

BASES DE DATOS

# Diseño físico

©Jesús García, 2021 | <http://jgarcia.dev>



BASES DE DATOS

# Diseño físico

## Introducción

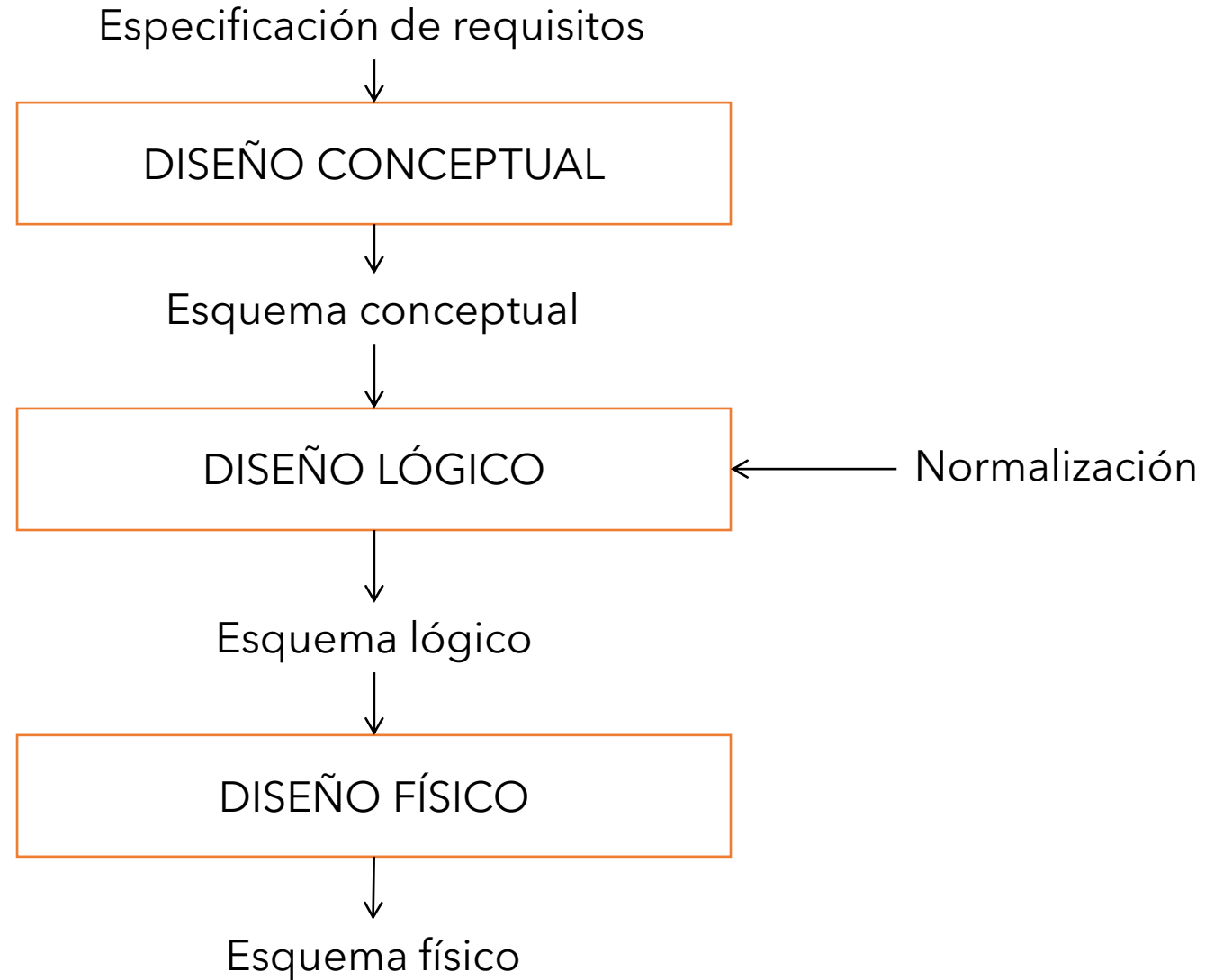
Modelo físico

Lenguaje SQL

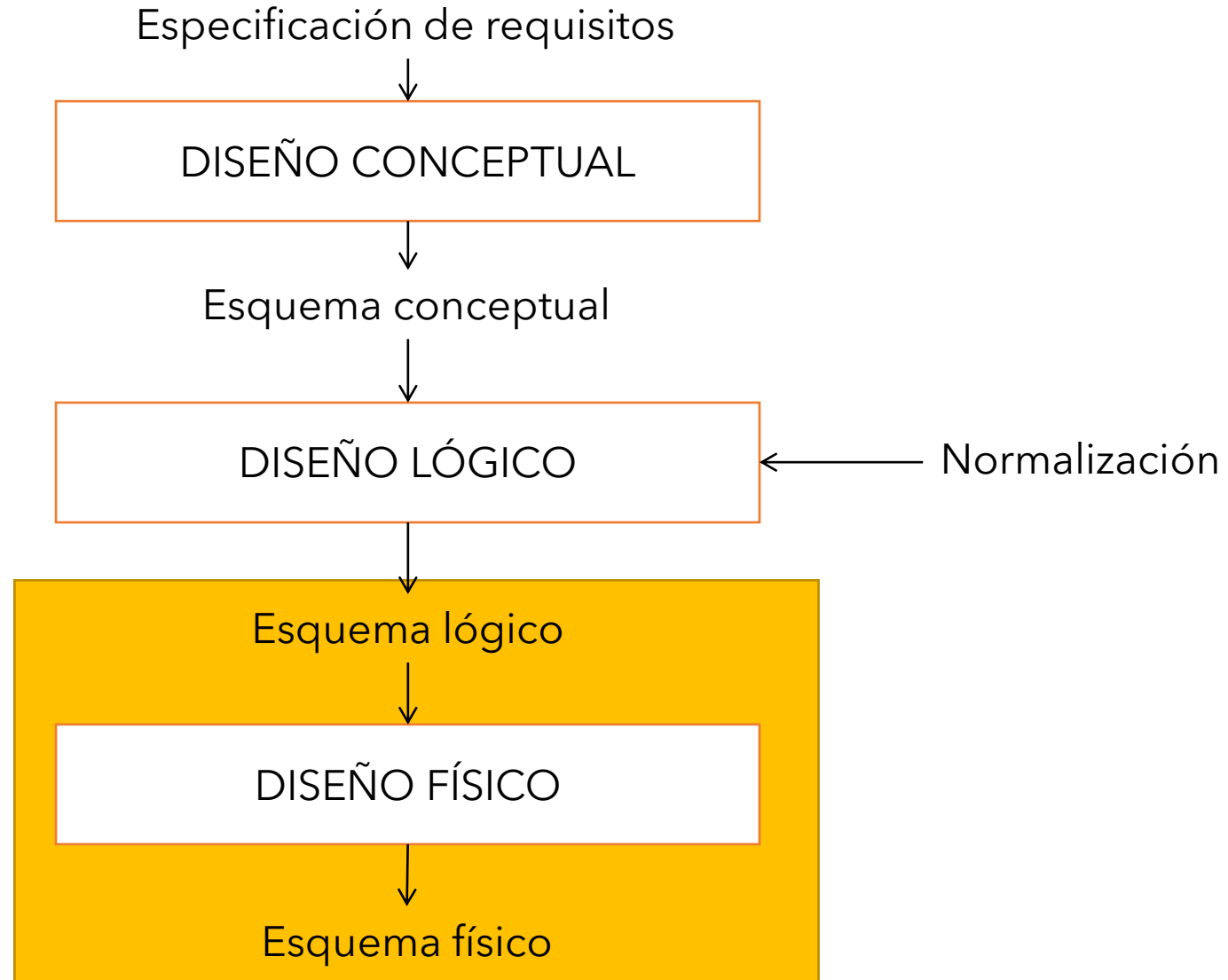
MySQL

Data Definition Language (DDL)

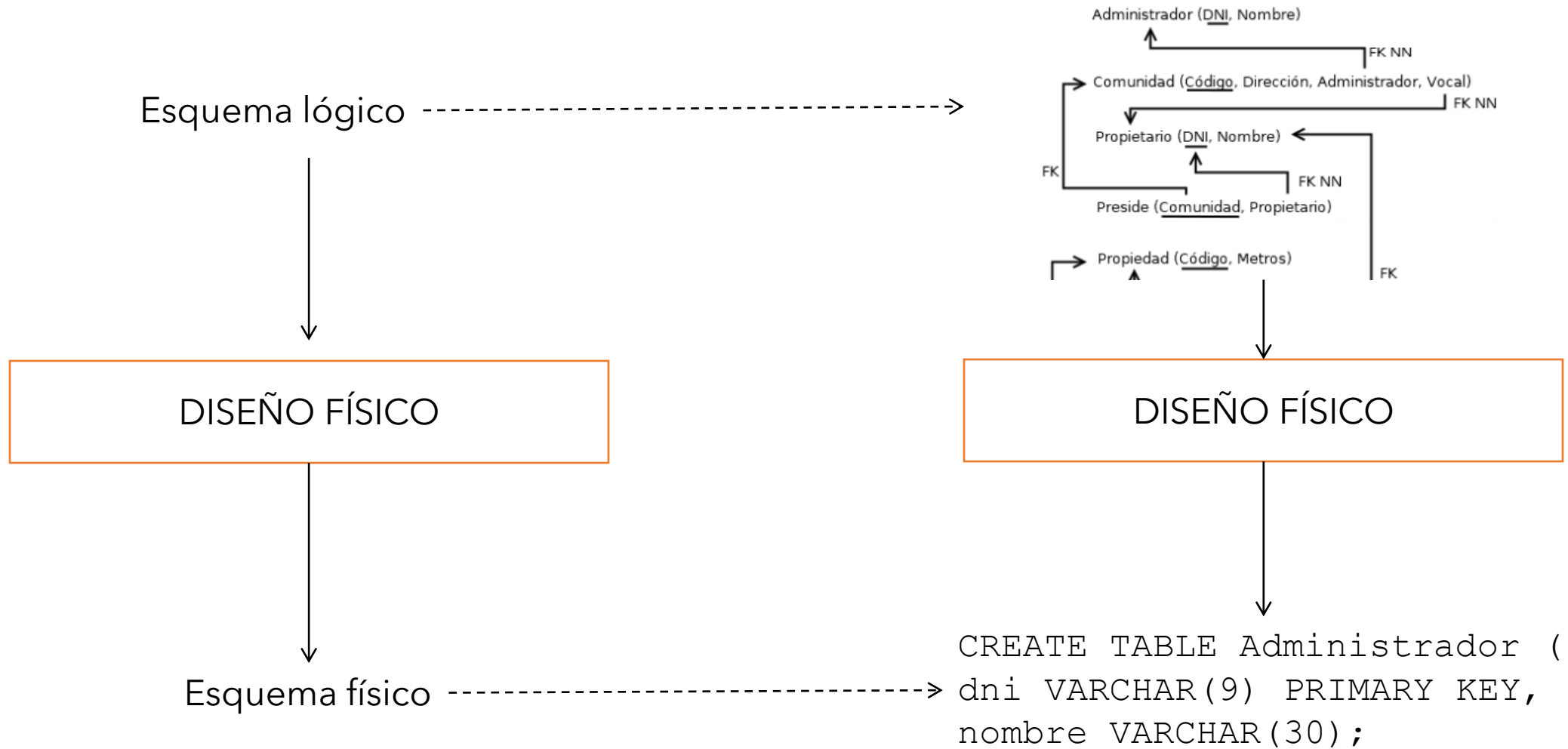
# Introducción



# Introducción



# Introducción



BASES DE DATOS

# Diseño físico

Introducción

**Modelo físico**

Lenguaje SQL

MySQL

Data Definition Language (DDL)

# Modelo físico

- Representa la estructura de la base de datos.
- Incluye:
  - Especificación de las tablas y columnas.
  - Restricciones de unicidad, valor no nulo, integridad referencial, etc.
- Puede variar según el SGBD.
- Para definirlo se hace uso del lenguaje SQL.

BASES DE DATOS

# Diseño físico

Introducción

Modelo físico

**Lenguaje SQL**

MySQL

Data Definition Language (DDL)



# Lenguaje SQL

- Lenguaje fundamental de las bases de datos relacionales.
- Lenguaje declarativo que permite establecer qué se quiere hacer y no cómo se debe hacer.
- Diseñado para el acceso y manejo de datos en bases de datos relacionales.

# Lenguaje SQL

- Es un estándar siendo su última versión la SQL:2016.
- Los SGBD relacionales implementan el estándar aunque pueden existir pequeñas diferencias.
- Se divide en cuatro categorías: lenguaje de definición de datos, lenguaje de manipulación de datos, lenguaje de consulta de datos y lenguaje de control de datos.

# Lenguaje SQL

## Lenguaje de definición de datos Data Definition Language (DDL)

Permite crear y modificar la estructura de las tablas de la base de datos.

```
CREATE TABLE pet (  
    name VARCHAR(20),  
    owner VARCHAR(20),  
    birth DATE  
);
```

# Lenguaje SQL

## Lenguaje de manipulación de base de datos Data Manipulation Language (DML)

Permite insertar, modificar y eliminar datos de la base de datos.

```
INSERT INTO Customers (CustomerName, ContactName)  
VALUES ('Cardinal', 'Tom B. Erichsen');
```

```
UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;
```

# Lenguaje SQL

## Lenguaje de consulta de datos Data Query Language (DQL)

Permite realizar consultas sobre los datos de la base de datos.

```
SELECT CustomerName, City FROM Customers;
```

```
SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
ORDER BY COUNT(CustomerID) DESC;
```

# Lenguaje SQL

## **Lenguaje de control de datos Data Control Language (DCL)**

Permite gestionar permisos de usuario y trabajar con transacciones.

```
GRANT CREATE TABLE TO juanito;
```

BASES DE DATOS

# Diseño físico

Introducción

Modelo físico

Lenguaje SQL

**MySQL**

Data Definition Language (DDL)

# MySQL

- SGBD relacional propiedad de Oracle.
- Una de las base de datos más populares junto a Oracle y SQL Server.
- Muy utilizada en aplicaciones web, como *Joomla*, *Wordpress*, *Drupal* o *phpBB*.

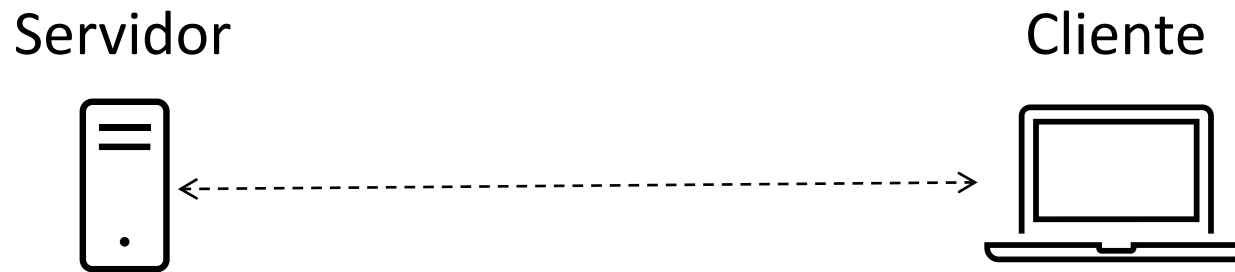




# MySQL

## Arquitectura cliente - servidor.

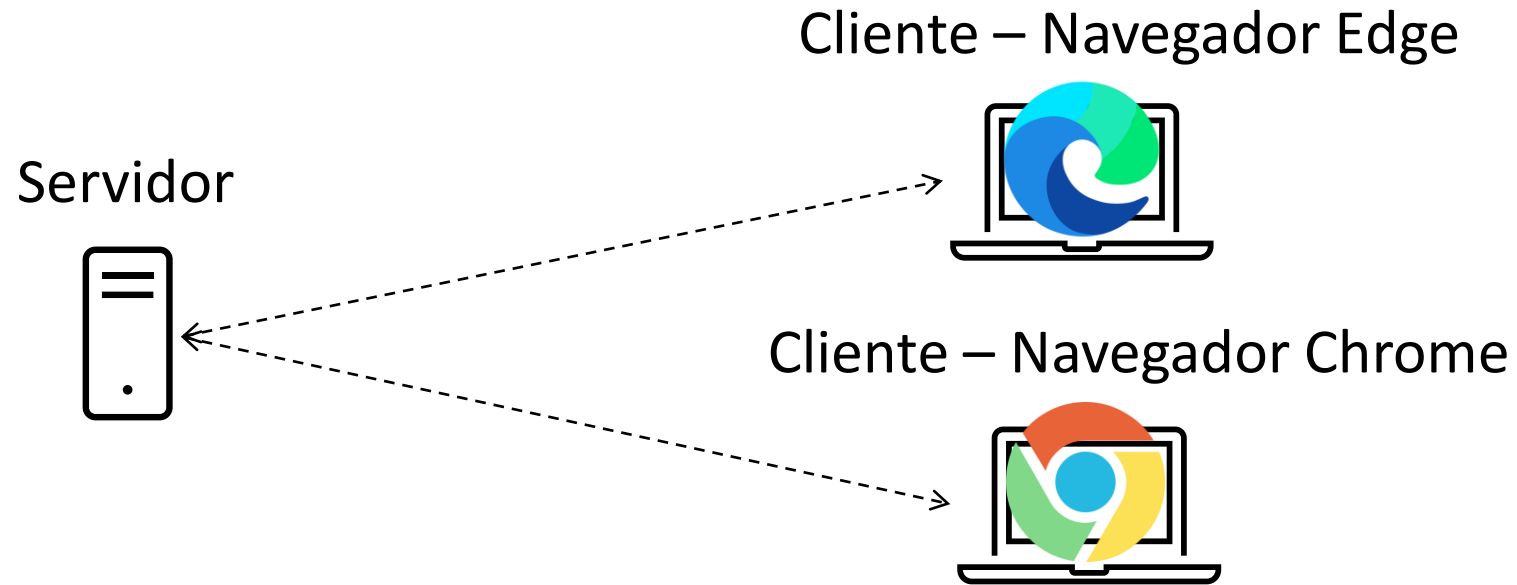
Modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**.



# MySQL

## Arquitectura cliente - servidor.

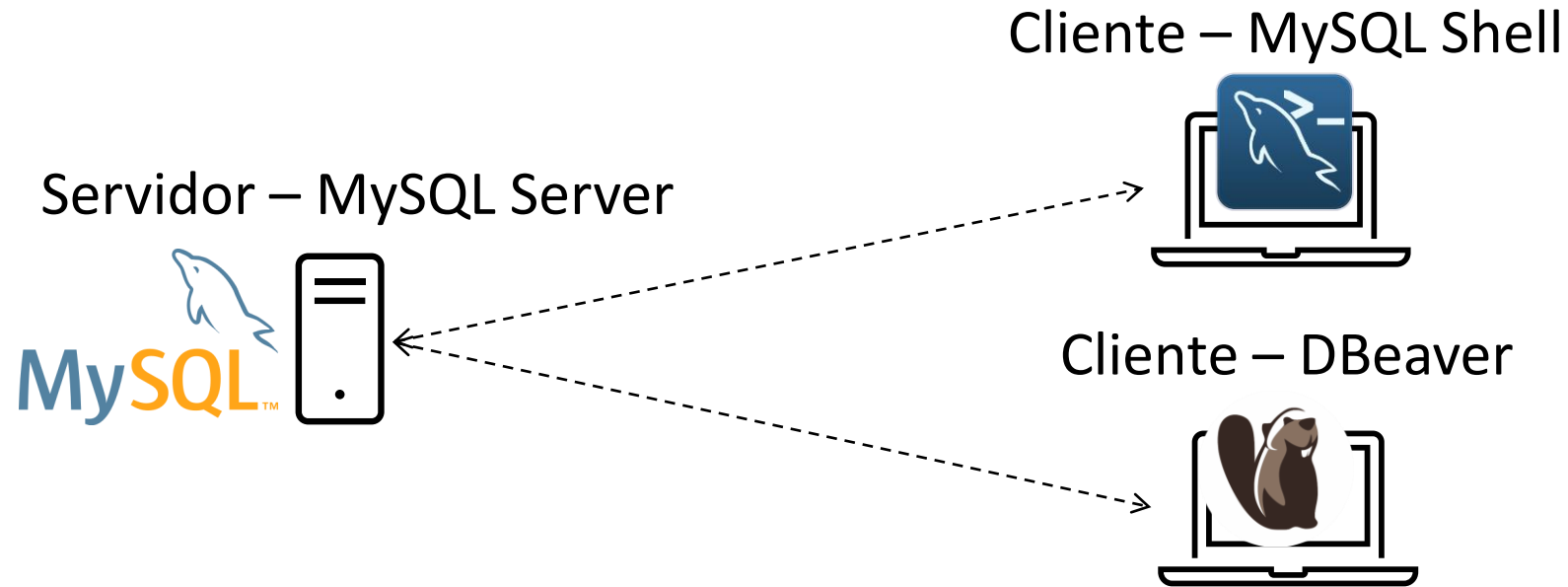
El ejemplo clásico de esta arquitectura es Internet.



# MySQL

## Arquitectura cliente - servidor.

Las bases de datos también utilizan esta arquitectura.





## Comandos MySQL Shell.

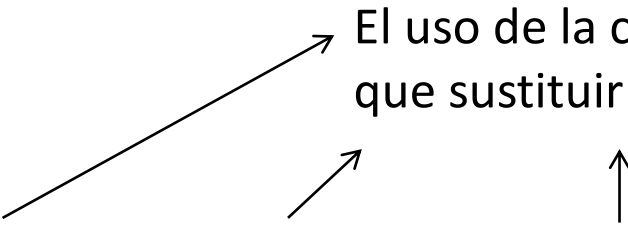
Comando	Descripción
<code>\help</code>	Muestra ayuda
<code>\connect</code>	Conecta a un servidor de MySQL
<code>\js</code>	Cambia el modo de ejecución a JavaScript
<code>\py</code>	Cambia el modo de ejecución a Python
<code>\sql</code>	Cambia el modo de ejecución a SQL.
<code>\quit</code>	Salir de MySQL Shell

# MySQL

## Comando ***connect***.

El uso de la cursiva significa que se tiene que sustituir por el valor adecuado.

```
\connect usuario@servidor:puerto
```



**usuario:** usuario con el que queremos conectarnos a la base de datos. Por defecto lo haremos con el usuario *root*.

**servidor:** IP del ordenador donde se encuentra el servidor MySQL. Puesto que nos vamos a conectar al servidor que está en nuestra propia máquina utilizaremos *localhost* o *127.0.0.1*.

**puerto:** número de puerto en el que escucha el servidor. Por defecto *3306*.

# MySQL

## Comando *connect*.

```
\connect usuario@servidor:puerto
```

### **Cómo interpretar la sintaxis de los comandos**

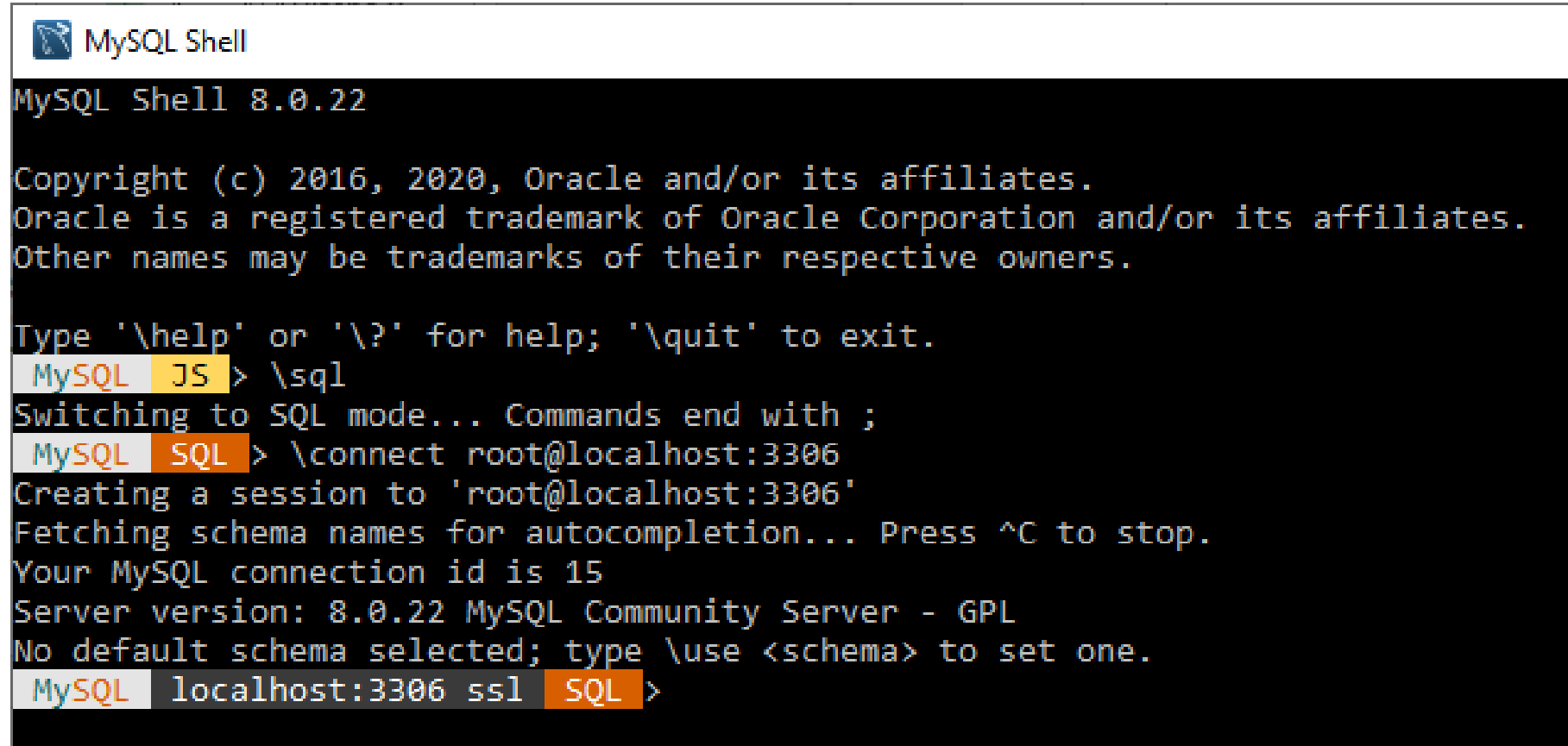
Cuando se utiliza letra en cursiva significa el usuario debe sustituir esa palabra por el valor que quiera.

En el ejemplo anterior, el usuario debe indicar el usuario, servidor y puerto, como veremos en la siguiente diapositiva.

# MySQL

## Comando *connect*.

```
\connect root@localhost:3306
```



```
MySQL Shell 8.0.22

Copyright (c) 2016, 2020, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \sql
Switching to SQL mode... Commands end with ;
MySQL SQL > \connect root@localhost:3306
Creating a session to 'root@localhost:3306'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 15
Server version: 8.0.22 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL localhost:3306 ssl SQL >
```

# MySQL

## Comandos SQL.

Comando	Descripción
<code>show databases;</code>	Lista el nombre de las bases de datos disponibles.
<code>use db_name;</code>	Selecciona la base de datos <i>db_name</i> .
<code>show tables;</code>	Muestra las tablas de la base de datos seleccionada.
<code>describe table_name;</code>	Muestra el formato de los campos de la tabla <i>table_name</i> .
<code>show columns from table_name;</code>	Muestra las columnas, su formato y restricciones de la tabla <i>table_name</i> .



Estos comandos solo se pueden ejecutar en modo de ejecución SQL (comando \sql).



BASES DE DATOS

# Diseño físico

Introducción

Modelo físico

Lenguaje SQL

MySQL

**Data Definition Language (DDL)**

# Data Definition Language

Parte del lenguaje SQL que realiza la función de definición de datos del SGBD.

Fundamentalmente se encarga de la creación, modificación y eliminación de los objetos de la base de datos .

# Data Definition Language

## Creación de una base de datos

```
CREATE DATABASE NombreBaseDatos;
```

Ejemplo:

```
CREATE DATABASE empresa;
```

# Data Definition Language

## Creación de una tabla

```
CREATE TABLE nombreDeTabla(  
    nombreDeLaColumna tipoDeDatos [DEFAULT valor] [restricción] [,...]  
    [restricción] [,...]);
```

### Cómo interpretar la sintaxis de los comandos

Los corchetes [] significa que ese fragmento es opcional. Los puntos suspensivos [,...] significa que el fragmento que le precede se puede repetir.

# Data Definition Language

## Creación de una tabla

```
CREATE TABLE nombreDeTabla (  
    nombreDeLaColumna tipoDeDatos [DEFAULT valor] [, ...]  
    [restricciones] [, ...]);
```

Ejemplo:

```
CREATE TABLE Cliente (  
    nombre VARCHAR(25));
```

# Data Definition Language

## Creación de una tabla

```
CREATE TABLE nombreDeTabla (  
    nombreDeLaColumna tipoDeDatos [DEFAULT valor] [, ...]  
    [restricciones] [, ...]);
```

Ejemplo:

```
CREATE TABLE Proveedor (  
    nombre VARCHAR(25),  
    localidad VARCHAR(30) DEFAULT 'Palencia');
```

# Data Definition Language

## Tipos de datos

Tipo	Descripción
INTEGER	Permite almacenar valores enteros entre -2147483648 y 2147483647.
DECIMAL ( $p$ , $s$ )	Número decimal exacto donde $p$ es el número de dígitos del número y $s$ cuántos de esos dígitos forman parte de la parte decimal.
DOUBLE	Número decimal aproximado
CHAR ( $n$ )	Cadena de caracteres de longitud fija $n$ .
VARCHAR ( $n$ )	Cadena de caracteres de longitud variable $n$ .

# Data Definition Language

## Tipos de datos

Tipo	Descripción
DATE	Utilizado para almacenar fechas en formato AAAA-MM-DD.
TIME	Contiene una hora en formato HH:MM:SS.
DATETIME	Almacena una fecha y hora en formato AAAA-MM-DD HH:MM:SS
TIMESTAMP	Incluye tanto la fecha como la hora y añade la zona horaria. Por ejemplo: 2017-05-32 20:40:58 UTC



# Data Definition Language

## Nomenclatura

A la hora de asignar nombres (bases de datos, tablas y columnas) hay que tener en cuenta que:

- Evitaremos tildes y caracteres especiales como %, \$, etc.
- El único carácter especial que utilizaremos será el guion bajo \_ para separar palabras.

Por ejemplo: `perfil_cliente`

- El nombre de las tablas podrán ser en singular o plural, pero **siempre seguiremos la misma nomenclatura.**

# Data Definition Language

## Valores por defecto

Valor que se guardará en una columna si a la hora de insertar una fila no se indica el valor para dicha columna.



Si el valor es una cadena de texto o una fecha se encierra entre comillas.

```
CREATE TABLE proveedor (  
    nombre VARCHAR(25),  
    localidad VARCHAR(30) DEFAULT 'Palencia');
```

# Data Definition Language

## Valores por defecto

```
CREATE TABLE personaje (  
    nombre VARCHAR(25),  
    nivel INT DEFAULT 1  
);
```

# Data Definition Language

## Valores por defecto

```
CREATE TABLE ciudad (  
    nombre VARCHAR(25),  
    temperatura_media DECIMAL(3,1) DEFAULT 24.5,  
    pais VARCHAR(25) DEFAULT 'España'  
);
```

# Data Definition Language

## Valores por defecto

```
CREATE TABLE socio (  
    nombre VARCHAR(25),  
    apellidos VARCHAR(25),  
    inscripcion DATE DEFAULT '2020/04/10'  
);
```

# Data Definition Language

## Borrar tablas

```
DROP TABLE IF EXISTS NombreTabla;
```

Ejemplo:

```
DROP TABLE IF EXISTS proveedor;
```

# Data Definition Language

## Transformación de relaciones

- Las relaciones se transforman en tablas.
- Los atributos se transforman en columnas.
- Debemos escoger el tipo adecuado para cada columna.
- Debemos aplicar las normas de nomenclatura a la hora de asignar nombres.
- Los atributos identificadores intentaremos que siempre sean de tipo INTEGER a no ser que sea un DNI.



De momento se explican las transformaciones sin restricciones (por eso no hay claves primarias, ajenas, no nulos, etc. En las próximas diapositivas se explica cómo transformar estas restricciones.

# Data Definition Language

## Transformación de relaciones

Ejemplo:

```
Alumno(id, nombre, fechaNacimiento);
```

```
CREATE TABLE alumno (  
    id INTEGER,  
    nombre VARCHAR(50),  
    fecha_nacimiento DATE  
);
```



# Data Definition Language

## Restricciones

Una restricción es una condición de obligado cumplimiento para una o más columnas de la tabla.

Las restricciones que podemos aplicar son:

- Restricción de clave primaria: PRIMARY KEY.
- Restricción de valor no nulo: NOT NULL.
- Restricción de unicidad: UNIQUE.
- Restricción de clave ajena: FOREIGN KEY.
- Restricción de validación: CHECK.

# Data Definition Language

## Restricciones de columna y tabla

Algunas restricciones se pueden definir a nivel de columna y/o tabla.

- Las restricciones de tabla pueden hacer referencia a múltiples columnas mientras que las restricciones de columna no.
- Si se trata de una restricción de tabla opcionalmente podemos asignar un nombre a la restricción.
- No todas las restricciones se pueden definir a nivel de columna o tabla.

# Data Definition Language

## Nombre de las restricciones

A la hora de nombrar una restricción es conveniente seguir un convenio. Nosotros seguiremos el siguiente:

*tabla\_columnas\_restricción*

Donde:

- *tabla* es el nombre de la tabla
- *columnas* el nombre de las columnas implicadas
- *restricción* son dos letras que identifican el tipo de restricción: pk, uk, ck, fk.

# Data Definition Language

## PRIMARY KEY

Autor (código, nombre)

```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50)  
);
```

**Restricción de columna**

# Data Definition Language

## PRIMARY KEY

Autor (código, nombre)

```
CREATE TABLE autor (  
    codigo INTEGER,  
    nombre VARCHAR(50),  
    CONSTRAINT autor_codigo_pk PRIMARY KEY (codigo)  
);
```

**Restricción de tabla**

# Data Definition Language

## PRIMARY KEY

Autor (código, nombre)

```
CREATE TABLE autor (  
    codigo INTEGER,  
    nombre VARCHAR(50),  
    CONSTRAINT PRIMARY KEY (codigo)  
);
```

**Restricción de tabla**

# Data Definition Language

## PRIMARY KEY

Autor (código, nombre)

```
CREATE TABLE autor (  
    codigo INTEGER,  
    nombre VARCHAR(50),  
    PRIMARY KEY (codigo)  
);
```

**Restricción de tabla**

# Data Definition Language

## PRIMARY KEY

Habitación(número, hotel, superficie)

```
CREATE TABLE habitacion (  
    numero INTEGER,  
    hotel INTEGER,  
    superficie DECIMAL (4,2),  
    CONSTRAINT habitacion_numero_hotel_pk  
    PRIMARY KEY (numero, hotel)  
);
```

**Restricción de tabla**



# Data Definition Language

## PRIMARY KEY

Habitación(número, hotel, superficie)

```
CREATE TABLE habitacion (  
    numero INTEGER,  
    hotel INTEGER,  
    superficie DECIMAL (4,2),  
    CONSTRAINT PRIMARY KEY (numero, hotel)  
);
```

**Restricción de tabla**

# Data Definition Language

## PRIMARY KEY

Habitación(número, hotel, superficie)

```
CREATE TABLE habitacion (  
    numero INTEGER,  
    hotel INTEGER,  
    superficie DECIMAL (4,2),  
    PRIMARY KEY (numero, hotel)  
);
```

**Restricción de tabla**

# Data Definition Language

## NOT NULL

Alumno (dni, nombre, nota, fecha\_nacimiento)  
NN

```
CREATE TABLE Alumno (  
    dni CHAR(9) PRIMARY KEY,  
    nombre VARCHAR(50),  
    nota DECIMAL(3,1),  
    fecha_nacimiento DATE NOT NULL  
);
```

**Restricción de columna**



La restricción NOT NULL solo se puede definir como restricción de columna

# Data Definition Language

## NOT NULL

Alumno (dni, nombre, nota, fecha\_nacimiento)  
NN

```
CREATE TABLE alumno (  
    dni CHAR(9) PRIMARY KEY,  
    nombre VARCHAR(50),  
    nota DECIMAL(3,1) DEFAULT 0 NOT NULL,  
    fecha_nacimiento DATE  
);
```

**Restricción de columna**



La restricción NOT NULL solo se puede definir como restricción de columna

# Data Definition Language

## UNIQUE

Autor (código, nombre, libro)  
UK

```
CREATE TABLE autor (  
    codigo INTEGER,  
    nombre VARCHAR(50),  
    libro VARCHAR(50) UNIQUE  
);
```

**Restricción de columna**

# Data Definition Language

## UNIQUE

Alumno (dni, número, nota)  
UK NN

```
CREATE TABLE alumno (  
    dni CHAR(9) PRIMARY KEY,  
    numero INTEGER,  
    nombre VARCHAR(50),  
    nota DECIMAL(3,1) DEFAULT 0 NOT NULL,  
    CONSTRAINT alumno_numero_uk UNIQUE (numero)  
);
```

**Restricción de tabla**

# Data Definition Language

## UNIQUE

Alumno (dni, número, nota)  
UK NN

```
CREATE TABLE alumno (  
    dni CHAR(9) PRIMARY KEY,  
    numero INTEGER,  
    nombre VARCHAR(50),  
    nota DECIMAL(3,1) DEFAULT 0 NOT NULL,  
    CONSTRAINT UNIQUE (numero)  
);
```

**Restricción de tabla**

# Data Definition Language

## UNIQUE

Alumno (dni, número, nota)  
UK NN

```
CREATE TABLE alumno (  
    dni CHAR(9) PRIMARY KEY,  
    numero INTEGER,  
    nombre VARCHAR(50),  
    nota DECIMAL(3,1) DEFAULT 0 NOT NULL,  
    UNIQUE (numero)  
);
```

**Restricción de tabla**



# Data Definition Language

## UNIQUE

Alumno (dni, número, nota)  
NN UK

```
CREATE TABLE alumno (  
    dni CHAR(9) PRIMARY KEY,  
    numero INTEGER,  
    nombre VARCHAR(50) ,  
    nota DECIMAL(3,1) DEFAULT 0 NOT NULL UNIQUE  
);
```

**Restricción de columna**

# Data Definition Language

## UNIQUE

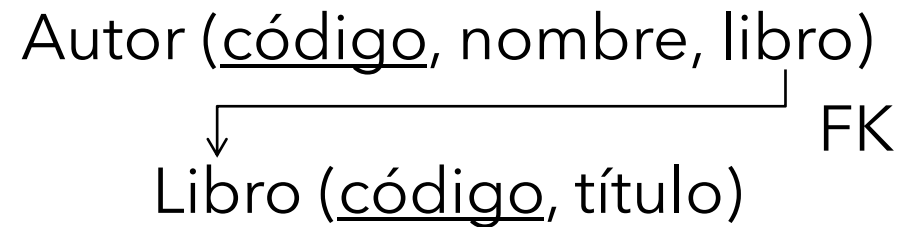
Alumno (dni, número, nota)  
NN UK

```
CREATE TABLE alumno (  
    dni CHAR(9),  
    numero INTEGER,  
    nombre VARCHAR(50),  
    nota DECIMAL(3,1) DEFAULT 0 NOT NULL,  
    CONSTRAINT alumno_dni_pk PRIMARY KEY(dni),  
    CONSTRAINT alumno_nota_uk UNIQUE (nota)  
);
```

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY

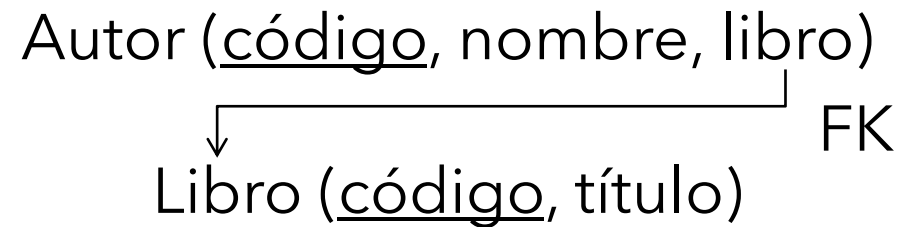


A la hora de crear la clave ajena hay que tener en cuenta que:

- El tipo de la columna que es clave ajena debe ser del mismo tipo que la clave primaria de la tabla a la que referencia.
- La columna referenciada debe ser clave primaria.

# Data Definition Language

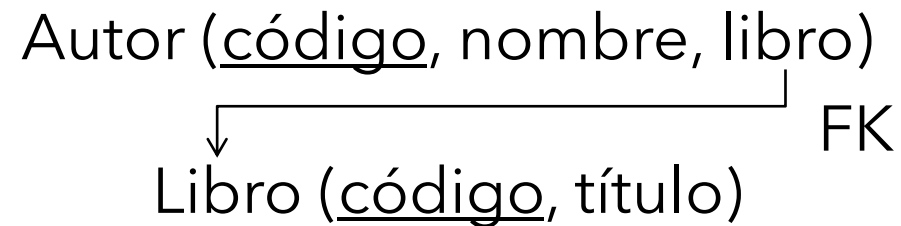
## FOREIGN KEY



```
CREATE TABLE libro (  
    codigo INTEGER PRIMARY KEY,  
    titulo VARCHAR(50)  
);
```

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER,  
    CONSTRAINT autor_libro_fk FOREIGN KEY (libro)  
    REFERENCES libro (codigo)  
);
```

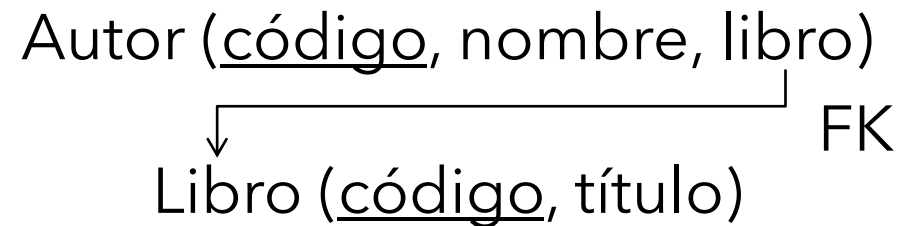


La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER,  
    CONSTRAINT FOREIGN KEY (libro)  
        REFERENCES libro (codigo)  
);
```

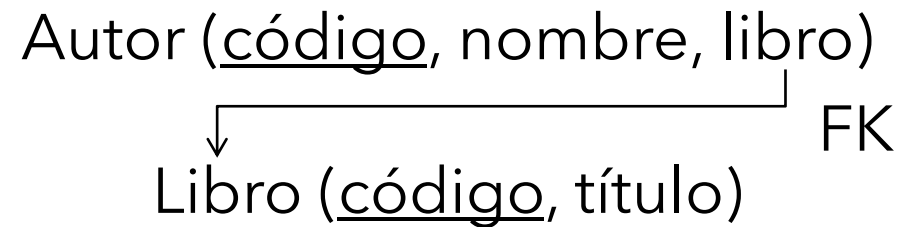


La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER,  
    FOREIGN KEY (libro) REFERENCES libro (codigo)  
);
```

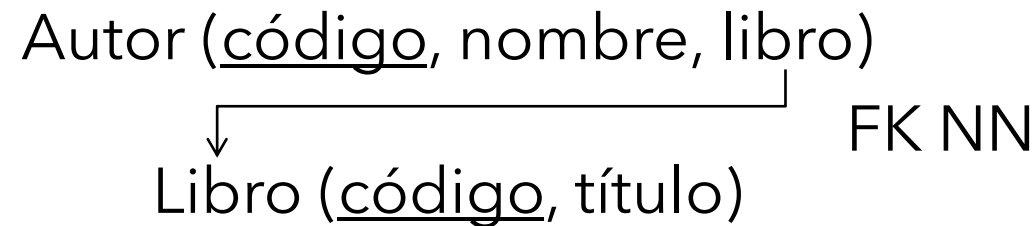


La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER NOT NULL,  
    FOREIGN KEY (libro) REFERENCES libro (codigo)  
);
```



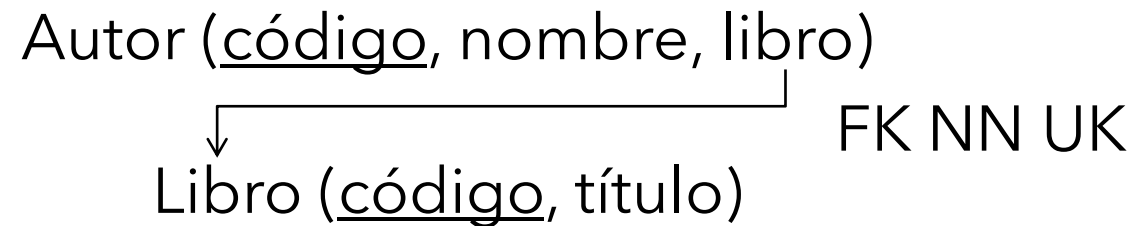
La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**



# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER NOT NULL UNIQUE,  
    FOREIGN KEY (libro) REFERENCES libro (codigo)  
);
```

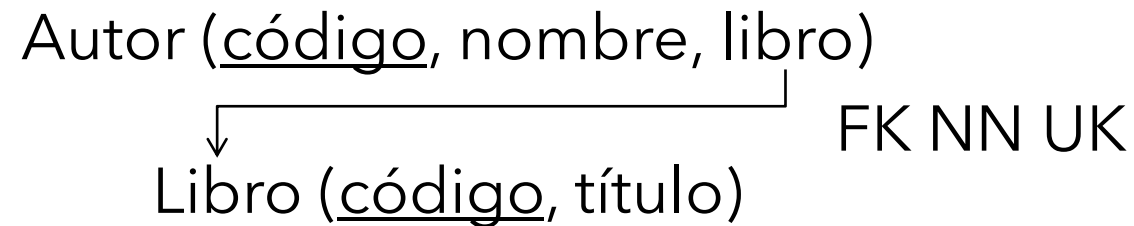


La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER NOT NULL,  
    UNIQUE (libro),  
    FOREIGN KEY (libro) REFERENCES libro (codigo)  
);
```

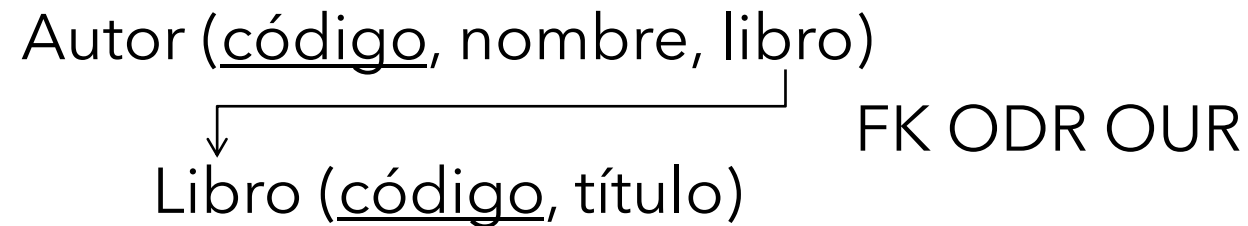


La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER,  
    FOREIGN KEY (libro) REFERENCES libro (codigo)  
);
```

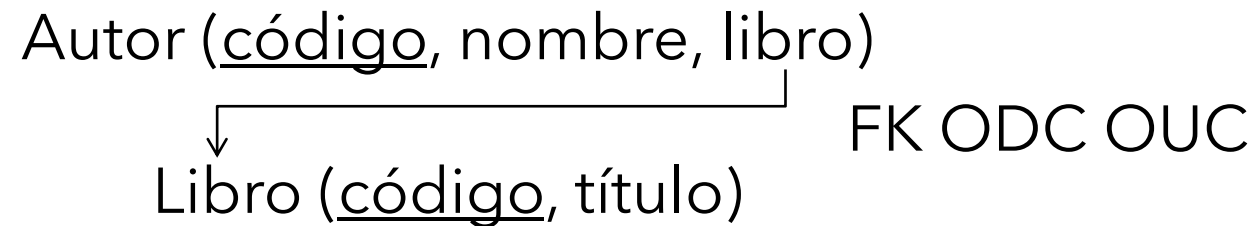


La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER,  
    FOREIGN KEY (libro) REFERENCES libro (codigo)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

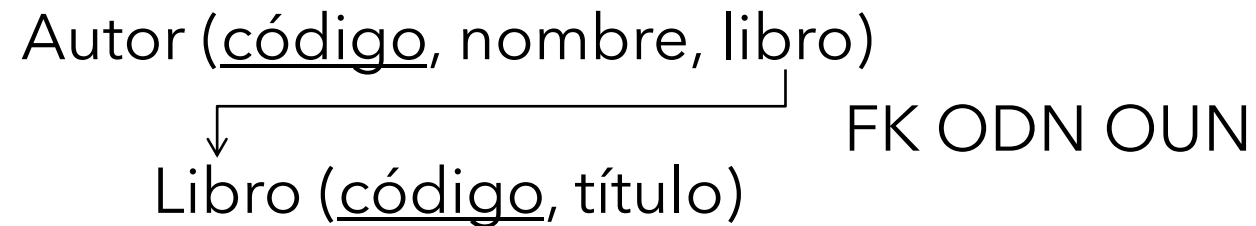


La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## FOREIGN KEY



```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY,  
    nombre VARCHAR(50),  
    libro INTEGER,  
    FOREIGN KEY (libro) REFERENCES libro (codigo)  
    ON DELETE SET NULL ON UPDATE SET NULL  
);
```



La restricción FOREIGN KEY solo se puede definir como restricción de tabla.

**Restricción de tabla**

# Data Definition Language

## CHECK

Son restricciones que establecen una condición que deben cumplir los datos de una columna.

- La condición se expresa entre paréntesis y debe devolver un resultado booleano (*true* o *false*).
- Podemos utilizar los operadores lógicos y relacionales:  
    >, <, >=, !=, <=, =, IS NULL, IS NOT NULL, AND, NOT, OR.
- Un campo puede tener varias restricciones CHECK.

# Data Definition Language

## CHECK

```
CREATE TABLE ingresos (  
    cod INT PRIMARY KEY,  
    concepto VARCHAR (40) NOT NULL,  
    importe INT,  
    CONSTRAINT ingresos_importe_ck  
        CHECK (importe > 0)  
);
```

# Data Definition Language

## CHECK

```
CREATE TABLE ingresos (  
    cod INT PRIMARY KEY,  
    concepto VARCHAR (40) NOT NULL,  
    importe INT CHECK (importe > 0)  
);
```



# Data Definition Language

## CHECK

```
CREATE TABLE ingresos (  
    cod INT PRIMARY KEY,  
    concepto VARCHAR (40) NOT NULL,  
    importe INT,  
    CONSTRAINT ingresos_importe_1_ck  
        CHECK (importe > 0),  
    CONSTRAINT ingresos_importe_2_ck  
        CHECK (importe < 8000)  
);
```

**Restricción de tabla**

# Data Definition Language

## CHECK

```
CREATE TABLE ingresos (  
    cod INT PRIMARY KEY,  
    concepto VARCHAR (40) NOT NULL,  
    importe INT,  
    CONSTRAINT ingresos_importe_ck  
        CHECK (importe > 0 AND importe < 8000)  
);
```

# Data Definition Language

## CHECK

```
CREATE TABLE ingresos (  
    cod INT PRIMARY KEY,  
    concepto VARCHAR (40) NOT NULL,  
    importe INT CHECK (importe > 0)  
                CHECK (importe < 8000)  
);
```

# Data Definition Language

## CHECK

```
CREATE TABLE ingresos (  
    cod INT PRIMARY KEY,  
    concepto VARCHAR (40) NOT NULL,  
    importe INT CHECK (importe > 0  
                        AND importe < 8000)  
);
```

# Data Definition Language

## **ALTER TABLE**

Sentencia que nos permite:

- Modificar el nombre de una tabla.
- Añadir, borrar o modificar las columnas.
- Añadir y eliminar restricciones de la tabla.

# Data Definition Language

## ALTER TABLE

Modificar el nombre de una tabla.

```
ALTER TABLE nombreActual RENAME nuevoNombre;
```

Ejemplo:

```
ALTER TABLE personal RENAME persona;
```



Cambiar el nombre de las tablas no afecta a las posibles claves ajenas que pudieran haber definidas sobre ellas.

# Data Definition Language

## ALTER TABLE

Añadir columnas a una tabla.

```
ALTER TABLE nombreTabla ADD COLUMN nombreColumna  
definicionColumna;
```

Ejemplo:

```
ALTER TABLE persona ADD COLUMN nombre VARCHAR(50) ;
```

# Data Definition Language

## ALTER TABLE

Eliminar columnas de una tabla.

```
ALTER TABLE nombreTabla DROP COLUMN nombreColumna;
```

Ejemplo:

```
ALTER TABLE persona DROP COLUMN nombre;
```



No podemos eliminar una columna si forma parte de una clave ajena. Para eliminarla primero tendremos que eliminar la restricción de clave ajena (ver diapositiva DROP CONSTRAINT).



# Data Definition Language

## ALTER TABLE

Renombrar columnas de una tabla.

```
ALTER TABLE nombreTabla RENAME COLUMN nombreColumna TO  
nuevoNombre;
```

Ejemplo:

```
ALTER TABLE persona RENAME COLUMN nombre TO nombre;
```



Cambiar el nombre de las columnas no afecta a las posibles claves ajenas que pudieran haber definidas sobre ellas.

# Data Definition Language

## ALTER TABLE

Modificar la definición de una columna.

```
ALTER TABLE nombreTabla CHANGE COLUMN nombreColumna  
nuevoNombre definicionColumna;
```

Ejemplo:

```
ALTER TABLE persona CHANGE COLUMN direccion direccion  
VARCHAR(100);
```



No podemos cambiar el tipo de una columna si forma parte de una clave ajena ya que se dejaría de cumplir la restricción de integridad referencial.

# Data Definition Language

## ALTER TABLE

Modificar la definición de una columna.

```
ALTER TABLE nombreTabla CHANGE COLUMN nombreColumna  
nuevoNombre definicionColumna;
```

Podemos utilizar esta sentencia para indicar las **restricciones de columna**:

```
ALTER TABLE persona CHANGE COLUMN direccion direccion  
VARCHAR(100) NOT NULL UNIQUE;
```



No podemos cambiar el tipo de una columna si forma parte de una clave ajena ya que se dejaría de cumplir la restricción de integridad referencial.

# Data Definition Language

## ALTER TABLE

Añadir una **restricción de tabla**.

```
ALTER TABLE nombreTabla ADD [CONSTRAINT] restriccion;
```

Ejemplos:

```
ALTER TABLE persona ADD PRIMARY KEY(id);
```

```
ALTER TABLE persona ADD CONSTRAINT PRIMARY KEY(id);
```

```
ALTER TABLE persona ADD CONSTRAINT persona_id_pk PRIMARY  
KEY(id);
```

# Data Definition Language

## ALTER TABLE

Eliminar una **restricción de tabla**.

```
ALTER TABLE nombreTabla DROP [CONSTRAINT]  
nombreRestriccion;
```

La sintaxis depende del tipo de restricción.

- En el caso de querer eliminar la clave primaria no es necesario indicar el nombre de la restricción:

```
ALTER TABLE persona DROP PRIMARY KEY;
```

# Data Definition Language

## **ALTER TABLE**

- En el caso de querer otro tipo de restricción de tabla será necesario indicar el nombre:

```
ALTER TABLE amigos DROP CONSTRAINT amigos_p1_fk;
```

# Data Definition Language

## AUTO\_INCREMENT


Genera automáticamente valores numéricos secuenciales cada vez que se inserta una fila.

Por ejemplo:

```
CREATE TABLE autor (  
    codigo INTEGER PRIMARY KEY AUTO_INCREMENT,  
    nombre VARCHAR(50)  
);
```

Al insertar varias filas:

```
INSERT INTO autor VALUES (DEFAULT, 'Juan');  
INSERT INTO autor VALUES (DEFAULT, 'Andrea');
```

 No os preocupéis por la sintaxis del INSERT ya que se estudiará en detalle en la siguientes unidades.

# Data Definition Language

## AUTO\_INCREMENT

Si consultamos la tabla:

```
SELECT * FROM autor;
```

Veremos que el código se ha asignado automáticamente:

	codigo	nombre
▶	1	Juan
	2	Andrea



No os preocupéis por la sintaxis del SELECT ya que se estudiará en detalle en la siguientes unidades.