Lenguaje SQL

SQL es un lenguaje estructurado de consultas de bases de datos. Es un lenguaje normalizado ya que es un estándar de las bases de datos relacionales

En este curso utilizaremos SQL en Postgres.

Comandos y Cláusulas

Existen dos tipos de **Comandos SQL**:

DDL (**Data Definition Language**) que permiten crear y definir nuevas bases de datos, campos e índices.

Comandos DDL	
Comandos	**Descripción**
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Almacena texto en formato binario.Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

DML (**Data Manipulation Language**), su función es la **manipulación** de datos. A través de él podemos seleccionar, insertar, eliminar y actualizar datos, también generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Comandos DML	
Comandos	**Descripción**
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

Cláusulas

Las cláusulas son **condiciones de modificación** utilizadas para definir los datos que desea seleccionar o manipular.

Cláusulas	
Cláusula	**Descripción**
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico. AulaFacil.com

Comando CREATE

Una tabla es una combinación de filas y columnas. Para crear una tabla, tenemos que definir la estructura de una tabla agregando nombres a las columnas y proporcionando el tipo de datos y el tamaño de los datos que se almacenarán en las columnas.

Ejemplo, creamos una tabla para almacenar datos de Clientes, de modo que el nombre de la tabla sea Cliente, las columnas sean ID, Nombre, País, edad, teléfono, etc.

```
CREATE TABLE Cliente(
    ID INT PRIMARY KEY,
    Nombre VARCHAR(50),
    Apellido VARCHAR(50),
    País VARCHAR(50),
    Edad INT(2),
  Teléfono INT(10)
);
```

También podemos usar CREATE TABLE para crear una copia de una tabla existente. En la nueva tabla, obtiene la definición exacta de la columna, se pueden seleccionar todas las columnas o columnas específicas.

```
CREATE TABLE CopiaTabla AS
SELECT ID, Nombre FROM Cliente;
```

Restricciones

Hay 6 restricciones principales que se usan más frecuentemente en SQL, son:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

Veamos cada una de estas restricciones con más detalle.

NOT NULL: Por defecto, las columnas pueden contener valores NULL. Se usa una restricción NOT NULL en SQL para evitar insertar valores NULL en la columna especificada, considerándolo entonces como un valor no aceptado para esa columna.

```
CREATE TABLE Cliente(
       ID INT NOT NULL,
   Nombre VARCHAR(50)
);
```

UNIQUE: La restricción UNIQUE en SQL se utiliza para garantizar que no se inserten valores duplicados en una columna específica o combinación de columnas que participen en la restricción UNIQUE y no formen parte de la CLAVE PRIMARIA. En otras palabras, el índice que se crea automáticamente cuando define una restricción ÚNICA garantizará que no haya dos filas en esa tabla que puedan tener el mismo valor para las columnas que participan en ese índice, con la capacidad de insertar solo un valor NULL único en estos columnas, esto si la columna permite NULL.

```
CREATE TABLE Cliente(
       ID INT UNIQUE,
   Nombre VARCHAR(50)
);
```

PRIMARY KEY: La restricción PRIMARY KEY consta de una columna o varias columnas con valores que identifican de forma única cada fila de la tabla.

Combina entre las restricciones UNIQUE y SQL NOT NULL, donde la columna o el conjunto de columnas que participan en PRIMARY KEY no pueden aceptar el valor NULL. Si la CLAVE PRIMARIA se define en varias columnas, entonces se puede insertar valores duplicados en cada columna individualmente, pero es importante mencionar que los valores de combinación de todas las columnas de CLAVE PRIMARIA deben ser únicos. Tened en cuenta que solo puede definir una CLAVE PRIMARIA por cada tabla, y se recomienda utilizar columnas pequeñas o INT en la CLAVE PRIMARIA.

```
CREATE TABLE Cliente (
       ID INT PRIMARY KEY,
   Nombre VARCHAR(50)
);
```

FOREIGN KEY: Una clave externa es una clave de base de datos que se utiliza para vincular dos tablas. La restricción FOREIGN KEY identifica las relaciones entre las tablas de la base de datos

haciendo referencia a una columna, o conjunto de columnas, en la tabla secundaria que contiene la clave externa, a la columna PRIMARY KEY o conjunto de columnas, en la tabla principal.

FOREIGN KEY permite insertar valores NULL si no hay una restricción NOT NULL definida en esta clave, pero PRIMARY KEY no acepta NULL.

La restricción FOREIGN KEY también le brinda la capacidad de controlar qué acción se tomará cuando el valor al que se hace referencia en la tabla principal se actualice o elimine, utilizando las cláusulas ON UPDATE y ON DELETE. Las acciones admitidas que se pueden realizar al eliminar o actualizar los valores de la tabla principal incluyen:

- **RESTRICT:** Cuando las cláusulas ON UPDATE o ON DELETE se establecen en RESTRICT, la operación de actualización o eliminación realizada en la tabla principal fallará con un error.
- **CASCADE** : Al configurar las cláusulas ON UPDATE o ON DELETE en CASCADE, la misma acción realizada en los valores referenciados de la tabla principal se reflejará en los valores relacionados en la tabla secundaria. Por ejemplo, si el valor al que se hace referencia se elimina en la tabla principal, también se eliminan todas las filas relacionadas en la tabla secundaria.
- **SET NULL** : con esta opción de cláusulas ON UPDATE y ON DELETE, si los valores a los que se hace referencia en la tabla principal se eliminan o modifican, todos los valores relacionados en la tabla secundaria se establecen en valor NULL.
- **SET DEFAULT** : el uso de la opción SET DEFAULT de las cláusulas ON UPDATE y ON DELETE especifica que, si los valores a los que se hace referencia en la tabla principal se actualizan o eliminan, los valores relacionados en la tabla secundaria con columnas FOREIGN KEY se establecerán en su valor predeterminado.

```
CREATE TABLE Proveedor
(
       CIF INT PRIMARY KEY,
 Nonmbre VARCHAR(50) NULL
);

CREATE TABLE Articulos
(
      Codigo INT PRIMARY KEY,
       Nombre VARCHAR (50),
       Precio DECIMAL (10),
       Proveedor INT FOREIGN KEY REFERENCES Proveedor(CIF)
);
```

CHECK: Una restricción CHECK se define en una columna o conjunto de columnas para limitar el rango de valores que se pueden insertar en estas columnas, utilizando una condición predefinida. La restricción CHECK entra en acción para evaluar los valores insertados o modificados, donde el valor que satisfaga la condición será insertado en la tabla, de lo contrario la operación de inserción será descartada. Se permite especificar múltiples restricciones CHECK para la misma columna.

Definir la condición de restricción CHECK es de alguna manera similar a escribir la cláusula WHERE de una consulta, utilizando los diferentes operadores de comparación, como AND, OR, BETWEEN, IN, LIKE e IS NULL para escribir su expresión booleana que devolverá VERDADERO,

FALSO o DESCONOCIDO. La restricción CHECK devolverá un valor DESCONOCIDO cuando esté presente un valor NULL en la condición. La restricción CHECK se usa para hacer cumplir la integridad del dominio limitando los valores insertados a los que siguen los valores definidos, el rango o las reglas de formato.

```
CREATE TABLE Empleado
(
     ID INT PRIMARY KEY,
    Nombre VARCHAR(50) NULL,
    Salario INT CHECK (Salario>0)
);
```

DEFAULT: Se utiliza una restricción DEFAULT para proporcionar un valor de columna predeterminado para las filas insertadas si no se especifica ningún valor para esa columna en la instrucción INSERT. La restricción Predeterminada ayuda a mantener la integridad del dominio al proporcionar valores adecuados para la columna, si el usuario no da valor. El valor predeterminado puede ser un valor constante, un valor de función del sistema o NULL.

```
CREATE TABLE Empleado
(
     ID INT PRIMARY KEY,
    Nombre VARCHAR(50) NULL,
    Salario INT DEFAULT 1200
);
```

Cláusulas CONSTRAINT

La cláusula CONSTRAINT se usa en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar restricciones. Hay dos tipos de cláusulas CONSTRAINT: uno para crear una restricción en un único campo y otro para crear una restricción en varios campos.

Sintaxis

Restricción de un único campo:

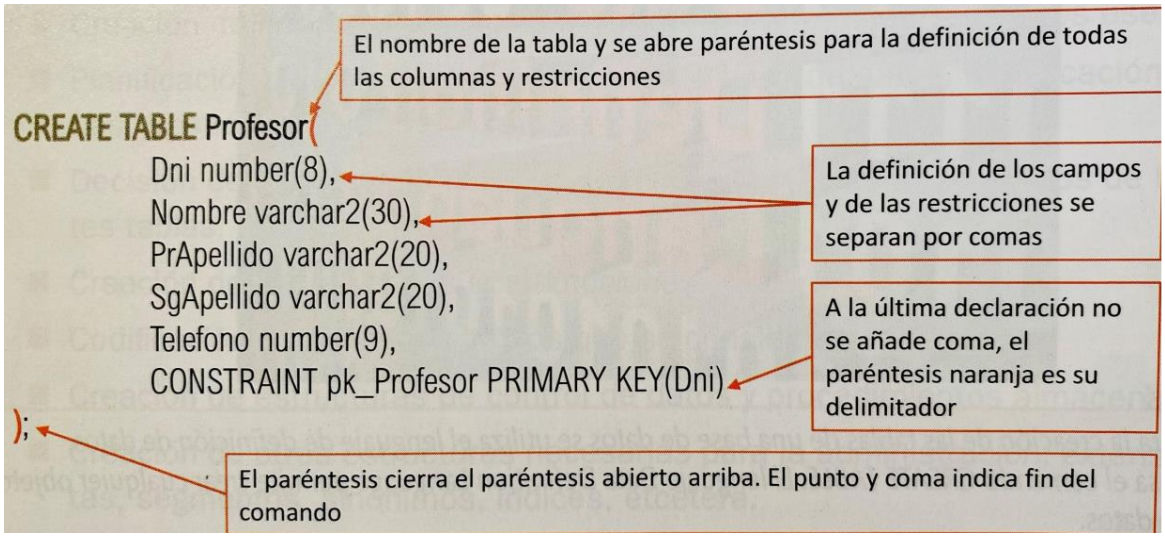
Nombre DE RESTRICCIÓN {PRIMARY KEY | UNIQUE | NOT NULL | REFERENCES foreigntable [(foreignfield1, foreignfield2)] [ON UPDATE CASCADE | SET NULL] [ON DELETE CASCADE | SET NULL]}

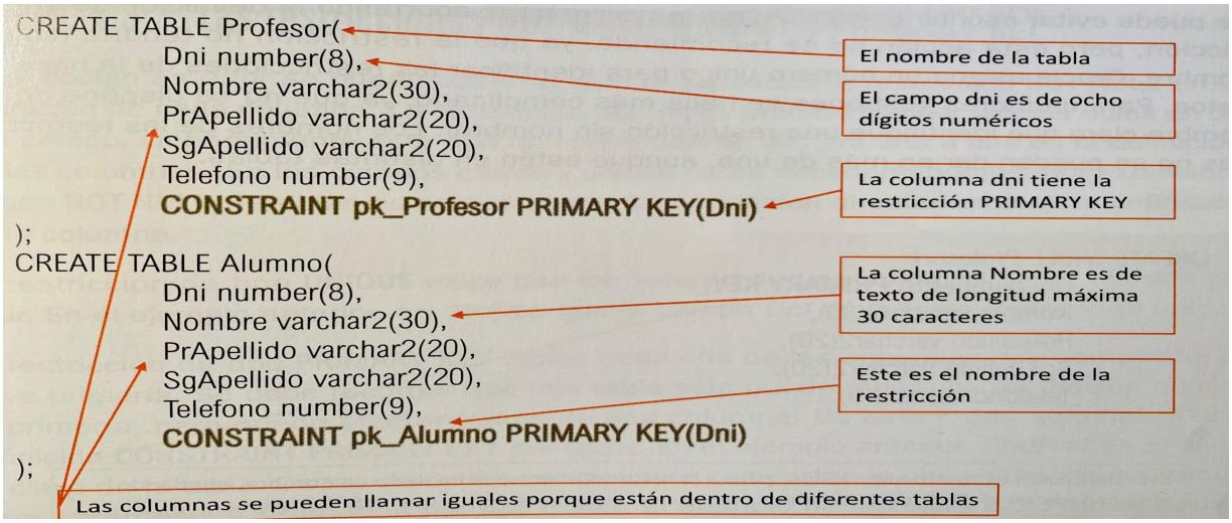
Restricción de varios campos:

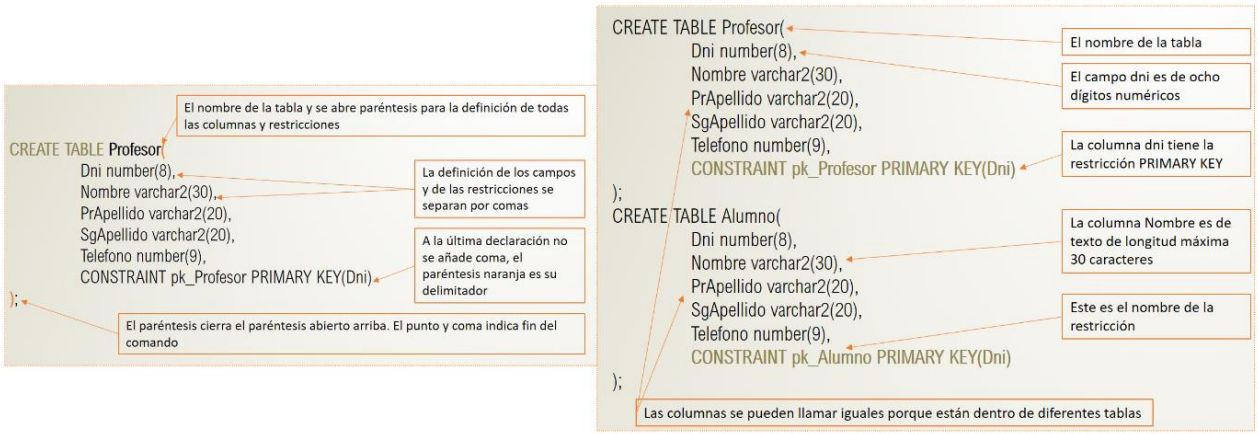
Nombre DE RESTRICCIÓN {PRIMARY KEY (primary1[, primary2 [, ...]]) | UNIQUE (unique1[, unique2 [, ...]]) | NOT NULL (notnull1[, notnull2 [, ...]]) | FOREIGN KEY [NO INDEX] (ref1[, ref2 [, ...]]) REFERENCES foreigntable [(foreignfield1 [, foreignfield2 [, ...]])] [ON UPDATE CASCADE | SET NULL] [ON DELETE CASCADE | SET NULL]}

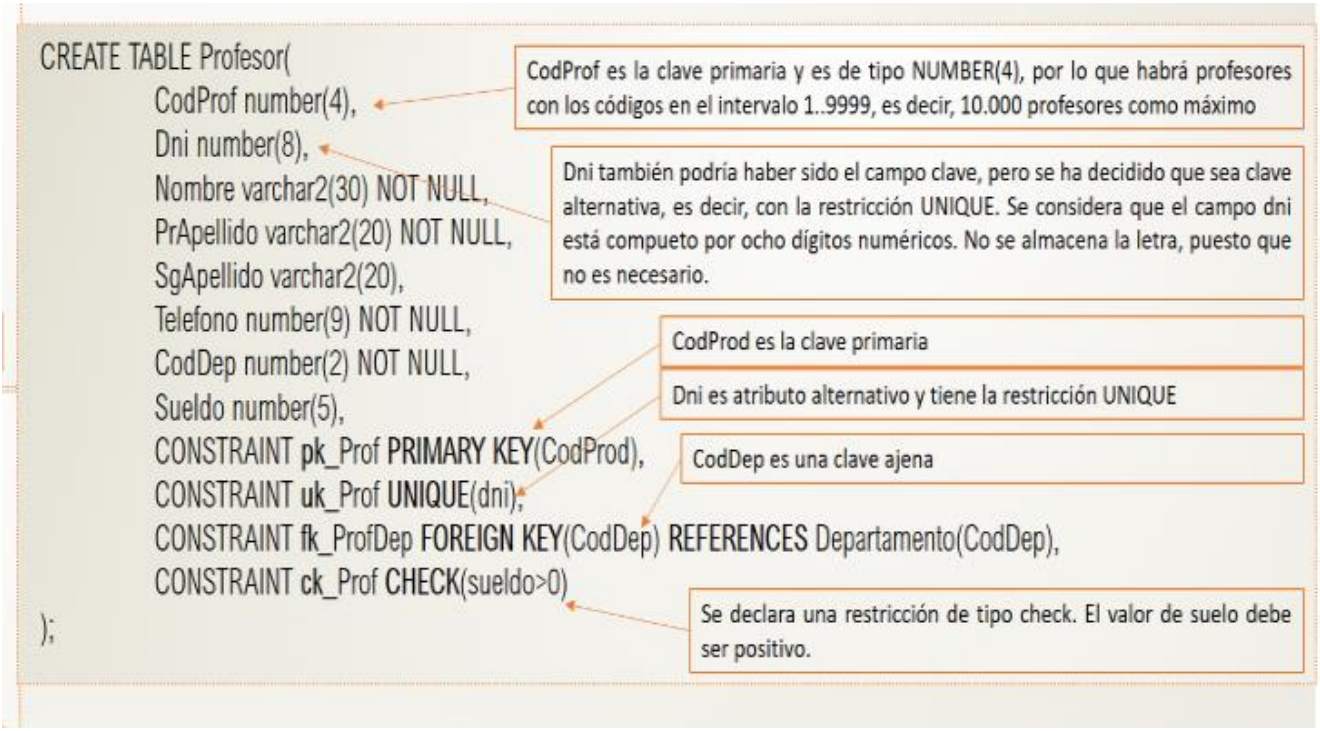
Restricciones más habituales:

- NOT NULL – Obliga a rellenar el campo.
- UNIQUE – Asegura que todos los valores son diferentes.
- PRIMARY KEY – Clave principal. De hecho, una combinación de NOT NULL + UNIQUE. Identifica unívocamente cada fila en una table.
- FOREIGN KEY – Clave foránea. Relaciona unívocamente al registro de otra tabla.
- CHECK – Establece reglas de validación para los datos.
- DEFAULT – Valor por defecto.
- INDEX – Acelera las búsquedas bajo este campo.






El nombre de la tabla y se abre paréntesis para la definición de todas las columnas y restricciones
CREATE TABLE Profesor(
Dni number(8),
Nombre varchar2(30),
PrApellido varchar2(20),
SgApellido varchar2(20),
Telefono number(9),
CONSTRAINT pk_Profesor PRIMARY KEY(Dni)
);
La definición de los campos y de las restricciones se separan por comas
A la última declaración no se añade coma, el paréntesis naranja es su delimitador
El paréntesis cierra el paréntesis abierto arriba. El punto y coma indica fin del comando
CREATE TABLE Profesor(
Dni number(8),
Nombre varchar2(30),
PrApellido varchar2(20),
SgApellido varchar2(20),
Telefono number(9),
CONSTRAINT pk_Profesor PRIMARY KEY(Dni)
);
CREATE TABLE Alumno(
Dni number(8),
Nombre varchar2(30),
PrApellido varchar2(20),
SgApellido varchar2(20),
Telefono number(9),
CONSTRAINT pk_Alumno PRIMARY KEY(Dni)
);
El nombre de la tabla
El campo dni es de ocho dígitos numéricos
La columna dni tiene la restricción PRIMARY KEY
La columna Nombre es de texto de longitud máxima 30 caracteres
Este es el nombre de la restricción
Las columnas se pueden llamar iguales porque están dentro de diferentes tablas


CREATE TABLE Profesor(
CodProf number(4),
Dni number(8),
Nombre varchar2(30) NOT NULL,
PrApellido varchar2(20) NOT NULL,
SgApellido varchar2(20),
Telefono number(9) NOT NULL,
CodDep number(2) NOT NULL,
Sueldo number(5),
CONSTRAINT pk_Prof PRIMARY KEY(CodProd),
CONSTRAINT uk_Prof UNIQUE(dni),
CONSTRAINT fk_ProfDep FOREIGN KEY(CodDep) REFERENCES Departamento(CodDep),
CONSTRAINT ck_Prof CHECK(sueldo>0)
);
CodProf es la clave primaria y es de tipo NUMBER(4), por lo que habrá profesores con los códigos en el intervalo 1..9999, es decir, 10.000 profesores como máximo
Dni también podría haber sido el campo clave, pero se ha decidido que sea clave alternativa, es decir, con la restricción UNIQUE. Se considera que el campo dni está compueto por ocho dígitos numéricos. No se almacena la letra, puesto que no es necesario.
CodProd es la clave primaria
Dni es atributo alternativo y tiene la restricción UNIQUE
CodDep es una clave ajena
Se declara una restricción de tipo check. El valor de suelo debe ser positivo.

```
CREATE TABLE Ejemplar(
        CodLibro number(4),
        CodEjemplar number(3),
        Estado char(1),
        CONSTRAINT pk_ejem PRIMARY KEY(CodLibro, CodEjemplar)
);
```

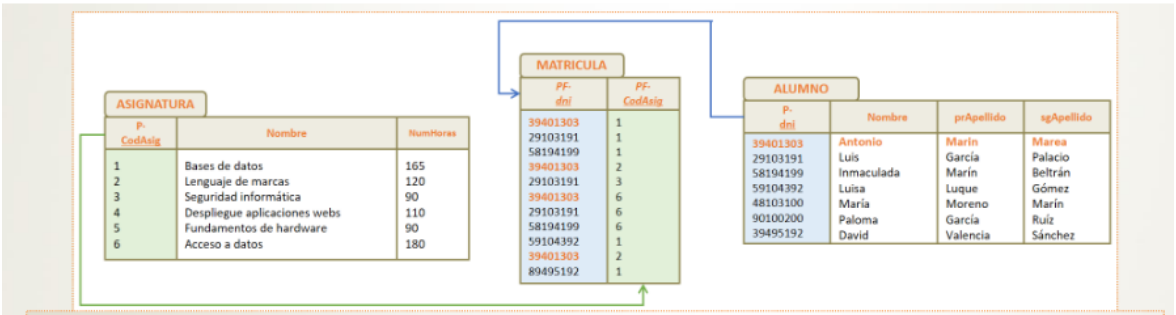
El campo clave de Ejemplar está compuesto por dos columnas, CodLibro y CodEjemplar. El número de ejemplares que se puede almacenar es 9.999.999

```
CREATE TABLE Libro(
        CodLibro number(4),
        numPags number(4),
        CONSTRAINT pk_libro PRIMARY KEY(CodLibro));
CREATE TABLE Ejemplar(
        CodLibro number(4),
        CodEjemplar number(3),
        Estado char(1),
        CONSTRAINT pk_ejem PRIMARY KEY(CodLibro, CodEjemplar));
CREATE TABLE Venta(
        CodVenta number(6),
        Fecha date,
        CONSTRAINT pk_venta PRIMARY KEY(CodVenta));
CREATE TABLE LineaVenta(
        CodVenta number(6),
        CodLibro number(4),
        CodEjemplar number(3),
        Cantidad number(2) DEFAULT 1,
        CONSTRAINT pk_lv PRIMARY KEY(CodVenta, CodLibro, CodEjemplar),
        CONSTRAINT fk_lv_venta FOREIGN KEY(CodVenta) REFERENCES Venta(CodVenta),
        CONSTRAINT fk_lv_ejemplar FOREIGN KEY(CodLibro,CodEjemplar) REFERENCES Ejemplar(CodLibro,CodEjemplar));
```

La tabla Ejemplar es una tabla débil por identificación de Libro. Hereda CodLibro como clave ajena y clave primaria

Los dos campos comportan la clave primaria

La tabla LineaVenta proviene de una relación N:M, que pretende almacenar en cada venta los ejemplares vendidos. El código clave de Venta es CodVenta. El código clave de ejemplar está compuesto por dos columnas CodLibro y CodEjemplar.



ASIGNATURA

P- CodAsig	Nombre	NumHoras
1	Bases de datos	165
2	Lenguaje de marcas	120
3	Seguridad informática	90
4	Despliegue aplicaciones webs	110
5	Fundamentos de hardware	90
6	Acceso a datos	180

MATRICULA

PF- dni	PF- CodAsig
39401303	1
29103191	1
58194199	1
39401303	2
29103191	3
39401303	6
29103191	6
58194199	6
59104392	1
39401303	2
89495192	1

ALUMNO

P- dni	Nombre	prApellido	sgApellido
39401303	Antonio	Marín	Marea
29103191	Luis	García	Palacio
58194199	Inmaculada	Marín	Beltrán
59104392	Luisa	Luque	Gómez
48103100	María	Moreno	Marín
90100200	Paloma	García	Ruiz
39495192	David	Valencia	Sánchez

```
CREATE TABLE Asignatura(
        CodAsig NUMBER(4),
        Nombre VARCHAR2(30) NOT NULL,
        numHoras NUMBER(3),
        CONSTRAINT pk_asig PRIMARY KEY(CodAsig));
CREATE TABLE Alumno(
        dni NUMBER(8),
        Nombre VARCHAR2(30) NOT NULL,
        prApellido VARCHAR2(30) NOT NULL,
        sgApellido VARCHAR2(30),
        CONSTRAINT pk_alum PRIMARY KEY(dni));
CREATE TABLE Matricula(
        dni NUMBER(8),
        CodAsig NUMBER4),
        CONSTRAINT pk_matri PRIMARY KEY(dni, CodAsig)
        CONSTRAINT fk_matr_alum FOREIGN KEY(dni) REFERENCES Alumno(dni) ON DELETE CASCADE,
        CONSTRAINT fk_matr_asig FOREIGN KEY(CodAsig) REFERENCES Asignatura(CodAsig) ON DELETE CASCADE);
```

La cláusula ON DELETE CASCADE y/o ON DELETE SET NULL se usa en la definición de la clave ajena

Restricciones de integridad referencial.

Comando DROP

La sentencia DROP nos permite borra una o más tablas de una base datos. Por ejemplo:

```
DROP TABLE Clientes;

DROP TABLE Asignaturas, Profesores;
```

Solo el propietario de la tabla, del esquema o el superusuario pueden eliminar una tabla. Para borrar filas sin borrar la tabla se utiliza DELETE como veremos más adelante con los DML.

Si la tabla contiene dependencias referenciales con otra tabla habrá que especificar con CASCADE si queremos eliminarlas.

Puede ser que vayamos a borrar una tabla que por el motivo que sea no exista en nuestra base de datos, esto nos provocaría un error. Para evitar este error podemos ejecutar la sentencia:

```
DROP TABLE IF EXISTS nombretabla;
```

INSERCIÓN DE DATOS (INSERT TO)

Una vez creada una tabla podemos introducir datos en ella mediante la sentencia INSERT, como se muestra a continuación:

```
INSERT INTO Clientes (codigo, nombre, apellido, edad, telefono) VALUES (12, 'Manolo', 'Segarra', 32, 666666777);

INSERT INTO Clientes (codigo, nombre, apellido, edad, telefono) VALUES (13, 'Tomas', 'Navarro', 41, 666666888);
```

Las cadenas de caracteres y las fechas se introducen entre comillas simples. Para introducir un valor nulo se utiliza la expresión NULL.

Algunos SGBD permiten la inserción de varias filas en una misma tabla mediante una sola sentencia INSERT, y se realizan inserciones de un modo más eficiente que si se hace mediante varias sentencias independientes.

```
INSERT INTO Clientes (codigo, nombre, apellido, edad, telefono)
 VALUES (12, 'Manolo', 'Segarra', 32, 666666777),
        (13, 'Tomas', 'Navarro', 41, 666666888);
```

Comando ALTER TABLE

La sentencia ALTER TABLE nos permite añadir, borrar o modificar columnas existentes en una tabla. También podemos añadir o eliminar restricciones (constraints) que existan sobre la tabla.

Ejemplos:

```
ALTER TABLE Alumnos ADD email VARCHAR (50);

ALTER TABLE Clientes DROP COLUMN edad;

ALTER TABLE Asignaturas RENAME COLUMN id_asig TO codigo;

ALTER TABLE Medicos ALTER COLUMN num_colegiado VARCHAR(15);

ALTER TABLE Empleados MODIFY cargo VARCHAR (15);  // Hay versiones que aceptan: ALTER TABLE Empleados MODIFY COLUMN cargo VARCHAR (15);
```

Ejemplo de creación, inserción y modificación de una tabla:

```
CREATE TABLE Clientes ( ID INT(32) PRIMARY KEY, Nombre VARCHAR (25));

INSERT INTO Clientes (ID, Nombre) VALUES (1, 'Marina);

ALTER TABLE Clientes ADD Telefono NUMERIC;

INSERT INTO Clientes (ID, Nombre, Telefono) VALUES (2, 'Carolina', 666777888);
```