LENGUAJE SQL

Nivel físico

SQL: Introducción

- Una vez vistos los diagramas E/R y relacional para representar los niveles de abstracción conceptual y lógico, es necesario conocer el nivel físico.
- Este nivel nos permite construir la base de datos y no da la posibilidad de utilizarla.
- Para ello utilizaremos Oracle:
  - Es un producto comercial ampliamente extendido y utilizado
  - Permite almacenar y manejar una gran cantidad de forma rápida y segura

SQL: Álgebra relacional

- SQL se basa en Teoría Matemática del Álgebra Relacional. Consta de varios elementos:
- **Lenguaje de Definición de Datos (DDL):** proporciona órdenes para definir, modificar o eliminar los distintos objetos de la base de datos (tablas, vistas, índices).
- **Lenguaje de Manipulación de datos (DML):** proporciona órdenes para insertar, suprimir y modificar registros o filas de las tablas.
- **Lenguaje de Control de Datos (DCL):** permite establecer derechos de acceso de los usuarios sobre los distintos objetos de la base de datos.

SQL: Proceso de ejecución

- El proceso de una instrucción SQL es el siguiente:
  - Se analiza la instrucción. Para comprobar la sintaxis de la misma
  - Si es correcta se valora si los metadatos de la misma son correctos. Se comprueba esto en el diccionario de datos.
  - Si es correcta, se optimiza, a fin de consumir los mínimos recursos posibles.
  - Se ejecuta la sentencia y se muestra el resultado al emisor de la misma.

SQL: Notación

Por convenio, las sentencias SQL se interpretan de la siguiente forma:

- Palabras clave de la sintaxis SQL en MAYÚSCULAS
- Los corchetes [] indican opcionalidad
- Las llaves {} delimitan alternativas separadas por "|" de las que se debe utilizar una.
- Los puntos suspensivos "…" indican repetición de varias veces la opción anterior

Nota: Las sentencias se ejecutan igualmente en mayúsculas que en minúsculas.

SQL: Normas y comentarios

- En SQL no se distingue entre mayúsculas y minúsculas. Da lo mismo como se escriba. El final de una instrucción o sentencia lo marca el signo de punto y coma.
- Los comentarios en el código SQL pueden ser de 2 tipos:
- De bloque: comienzan por /* y terminan por */
- De línea: comienzan por – y terminan en final de línea

```
-- Esto es un comentario de una línea
```

```
/*
  Esto es un comentario
  de varias líneas.

  Fin.
*/
```

Tipos de datos

SQL: Tipos de datos

VARCHAR2(n)

Cadena de caracteres de longitud variable. Se debe especificar el número de caracteres que tendrá (n).

NUMBER(n)

Dato de tipo numérico de un máximo de 40 dígitos, además del signo y el punto decimal. Se puede utilizar notación científica (1.273E2 es igual a 127.3). Se usa para Números enteros. Se puede especificar el número de dígitos (n).

SQL: Tipos de datos

CHAR(n)

Cadena de caracteres de longitud fija. Se puede especificar el número de caracteres que tendrá (n).

NUMBER(p,d)

Números reales. Donde “p” especifica el número total de dígitos (máximo 38 dígitos) y “d” el número total de decimales. Por ejemplo NUMBER(4,2) tiene como máximo valor 99.99.

DATE.

El tipo DATE permite almacenar fechas.

Expresiones y operadores

SQL: Expr. y operadores

- Las condiciones son expresiones lógicas (devuelven verdadero o falso) que se sitúan normalmente junto a una cláusula SQL que utilizan muchos comandos.
- Dentro del DDL se utilizarán con la cláusula **CHECK** que sirve para establecer las condiciones que deben cumplir sobre los valores que se almacenarán en una tabla.
- Las condiciones se construyen utilizando los operadores de comparación y los operadores lógicos.
- A continuación se describen los operadores más importantes:

SQL: Oper. comparación

- Con ellos se pueden formar expresiones que pueden ser evaluadas y de las que se desprende un resultado que puede ser verdadero o falso.
- Los símbolos más usados son: igual que (=), mayor o menor (<>), distinto a (!=), menor o igual (<=), mayor o igual (>=), menor que (<) y mayor que (>).
- Ejemplo:
  - horas <= 10.5 (las cifras no llevan ‘ ‘)
  - nombre = ‘pepe’ (un ‘=’ compara, no asigna)
  - fecha < ‘1-ene-93’

SQL: [not] IN lista_valores

- Comprueba la pertenencia a la lista de valores. Generalmente, los valores de la lista se habrán obtenido como resultado de un comando SELECT (comando de consulta).
- Se utiliza cuando queremos que el valor introducido se encuentre en una lista, o por el contrario, sea cualquier valor excepto los de esa lista.
- Ejemplo:

```
SELECT * FROM clientes WHERE nombre NOT IN  ('PEPE', 'LOLA');
```

SQL: oper {any/some} lista

- Comprueba que se cumple al operación “oper” de la lista de valores.
- Operaciones posibles: =, <>, !=, <, >,<=, >=
- Permite hacer comprobaciones numéricas con los valores introducidos en una sentencia:

nombre = ANY (‘PEPE’, ‘LOLA’)

Ejemplo:

SELECT DISTINCT Name FROM Store WHERE s.Name = ANY

(SELECT v.Name FROM Vendor AS v) ;

SQL: oper ALL lista valores

- Comprueba que se cumple al operación “oper” con todos los elementos de la lista.
- Operaciones posibles: =, <>, !=, <, >,<=, >=
- Permite hacer comprobaciones muy restrictivas:

nombre <> ANY (‘PEPE’, ‘LOLA’)

SQL: [NOT] BETWEEN X AND Y

- Comprueba que el valor introducido se encuentre en el rango de valores entre x e y (o lo contrario, si se utiliza el operador NOT).
- Horas es un valor entre 10 y 20 (incluidos):

```
SELECT * nombre FROM trabajadores
WHERE horas_trabajadas BETWEEN 10 AND 20;
```

- Horas es menor que 10 o mayor que 20:

```
SELECT * nombre FROM trabajadores
WHERE horas_trabajadas horas NOT BETWEEN 10 AND 20;
```

SQL: [NOT] EXIST lista_valores

- Evalúa como true/false una sentencia y la utiliza en otra principal.

  EXISTS (‘ALGO’) o NOT EXISTS (‘ALGO’)

- Ejemplo:

```
SELECT cliente, numero FROM facturas as f
WHERE EXISTS (SELECT * FROM detalles AS d
    WHERE f.numero = d.numerofactura
        AND d.articulo='lapiz');
```

SQL: [NOT] LIKE texto

Comprueba si los valores cumplen un patrón de texto.

Los símbolos comodín que pueden usarse son dos:

- _ : sustituye a un único carácter.

```
SELECT  código FROM artículos
WHERE codigo NOT LIKE 'cod1_'
```

- % : sustituye a varios caracteres.

```
SELECT nombre, direccion FROM ALUMNOS
WHERE nombre LIKE 'Pedro%'
```

SQL: IS [NOT] NULL

- Comprueba si el valor de un campo está vacío (es nulo) o contiene algún valor.
- Seleccionamos los clientes de los que no consta su número de teléfono:

```
SELECT name FROM clientes WHERE telefono IS NULL;
```

SQL: OPERADORES LÓGICOS

- Podemos concatenar varias comprobaciones mediante los operadores lógicos OR, AND y NOT, con la intención de realizar sentencias más precisas (y complejas).
- AND: Deben cumplirse todas las condiciones

  horas > 10 AND telefono IS NULL
- OR: Debe cumplirse una de las condiciones

  nombre = ‘PEPE’ OR saldo > 0
- NOT: Invierte la condición

  NOT (nombre IN (‘PEPE’, ‘LUIS’))

LENGUAJE DLL : USUARIOS

SQL: Bases de datos

- Dentro del SGBD es necesario crear una base de datos para poder comenzar a crear objetos y usuarios en ella.
- Para ello es necesario contar con un usuario con permisos de administrador (permiso DBA).
- Por lo general, al instalar el sistema SGBD se crean usuarios de administrador (sys, sysdba…) y una base de datos por defecto.
- La base de datos se crea con el comando:

```
CREATE DATABASE nombre;
```

SQL: Bases de datos

Normalmente, se indican más parámetros:

- CREATE DATABASE prueba
- LOGFILE prueba.log
- MAXLOGFILES 25 MAXINSTANCES 10
- ARCHIVELOG
- CHARACTER SET AL32UTF8
- NATIONAL CHARACTER SET UTF8
- DATAFILE prueba.dbf AUTOEXTEND ON
- MAXSIZE 500MB;

SQL: Bases de datos

- Para eliminar la base de datos y todos sus objetos, se utiliza:

```
DROP DATABASE prueba;
```

- Para modificar la base de datos se utiliza ALTER DATABASE y todos los argumentos y parámetros que queremos modificar con sus nuevos valores (establecemos el nuevo máximo de ficheros de los e instancias a 30):

```
ALTER DATABASE prueba
MAXLOGFILES 30 MAXINSTANCES 30;
```

SQL: Bases de datos

Relacionando la administración de usuarios con la seguridad se puede diferenciar entre;

- **Seguridad del sistema:** Mecanismos que controlan el acceso y uso de la base de datos a nivel de sistema. Ej: comprobar si el usuario está autorizado a conectarse a la bbdd, privilegios para crear objetos nuevos, etc.
- **Seguridad de los datos:** Mecanismos que controlan el acceso y uso de la base de datos a nivel de objeto. Ej: Cada vez que se accede a una tabla, vista… se comprueba si el usuario está autorizado a realizar INSERT, SELECT...

SQL: Bases de datos

En el proceso de conexión de un usuario a la base de datos se diferencian dos etapas:

Autenticación (autentificación):

- Verificar la identidad del usuario que se quiere conectar y permitirle o denegarle la conexión

Autorización:

- Cuando un usuario se ha autentificado, el servidor comprueba cada comando que ejecuta para ver si tiene suficientes permisos para ello.

SQL: Usuarios / Esquemas

Usuario (o cuenta de usuario)

- Nombre definido en la base de datos que se puede conectar a ella y acceder a determinados objetos según los privilegios que se le condedan.

Esquema

- Colección de objetos (tablas, vistas, sinónimos, índices, etc.) de los que es dueño un usuario.
- Un esquema tiene el mismo nombre que el de su usuario

SQL: Usuarios / Esquemas

- Cada cuenta de usuario tiene asociado un esquema con su mismo nombre y es el dueño de los objetos creados en él.
- Cuando se crea un usuario se crea automáticamente un esquema con el mismo nombre, el cual contiene sus objetos.
- El usuario tiene control total sobre los objetos de su esquema y puede otorgar permisos a otros usuarios sobre sus objetos (por defecto, nadie más tiene acceso a su esquema).

SQL: Tablespaces

- Se define tablespace como una ubicación de almacenamiento donde pueden ser guardados los datos correspondientes a los objetos de una base de datos.
- Un tablespace puede contener objetos de diferentes esquemas.
- Se suele definir:
  - Un **tablespace** por defecto para almacenar los objetos del usuario
  - Un **tablespace temporal** por defecto para las operaciones del usuario

SQL: Usuarios

Cada cuenta de usuario tiene:

- Un nombre único
- Un método de autentificación
- Un tablespace por defecto
- Un tablespace temporal por defecto
- Un perfil
- Un estado: bloqueado (locked) /no bloqueado (unlocked).

SQL: Creación de usuario

La sintaxis es la siguiente:

```
CREATE USER <usuario>
[IDENTIFIED BY <contraseña>/EXTERNALLY/<datos global>]
[DEFAULT TABLESPACE <nombre_tablespace>]
[TEMPORARY TABLESPACE <nombre_tablespace>/<grupo_tablespace>]
[QUOTA <xx>/UNLIMITED ON < nombre_tablespace >]
[PROFILE <perfil>]
[PASSWORD EXPIRE]
[ACCOUNT LOCK/UNLOCK];
```

```
CREATE USER usuario1
IDENTIFIED BY usuario1
QUOTA 20M ON USERS
PASSWORD EXPIRE;
```

```
CREATE USER usuario2
IDENTIFIED BY usuario2
DEFAULT TABLESPACE data01
TEMPORARY TABLESPACE temp
QUOTA 15m ON data01
PASSWORD EXPIRE;
```

SQL: Creación de usuario

Nombre de usuario

- No puede exceder de 30 bytes.
- Debe comenzar con una letra.

Autenticación (IDENTIFIED BY)

- Para acceder a la base de datos el usuario tienen que autentificarse.
- Para cada usuario se puede definir su método de autentificación.

SQL: Creación de usuario

Default tablespace

- Indica el espacio de almacenamiento donde se crearán los objetos del esquema del usuario cuando al hacerlo no se especifique ninguno en particular.
- Si se omite la cláusula, los objetos se crean en el "tablespace por defecto" de la base de datos.
- Si el "default user tablespace" no se ha especificado, el espacio por defecto es SYSTEM (¡NO RECOMENDADO!).

SQL: Creación de usuario

Temporary tablespace

- Indica el tablespace o grupo de tablespace de almacenamiento para los segmentos temporales requeridos por el usuario.
- No debe indicarse cuota.
- Si se omite la cláusula, los objetos se crean en el tablespace temporal por defecto de la bd (default temporaly tablespace).
- Si default temporaly tablespace no se ha especificado el espacio por defecto es SYSTEM (¡NO RECOMENDADO!)

SQL: Creación de usuario

Quota

- Indica la cantidad máxima de espacio que un usuario puede utilizar en un determinado tablespace de almacenamiento.
- Puede indicarse cuota en múltiples tablespaces al mismo tiempo.
- El creador del usuario puede indicar cuota sobre tablespace de almacenamiento aunque él no las posea.
- Por defecto no se tiene cuota en ningún tablespace

SQL: Creación de usuario

Quota

Opciones

- UNLIMITED es ilimitado espacio a usar
- **NOTA:** UNLIMITED TABLESPACE, prevalece sobre las cuotas.
- Pueeden usarse distintas abreviaturas para indicar el tamaño: kilobytes (K), megabytes (M), gigabytes (G), terabytes (T), petabytes (P), o exabytes (E).
- Puede revocarse el acceso a un espacio de almacenamiento asignando cuota cero en el mismo. Los objetos ya creados permanecen pero no pueden crecer ni crearse ninguno mas.

SQL: Creación de usuario

Profile

- Indica el perfil del usuario (conjunto de limitaciones y restricciones
- Si se omite se asigna el perfil DEFAULT que ofrece recursos ilimitados (pero no privilegios).

Password Expire

- Fuerza al usuario a cambiar la clave antes de conectarse a la base de datos.

Account

- “Account lock” bloquea la cuenta de usuario
- ”Account unlock” desbloquea la cuenta

SQL: Modificación de usuario

```
ALTER USER <usuario>
[IDENTIFIED BY <contraseña>/EXTERNALLY]
[DEFAULT TABLESPACE < nombre_tablespace >]
[TEMPORARY TABLESPACE <nombre_tablespace>/<grupo_tablespace>]
[QUOTA <xx>/UNLIMITED ON < nombre_tablespace >
[DEFAULT ROLE <role>/ALL/ALL EXCEPT <role>/NONE]
[PROFILE <perfil>]
[PASSWORD EXPIRE]
[ACCOUNT LOCK/UNLOCK];
```

```
ALTER USER usuario1
QUOTA 40M ON USERS;
```

```
ALTER USER usuario2
IDENTIFIED BY otra
DEFAULT TABLESPACE USER
TEMPORARY TABLESPACE temp
QUOTA UNLIMITED ON USER
PASSWORD EXPIRE;
```

SQL: Modificación de usuario

Default Role

- Permite activar y desactivar roles de usuarios
- Oracle activa los roles indicados sin necesidad de especificar sus contraseñas.
- Indica los roles otorgados por defecto al usuario

Los usuarios pueden cambiar sus propias claves.

Para cambiar cualquier otro parámetro es necesario el privilegio “ALTER USER” (es un permiso).

SQL: Borrado de usuario

Drop

- Al borrar un usuario el esquema asociado, con todos sus objetos desaparecen.
- Oracle no borra esquemas de usuario no vacíos a menos que se indique CASCADE (realiza un borrado de objetos previo) o se hayan eliminado con anterioridad los objetos.
- No se eliminan roles creados por el usuario.

```
DROP USER <usuario> <CASCADE>;
```

```
DROP USER usuario1;
```

```
DROP USER usuario2 CASCADE;
```

SQL: Borrado de usuario

- Una posible solución para que permanezca el usuario y los objetos pero impedir la conexión es revocar el privilegio “CREATE SESSION”.
- No es posible eliminar un usuario que permanezca conectado a la base de datos. Debe esperarse a que concluya o matar su sesión (ALTER SYSTEM KILL SESSION).
- Es necesario tener el privilegio de sistemas “DROP USER”.

LENGUAJE DLL : TABLAS

SQL: Creación de tablas

- Una tabla es un objeto de la base de datos que se utiliza para almacenar la información.
- Se estructura en filas y columnas de una forma similar a una hoja de cálculo
  - Las filas representan cada elemento de la tabla (elementos del tipo entidad en el diagrama ER)
  - Las columnas representan las características o propiedades que tiene cada elemento de la tabla.

SQL: Creación de tablas

El nombre de las tablas debe cumplir las siguientes reglas:

- Deben comenzar con una letra
- No deben tener más de 30 caracteres
- Sólo se permiten utilizar letras del alfabeto (inglés), números o el signo de subrayado (y algún signo especial, pero no es recomendado)
- No puede haber dos tablas con el mismo nombre para el mismo usuario (pero sí podría haberla en otro esquema distinto).
- No puede coincidir con el nombre de una palabra reservada de SQL (sintaxis reservada).

SQL: Creación de tablas

- Se crean tablas con la siguiente instrucción:

```
CREATE TABLE nombre_tabla (
  columna1  tipo_dato  [ restricciones de columna1 ],
  columna2  tipo_dato  [ restricciones de columna2 ],
  columna3  tipo_dato  [ restricciones de columna3 ],
  ...
  [ restricciones de tabla ]
);
```

- Para separar las columnas y restricciones se utiliza la coma.
- Para terminar la sentencia se utiliza punto y coma.

SQL: Creación de tablas

- Las tablas tienen varios tipos de restricciones, las cuales pueden añadirse a la tabla en el momento de su creación o bien después, modificando la tabla creada.
- Existen restricciones de columna:

```
CONSTRAINT nombre_restricción {
  [NOT] NULL | UNIQUE | PRIMARY KEY | DEFAULT valor | CHECK (condición)
}
```

- Existen restricciones de tabla:

```
CONSTRAINT nombre_restricción {
  PRIMARY KEY (columna1 [,columna2] ... )
| UNIQUE (columna1 [,columna2] ... )
| FOREIGN KEY (columna1 [,columna2] ... )
    REFERENCES nombre_tabla (columna1 [,columna2] ... )
    [ON DELETE {CASCADE | SET NULL}]
| CHECK (condición)
}
```

SQL: Creación de tablas

Cuando una restricción afecte a varias columnas se debe hacer obligatoriamente una restricción de tabla. Las restricciones son:

- **PRIMARY KEY:** Establece ese atributo o conjunto de atributos como clave primaria de la tabla. Implica UNIQUE y NOT NULL.
- **UNIQUE:** Impide que se produzcan valores repetidos para ese atributo. Claves candidatas.
- **NOT NULL:** Evita que se introduzcan valores NULL en ese atributo.

NOTA: Ni UNIQUE ni NOT NULL se usan con PRIMARY KEY porque ya lo incluyen de forma implícita.

SQL: Creación de tablas

- **DEFAULT:** Especifica el valor por defecto si no se indica ninguno.
- **CHECK (condición)**: Establece condiciones que deben cumplir los valores de una columna al hacer INSERT en la tabla. Puede ser de columna o tabla.
- Solo permite hacer referencia a columnas de la propia tabla ni subconsultas.
- Además las funciones SYSDATE y USER no se pueden utilizar dentro de la condición.
- Están permitidas comparaciones simples de atributos y operadores lógicos.

SQL: Creación de tablas

- **FOREIGN KEY**: Define un campo como clave externa(foránea) de la clave primaria de otra tabla. Es necesario que la PK esté creada antes de esta asignación.
  - **ON DELETE CASCADE:** Mantiene la integridad referencial borrando los valores de la clave foránea correspondientes a una clave primaria
  - **ON DELETE SET NULL:** Especifica que se ponga a NULL los alores de la llave externa correspondientes a un valor borrado de la tabla referenciada.

SQL: Creación de tablas

Las tablas pueden tener restricciones, las cuales se establecen mediante la instrucción CONSTRAINT.

- Pueden ser etiquetadas (recomendado) aunque no es obligatorio.
- Si no se etiquetan, el sistema les pone un nombre “SYS_Cn” donde n es un número.
- Las restricciones pueden ser establecidas en la misma sentencia de creación o pueden añadirse/borrarse/modificarse posteriormente.

SQL: Creación de tablas

- Se puede consultar las tablas creadas por el usuario en la vista USER_TABLES.

```
SELECT * FROM USER_TABLES;
```

- Se puede consultar las características de una tabla con la sentencia DESCRIBE tabla.

```
DESCRIBE COCHES;
```

- Se puede consultar las restricciones de un usuario en la vista USER_CONSTRAINTS.

SQL: Creación de tablas

- Sin etiquetar:

```
CREATE TABLE usuarios (
  id            NUMBER                    PRIMARY KEY,
  dni           CHAR(9)                   UNIQUE,
  nombre        VARCHAR2(50)      NOT NULL,
  edad          NUMBER                    CHECK (edad>=0 and edad<120)
);
```

- Etiquetadas:

```
CREATE TABLE usuarios (
  id          NUMBER        CONSTRAINT usu_id_pk  PRIMARY KEY,
  dni         CHAR(9)       CONSTRAINT usu_dni_uq UNIQUE,
  nombre      VARCHAR2(50)  CONSTRAINT usu_nom_nn NOT NULL,
  edad        NUMBER        CONSTRAINT usu_edad_ck
                               CHECK (edad>=0 and edad<120)
);
```

SQL: Creación de tablas

- Ejemplo:

```
CREATE TABLE COCHES (
  matricula              VARCHAR2(8),
  marca                  VARCHAR2(15) NOT NULL,
  color                  VARCHAR2(15),
  codTaller              VARCHAR2(10),
  codProp                VARCHAR2(10),
  CONSTRAINT coches_mat_pk PRIMARY KEY (matricula),
  CONSTRAINT coches_codtaller_fk1 FOREIGN KEY (codTaller)
      REFERENCES TALLER(codTaller),
  CONSTRAINT coches_codprop_fk2 FOREIGN KEY (codProp)
      REFERENCES PROPIETARIO(codProp),
  CONSTRAINT coches_color_ck1
      CHECK (color IN ('ROJO','AZUL','BLANCO','GRIS','VERDE','NEGRO'))
);
```

SQL: Borrado de tablas

- Para eliminar una tabla del esquema se utiliza la sentencia DROP TABLE

```
DROP TABLE nombre_tabla
[ CASCADE CONSTRAINTS ];
```

- La opción CASCADE CONSTRAINTS permite eliminar una tabla que contenga atributos referenciados por otras tablas (claves foráneas), eliminando esas referencias.
- Si no se utiliza CASCADE CONSTRAINTS no podrá eliminarse una tabla que sea apuntada por otras tablas (con sus FOREIGN KEY).

SQL: Modificación de tablas

A nivel de tabla se pueden hacer dos tipos de modificaciones en una tabla:

- Se pueden renombrar las tablas con la orden RENAME:

```
RENAME nombre TO nombre_nuevo;
```

- Se puede vaciar el contenido de una tabla con la orden TRUCATE TABLE:

```
TRUNCATE TABLE AUTOMOVILES;
```

SQL: Modificación de tablas

Con la cláusula ALTER TABLE se puede hacer cambios en la estructura de una tabla.

Añadir columna

```
ALTER TABLE VEHICULOS ADD ( fechaMatric DATE );
```

Modificar columna

```
ALTER TABLE AUTOMOVILES
MODIFY (color VARCHAR2(20) NOT NULL, codTaller VARCHAR2(15));
```

Eliminar columna

```
ALTER TABLE VEHICULOS DROP (tipoFaros);
```

SQL: Modificación de tablas

Los cambios que se permiten son:

- **Incrementar la precisión o anchura** de los tipos de datos.
- **Reducir la anchura máxima de un campo**, UNICAMENTE, si esa columna posee nulos en todos los registros, o no hay registros.
- Se puede pasar de **CHAR a VARCHAR2** y viceversa (si no se modifica la anchura).
- Se puede pasar de **DATE a TIMESTAMP** y viceversa.

SQL: Modificación de tablas

También se permite modificar las restricciones de una tabla:

Añadir/modificar

```
ALTER TABLE nombre_tabla { ADD | MODIFY } (
  CONSTRAINT nombre_restricción1   tipo_restricción  (columnas)
  [, CONSTRAINT nombre_restricción2   tipo_restricción  (columnas) ] ...
);
```

Eliminar restricción

```
ALTER TABLE nombre_tabla
DROP {
  PRIMARY KEY
| UNIQUE  (columnas)
| CONSTRAINT nombre_restricción [ CASCADE ]
}
```

```
ALTER TABLE CURSOS DROP (fechaIni);
```

SQL: Modificación de tablas

Las restricciones pueden renombrarse:

```
ALTER TABLE nombre_tabla
RENAME CONSTRAINT nombre_restricción TO nombre_restricción_nuevo;
```

Las restricciones cuentan con un “estado” por lo que pueden estar activas e inactivas, no siendo necesario eliminarlas para deshabilitarlas temporalmente. Por defecto se activan al crearse.

Activar restricciones

```
ALTER TABLE nombre_tabla ENABLE CONSTRAINT restricción [ CASCADE ];
```

Desactivar restricciones

```
ALTER TABLE nombre_tabla DISABLE CONSTRAINT restricción [ CASCADE ];
```

VISTAS

SQL: Definición de vistas

Una vista es una consulta almacenada con la intención de ser reutilizada posteriormente. Se utiliza para:

- Proporcionar tablas con datos complejos
  - Nos permite ver varias tablas como una sola
- Utilizar visiones especiales de los datos
  - Nos permite mostrar tablas personalizadas
- Servir como mecanismo de seguridad
  - Nos permite ocultar algunas columnas

SQL: Tipos de vistas

Hay dos tipos de vistas:

- **Simples:** Formadas por una tabla, sin funciones de agrupación. Siempre permiten realizar operaciones DML sobre ellas.
- **Complejas:** Formadas por datos de varias tablas, pudiendo usar funciones de agrupación. No siempre permiten operaciones DML.

Nota: *Una función de agrupación o agregada, es la que realiza un cálculo con varios datos y devuelve un resultado. Ej: AVG(edad), MIN(saldo), COUNT(*)…*

SQL: Creación/Modificación de vistas

Sintaxis:

```
CREATE [ OR REPLACE ] VIEW nombre_vista [ (alias1 [, alias2] ...) ]
AS SELECT ...
```

Parámetros:

- **OR REPLACE:** Reemplaza la vista si ya existe.
- **ALIAS:** Es una forma de etiquetar temporalmente la tabla dentro de una consulta con el objetivo de acortar su nombre.

CREATE OR REPLACE VIEW miVista AS SELECT cli.clientes FROM clientes AS cli WHERE saldo < 100;

SQL: Modificación/Borrado de vistas

- Se puede utilizar la orden ALTER VIEW para forzar el cambio de una vista, aunque suele usarse CREATE OR REPLACE VIEW.
- La sentencia ALTER VIEW permite localizar errores de recompilación antes de la ejecución.

Sintaxis:

```
DROP VIEW nombre_vista;
```

SQL: Índices

Los índices son objetos que forman parte del esquema del usuario y permiten una rápida ejecución de las consultas.

- Se almacena parte de la tabla a la que hace referencia, lo que permite crearlos y borrarlos.
- Consiste en una copia parcial y ordenada de la tabla, la cual se actualiza ante cualquier inserción/borrado de cualquier registro.
- Cuantos más índices más costoso en recursos es realizar un INSERT, pero más rápida es la consulta.

SQL: Creación de índices

La mayoría de los índices se crean de manera implícita como consecuencia de las claves primarias, foráneas y los atributos UNIQUE.

- Estos índices son obligatorios en el sistema.
- Estos índices los crea el propio SGBD.

Aparte de estos índices, se pueden crear otros de forma explícita, con la intención de acelerar otro tipo de consultas que utilicen otro tipo de campo.

- Consultas del número de teléfono o el email, p.ej.

SQL: Creación/Borrado de índices

Sintaxis:

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

Ejemplo:

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

- En este ejemplo, se crea un índice con los campos de apellido y nombre.
- No es lo mismo que utilizar un índice para cada campo, ya que es efectivo cuando se busca por los 3 campos a la misma vez.

Sintaxis de borrado:

```
DROP INDEX nombre_indice;
```

SQL: Creación de índices

Se aconseja la creación de índices en campos que:

- Contengan una gran cantidad de valores
- Contengan gran cantidad de nulos
- Sean habituales en cláusulas WHERE, GROUP BY, ORDER BY.
- Sean parte de listados de consultas grandes tablas sobre las que se muestran pequeñas partes de su contenido (alrededor de un 4%)

SQL: Creación de índices

No se aconseja la creación de índices en campos que:

- Pertenezcan a tablas pequeñas
- No sean habituales en consultas
- Pertenecen a tablas cuyas consultas muestran gran cantidad de datos (más de un 4%)
- Pertenecen a tablas que se actualizan frecuentemente
- Utilizan expresiones

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

SQL: Secuencias

Una secuencia sirve para generar automáticamente números distintos:

- Se utiliza para campos utilizados como ID
- Es una rutina interna de la base de datos que genera números distintos cada vez que es llamada
- Se puede utilizar para diversas tablas
- Puede ser llamado desde una sentencia INSERT para generar un autonumériс.

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

SQL: Creación de secuencias

Sintaxis:

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

Ejemplo:

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

- **Secuencia:** Nombre de la secuencia
- **INCREMENT BY:** Indica cuanto avanza/retrocede el valor de la secuencia cada vez que es llamada.
- **START WITH:** Valor inicial.
- **MAXVALUE:** Máximo valor admitido. NOMAXVALUE llega hasta 1027
- **MINVALUE:** Mínimo valor admtido. NOMINVALUE llega hasta -1026
- **CYCLE:** Hace que la secuencia se reinicie cuando llegue a su límite.

SQL: Secuencias

En el diccionario tenemos la vista USER_SEQUENCES que muestra la lista de secuencias del usuario.

La columna LAST_NUMBER nos permite consultar el último número asignado.

Los métodos que nos permiten conocer los valores de la secuencia son:

- Nextval: Próximo valor
- Currval: Valor actual

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

SQL: Modificación/Borrado

Modificar

Para modificar una secuencia se utiliza ALTER SEQUENCE:

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

Borrar:

Para eliminar una secuencia se utiliza DROP SEQUENCE:

```
DROP SEQUENCE nombre_secuencia;
```

SQL: Sinónimos

En Oracle, un sinónimo es un nombre que se le asigna a un objeto cualquiera.

- Normalmente es un nombre menos descriptivo que el original
- Sirve para facilitar su uso
- Se utiliza en expresiones para evitar realizar órdenes demasiado extensas

La vista USER_SYNONYMS permite observar la lista de sinónimos creada por el usuario

SQL: Crear/Modificar/Borrar Sinónimos

La sintaxis es la siguiente:

Creación

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

La clausula PUBLIC hace que el sinónimo esté disponible para cualquier usuario.

Modificación

```
ALTER [PUBLIC] SYNONYM nombre [{COMPILE|EDITIONABLE|NONEDITIONABLE}];
```

Borrado

```
DROP SYNONYM nombre;
```