C.F.G.S. DESARROLLO DE APLICACIONES WEB

DISTANCIA

MÓDULO:

BASES DE DATOS

Unidad 6:

Ampliación de SQL

Indice

1.- Vistas.

Las vistas definen una tabla virtual basada en una o más tablas o vistas. Esta tabla virtual se almacena permanentemente en la base de datos, generando, al igual que las tablas, una entrada en el diccionario de datos.

Las vistas permiten organizar los datos de diferentes formas, de modo que los usuarios los ven desde diferentes perspectivas. De igual modo, se restringe el acceso a los datos, permitiendo que determinados usuarios a través de las vistas sólo puedan acceder a determinadas filas y columnas de las tablas.

Desde el punto de vista del usuario, la vista es como una tabla real, con estructura de filas y columnas; pero a diferencia de éstas sus datos no se almacenan físicamente en la base de datos. Las filas y columnas visibles para el usuario son consecuencia de la consulta que se realiza en las tablas reales sobre las que se definió la vista.

Una vista es una sentencia SQL. La consulta que define la vista permite los operadores relacionales y los de conjunto que se han estudiado en los temas anteriores. Así podemos aplicar a las vistas los operadores de restricción, proyección, reunión, división. También podemos utilizar los operadores tradicionales de conjuntos: unión, intersección y diferencia.

- **Ventajas de las Vistas.**
  - Las vistas restringen el acceso a los datos porque muestra las columnas seleccionadas de la tabla.
  - Las vistas se pueden utilizar para crear consultas simples para recuperar los resultados de consultas complicadas. Por ejemplo, las vistas se pueden utilizar para consultar información de varias tablas sin que el usuario sepa cómo escribir una sentencia de unión.
  - Las vistas proporcionan independencia de los datos para usuarios y programas de aplicación.
  - Las vistas proporcionan a los grupos de usuarios acceso a los datos según sus criterios concretos.

1.1.- Creación de una vista.

Para crear una vista utilizamos la sentencia CREATE VIEW, que su formato tiene la siguiente estructura:

```
CREATE [OR REPLACE]  VIEW  nombre_vista  (nom_col [,nom_col])]  AS consulta_sql;
```

Dónde:

OR REPLACE: crea de nuevo la vista si ya existía.

nombre_vista: nombre de la vista

nom_col: nombre de las columnas de la vista, si no se especifican se adoptarán los mismos que los de la consulta

consulta: determina las columnas y las tablas que aparecerán en la vista

Se puede hacer una clasificación de las vistas atendiendo al tipo de consulta.

- Si la vista se crea a partir de determinadas filas de la tabla fuente pero con todas sus columnas, obtenemos una **vista por restricción**.

Ejemplo-1:

Creación de una vista de la tabla empleado, con todos los empleados que no tengan comisión.

```
CREATE VIEW sinComision AS SELECT * FROM empleado WHERE comision IS NULL;
```

- Si construimos la vista con todas o parte de las filas pero con algunas de las columnas de la tabla fuente tendríamos una **vista por proyección**.

Ejemplo-2:

Creación de una vista a partir de una consulta solo de los empleados del departamento 30.

```
CREATE VIEW dep30A AS
        SELECT apell, oficio, salario, depnume FROM  empleado WHERE depnume =30;

CREATE VIEW dep30B (apellido, oficio, salario, numerodep) AS
        SELECT apell, oficio, salario, depnume FROM  empleado WHERE depnume =30;
```

- Otra construcción posible es la que procede de agrupaciones de filas de la tabla original GROUP BY, se denomina **vista agrupada**, ya que sus filas están construidas por agrupaciones de la tabla fuente. En este caso hay que especificar las columnas que se tienen que corresponder con las de la consulta.

Ejemplo-3:

Creación de una vista con el numero de departamento y salario de los empleados que tienen comisión y agrupados por numero de departamento.

```
CREATE VIEW sumaSalarios (numerodep, salario) AS
        SELECT depnume, sum (salario) FROM empleado
                WHERE comision IS NULL  GROUP BY depnume;
```

Al crear una vista podemos dejar que herede los nombres de las columnas de las expresiones que aparecen detrás de la SELECT o darles nosotros nombre.

- Las vistas no tienen por qué definirse sobre una sola tabla, pueden estarlo sobre varias tablas. Estas vistas se llaman **vistas compuestas**.

Ejemplo-4:

Creación de una vista con los nombres de los empleados y el nombre del departamento al que pertenecen, excluyendo aquellos empleados cuyo salario es inferior a 2000.

```
CREATE VIEW empledept AS
        SELECT apell, nombredep FROM empleado, departamento
                WHERE empleado.depnume= departamento.numedep AND salario > 2000;
```

Recordar:

Un usuario solo podrá crear vistas si tiene el privilegio concedido

La vista no es una tabla, sino que contiene la consulta que se ejecutara siempre que se invoque dicha vista.

1.2.- Acceso a las vistas.

Para consultar las vistas creadas se dispone de la vista **USER_VIEWS:**

```
SELECT * FROM USER_VIEWS;
```

Si queremos seleccionar solo el nombre de la vista y los campos que la forman, deberemos visualizar solo las columnas de la vista:

```
SELECT VIEW_NAME, TEXT FROM USER_VIEWS
```

Dónde:

VIEW_NAME: muestra el nombre de la vista
TEXT: muestra la sentencia que permite obtener la vista

El usuario accede a los datos de una vista exactamente igual que si estuviese haciéndolo a una tabla. De hecho, en muchos casos el usuario no sabrá siquiera que está consultando una vista. En general, el uso de las vistas simplifica las consultas a la base de datos.

Las operaciones a realizar son las mismas que las que se llevan a cabo sobre tablas:

- SELECT
- INSERT
- UPDATE
- DELETE

> **NOTA**: Las operaciones sobre una vista realmente se están realizando sobre la(s) tabla(s)

Las consultas actúan sobre las vistas igual que sobre las tablas reales, por lo tanto podemos trabajar con las mismas cláusulas de la SELECT que se describieron en las consultas sobre tablas.

Ejemplo-5:

Queremos consultar la vista que creamos en el ejemplo-4

```
SELECT * FROM empledept;
```

1.3.- Actualización de vistas.

Para que una vista se pueda actualizar debe de existir una relación directa entre las filas y las columnas de la vista y las de la tabla fuente.

En el **ejemplo-1** habíamos creado la siguiente vista.

```
CREATE VIEW sinComision AS SELECT * FROM empleado WHERE comision IS NULL;
```

En este caso sí se pueden realizar inserciones, borrados y actualizaciones, ya que la vista sinComision tiene la misma estructura que la tabla fuente, luego las actualizaciones que se realicen sobre ella se realizan sobre la tabla fuente.

No se pueden actualizar las vistas que para su construcción se hayan incluido en las consultas las cláusulas:

- GROUP BY o HAVING
- En la WHERE subconsultas
- En las columnas que tienen expresiones o funciones
- En la FROM varias tablas
- La cláusula DISTINCT

Estas reglas no son aplicables a todos los Sistemas Gestores de Bases de Datos. Se consideran reglas generales reguladas por ANSI/ISO, pero cada SGBD tiene sus particularidades y por lo tanto se debe estudiar en cada caso.

Ejemplo-6

Vamos a insertar una fila en la vista **sinComision** (vista que creamos en el ejemplo-1)

```
INSERT INTO sinComision (numemp,apell,salario,depnume) VALUES (9999,'BALDOMERO', 9000,10);
```

Si ahora seleccionamos las filas de la tabla fuente

```
SELECT numemp, apell, salario, depnume FROM empleado;
```

NUMEMP	APELL	SALARIO	DEPNUME
9999	BALDOMERO	9000	10
7369	SANCHEZ	1040	20
7499	ARROYO	2080	30
7521	SALA	1625	30
7566	JIMENEZ	3867	20
7654	MARTIN	1625	30
7698	NADAL	3705	30
7782	CEREZO	3185	10
7788	GIL	3900	20
7839	REY	6500	10
7844	TOVAR	1950	30
7876	ALONSO	1430	20
7900	JIMENO	1235	30
7902	FERNANDEZ	3900	20
7934	MUÑOZ	1690	10

Vemos que insertó la fila a través de la vista.

Si consultamos la vista

```
SELECT numemp, apell, salario, depnume FROM sinComision;
```

NUMEMP	APELL	SALARIO	DEPNUME
9999	BALDOMERO	9000	10
7369	SANCHEZ	1040	20
7566	JIMENEZ	3867	20
7698	NADAL	3705	30
7782	CEREZO	3185	10
7788	GIL	3900	20
7839	REY	6500	10
7876	ALONSO	1430	20
7900	JIMENO	1235	30
7902	FERNANDEZ	3900	20
7934	MUÑOZ	1690	10

Ejemplo-7

Vamos a intentar insertar una fila en la vista **empledept** (creada en el ejemplo-4).

```
INSERT INTO empledept VALUES ('Baldomero', 'Contabilidad');
```

El Sistema visualiza el siguiente mensaje:

```
ORA-01776: no se puede modificar más de una tabla base a través de una vista de unión
```

El error se debe a la imposibilidad de actualizar las tablas fuente cuando la vista se construye con una selección multitabla.

Cuando hacemos una inserción en una vista definida sobre una sola tabla pero que no coge todas sus columnas, sucede que:

- Las columnas de la tabla que no están en la vista y tienen la restricción DEFAULT asumen el valor por defecto.
- Las columnas de la tabla que no están en la vista y no tienen la restricción DEFAULT, asumen nulos.
- Si las columnas de la tabla que no están en la vista tienen la restricción NOT NULL y no la DEFAULT no se inserta la fila.

1.4.- Clausulas de CREATE VIEW.

Vamos a ver algunas de las opciones que se pueden utilizar cuando se crea una vista.

- **Cláusula CHECK OPTION**: Especifica que solo registros accesibles a la vista puedan ser insertados o actualizados.
  Hemos visto que las vistas son actualizables cuando existe una relación directa entre las filas y columnas de la vista con las de la tabla fuente, sin embargo podemos observar que si insertamos filas con una condición, la tabla fuente se actualiza con dichas filas, pero si la consulta que define la vista no satisface dicha condición para las filas insertadas, éstas no son visibles a través de la vista.
  Dicho de otra forma, cuando se define una vista simple por restricción, a través de la vista solo son visibles aquellas filas de la tabla base que cumplen la restricción.

Ejemplo-8.

Veamos el contenido de la tabla empleado

```
SELECT numemp, apell, salario,depnume FROM empleado
```

NUMEMP	APELL	SALARIO	DEPNUME
9999	BALDOMERO	9000	10
7369	SANCHEZ	1040	20
7499	ARROYO	2080	30
7521	SALA	1625	30
7566	JIMENEZ	3867	20
7654	MARTIN	1625	30
7698	NADAL	3705	30
7782	CEREZO	3185	10
7788	GIL	3900	20
7839	REY	6500	10
7844	TOVAR	1950	30
7876	ALONSO	1430	20
7900	JIMENO	1235	30
7902	FERNANDEZ	3900	20
7934	MUÑOZ	1690	10

Creamos una vista con los empleados cuyo salario exceda de 2000.

```
CREATE VIEW vistaemple AS SELECT numemp, apell, salario, depnume FROM empleado
        WHERE salario > 2000;
```

Visualizamos la vista

```
SELECT * FROM vistaemple;
```

NUMEMP	APELL	SALARIO	DEPNUME
9999	BALDOMERO	9000	10
7499	ARROYO	2080	30
7566	JIMENEZ	3867	20
7698	NADAL	3705	30
7782	CEREZO	3185	10
7788	GIL	3900	20
7839	REY	6500	10
7902	FERNANDEZ	3900	20

Tal como fue definida esta vista, es posible insertar filas que luego no sean visibles a través de ella misma.

Insertamos una fila con salario inferior a 2000

```
INSERT INTO vistaemple VALUES (9000,'LAURA', 1000,20);
```

Si ahora realizamos una consulta de la **vista** comprobamos que la fila insertada **NO** es visible

```
SELECT * FROM vistaemple;
```

NUMEMP	APELL	SALARIO	DEPNUME
9999	BALDOMERO	9000	10
7499	ARROYO	2080	30
7566	JIMENEZ	3867	20
7698	NADAL	3705	30
7782	CEREZO	3185	10
7788	GIL	3900	20
7839	REY	6500	10
7902	FERNANDEZ	3900	20

Sin embargo la fila **SI** está en la tabla fuente.

Lo comprobamos:

```
SELECT numemp, apell, salario,depnume FROM empleado
```

NUMEMP	APELL	SALARIO	DEPNUME
9999	BALDOMERO	9000	10
9000	LAURA	1000	20
7369	SANCHEZ	1040	20
7499	ARROYO	2080	30
7521	SALA	1625	30

Con la cláusula **CHECK OPTION** se asegura que las operaciones de inserción y actualización sobre la vista satisfagan el criterio de búsqueda en la WHERE de la definición de la vista.

De esta forma se asegura que las filas actualizadas o insertadas van a ser visibles a través de la vista.

Ejemplo-9

Creamos de nuevo la vista con la opción CHECK OPTION

```
CREATE OR REPLACE VIEW vistaemple AS SELECT numemp, apell, salario, depnume
        FROM empleado WHERE salario > 2000 WITH CHECK OPTION;
```

Si ahora intentamos insertar en la vista una fila con salario menor de 2000 no nos dejará.

```
INSERT INTO vistaemple VALUES (9000,'LAURA', 1000,20);
```

```
ORA-01402: violación de la cláusula WHERE en la vista WITH CHECK OPTION
```

- **Cláusula OR REPLACE.**

  Cuando se necesitan modificar la definición de una vista, la única solución es crear una nueva vista con el mismo nombre, ya que no existe una sentencia parecida a la **ALTER TABLE** que permitía modificar la definición de una tabla.

  La cláusula **OR REPLACE** de la sentencia **CREATE VIEW** permite sustituir la definición de una vista por una nueva definición

```
CREATE OR REPLACE VIEW nombre_vista  (nom_col [,nom_col])]   AS  consulta_sql;
```

- **Cláusula WITH READ ONLY**

  Esta cláusula asegura que no se realicen operaciones DML a través de vistas. Es decir no se permiten borrados, inserciones o actualizaciones a través de la vista.

1.5.- Eliminación de Vistas.

Las vistas se eliminan con la sentencia DROP, la cual tiene el mismo formato que el utilizado para eliminar tablas.

```
DROP VIEW nombre_de_la_vista
```

Si la vista está en otro esquema y tenemos derechos, podemos borrarla anteponiéndole el nombre del usuario.

```
DROP VIEW nombre_usuario.nombre_vista
```

NOTA: Recordar que si se elimina una tabla utilizada para obtener una vista, esta última seguirá existiendo pero quedará inutilizada.

2.- Sinónimos.

Un sinónimo es un nombre alternativo que se puede dar a una tabla o una vista, permitiendo de esta forma abreviar a la hora de escribir el acceso a la misma.

2.1.- Creación de sinónimos.

El formato para la creación de sinónimos es el siguiente:

```
CREATE [PUBLIC] SYNONYM nombre_del_sinónimo
        FOR [usuario].{nombre_tabla | nombre_vistavista};
```

Public: hace que el sinónimo esté disponible para todos los usuarios. Solo puede ser usado por el DBA o usuarios con el privilegio CREATE PUBLIC SYNONYM.

Los sinónimos pueden ser utilizados en las sentencias de manipulación **INSERT, UPDATE, DELETE, SELECT** y por las de control **GRANT y REVOKE**.

Los sinónimos proporcionan independencia para tratar los objetos (tablas, vistas, procedimientos, etc) ya que podemos acceder a objetos de otros propietarios o bases de datos remotas sin necesidad de especificar el propietario ya que va implícito en el nombre del sinónimo.

Ejemplo-10

Suponiendo que el propietario de la tabla EMPLEADO sea UNIDAD4D, para hacer una consulta sobre dicha tabla desde otro usuario, que tenga los correspondientes derechos, haríamos lo siguiente:

```
SELECT * FROM unidad4D.empleado;
```

Si creamos el siguiente sinónimo

```
CREATE SYNONYM conEmp FOR unidad4D.empleado;
```

Podemos realizar la consulta de la siguiente forma

```
SELECT * FROM conEmp;
```

También el administrador podría crear el siguiente sinónimo público:

```
CREATE PUBLIC SYNONYM conEmpl FOR unidad4D.empleado;
```

2.2.- Eliminación de Sinónimos.

Para eliminar, sinónimos, se utiliza la sentencia **DROP**, al igual que para borrar tablas.

Cuando se utiliza esta sentencia, la definición de los objetos se pierde junto con su contenido, y no hay forma de recuperarlo.

El formato seria:

```
DROP [PUBLIC] SYNONYM [usuario]. nombre_del_sinónimo;
```

Al igual que antes, solo el **DBA** y los usuarios con el privilegio **DROP PUBLIC SYNONYM** pueden suprimir sinónimos públicos.

Los usuarios con el privilegio **DROP ANY SYNONYM** pueden borrar los sinónimos de otros usuarios

2.3.- Consulta de sinónimos.

Para ver los sinónimos, existe una vista **USER_SYNONYMS** que permite ver los sinónimos que son propiedad del usuario.

Ejemplo-11

Si estamos conectados con el usuario UNIDAD4D y queremos saber todos los sinónimos

```
SELECT * from USER_SYNONYMS
```

Resultados | Explicar | Describir | SQL Guardado | Historial

SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
CONEMP	UNIDAD4D	EMPLEADO	-

1 filas devueltas en 0,00 segundos Exportación de CSV

2.4.- Renombrar sinónimos.

La sentencia RENAME permite cambiar el nombre a una tabla, vista o un sinónimo.

Su formato es:

```
RENAME nombre_viejo TO nombre_nuevo;
```

Oracle invalida todos los objetos que dependan del objeto renombrado, como las vistas, los sinónimos que hacen referencia a la tabla renombrada.

Ejemplo-12

Creamos el sinónimo **depto** de la tabla **departamento**

```
CREATE SYNONYM  depto.  FROM departamento;
```

Si renombramos la tabla departamento

```
RENAME departamento TO  tabladepto;
```

El sinónimo **depto** deja de funcionar

3.- Índices.

Los índices se crean para disminuir el tiempo de respuesta a una consulta. Pueden afectar brutalmente al rendimiento, para bien o para mal. Se definen sobre una columna o conjunto de columnas de una tabla concreta.

Los SGBD suelen crear automáticamente un índice para las columnas que forman la clave primaria o si hay restricción de unicidad, es decir cuando se establecen **CONSTRAINS** de tipo **PRIMARY KEY** o **UNIQUE.**

Crear un índice es muy sencillo, lo difícil es saber cuándo hacerlo.

Es recomendable crearlo si:

- El número de inserciones, modificaciones y borrados es despreciable frente al número de consultas por esa columna.
- La tabla es muy grande.
- Hay muchas consultas por una columna que no es clave primaria.

NO es recomendable crearlo si:

- Hay muchos valores repetidos o nulos en la columna.
- La tabla es muy pequeña.
- La tabla tiene muchas inserciones, modificaciones y borrados en proporción al número de consultas por esa columna.

3.1.- Creación de índices.

La sintaxis más simple para crear un índice es la siguiente:

```
CREATE INDEX nombreindice
        ON nombretabla (listacolumnas[ASC|DESC]);
```

El nombre del índice crea una entrada en el diccionario de datos. El número de columnas que se puede indexar por tabla dependerá de la base de datos con la que estemos trabajando. Los nulos no son indexados.
Los índices aceleran las búsquedas pero hacen más lentas las actualizaciones, ya que ellos mismos deben actualizarse.

3.2.- Consulta de índices.

Las vistas del diccionario de datos con información sobre los índices son las siguientes:

```
DBA_INDEXES

DBA_IND_COLUMNS
```

3.3.-Borrado de índices.

Para borrar un índice, se utiliza el siguiente formato:

```
DROP INDEX nombreindice ON nombretabla(listacolumnas);
```

4.- LCD. Lenguaje de Control de Datos.

Todo acceso a una base de datos requiere conectar mediante un usuario y una contraseña. Dicho usuario dará derecho a utilizar ciertos objetos de la base de datos, pero se puede restringir el uso de otros.

A los usuarios se les asigna una serie de privilegios que son los que dan permiso de uso a ciertos objetos. Para organizarse mejor la mayoría de los Sistemas Gestores de Bases de Datos permiten agrupar permisos que normalmente se aplican conjuntamente en estructuras llamadas perfiles y roles, que en definitiva son un conjunto de permisos.

Cuando un usuario se conecta debe probar que es quien dice ser (normalmente con un nombre de usuario y una contraseña) es decir se autentifica. Por otro lado esta autentificación dará lugar a unos privilegios: unos derechos, y unas restricciones.

El Lenguaje de Control de Datos - DCL (Data Control Language), es la parte de SQL que nos permite gestionar:

- El acceso o gestión de usuarios
- La confidencialidad
- Las transacciones a la BD.

5.- Características de los usuarios de ORACLE.

A los usuarios de Oracle se les puede asignar la configuración referida a:

- **Nombre de usuario**:
  No puede repetirse y como máximo debe tener 30 caracteres que sólo podrán contener letras del alfabeto inglés, números, el signo dólar y el signo de guión bajo (_).
- **Configuración física**:
  Se refiere al espacio asociado al usuario para almacenar sus datos (lo que Oracle llama tablespace) y la cuota (límite de almacenamiento) que se le asigna.
- **Perfil asociado**:
  El perfil del usuario indica los recursos y configuración que tomará el usuario al sistema.
- **Privilegios y roles:**
  Permiten especificar las acciones que se le permiten realizar al usuario.
- **Estado de la cuenta de usuario:**
  - ✓ **Abierta**: El usuario puede conectar y realizar sus acciones habituales
  - ✓ **Bloqueada**: EL usuario no podrá conectar mientras siga en estado bloqueado. El bloqueo lo realiza el DBA:
    ALTER USER usuario **ACCOUNT LOCK**
  - ✓ **Expirada**: La cuenta agotó el tiempo máximo asignado a ella. Para salir de este estado, el usuario/a debe resetear su contraseña de usuario.
  - ✓ **Expirada y bloqueada**.
  - ✓ **Expirada en periodo de gracia**: Está en los últimos momentos de uso antes de pasar a estado de expirada

6.- Autentificación.

La autentificación define la forma en la que el usuario verifica quién es. Hay métodos de autentificación más seguros que otros. Asegurar la autentificación implicaría asegurar el medio de la comunicación entre usuario y base de datos con protocolos de cifrado. Por otro lado hay que proteger especialmente a los usuarios administradores.

- **Autentificación por el sistema operativo.**

  Se permite el uso sólo en usuarios con privilegios administrativos. En el sistema operativo en el que se instale Oracle se crean dos grupos de usuarios relacionados con los dos privilegios de sistema:

  - ✓ SYSDBA
  - ✓ SYSOPER.

  En Windows se llaman ORA_DBA y ORA_OPER respectivamente, en Linux simplemente dba y oper.

  Los usuarios de esos grupos conectarían mediante CONNECT/AS SYSDBA o CONNECT/AS SYSOPER.

  Otra posibilidad es conectar con: CONNECT /@servicioRed AS SYSDBA

- **Autentificación por archivo de contraseñas**

  Se usa en los mismos casos que la anterior. Cuando no se considera que el Sistema Operativo sea muy seguro, se utiliza como opción. Para usar esta forma de autentificación los usuarios de tipo SYSDBA o SYSOPER indican su nombre de usuario y contraseña al conectar (opcionalmente indican el host al que se desean conectar) esos datos se contrastarán con los del archivo de contraseñas utilizado. Esta forma (y la anterior) permite conectar la base de datos aunque no esté montada todavía la base de datos.

  La utilidad ORAPWD permite crear, si no existe el archivo de contraseñas:

  ORAPWD FILE= ruta [**ENTRIES**= n [**FORCE**=y | n[**IGNORECASE**=y|n]]]

  Funcionamiento:

  - ✓ **FILE**: Permite indicar el nombre del archivo de contraseñas
  - ✓ **ENTRIES**: Indica el máximo número de contraseñas que admitirá el archivo
  - ✓ **FORCE**: En caso de darle el valor y sobrescribe las contraseñas cuando asignemos una nueva a un usuario administrativo
  - ✓ **IGNORECASE**: No tiene en cuenta mayúsculas ni minúsculas en las contraseñas.

  Por otra lado el parámetro de sistema **REMOTE_LOGIN_PASSWORDFILE** (modificable con ALTER SYSTEM SET…), permite indicar la forma en la que funciona el archivo de contraseñas.
  Valores posibles:

  - ✓ **NONE**: No permite usar el archivo de contraseñas, las conexiones de usuarios con privilegios administrativos tendrán que usar otros métodos.
  - ✓ **EXCLUSIVE**. El archivo de contraseñas se usa sólo en la instancia actual.
  - ✓ **SHARED:** Se comparte el archivo de contraseñas entre varias instancias de tipo Real Application Cluster de Oracle (para bases de datos distribuidas). En este caso no se pueden cambiar las contraseñas de los usuarios administrativos.

- **Autentificación por contraseña.**

  En este caso los usuarios son autentificados mediante una contraseña que se contrastará en el diccionario de datos que es donde se almacenan estas contraseñas. Es el método habitual de autentificación para usuarios no administrativos. Está configuración requiere la base de datos montada y abierta (al tener que usar el diccionario de datos).

  La contraseña se pasa encriptada desde el ordenador cliente al servidor mediante el algoritmo **AES**.

- **Autentificación externa.**

  Oracle delega la autentificación a un servicio externo que se asociará a Oracle. Ejemplos de servicios externos son Kerberos o RADIUS, este último sólo disponible en Windows.

- **Autentificación global**

  Se trata de utilizar un servicio LDAP para realizar la autentificación. Oracle dispone de un servicio LDAP global integrado en Oracle Applications (plataforma de Oracle para le creación de aplicaciones) que se llama Oracle Internet Directory.

  Si los usuarios sólo están dados de alta en el directorio externo, usarán todos la misma cuenta de Oracle; para independizarles se requiere darles de alta en ambos servicios (Oracle y el Oracle Internet Directory).

7.- Control de Usuarios.

7.1.- Creación de Usuarios.

La sintaxis completa para crear usuarios es:

```
CREATE USER nombre_usuario
IDENTIFIED BY contraseña
[ DEFAULT     TABLESPACE   espacio_tabla ]
[ TEMPORARY   TABLESPACE   espacio-tabla ]
[ QUOTA { nume_entero {K | M} | UNLIMITED } ON espacio-tabla ]
[ PASSWORD EXPIRE ]
[ ACCOUNT {UNLOCK | LOCK }]
[ PROFILE  perfil ];
```

Si no se indican las cláusulas opcionales, coge las opciones por defecto. No hace falta especificar todas las opciones, solo las que nos interese.

Por ejemplo:

```
CREATE USER pepe IDENTIFIED BY 'pepin'
    DEFAULT TABLESPACE 'Usuarios'
    QUOTA 15M ON 'Usuarios' //Se dan 15MBytes de espacio en el tablespace
    ACCOUNT LOCK; //La cuenta estará bloqueada
```

7.2.- Modificación de usuarios.

Cada parámetro indicado anteriormente se puede modificar mediante la instrucción **ALTER USER** que se utiliza igual que **CREATE USER**.

Se debe tener privilegios para modificar usuarios o ser DBA de la base de datos.

Formato:

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

Ejemplo:

```
ALTER USER pepe QUOTA UNLIMITED ON usuarios
```

7.3.- Borrado de usuarios

Formato:

```
DROP USER nombre_usuario [CASCADE];
```

La opción **CASCADE** elimina los objetos del esquema del usuario antes de eliminar al propio usuario. Es obligatorio si el esquema contiene objetos.´

7.4.- Consultar usuarios

Para consultar los usuarios que tenemos las siguientes vistas del diccionario para usuarios

ALL_USERS: Permite ver todos los usuarios creados

DBA_USERS: para comprobar los tablespaces asignados a cada usuario (solo usuario administrador)

USER_USERS: para ver la información del usuario actual

8.- Privilegios.

Los privilegios son permisos que damos a los usuarios para que puedan realizar ciertas operaciones con la base de datos.

Es decir es la capacidad de un usuario de realizar determinadas operaciones y/o acceder a determinados objetos

Una vez creado un usuario es necesario darle privilegios para que pueda hacer algo:

- Privilegios específicos de una acción.
- Privilegios asociados a roles predefinidos en ORACLE.
- Privilegios asociados a roles definidos por el DBA.

- **Los privilegios asociados a los roles predefinidos en ORACLE**.

ROLES	PRIVILEGIOS
CONNECT	CREATE SESSION
RESOURCE	CREATE TRIGGER CREATE SECUENCE CREATE TYPE CREATE PROCEDURE CREATE CLUSTER CREATE OPERATOR CREATE INDEX TYPE CREATE TABLE

- **Los privilegios asociados a los roles predefinidos por el DBA:**

ROLES	PRIVILEGIOS
DBA	Todos los privilegios del sistema.
EXP_FULL_DATABASE	SELECT ANY TABLE BACKUP ANY TABLE INSERT UPDATE DELETE sobre tablas SYS.INCVID SYS.INCFILL SYS.INCEXP
IMP_FULL_DATABASE	BECOME USER

En Oracle hay más de cien posibles privilegios. Se dividen en:

- **Privilegios sobre los objetos**: Son permisos que se aplican a un objeto concreto de la base de datos. Permiten acceder y realizar cambios en los datos de los usuarios
  Ejemplo: consulta de tablas de otro usuario, insertar o borrar datos en una tabla

  El creador de un objeto, tiene automáticamente concedidos todos los privilegios que son aplicables a dicho objeto. Por ejemplo, el creador de la tabla tiene automáticamente los privilegios SELECT, INSERT, UPDATE, DELETE y REFERENCES.
  Además, tiene concedidos estos privilegios con WITH GRANT OPTION, es decir con autoridad para concederlos a otros usuarios.

  Sintaxis de la sentencia GRANT

```
GRANT {lista de privilegios | ALL [PRIVILEGES]} [(columna1, columna2,...)]
ON objeto
TO {lista de usuarios | roles | PUBLIC}
[WITH GRANT OPTION ]
```

  Donde:

  - ✓ La **lista de privilegios** consiste en uno o más nombres de privilegios separados por comas. (ALTER, DELETE, EXECUTE, INDEX, INSERT, REFERENCES, SELECT, UPDATE)
  - ✓ **ALL PRIVILEGES** concede todos los privilegios al objeto especificado.
  - ✓ **Columna1, columna2**,.. especifica las columnas de la tabla o vista sobre las que se conceden los privilegios (solo para INSERT, REFERENCES y UPDATE)
  - ✓ **Objeto** se refiere al nombre del objeto sobre el que se conceden privilegios. (tablas, vistas, secuencias, procedimientos, funciones, paquetes, sinónimos)
  - ✓ La **lista de usuarios** serán los identificativos de uno o más usuarios separados por comas. Se puede usar PUBLIC, con el significado de todos los usuarios definidos en el sistema.
  - ✓ Si se especifica **WITH GRANT OPTION**, los usuarios a los que se conceden los privilegios podrán a su vez concederlos a otros usuarios.

  Un usuario que concede ciertos privilegios a otro, puede posteriormente revocar dicha concesión. La eliminación de privilegios sólo puede ser realizada por aquél que los concedió y para llevar a cabo la revocación de la concesión de uno o más privilegios se utiliza la sentencia **REVOKE**.

Sintaxis de la sentencia REVOKE

```
REVOKE {lista de privilegios | ALL [PRIVILEGES] } ON objeto
FROM {lista de usuarios | roles | PUBLIC} [ CASCADE CONSTRAINTS ]
```

Dónde:

- ✓ **Lista de privilegios, objeto, lista de usuarios** tienen el mismo significado que para GRANT
- ✓ **CASCADE CONSTRAINTS**, cuyo significado se explica a continuación.
  Supongamos que el usuario **A** concede el privilegio **p** sobre cierto objeto al usuario **B**, quien a su vez lo concede al usuario **C**. ¿Que ocurriría si ahora **A** revoca **p** de **B**?. Si el privilegio no fue concedido con la opción CASCADE, la correspondiente sentencia REVOKE no ejecutaría con éxito. Si el privilegio sí fue concedido con la opción CASCADE, la REVOKE se ejecutaría con éxito y además se revocaría el privilegio al usuario **C**.
  Si se elimina un objeto, se revocan automáticamente todos los privilegios concedidos sobre el objeto.

- **Privilegios del sistema:** Dan derecho a ejecutar un tipo de comando SQL o a realizar alguna acción sobre objetos. Son permisos para modificar el funcionamiento de la base de datos. Son cambios que afectan a todos los usuarios.
  Ejemplo. Crear tablespaces, crear vistas, crear tablas, consultar tablas, borrar tablas.
  Hay más de 60 privilegios del sistema.
  Los privilegios del sistema son concedidos o revocados a usuarios y roles utilizando las sentencias GRANT y REVOKE.

Sintaxis de la sentencia GRANT

```
GRANT {privilegios del sistema |roles }
TO {lista de usuarios | roles | PUBLIC} [ WITH ADMIN OPTION ]
```

Solo los usuarios que tienen concedido el privilegio GRANT ANY PRIVILEGE pueden conceder o revocar privilegios del sistema a otros usuarios. Cuando un usuario tiene concedido un privilegio del sistema con la opción WITH ADMIN OPTION, puede conceder o revocar este privilegio a otros usuarios.

Sintaxis de la sentencia REVOKE

```
REVOKE {privilegios del sistema |roles }
        FROM {lista de usuarios | roles | PUBLIC}
```

Para conocer los privilegios concedidos o recibidos por los usuarios sobre los **objetos** o a nivel de **sistema**, se pueden consultar las siguientes vistas del diccionario de datos:

- ✓ **SESSION_PRIVS**: privilegios del usuario activo.
- ✓ **USER_SYS_PRIVS**: privilegios de sistema asignados al usuario.
- ✓ **DBA_SYS_PRIVS**: privilegios del sistema asignado a los usuarios y a los roles.
- ✓ **USER_TAB_PRIVS**: concesiones sobre objetos que son propiedad del usuario, concedidos o recibidos por éste.
- ✓ **USER_TAB_PRIVS_MADE**: concesiones sobre objetos que son propiedad del usuario (asignadas).
- ✓ **USER_TAB_PRIVS_RECD**: concesiones sobre objetos que recibe el usuario.
- ✓ **SER_TAB_GRANTS**: concesiones en objetos para los que el usuario es el propietario, el que concedió el privilegio o al que se le concedió el privilegio.
- ✓ **USER_TAB_GRANTS_MADE**: todas las concesiones hechas en objeto que son propiedad del usuario.
- ✓ **USER_TAB_GRANTS_RECD**: concesiones en objetos en las que el usuario es aquél al que se ha concedido el privilegio (concesiones recibidas)
- ✓ **ALL_TAB_GRANTS, ALL_TAB_GRANTS_MADE, ALL_TAB_GRANTS_RECD**: son iguales que las anteriores vistas, con la diferencia de que aparecen las concesiones de todos los usuarios.
- ✓ **USER_COL_GRANTS**: concesiones en columnas para las que el usuario es el propietarios, el que concedió el privilegio o al que se le concedió el privilegio.
- ✓ **USER_COL_GRANTS_MADE, USER_COL_GRANTS_RECD**: iguales que las anteriores vistas pero para columnas.
- ✓ **ALL_COL_GRANTS, ALL_COL_GRANTS_MADE, ALL_COL_GRANTS_RECD**: son iguales que las anteriores vistas, pero aparecen concesiones de todos los usuarios de la base de datos.
- ✓ **USER_COL_PRIVS**: concesiones sobre columnas en las que el usuario es el propietario, asigna el privilegio o lo recibe.
- ✓ **USER_COL_PRIVS_MADE:** todas las concesiones sobre columnas de objetos que son propiedad del usuario.
- ✓ **USER_COL_PRIVS_RECD:** concesiones sobre columnas recibidas por el usuario.

9.- Roles.

Los roles son privilegios aglutinados sobre un mismo nombre, bajo la idea de que ese conjunto denote un uso habitual sobre la base de datos. Gracias a los roles se facilita la asignación de privilegios a los usuarios. Un usuario puede tener asignado varios roles.

- **Sintaxis de creación.**

```
CREATE ROLE nombre_rol  [IDENTIFIED BY contraseña];
```

- **Sintaxis de Supresión de privilegios en los roles**

```
REVOKE privilegios [ON nom_tabla] FROM nombre_rol;
```

- **Sintaxis de Supresión de un rol**

```
DROP ROLE nombre_rol;
```

- **Sintaxis de Establecer rol por defecto**

```
ALTER USER nombre_usuario DEFAULT  {ROLE nombre_rol] | [none]};
```

- **Vistas sobre roles en el diccionario**

  - **SESSION_ROLES**: roles activos para el usuario
  - **USER_ROLE_PRIVS:** roles asignados al usuarios
  - **DBA_SYS_PRIVS**: privilegios del sistema asignados a los usuarios o roles
  - **DBA_ROLE_PRIVS**: privilegios asignados a todos los usuarios y roles
  - **DBA_ROLES**: todos los roles

9.1.- Funcionalidad de los roles:

- A un rol se le pueden conceder privilegios del sistema o de objeto.
- A un rol se le pueden conceder otros roles. No obstante, un rol no puede concederse a sí mismo ni de forma circular (por ejemplo, el rol A no se puede conceder al rol B si el rol B ha sido previamente concedido al rol A).
- Cualquier rol puede ser concedido a cualquier usuario.

- Cada rol concedido a un usuario puede, en un momento dado, estar habilitado o deshabilitado. Oracle permite a los usuarios habilitar y deshabilitar roles para lograr distintas disponibilidades de privilegios.
- Un rol concedido indirectamente (un rol concedido a otro rol) puede ser explícitamente habilitado o deshabilitado por el usuario. Sin embargo, al habilitar un rol que contiene otros roles, implícitamente se habilitan todos los roles concedidos indirectamente por el rol directamente concedido.
- Cualquier usuario con el privilegio del sistema GRANT ANY ROLE puede conceder o revocar cualquier rol a otros usuarios o roles. Para conceder o revocar roles se usan las sentencias GRANT y REVOKE.
- La sentencia CREATE ROL permite crear un rol con un cierto nombre. El rol se crea vacío, los privilegios se le asignan posteriormente con la sentencia GRANT.
- La sentencia SET ROL permite habilitar y deshabilitar roles.
- A través de la sentencia ALTER USER se pueden establecer los roles por defecto, o sea, aquellos roles concedidos al usuario que se habilitan automáticamente al iniciar la sesión.
- Al crear una base de datos Oracle, se definen automáticamente los roles CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE y IMP_FULL_DATABASE. Estos roles existen por compatibilidad con versiones anteriores de Oracle y se pueden manejar igual que cualquier otro rol.