**: Para borrar tablas de cualquier esquema.
    - ▶ **DELETE ANY TABLE**: Borrar filas de tablas/vistas de cualquier esquema.
    - ▶ **INSERT ANY TABLE**: Insertar filas en tablas/vistas de cualquier esquema.
    - ▶ **UPDATE ANY TABLE**: Actualizar tablas/vistas de cualquier esquema.
    - ▶ **SELECT ANY TABLE**: Consultar tablas/vistas de cualquier esquema.
    - ▶ **EXECUTE ANY PROCEDURE**: Para ejecutar procedimientos y funciones.
  - ▶ **ALL PRIVILEGES**: Especifica TODOS los Permisos del Sistema. ¡NO USAR NUNCA!
  - ▶ ¿Qué tienen en común todos desde un punto de vista de Seguridad?

# Permisos: GRANT y REVOKE

- ▶ Formato:

```
GRANT <Lista_Permisos> [ON <Objeto>]  
TO <Usuarios> [WITH {GRANT|ADMIN} OPTION] ;
```

  - ▶ Con Permisos del Sistema se elimina la cláusula ON.
  - ▶ <Usuarios>: Pueden ser también un Rol (al que se asigna ese permiso). Si se asignan los permisos a PUBLIC se asignan a todos los usuarios.
  - ▶ WITH GRANT OPTION: Permite al <Usuario> conceder dicho privilegio a otro usuario. Con permisos del sistema usar WITH ADMIN OPTION.
  - ▶ Con algunos permisos se puede especificar la columna: GRANT UPDATE (col) ON Table
  - ▶ Existen unos Roles Predefinidos que Oracle crea automáticamente:
    - ▶ DBA: Contiene todos los privilegios de la BD y debe concederse con cuidado.
    - ▶ CONNECT: Para conectarse
    - ▶ RESOURCE: Para crear objetos (tablas, triggers, secuencias, procedimientos...).
- ▶ Formato:

```
REVOKE <Lista_Permisos> [ON <Objeto>]  
FROM <Usuario> ;
```
- ▶ El destinatario puede ser un rol o un usuario. Se le pueden asignar permisos o roles.

# Permisos: GRANT y REVOKE

## ▶ Ejs.:

- ▶ `GRANT CONNECT, Rol_Programador TO Araujo;`
- ▶ `GRANT CREATE USER, ALTER USER, DROP USER TO Nous, Zeus WITH ADMIN OPTION;`
- ▶ `GRANT CREATE ANY PROCEDURE, CREATE TRIGGERS TO Apolonio;`
- ▶ `GRANT ALL ON Empleados TO Casandra WITH GRANT OPTION;`
- ▶ `GRANT SELECT ON Empleados TO PUBLIC;`
- ▶ `GRANT REFERENCES (DNI), UPDATE (Salario, Cta_Banco) ON Empleados TO Rol_Nominas;`
- ▶ `REVOKE CREATE SESSION FROM Usuario1, Usuario2;`
- ▶ Los roles asignados tienen que ser explícitamente activados por el usuario receptor (o reiniciar sesión si es un rol por defecto)
- ▶ ¿Cómo podemos dar permiso a un usuario para que pueda leer unas columnas de una tabla sin que pueda leer de todas?

# Permisos en Vistas y Procedimientos

- ▶ Si el propietario le ha concedido a un usuario permiso de SELECT e INSERT sobre una vista V y esta vista se define sobre una tabla T sobre la que el usuario no tiene ningún permiso:
  - ▶ ¿Qué ocurre cuando el usuario intenta leer de V?
  - ▶ ¿Qué ocurre cuando el usuario intenta escribir en V?
- ▶ A un usuario se le da permiso de EXECUTE sobre un procedimiento P. El procedimiento escribe y lee de la tabla T anterior
  - ▶ ¿Qué ocurre cuando el usuario ejecuta P?
- ▶ Un usuario ejecuta un procedimiento P1. Este procedimiento ejecuta una instrucción EXECUTE IMMEDIATE que crea una tabla (o cualquier otra instrucción DDL)
  - ▶ ¿En qué casos se permite realizar la instrucción? ¿Por qué?



***Lo veremos más adelante***



### VIEW

► **VISTA:** Es una **tabla virtual** cuyas tuplas derivan de otras tablas (que pueden ser tablas base o también otras vistas).

- Sus tuplas no se almacenan sino que se calculan a partir de las tablas de las que dependa.

► **Formato:** `CREATE [OR REPLACE] [[NO] FORCE] VIEW <NombreV> [( <Lista_Atrbs> )] AS ( <Subconsulta> ) [WITH READ ONLY];`

- Crea la vista <NombreV>, asociada a la subconsulta especificada.
- La **lista de atributos** es el nombre de los atributos de la vista: Por defecto toma los nombres de los atributos de la subconsulta. Son obligatorios si los atributos son calculados (funciones de grupo...).
- **OR REPLACE:** Permite modificar una vista ya existente sin necesidad de borrarla antes.
- **WITH READ ONLY:** Indica que no se permitirán borrados, inserciones o actualizaciones en la vista.
- **FORCE:** Fuerza a que la vista se cree aunque no existan los objetos que se usan en ella (tablas, otras vistas...) o no se tengan privilegios suficientes. Esas condiciones serán necesarias para usar la vista. La opción contraria es **NO FORCE**, que es la opción por defecto.

## ► Ejemplos:

- **CREATE OR REPLACE VIEW** SumiNombres  
**AS (SELECT** NombreS, NombreP **FROM** Suministros SP,  
Suministrador S, Pieza P **WHERE** SP.S#=S.S# **AND** SP.P#=P.P#);
- **CREATE OR REPLACE VIEW** Cantidad (NombreS, NumPiezas)  
**AS (SELECT** NombreS, **COUNT (\*)** **FROM** Suministros SP,  
Suministrador S **WHERE** SP.P#=S.P#  
**GROUP BY** NombreS);

## ► Observaciones:

- Las vistas pueden consultarse como si fueran tablas.
- Una vista está **siempre actualizada** (*up to date*): Si se modifican las tablas de las que depende, la vista reflejará esos cambios.
- Para que la vista **NO se actualice**, no hay que crear una vista, sino una “instantánea”, “foto” o vista materializada (*materialized view*) de la BD (con **CREATE MATERIALIZED VIEW**).
- Para borrar una vista que ya no es útil: **DROP VIEW** <NombreV>;

## ► Escritura en una vista:

- Es posible si es una vista sobre una sola tabla, sin funciones de agregación e incluye todos los campos declarados **NOT NULL** (primary keys, etc.) de la tabla original.
- En general, se consideran vistas **NO actualizables** si están definidas sobre varias tablas (en una reunión) o si usan agrupamiento y/o funciones de agregación.
- Se suele utilizar un *trigger* **INSTEAD OF** para definir el comportamiento

# Objetos del Esquema: VISTAS

## ► Utilidades de las Vistas:

- **Seguridad:** Restringir el acceso a ciertas *filas/columnas* de una tabla.
- **Esconder la complejidad de los datos:** Una vista puede incluir una operación de reunión, muchas tablas...
  - Simplificar sentencias al usuario: Evitan tener que conocer el nombre de todas las tablas y simplifican consultas habituales complejas.
  - Ejemplo: Las vistas del diccionario de datos.
- **Presentar los datos desde otra perspectiva:** Cambiando los nombres de atributos, introduciendo operaciones...
- **Que las aplicaciones sean independientes a cambios en las tablas base.**
- **Efectuar consultas que no se pueden hacer sin vistas:** Como hacer una reunión entre una tabla y una consulta con **GROUP BY** (o con una **UNION**).

# Objetos del Esquema: SINÓNIMOS

- ▶ **SINÓNIMOS**: Un sinónimo es un nombre alternativo, o alias, de una tabla, vista, *vista materializada*, secuencia, procedimiento, función o paquete.
  - ▶ No requieren almacenar más que su definición en el diccionario de datos.
  - ▶ Los sinónimos tienen una doble función: conveniencia y seguridad (ocultando el nombre y el propietario de un objeto, y su localización en BD distribuidas).
  - ▶ Un sinónimo público es aquel cuyo propietario es el grupo de usuarios **PUBLIC** y todos los usuarios pueden acceder a él.
  - ▶ Un sinónimo privado pertenece al subesquema de un determinado usuario que puede controlar el uso del mismo.
  - ▶ **Formato:** `CREATE [PUBLIC] SYNONYM <Nombre> FOR <Objeto>;`
  - ▶ **Ejemplos:**
    - ▶ `CREATE SYNONYM pventa FOR Paco.Proyect_Venta;`
      - ▶ Crea un sinónimo privado para una tabla del esquema de Paco.
    - ▶ `CREATE PUBLIC SYNONYM Prod FOR Scott.Prod@Ventas;`
      - ▶ Crea un sinónimo público para una tabla de Scott en una BD remota llamada Ventas (DBLINK).
  - ▶ Si un sinónimo público para una tabla tiene el mismo nombre que una tabla de un usuario, para ese usuario no tendrá efecto el sinónimo.

# Objetos del Esquema: SNAPSHOTS

## ► VISTAS MATERIALIZADAS (*materialized views, snapshots*):

- Calcular y almacenar agregaciones (sumas, medias...), reuniones o, en síntesis, consultas lentas.
- En entornos distribuidos, pueden usarse para duplicar los datos localmente, evitando accesos lejanos y lentos.
- Pueden refrescarse manualmente, a intervalos regulares de tiempo o cuando termina una transacción sobre las tablas base (*master tables*).
- Para crearlas se debe tener los permisos **CREATE MATERIALIZED VIEW** y **CREATE TABLE**, a parte de los permisos de acceso a las tablas base y de espacio suficiente en el *tablespace* correspondiente.

► Formato: **CREATE MATERIALIZED VIEW** <Nombre> <Cláusulas>  
**AS** <subconsulta>;

- <Cláusulas> incluye multitud de características, entre las que destacan:
  - **BUILD [IMMEDIATE | DEFERRED]**: Indica que la vista materializada será llenada de forma inmediata (por defecto), o lo será en la primera operación de **REFRESH** (refresco), el cual será un refresco completo. Hasta entonces no podrá ser usada.

# Objetos del Esquema: SNAPSHOTS

- ▶ [REFRESH <Opciones> | NEVER REFRESH]: Establece las opciones para refrescar automáticamente o no los datos. Estas opciones pueden ser:

- ▶ FAST: El refresco rápido se lleva a cabo usando los cambios hechos sobre las tablas base. Esos cambios se almacenan en una tabla especial asociada a cada tabla base que se crea con:

```
CREATE MATERIALIZED VIEW LOG ON <Tabla>  
INCLUDING NEW VALUES;.
```

- ▶ Este M.V. LOG almacena los cambios efectuados sobre una tabla y se actualiza con cada comando DML sobre esa tabla.
- ▶ Un mismo M.V. LOG sobre cierta tabla puede servir para actualizar todas las M.V. que usen esa tabla.

- ▶ No todas las subconsultas pueden beneficiarse de este tipo de refresco.

- ▶ INCLUDING NEW VALUES: Especifica que en la tabla del LOG se incluyan los viejos y los nuevos valores. La opción por defecto es EXCLUDING NEW VALUES.
- ▶ COMPLETE: El refresco completo consiste en rehacer la consulta.
- ▶ FORCE: Cuando haya que refrescar, se ejecutará un refresco FAST si es posible. En otro caso se hará un refresco COMPLETE. Esta es la opción por defecto.

# Objetos del Esquema: SNAPSHOTS

## ► Opciones de Cuando Refrescar:

- ON [COMMIT | DEMAND]: Especifica si el refresco se ejecutará al final de cada transacción que modifique una tabla base, o bajo petición explícita (ejecutando un procedimiento del paquete DBMS\_MVIEW). Esta última es la opción por defecto.
- START WITH <Fecha>: Especifica la fecha del primer refresco.
- NEXT <Fecha>: Fecha para calcular el intervalo entre refrescos automáticos (con respecto a START WITH).



# Objetos del Esquema: MVs

## ► Ejemplos:

- Usa dos tablas del esquema Ventas en una BD remota:

```
CREATE MATERIALIZED VIEW Ventas.Clientes_Recientes AS  
SELECT * FROM Ventas.Clientes@dbs1.uma.es C  
WHERE EXISTS (SELECT * FROM Ventas.Ordenes@dbs1.uma.es O  
              WHERE C.DNI = O.DNI_Cliente);
```

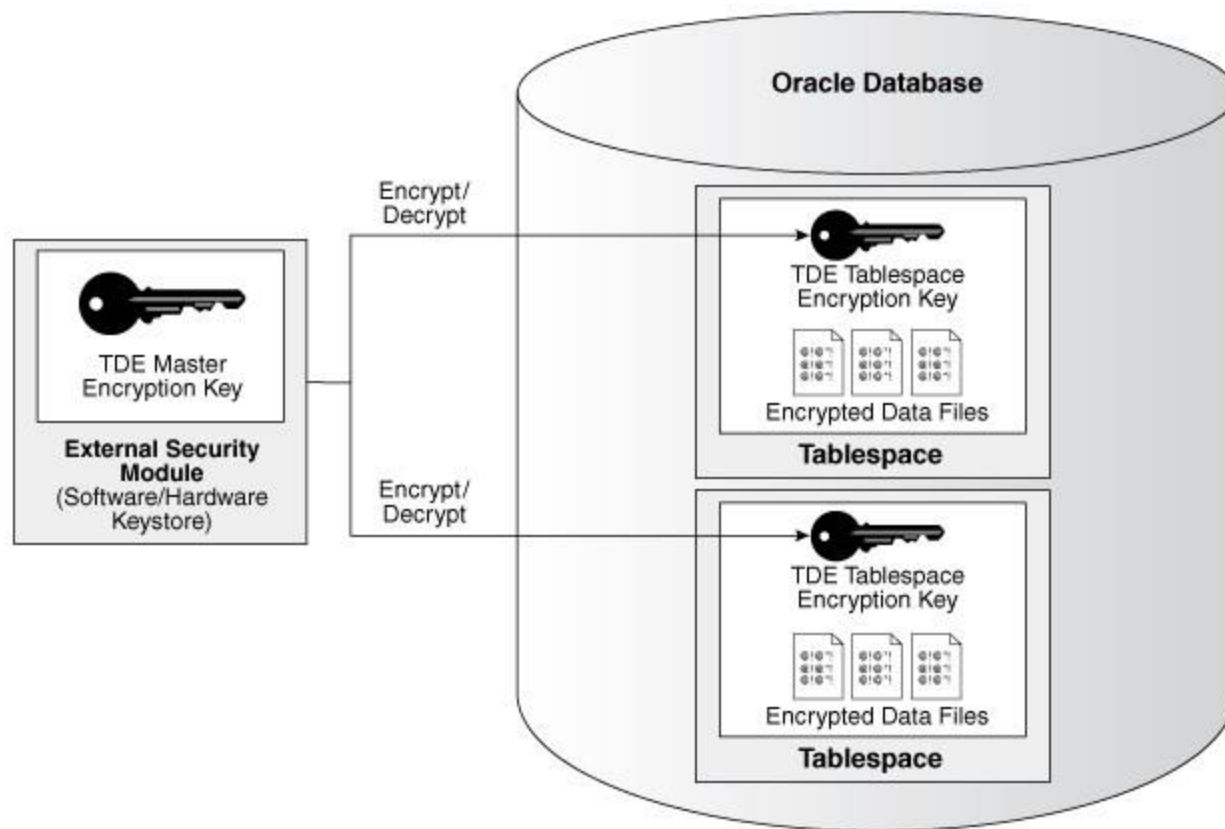
- Cada 7 días, a partir de hoy obtengo la unión de 2 tablas de sendos usuarios en BD remotas:

```
CREATE MATERIALIZED VIEW sales_emp TABLESPACE Mi_Tablespace  
REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 7  
AS SELECT * FROM Patricia.Cosas@Granada UNION  
SELECT * FROM Miriam.Cosas@Malaga;
```

# Seguridad de datos avanzada

- ▶ **Encriptación a medida.** Se utiliza el paquete DBMS\_CRYPTO.
- ▶ **Encriptación transparente de datos (TDE).** Se pueden encriptar columnas o tablespace enteros.
  - ▶ Especialmente útil para datos sensibles
  - ▶ Es transparente y sólo afecta a como son almacenados los datos
- ▶ **Virtual Private Database.** Se restringe el acceso a nivel de *fila* y *columna* introduciendo clausulas WHERE a todas las sentencias SQL de forma automática mediante una policy function (PL/SQL). Se utiliza cuando ya se tienen datos para poder asignar los permisos
- ▶ **Oracle Label Security.** Se protegen las tablas a nivel de *fila* asignando etiquetas a cada fila. Los usuarios se autorizan teniendo en cuenta estas etiquetas. Se utilizan dando niveles de “sensibilidad” a las filas de las tablas
- ▶ **Oracle Database Vault.** Permite restringir el acceso a los datos incluso a los administradores.

# TDE



► Precisa de una configuración inicial. Los pasos son los siguientes:

1. Establecer el directorio dónde se va a almacenar el keystore (si es de tipo FILE) mediante el parámetro de inicialización estático `WALLET_ROOT`.

```
ALTER SYSTEM SET "WALLET_ROOT"='/path/to/directory' scope=SPFILE;
```

-- IMPORTANTE: El path utilizado deber existir (sino, crearlo) y ser accesible por el servicio de Oracle (el servicio debe tener permisos). Depende de cada instalación. Por ejm.  
C:\Users\alumnos\Oracle\wallet en las máquinas virtuales de este curso.

-- Como se trata de un parámetro estático tendremos que reiniciar la instancia. La forma más rápida será reiniciar el servicio de Windows.

2. Establecer el tipo de Keystore que vamos a utilizar:

```
ALTER SYSTEM SET TDE_CONFIGURATION="KESTORE_CONFIGURATION=FILE" scope=both;
```

3. Tres tipos de software keystores:

- Password-protected (en realidad es el único que existe)
- Auto-login
- Local auto-login

# Administración de claves

5. Creamos primero un password protected software keystore. Se precisa el privilegio de administración de SYSKM (desde un terminal ejecutar):

Sqlplus / as syskm -- Habremos entonces entrado con ese privilegio de administración y podemos ya ejecutar todas las órdenes de administración de claves.

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY password;
```

6. Luego pasamos a autologin dicho keystore:

```
ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE IDENTIFIED BY password;
```

7. Creamos la Master key:

```
ADMINISTER KEY MANAGEMENT SET KEY identified by password with backup;
```

-- Sólo si hay algún problema al ejecutar esta orden, cerrar y abrir el keystore antes de volver a probar:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE close;
```

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE open [IDENTIFIED BY password];
```

# Identificar columnas sensibles

- ▶ **CREATE TABLE** <table\_name> (<column\_name> <datatype> ENCRYPT [algorithm] [nomac] [no salt])
- ▶ **Ejemplo:**

```
CREATE TABLE employee (  
    first_name VARCHAR2(128),  
    last_name VARCHAR2(128),  
    empID NUMBER,  
    salary NUMBER(6) ENCRYPT);
```

  - ▶ La columna será cifrada con AES 192 bits, MAC y salt (por defecto) cuando haya inserciones y modificaciones.
  - ▶ Será descifrada automáticamente en las consultas
- ▶ Cuando un usuario introduce datos, Oracle:
  1. Coge la llave maestra del wallet.
  2. Desencripta la clave usando la llave maestra.
  3. Usa la clave para encriptar los datos del usuario.
  4. Almacena los datos encriptados en la base de datos.

- ▶ En general podemos encontrar información del keystore en la vista de diccionario V\$ENCRYPTION\_WALLET
- ▶ Restricciones importantes de TDE son:
  - ▶ No puede usarse en Identity Columns (columnas numéricas autogeneradas)
  - ▶ No puede usarse en columnas que formen parte de una restricción foránea.
- ▶ Se pueden encriptar columnas o tablespace enteros pero no es recomendable utilizar los dos a la vez
- ▶ Para saber lo que hay encriptado:  
V\$ENCRYPTED\_TABLESPACES, DBA\_ENCRYPTED\_COLUMNS

# Seguridad a medida

## ▶ DBMS\_CRYPTO

- ▶ Generar claves: `DBMS_CRYPTO.RANDOMBYTES(p_length)`
- ▶ Encriptar: `DBMS_CRYPTO.ENCRYPT (`  
    `src IN RAW, -- fuentes a encriptar`  
    `typ IN pls_integer, -- tipo de encriptación`  
    `key IN RAW, -- clave`  
    `iv IN RAW DEFAULT NULL) --vector de inicialización`
- ▶ Desencriptar: `DBMS_CRYPTO.DECRYPT (`  
    `src IN RAW, -- fuentes a desencriptar`  
    `typ IN pls_integer, -- tipo de encriptación`  
    `key IN RAW, -- clave`  
    `iv IN RAW DEFAULT NULL) --vector de inicialización`



# Oracle Virtual Private Database

- ▶ Crear una función que se ejecutará cada vez que se acceda a la tabla. La función devuelve un VARCHAR2 con la condición WHERE

```
CREATE OR REPLACE FUNCTION Solo_depto_30 (  
    P_ESQUEMA IN VARCHAR2  
    , P_OBJETO IN VARCHAR2  
    ) RETURN VARCHAR2 AS  
BEGIN  
    RETURN 'CODIGO = 30';  
END Solo_depto_30;
```

- ▶ Se añade una política de seguridad

```
begin dbms_ols.add_policy (object_schema =>'USUARIO',  
object_name =>'DEPARTAMENTOS',  
policy_name =>'POL_DEPTO_30',  
function_schema =>'USUARIO',  
policy_function => 'SOLO_DEPTO_30',  
statement_types => 'SELECT, INSERT, UPDATE, DELETE' ); end;
```

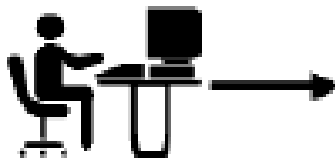
- ▶ Obviamente, la función puede ser tan compleja como se necesite, comprobando usuarios, fecha del sistema, IP de la conexión...

# Oracle Label Security

- ▶ Se debe activar la opción en la base de datos

- ▶ `SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Label Security';`

```
EXEC LBACSYS.CONFIGURE_OLS;  
-- This procedure registers Oracle Label Security.  
EXEC  
LBACSYS.OLS_ENFORCEMENT.ENABLE_OLS;  
-- This procedure enables it.
```



GRADE	RATE	ROW LABEL
Manager	800	UNCLASSIFIED
Senior	400	UNCLASSIFIED
Director	750	HIGHLY_SENSITIVE
Principal	600	SENSITIVE
Senior	450	SENSITIVE

- ▶ Crear la política de seguridad
- ▶ Crear las etiquetas asignándole un número
- ▶ Asignar a los usuarios las etiquetas a las que tienen acceso
- ▶ Añadir la tabla a la política (se le añade una columna por defecto que contiene la etiqueta)
- ▶ Asignar a la columna las etiquetas correspondientes

# Oracle DATABASE VAULT

- ▶ Cuando se activa, los usuarios SYSTEM y SYS ya no pueden crear usuarios. Habrá un Account Manager y un Database Vault Owner (para las políticas DBV). Esto favorece la separación de obligaciones.
- ▶ Se crea un “realm” conjunto de objetos a proteger
- ▶ Se asocian objetos a ese realm
- ▶ Se dan privilegios sobre ese realm a usuarios concretos
- ▶ Ni SYSTEM ni SYS pueden acceder a esos objetos por defecto.