IES Ribera del Tajo
Enseñanza Bilingüe

UT4: DISEÑO FÍSICO RALACIONAL
DDL

Módulo: BASES DE DATOS
Curso: 2023-2024

INDICE DE CONTENIDOS

1. Introducción

La primera fase del trabajo con cualquier base de datos implica el uso de ***sentencias DDL*** que nos permiten ***crear las estructuras básicas de almacenamiento***, como las tablas.

En el mercado existen asistentes gráficos que facilitan esta labor sin apenas conocer el SQL, pero como futuros programadores se hace **imprescindible comprender y conocer el lenguaje.** Pueden darse muchas situaciones en las que tendremos que crear**, modificar o eliminar un objeto** sin depender de estas herramientas.

Las instrucciones básicas del **DDL** son tres:

CREATE tipo_objeto Nombre Definicion.
Crea un objeto de un determinado tipo (DATABASE, TABLE, INDEX, etc.) con un nombre (PRODUCTOS, COMPRAS, …) y una definición (CodigoProducto, nombre, precio, etc.).

DROP tipo_objeto Nombre.
Elimina un tipo de objeto especificado mediante un nombre. Por ejemplo, la sentencia
DROP TABLE Productos *borraría de la base de datos la tabla Productos junto con todos sus datos*.

ALTER tipo_objeto Nombre Modificación. Modifica la definición de un objeto. Por ejemplo, la sentencia
ALTER TABLE Productos DROP COLUMN precio *eliminaría la columna Fecha de la tabla Productos*.

2. Creación de bases de datos.

La creación de la base de datos consiste en definir un espacio de nombres separado para cada conjunto de tablas. Es lo que antes hemos llamado **esquemas** o **usuarios**.

Para crear una base debemos tener privilegios de Administrador.

Con el estándar de SQL la instrucción a usar sería ***Create Database***, pero cada SGBD tiene un procedimiento para crear las bases de datos. Crearíamos una base de datos con el nombre que se indique a continuación.

```
CREATE DATABASE NombredemiBasedeDatos;
```

Si la base de datos que vamos a crear se llamara GestionRibera, entonces nos quedaría:

```
CREATE DATABASE GestionRibera;
```

Una base de datos es un conjunto de objetos que nos servirán para gestionar los datos.

3. Creación de tablas.

La tabla es el objeto más importante con los que trabaja SQL, aunque no el único.

Antes de crear la tabla es conveniente planificar algunos detalles:

- ✓ Qué nombre le vamos a dar a la tabla.
- ✓ Qué nombre le vamos a dar a cada una de las columnas.
- ✓ Qué tipo y tamaño de datos vamos a almacenar en cada columna.
- ✓ Qué restricciones tenemos sobre los datos.
- ✓ Qué información adicional que necesitemos.

3.1. Nombres.

Existen una serie de reglas que debemos cumplir para los nombres de las tablas:

- ✓ Debe ser significativo.
- ✓ No podemos tener nombres de tablas duplicados en un mismo esquema (usuario).
- ✓ Deben comenzar por un carácter alfabético.
- ✓ Su longitud máxima es de 30 caracteres.
- ✓ Sólo se permiten caracteres alfabéticos, dígitos o el signo de guion bajo.
- ✓ No puede coincidir con las palabras reservadas de SQL.
- ✓ No distingue entre mayúsculas y minúsculas.

3.2. Tipos de datos

En la siguiente tabla se recogen los tipos de datos que podemos usar en ORACLE.

TIPO	CARACTERISTICAS	OBSERVACIONES
CHAR	*Cadena de caracteres alfanuméricos. de **longitud fija***	Entre 1 y 2000 bytes como máximo. Es de longitud fija, siempre ocupará lo mismo independientemente del valor que contenga
VARCHAR2	*Cadena de caracteres de **longitud variable***	Entre 1 y 4000 bytes como máximo. El tamaño del campo dependerá del valor que contenga.
VARCHAR	*Cadena de caracteres de **longitud variable***	**En desuso**, se utiliza VARCHAR2 en su lugar
NCHAR	*Cadena de caracteres de **longitud fija** de caracteres Unicode*	Entre 1 y 2000 bytes como máximo. El juego de caracteres del tipo de datos (datatype) NCHAR sólo puede ser AL16 UTF16 ó UTF8.
NVARCHAR2	*Cadena de caracteres **de longitud variable** que sólo almacena caracteres Unicode*	Entre 1 y 4000 bytes como máximo. El juego de caracteres del tipo de datos (datatype) NCHAR sólo puede ser AL16 UTF16 ó UTF8.
LONG	*Cadena de caracteres de **longitud variable***	Como máximo admite hasta 2 GB (2000 MB). Los datos LONG deben convertirse al moverse entre diversos sistemas. En desuso, se utilizan tipos LOB (CLOB, NCLOB).
LONG RAW	*Almacenan **cadenas binarias de ancho variable***	Hasta 2 GB. En desuso, se sustituye por los tipos LOB.

RAW	*Almacenan cadenas **binarias de ancho variable***	Hasta 32767 bytes. En desuso, se sustituye por los tipos LOB.
LOB (BLOG, CLOB, NCLOB, BFILE)	*Bloques grandes de datos no estructurados (texto, imágenes, texto, sonido, etc) en formato binario o del carácter.*	Admiten hasta 8 terabytes (8000 GB). Una tabla puede contener varias columnas de tipo LOB. Soportan acceso aleatorio. Las tablas con columnas de tipo LOB no pueden ser replicadas.
BLOB	*Permite almacenar datos **binarios no estructurados***	Admiten hasta 8 terabytes
CLOB	*Almacena datos de **tipo carácter***	Admiten hasta 8 terabytes
NCLOB	*Almacena datos de **tipo carácter***	Admiten hasta 8 terabytes. Guarda los datos según el juego de caracteres Unicode nacional.
BFILE	*Almacena **datos binarios no estructurados** en archivos del sistema operativo, fuera de la base de datos. Una columna BFILE almacena un localizador del archivo a uno externo que contiene los datos*	Admiten hasta 8 terabytes. El administrador de la base de datos debe asegurarse de que exista el archivo en disco y de que los procesos de Oracle tengan permisos de lectura para el archivo.
NUMBER	*Almacena **números fijos y en punto flotante***	Se admiten hasta 38 dígitos de precisión. Se puede declarar: ***nombre_columna NUMBER (precision, escala)*** Ej.- PRECIO NUMBER (4,2), donde precio sería un número de dos dígitos enteros y dos decimales. Rango: [00,00 – 99,99] Si no se indica la precisión se tomará en función del número a guardar. Si no se indica la escala se tomará escala cero. Para no indicar la precisión y sí la escala podemos utilizar: *nombre_columna NUMBER (*, escala)* Para introducir números que no estén en el formato estándar de Oracle se puede utilizar la función TO_NUMBER.
FLOAT	*Almacena tipos de datos numéricos en punto flotante*	Es un tipo NUMBER que sólo almacena números en punto flotante
DATE	*Almacena un punto en el tiempo (fecha y hora)*	El tipo de datos DATE de Oracle, almacena de longitud fija de siete octetos cada uno, correspondiendo al siglo, año, mes, día, hora, minuto, y al segundo. La entrada/salida de fechas, por defecto usa el formato *DD-MMM-AA*. Para cambiar el formato se utiliza el parámetro *NLS_DATE_FORMAT*.
TIMESTAMP	*Almacena datos de **tipo hora, fraccionando los segundos***	
TIMESTAMP WITH TIME ZONE	*Almacena datos de **tipo hora incluyendo la zona horaria** (explícita), fraccionando los segundos*	
TIMESTAMP WITH LOCAL TIME ZONE	*Almacena datos de **tipo hora incluyendo la zona horaria local (relativa), fraccionando los segundos***	Cuando se usa un SELECT para mostrar los datos de este tipo, el valor de la hora será ajustado a la zona horaria de la sesión actual
XMLType	***Tipo de datos abstracto**. En realidad, se trata de un CLOB.*	Se asocia a un esquema XML para la definición de su estructura.

3.3. Sintaxis básica CREATE TABLE

La sintaxis básica del comando que permite crear una tabla es la siguiente:

CREATE TABLE [esquema.] nombredeTabla (
columna1 Tipo_Dato,
columna2 Tipo_Dato,
...
columnaN Tipo_Dato);

3.4.Restricciones.

Una restricción es una condición que una o varias columnas deben cumplir obligatoriamente. Cada restricción llevará un nombre, si no se lo ponemos nosotros, lo hará Oracle o el SGBD que estemos usando. Es conveniente proporcionarle un nombre que nos ayude a identificarla y que sea único en el esquema de usuario.

Las recomendaciones para definir el nombre de una restricción son las siguientes:

- Tres letras para el nombre de la tabla.
- Carácter de subrayado.
- Tres letras con la columna afectada por la restricción.
- Carácter de subrayado.
- Dos letras con la abreviatura del tipo de restricción. La abreviatura puede ser:
  - PK = Primary Key.
  - FK = Foreign Key.
  - NN = Not Null.
  - UK = Unique.
  - CK = Check (validación).

Existen dos formas de especificar restricciones:

- ✓ **restricción de tabla**: al final, una vez definidas todas las columnas

```
CREATE TABLE NOMBRETABLA (
        Columna1 Tipo_Dato
        Columna2 Tipo_Dato
        ……
        ColumnaN Tipo_Dato,
        [CONSTRAINT nombredelarestricción PRIMARY KEY (Columna
        [CONSTRAINT nombredelarestricción NOT NULL (ColumnaX)],
        [CONSTRAINT nombredelarestricción UNIQUE (ColumnaX)],
        [CONSTRAINT nombredelarestricción REFERENCES nombreTabla [(columnaX)]],
        [CONSTRAINT nombredelarestricción CHECK condicion],
        [CONSTRAINT nombredelarestricción NOT NULL (ColumnaX)],
        [CONSTRAINT nombredelarestricción NOT NULL (ColumnaX)]);
```

```
CREATE TABLE USUARIOS (
        DNI VARCHAR2(9),
        login VARCHAR2 (25) NOT NULL,
        nombre VARCHAR2(30) NOT NULL,
        edad NUMBER (2),
        curso NUMBER,
        CONSTRAINT USU_DNI_PK PRIMARY KEY(DNI),
        CONSTRAINT USU_EDAD CHECK (EDAD BETWEEN 5 AND 20),
        CONSTRAINT USU_CURSO CHECK (CURSO IN (1, 2, 3)));
```

- ✓ **Restricción de columna**: como parte de la definición de columnas.

```
CREATE TABLE NOMBRETABLA (
Columna1 Tipo_Dato
        [CONSTRAINT nombredelarestricción]
        [NOT NULL]
        [UNIQUE]
        [PRIMARY KEY]
        [FOREIGN KEY]
        [DEFAULT valor]
        [REFERENCES nombreTabla [(columna [, columna ])]
        [ON DELETE CASCADE]]
        [CHECK condición],
Columna2 Tipo_Dato
        [CONSTRAINT nombredelarestricción]
        [NOT NULL]
        [UNIQUE]
        [PRIMARY KEY]
        [FOREIGN KEY]
        [DEFAULT valor]
        [REFERENCES nombreTabla [(columna [, columna ])]
        [ON DELETE CASCADE]]
        [CHECK condición],...  );
```

```
CREATE TABLE USUARIOS (
        DNI VARCHAR2(9) CONSTRAINT USU_DNI_PK PRIMARY KEY,
        login VARCHAR2 (25) NOT NULL,
        nombre VARCHAR2(30) NOT NULL,
        edad NUMBER (2) CONSTRAINT USU_EDAD CHECK (EDAD BETWEEN 5 AND 20),
        curso NUMBER CONSTRAINT USU_CURSO CHECK (CURSO IN (1, 2, 3)));
```

3.4.2. Restricción PRIMARY KEY.

Declaramos una PRIMARY KEY siguiendo el siguiente formato:

- ✓ **Para claves SIMPLES:**

```
CREATE TABLE USUARIOS (Login VARCHAR2 (25),
                       Nombre VARCHAR2 (25),
                       Apellidos VARCHAR2 (30),
                       F_Nacimiento DATE,
                       CONSTRAINT Usu_log_PK PRIMARY KEY(Login));
```

- ✓ **Para claves COMPUESTAS:**

```
CREATE TABLE USUARIOS (
        Nombre VARCHAR2 (25),
        Apellidos VARCHAR2 (30),
        F_Nacimiento DATE,
        CONSTRAINT Usu_PK PRIMARY KEY (Nombre, Apellidos, F_Nacimiento));
```

Cuando se crea una clave primaria, automáticamente se crea un índice que facilita el acceso a la tabla. La clave primaria tiene dos restricciones inherentes (por definición, no hay que declararlas):

- ***RESTRICCIÓN NOT NULL.***
- ***RESTRICCIÓN UNIQUE.***

3.4.3. Restricción FOREIGN KEY

Declaramos una FOREING KEY siguiendo el siguiente formato:

- ✓ **Nivel COLUMNA:**

```
CREATE TABLE USUARIOS (
        Cod_Usuario NUMBER (6),
        Nom_Usuario VARCHAR2(30),
        Cod_Partida NUMBER (8) CONSTRAINT Cod_Part_FK REFERENCES PARTIDAS(Cod_Partida));
```

Si el campo al que hace referencia es clave principal en su tabla no es necesario indicar el nombre del campo:

```
CREATE TABLE USUARIOS (
        Cod_Usuario NUMBER (6),
        Nom_Usuario VARCHAR2(30),
        Cod_Partida NUMBER (8) CONSTRAINT Cod_Part_FK REFERENCES PARTIDAS);
```

¡¡RECUERDA!!

- ***Las claves ajenas o foráneas son campos de una tabla que se relacionaban con la clave primaria de otra tabla.***
- ***La misión de la clave foránea es mantener la integridad referencial, es decir, antes de insertar un valor en la tabla donde fu declarada, el SGBD comprobará la existencia de ese valor en la tabla donde hace referencia.***

- ✓ **Nivel TABLA:**

```
CREATE TABLE USUARIOS (
        Cod_Partida NUMBER(8),
        F_Partida DATE,
        CONSTRAINT Partida_Cod_F_FK FOREIGN KEY (Cod_Partida, F_Partida) REFERENCES PARTIDAS);
```

¡¡OJO!! Cuando la clave foránea es compuesta debemos declararla a nivel de tabla, es decir, al final de todas las columnas. Cuando la clave foránea es simple, tenemos las dos opciones.

La integridad referencial puede dar los siguientes **ERRORES**:

- *Si hacemos referencia a una tabla que no está creada: Oracle buscará la tabla referenciada y al no encontrarla dará fallo* → ***CREAR EN PRIMER , LAS TABLAS QUE NO TIENEN CLAVES AJENAS.***
- *Si queremos borrar las tablas* → ***BORRAR PRIMERO LAS TABLAS CON CLAVES AJENAS***.

Tenemos otras soluciones y es añadir tras la cláusula REFERENCES la siguiente expresión:

- ✓ **ON DELETE CASCADE**: te permitirá borrar todos los registros cuya clave ajena sea igual a la clave del registro borrado.
- ✓ **ON DELETE SET NULL**: colocará el valor NULL en todas las claves ajenas relacionadas con la borrada.
- ✓ **ON DELETE SET DEFAULT:** colocará el valor que le indiquemos nosotros tras el default.

Estas expresiones también se usan para las modificaciones de valores:

✓ Imaginemos el siguiente caso:

ZONAS				
Letra	Tipo_barco	Num_barco	Profundidad	Ancho
A	Tipo1	20	2	6
B	Tipo1	15	2	6
C	Tipo2	30	3	12

#num_amarre	Lectura_agua	#Letra
1	15	A
2	35	A
11	20	B
23	23	C

✓ En el club naútico deciden cambiar las letras que definen a las ZONAS, ahora la A será la Z.
✓ Al tener dos amarres asignados a esta zona, se producirá un error en la base de datos al realizar esta modificación. Podemos evitarlo declarando la clave foránea de la siguiente forma:

```
CREATE TABLE AMARRES
        #num_amarre NUMBER(4) PRIMARY KEY,
        Lectura_agua NUMBER(6,2) NOT NULL,
        #letra VARCHAR2(1),
        CONSTRAINT ama_let_FK REFERENCES ZONAS ON DELETE SET NULL ON UPDATE CASCADE);
```

(*) Si se elimina la zona, el amarre se mantiene en la tabla porque no queremos perder los datos del contador del agua y en la zona se pondría un NULL. Si se modifica la letra de la zona, también se modifica en la tabla de amarres, que aparecerían asignados a la Z.

3.4.4. RESTRICCIÓN NOT NULL

Con esta restricción se prohíben los valores nulos para una columna en una determinada tabla. Se puede declarar a nivel tabla o columna:

```
CREATE TABLE USUARIOS (
   F_Nacimiento DATE,
   Login VARCHAR2(20),
  CONSTRAINT Usu_Fnac_NN NOT NULL);
```

```
CREATE TABLE USUARIOS (
 F_Nacimiento DATE NOT NULL,
 Login VARCHAR2(20));
```

Debemos tener cuidado con los valores nulos en las operaciones, ya que cualquier operación con un valor NULL, da como resultado NULL. Por ejemplo, 1*NULL es igual a NULL.

3.4.5. Restricción UNIQUE.

La restricción UNIQUE sobre un campo evite que este pueda contener valores repetidos. Las claves candidatas deben declararse con una restricción UNIQUE.

✓ **A nivel COLUMNA:**

```
CREATE TABLE USUARIOS (
Login VARCHAR2 (25) CONSTRAINT Usu_Log_UK UNIQUE);
```

Veamos otra forma:

```
CREATE TABLE USUARIOS (Login VARCHAR2 (25) UNIQUE);
```

- ✓ **A nivel TABLA:**

```
CREATE TABLE USUARIOS (
Login VARCHAR2 (25),
Correo VARCHAR2 (25),
CONSTRAINT Usuario_UK UNIQUE (Login, Correo));
```

Cuando la restricción afecta a un campo compuesto, siempre se declaran al final. Cuando afecta a un campo simple, tenemos las dos opciones.

3.4.6. Restricción CHECK.

La restricción CHECK permite establecer condiciones que debe cumplir una determinada columna, es decir, los valores de los datos que contienen. Veamos unos ejemplos declarando la restricción a nivel columna y al nivel tabla:

```
CREATE TABLE USUARIOS (
        login VARCHAR2 (25), not null primary key,
        nombre VARCHAR2 (30) not null check (nombre=upper(nombre)),
        edad NUMBER (2) check (edad >=18),
        curso number check (curso in (1,2,3)));

CREATE TABLE USUARIOS (
        login VARCHAR2 (25) NOT NULL,
        nombre VARCHAR2(30) NOT NULL,
        edad NUMBER (2),
        curso NUMBER,
        CONSTRAINT CLAVE_P PRIMARY KEY(DNI),
        CONSTRAINT COMP_EDAD CHECK (EDAD BETWEEN 5 AND 20),
        CONSTRAINT NOMBRE_MAYUS CHECK (NOMBRE = UPPER(NOMBRE)),
        CONSTRAINT COMP_CURSO CHECK (CURSO IN (1, 2, 3)));
```

3.4.7. Restricción DEFAULT.

La restricción DEFAULT nos permite dar un valor por defecto a un campo. El usuario sólo tiene que dar valor al campo si desea dar un valor diferente al valor por defecto.

```
CREATE TABLE USUARIOS (Pais VARCHAR2(20) DEFAULT ' España ' );
```

En las especificaciones de DEFAULT vamos a poder añadir distintas expresiones: constantes, funciones SQL y variables. Si queremos incluir en un campo la fecha actual, independientemente del día en el que estemos, podremos utilizar la función SYSDATE como valor por defecto:

CREATE TABLE USUARIOS (Fecha_ingreso DATE DEFAULT SYSDATE);

4. Eliminación de TABLAS.

Para eliminar una tabla utilizaremos el comando **DROP TABLE.**

DROP TABLE NombreTabla [CASCADE CONSTRAINTS];

Esta instrucción borrará la tabla de la base de datos incluido sus datos (filas). También se borrará toda la información que existiera de esa tabla en el Diccionario de Datos.

La opción CASCADE CONSTRAINTS (**observa que CONSTRAINTS en este caso acaba en S**) se puede incluir para los casos en que alguna de las columnas sea clave ajena en otra tabla secundaria, lo que impediría su borrado. Al colocar esta opción las restricciones donde es clave ajena se borrarán antes y a continuación se eliminará la tabla en cuestión.

Vamos a eliminar la tabla con la que hemos estado trabajando:

DROP TABLE USUARIOS;

Ten cuidado al utilizar este comando, el borrado de una tabla es irreversible y no hay una petición de confirmación antes de ejecutarse, además sólo podrás borrar aquellas tablas sobre las que tengas permiso de borrado.

Oracle dispone de la orden TRUNCATE TABLE que te permitirá eliminar los datos (filas) de una tabla sin eliminar su estructura, no genera información de retroceso (ROLLBACK), es decir, una sentencia TRUNCATE no se puede anular, como tampoco activa disparadores, a diferencia de lo que hace DELETE que sí los activa. Por eso, la eliminación de filas con la orden TRUNCATE es más rápida que con DELETE.

TRUNCATE TABLE USUARIOS;

5. MODIFICACIONES

Podemos añadir, eliminar o modificar cualquier elemento de la tabla, usamos el comando ***ALTER TABLE***.

5.1. Cambiar de nombre una TABLA.

RENAME NombreViejo TO NombreNuevo; *(*) Sin ALTER*.

5.2. MODIFICACIONES sobre COLUMNAS.

5.2.1. AÑADIR COLUMNAS(ADD).

Usamos **ADD**, siguiendo la siguiente sintaxis:

```
ALTER TABLE NombreTabla ADD (
ColumnaNueva1 Tipo_Datos [Propiedades]
[, ColumnaNueva2 Tipo_Datos [Propiedades] ... );
```

A la hora de añadir una columna a una tabla hay que tener en cuenta varios factores:

- La columna se añadirá al final de la tabla.
- Si la columna NO está definida como NOT NULL, se le puede añadir en cualquier momento.
- Si la columna está definida como NOT NULL, tenemos dos opciones:
  - Añadir la columna sin especificar NOT NULL, dar valor a la columna para cada una de las filas y finalmente, modificar la columna a NOT NULL.
  - Crear la columna con un valor por defecto.

EJEMPLO: partimos de la siguiente tabla.

```
CREATE TABLE USUARIOS (
   login VARCHAR2 (25), not null primary key,
   nombre VARCHAR2 (30) not null check (nombre=upper(nombre)),
   edad NUMBER (2) check (edad >=18),
   curso number check (curso in (1,2,3)) );
```

Vamos a suponer que hemos introducido datos en la tabla. Si añadimos la columna **tipo** not null:

```
ALTER TABLE USUARIOS ADD (tipo Varchar2(5) not null);
```

Tendremos un error, ya que para los valores que hemos añadido anteriormente, el campo tipo no puede tomar valor y dará un error.

Para solucionar este error podemos dar un valor por defecto al campo que se asignará a todos los campos creados anteriormente:

```
ALTER TABLE USUARIOS ADD (tipo Varchar2(5) default ' ' not null);
```

5.2.2. Borrar una columna

Para eliminar columnas de una tabla, utilizaremos la cláusula DROP COLUMN.

Mediante el borrado, además de la definición de la columna, se eliminarán todos los datos que contuviera. No se puede eliminar una columna si es la única que forma la tabla, para ello tendremos que borrar la tabla directamente.

La sintaxis es la siguiente:

ALTER TABLE NombreTabla DROP COLUMN (Columna1 [, Columna2, ...]);

Veamos un ejemplo:

ALTER TABLE USUARIOS DROP COLUMN tipo;

5.2.3. MODIFICAR COLUMNAS.

Para modificar columnas de una tabla, utilizaremos la cláusula MOFIFY.

Podemos modificar el tipo de datos y las propiedades de una columna. Todos los cambios son posibles si la tabla no contiene datos.

Si la tabla no está vacía, hay que tener en cuenta las siguientes consideraciones:

- ✓ Se puede aumentar la longitud de una columna en cualquier momento.
- ✓ Al disminuir la longitud de una columna que tiene datos no se puede dar menor tamaño que el máximo valor almacenado.
- ✓ Es posible aumentar o disminuir el número de posiciones decimales en una columna de tipo NUMBER.
- ✓ Si la columna es NULL en todas las filas de la tabla, se puede disminuir la longitud y modificar el tipo de dato.
- ✓ La opción MODIFY ... NOT NULL sólo será posible cuando la tabla no contenga ninguna fila con valor nulo en la columna que se modifica.

La sintaxis es la siguiente:

ALTER TABLE NombreTabla MODIFY (Columna1 TipoDatos [propiedades] [, columna2 TipoDatos [propiedades] ...]);

Veamos un ejemplo:

ALTER TABLE USUARIOS MODIFY (nombre VARCHAR2(25), tipo VARCHAR2 (10)) ;

Como podemos ver, podemos modificar varias columnas al mismo tiempo.

Otro elemento a destacar es que no hace falta indicar los valores que tenía anteriormente el campo, sólo hace falta indicar los nuevos

5.2.4. RENOMBRAR UNA COLUMNA.

Para modificar el nombre de una columna de una tabla, utilizaremos la cláusula RENAME COLUMN.

La sintaxis es la siguiente:

ALTER TABLE NombreTabla RENAME COLUMN NombreAntiguo TO NombreNuevo;

Vemos un ejemplo:

ALTER TABLE USUARIOS RENAME COLUMN nombre TO nombre_usuario;

5.3. Modificación de Restricciones.

5.3.1. Añadir Restricciones.

Podemos añadir diferentes restricciones a las tablas:

- ✓ Añadiendo una restricción CHECK.
  Restricción de CHECK para que la EDAD sea mayor de 18 en la tabla USUARIOS:

  ALTER TABLE USUARIOS ADD CONSTRAINT EDAD_CK CHECK (EDAD > 18);

- ✓ Añadiendo una restricción UNIQUE.
  Restricción de DNI único en la tabla de USUARIOS:

  ALTER TABLE USUARIOS ADD CONSTRAINT DNI_UQ UNIQUE(DNI);

- ✓ Añadiendo una restricción PRIMARY KEY.
  Restricción de clave primaria a la columna COD_USUARIO de la tabla USUARIOS:

  ALTER TABLE USUARIOS ADD CONSTRAINT PK_USUARIO PRIMARY KEY(COD_USUARIO);

  Restricción de clave primaria a la columna COD_ALMACEN de la tabla ALMACEN:

  ALTER TABLE ALMACEN ADD CONSTRAINT PK_ALMACEN PRIMARY KEY(COD_ALMACEN);

- ✓ Añadiendo una restricción FOREIGN KEY.
  Restricción de clave ajena a la columna DEPT_NO de la tabla EMPLE que referencia a la clave primaria de la tabla DEPART:

  ALTER TABLE USUARIOS ADD CONSTRAINT FK_COMP_TELEFONICA FOREIGN KEY(COD_COMP_TELEFONICA) REFERENCES COMP_TELEFONICA ON DELETE CASCADE;

5.3.2. Borrar restricciones.

Para borrar una restricción, utilizaremos la cláusula **DROP CONSTRAINT**.

Igual que hemos visto en el apartado anterior podemos borrar diferentes tipos de restricciones: FOREING KEY, NOT NULL, UNIQUE….

La sintaxis sería idéntica a los casos anteriores indicando el nombre de la restricción que queremos eliminar. De forma general la sintaxis sería:

ALTER TABLE NombreTabla DROP CONSTRAINT NombreRestriccion;

Veamos algunos ejemplos:

ALTER TABLE USUARIOS DROP CONSTRAINT EDAD_CK;

ALTER TABLE USUARIOS DROP CONSTRAINT DNI_UQ;

ALTER TABLE USUARIOS DROP CONSTRAINT FK_COMP_TELEFONICA;

5.3.3. Modificar restricciones.

Para modificar el nombre de una restricción, utilizaremos la cláusula RENAME.

La sintaxis para renombrar una restricción es la siguiente:

ALTER TABLE NombreTabla RENAME CONSTRAINT NombreViejo TO NombreNuevo;

Veamos algunos ejemplos:

ALTER TABLE USUARIOS RENAME CONSTRAINT EDAD_CK TO CK_EDAD;

ALTER TABLE USUARIOS DROP CONSTRAINT DNI_UQ to UQ_DNI;

5.3.4. ACTIVAR Y DESACTIVAR RESTRICCIONES.

Para activar y desactivar una restricción, utilizaremos las cláusulas `ENABLE y DISABLE`.

A veces es conveniente desactivar temporalmente una restricción para hacer pruebas o porque necesitemos saltarnos esa regla. Para ello usaremos esta sintaxis:

ALTER TABLE NombreTabla DISABLE CONSTRAINT NombreRestriccion [CASCADE];

Para activar de nuevo la restricción:

ALTER TABLE NombreTabla ENABLE CONSTRAINT NombreRestriccion [CASCADE];

Veamos un ejemplo

ALTER TABLE USUARIOS DISABLE CONSTRAINT EDAD_CK;

ALTER TABLE USUARIOS ENABLE CONSTRAINT EDAD_CK;