Unidad 3

Gestión de usuarios y permisos

Administración de sistemas gestores de bases de datos

3

Índice

Introducción

Ya hemos instalado y realizado una configuración inicial de nuestro SGBD. Aunque ya hemos visto un ejemplo en la unidad anterior, vamos a ver en detalle cómo habilitar el acceso a los usuarios para que puedan conectarse al servidor.

Para poder garantizar apropiadamente la confidencialidad de la información, los SGBD multiusuario disponen de mecanismos para gestionar los usuarios y niveles de acceso a la información permitidos.

Esta funcionalidad se puede configurar de forma sencilla a nivel de sintaxis o utilizando interfaces de usuario. Quizás, la parte más compleja sea a nivel organizativo, pues las casuísticas son muy variadas y complejas.

Dentro de la labor documental del DBA, una de las tareas que ello conlleva es el análisis y toma de requisitos de acceso a la información. Cuántos usuarios son necesarios, a qué información deben poder acceder y con qué tipo de acceso (por ejemplo, lectura o escritura).

Vamos a introducir algunos de estos conceptos relacionados con la configuración de usuarios, permisos y esquemas. Algunos términos son genéricos y otros específicos de Oracle.

- **Rol.** Es una agrupación de privilegios o permisos que facilita la configuración de múltiples usuarios con el mismo nivel de acceso.
- **Perfil:** Agrupación de restricciones sobre los recursos accesibles para los usuarios. Por ejemplo, a través de los perfiles podemos implementar políticas de seguridad de las contraseñas.
- **Sinónimo:** Objeto que permite referenciar a otro (análogo a un acceso directo de un archivo). A través de este mecanismo, se simplifica el acceso a los objetos de otros esquemas y organizar una estructura de nombres de objeto homogénea.
- **Archivo de Datos (Datafile):** Es la estructura donde se almacena la información físicamente. Tiene un tamaño predeterminado en el momento de su creación, aunque puede ser modificado posteriormente. Un espacio de tabla puede tener uno o varios archivos de datos.
- **DCL:** Mediante este sublenguaje de SQL, podremos asignar o quitar permisos a roles determinados o asignar o quitar permisos o roles a los usuarios.
- **Vista:** Consulta almacenada como un objeto lógico pero que no ocupa espacio en disco, excepto los metadatos.
- **Usuario Externo:** Estos usuarios entran al sistema mediante un mecanismo de autenticación externo al SGBD (SO, red, kerberos, etc.).
- **Esquema externo:** Es la parte de la base de datos accesible para el usuario.
- **Esquema conceptual:** Establece los datos existentes en las bases de datos y su relación.

Al finalizar esta unidad

- Conoceremos los mecanismos de seguridad del SGBD asociados a los usuarios: autenticación, permisos y restricciones.
- Veremos las diferencias entre esquema externo e interno.
- Practicaremos el uso del diccionario de datos para realizar tareas de gestión de usuarios y permisos.
- Analizaremos en profundidad las posibilidades de configuración de usuarios, roles, permisos, vistas y sinónimos.

3.1. Gestión de usuarios y permisos

La tarea de gestión de usuarios puede ser implementada mediante los siguientes mecanismos:

- **Autenticación de usuarios:** los usuarios que vayan a acceder a las bases de datos tendrán que identificarse con sus credenciales de acceso. Este mecanismo sirve para monitorizar qué usuarios acceden y a qué información, facilitando de este modo la realización de auditorías.
- **Permisos y roles:** limitaremos o permitiremos el acceso a la información de las bases de datos.
- **Cuotas y perfiles:** además de los permisos de acceso, podremos limitar los recursos del sistema (número de peticiones, límites de transferencia de datos, etc.).

Fuera de los mecanismos del propio SGBD, es necesario coordinación entre el administrador de sistemas y el DBA, pues la autenticación y permisos de usuario debe existir también a nivel de sistema. Por ejemplo, si limitamos el acceso a una determinada base de datos a un usuario, pero éste puede acceder al fichero a través del sistema operativo del servidor, no estaremos garantizando la confidencialidad de los datos. Por este motivo, es necesario poner atención a:

- **Archivos de datos:** son los archivos que se crean con las bases de datos y las tablas. Los usuarios no autorizados no deben tener acceso a estos archivos.
- **Ficheros de Logs:** Para prevenir vulnerabilidades de seguridad, no permitiremos el acceso a estos ficheros a usuarios no administradores.
- **Ficheros de configuración:** además de los permisos de acceso, podremos limitar los recursos del sistema (número de peticiones, límites de transferencia de datos, etc.)
- **Ficheros de automatización y carga:** aquellos scripts que puedan acceder a datos sensibles deben permanecer fuera de los usuarios no autorizados.

3.1.1. Usuarios

Existe una vinculación fuerte entre el mecanismo de autenticación de usuarios y el sistema de asignación de permisos del SGBD.

Como los SGBD están implementados con arquitectura cliente/servidor, dentro de la arquitectura de la organización, tendremos mecanismos de autenticación del sistema (equipos, redes, conectividad externa, etc.), en ocasiones, los SGBD se integran con estos mecanismos a nivel de SO distribuido y delegan esos mecanismos de autenticación en el sistema, aunque internamente gestionen los permisos asociados a cada cuenta de usuario. Esta integración depende en gran medida del SGBD.

En Oracle, por cada usuario se crea un esquema, es decir un conjunto de objetos (tablas, vistas, índices, procedimientos almacenados, funciones, etc.), que son propiedad de éste.

En MySQL, un esquema es una base de datos y los usuarios no pueden tener sus propios objetos.

Aunque la sintaxis es parecida al SQL estándar, cada gestor añade ciertas modificaciones.

- **Para crear un usuario:**

```
CREATE USER <usuario> IDENTIFIED BY <contraseña>
```

- **Eliminar usuario:**

```
DROP USER <usuario> [CASCADE]
```

PARA TENER EN CUENTA...

El parámetro opcional "CASCADE", nos permitirá eliminar todos los objetos del esquema. Esto solo es posible en sistemas gestores como Oracle, que disponen de un esquema asociado al usuario.

Es una instrucción del DDL, y por lo tanto no es transaccional (no permite Rollback).

3.1.2. Permisos

A través de los permisos, determinaremos las posibles operaciones que podrá realizar un determinado usuario o rol (conjunto de permisos) sobre el SGBD. Estos permisos pueden ser a nivel de sistema u objetos.

- **Sistema:** Definen las operaciones que podrá realizar el usuario sobre el SGBD relativas a conexión, creación de tablas u otro tipo de objetos, consultas, tareas administrativas, etc. La sintaxis para modificar los permisos del sistema:

```
GRANT privilegio TO [usuario | rol | PUBLIC]
[WITH ADMIN OPTION];
```

- **Objetos:** Los permisos sobre los objetos definirán una vez creados los objetos las operaciones que los usuarios podrán realizar sobre los datos (leer, modificar, eliminar.

```
GRANT privilegio ON propietario.objeto TO [usuario
| rol | PUBLILC] [WITH GRANT OPTION]; # Habilitar
privilegios

REVOKE privilegio FROM [usuario | rol]; # Quitar
privilegios
```

3.1.3. Roles

El hecho de agrupar los privilegios en roles nos facilitará la asignación de permisos a los usuarios. Es frecuente tener que crear usuarios con los mismos privilegios, de este modo, no hay que especificar individualmente cada uno de los diferentes permisos, sino que una vez definido el rol, le asignamos todos los permisos de ese rol. Además, no siempre será necesario definir roles, de hecho, casi todos los SGBD tienen definidos unos roles por defecto, por ejemplo, el DBA, habitualmente tiene todos los permisos.

Los roles por defecto en Oracle lo podemos consultar en la tabla de roles:

También podremos consultar los permisos con la siguiente sentencia SQL:

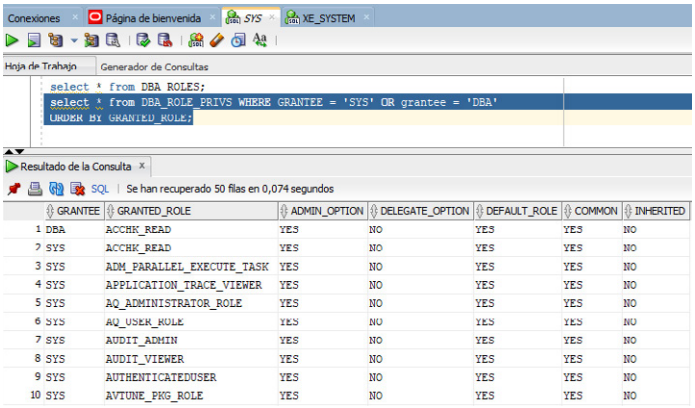

Imagen 2. Ejemplo de consulta sobre los roles DBA y SYS

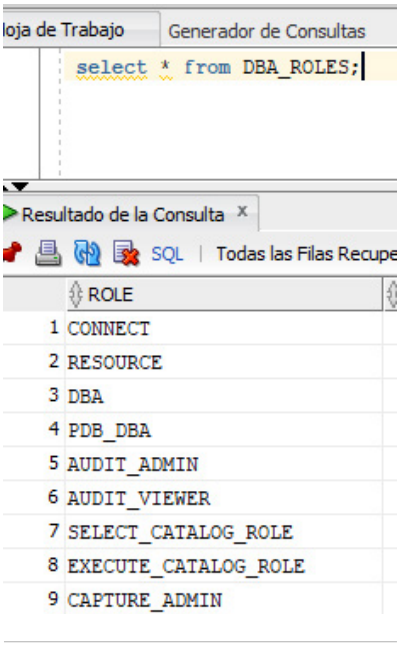

Imagen 1. Algunos de los roles por defecto de Oracle

Veamos algunos ejemplos de operaciones sobre los roles:

```
CREATE ROLE rol_nombre; # Crear un rol
GRANT privilegio [ON objeto] TO rol_nombre; # Agregar privilegio a un rol
REVOKE privilegio FROM ON rol_nombre; # Deshabilitar privilegio
GRANT rol_nombre to [usuario]; # Asignar rol a un usuario
REVOKE rol_nombre FROM [usuario | rol]; # Quitar un rol a un usuario
DROP ROLE rol_nombre; # Quitar un rol a un usuario
```

3.1.4. Perfiles

Los perfiles nos permitirán especificar restricciones que los usuarios deberán cumplir en dos niveles:

- **Recursos del sistema:** Podremos limitar el uso de los recursos para que un usuario determinado no haga uso excesivo de los mismos, sea o no de forma intencionada. Podremos configurar limitaciones en cuanto a número de sesiones, tiempos de espera, acceso a recursos de hardware (disco, cpu, ...), descarga de datos, etc.
- **Políticas de contraseñas:** Si queremos mantener unos estándares de seguridad determinados, mediante los perfiles definiremos las características de las contraseñas, así como intentos de conexión, expiración, etc.

En los SGBD, normalmente encontraremos el perfil, aunque unos sistemas lo implementan de forma similar a los roles y otros de forma individual.

3.2. Gestión de usuarios y permisos en Oracle

Oracle permite la autenticación de usuarios externos, delegando el proceso en la infraestructura de red y sistema operativo (LDAP, Kerberos o Radius).

Cada usuario externo debe ir asociado a un usuario del SGBD, que debe tener una serie de permisos establecidos, sobre el que se generarán sus esquemas propios. De este modo, cuando el usuario se autentica mediante el sistema operativo, se mapea su usuario con el usuario del SGBD.

3.2.1. Usuarios y arquitectura en Oracle

El siguiente diagrama, muestra la arquitectura multiusuario de Oracle:

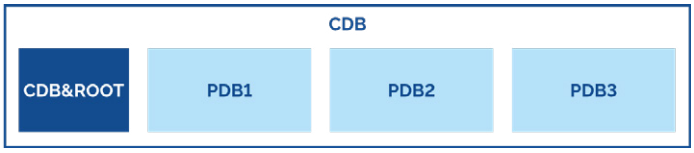

Imagen 3. Arquitectura multiusuario de Oracle: Instancia, contenedor principal (CDB) y bases de datos intercambiables (PDB)

Esta arquitectura, está diseñada para organizaciones de gran tamaño, permitiéndoles disponer de diferentes bases de datos independientes. Facilita el reparto de recursos disponibles entre las diferentes bases de datos del servidor. Las PDBs comparten los recursos de la instancia, pero pueden gestionarse de forma independiente.

Tendremos un diccionario de datos para cada PDB, que se guardará en el SYS de del PDB. Además, existirá un diccionario global en el esquema SYS del CDB$ROOT.

Oracle permite la configuración de usuarios y permisos a nivel local y global.

Configuración de usuarios locales

Este tipo de usuarios los crearemos a nivel de PDB y solamente podrán acceder a ésta. De hecho, en diferentes PDBs podremos crear usuarios con el mismo nombre, siendo éstos independientes. La sintaxis es la del estándar SQL, aunque podremos agregar adicionalmente parámetros propios de Oracle:

```
CREATE USER <usuario> IDENTIFIED BY <contraseña>
[DEFAULT TABLESPACE <tablespace>]
[TEMPRARY TABLESPACE <tablespace_temp>]
[QUOTA <tamaño> ON <tablespace>]
[PROFILE <perfil>];
```

Especificaremos el tablespace por defecto, donde se guardarán los objetos del usuario, si queremos reservar un tablespace temporal para almacenar las consultas del usuario y podremos asignar una cuota máxima de almacenamiento limitada (por ejemplo, 500K, 100M).

Por supuesto que, podemos crear en primer lugar el usuario y luego modificar esos parámetros con el comando ALTER.

Configuración de usuarios globales

Estos usuarios son comunes a todas las PDBs. Y se crean en el CDB$ROOT.

3.3.2. Permisos en Oracle

Oracle puede gestionar dos tipos de permisos: de sistema y sobre los objetos. También los podemos especificar a nivel global o local. Para que los permisos tengan efecto sobre todos los contenedores, hay que especificar CONTAINER = ALL, de lo contrario, solamente tendrán efecto sobre la PDB.

Sistema

Todos los permisos que tengan relación con las operaciones entre usuario y sistema gestor: Conexión creación de tablas, acceso al Enterprise Manager.

Objetos

Son los permisos que podremos conceder sobre los objetos de las bases de datos (Consultar, insertar, modificar o borrar datos de tablas, etc.)

Para otorgar o eliminar permisos, usaremos lasa palabras clave del lenguaje SQL GRANT y REVOKE.

Los permisos comunes no se pueden revocar desde local.

3.3.3. Roles en Oracle

Oracle también puede gestionar los roles tanto en modo local como global. Para crear o modificar los roles a nivel global, debemos especificar el parámetro CONTAINER = ALL.

```
CREATE ROLE <rol> CONTAINER = ALL; # Crear un rol Global
CREATE ROLE <rol> CONTAINER = CURRENT; # Local
GRANT <permiso> TO <rol>; # asignar permisos a un rol
REVOKE <permiso> [ON <objeto>] FROM <rol>; # revocar permisos a un rol
```

Asignación múltiple: Asignar roles a roles o asignar más de un rol a un usuario:

```
GRANT <rol> TO [<usuario> | <rol>];
REVOKE <rol> FROM [<usuario> | <rol>]; # Quitar un rol
DROP ROLE <rol>; # Eliminar rol
```

PARA TENER EN CUENTA...

En el proceso de asignación de roles, los permisos otorgados van a depender de la PDB. Por ejemplo, si un usuario local se le asigna un rol global, solamente se le serán concedidos los permisos de su PDB.

En el otro lado, si a un usuario global se le asigna un rol local, se le serán asignados los permisos del rol local en la PDB.

3.3.4. Perfiles en Oracle

Cuando creamos un usuario en Oracle, automáticamente se le asigna un perfil DEFAULT, que no tienen ningún tipo de límite establecido en cuanto al uso de los recursos.

Si queremos configurar algún tipo de restricción sobre los citados recursos, debemos habilitar los perfiles de usuario previamente, pues vienen deshabilitados por defecto:

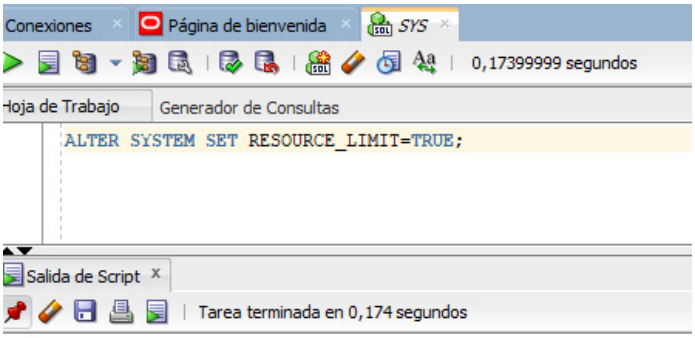

Imagen 4. Desbloquea las restricciones sobre perfiles con límites de recursos

Una vez activados, podremos crear los perfiles con la siguiente sintaxis:

```
CREATE PROFILE <perfil> LIMIT
{parámetro_recurso | contraseña} [<valor>[K|M] | UNLIMITED]
…
;
```

También podremos crear los perfiles tanto a nivel local como global (CDB$ROOT), con el parámetro "CONTAINER = ALL".

Posibles parámetros de limitación de recursos

- `SESSIONS_PER_USER:` número de sesiones simultáneas a las que desea limitar el usuario.
- `CONNECT_TIME:` límite de tiempo total transcurrido para una sesión, expresado en minutos.
- `IDLE_TIME:` periodo de tiempo inactivo continuo durante una sesión, expresado en minutos. Las consultas de larga duración y otras operaciones no están sujetas a este límite.
- `CPU_PER_SESSION:` Tiempo máximo de uso de CPU, medido en centésimas de segundo.
- `CPU_PER_CALL:` Tiempo máximo de uso de CPU en cada llamada, medido en centésimas de segundo.
- `LOGICAL_READS_PER_SESSION:` número permitido de bloques de datos leídos en una sesión, incluidos los bloques leídos de la memoria y el disco.
- `LOGICAL_READS_PER_CALL:` número permitido de bloques de datos leídos para que una llamada procese una instrucción SQL (un análisis, una ejecución o una obtención).
- `PRIVATE_SGA:` cantidad de espacio privado que una sesión puede asignar en el grupo compartido del área global del sistema (SGA). Consulte size_clause para obtener información sobre esa cláusula.

Ejemplo:

```
alter session set "_oracle_script"=true;
CREATE PROFILE mi_perfil LIMIT
   SESSIONS_PER_USER          UNLIMITED
   CPU_PER_SESSION            UNLIMITED
   CPU_PER_CALL               3000
   CONNECT_TIME               45
   LOGICAL_READS_PER_SESSION  DEFAULT
   LOGICAL_READS_PER_CALL     1000
   PRIVATE_SGA                15K
   COMPOSITE_LIMIT            5000000;
```

Imagen 5. Ejemplo de creación de perfil con límite de recursos

Posibles parámetros de restricción de contraseñas

Las siguientes cláusulas establecen los parámetros de restricciones sobre contraseñas. Las que se refieren a unidades de tiempo, van en días, aunque podremos utilizar unidades más pequeñas para realizar pruebas (n/1440) para minutos, e incluso (n/86400) para segundos.

- `FAILED_LOGIN_ATTEMPTS:` Número de fallos permitidos antes del bloqueo.
- `PASSWORD_LIFE_TIME:` Número de días de validez de la contraseña.
- `PASSWORD_REUSE_TIME / PASSWORD_REUSE_MAX:` Se usan a la vez. Especifican el número de días antes de los cuales no puede reutilizarse una contraseña y el número de cambios totales antes de poder reutilizar la contraseña actual respectivamente. Ambos deben ir con números enteros.
- **PASSWORD_LOCK_TIME:** Número de días que se bloquearía una cuenta en caso de fallos de login.
- **PASSWORD_GRACE_TIME:** Número de días en los que se advierte sobre la caducidad de la contraseña.
- **PASSWORD_VERIFY_FUNCTION:** Función que valida la complejidad de la contraseña, pudiendo definir una función personalizada, aunque Oracle proporciona un script predeterminado.

Ejemplos:

```
CREATE PROFILE mi_pw_prof LIMIT
   FAILED_LOGIN_ATTEMPTS 5
   PASSWORD_LIFE_TIME 60
   PASSWORD_REUSE_TIME 60
   PASSWORD_REUSE_MAX 5
   PASSWORD_LOCK_TIME 1/24
   PASSWORD_GRACE_TIME 10;
```

Imagen 6. Ejemplo de creación de perfil con políticas de contraseñas

3.3.5. Diccionario de datos

Si navegamos por el esquema SYS, de nuestro PDB, podemos examinar la estructura del diccionario. A través de las vistas, podemos ver que contiene información de usuarios, permisos, roles, etc.

Los tres tipos de vistas principales, siguien el siguiente esquema de nombres:

- `USER_[tabla]:` Muestra la información particular del usuario.
- `ALL_[tabla]:` muestra la información de los objetos a los que puede acceder el usuario.
- `DBA_[tabla]:` Información de todos los objetos del PDB.

```
SELECT * FROM USER_USERS;
SELECT * FROM DBA_USERS;
SELECT * FROM ALL_USERS;
```

Imagen 7. Consultando vistas sobre el diccionario de datos

```
SELECT * FROM DBA_ROLES
SELECT * FROM DBA_PROFILES
```

Imagen 8. Consultando vistas sobre el diccionario de datos: roles y perfiles

Además, en el diccionario de datos común (Usuario SYS del CDB$ROOT), podremos consultar los objetos comunes. Los nombres de las vistas con iguales, pero precedidos de CDB:

```
SELECT * FROM CDB_ROLES;
SELECT * FROM CDB_PROFILES;
SELECT * FROM CDB_ROLE_PRIVS;
```

Imagen 9. Consultando vistas sobre el diccionario de datos global: roles, perfiles y privilegios

3.3. Esquemas externos

A la hora de asignar permisos sobre los objetos a los usuarios de la base de datos, tendremos dos formas de hacerlo.

La primera consiste en definir qué usuarios tienen acceso a determinadas tablas. Esta opción es menos segura, ya que estamos dando información sobre la estructura de nuestras tablas a la aplicación, además si modificamos las tablas añadiendo o quitando campos, el conjunto de resultados puede variar y causar fallos en la aplicación.

La alternativa es hacerlo a través de vistas específicas por usuario y rol. Por cada usuario y rol estaríamos creando un esquema externo. Si los usuarios tienen que acceder a todo el contenido de las tablas en caso de tener muchos usuarios, la gestión se complica.

La opción recomendada, es una solución híbrida, de forma que sobre algunas tablas daremos acceso completo, mientras que para otras emplearemos vistas.

3.3.1. Aplicaciones externas y usuarios

En una organización, las aplicaciones de gestión también hacen uso de usuarios y autenticación, pero no es necesario ni recomendable que cada usuario de la aplicación tenga asignado un usuario de la base de datos. De hecho, los usuarios de las aplicaciones de gestión empresarial no acceden directamente a las bases de datos, sino que es la aplicación la que gestiona este acceso. Con un usuario para acceder a la aplicación sería suficiente.

Lo habitual en el momento de creación de usuarios es crear un usuario propietario de la base de datos, con permisos para todo. Este usuario creará vistas y definirá lo que pueden ver los demás usuarios.

Se suele crear un usuario por cada aplicación externa que accede a los datos y estos accederán solamente a la información que el propietario les ha permitido.

Para facilitar la auditoría de la información, cuando una aplicación comparte un mismo usuario de base de datos para varios usuarios de la aplicación, normalmente se deja registro del usuario de la aplicación sobre las operaciones realizadas. Conviene crear diferentes usuarios para diferentes aplicaciones, aunque tengan los mismos permisos sobre los datos.

3.3.2. Vistas

Una vista es una representación virtual de una tabla en base de datos. Puede ser formada mediante una o varias tablas. La tabla realmente no se almacena, sino que se almacena la consulta para su uso posterior. Es decir, no ocupan espacio, más allá de los metadatos.

Este mecanismo permite flexibilidad y facilita la gestión cuando tenemos que limitar la información a la que pueden acceder determinados usuarios.

Las vistas las definirá el usuario propietario de los datos.

Encontraremos tres tipos de vistas:

- **Horizontales:** Restringen al usuario el acceso a determinadas filas. Alguna condición sobre una determinada columna actúa como filtro. Por ejemplo, SELECT empleado_id, empleado_nombre, empleado_sueldo FROM empleados WHERE departamento_id = 1
- **Verticales:** Restringen al usuario el acceso a determinadas columnas. Por ejemplo, SELECT empleado_id, empleado_nombre, departamento_id FROM empleados. (Ocultamos la columna empleado_sueldo)
- **Mixtas:** Hacemos uso de los dos tipos de restricciones, por ejemplo: SELECT empleado_id, empleado_nombre FROM empleados WHERE departamento_id = 1 (Ocultamos la columna empleado_sueldo y solamente mostramos los del departamento 1)

Para crear una vista en SQL, la sintaxis sería:

```
CREATE [OR REPLACE] VIEW <vista_nombre> AS
<SELECT QUERY>
Según el ejemplo anterior:
CREATE VIEW <vista_empleados_sin_sueldo_dpt1> AS
SELECT empleado_id, empleado_nombre FROM empleados
WHERE departamento_id = 1
```

La sentencia ALTER, no puede utilizarse para modificar una vista, como si de una tabla se tratase. Se recomienda no utilizar el "*" y especificar explícitamente los campos que se van a mostrar. Esto es para prevenir fallos en las aplicaciones, en caso de agregar columnas nuevas sobre las tablas.

Para eliminar una vista, emplearemos la sintaxis:

```
DROP VIEW <vista_nombre>
```

3.3.3. Sinónimos

Los sinónimos son en bases de datos lo que los accesos directos a los ficheros. Este mecanismo nos aportará diversas ventajas:

- **Seguridad:** Gracias a que podemos ocultar la información a la que están accediendo los usuarios.
- **Facilidad de uso:** Al ser creados en el esquema de usuario, podemos hacer uso de nombres propios del usuario que pueden ser referenciados a las mismas tablas.
- **Homogeneidad:** podemos definir un sinónimo con el mismo nombre en distintos usuarios que acceda a información distinta. Por ejemplo, si una consultora gestiona un ERP para diferentes organizaciones, usando una misma base de datos, los usuarios pueden tener el sinónimo "ventas", haciendo referencia para el usuario "empresa_1" a la tabla o vista "ventas_1", mientras que el usuario "empresa_2", el sinónimo "ventas" hace referencia a la tabla o vista "ventas_2".

Algunos SGBD, no tienen este mecanismo, pero se puede simular a través de las vistas.

3.4. Esquemas externos en Oracle

En Oracle, para cada usuario, internamente se crea un esquema en la base de datos.

Para crear los esquemas externos, se crea un usuario propietario, en el que se va a definir todo el esquema conceptual de la base de datos, así como las vistas necesarias para limitar el acceso a los datos al resto de usuarios.

Para cada aplicación o usuario que tenga que acceder a la base de datos, se creará un usuario "invitado", en cuyo esquema se definirán los sinónimos que necesite para apuntar a las tablas y/o vistas del usuario propietario.

La sintaxis para crear las vistas y los sinónimos en Oracle es similar al estándar SQL:

```
CREATE [OR REPLACE] VIEW <vista_nombre> [(lista_de_columnas)]
AS
<SELECT QUERY>

CREATE [OR REPLACE] SYNONYM <sinónimo> FOR esq_propietario.<vista o tabla>
```

CASO PRÁCTICO

Supongamos que una consultora, gestiona el software ERP de varias empresas comercializadoras de electricidad, mediante una base de datos Oracle centralizada.

La empresa consultora tiene un usuario propietario llamado "erp", en cuyo esquema se guarda la tabla "facturas", que contiene datos de facturas de los clientes de todas las comercializadoras que gestiona. Como cada comercializadora solamente debe tener acceso a las facturas de sus propios clientes, se va a limitar el acceso a las mismas mediante vistas:

```
CREATE VIEW erp.facturas_comer1 AS SELECT factura_id, cliente_id, cantidad FROM erp.facturas WHERE comer_id = 1
CREATE VIEW erp.facturas_comer2 AS SELECT factura_id, cliente_id, cantidad FROM erp.facturas WHERE comer_id = 2
```

Cada comercializadora tendrá su propio usuario (erp_comerx) y por lo tanto su propio esquema.

A cada usuario se le dará permiso de acceso a la vista que le corresponda, creando un sinónimo con el mismo nombre para todas las comercializadoras, pero realmente acceden a diferentes datos.

```
GRANT SELECT ON VIEW erp.facturas_comer1 TO erp_comer1;
CREATE SYNONYM erp_comer1.facturas FOR erp.facturas_comer1;
GRANT SELECT ON VIEW erp.facturas_comer2 TO erp_comer2;
CREATE SYNONYM erp_comer2.facturas FOR erp.facturas_comer2;
```

Todas las comercializadoras usarán la misma aplicación, pero al tratarse de diferentes usuarios, la consulta interna que realiza la aplicación "SELECT * FROM facturas" devolverá datos diferentes para cada comercializadora.

www.universae.com