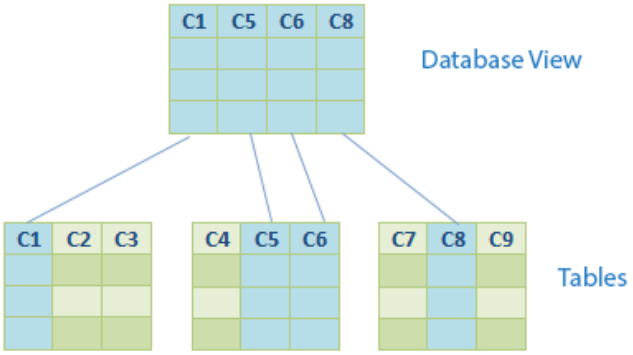
TEMA 11. PARTE I.

CREACIÓN Y MANIPULACIÓN DE OTROS OBJETOS DE LA BASE DE DATOS.

Contenido

1. VISTAS

Las vistas son **objetos** de la base de datos, conocidas también como ***tabla virtual***, y cuyo contenido está definido por una consulta con SELECT. También se puede decir que una vista es una consulta almacenada a fin de utilizarla tantas veces como se desee.



Como se ha dicho anteriormente, las vistas almacenan la consulta realizada para la definición de la vista, y no los datos accesibles a través de la vista. Por tanto, una vista **no contiene datos** sino la instrucción SELECT necesaria para crear la vista. Eso asegura que los datos sean coherentes al utilizar los datos almacenados en las tablas. Por todo ello, las vistas gastan muy poco espacio de disco.

Las vistas se emplean para:

- ✓ Realizar consultas complejas más fácilmente, ya que permiten dividir la consulta en varias partes.
- ✓ Utilizar visiones especiales de los datos.
- ✓ Ser utilizadas como tablas que resumen todos los datos.
- ✓ Ser utilizadas como *cursores de datos* en los lenguajes procedimentales (como PL/SQL).
- ✓ **Otras razones**: por seguridad, por simplicidad (las vistas pueden simplificar el modo en el que los usuarios interactúan con la base de datos, por ej: se puede crear una vista con una consulta compleja y utilizar ésta para realizar informes, subconsultas, combinaciones, etc.)

Hay dos tipos de vistas:

- **Simples**. Acceden a **una sola tabla** y **no** contienen **funciones de agrupación**. Su ventaja es que permiten siempre realizar operaciones DML sobre ellas.
- **Complejas**. Obtienen datos de **varias tablas**, pueden utilizar **funciones de agrupación**. No siempre permiten operaciones DML (SELECT, INSERT, UPDATE, DELETE)

Otras consideraciones sobre las VISTAS:

  - Las vistas se crean en la base de datos activa.

- Es conveniente probar la sentencia SELECT con la cual definiremos la VISTA, para asegurarnos que el resultado que retorna es lo deseado.
- Las VISTAS siempre están actualizadas; si se modifican las tablas base (a las que hace referencia una determinada VISTA), la VISTA reflejará los cambios.
- Se pueden construir VISTAS sobre otros VISTAS.
- Si se modifican los datos de una VISTA, se modifica la tabla base.

1.1 Creación de vistas

Sintaxis:

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW vista [(alias[, alias2...])]
AS consultaSELECT
[WITH CHECK OPTION [CONSTRAINT nombre-restricción]]
[WITH READ ONLY [CONSTRAINT nombre -restricción]]
```

- **OR REPLACE**. Si la vista ya existía, la cambia por la actual
- **FORCE**. Crea la vista aunque los datos de la consulta SELECT no existan
- **vista**. Nombre que se le da a la vista
- **alias**. Lista de alias que se establecen para las columnas devueltas por la consulta SELECT en la que se basa esta vista. El número de alias debe coincidir con el número de columnas devueltas por SELECT.
- **WITH CHECK OPTION**. Es la opción de comprobación para una vista. Si se especifica, SQL comprueba automáticamente cada operación INSERT y UPDATE sobre la vista para asegurarse de que los datos satisfacen la condición de definición de la vista.

Ejemplo:

```
create view vista_empleados
 as
  select apellido, nombre, sexo, seccion
  from empleados
  where seccion='Administracion'
  with check option;
```

Con la cláusula “witn check option” no se permiten modificaciones en aquellos campos que afecten a los registros que retorna la vista. Es decir, no podemos modificar el campo "seccion" porque al hacerlo, tal registro ya no aparecería en la vista, pero podemos actualizar los demás campos. Por ejemplo, si intentamos actualizar a "Sistemas" el campo "seccion" de un registro mediante la vista, Oracle muestra un mensaje de error.

La misma restricción surge al ejecutar un "insert" sobre la vista; solamente podemos insertar registros con el valor "Administracion" para "seccion"; si intentamos insertar un registro con un valor diferente de "Administracion" para el campo "seccion", Oracle mostrará un mensaje de error.

- **WITH READ ONLY.** Hace que la vista sea de sólo lectura. Por ejemplo, creamos la vista del siguiente modo:

```
create view vista_empleados
 as
  select apellido, nombre, sexo, seccion
  from empleados
  where seccion='Administracion'
  with read only;
```

El sistema gestor (oracle en nuestro caso), responderá con un mensaje de error ante cualquier INSERT, UPDATE, DELETE realizado sobre la vista.

Lo bueno de las vistas es que tras su creación se utilizan como si fueran una tabla.

Ejemplo:

```
CREATE VIEW resumen
/* alias */
(id_localidad, localidad, poblacion, n_provincia, provincia, superficie,
capital_provincia, id_comunidad, comunidad, capital_comunidad)
AS
(  SELECT l.id_localidad, l.nombre, l.poblacion,
          n_provincia, p.nombre, p.superficie, l2.nombre,
          id_comunidad, c.nombre, l3.nombre
  FROM localidades l
  JOIN provincias p USING (n_provincia)
  JOIN comunidades c USING (id_comunidad)
  JOIN localidades l2 ON (p.id_capital=l2.id_localidad)
  JOIN localidades l3 ON (c.id_capital=l3.id_localidad)
);

SELECT DISTINCT (comunidad, capital_comunidad)
FROM resumen;
/* La vista pasa a usarse como una tabla normal*/
```

La creación de la vista[1] del ejemplo es compleja ya que hay relaciones complicadas, pero una vez creada la vista, se le pueden hacer consultas como si se tratara de una tabla normal. Incluso se puede utilizar el comando DESCRIBE sobre la vista para mostrar la estructura de los campos que forman la vista o utilizarse como subconsulta en los comandos UPDATE o DELETE.

[1] LECTURA ADICIONAL: http://www.w3resource.com/sql/creating-views/creating-view.php

1.2 Mostrar la lista de vistas

La vista del diccionario de datos de Oracle USER_VIEWS permite mostrar una lista de todas las vistas que posee el usuario actual. Es decir, para saber qué vistas hay disponibles se usa:

```
SELECT * FROM USER_VIEWS;
```

La columna TEXT de esa vista contiene la sentencia SQL que se utilizó para crear la vista (sentencia que es ejecutada cada vez que se invoca a la vista).

1.3 Borrar vistas

Se utiliza el comando DROP VIEW:

```
DROP VIEW nombreDeVista;
```

2. SINÓNIMOS

Cuando tenemos acceso a las tablas de otro usuario y deseamos consultarlas es preciso teclear el nombre del usuario propietario antes del nombre de la tabla. Es decir, si DIEGO tiene acceso a la tabla DEPART de PEDRO y la quiere consultar, tendrá que teclear la siguiente orden para poder hacerlo: SELECT * FROM **PEDRO.DEPART;**

Mediante el uso de sinónimos, DIEGO puede crear un sinónimo para referirse a la tabla de PEDRO sin necesidad de incluir su nombre: SELECT * FROM **TABLAPEDRO;**

Un **sinónimo** es un nuevo nombre que se puede dar a una tabla o vista. Con los sinónimos se pueden utilizar **dos nombres diferentes para referirse a un mismo objeto**. Resultan interesantes cuando se tiene acceso a tablas de otros usuarios; se pueden crear sinónimos para hacer referencia a esas tablas y, así, no hay que escribir el nombre de usuario propietario de la tabla delante de la tabla a la que tenemos acceso cada vez que deseemos consultarla.

Ejemplo: https://orlandoolguin.wordpress.com/2009/02/22/manejo-de-sinonimos/ (un poco antiguo, pero bien explicado).

2.1 Creación

Sintaxis:

```
CREATE [PUBLIC] SYNONYM nombre FOR objeto;
```

objeto es el objeto al que se referirá el sinónimo.

La cláusula **PUBLIC** hace que el sinónimo esté disponible para cualquier usuario (sólo se permite utilizar si disponemos de privilegios administrativos).

2.2 Lista de sinónimos

La vista **USER_SYNONYMS** permite observar la lista de sinónimos del usuario, la vista **ALL_SYNONYMS** permite mostrar la lista completa de sinónimos de todos los esquemas a los que tenemos acceso.

2.3 Borrar sinónimos

Únicamente los DBA y los usuarios con el privilegio DROP PUBLIC SYNONYM pueden suprimir sinónimos PUBLIC.

Igualmente, sólo los DBA y los usuarios con el privilegio DROP ANY SYNONYM pueden borrar los sinónimos de otros usuarios.

```
DROP SYNONYM nombre;
```

3. ÍNDICES

Los índices son otro tipo de **objetos** de la BBDD que forman parte del esquema, y que ***hacen que las bases de datos aceleren las operaciones de consulta*** y ordenación sobre los campos a los que el índice hace referencia. Los índices son estructuras asociadas a las tablas, se podría decir que los índices de una bbdd es "una tabla que almacena los campos indexados de la BBDD", creada para acelerar las consultas.

Se almacenan aparte de la tabla a la que hace referencia, lo que permite crearlos y borrarlos en cualquier momento.

Por ejemplo:

Se supone que Hacienda dispone de una tabla de CONTRIBUYENTES en la que cada fila se identifica por el NIF del contribuyente. Para consultar un NIF determinado realizamos la siguiente consulta: SELECT * FROM CONTRIBUYENTES WHERE NIF='7866978-A'; si la columna NIF no está indexada, Oracle recorre la tabla de CONTRIBUYENTES secuencialmente, desde la primera fila hasta el final de la tabla, tanto si encuentra como si no encuentra el dato que se busca. Si la columna NIF está indexada, Oracle realizará una búsqueda binaria en el índice hasta encontrar el dato buscado, obteniendo el **ROWID**[2] de la fila asociada mediante un único acceso sobre la tabla CONTRIBUYENTES.

Lo que realizan es una lista ordenada por la que Oracle puede acceder para facilitar la búsqueda de los datos. Cada vez que se añade un nuevo registro, los índices involucrados se actualizan a fin de que su información esté al día. De ahí que cuantos más índices haya, más le cuesta a Oracle añadir registros, pero más rápidas se realizan las instrucciones de consulta.

La mayoría de los índices se crean de manera implícita, como consecuencia de las restricciones PRIMARY KEY, UNIQUE y FOREIGN KEY. Estos son índices obligatorios, por lo que los crea el propio SGBD.

[2] ¿Qué es eso del **ROWID**?. El **rowid**, como su nombre indica "identificador de fila", es una cadena de texto alfanumérica que *identifica de forma única cada fila de cada tabla.*
Echar un vistazo a: https://davidburgos.blog/como-utilizar-y-diferenciar-el-rowid-y-rownum-de-oracle/

3.1 Creación de índices

Aparte de los índices obligatorios comentados anteriormente, se pueden crear índices de forma explícita. Éstos se crean para aquellos campos sobre los cuales se realizarán búsquedas e instrucciones de ordenación frecuente.

Sintaxis:

```
CREATE INDEX nombre ON tabla (columna1 [,columna2...])
```

Ejemplo:

```
CREATE INDEX nombre_completo ON clientes (apellido1, apellido2, nombre);
```

El ejemplo crea un índice para los campos apellido1, apellido2 y nombre. Esto no es lo mismo que crear un índice para cada campo, este índice es efectivo cuando se buscan u ordenan clientes usando los tres campos (apellido1, apellido2, nombre) a la vez.

Se aconseja crear índices en campos que:

- Contengan una gran cantidad de valores.
- Contengan una gran cantidad de nulos.
- Sean parte habitual de cláusulas WHERE, GROUP BY u ORDER BY.

No se aconseja crear índices en campos que:

- Pertenezcan a tablas pequeñas.
- No se usen a menudo en las consultas.
- Pertenecen a tablas que se actualizan frecuentemente.
- Se utilizan en expresiones.

Los índices se pueden crear utilizando expresiones complejas:

```
CREATE INDEX nombre_complejo ON clientes (UPPER(nombre));
```

Esos índices tienen sentido si en las consultas se utilizan exactamente esas expresiones.

3.2 Lista de índices

Para ver la lista de índices en Oracle se utiliza la vista **USER_INDEXES**. Mientras que la vista **USER_IND_COLUMNS** muestra la lista de columnas que son utilizadas por índices

3.3 Borrar índices

La instrucción **DROP** INDEX seguida del nombre del índice permite eliminar el índice en cuestión.

4. SECUENCIAS

Hemos visto que las **vistas**, los **índices**, las **tablas** son "objetos" de una base de datos.

Otro tipo de objeto de una bbdd es ***una secuencia.*** Una secuencia se usa para generar valores enteros secuenciales únicos y asignárselos a campos numéricos. Los valores generados por una secuencia se utilizan generalmente para las claves primarias de las tablas (claves cuyo valor no interesa, sólo sirven para identificar los registros de una tabla), y garantizar así que sus valores no se repiten.

Es decir se utilizan en los identificadores de las tablas, siempre y cuando no importe qué número se asigna a cada fila.

Es una rutina interna de la base de datos la que realiza la función de generar un número distinto cada vez. Las secuencias se almacenan independientemente de la tabla, por lo que la misma secuencia se puede utilizar para diversas tablas.

4.1 Creación de secuencias

Una **secuencia** es un objeto de base de datos que sirve para generar números enteros únicos. Se utilizan para generar valores para campos que se utilizan como clave forzada – subrogada o sustituta - (claves cuyo valor no interesa, sólo sirven para identificar los registros de una tabla). Para crear una secuencia en el esquema propio es necesario tener el privilegio CREATE SEQUENCE. Se crea una secuencia en cualquier otro esquema con el privilegio CREATE ANY SEQUENCE.

El formato para crear una secuencia es éste:

```
CREATE SEQUENCE nombre_secuencia
[START WITH n]
[INCREMENT BY n]
[{MAXVALUE n|NOMAXVALUE}]
[{MINVALUE n|NOMINVALUE}]
[{CYCLE|NOCYCLE}]
```

Donde:

- **nombre_secuencia**. Es el nombre que se le da al objeto de secuencia.
- **START WITH**. Indica el valor inicial de la secuencia (por defecto 1).
- **INCREMENT BY**. Indica cuánto se incrementa la secuencia cada vez que se usa. Por defecto se incrementa de uno en uno
- **MAXVALUE**. Máximo valor que puede tomar la secuencia. Este número entero debe ser mayor o igual que el entero especificado en MINVALUE.
- **NOMAXVALUE.** Señala que el valor máximo para una secuencia ascendente es 10^{27} y para una secuencia descendente –1
- **MINVALUE**. Mínimo valor que puede tomar la secuencia. Por defecto -1026

- **CYCLE**. Hace que la secuencia vuelva a empezar si se ha llegado al máximo valor. Por defecto es NOCYCLE.

Ejemplo:

```
CREATE SEQUENCE numeroPlanta
STARTS WITH 100
INCREMENT BY 100
MAXVALUE 2000;
```

4.2 Ver lista de secuencias

La vista del diccionario de datos de Oracle USER_SEQUENCES muestra la lista de secuencias actuales. La columna LAST_NUMBER muestra cual será el siguiente número de secuencia disponible.

4.3 Uso de las secuencias

Como hemos dicho, las secuencias son tablas y por lo tanto se accede a ellas mediante consultas, empleado un SELECT. Sin embargo se usan de un modo *especial* utilizando pseudocolumnas que recuperan el valor actual y el siguiente valor de la secuencia. Estas pseudocolumnas pueden incluirse en el FROM de una consulta a otra tabla o de la tabla DUAL.

Para recuperar los valores de una secuencia utilizamo lo métodos **NEXTVAL** y **CURRVAL** (o pseudocolumnas), con los que se obtienen el siguiente número de la secuencia y el valor actual respectivamente.

Ejemplo de uso (Oracle):

```
SELECT numeroPlanta.NEXTVAL FROM DUAL;
```

En SQL estándar:

```
SELECT nextval('numeroPlanta');
```

Eso muestra en pantalla el siguiente valor de la secuencia. Realmente NEXTVAL incrementa la secuencia y devuelve el valor actual. CURRVAL devuelve el valor de la secuencia, pero sin incrementar la misma.

Ambas funciones pueden ser utilizadas en:

- Una consulta SELECT que no lleve DISTINCT, ni grupos, ni sea parte de una vista, ni sea subconsulta de otro SELECT, UPDATE o DELETE
- Una subconsulta SELECT en una instrucción INSERT
- La cláusula VALUES de la instrucción INSERT
- La cláusula SET de la instrucción UPDATE

No se puede utilizar (y siempre hay tentaciones para ello) como valor para la cláusula DEFAULT de un campo de tabla.

Su uso más habitual es como apoyo al comando INSERT (en Oracle):

```
INSERT INTO plantas (num, uso)
VALUES (numeroPlanta.NEXTVAL, 'Suites');
```

4.4 Modificar secuencias

Se pueden modificar las secuencias, pero la modificación sólo puede afectar a los futuros valores de la secuencia, no a los ya utilizados. Sintaxis:

ALTER SEQUENCE secuencia
[**INCREMENT BY** n]
[**START WITH** n]
[{**MAXVALUE** n|**NOMAXVALUE**}]
[{**MINVALUE** n|**NOMINVALUE**}]
[{**CYCLE**|**NOCYCLE**}]

4.5 Borrar secuencias

Lo hace el comando DROP SEQUENCE seguido del nombre de la secuencia a borrar.

4.6 Listar secuencias

En SQL estándar, a través de **INFORMATION_SCHEMA.SEQUENCES** podemos acceder a la información sobre todas las secuencias creadas.

En Oracle se hace mediante la vista **USER_SEQUENCES,** permite observar la lista de secuencias del usuario

5. USUARIOS

5.1 Creación de usuarios

Al instalar una BD se crean automáticamente usuarios Administrados (DBA). En Oracle son los usuarios **SYS**, **SYSTEM**, MDSYS, y otros usuarios como ANONIMOUS , HR (que como vimos en la instalación de Oracle, está bloqueado en 11g).

Los usuarios SYS y SYSTEM tienen en principio la misma contraseña que se introduce cuando se instala el gestor de Base de datos. Cuando se termina la instalación podemos cambiar la contraseña y desbloquear el usuario HR. **SYS** es el propietario de las tablas del **diccionario de datos** y solamente él puede modificarlas. **SYSTEM** es el usuario DBA que crea ORACLE para realizar las tareas de administración de la BD.

Sintaxis de creación de nuevos usuarios:

```
CREATE USER nombre_usuario IDENTIFIED (BY password | EXTERNALLY)
[DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA (integer (K|M) | UNLIMITED ) ON tablespace)
[PASSWORD EXPIRE]
[ACCOUNT (LOCK | UNLOCK)]
[PROFILE ( perfil | DEFAULT)];
```

- **DEFAULT TABLESPACE**. Indica el *tablespace*[3] por defecto que se asigna al usuario.
- **TEMPORARY TABLESPACE**. Indica el *tablespace* temporal que se asigna al usuario (donde el usuario crea sus objetos temporales).
- **QUOTA** (integer (K|M) | **UNLIMITED**) **ON**. Asigna un espacio en Kbytes o en Megabytes en el tablespace asignado. Si no se especifica esta cláusula el usuario no puede crear objetos en el tablespace. Para conceder recursos ilimitados a un usuario se le debe dar el privilegio **UNLIMITED TABLESPACE** con la orden **GRANT**. Por defecto el valor de este parámetro es **UNLIMITED**.
- **PASSWORD EXPIRE**. Permite al usuario cambiar de clave al conectarse la primera vez.
- **ACCOUNT** (**LOCK** | **UNLOCK**). Permite bloquear y desbloquear la cuenta del usuario.
- **PROFILE** (perfil | **DEFAULT**)]. Se utiliza para controlar el uso de los recursos. Sirve para asignar un perfil determinado. Un perfil limita el número de sesiones concurrentes de un usuario, limita el tiempo de uso de la CPU, tiempo de una sesión, desconecta al usuario si sobrepasa el tiempo máximo de uso,.... Por defecto asigna el perfil por omisión al usuario creado.

Ejemplo:

```
CREATE USER milagros
IDENTIFIED BY milagros
DEFAULT TABLESPACE trabajo
TEMPORARY TABLESPACE temp
QUOTA 500K on trabajo
```

---

[3] En Oracle, los **espacios de tabla** o **tablespaces**, son una estructura lógica (una especie de "contenedor virtual") que conceptualmente, se sitúa entre la lógica de la base de datos y las estructuras físicas que almacenarán los datos.

En la base de datos, se manejan objetos a nivel lógico (tablas, columnas, filas, vistas, índices,…). La información de esos objetos se tiene que almacenar en archivos de datos. Oracle crea los **tablespaces** como un elemento intermedio entre el nivel lógico y el nivel físico de la base de datos. Relaciona ambas ópticas para optimizar el funcionamiento del sistema. Ver: https://jorgesanchez.net/manuales/abd/arquitectura-oracle.html

Para entenderlo fácilmente: el **tablespace** es donde el usuario va a ir almacenando todos los objetos creados. Tendremos que asignar un tablespace DEFAULT (por defecto) y uno TEMPORARY (temporal).

NOTA ADICIONAL: Como veremos en los ejercicios en la creación de los usuarios se les asignará un ***Tablespace default***, que como se ha nombrado anteriormente es donde el usuario va a poder guardar sus objetos por defecto. Sin embargo, esto no significa que pueda crear objetos sin más, o que tenga una cuota de espacio para almacén de sus objetos. Estos permisos se asignan de forma separada, es decir: por un lado se le debe asignar una tablaspace al usuario, por otro una quota, y finalmente el privileo "create table". Sin embargo, si se le asigna al usuario el **rol RESOURCE (resource como veremos en una tabla más abajo es un conjunto de privilegios)**, se asigna una quota **unlimited** para almacén de objetos en el Tablespace por defecto indicado (incluso en el Tablespace SYSTEM), y por tanto podrá crear tablas sin necesidad de haber indicado una quota. Si no se asigna un tablespace por defecto al usuario, se toma el tablespace de SYSTEM (¡menudo peligro en un entorno real!).

5.2 Listar usuarios

Para obtener información de todos los usuarios creados en la BD existen las vistas: **ALL_USERS**, y **DBA_USERS**.

```
DESC SYS.ALL_USERS;
```

Esta sentencia muestra la estructura de la vista ALL_USERS.

Ejemplo:

```
SELECT * FROM ALL_USERS;
```

muestra los usuarios. Parte de los que visualiza:

USERNAME	USER_ID	CREATED
SYS	0	12/05/02
SYSTEM	5	12/05/02
OUTLN	11	12/05/02
WMSYS	19	12/05/02

```
DESC SYS.DBA_USERS;
```

muestra la estructura de la vista DBA_USERS. (conectado como SYSTEM)

La siguiente consulta:

```
SELECT username, DEFAULT_TABLESPACE FROM DBA_USERS
```

muestra el usuario y el tablespace por defecto de cada usuario.

USERNAME	DEFAULT_TABLESPACE
SYS	SYSTEM
SYSTEM	SYSTEM

OUTLN	SYSTEM

5.3 Modificación de usuarios

Sintaxis:

```
ALTER USER nombreusuario
IDENTIFIED BY clave_acceso
[DEFAULT TABLESPACE espacio_tabla]
[TEMPORARY TABLESPACE espacio_tabla].
[QUOTA {entero {K|M} | UNLIMITED} ON espacio_tabla]
[PROFILE perfil];
```

Puede cambiarse la clave de acceso de un usuario y necesita el privilegio ALTER USER para cambiarse el tablespace, la cuota o el perfil.

5.4 Borrado de usuarios

La orden DROP USER usuario [CASCADE] borra el usuario y todos sus objetos. Para borrar un usuario que tenga objetos creados necesitamos la opción [CASCADE].

Ejemplo:

```
SELECT owner, table_name
FROM dba_tables
WHERE owner = 'PEDRO';
```

nos da las tablas de Pedro.

Para borrar el usuario Pedro y todos sus objetos (los que lista la consulta de arriba) se utilizaría la siguiente sentencia:

```
DROP USER Pedro CASCADE;
```

Borra a Pedro y además sus objetos (tablas,vistas, ...).

6. PRIVILEGIOS

Un privilegio es la capacidad de un usuario dentro de la base de datos para realizar determinadas operaciones o a acceder a determinados objetos de otros usuarios.

En Oracle existen dos tipos de privilegios:

- **Privilegios del Sistema**: permite al usuario hacer ciertas tareas sobre la BD, como por ejemplo crear un Tablespace. Estos permisos son otorgados por el administrador o por alguien

que haya recibido el permiso para administrar ese tipo de privilegio. Existen como 100 tipos distintos de privilegios de este tipo.

En general los permisos de sistema, permiten ejecutar comandos del tipo DDL, como CREATE, ALTER y DROP o del tipo DML. Oracle 11g tiene más de 170 privilegios de sistema los cuales pueden ser vistos consultando la vista: **SYSTEM_PRIVILEGE_MAP.**

Entre todos los privilegios de sistema que existen, hay dos que son los importantes: **SYSDBA** y **SYSOPER**. Estos son dados a otros usuarios que serán administradores de base de datos.

- **Privilegios sobre Objetos:** Este tipo de privilegio le permite al usuario realizar ciertas acciones en objetos de la BD, como una Tabla, Vista, etc. Si a un usuario no se le dan estos permisos sólo puede acceder a sus propios objetos.

Este tipo de permisos los da el dueño del objeto, el administrador o alguien que haya recibido este permiso explícitamente (con la opción WITH GRANT OPTION).

La palabra clave **ANY** en los privilegios significa que los usuarios cuentan con el privilegio en cada esquema.

El comando **GRANT** agrega un privilegio a un usuario o grupo de usuarios.

El comando **REVOKE** elimina los privilegios.

Cuando se crea un usuario, es necesario darle privilegios para que pueda hacer algo. Oracle ofrece varios **roles** (***conjunto de privilegios***), Entre ellos tenemos: **CONNECT**, **RESOURCE**, **DBA**, **EXP_FULL_DATABASE** y **IMP_FULL_DATABASE** (los dos últimos ofrecen derechos de exportar o importar la base de datos completa).

El siguiente cuadro recoge los privilegios de alguno de los ROLES más importantes de oracle: *(estos roles incluyen privilegios de sistema).*

ROL	PRIVILEGIOS
CONNECT	Alter session, create cluster, create database link, create sequence, create session, create synonym, create table y create view.
RESOURCE	Create cluster, create procedure, create table, create sequence y create trigger.
DBA	Posee todos los privilegios del sistema
EXP_FULL_DATABASE	Select any table, backup any table, insert, update, delete sobre las tablas sys.incvid, sys.incfil y sys.incexp
IMP_FULL_DATABASE	Become user.

6.1 Privilegios sobre objetos

Estos privilegios nos permiten acceder y realizar cambios en los datos de otros usuarios y son los que se ofrecen en la tabla siguiente:

PRIVILEGIO SOBRE OBJETOS	TABLA	VISTA	SECUENCIA	PROCEDURE	SETENCIAS SQL PERMITIDAS EN CADA CASO
ALTER	X		X		ALTER
DELETE	X	X			DELETE
EXECUTE				X	EXECUTE
INDEX	X				CREATE INDEX ON
INSERT	X	X			INSERT INTO
REFERENCES	X				CREATE o ALTER TABLE
SELECT	X	X	X		SELECT
UPDATE	X	X			UPDATE

Sintaxis:

```
GRANT { priv_objeto, [priv_objeto], .... | ALL [PRIVILEGES]}
[(columna, [columna],....)]
ON [usuario.]objeto
TO {usuario|rol|PUBLIC} [,{usuario|rol|PUBLIC}...]
[WITH GRANT OPTION];
```

donde:

- **ON** designa los objetos donde se conceden privilegios.
- **TO** identifica usuarios o roles a los que se concede privilegios.
- **ALL** concede todos los privilegios sobre los objetos.
- **WITH GRANT OPTION** concede al usuario receptor el privilegio de concederlo/s a otros usuarios.
- **PUBLIC** incluye todos los usuarios actuales y futuros.

Ejemplos:

```
GRANT SELECT, INSERT ON TABLA1 TO FRANCISCO;

GRANT ALL ON TABLA2 TO JUAN;

GRANT UPDATE(TEMPERATURA) ON TABLA1 TO JUAN;
```

6.2 Privilegios del sistema

Los privilegios del sistema son los que dan derecho a ejecutar un tipo de comando SQL o a realizar alguna acción sobre objetos de un tipo especificado.

De todos los privilegios existentes éstos son algunos de ellos:

AUDIT	AUDIT ANY	Auditar un objeto
CLUSTER	CREATE CLUSTER CREATE ANY CLUSTER ALTER ANY CLUSTER DROP ANY CLUSTER	Crear cluster es propio esquema Crear cluster en cualquier esquema.
DATABASE	ALTER DATABASE CREATE DATABASE LINK CREATE PUBLIC DATABASE LINK	 Crear links privados para acceder a otra BD Crear links publicos para acceder a otra BD.
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX	Crear, modificar o borrar un índice de la base de datos (esquema o tabla)
PRIVILEGE	GRANT ANY PRIVILEGE	Conceder cualquier privilegio del sistema.
PROCEDURE	CREATE PROCEDURE CREATE ANY PROCEDURE ALTER ANY PROCEDURE DROP ANY PROCEDURE EXECUTE ANY PROCEDURE	Crear proced. almacenados, funciones y paquetes en esquema propio. Crear, modificar o borrar proced. almacenados, funciones y paquetes en cualquier esquema. Ejecutar proced. almacenados, funciones y paquetes en cualquier esquema.
PROFILE	CREATE PROFILE, ALTER PROFILE, DROP PROFILE	Crear un perfil de usuario, Modificar cualquier perfil, Borrar cualquier perfil

ROLE	CREATE ROLE, ALTER ANY ROLE, DROP ANY ROLE, GRANT ANY ROLE	Crea roles, Modifica cualquier role, Borra cualquier role, Dar permisos para cualquier rol.
ROLLBACK_SEGMENT	CREATE,ALTER,DROP	
SEQUENCE	CREATE SEQUENCE, ALTER ANY SEQUENCE, DROP ANY SEQUENCE, SELECT ANY SEQUENCE	Crear sequencias en nuestro esquema. Modifcar cualquier secuencia. Borrar secuencias de cualquier esquema Referenciar secuencias de cualquier esquema
SESSION	CREATE SESSION, ALTER SESSION; RESTRICTED SESSION	Conectarnos a la base de datos, Manejar la orden ALTER SESSION Conectarnos a la base de daros cuando se ha levantado con STARTUP RESTRICT
SYNONYM	CREATE SYNONYM, CREATE PUBLIC SYNONYM, DROP PUBLIC SYNONYM, CREATE ANY SYNONYM, DROP ANY SYNONYM	Crear sinónimos en nuestro esquema. Crear sinónimos públicos Borrar sinónimos públicos Crear sinónimos en cualquier esquema Borrar sinónimos en cualquier esquema
TABLE	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE LOCK ANY TABLE SELECT ANY TABLE INSERT ANY TABLE	Crear tabla en nuestro esquema Crear una tabla en cualquier esquema Modificar tabla en cualquier esquema Borrar tabla en cualquier esquema Bloquear tabla en cualquier esquema Hacer SELECT en cualquier esquema Insertar tabla en cualquier esquema

	UPDATE ANY TABLE DELETE ANY TABLE	Modificar tabla en cualquier esquema Borrar tabla en cualquier esquema
TABLESPACES	CREATE TABLESPACE, ALTER TABLESPACE, MANAGE TABLESPACE, DROP TABLESPACE, UNLIMITED TABLESPACE	Crear espacios de tablas Modificar tablespaces Poner on_line o off_lines a tablespaces Eliminar tablespaces Utilizar cualquier espacio de cualquier tablespaces
USER	CREATE USER, ALTER USER, DROP USER	Crear usuarios Modificar cualquier usuario. Puede modificar contraseña de otro usuario Eliminar usuarios
VIEW	CREATE VIEW, CREATE ANY VIEW, DROP ANY VIEW	Crear vistas en el esquema propio Crear vistas en cualquier esquema Borrar vistas en cualquier esquema
SYSDBA	SYSDBA	Ejecutar operaciones: STARTUP, SHUTDOWN, CREATE DATABASE, ALTER DATABASE, ARCHIVELOG Y RECOVERY
SYSOPER	SYSOPER	Ejecutar operaciones: STARTUP, SHUTDOWN, ALTER DATABASE, ARCHIVELOG Y RECOVERY

Si el privilegio lleva la palabra **ANY** se entiende que el privilegio se concede sobre cualquier esquema y si no la lleva solamente sobre el propio.

Sintaxis:

El formato de la orden **GRANT** para asignar privilegios del sistema es:

```
GRANT {privilegio|rol} [,{privilegio|rol},…]
TO {usuario|rol|PUBLIC} [,{usuario|rol|PUBLIC} ] ….
[WITH ADMIN OPTION];
```

Ejemplos:

```
CREATE USER PEDRO IDENTIFIED BY PEDRO
DEFAULT TABLESPACE USER_DATA;
GRANT CREATE SESSION TO PEDRO; para que Pedro cree sesión.
GRANT CONNECT TO PEDRO;
```

Se concede a Pedro el rol **CONNECT** que conlleva los siguientes privilegios (alter session, create cluster, create database link, create sequence, create session, create synonym, create table y create view).

```
GRANT DBA TO PEDRO, JUAN;
GRANT DROP USER TO MILAGROS WITH ADMIN OPTION;
```

La última línea permite otorga al usuario MILAGROS el permiso de borrar usuarios y poder conceder este privilegio a otros usuarios. Cuando se otorga un privilegio del sistema a un usuario, se puede añadir la cláusula **WITH ADMIN OPTION**, con esto, el usuario puede ceder a terceros el privilegio de sistema obtenido. Con este ejemplo, concedemos permiso al usuario MILAGROS para ceder a terceros el privilegio de borrar usuarios.

```
GRANT SELECT ANY TABLE TO PUBLIC;
```

Cualquier usuario puede hacer select en cualquier tabla.

La vista **DBA_SYS_PRIVS** contiene los privilegios del sistema que se han otorgado a los usuarios y a los roles.

La vista **SESSION_PRIVS** contiene los privilegios disponibles para la sesión actual para un usuario.

6.3 Retirada de privilegios

Para revocar un privilegio del sistema o roles a usuarios o para retirar privilegios de sistema a roles se utiliza:

```
REVOKE (system_priv | role) [, (system_priv | role) ] ...
FROM    (user | role | PUBLIC) [, (user | role | PUBLIC) ] ...
```

Para retirar privilegios sobre objetos o roles a usuarios o para retirar privilegios concedidos a los roles la orden es REVOKE y el formato es :

```
REVOKE {priv_objeto [,priv_objeto]....|ALL [PRIVILEGES]}
ON [usuario.]objeto
FROM {usuario|rol|PUBLIC} [,{usuario|rol|PUBLIC}] ...;
```

Ejemplo:

```
REVOKE UPDATE ON TABLE1 FROM PEDRO;
REVOKE ALL  ON TABLA1 FROM FRANCISCO;
REVOKE SELECT ANY TABLE FROM PUBLIC;
```

```
REVOKE DBA FROM PEDRO;
```

7. ROLES

Los roles son grupos específicos de privilegios relacionados que se otorgan a los usuarios o a otros roles. Están diseñados para facilitar la gestión de los privilegios en la base de datos.

- Se otorgan y se revocan a los usuarios con los mismos comandos que se utilizan para otorgar y revocar los privilegios del sistema
- Pueden estar formados tanto por privilegios del sistema como por objetos.
- Se pueden activar o desactivar para cada usuario al que se le haya otorgado el rol.
- Pueden necesitar una contraseña para activarse.
- El nombre de cada rol debe ser único entre los nombres de usuario y nombre de rol existentes.
- No lo posee nadie, no están en ningún esquema.
- Sus descripciones se almacenan en el diccionario de datos.

7.1 Creación de roles

```
CREATE ROLE  rol [NOT IDENTIFIED | IDENTIFIED BY password )
```

Identified by password indica que el usuario que desee usar los privilegios tiene que introducir la clave de acceso en la orden SET ROLE para activar el rol.

7.2 Modificación de roles

```
ALTER ROLE  role [NOT IDENTIFIED | IDENTIFIED BY password );
```

7.3 Conceder privilegios a los roles

Una vez creado, hemos de concederle privilegios usando la orden

```
GRANT privilegios ON tabla TO rol;
```

7.4 Asignación de roles

```
GRANT rol [, rol] ...
 TO (user | rol | PUBLIC) [, (user | rol | PUBLIC)] ...  [WITH  ADMIN  OPTION];
```

7.5 Activación y desactivación de roles

```
SET ROLE (rol [ IDENTIFIED BY password]
[, rol ( IDENTIFIED BY password)] ... | ALL [ EXCEPT rol [, rol]...]        | none];
```

- **ALL [EXCEPT role [, role]...]**: activa todos los roles excepto…
- **none**: desactiva todos los roles

7.6 Eliminación de roles de usuario

```
REVOKE role [, role]...
FROM    (user | role | PUBLIC) [, (user | role | PUBLIC) ]...;
```

7.7 Eliminación de roles

```
DROP ROLE role;
```

8. PERFILES

Un perfil es un conjunto de límites a los recursos de la BD, que se usa para poner límites al uso de recursos de la BD por parte de los usuarios, como por ejemplo el tiempo de conexión.

Por omisión se asigna el perfil DEFAULT cuando se da de alta al usuario. Este perfil supone recursos ilimitados (UNLIMITED).

Los recursos son:

SESSIONS_PER_USER	Número de sesiones múltiples concurrentes permitidas por nombre de usuario
CPU_PER_SESSION	Tiempo máximo de CPU por sesión en centésimas de segundo
CPU_PER_CALL	Tiempo máximo de CPU por llamada en centésimas de segundo
CONNECT_TIME	Tiempo de conexión en minutos.
IDLE_TIME	Tiempo de inactividad antes de que el usuario sea desconectado
LOGICAL_READS_PER_SESSION	Bloques de datos leídos en una sesión.
LOGICAL_READS_PER_CALL	Bloques de datos leídos por llamada.
PRIVATE_SGA	Bytes enteros de espacio en SGA (área SQL compartida que se usa para la opción servidor compartido).
COMPOSITE_LIMIT	Coste total de recursos en una sesión expresado como una suma ponderada de “unidades de servicio”.
FAILED_LOGIN_ATTEMPTS	Número de intentos de acceso sin éxito consecutivos que producirá el bloqueo de la

	cuenta
PASSWORD_LIFE_TIME	Número de días que puede utilizarse una contraseña antes que caduque
PASSWORD_REUSE_TIME	Número de días que deben pasen antes de que se pueda utilizar una contraseña
PASSWORD_REUSE_MAX	Número de veces que debe cambiarse una contraseña antes de poder reutilizarla
PASSWORD_LOCK_TIME	Número de días que quedará bloqueada una cuenta si se sobrepasa el valor del parámetro FAILED_LOGIN_ATTEMPTS
PASSWORD_GRACE_TIME	La duración en días del periodo de gracia durante el cual una contraseña puede cambiarse cuando ha alcanzado su valor

Sintaxis:

CREATE PROFILE nombreperfil **LIMIT** recurso **{**Entero **[K | M] | UNLIMITED | DEFAULT}**

[,recurso { }]

Los recursos posibles son los anteriores listados en la tabla.

Con **UNLIMITED** se entiende que no hay límite y **DEFAULT** recoge el valor para el perfil **DEFAULT**.

Para activar el uso de perfiles en el sistema, el administrador ha de ejecutar

ALTER SYSTEM SET RESOURCE_LIMIT = TRUE | **FALSE**

activa o desactiva el uso de perfiles en el sistema.

DROP PROFILE perfil;

borra el perfil.

Pasos para crear un perfil:

1. Crear el perfil. **CREATE PROFILE**
2. Crear el usuario. **CREATE USER**
3. Conceder al usuario como mínimo el rol **CONNECT**. **GRANT CONNECT**
4. Activar el uso de perfiles en el sistema, el administrador ha de ejecutar **ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;**
5. A partir de ahora ya se puede conectar el nuevo usuario utilizando el nuevo perfil