Bases de Datos en SQL

El objetivo del presente texto es permitir al lector introducirse y profundizar de forma práctica y aplicada en el lenguaje SQL (Structured Query Language).

SQL (Structured Query Language) es un lenguaje de programación diseñado para administrar datos en una base de datos relacional. Tiene una variedad de funciones que permiten a sus usuarios leer, manipular y cambiar datos.

Recordemos que:

Una **base de datos** es un conjunto de datos relacionados entre sí. Un **sistema de gestión de base de datos (SGBD)** es un conjunto de programas que permiten almacenar y procesar la información contenida en una base de datos.

Una **base de datos relacional** es aquélla en la cual toda la información se almacena en tablas.
Cada base de datos admite estas cuatro operaciones básicas:

- Añadir información a una tabla
- Actualizar la información que ya existe en una tabla
- Eliminar información que ya existe en una tabla
- Ver la información contenida en una table

Una **tabla** está formada por **filas y columnas**. Cada tabla tiene un nombre único y un conjunto de filas y columnas.
Cada fila contiene información sobre una única entidad.
Cada columna contiene información sobre una única propiedad o atributo de las entidades.
Una celda es una intersección de una fila y de una columna. Cada celda puede contener bien un valor o bien ningún valor. Cuando no contiene ningún valor, se dice que tiene el valor nulo.

Existen dos tipos de elementos en la creación de una tabla, los campos y las restricciones, aunque sólo los primeros son obligatorios.

Las restricciones (Primary Key, Unique, Foreign Key) son los elementos del modelo relacional que permiten que éste funcione, ya que sin ellas una base de datos no es más que un mero almacén de información pero sin ningún tipo de conexión entre los datos. La definición de éstas presenta una misma estructura:

constraint nombre tipo parámetros

De las restricciones mostradas, las dos primeras únicamente indican los campos que conforman la clave correspondiente y la tercera especifica qué atributos definen una clave ajena a otra tabla y las características necesarias para su correcto funcionamiento.

La **clave primaria** es una columna o conjunto de columnas que identifican de forma única a cada una de las entidades (filas) que componen las tablas.

Una clave principal es una información de identificación exclusiva que le permite buscar un registro determinado dentro de una tabla, en la misma tabla no puede haber dos registros con el mismo valor en el campo de la clave principal, la clave principal podría estar compuesta por un solo campo o por varios PRIMARY KEY.

Al crear una tabla puede crear una sola restricción PRIMARY KEY como parte de la definición de tabla. Si la tabla ya existe, puede agregar una restricción PRIMARY KEY, siempre que no exista ya otra restricción PRIMARY KEY. Una tabla puede contener una sola restricción PRIMARY KEY.

Reglas en las claves principales:

1. Las columnas no admiten valores NULL
2. No puede haber valores duplicados. Si se agrega una restricción PRIMARY KEY a una columna que tiene valores duplicados o permite valores NULL, el Motor de base de datos devuelve un error y no agrega la restricción.

La **clave ajena o foránea** es una columna o conjunto de columnas cuyos valores coinciden con algunos valores de una clave primaria de una tabla.

Aspectos adicionales de la Foreign Key:

- La tabla referida debe existir para poder ser incluida en la definición de la clave foránea, por lo que la definición de las tablas debe seguir un orden. Así, la tabla de ciudades no puede ser creada antes que la tabla de provincias.
- Una clave ajena o foránea debe tener el mismo número de atributos que la clave primaria de la tabla a la que hace referencia, y además deben corresponder en tipo y dimensión.
- Si la clave ajena es compuesta se recomienda especificar las columnas de la tabla actual y de la tabla referida, para asegurar que la correspondencia entre campos sea la adecuada.

Reglas que permiten mantener la integridad de la información de las bases de datos:

La **Regla de Integridad de Entidades** especifica que ninguna de las columnas que componen la clave primaria de una tabla puede contener valores nulos.

La **Regla de Integridad Referencial** especifica que las claves ajenas o bien tienen valores nulos o bien contienen valores tales que coinciden con algún valor de la clave primaria a la que referencian.

Es deseable que la información contenida en las bases de datos cumpla ambas reglas. En caso contrario, la información se encuentra en un estado inconsistente que a la larga sólo puede producir errores. Pensemos, por ejemplo, en facturas para clientes que no existen, en ventas de artículos que no existen, etc.

Para crear las llaves foráneas usamos el comando CONSTRAINT, indicamos el nombre de la llave foránea que tiene que estar relacionada, luego el comando FOREING KEY y entre () la columna que será llave foránea.

Con el comando REFERENCE le indicamos a que tabla hace referencia y entre () le agregamos la columna que es la llave primaria

El lenguaje SQL consta de dos partes claramente diferenciadas:

•**Lenguaje de Definición de Datos** (en inglés Data Definition Language **o DDL**): Incluye aquellas sentencias que sirven para definir los datos o para modificar su definición, como por ejemplo la creación de tablas, índices, etc.
Sentencias de definición de datos:
Create
Drop
Alter

•**Lenguaje de Manipulación de Datos** (en inglés Data Manipulation Language **o DML**): Incluye aquellas sentencias que sirven para manipular o procesar los datos, como por ejemplo la inserción, borrado, modificación o actualización de datos en las tablas.

Sentencias de manipulación de datos:
select: Permite extraer información almacenada en la base de datos. Es una operación de sólo lectura.
insert: Permite insertar información en la base de datos.
update: Permite modificar información almacenada en la base de datos.
delete: Permite borrar información existente en la base de datos.

La cláusula **delete** indica qué ocurrirá cuando se intente borrar en la tabla referida una fila cuya clave primaria aparece como valor de clave ajena en alguna fila de la tabla actual. Definiéndose cinco opciones,

- **no action**, es decir, no hacer nada en la tabla actual, pero borrar la fila en la tabla referida, lo que puede provocar problemas de falta de integridad de los datos.
- **set null,** se asigna el valor null en los campos que forman la clave ajena de aquellas filas que tengan como clave ajena el valor de la clave primaria que se desea borrar en la tabla referida, y posteriormente se borra esta fila.
- **set default**, se asigna el valor por defecto en los campos que forman la clave ajena de aquellas filas que tengan como clave ajena el valor de la clave primaria que se desea borrar en la tabla referida, y posteriormente se borra esta fila.

- **cascade**, se borran las filas que tienen como clave ajena el valor de la clave primaria que se desea borrar en la tabla referida antes de borrar éstas.
- **restrict,** si existe alguna fila que tiene como clave ajena el valor de la clave primaria que se desea borrar en la tabla referida, la fila asociada no se borra.

Uso de comentarios:
Podemos usar --(dos guiones) para comentar todo a la derecha de ellos en una línea determinada. También podemos dejar comentarios en varias líneas usando /*para comenzar el comentario y */cerrarlo

Aplicaremos éstos conceptos para la creación de bade de datos, de tablas; ingreso de datos, borrado de tablas, datos; actualización de tablas y datos para generar un control de stocks y facturación de una determinada empresa. Luego aplicaremos las distintas sentencias para generar las consultas.

1 Base de Datos: Facturación

Comencemos presentando y describiendo las distintas tablas que componen nuestra base de datos.

Para cada tabla se presenta su nombre y las columnas de que consta entre paréntesis. Las claves primarias aparecen subrayadas. Las claves ajenas o foráneas están en cursiva. También se muestra diversa información sobre las columnas de las tablas: si aceptan nulos y su tipo de datos (o dominio).

Tabla **provincias(codpro, nombre)**: Esta tabla almacena las provincias de Argentina, cada una con su código de provincia (clave primaria) y su nombre.

Tabla Provincias

Columna	¿Nulo?	Tipo de Datos
codpro	not null	VARCHAR2(2)
nombre	not null	VARCHAR2(30)

Tabla **ciudad(codciu, nombre, *codpro*)**: Almacena las ciudades de Argentina o, por lo menos, aquéllos donde tenemos clientes. Para cada ciudad se dispone de su código de ciudad (clave primaria), su nombre y el código de la provincia a la que pertenece (clave ajena).

Tabla Ciudades

Columna	¿Nulo?	Tipo de Datos
codciu	not null	VARCHAR2(5)
nombre	not null	VARCHAR2(40)
codpro	not null	VARCHAR2(2)

Tabla **clientes(codcli, nombre, direccion, codpostal, *codciu*)**: Almacena información sobre los clientes de la empresa. Para cada cliente se dispone de su código de cliente (clave primaria), su nombre, su dirección, su código postal y el código de la ciudad donde reside (clave ajena).

Tabla Clientes

Columna	¿Nulo?	Tipo de Datos
codcli	not null	NUMBER(5)
nombre	not null	VARCHAR2(50)
direccion	not null	VARCHAR2(50)
codpostal		VARCHAR2(5)
codciu	not null	VARCHAR2(5)

Tabla **vendedores(codven, nombre, direccion, codpostal, codciu)**: Almacena información sobre los vendedores de la empresa. Para cada vendedor se dispone de su código de vendedor (clave primaria), su nombre, su dirección, su código postal, el código de la ciudad donde reside (clave ajena a la tabla ciudades).

Tabla Vendedores

Columna	¿Nulo?	Tipo de Datos
codven	not null	NUMBER(5)
nombre	not null	VARCHAR2(50)
direccion	not null	VARCHAR2(50)
codpostal		VARCHAR2(6)
codciu	not null	VARCHAR2(5)

codjefe	not null	NUMBER(5)

Tabla **articulos(codart, descrip, precio, stock, stock_min)**: Almacena información sobre los artículos que ofrece la empresa y sus cantidades disponibles en el almacén (stocks). Para cada artículo se dispone de su código de artículo específico (clave primaria), su descripción, su precio actual, su stock y su stock mínimo, es decir, el valor umbral por debajo del cual se debe reponer.

Tabla Articulos

Columna	¿Nulo?	Tipo de Datos
codart	not null	VARCHAR2(8)
descrip	not null	VARCHAR2(40)
precio	not null	NUMBER(7,2)
stock		NUMBER(6)
stock_min		NUMBER(6)

Tabla **facturas(codfac, fecha, *codcli*, *codven*, iva, dto)**: Almacena toda la información sobre las facturas, excepto sus líneas. Como en cada factura el número de líneas es variable, todas las líneas de todas las facturas se almacenan juntas en otra tabla. Para cada factura en esta tabla se guarda su código de factura (clave primaria), su fecha, el código del cliente que ha realizado la compra (clave ajena), el código del vendedor que ha realizado la venta (clave ajena), el iva aplicado y el descuento global de la factura.

Tabla Facturas

Columna	¿Nulo?	Tipo de Datos
codfac	not null	NUMBER(6)
fecha	not null	DATE
codcli		NUMBER(5)
codven		NUMBER(5)
iva		NUMBER(2)
dto		NUMBER(2)

Los nombres de las tablas y de sus columnas se han escrito sin acentuar para evitar problemas con algunos sistemas de gestión de bases de datos.

A partir de ésta información comencemos a crear la base de datos y las tablas.

```
--creamos la base de datos--
create database Facturacion
use Facturacion
```

```
-- creamos la tabla provincias --
create table provincias (
codpro character(2) not null,
nombre character(20) not null default ' ', constraint pk_provincias primary key
(codpro)
);
```

```
-- creamos tabla vendedores--
create table vendedores(
codven numeric (5) primary key not null,
nombre varchar(50) not null,
direccion varchar (50) not null,
codpostal varchar (6),
codciu varchar (5),
)
```

```
--creamos tabla ciudades--
create table ciudades(
codciu varchar (5) primary key not null,
nombre varchar (40) not null,
codpro character (2) not null FOREIGN KEY (codpro) REFERENCES provincias (codpro)
)
```

```
--creamos tabla clientes--
create table clientes(
codcli numeric (5) primary key not null,
nombre varchar (50) not null,
direccion varchar (50) not null,
codpostal varchar (5),
codciu varchar (5) not null Foreign key ( codciu) references ciudades ( codciu)
)
```

```
--creamos la tabla articulos --
create table articulos (
codart character(8) primary key not null,
descript character(40) not null,
precio numeric (7,2) not null default 0.0,
stock  numeric (6),
stock_min numeric (6),
);
```

```
-- creacion tabla facturas--
create table facturas (
codfac numeric (6) not null,
fecha nchar(8)   not null,
codcli numeric (5) foreign key (codcli) references clientes (codcli),
codven numeric(5) foreign key (codven) references vendedores (codven),
iva    numeric (2),
dto    numeric (2),
)
```

Insertar datos

Con la información que se presenta en las siguientes tablas, comencemos a cargar nuestros datos.

Tabla PROVINCIAS

CODPRO	NOMBRE
12	BUENOS AIRES
44	CORDOBA
46	NEUQUEN

```
--Ingreso Datos para la tabla provincias--

INSERT INTO provincias
VALUES (12, 'BUENOS AIRES')

INSERT INTO provincias
VALUES (44, 'CORDOBA')

INSERT INTO provincias
VALUES (46, 'NEUQUEN')
```

Tabla CIUDADES

CODCIU	NOMBRE	CODPRO
101	LA PLATA	12
102	QUILMES	12
103	CORDOBA	44
104	NEUQUEN	46
105	ALUMINE	46

```
--Ingreso Datos para la tabla ciudades --
INSERT INTO ciudades
VALUES (101, 'LA PLATA', 12)

INSERT INTO ciudades
VALUES (102, 'QUILMES', 12)

INSERT INTO ciudades
VALUES (103, 'CORDOBA', 44)

INSERT INTO ciudades
VALUES (104, 'NEUQUEN', 46)

INSERT INTO ciudades
VALUES (105, 'ALUMINE', 46)
```

Tabla CLIENTES

CODCLI	NOMBRE	DIRECCIÓN	CODPOSTAL	CODCIU
435	Alberto	Cuesta, 5	1000	101
567	Carlos	En proyecto, 3	1002	102
789	Pedro	Colón, 4	1001	103

```
--Ingreso Datos para la tabla clientes --
INSERT INTO clientes
VALUES (435, 'ALBERTO','CUESTA,5',1000,101)

INSERT INTO clientes
VALUES (567, 'CARLOS','EN PROYECTO,3',1002, 102)

INSERT INTO clientes
VALUES (789, 'PEDRO','COLON,4',1001, 103)
```

Tabla VENDEDORES

CODVEN	NOMBRE	DIRECCION	CODPOSTAL	CODCIU
21	PEDRO	AV. 5 N° 34	1000	101
22	JUAN	ESMERALDA 267	1005	102
24	MARIA	47 N° 894	1000	101
25	CARLA	LIBERTADOR 87	1001	103
26	ESTEBAN	CHICLANA 66	1005	102
27	FLORENCIA	MORENO1 762	1004	104

```
--Ingreso Datos para la tabla vendedores --
INSERT INTO vendedores
VALUES (21, 'PEDRO','AV. 5 N° 34',1000,101)
INSERT INTO vendedores
VALUES (22, 'JUAN','ESMERALDA 267',1005,102)
INSERT INTO vendedores
VALUES (24, 'MARIA','47 N° 894',1000,101)
INSERT INTO vendedores
VALUES (25,'CARLA','LIBERTADOR 87',1001, 103)
INSERT INTO vendedores
VALUES (26, 'ESTEBAN','CHICLANA 66',1005,102)
INSERT INTO vendedores
VALUES (27, 'FLORENCIA','MORENO 762',1004,104)

select *from vendedores
```

Tabla ARTICULOS

CODART	DESCRPCION	PRECIO	STOCK	STOCK_MIN
A1	HELADERA	90000	15	15
A2	TELEVISOR	52000	21	5
A3	BATIDORA	7800	7	30
A4	CAFETERA	5430	19	50
A5	PLANCHA	2579	8	10

Tabla FACTURAS

CODFAC	FECHA	CODCLI	CODVEN	IVA	DTO
1	14/1/2005	435	22	16	5
2	14/2/2005	435	27	16	0
3	14/1/2006	435	25	7	5
4	14/1/2007	567	22	16	0
5	15/4/2008	789	26	0	0
6	15/4/2008	789	24	7	5

```
--Ingreso Datos para la tabla articulos--
INSERT INTO articulos
VALUES ('A1','HELADERA', 90000,15,15)

INSERT INTO articulos
VALUES ('A2','TELEVISOR', 52000,21, 5)

INSERT INTO articulos
VALUES ('A3','BATIDORAA', 7800,7,30)

INSERT INTO articulos
VALUES ('A5','CAFETERA', 54300,19,50)

INSERT INTO articulos
VALUES ('A6','PLANCHA', 2579,8,10)
```

```
--Ingreso Datos para la tabla facturas--
INSERT INTO facturas
VALUES (1, '14/01/05',435,22,16,5)

INSERT INTO facturas
VALUES (2, '14/02/05',435,27,16,0)

INSERT INTO facturas
VALUES (3, '14/01/06',435,25,7,5)

INSERT INTO facturas
VALUES (4, '14/04/07',567,22,16,0)

INSERT INTO facturas
VALUES (5, '15/04/08',789,26,0,0)

INSERT INTO facturas
VALUES (6, '15/04/08',789,24,7,5)
```

Consultas

Una vez que esta todo cargado ya podemos comenzar a hacer las consultas en nuestra base de datos

Para realizar una consulta se utilizan la cláusula `SELECT` y la cláusula `FROM`.

`SELECT` indica qué columnas le gustaría ver y `FROM` identifica la tabla en la que se encuentran los datos. Habitualmente cada una se escribe en una línea distinta para mejorar la legibilidad, aunque nada impide escribirlas en una única línea.

La cláusula `SELECT` permite especificar qué información se desea obtener. Se pueden poner tantas columnas de la tabla como se desee. Además, pueden ponerse expresiones de una o más columnas de la tabla. El carácter * indica que se muestren todas las columnas
La cláusula `FROM` permite indicar de qué tabla se deben extraer los datos.

Veamos cómo funciona:
Verifiquemos que tos datos de todas las tablas se cargaron correctamente

```
select * FROM provincias;
SELECT * FROM ciudades
SELECT * FROM clientes
select * from vendedores
SELECT * FROM artículos
SELECT * FROM facturas
```

Mostrar el código y nombre de las provincias. Lo podemos hacer de dos formas

```
select * from provincias;
select codpro, nombre from provincias;
```

Mostrar el nombre y después el código de las provincias

```
select nombre, codpro from provincias;
```

La sentencia `select` admite opcionalmente el uso del modificador distinct en su primera cláusula.
Mostrar los distintos tipos de ivas aplicados en las facturas.

```
select distinct iva from facturas;
```

En numerosas ocasiones resulta conveniente restringir o seleccionar sólo algunas filas que cumplen una determinada condición de entre todas las existentes en una tabla. Esta operación se lleva a cabo con la cláusula `WHERE`, la cual es opcional y de aparecer, deber hacerlo siempre tras la cláusula `from`.

La forma más básica de filtrar datos es utilizar operadores de comparación.

Igual a	=
No igual a	<> o !=
Mas grande que	>
Menos que	<
Mayor qué o igual a	>=
Menos que o igual a	<=

Mostrar el código y nombre de aquellas provincias cuyo código es menor que ‘20'.

```
select codpro, nombre from  provincias where codpro < 20 ;
```

Cuando el SGBD ejecuta una sentencia con las tres cláusulas `select, from y where,` examina en primer lugar la cláusula `from` para saber qué tablas debe procesar, a continuación realiza una criba dejando sólo aquellas filas que cumplan la condición o condiciones de la cláusula `where` y, finalmente, examina la cláusula `select` para determinar qué información se desea mostrar a partir de dichas tablas.

Mostrar los distintos tipos de descuentos aplicados por los vendedores cuyos códigos no superan el valor 20

```
select codpro, nombre from  provincias where codpro < 20 ;
```

Mostrar el código y descripción de aquellos artículos cuyo stock iguala o supera las 5 unidades.

```
select codart, descrip from articulos where stock >= 5 ;
```

Mostrar el código de factura y fecha de las facturas con iva 16 y del cliente 435.

```
select codfac, fecha from   facturas where iva = 16
and    codcli = 435 ;
```

Mostrar el código de artículo y el precio de aquellos artículos cuyo precio supera los 4000$ y cuyo stock supera las 8 unidades.

```
select codart, precio from  articulos where precio > 4000 and  stock > 8 ;
```

Los operadores lógicos permiten utilizar varios operadores de comparación en una consulta.

- `LIKE` le permite hacer coincidir valores similares, en lugar de valores exactos.
- `IN` le permite especificar una lista de valores que le gustaría incluir.
- `BETWEEN` le permite seleccionar solo filas dentro de un cierto rango.
- `IS NULL` le permite seleccionar filas que no contienen datos en una columna determinada.
- `AND` le permite seleccionar solo filas que satisfacen dos condiciones.
- `OR` le permite seleccionar filas que satisfagan cualquiera de dos condiciones.
- `NOT` le permite seleccionar filas que no coinciden con una determinada condición.

Escribir una expresión que devuelva todas las columnas de los artículos cuyo stock no se halla entre el stock mínimo menos 10 unidades y el stock mínimo más 10 unidades. (Usa el operador not between y del carácter * en la cláusula select.)

```
select *
from   articulos
where stock not between stock_min - 10 and stock_min + 10;
```

Escribir el código y nombre de los ciudad pertenecientes a la comunidad bonaerense (Buenos Aires tiene el código de provincia ‘12'; La Plata, el ‘101' y Quilmes, el ‘102').

```
select codciu, nombre from  ciudades
where codpro in ( 12, 101, 102 ) ;
```

Los operadores de comparación también funcionan con datos no numéricos, `=` y `!=` permiten seleccionar filas que coinciden o no con ningún valor, respectivamente.

Sin embargo, existen algunas reglas importantes al utilizar estos operadores. Si está utilizando un operador con valores que son no numérico, hay que poner el valor entre comillas simples: 'value'.

Nota: SQL utiliza comillas simples para hacer referencia a los valores de las columnas.

También puede utilizar `>`, `<` y el resto de los operadores de comparación en columnas no numéricas; filtran según el orden alfabético

2 Base de Datos: Empleados

A continuación vamos a crear una base de datos denominada **Empresa** formada por 4 tablas, donde la **tabla Empleados** almacena los empleados de una empresa, la **tabla Domicilios** almacena los domicilios de estos empleados, la **tabla Teléfonos** almacena los teléfonos de los empleados y la **tabla Códigos postales** almacena los códigos postales de España con referencia a la población y provincia correspondientes.

Hay que considerar que un empleado puede tener varios domicilios o que puede compartir vivienda con otro empleado e incluso que puede no conocerse su domicilio, y también que puede tener varios (o ningún) teléfonos, así como un teléfono puede estar compartido por varios empleados.

1- Creamos la BD

```
/*creamos de base de datos*/

CREATE DATABASE Empresa
```

Seleccionamos, ejecutamos, refrescamos. Vemos que aparece la BD Empresa en el panel izquierdo.

Para poner en la memoria esa BD usamos el comando USE

```
USE Empresa
```

Selecciono, ejecuto y me aparece en un recuadro de la barra de herramientas. Todo el código siguiente que escribamos se estaría aplicando a la BD Empleados.

2- Creamos las Tablas

La primera es la llamada **Empleados**, con su campo DNI de tipo `TEXT(10),` establecemos que esa columna será la llave primaria, y con `NOT NULL` le decimos que esa columna no acepte valores nulos. A través de una coma, agregamos una nueva columna que será Nombre. La misma será de tipo `varchar` con una longitud de 50 caracteres, y también queremos que no tenga valores nulos. Otra columna con la etiqueta Apellido de tipo `varchar` con 50 caracteres, otra Fecha _Nac de tipo `DATETIME`, y Sueldo de tipo `INT.`

```
/* Creamos la tabla Empleados*/

CREATE TABLE Empleados(
Nombre varchar (50) NOT NULL,
Apellido varchar (50) NOT NULL,
DNI varchar (10) Primary key  NOT NULL,
Fecha_Nac char(8),
Sueldo INT
);
```

Seleccionamos, ejecutamos, refrescamos.

Creamos las tablas restantes y luego las vamos a relacionar.

Creamos la tabla llamada **Teléfonos**, con su campo DNI de tipo `TEXT(10)` el cual hace referencia al campo DNI de la tabla Empleados; establecemos como llave primaria el campo Teléfonos y escribimos la sentencia para que no acepte valores nulos.

```
/* Creamos la tabla Teléfonos*/

CREATE TABLE Teléfonos(
DNI varchar (10) REFERENCES Empleados(DNI), Teléfono varchar(9) Primary key  NOT
NULL,
);
```

Seleccionamos, ejecutamos, refrescamos.

```
/* Creamos la tabla Códigos postales*/

CREATE TABLE Códigos_postales(
Codigo_postal varchar(10) Primary key  NOT NULL,
Poblacion varchar (50) NOT NULL,
Provincia varchar (50) NOT NULL,
);
```

Seleccionamos, ejecutamos, refrescamos.

```
/* creamos la tabla Domicilio*/
CREATE TABLE Domicilios(
DNI varchar(10) REFERENCES Empleados(DNI),
Calle varchar(50),
Codigo_postal varchar (10) REFERENCES [Códigos_postales]([Codigo_postal]), Primary key
(DNI, Calle, Codigo_postal))
```

Seleccionamos, ejecutamos, refrescamos.

3- Relacionamos las Tablas

El resultado de las relaciones mostrando las restricciones de cardinalidad es:

Insertar datos

Agregar los datos que se muestran a continuación a las tablas correspondientes.

Tabla Empleados				
Nombre	Apellido	DNI	Fecha_Nac	Sueldo
Antonio	Arjona	12345678A	23/9/1978	5000
Carlota	Cerezo	12345678C	17/8/1981	1000
Laura	López	12345678L	6/2/1985	1500
Pedro	Pérez	12345678P	11/4/1976	2000

Tabla Teléfono	
DNI	**Teléfono**
12345678C	931111111
12345678L	913333333
12345678P	913333333
12345678P	644444444
12345678C	611111111

Tabla Códigos postales		
Código postal	Población	Provincia
8050	Parets	Barcelona
14200	Peñarroya	Córdoba
14900	Lucena	Córdoba
28040	Madrid	Madrid
50008	Zaragoza	Zaragoza
28004	Arganda	Madrid
28000	Madrid	Madrid

Tabla Domicilios		
DNI	Calle	Código Postal
12345678A	Avda. Complutense	28040
12345678A	Cántaro	28004
12345678P	Diamante	15200
12345678P	Carbón	14900
12345678L	Diamante	14200

Para insertar los registros lo podemos hacer de uno en uno, o podemos agregar varios registros a través de una misma instrucción.

```
INSERT INTO Empleados
VALUES ('Antonio', 'Arjona', '12345678A', '23/09/1978', '5000')

INSERT INTO Empleados
VALUES ('Carlota', 'Cerezo', '12345678C', '17/08/1981', '1000')

INSERT INTO Empleados
VALUES ('Laura', 'López', '12345678L', '23709/1978', '1500')

INSERT INTO Empleados
VALUES ('Pedro', 'Pérez', '12345678P', '23709/1978', '2000')
```

Seleccionamos, ejecutamos, refrescamos.

```
INSERT INTO Teléfonos
VALUES ('12345678C', '931111111')

INSERT INTO Teléfonos
VALUES ('12345678L', '913333333')

INSERT INTO Teléfonos
VALUES ('12345678P', '913333333')

INSERT INTO Teléfonos
VALUES ('12345678P', '644444444')

INSERT INTO Teléfonos
VALUES ('12345678C', '611111111')
```

Seleccionamos, ejecutamos, refrescamos.

```
INSERT INTO Códigos_Postales
VALUES ('8050', 'Parets', 'Barcelona')

INSERT INTO Códigos_Postales
VALUES ('14200', 'Peñarroya', 'Córdoba')

INSERT INTO Códigos_Postales
VALUES ('14900', 'Lucena', 'Córdoba')

INSERT INTO Códigos_Postales
VALUES ('28040', 'Madrid', 'Madrid')

INSERT INTO Códigos_Postales
VALUES ('50008', 'Zaragoza', 'Zaragoza')

INSERT INTO Códigos_Postales
VALUES ('28004', 'Arganda', 'Madrid')

INSERT INTO Códigos_Postales
VALUES ('28000', 'Madrid', 'Madrid')
```

Seleccionamos, ejecutamos, refrescamos.

```
INSERT INTO Domicilios
VALUES ('12345678A', ' Avda. Complutense ', '28040')

INSERT INTO Domicilios
VALUES ('12345678A', ' Cántaro', '28004')

INSERT INTO Domicilios
VALUES ('12345678P', ' Diamante ', '15200')

INSERT INTO Domicilios
VALUES ('12345678P', ' Carbón ', '14900')

INSERT INTO Domicilios
VALUES ('12345678L', ' Diamante ', '14200')
```

Consultas

1- Hacer un listado de empleados que muestre Nombre, Calle y Código postal ordenados por Código postal y Nombre.

```
SELECT Nombre, Calle, [Codigo_postal]
FROM Empleados, Domicilios
WHERE Empleados. DNI=Domicilios. DNI
ORDER BY [Codigo_postal], Nombre;
```

2- Listar empleados ordenados por nombre que muestre Nombre, DNI, Calle, Código postal, Teléfono

```
SELECT Nombre, Empleados.DNI, Calle, [Codigo_postal], Teléfono
FROM (Empleados LEFT JOIN Domicilios ON Empleados.DNI=Domicilios.DNI)
INNER JOIN Teléfonos ON Teléfonos.DNI=Empleados.DNI
ORDER BY Nombre;
```

3- Listado de los empleados ordenados por nombre que muestre Nombre, DNI, Calle, Código postal, tengan o no Teléfono

```
SELECT Nombre, Empleados.DNI, Calle, [Codigo_postal], Teléfono
FROM Empleados LEFT JOIN (Teléfonos LEFT JOIN Domicilios ON
Teléfonos.DNI=Domicilios.DNI)
ON Empleados.DNI=Teléfonos.DNI
ORDER BY Nombre;
```

4- Listado de los empleados que muestre Nombre, DNI, Calle, Población, Provincia y Código postal ordenados por nombre

```
SELECT Nombre, Empleados.DNI, Calle, Poblacion, Provincia, Domicilios.[Codigo_postal]
FROM (Empleados LEFT JOIN Domicilios ON Empleados.DNI=Domicilios.DNI)
LEFT JOIN [Códigos postales] ON
Domicilios.[Codigo_postal]=[Códigos_Pstales].[Codigo_postal]
ORDER BY Nombre;
```

5- Incrementar en un 10% el sueldo de todos los empleados, de forma que el sueldo aumentado no supere en ningún caso 1.900.

```
UPDATE Empleados SET Sueldo = Sueldo*1.1
WHERE Sueldo <= 1900/1.1;
```

6- Deshacer la operación anterior con una consulta (comprobar que los datos coinciden con los de la tabla original).

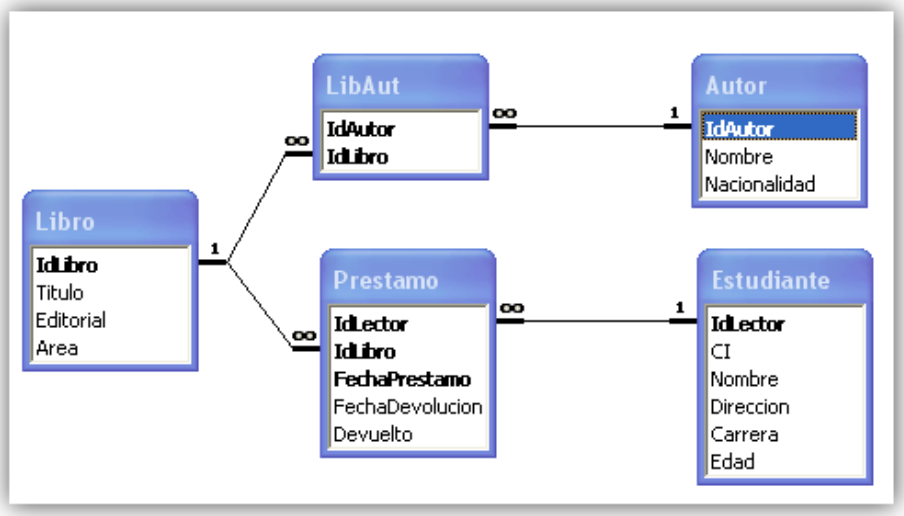
```
UPDATE Empleados SET Sueldo = Sueldo/1.1
WHERE Sueldo <= 1900;
```

7- Listado del número total de empleados, el sueldo máximo, el mínimo y el medio.

```
SELECT COUNT(*) AS Empleados, MIN(Sueldo) AS [Sueldo mínimo],
MAX(Sueldo) AS [Sueldo máximo], AVG(Sueldo) AS [Sueldo medio (AVG)],
SUM(Sueldo)/Empleados AS [Sueldo medio (SUM)]
FROM Empleados;
```

3 Crear Base de Datos: Biblioteca

Un cliente los contrata para generar una base de datos para una biblioteca, como la que se ve en la imagen.



1) Creamos la BD

```
/*creamos de base de datos*/
CREATE DATABASE Biblioteca
```

Seleccionamos, ejecutamos, refrescamos.

Para habilitarla BD usamos el comando USE

```
USE Biblioteca
```

Selecciono, ejecuto, refrescamos.

Generamos la tabla Libro.

```
CREATE TABLE Libro
(IdLibro int identity not null primary key,
Titulo nvarchar(100) not null,
Editorial nvarchar(100) not null,
Area nvarchar(100) not null,
```

Generamos la tabla Estudiante

```
create table Estudiante(
IdLector int identity not null primary key,
CI nvarchar(20) not null,
Nombre nvarchar(100) not null,
Dirección nvarchar(100) not null,
Carrera nvarchar(60) not null,
Edad int not null,
)
```

Generamos la tabla Prestamo

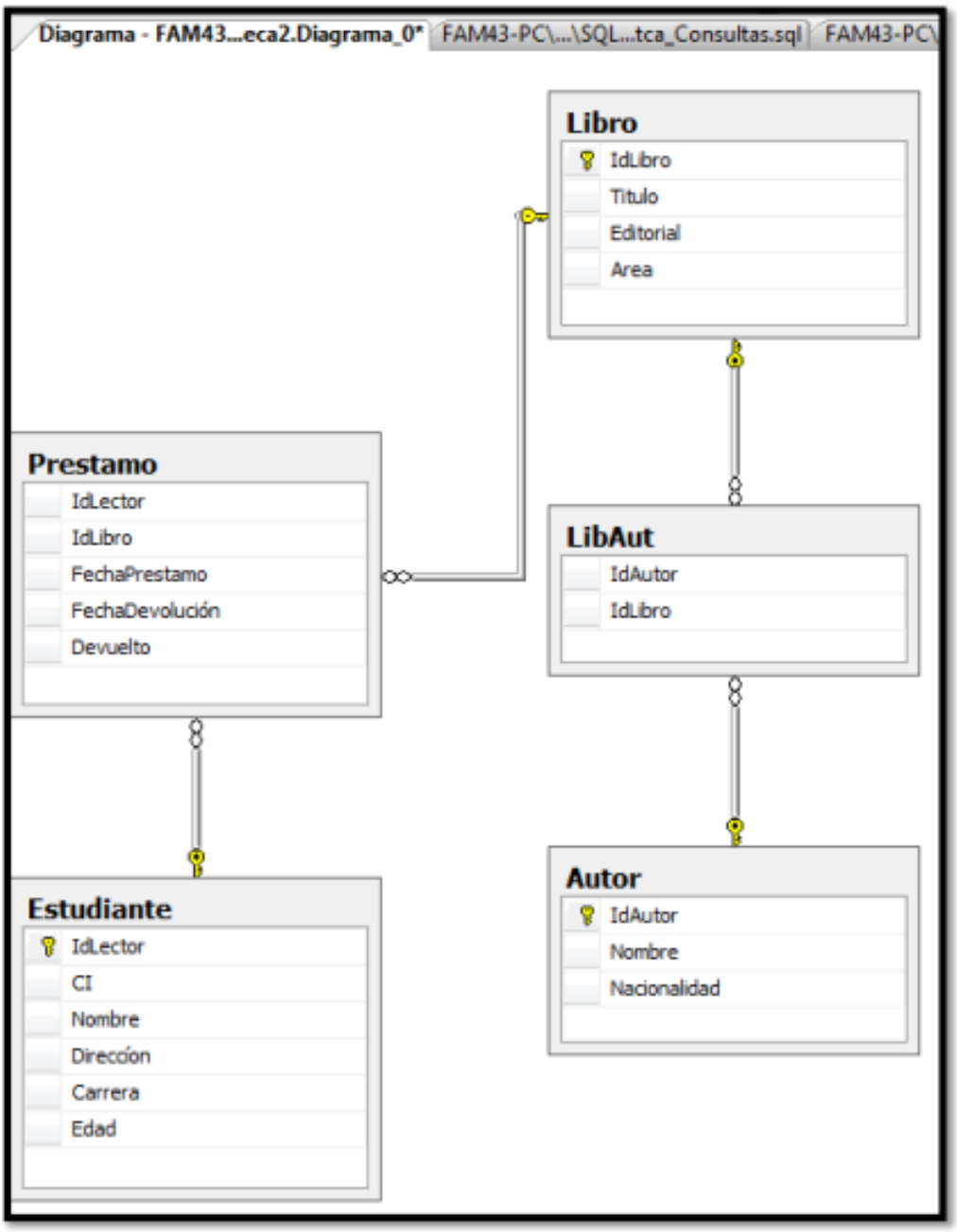
```
create table Prestamo(
IdLector int not null Foreign key
(IdLector) References Estudiante(IdLector),
IdLibro int not null Foreign key (IdLibro) References Libro(IdLibro),
FechaPrestamo nchar(8),
FechaDevolución nchar(8),
Devuelto nchar(8),
)
```

Generamos la tabla Autor

```
create table Autor (
IdAutor int identity not null primary key,
Nombre nvarchar(100) not null,
Nacionalidad nvarchar(30) not null,
)
```

Generamos la tabla LibAut

```
create table LibAut(
IdAutor int not null foreign key
(IdAutor) References Autor(IdAutor),
IdLibro int foreign key
(IdLibro) References Libro(IdLibro),
)
```



Cargar los siguientes datos en cada una de las tablas

```
*** Datos para la tabla Prestamo***

INSERT INTO Prestamo
VALUES (3,1, ' 25/03/2012 ', '10/04/2012', 'NO')

INSERT INTO Prestamo
VALUES (2,3, '02/03/2013', '20/02/2013', '22/02/2013')

INSERT INTO Prestamo
VALUES (5,2, ' 18/02/2013 ', '20/02/2013', '25/02/2013')

INSERT INTO Prestamo
VALUES (6,5, ' 21/02/2013 ', '03/03/2013', '05/03/2013')

INSERT INTO Prestamo
VALUES (12,9, ' 21/02/1999 ', '05/03/2013', '30/02/2013')

INSERT INTO Prestamo
VALUES (4,2, ' 26/02/2013 ', '07/03/2013', '01/03/2013')

INSERT INTO Prestamo
VALUES (3,4, ' 30/02/2013 ', '07/03/2013', '08/03/2013')

INSERT INTO Prestamo
VALUES (1,1, ' 01/03/2012 ', '10/04/2012', 'NO')

INSERT INTO Prestamo
VALUES (6,3, ' 03/03/2013 ', '09/3/2013', '09/03/2013')

INSERT INTO Prestamo
VALUES (3,7, ' 03/03/2013 ', '18/3/2013', '15/03/2013')

INSERT INTO Prestamo
VALUES (2,3, ' 05/03/2013 ', '22/3/2013', '20/03/2013')

INSERT INTO Prestamo
VALUES (4,1, ' 03/04/2013 ', '15/04/2013', 'NO')

INSERT INTO Prestamo
VALUES (2,6, ' 28/03/2013 ', '10/04/2013', 'NO')

INSERT INTO Prestamo
VALUES (1,7, ' 12/03/2013 ', '24/3/2013', '20/03/2013')

INSERT INTO Prestamo
VALUES (12,2, ' 18/03/2013 ', '03/04/2013', '02/04/2013')
```

```
*** Datos para la tabla Autor***

INSERT INTO Autor
VALUES ('Miguel de Cervantes', 'España')

INSERT INTO Autor
VALUES ('william Faulkner', 'USA')

INSERT INTO Autor
VALUES ('Antoine Saint-Exupery', 'Francia')

INSERT INTO Autor
VALUES ('Maquiavelo', 'Italia')

INSERT INTO Autor
VALUES ('Henry Kissinger', 'Alemania')

INSERT INTO Autor
VALUES ('Kitty Kelley', 'Gran Bretaña')

INSERT INTO Autor
VALUES ('Pu-Yi', 'China')

INSERT INTO Autor
VALUES ('Pérez Galdós', 'España')

INSERT INTO Autor
VALUES ('José Manuel Alarcón Aguín', 'España')

INSERT INTO Autor
VALUES ('Federico G. Rudolph', 'España')

INSERT INTO Autor
VALUES ('Guillermo Cicilio', 'Argentina')

INSERT INTO Autor
VALUES ('UÑA, Isaías; SAN MARTÍN, Jesús', 'España')

INSERT INTO Autor
VALUES ('López, Francisco; MALDONADO, Saturnino; RUSA, Manuel', 'España')

INSERT INTO Autor
VALUES ('Mario Benedetti', 'Uruguay')
```

```
*** Datos para la tabla LibAut***

INSERT INTO LibAut
VALUES (1,1)

INSERT INTO LibAut
VALUES (2,2)

INSERT INTO LibAut
VALUES (3,3)

INSERT INTO LibAut
VALUES (4,4)

INSERT INTO LibAut
VALUES (5,5)

INSERT INTO LibAut
VALUES (6,6)

INSERT INTO LibAut
VALUES (7,7)

INSERT INTO LibAut
VALUES (8,8)

INSERT INTO LibAut
VALUES (9,9)

INSERT INTO LibAut
VALUES (10,10)

INSERT INTO LibAut
VALUES (11,11)

INSERT INTO LibAut
VALUES (12,12)

INSERT INTO LibAut
VALUES (13,13)

INSERT INTO LibAut
VALUES (14,14)

INSERT INTO LibAut
VALUES (15,15)
```

```
*** Datos para la tabla Libro***

INSERT INTO Libro
VALUES ('Don Quijote de la Mancha I', 'Anaya', 'Caballeresco')

INSERT INTO Libro
VALUES ('Don Quijote de la Mancha II', 'Anaya', 'Caballeresco')

INSERT INTO Libro
VALUES ('historias de nueva Orleans', 'Alfaguara', 'Novela')

INSERT INTO Libro
VALUES ('El Principito', 'Andina', 'Aventura')

INSERT INTO Libro
VALUES ('El Principe', 'S.M', 'Político')

INSERT INTO Libro
VALUES ('Los Windsor', 'Plaza & Janés', 'Biografías')

INSERT INTO Libro
VALUES ('El Último Emperador', 'Caralt', 'Autobiografías')

INSERT INTO Libro
VALUES ('Fortunata y Jacinta', 'Plaza & Janés', 'Novela')

INSERT INTO Libro
VALUES ('Programación Web con Visual Studio y ASP.NET 2.0', 'Krasis Press',
'Informática')

INSERT INTO Libro
VALUES ('Introducción a Visual Studio.NET', 'Bubok', 'Informática')

INSERT INTO Libro
VALUES ('IPv6 para Todos. Guía de uso y Aplicación para diversos entornos', 'capítulo
Argentino', 'Informática')

INSERT INTO Libro
VALUES ('calculo en una Variable-707 Ejercicios Desarrollados', 'Alfaomega',
'Ciencias')

INSERT INTO Libro
VALUES ('Análisis de circuitos Lineales- 3° Ed.', 'Alafaomega, Ra-Ma', 'Eléctrica')

INSERT INTO Libro
VALUES ('La Tregua', 'Sudamericana', 'Ficción y Literatura')
```

```
*** Datos para la tabla Estudiante***

INSERT INTO Estudiante
VALUES ('42.117.892-S', 'Inés Maza Rodríguez', 'Av. Escalerita 12', 'Contaduría','19')

INSERT INTO Estudiante
VALUES ('31.765.348-D', 'José Meléndez García', 'Mesa y López 51', 'TIC-SI','18')

INSERT INTO Estudiante
VALUES (11.542.981-G', 'Miguel', 'Gran Vía 71', 'Alimentos','20')

INSERT INTO Estudiante
VALUES ('78.542.450-L', 'Eva', 'pío Baroja 23', 'Paramédico','19')

INSERT INTO Estudiante
VALUES ('44.312.870-Z', 'Yolanda Lozano Encabo', 'El Cid 45', 'Cs. Políticas','19')

INSERT INTO Estudiante
VALUES ('73.735.398-C', 'Louisa  Lopez Rubin', 'Av. Clavijero 101', 'Informática','20')

INSERT INTO Estudiante
VALUES ('47.234.471-P', 'Juan López Vazquez', 'Jaime I, 65', 'Administración,'21')

INSERT INTO Estudiante
VALUES ('84.954.509-A', 'Anselmo Menendez', 'Privada 102', 'Astronomía','21')

INSERT INTO Estudiante
VALUES ('98.765.627-A', 'Gudencio Noriega', 'Av. Rosas 604', 'Arquitectura','19')

INSERT INTO Estudiante
VALUES ('34.465.676-B', 'Genaro Lozano', '21 Oriente 1009', 'Informática','25')

INSERT INTO Estudiante
VALUES ('75.265.983-A', 'Artemio Armesto Rodriguez', 'Coyoacan #28', 'Cs.
Políticas','21')

INSERT INTO Estudiante
VALUES ('23.365.676-G', 'Raúl Valdéz', '14 Sur N° 522', 'Criminalística','20')

INSERT INTO Estudiante
VALUES ('87.265.313-R', 'Carlos Arias', '10 Oriente #7', 'Derecho','23')

INSERT INTO Estudiante
VALUES ('65.765.348-E', 'Juan Anaya', '2 Oriente No. 212', 'Físico Matemática','19')
```

Generar las siguientes consultas

1- Listar los datos de los autores

2- Listar nombre y edad de los estudiantes

3- ¿Qué estudiantes pertenecen a la carrera de Informática?

4- Listar los nombres de los estudiantes cuyo apellido comience con la letra G?

5- ¿Quiénes son los autores del libro "Introducción a Visual Studio .NET", listar solamente los nombres?

6- ¿Qué autores son de nacionalidad USA o Francia?

7- ¿Qué libros No Son del Área de Internet?

8- ¿Qué libros se prestó el Lector "Raúl Valdez "?

9- Listar el nombre del estudiante de menor edad

10- Listar los nombres de los estudiantes que se prestaron Libros de Base de Datos

11- Listar los libros de editorial AlfaOmega

12- Listar los libros que pertenecen al autor Mario Benedetti

13- Listar los títulos de los libros que debían devolverse el 10/04/12

14- Hallar la suma de las edades de los estudiantes

15- Listar los datos de los estudiantes cuya edad es mayor al promedio

Creación de "Vistas" en la BD de Biblioteca:

```
CREATE VIEW Vista1
AS
SELECT
Libro. Area,
Libro. Titulo,
Prestamo.IdLibro,
Prestamo.FechaPrestamo
FROM
Libro, Prestamo
WHERE
Libro.IdLibro= Prestamo.IdLibro

SELECT *FROM Vista1
```

```
CREATE VIEW Vista2
AS
SELECT
Estudiante.CI,
Estudiante.Nombre,
Estudiante.Carrera,
Prestamo.IdLibro,
Prestamo.FechaPrestamo
FROM
Estudiante, Prestamo
WHERE
Estudiante.IdLector = Prestamo.IdLector

SELECT *FROM Vista2
```

```
CREATE VIEW Vista3
AS
SELECT
LibAut.IdLibro,
Autor.Nombre,
Autor.Nacionalidad,
FROM
LibAut, Autor
WHERE
LibAut.IdAutor = Autor.IdAutor

SELECT *FROM Vista3
```

Creación de Índices en la BD

```
CREATE INDEX IDX_LIBRO_LibrosxArea ON Libro (Titulo, Area)

CREATE INDEX IDX_AUTOR_Nacion ON (Nacionalidad)

CREATE INDEX IDX_ESTUDIANTE_CodigoE ON Estudiante (CI, Carrera)
```

Respuestas BD Biblioteca

1- Listar los datos de los autores

```
SELEC * FROM Autor
```

2- Listar nombre y edad de los estudiantes

```
SELECT Nombre, Edad FROM Estudiante
```

3- ¿Qué estudiantes pertenecen a la carrera de Informática?

```
SELECT Nombre FROM Estudiante WHERE Carrera= 'Informática'
```

4- Listar los nombres de los estudiantes cuyo apellido comience con la letra G?

```
SELECT Nombre FROM Estudiante WHERE Nombre LIKE 'G%'
```

5- ¿Quiénes son los autores del libro "Introducción a Visual Studio .NET", listar solamente los nombres?

```
SELECT Nombre FROM Autor WHERE IdAutor IN
(
 SELECT IdAutor FROM LibAut WHERE IdLibro IN
 (SELECT IdLibro FROM Libro WHERE Titulo= 'Introducción a la Visual Studio.NET'
)
```

6- ¿Qué autores son de nacionalidad USA o Francia?

```
SELECT * FROM Autor WHERE Nacionalidad  IN 'USA', 'Francia'
```

7- ¿Qué libros No Son del Área de Internet?

```
SELECT * FROM Libro WHERE Area <> 'Internet'
```

8- ¿Qué libros se prestó el Lector "Raúl Valdez "?

```
SELECT * FROM Libro WHERE IdLibro IN
(
 SELECT IdLibro FROM Prestamo WHERE IdLector IN
 (SELECT IdLector FROM Estudiante WHERE Nombre= 'Raúl Valdéz'
)
```

9- Listar el nombre del estudiante de menor edad

```
SELECT Nombre FROM Estudiante WHERE Edad IN ( SELECT MIN (Edad) FROM Estudiante)
```

10- Listar los nombres de los estudiantes que se prestaron Libros de Base de Datos

```
SELECT * FROM Estudiante WHERE IdLector IN
(
 SELECT IdLector FROM Prestamo WHERE IdLibro IN
 (SELECT IdLibro FROM Libro WHERE Area= 'Base de Datos'
)
```

11- Listar los libros de editorial AlfaOmega

```
SELECT * FROM Libro WHERE Editorial= 'AlfaOmega'
```

12- Listar los libros que pertenecen al autor Mario Benedetti

```
SELECT * FROM Libro WHERE IdLibro IN
(
 SELECT IdLibro FROM LibAut WHERE IdAutor IN
 (SELECT IdAutor FROM Autor WHERE Nombre = 'Mario Benedetti'
)
```

13- Listar los títulos de los libros que debían devolverse el 10/04/12

```
SELECT * FROM Libro WHERE IdLibro IN
(
 SELECT IdLibro FROM Prestamo WHERE Fechadevolución = '10/04/2012'
 AND Devuelto = 'NO'
 )
```

14- Hallar la suma de las edades de los estudiantes

```
SELECT SUM (Edad) AS [ La suma de las edades es:] FROM Estudiante
```

15- Listar los datos de los estudiantes cuya edad es mayor al promedio

```
SELECT * FROM Estudiante WHERE Edad >
 ( SELECT AVG (Edad) FROM Estudiante )
```