# Control de la Seguridad de una BD

Tema 2

# **1** - Tareas de seguridad

La seguridad de una BD se basa en las 4 As:

- **Autenticación**
  - Contraseñas.
  - Autenticación multi-factor o smartcards.

- **Autorización** y **Acceso**
  - Privilegios.
  - Oracle Label Security ™.
  - Oracle Data Vault ™.

- **Auditoría**
  - Registro de accesos a los datos para validar las políticas de seguridad.

Las principales tareas de seguridad son las siguientes:

- Asegurar que la **instalación** y la **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 y privilegios de administración, sistema...
  - Restricción del acceso a datos para ciertos usuarios.
  - (...)

- Asegurar que las **conexiones de red** son **seguras**.

- **Cifrar los datos sensibles**.

- Asegurar la BD contra **agujeros de seguridad** y **protegerla de los intrusos**.

- Decidir **qué** componentes de la BD **auditar** y **cómo** hacerlo.

- **Descargar e instalar parches de seguridad**.

# **2** - Parámetros de seguridad

Si el **parámetro *O7_DICTIONARY_ACCESIBILITY*** está como *FALSE*, un usuario que reciba (mediante *GRANT*) el permiso *SELECT ANY TABLE* no podrá ver las tablas del sistema.

El **parámetro *RESOURCE_LIMIT*** activa / desactiva la capacidad de limitar los recursos de los perfiles; se modifica con la instrucción:

```
ALTER SYSTEM SET RESOURCE_LIMITS TRUE / FALSE;
```

**Estos cambios duran hasta que la BD se apague o se reinicie**, añadiendo SCOPE=SPFILE a la instrucción, el cambio se hace permanente.

Usando la instrucción **ALTER SYSTEM RESET;** se establecen los valores por defecto.

Estos parámetros pueden encontrarse usando SQL Developer en:
*Base de Datos > Configuración de la Base de Datos > Parámetros de Inicialización.*

# **3** - Gestión de usuarios

## Tipos de usuarios

Oracle tiene 3 tipos de usuario por defecto:

- **Administrador**
  - SYS: Tiene todos los privilegios y casi nunca debería usarse, ni crearse objetos en su esquema. Contiene todas las tablas y vistas del diccionario.
  - SYSTEM: Permite realizar todas las tareas administrativas, salvo *backup*, *recovery*, *upgrade*, *shutdown* y *startup* de la BD.

- **General:** Creado por un administrador.

- **Interno:** Utilizado por los procesos externos y en segundo plano de Oracle; sus cuentas están bloqueadas y nunca deben eliminarse del sistema.

## Tipos de gestión

### Sistema Operativo

- Cada persona / proceso que se conecte al servidor debe identificarse.
- Existen grupos creados por defecto en Windows.
- Conexión sin necesidad de credenciales, como **`sqlplus /`**.
  - Para ello, debe crearse un usuario con opción de identificación externa:

```
CREATE USER 'OPS$HOST\usuario'
     IDENTIFIED EXTERNALLY;
```

    - *OPS$* es un prefijo que Oracle precisa para esos usuarios, puede modificarse cambiando el parámetro *OS_AUTHENT_PREFIX*.
    - *usuario* debe existir en el sistema operativo.
    - *Cada sistema operativo impone sus reglas* case-sensitive.

## Oracle

- La conexión requiere 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.
    - **Los perfiles simplifican la gestión de usuarios, permitiendo que se fijen restricciones de contraseñas y recursos.**

```
CREATE PROFILE perfil
      CONNECT_TIME UNLIMITED          -- Duración de la conexión ilimitado
      LIMIT SESSIONS_PER_USER 3       -- Nº máx de sesiones para ese usuario
      IDLE_TIME 10                    -- Nº máx de minutos de tiempo ausente
      FAILED_LOGIN_ATTEMPTS 4         -- Nº máx de intentos para bloquear cuenta
      PASSWORD_LIFE_TIME 90           -- Nº máx de días de expiración de la contraseña
      PASSWORD_GRACE_TIME 3;          -- Tiempo de gracia tras caducar la contraseña
```

# Espacio de tablas

Antes de crear un usuario, Oracle hace una división lógica de los datos de una BD en **espacios de tablas**, que es el espacio físico en el que se almacenan los datos de las diferentes tablas de la BD.

- Al crear **una BD**, se define un espacio de tablas por defecto para los usuarios.
- Al crear **un usuario**, se puede cambiar su espacio de tablas por defecto.
- Al crear **una tabla**, se puede cambiar el espacio de tablas donde se crea.

- *No es nada aconsejable usar el espacio de tablas del usuario SYSTEM.*
- *Los espacios de tablas pueden ser temporales.*

*Ejemplos:*

- ***Comprobar los espacios de tablas existentes.***

```
SELECT *
      FROM DBA_TABLESPACES;
```

- ***Crear un espacio de tablas con un fichero de tamaño 10M autoextensible (para que un usuario pueda usarlo debe asignársele una cuota de uso).***

```
CREATE TABLESPACE nombre
      DATAFILE 'fichero.dbf'
      SIZE 10M
      AUTOEXTEND ON;
```

## Crear usuarios

```
CREATE USER usuario
     IDENTIFIED BY contraseña
     TEMPORARY TABLESPACE ts_usuario  -- Creación de un espacio de tablas temporal
     DEFAULT TABLESPACE ts_usuario    -- Tabla por defecto para sus objetos
     QUOTA 100K ON ts_usuario         -- Espacio máximo de 100 Kilobytes en ts_usuario
     PROFILE perfil;                  -- Asigna un perfil al usuario
```

- La única **cláusula obligatoria** es `IDENTIFIED`.
- Las **cuotas máximas** se establecen en cada espacio de tablas y pueden especificarse en megabytes (`M`), gigabytes (`G`) ... o bien, sin límite (`UNLIMITED`).
- **Un usuario recién creado no tiene privilegios** y deben concederle los permisos necesarios para conectarse a la BD y crear objetos (como mínimo).

- Comandos:
    - **Modificar** un usuario: `ALTER USER usuario;`
    - **Borrar** un usuario: `DROP USER usuario CASCADE;`

      *Se usa CASCADE si el esquema del usuario no está vacío, para borrar todos sus objetos.*

## Roles

Un rol es un **conjunto de privilegios**, aunque también puede ser un conjunto compuesto de otros roles.

**Primero, se crea el rol vacío y luego se le asignan privilegios y / o otros roles.**

Por defecto, se crea como no identificado, aunque puede imponerse una identificación adicional que deba aportar el usuario cuando use los privilegios de dicho rol.

- *Para poder usar un rol identificado por contraseña, primero debe utilizarse la instrucción: `SET ROLE rol IDENTIFIED BY contraseña`.*

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

Oracle crea automáticamente **3 roles predefinidos**:

- **DBA**: Contiene todos los permisos de la BD. *Debe concederse con cuidado.*
- **CONNECT**: Permite conectarse a la BD.
- **RESOURCE**: Permite crear objetos.

*Ejemplos:*

- ***Creación del rol llamado «Gestor» identificado con la contraseña «123».***

```
CREATE ROLE Gestor
      IDENTIFIED BY 123;
```

- ***Creación de un rol por defecto.***

```
CREATE ROLE usuario_normal;
```

# **4** - Permisos

Siempre debe seguirse la **política LP (*Least Privileges*)**, donde los usuarios deben recibir los mínimos privilegios posibles. **Los permisos se conceden con el comando *GRANT* y se revocan con el comando *REVOKE***. Estas sentencias sólo pueden usarse por:

- DBAs.
- Propietarios de los objetos.
- Usuarios con el privilegio y su parámetro *GRANTEABLE* activado.

*El destinatario puede ser un usuario o un rol, y se le pueden asignar permisos o roles.*

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

```
REVOKE <permisos> [ON <objeto>] FROM <usuarios>;
```

- La cláusula *ON* se elimina si se trata de un *permiso del sistema*.
- La lista <*usuarios*> que recibe el permiso **puede ser una lista de roles**.
    - Asignando permisos a *PUBLIC* (como usuario) se asignan a todos los usuarios.
- La opción *WITH … OPTION* permite a <*usuarios*> conceder dicho privilegio a:
    - GRANT: otros usuarios.
    - ADMIN: otros usuarios, si se trata de un *permiso del sistema*.
- Algunos permisos pueden especificarse solamente sobre ciertas columnas de las tablas, con la instrucción: *GRANT UPDATE columna ON tabla*.

## Permisos sobre objetos

Los permisos pueden aplicarse sobre objetos, para ello es necesario especificar el tipo de permiso y el objeto sobre el que se aplicará.

Tipos de permisos:

- SELECT
- INSERT
- DELETE
- UPDATE (sobre uno o varios atributos)
- ALTER
- REFERENCES
- INDEX
- EXECUTE
- ALL (todos los posibles sobre el objeto)

Tipos de objetos:

- Tablas
- Vistas
- Secuencias
- Procedimientos
- Funciones
- Vistas materializadas
- Etcétera

*Por defecto, el propietario de un objeto tiene todos los permisos posibles sobre él.*

*Ejemplos:*

- ***Conceder el permiso de conectarse a la BD y el rol Rol_Programador al usuario Araujo.***

```
GRANT CONNECT, Rol_Programador
      TO Araujo;
```

- ***Conceder los permisos de crear, modificar y eliminar usuarios a los usuarios Nous y Zeus, con la opción ADMIN porque son permisos del sistema.***

```
GRANT CREATE USER, ALTER USER, DROP USER
      TO Nous, Zeus
      WITH ADMIN OPTION;
```

- ***Conceder el permiso de crear procedimientos en cualquier esquema y el de crear triggers al usuario Apolonio.***

```
GRANT CREATE ANY PROCEDURE, CREATE TRIGGERS
      TO Apolonio;
```

- ***Conceder todos los permisos del rol Empleado al usuario Casandra, con opción a que dicho usuario también pueda conceder esos permisos.***

```
GRANT ALL ON Empleado
      TO Casandra
      WITH GRANT OPTION;
```

- ***Conceder el permiso de selección sobre la tabla Empleados a todos los usuarios.***

```
GRANT SELECT
      ON Empleados
      TO PUBLIC;
```

- ***Conceder el permiso para referenciar la columna DNI y actualizar las columnas Salario y Cta_Banco de la tabla Empleados al rol Rol_Nominas.***

```
GRANT REFERENCES (DNI), UPDATE (Salario, Cta_Banco)
      ON Empleados
      TO Rol_Nominas;
```

- ***Revocar el permiso de crear sesión de la lista de usuarios <Usuario1, Usuario2>.***

```
REVOKE CREATE SESSION
      FROM Usuario1, Usuario2;
```

## Permisos del sistema

Son permisos que **no se definen para un objeto concreto**.
Estos son algunos ejemplos:

| Comando | Descripción |
|---|---|
| CREATE TABLE | Crear tablas. |
| CREATE ANY TABLE | Crear tablas en cualquier esquema. |
| ALTER ANY TABLE | Modificar cualquier tabla. *Sin ANY no tiene sentido.* |
| DROP ANY TABLE | Borrar tablas de cualquier esquema. |
| DELETE ANY TABLE / VIEW | Borrar filas de tablas / vistas de cualquier esquema. |
| INSERT ANY TABLE / VIEW | Insertar filas en tablas / vistas de cualquier esquema. |
| UPDATE ANY TABLE / VIEW | Actualizar tablas / vistas de cualquier esquema. |
| SELECT ANY TABLE / VIEW | Consultar tablas / vistas de cualquier esquema. |
| EXECUTE ANY PROCEDURE | Para ejecutar procedimientos y funciones. |

*ALL PRIVILEGES especifica todos los permisos del sistema y nunca debe usarse.*

# **5** - Objetos del esquema

*Las comillas dobles ( " ) se usan para los metadatos y las simples ( ' ) para los datos.*

## Vistas

**Tabla virtual cuyas tuplas derivan de otras tablas o vistas**. A diferencia de los datos de las tablas -almacenados en memoria-, los datos de las vistas se calculan a partir de las tablas de las que dependen.

- Pueden **consultarse** como si fueran tablas.
- **Siempre están actualizadas**, ya que cualquier cambio en las tablas de las que dependen queda reflejado en ellas.
    - Una **vista que no se actualiza** se llama «vista materializada» o «***snapshot***».

Se puede **escribir sobre una vista** si:
- Está definida sobre una sola tabla.
- La tabla no posee funciones de agregación.
- Todos los campos de la tabla están declarados *`NOT NULL`*.

Se consideran vistas **no actualizables** si están definidas sobre varias tablas (reunión) o si usan agrupamiento y/o funciones de agregación.

*Se suele utilizar un trigger `INSTEAD OF` para definir el comportamiento.*

Utilidades de las vistas:
- **Seguridad**: Restringir el acceso a ciertas filas/columnas de una tabla.
- **Ocultar la complejidad** de los datos.
- **Presentar los datos** desde otra perspectiva: Cambiando los nombres de las columnas, introduciendo operaciones...
- **Aplicaciones independientes** a cambios en la tabla base.
- **Efectuar consultas** que no pueden hacerse sin vistas.

```
CREATE [OR REPLACE] [[NO] FORCE] VIEW vista [(<columnas>)]
     AS (-subconsulta-) [WITH READ ONLY];
```

- La opción *`FORCE`* fuerza que la vista se cree aunque no existan los objetos que se usan en ella o no se tengan privilegios suficientes.
- La lista <*columnas*> especifica las columnas que se incluirán en la vista.
    - Por defecto, si no se especifica nada, se incluyen todas las columnas.
- La opción *`WITH READ ONLY`* hace que los datos de la vista solo puedan leerse.

*Ejemplos:*

- ***Crear o reemplazar la vista SumiNombres como resultado de la consulta de las columnas NombreS y NombreP de las tablas {Suministros, Suministrador y Piezas}, donde…***

```
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#);
```

- ***Crear o reemplazar la vista Cantidad formada por las columnas NombreS y NumPiezas, como resultado de la consulta de contar las filas de la columna NombreS de las tablas {Suministros y Suministrador}, donde…***

```
CREATE OR REPLACE VIEW Cantidad (NombreS, NumPiezas)
      AS (SELECT NombreS, COUNT(*)
            FROM Suministros SP, Suministrador S
                  WHERE SP.P#=S.P# GROUP BY NombreS);
```

## Sinónimos

Un sinónimo es un nombre alternativo o **alias de un objeto**.

- **Solo requieren almacenar su definición** en el diccionario de datos.
- Tienen una **doble función**:
  - **Conveniencia**.
  - **Seguridad**: Ocultando el nombre y dueño del objeto, y su localización en BD distribuidas.

Tipos de sinónimos:

- **Público**: pertenece al grupo de usuarios PUBLIC y cualquier usuario puede usarlo.
- **Privado**: pertenece al subesquema de un usuario, que es el que controla su uso.

  *Si un sinónimo público tiene el mismo nombre que una tabla de usuario, no tendrá efecto para ese usuario.*

```
CREATE [PUBLIC] SYNONYM nombre FOR <objetos>;
```

*Ejemplos:*

- ***Crea un sinónimo privado pventa para la tabla Proyect_Venta del usuario Paco.***

```
CREATE SYNONYM pventa FOR Paco.Proyect_Venta;
```

- ***Crea un sinónimo público Prod para una BD remota del usuario Scott.***

```
CREATE PUBLIC SYNONYM Prod FOR Scott.Prod@Ventas;
```

# Snapshots

**Vistas que no se actualizan automáticamente**. Sus funciones son:

- Calcular y almacenar agregaciones, reuniones... en general, consultas lentas.
- Duplicar los datos localmente en accesos distribuidos, evitando accesos lejanos y lentos.
- Pueden refrescarse manualmente:
    - En intervalos regulares de tiempo.
    - Al terminar una transacción sobre las tablas base.

```
CREATE MATERIALIZED VIEW nombre <cláusulas>
     AS -subconsulta-;
```

Sobre el campo <*cláusulas*>:

- *BUILD [IMMEDIATE | DEFERRED]*: La snapshot se rellenará de forma inmediata (por defecto) o lo hará en la primera operación de REFRESH. Hasta entonces, no podrá ser usada.

- *[REFRESH <opciones> | NEVER REFRESH]*: Establece las opciones para refrescar automáticamente los datos. Las opciones pueden ser:
    - *FAST*: Se lleva a cabo usando los cambios sobre las tablas base, que se almacenan en una tabla auxiliar asociada a cada tabla base que se crea:

      ```
      CREATE MATERIALIZED VIEW LOG ON tabla INCLUDING NEW VALUES;
      ```

        - *INCLUDING NEW VALUES* especifica que en la tabla del LOG se incluyan los nuevos valores además de los viejos. La opción por defecto es *EXCLUDING NEW VALUES* (lo contrario).
        - Esta *snapshot LOG* almacena los cambios de una tabla y se actualiza con cada comando DML sobre ella.
        - Una misma *snapshot LOG* sobre cierta tabla puede servir para actualizar las *snapshot* que usen esa tabla.

          *No todas las subconsultas pueden beneficiarse de este refresco.*

    - *COMPLETE*: Se rehace la consulta.
    - *FORCE*: Se ejecutará un refresco *FAST* cuando sea posible, si no, un *COMPLETE*. Esta es la opción por defecto.
    - *ON [COMMIT | DEMAND]*: Se refresca tras cada transacción o bajo petición.
    - *START WITH <fecha>*: Especifica la fecha del primer refresco.
    - *NEXT <fecha>*: Especifica el intervalo entre refrescos (respecto a *START*).

# 6 - Seguridad de datos avanzada

## TDE

Proporciona **encriptación transparente de datos**.

- Especialmente útil para datos sensibles.
- Solo afecta a *cómo* se almacenan los datos.

**No puede usarse en:**

- Columnas numéricas auto-generadas (*Identity Columns*).
- Columnas que formen parte de una restricción foránea.

Se pueden encriptar columnas o espacios de tablas enteros, pero no es recomendable utilizar ambas encriptaciones a la vez.

*Antes de realizar una encriptación, es necesario realizar una **configuración inicial**.*

### Configuración inicial

1. **Establecer el directorio donde se va a almacenar el keystore** (si es de tipo FILE) mediante el parámetro de inicialización estático «WALLET_ROOT».

```
ALTER SYSTEM SET "WALLET_ROOT"='directorio'
      SCOPE=SPFILE;
```

*El parámetro directorio debe ser una ruta dentro de* `C:Users\user\oracle\` *o bien, el nombre de un directorio* `X` *dentro de la ruta especificada. Si no existe, lo creará y su ruta por defecto será* `C:Users\user\oracle\X`*.*

2. **Establecer el tipo de keystore que se va a usar**. Hay 3 tipos de keystores:
    - Protegida con contraseña.
    - Login-automático.
    - Login-automático local.

```
ALTER SYSTEM SET TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=FILE"
      SCOPE=BOTH;
```

*En este paso no se indica el tipo de keystore que se va a crear (paso 4), solo se inicializa su configuración.*

3. Si no funciona (error), es recomendable **reiniciar la instancia**. Para ello, debe visitarse los *Servicios de Windows* y reiniciar el servicio *OracleServices*.

4. **Crear primero una *keystore protegida con contraseña***. Se necesita el privilegio de administración de SYSKM, por lo que es recomendable realizar este paso mediante la terminal.
    - Abrir el terminal.
    - Ejecutar ***`isqlplus / as syskm`*** para acceder a través del S.O.
    - Ejecutar la instrucción:

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

*A partir de este paso, es recomendable seguir a través de la terminal hasta el final.*

5. Una vez creada la keystore, se pasa a una *keystore de login-automático*.

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

6. Cerrar el keystore.

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

7. Volver a abrir el keystore.

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY contraseña;
```

8. Crear la *master key*.

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY contraseña
WITH BACKUP;
```

Puede encontrarse información del keystore en la vista *`V$ENCRYPTION_WALLET`*.

Para saber lo que hay encriptado pueden usarse las vistas del diccionario:
- *`V$ENCRYPTED_TABLESPACES`*.
- *`DBA_ENCRYPTED_COLUMNS`*.

### Identificar columnas sensibles

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 BD.

```
CREATE TABLE tabla (columna tipo_dato
ENCRYPT [algorimo] [nomac] [no salt])
```

*Se encriptará con AES 192 bits, MAC y salt si no se especifican opciones para ENCRYPT.*
*Las columnas serán descifradas automáticamente en las consultas.*

*Ejemplo:*

- ***Creación de la tabla employee con las columnas sensibles {first_name, last_name, empID y salary}, encriptadas con AES 192 bits, MAC y salt.***

```
CREATE TABLE empleado (
      id NUMBER,
      nombre VARCHAR2(128),
      apellidos VARCHAR2(128),
      salario NUMBER(6) ENCRYPT);
```

## Paquete DBMS_CRYPTO

Proporciona **encriptación a medida**. Ofrece los siguientes comandos:

- Generar claves: `DBMS_CRYPTO.RANDOMBYTES(longitud);`

- Encriptar:

```
DBMS_CRYPTO.ENCRYPT(
      fuentes IN RAW,          Fuentes a encriptar
      tipo IN pls_integer,     Tipo de encriptación
      clave IN RAW,            Clave
      vector IN RAW            Vector de Inicialización
      DEFAULT NULL);
```

- Desencriptar:

```
DBMS_CRYPTO.DECRYPT(
      fuentes IN RAW,          Fuentes a desencriptar
      tipo IN pls_integer,     Tipo de encriptación
      clave IN RAW,            Clave
      vector IN RAW            Vector de Inicialización
      DEFAULT NULL);
```

## Oracle Virtual Private Database

**Restringe el acceso a nivel de *fila* y *columna* introduciendo cláusulas *WHERE* a todas las sentencias SQL de forma automática**. Se usa cuando ya existen datos para poder asignar los permisos.

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

  *Ejemplo:*

  - ***Función que dado un esquema y un objeto, devuelve un VARCHAR2 de aquellos objetos del esquema cuyo código sea 30.***

```
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.

  *Ejemplo:*

  - ***Política llamada* pol_depto_30*, declarada sobre la tabla* usuario.departamentos *que ejecuta la función* solo_depto_30 *(definida en el ejemplo anterior).***

```
BEGIN
      DBMS_RLS.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;
```

## Oracle Label Security

**Protege las tablas a nivel de *fila*, asignando etiquetas a cada una**. Los usuarios se autorizan teniendo en cuenta dichas etiquetas, que se utilizan dando «niveles de sensibilidad» a las filas.

- Se debe activar la opción en la BD:

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

    - `EXEC LBACSYS.CONFIGURE_OLS;`
      Procedimiento que registra *Oracle Label Security*.

    - `LBACSYS.OLS_ENFORCEMENT.ENABLE_OLS;`
      Procedimiento que activa *Oracle Label Security*.

- Funciones:
    - Crear la política de seguridad.
    - Crear las etiquetas con un número.
    - Asignar usuarios a las etiquetas a las que tienen acceso.
    - Añadir la tabla a la política (por defecto, una columna con la etiqueta).
    - Asignar a la columna las etiquetas correspondientes.

## Oracle Database Vault

**Permite restringir el acceso a datos incluso a administradores**. Cuando se activa, los usuarios SYS y SYSTEM ya pueden crear usuarios, sino que habrá un *Account Manager* y un *Database Vault Owner* (para políticas de DBV). Todo esto favorece la separación de obligaciones.

- Se crea un conjunto de objetos a proteger (*realm*).
- Se asocian objetos a ese *realm*.
- Se otorgan privilegios sobre ese *realm* a usuarios concretos.
- SYS y SYSTEM no pueden acceder a esos objetos por defecto.