Índice

LENGUAJE DE DEFINICION DE DATOS (DDL)

1.CREACION DE TABLAS

Para crear una tabla en la base de datos se utiliza el comando **CREATE TABLE**. Este comando insertará la definición de la tabla en el diccionario de datos y reservará espacio en los ficheros de la base de datos para la introducción de la información.

En el comando se deberá indicar:

- el nombre de la tabla
- por cada columna de la tabla
  - el nombre de la columna
  - el tipo de datos (CHAR,VARCHAR2, NUMBER, DATE, LONG)
  - el tamaño máximo de las columnas de tipo numérico y alfanumérico (sólo es obligatorio para las de tipo alfanumérico)
  - si se permite que la columna contenga valores nulos

Las anteriores características se almacenan en las tablas del diccionario de datos para comprobaciones posteriores.

El comando CREATE TABLE se puede utilizar con dos formatos distintos.

En el primer formato tenemos que definir una por una todas las columnas de la tabla.

```
CREATE TABLE  <nombre_de_la_tabla>

(<nombre_de_la_columna1> <tipo_de_datos> [(tamaño)] [NOT NULL],
 <nombre_de_la_columna2> <tipo_de_datos> [(tamaño)] [NOT NULL],
...
<nombre_de_la_columna254> <tipo_de_datos> [(tamaño)] [NOT NULL]);
```

Ejemplo:

```
SQL>CREATE TABLE libros
2       ( título                  VARCHAR2(40)      NOT NULL,
3        fecha_de_publicación     DATE,
4       precio                    NUMBER,
5        resumen                  LONG,
6        autor                    VARCHAR2(40) );
```

Table created.

Se puede comprobar que la tabla no contiene ninguna fila.

```
SQL>SELECT * FROM libros;

no records selected.
```

Con el segundo formato se copia la definición de tablas ya existentes en el diccionario de datos, y además se copian también las filas de esas tablas.

CREATE TABLE <nombre_de_la_tabla> [< lista_nuevas_columnas >]
AS SELECT < lista_de_columnas_existentes >
FROM < lista_de_tablas_existentes >
WHERE < lista_de_condiciones >;

< lista_nuevas_columnas >
Nombre de las nuevas columnas de la tabla separados por comas.
< lista_de_columnas_existentes >
Nombre de columnas de tablas que aparecen en el FROM.
< lista_de_tablas_existentes >
Nombre de las tablas de las que se va a copiar la definición de la columnas.
< lista_de_condiciones >
Condiciones que deben cumplir las filas que se van a copiar.

Ejemplo:

```
CREATE TABLE libros_baratos
AS SELECT título, precio, fecha_de_publicación
FROM libros
WHERE precio <= 500;

Table created.
```

Cuando se utiliza el segundo formato no se pueden especificar tipos de datos, pero se pueden utilizar las funciones **TO_CHAR**, **TO_DATE** y **TO_NUMBER** para cambiar el tipo de datos de las antiguas columnas.

Nombres de tablas y columnas

NOMBRES UNICOS

- los nombres de las tablas de un usuario no se pueden repetir
- los nombres no pueden coincidir con palabras clave de ORACLE
- los nombres de las columnas deben ser únicos dentro de una tabla

CARACTERES

- el primer carácter de un nombre debe ser una letra mayúscula o minúscula
- el resto de los caracteres es libre con la excepción de las comas
- los nombres pueden tener como máximo 30 caracteres
- si se incluye el nombre entre dobles comillas no se tienen por qué tener en cuenta las restricciones anteriores

1.1 TIPOS DE DATOS ORACLE

Los datos de la base de datos ORACLE se almacenan en las filas de las tablas. Una tabla puede contener hasta 254 columnas y cada columna puede ser de uno de los siguientes tipos de datos:

- VARCHAR2
- NUMBER
- DATE
- LONG

Tipo de datos alfanumérico **(VARCHAR2)**

VARCHAR2 (n)

Los datos de tipo alfanumérico pueden contener letras, números, signos de puntuación y caracteres especiales como el +, -, $ o el %.

La longitud máxima de una columna de tipo carácter se especifica en la definición de la columna y podrá ser como máximo de 240 caracteres.

No importa especificar tamaños muy grandes puesto que los datos se almacenan como cadenas de longitud variable, no se reserva espacio para el tamaño máximo.

Tipo de datos numérico (**NUMBER**)

NUMBER (n,p)

Se permiten números, el signo y un punto decimal. El primer número dentro del paréntesis indica el número total de dígitos que se pueden almacenar y p los dígitos que habrá a la derecha del punto decimal. Si se omite la especificación de tamaño se podrán introducir hasta 105 dígitos, pero se almacenarán únicamente 40.

Tipo de datos fecha **(DATE)**

DATE

Los datos de tipo fecha se almacenan como una cadena de caracteres de longitud fija de siete bytes. Para cada dato de tipo fecha se almacena la siguiente información:

- Siglo (1 byte)
- Año (1 byte)
- Mes (1 byte)
- Día (1 byte)
- Hora (1 byte)
- Minuto (1 byte)
- Segundo (1 byte)

Si al insertar un dato de tipo fecha no se especifica la hora con la función **TO_DATE,** ORACLE tomará por defecto las doce de la mañana.

El rango de valores está entre el uno de enero del año 4712AC y el 31 de diciembre del año 4712DC.

Tipo de datos **LONG**

Las columnas de tipo LONG son cadenas de caracteres de longitud variable de 64K de tamaño. Se utilizan para almacenar cadenas de caracteres o incluso pequeños documentos.

Las columnas de este tipo están sujetas a una serie de restricciones:

- sólo se admite una columna de este tipo por tabla
- tablas con columnas de este tipo no se pueden agrupar
- columnas de este tipo no se pueden indexar
- no se se pueden utilizar este tipo de columnas en cláusulas WHERE, GROUP BY, CONNECT BY, ORDER BY, DISTINCT
- no se pueden utilizar funciones con columnas de este tipo
- columnas de este tipo no pueden aparecer en columnas seleccionadas en consultas anidadas
- no pueden aparecer en ningún tipo de expresiones
- no pueden aparecer en consultas que estén unidas con otras consultas mediante UNION, MINUS, INTERSECT

1.2 RESTRICCIONES DE INTEGRIDAD (CONSTRAINT = RESTRICCIÓN)

La versión 6 de ORACLE admite en la sintaxis de las sentencias SQL CREATE TABLE y ALTER TABLE la especificación de restricciones de integridad de tabla y de columna. Estas restricciones se introducen en las tablas del diccionario de datos para que se verifiquen en posteriores versiones de ORACLE, puesto que en la versión actual no se comprueban (a excepción de NOT NULL).

Las restricciones se deben comprobar al ejecutar cualquier consulta de manipulación de datos y se deben cumplir para que las sentencias DML tengan éxito.

Las restricciones de tabla forman parte de la definición global de la tabla, se especifican después de la definición de las columnas y se pueden referir a una o más columnas de la tabla:

```
[ { UNIQUE | PRIMARY KEY }  ( <lista_de_columnas> )

[CONSTRAINT <nombre_restricción>] ]   [ FOREIGN KEY ( <lista_de_columnas> )

REFERENCES [<usuario>.]<tabla> [ (lista_de_columnas) ]

[CONSTRAINT<nombre_restricción>]][CHECK(<condición>)[CONSTRAINT<nombre_restricción>] ]
```

Las restricciones de integridad **de columna** se refieren a la columna en cuya descripción se encuentra la restricción:

```
[ { NULL | NOT NULL } [CONSTRAINT <nombre_restricción>] ]
[ { UNIQUE | PRIMARY KEY }
[CONSTRAINT <nombre_restricción>] ]
[ FOREIGN KEY REFERENCES [<usuario>.]<tabla> [(columna)]
  [CONSTRAINT <nombre_restricción>] ]
[ CHECK (<condición>) [CONSTRAINT <nombre_restricción>] ]
```

Si a una restricción no se le da nombre el sistema se encarga de dárselo con el siguiente formato: SYS_Cn, donde la n es un número para garantizar la unicidad de los nombres de restricciones.

Tipos de restricciones, **CONSTRAINT**:

NULL: Indica si una columna puede ser o no nula en una tabla. Not null indica **prohibir nulos.**

```
CREATE TABLE cliente(dni VARCHAR2(9) NOT NULL);
```

(el nombre lo coloca oracle)

```
CREATE TABLE cliente
(dni VARCHAR2(9)
   CONSTRAINT cliente_dni_sinnulos  NOT NULL);
```

(le ponemos nombre a la restricción nosotros)

UNIQUE: Indica que todas las filas de una tabla deben tener un valor distinto para esa columna o columnas que deben ser NOT NULL y no pueden ser PRIMARY KEY. Los **valores únicos** son las claves candidatas de la tabla y se pueden declarar también de dos formas.

```
CREATE TABLE cliente(dni VARCHAR2(9) UNIQUE) ;
```

(el nombre lo coloca oracle)

```
CREATE TABLE cliente(dni VARCHAR2(9)
CONSTRAINT cliente_dni_u  UNIQUE);
```

(le ponemos nombre a la restricción nosotros)

PRIMARY KEY: Indica que esa columna o columnas constituyen la **clave primaria** de la relación y digamos que lleva implícito que sea NOT NULL y UNIQUE. Se puede especificar sólo una vez en cada tabla. Si la clave primaria es una única columna se puede especificar como restricción de tabla o de columna, si la clave primaria la forman más de una columna se puede especificar sólo como restricción de tabla.

```
CREATE TABLE cliente(dni VARCHAR2(9) PRIMARY KEY);
```

(el nombre lo coloca oracle)

```
CREATE TABLE cliente(dni VARCHAR2(9)
CONSTRAINT cliente_dni_PK   PRIMARY KEY);
```

(le ponemos nombre a la restricción nosotros)

FOREIGN KEY/ REFERENCES: Indica que esa columna o columnas son una clave ajena de la columna o columnas que son clave primaria de la tabla que se indica detrás de la cláusula REFERENCES. **Clave secundaria o foránea** y no va a permitir inserciones ni actualizaciones si el valor correspondiente no existe ya en la otra tabla. **No va a permitir borrados que violen la restricción de integridad.**

```
CREATE TABLE alquiler(
        dni VARCHAR2(9),
        cod_pelicula NUMBER(5),
        CONSTRAINT alquiler_pk   PRIMARY KEY(dni,cod_pelicula),
        CONSTRAINT dni_fk        FOREIGN KEY (dni)
                REFERENCES clientes(dni),
        CONSTRAINT pelicula_fk   FOREIGN KEY (cod_pelicula)
                REFERENCES peliculas(cod));
```

Esta completa forma de crear la tabla alquiler incluye sus claves foráneas, el campo *dni* hace referencia al campo dni de la tabla clientes y el campo *cod_pelicula* que hace referencia al campo *cod* de la tabla *peliculas*. También hubiera bastado con indicar sólola tabla a la que hacemos referencia, si no se indican los campos relacionados de esa tabla, se toma su clave principal (que es lo normal).

Importante mantener una **nomenclatura que indique que tabla y que tipo de índice o restricción tiene ese campo, _FK , _PK, _IX, etc.**

La integridad referencial es una herramienta imprescindible de las bases de datos relacionales. Pero provoca varios problemas. Por ejemplo, si borramos un registro en la tabla principal que está relacionado con uno o varios de la secundaria ocurrirá un error, ya que de permitírsenos borrar el registro ocurrirá fallo de integridad (habrá claves secundarios refiriéndose a una clave principal que ya no existe).

Por ello Oracle nos ofrece dos soluciones a añadir tras la cláusula REFERENCES:

ON DELETE SET NULL. Coloca nulos todas las claves secundarias relacionadas con la borrada.
ON DELETE CASCADE. Borra todos los registros cuya clave secundaria es igual que la clave del registro borrado.

Si no se indica esta cláusula, no se permite el borrado de registros relacionados.

El otro problema ocurre si se desea cambiar el valor de la clave principal en un registro relacionado con claves secundarias. En muchas bases de datos se implementan soluciones consistentes en añadir ON UPDATE CASCADE u ON UPDATE SET NULL. Oracle no implementa directamente estas soluciones. Por lo que hay que hacerlo de otra forma. Las soluciones son:

Implementar un **TRIGGER** para que cuando se actualice el registro se actualicen las claves secundarias. El mecanismo de funcionamiento es parecido al que se muestra en el siguiente párrafo

Añadir un registro igual que el que se quiere cambiar en la tabla principal, pero con el nuevo valor de la clave. Mediante una instrucción UPDATE actualizar a ese valor de clave todos los registros de la tabla secundaria cuyo valor coincida con la antigua clave. Finalmente borrar el registro en la tabla principal con el valor antiguo de la clave.

- **CHECK:** Indica que los valores de una columna o un grupo de columnas de una fila deben cumplir la condición antes de que se pueda insertar o actualizar esa fila.

2. MODIFICACION DE LA ESTRUCTURA DE UNA TABLA

Una de la principales características de ORACLE es la posibilidad que ofrece de modificar la definición de las tablas después de haber sido creadas.

Para ello se dispone del comando ALTER TABLE.

2.1 Añadir nuevas columnas

Para añadir nuevas columnas a la definición de una tabla el comando ALTER TABLE tiene el siguiente formato:

```
ALTER TABLE  <nombre_de_la_tabla>
ADD  ( <nombre_de_la_nueva_columna1>  <tipo_de_datos>,
          <nombre_de_la_nueva_columna2>  <tipo_de_datos>,
          ... );
```

Ejemplo:

```
ALTER TABLE libros
ADD   ( género        VARCHAR2(30),
país    VARCHAR2(10) );
```

No se pueden especificar las columnas como no nulas ya que necesariamente contendrán valores nulos en un principio.

2.2 Modificar la definición de una columna

Para modificar la definición de una columna utilizaremos el siguiente formato del comando ALTER TABLE:

```
ALTER TABLE  <nombre_de_la_tabla>
MODIFY  ( <nombre_de_la_columna1> <tipo_de_datos> [NULL/NOT NULL],
        <nombre_de_la_columna2> <tipo_de_datos> [NULL/NOT NULL],
                ... );
```

Ejemplo:

```
ALTER TABLE  libros
MODIFY  ( título CHAR(100) );
```

2.3 Restricciones al modificar la estructura de una tabla

Se podrá siempre:
- cambiar la definición de una columna para permitir valores nulos
- aumentar el tamaño de definición de una columna
- añadir una columna que permita valores nulos

Si en una tabla no hay filas que tengan valores en la columna que van a ser modificada, se podrá:
- disminuir el tamaño de la columna
- cambiar el tipo de dato de la columna

Si en una tabla todas las filas tienen valores no nulos en la columna que va a ser modificada, se podrá:
- cambiar la definición de la columna para que no pueda contener valores nulos

Si una tabla está vacía, se podrá:
- añadir una columna que no admita nulos (se añade primero la columna y después se modifica para que no admita valores nulos)

3. AÑADIR COMENTARIOS

Para incluir comentarios sobre tablas y columnas en el diccionario de datos se utiliza el comando **COMMENT.**

```
COMMENT ON TABLE <nombre_de_la_tabla> IS 'texto';
```

```
COMMENT ON COLUMN
<nombre_de_la_tabla>.<nombre_de_la columna> IS 'texto';
```

Ejemplo:

```
COMMENT ON TABLE libros IS 'INFORMACION SOBRE LOS LIBROS';

COMMENT ON COLUMN género.libros
        IS 'GENERO LITERARIO DEL LIBRO EN CUESTION';
```

El propietario es el único autorizado en hacer comentarios sobre sus tablas y/o columnas.

El texto del comentario puede tener 240 caracteres como máximo.

4. SINONIMOS

Dado que es difícil trabajar con nombres demasiado largos o complicados, el usuario puede crear nombres alternativos para esas tablas.

```
CREATE [ PUBLIC ] SYNONYM <nombre_del_sinónimo>
FOR  [propietario.]<nombre_de_la_tabla o vista>;
```

Las tablas o vistas se podrán referenciar por su nombre o por el de sus sinónimos.

El administrador de la base de datos puede crear sinónimos de uso general para tablas y vistas que sean utilizadas por la totalidad de los usuarios, usando para ello la palabra clave PUBLIC.

Los usuarios no pueden crear tablas que tengan el mismo nombre que sinónimos públicos.

Ejemplo:

```
CREATE SYNONYM  empleado  FOR  scott.employee;
```

A partir de este momento la tabla 'emp' del usuario 'scott' se podrá referenciar también con el nombre 'empleado'.

```
SELECT * FROM empleado;
```

Para borrar un sinónimo se utiliza el siguiente comando SQL:

```
DROP [PUBLIC] SYNONYM  <nombre_del_sinónimo>;
```

La tabla a la que se refiere el sinónimo no se verá afectada.

5. CAMBIO DEL NOMBRE A UNA TABLA

Para cambiar de nombre a una tabla sin que se vea afectado el almacenamiento de sus datos se utiliza el comando RENAME.

RENAME <antiguo_nombre> TO <nuevo_nombre>;

Ejemplo:

```
RENAME employee TO empleados;
```

En el diccionario de datos la tabla quedará reflejada con el nuevo nombre, pero no tendrá ninguna repercusión sobre el almacenamiento de los datos.

La tabla no se podrá volver a referenciar por el antiguo nombre.

El propietario es el único que puede cambiar de nombre a una tabla.

6. BORRADO DE TABLAS

Para borrar la definición de las tablas del diccionario de datos se utiliza el comando DROP TABLE.

DROP TABLE <nombre_de_la_tabla>;

Cuando se ejecuta este comando se producen los siguientes efectos:

- la definición de la tabla será borrada del diccionario de datos
- el espacio que ocupaba la tabla se recupera para nuevos datos de la base de datos
- se borrarán todas las filas de la tabla sin previa advertencia
- no se podrán recuperar ni la definición ni los datos de la tabla
- una tabla sólo la puede borrar su propietario

Si borramos la tabla que habíamos creado con anterioridad:

```
DROP TABLE libros_baratos;
```

```
Table dropped.
```

A continuación intentamos consultar esa tabla:

```
SELECT *
FROM libros_baratos;
```

ERROR at line 1: table or view does **not exist**

Diferencia entre la sentencia DROP TABLE y DELETE

La diferencia entre la sentencia DROP TABLE y la sentencia DELETE consiste en que la sentencia DROP TABLE además de borrar las filas de la tabla, también borra su definición en el diccionario de datos.

La sentencia DROP TABLE, por ser un comando del lenguaje de definición de datos se valida de forma automática por lo que no se podrá volver al estado anterior mediante un "rollback".

La sentencia DELETE borra todas las filas de una tabla si no imponemos ninguna condición, pero al ser un comando de manipulación de datos se puede volver la situación anterior si no se han validado los cambios.

7. SECUENCIAS

El generador de secuencias se utiliza para generar números secuenciales que son utilizados con el fin de manipular las tablas (p.e. generación automática de claves primarias o coordinación de claves para el acceso a varias tablas.

En primer lugar hay que ejecutar una sentencia SQL que defina la secuencia especificando el nombre de la secuencia, si es descendente o ascendente, los valores máximos y mínimos o la ciclicidad de la secuencia. Una vez se ha creado una secuencia, ésta la pueden utilizar múltiples usuarios para múltiples tablas.

Creación de una secuencia:

```
CREATE SEQUENCE < nombre_secuencia >

[ INCREMENT BY entero ]
[ START WITH entero ]
[ MAXVALUE entero | NOMAXVALUE ]
[ MINVALUE entero | NOMINVALUE ]
[ CYCLE | NOCYCLE ]
[ CACHE entero | NOCACHE ]
[ ORDER | NORDER] ;
```

Por defecto la sentencia de creación de una secuencia será la siguiente:

```
CREATE SEQUENCE defecto
        INCREMENT BY 1
        START WITH 1
        NOMAXVALUE
        NOCYCLE
        CACHE 20
        NOORDER ;
```

Las pseudocolumnas CURRVAL y NEXTVAL se utilizan para obtener el valor actual y siguiente de la secuencia y se pueden especificar en sentencias de consulta, de inserción y de actualización, refiriéndose a ellos como nombre_secuencia.currval o nombre_secuencia.nextval.

Utilización de NEXTVAL y CURRVAL:

- No se puede utilizar ninguna de las dos pseudocolumnas en clausulas de condición ni en un nivel de anidamiento por debajo del primero, ni en consultas que contengan GROUP BY, ORDER BY, DISTINCT o CONNECT BY. Tampoco se pueden utilizar en vistas.
- NEXTVAL y CURRVAL se pueden utilizar en la cláusula VALUES de las sentencias de inserción, en la parte derecha de la cláusula SET de las sentencias de actualización y en el nivel exterior de una consulta con las restricciones comentadas con anterioridad. También se puede utilizar en el SELECT exterior de las sentencias CREATE TABLE AS, INSERT ... SELECT y a la derecha de la cláusula SET.

Para cambiar las opciones especificadas en la creación de la secuencia se puede utilizar la sentencia ALTER SEQUENCE, que tiene la misma sintaxis que la de creación (hay que tener en cuenta que no se puede poner MAXVALUES por debajo del actual).

8. DICCIONARIO DE DATOS

En el diccionario de datos se almacena la información sobre todos los objetos de la base de datos (tablas, vistas, sinónimos, índices, agrupamientos, secuencias, columnas, etc...).

El diccionario de datos es consultado tanto por ORACLE para hacer comprobaciones, como por los usuarios para obtener información sobre la base de datos.

El diccionario de datos es actualizado automáticamente por ORACLE sin interrumpir el trabajo de otros usuarios cuando se ejecutan comandos del lenguaje de definición de datos (DDL) o del lenguaje de control de datos (DCL).

Los usuarios pueden acceder a la información del diccionario de datos a través de las vistas que se crean al inicializar la base de datos con este fin.

Para visualizar las vistas que puede consultar el usuario se puede ejecutar la siguiente sentencia SQL.

SQL>SELECT * FROM DICTIONARY ;

Las vistas del diccionario de datos se dividen en tres grandes grupos según el prefijo que lleven, que nos dará una indicación de la información contenida en la vista:

USER_* Este tipo de vistas se refieren a los objetos creados por un usuario en concreto o los privilegios concedidos por él.

ALL_* Este tipo de vistas se refieren a los objetos creados por el usuario y a los que puede acceder porque se le han concedido privilegios de acceso.

DBA_* Este tipo de vistas sólo las puede consultar los usuarios con privilegios de administrador puesto que contienen información general de la base de datos.

Las vistas del diccionario de datos se consultan como cualquier otra tabla del usuario.

Para visualizar las tablas, vistas, sinónimos y agrupamientos creados por el usuario se puede ejecutar la siguiente consulta:

```
SQL>SELECT OBJECT_NAME, OBJECT_TYPE FROM USER_OBJECTS;

OBJECT_NAME                    OBJECT_TYPE
------------------------------ -----------
LIBROS                          TABLE
PRIMA                           TABLE
SALGRADE                         TABLE
EMP                            TABLE
DEPT                           TABLE
LIBROS_BARATOS                  TABLE
ESTADISTICA                     VIEW
ESTADISTICA1                    VIEW
EMPLEADO                       SYNONYM

9 records selected.
```

9. EJERCICIOS PROPUESTOS

1º. Crear las siguientes tablas de acuerdo con las restricciones que se mencionan:

Tabla TIENDAS

Campo	tipo	long	NULOS?
NIF	ALFANUM	10	NO ADMITE
NOMBRE	ALFANUM	20	
DIRECCION	ALFANUM	20	
POBLACION	ALFANUM	20	
PROVINCIA	ALFANUM	20	
COD_POSTAL	NUMERICO	5	

- La clave primaria es NIF.
- PROVINCIA ha de almacenarse en mayúsculas.

Tabla FABRICANTES

Campo	tipo	long	NULOS?
COD_FABRICANTE	NUMERICO	3	NO ADMITE
NOMBRE	ALFANUM	15	
PAIS	ALFANUM	15	

- La clave primaria es COD_FABRICANTE.
- Las columnas NOMBRE y PAIS han de almacenarse en mayúsculas.

Tabla ARTICULOS

Campo	tipo	long	NULOS?
ARTICULO	ALFANUM	20	NO ADMITE
COD_FABRICANTE	NUMERICO	3	NO ADMITE
PESO	NUMERICO	3	NO ADMITE
CATEGORIA	ALFANUM	10	NO ADMITE
PRECIO_VENTA	NUMERICO	4	
PRECIO_COSTO	NUMERICO	4	
EXISTENCIAS	NUMERICO	5	

- La clave primaria está formada por las columnas ARTICULO, COD_FABRICANTE, PESO y CAREGORIA.
- COD_FABRICANTE es clave ajena que referenncia a la tabla FABRICANTES.
- PRECIO_VENTA, PRECIO_COSTO y PESO han de ser >0.
- CATEGORIA ha de ser ‘primera’, ‘segunda’ o ‘tercera’.

Tabla PEDIDOS

Campo	tipo	long	NULOS?
NIF	ALFANUM	10	NO ADMITE
ARTICULO	ALFANUM	20	NO ADMITE
COD_FABRICANTE	NUMERICO	3	NO ADMITE
PESO	NUMERICO	3	NO ADMITE
CATEGORIA	ALFANUM	10	NO ADMITE
FECHA_PEDIDO	FECHA		NO ADMITE
UNIDADES_PEDIDAS	NUMERICO	4	NO ADMITE

- La clave primaria está formada por las columnas NIF, ARTICULO, COD_FABRICANTE, PESO, CATEGORIA Y FECHA_PEDIDO.
- COD_FABRICANTE es clave ajena que referencia a la tabla FABRICANTES.
- Las columnas ARTICULO, COD_FABRICANTE, PESO y CATEGORIA son clave ajena y referencian a la tabla ARTICULOS. Realizar un borrado en cascada.
- NIF es clave ajena y referencia a la tabla TIENDAS.
- UNIDADES_PEDIDAS ha de ser >0.
- CATEGORIA ha de ser 'primera', 'segunda' o 'tercera'.

Tabla VENTAS

Campo	tipo	long	NULOS?
NIF	ALFANUM	10	NO ADMITE
ARTICULO	ALFANUM	20	NO ADMITE
COD_FABRICANTE	NUMERICO	3	NO ADMITE
PESO	NUMERICO	3	NO ADMITE
CATEGORIA	ALFANUM	10	NO ADMITE
FECHA_VENTA	FECHA		NO ADMITE
UNIDADES_VENDIDAS	NUMERICO	4	

- La clave primaria está formada por las columnas NIF, ARTICULO, COD_FABRICANTE, PESO, CATEGORIA, y FECHA_VENTA.
- COD_FABRICANTE es clave ajena que referencia a la tabla FABRICANTES.
- Las columnas ARTICULO, COD_FABRICANTE, PESO Y CATEGORIA son clave ajena y referencian a la tabla ARTICULOS. Realizar un borrado en cascada.
- NIF es clave ajena y referencia a la tabla TIENDAS.
- UNIDADES_VENDIDAS ha de ser >0.
- CATEGORIA ha de ser 'primera', 'segunda' o 'tercera'.

2º. Añadir una restricción a la tabla TIENDAS para que el NOMBRE de la tienda sea del tipo titulo.

3º. Visualizar las constraints definidas para las tablas anteriores.

4º. Modificar las columnas de las tablas PEDIDOS y VENTAS para que las UNIDADES_VENDIDAS y las UNIDADES_PEDIDAS puedan almacenar cantidades numéricas de 6 dígitos.

5º. Impedir que se den de alta mas tiendas en la provincia de 'TOLEDO'.

6º. Añadir a las tablas PEDIDOS y VENTAS una nueva columna para que almacenen el PVP del artículo.

7º. Añadir el siguiente comentario a la tabla VENTAS: Unidades vendidas de los distintos artículos de los fabricantes.

8º. Añadir el siguiente comentario al campo PESO de la tabla PEDIDOS 'peso del artículo pedido'.

9º. Crear un sinónimo llamado PED para la tabla PEDIDOS

10º. Cambiar de nombre a la tabla VENTAS y llamarla NUEVAS_VENTAS.

11º. Borrar la tabla Tiendas.

12º. Cree el siguiente modelo de datos:

SOCIOS

Cod_socio	nombre	apellidos	domicilio	poblacion	Cod_postal	provincia	telefono	sexo	observaciones
10300106	Santiago	Blanco Ruiz	Balmes 36	Llodio	01400	Álava	4531221	H	Todos los meses compra dos pelis
11369214	Carlos	Flores Vargas	Potosí 13	Alcorcon	28921	Madrid	5436785	H	
17269813	Ana	López Pintado	Mayor 43	Ulldecona	43550	Tarragona	897654	M	Devuelve las pelis sin rebobinar

TEMAS

Cod_tema	tema
1	Suspense
2	terror
3	Histórica
4	Drama
5	Ciencia-ficción

PELICULAS

Cod_pelicula	titulo	Interprete_1	Interprete_2
1	El color púrpura	Danny Glover	Whoopi Goldberg
2	A Propósito de Henry	Harrison Ford	Anette Bening
3	J.F.K.	Kevin Costner	Sissy Spacek

Director	productora	Cod_tema	Año_produccion	duracion	Precio_coste
Steven Spielberg	Warner Bros	5	1985	148	18
Mike Nichols	Columbia Pictures	4	1991	108	15
Oliver Stone	Warner Bros	3	1991	189	20

ALQUILER

Cod_alquiler	Cod_socio	Cod_pelicula	Fecha_alquiler	Fecha_devolucion	devuelta	precio
1	10300106	1	12/10/1998	18/10/1998	si	5
2	11369214	2	09/09/2000	15/09/2000	no	6
3	17269813	3	23/04/2002	01/05/2002	si	6
4	10300106	2	16/07/2004	25/07/2004	no	6
5	11369214	3	15/03/2006	18/03/2006	si	7
6	17269813	1	08/01/2007	15/01/2007	si	7

Añadiendo las Constraint que no sean de primary key, foreign key o no admitir nulos después de crear las tablas.

13º. Modificar las columnas de las tablas TEMAS y PELICULAS para que las Cod_tema puedan almacenar cantidades numéricas de 4 dígitos.

14º. Nos mosqueamos con el proveedor de películas de la productora Warner Bros. Impedir que se den de alta más películas de dicha productora en la tabla películas.

15º. Añadir el siguiente comentario a la tabla PELICULAS: “Películas producidas a partir de la segunda mitad del siglo XX”.

16º. Añadir el siguiente comentario al campo interprete_1 de la tabla PELICULAS: 'actor principal'.

17º. Crear un sinónimo llamado ALQ para la tabla ALQUILER.

18º. Cambiar de nombre a la tabla SOCIOS y llamarla SOCIOS_NUEVOS.

19º. Borrar la tabla PELICULAS.