al crear la tabla hemos de incidar el indice y el A-I (auto increment)


# MYSQL

Generador organigramas base datos
https://www.draw.io/

Manual online:
https://dev.mysql.com/ - Version Antigua
http://php.net/manual/es/book.mysqli.php - Version mejorada --> mysqli
ir a Guia Rapida/Interfaz dual: procedimental y orientada a objetos

# Introducción I

- Las bases de datos permiten almacenar de una forma estructurada y eficiente toda la información de un sitio WEB.

- De esta manera existe un histórico de datos.

- Ventajas sobre otros tipos de almacenamiento:
  - Se almacenan datos independientemente del lenguaje informático o programa que los usa: PHP, C++,…
  -Proporciona información actualizada
  -Facilita la realización de búsquedas
  -Disminuye el coste de mantenimiento
  -Implementa sistemas de control de acceso
  -Almacena usuarios, productos, ….

# Introducción II

A pesar de que existen otros Sistemas de Gestión de Bases de Datos Relacionales comerciales (**Oracle, Microsoft SQL Server…**), MySQL presenta una serie de ventajas:

- **Licencia GPL**, General Public License, que permite su uso gratuito en la mayoría de los casos.

- **Alta velocidad y rendimiento**, se trata de un sistema muy ligero que apenas consume recursos del equipo en el que se encuentra.

- **Conectividad con otras tecnologías**, se puede combinar con PHP para su uso en páginas web, pero también con aplicaciones C, Java…

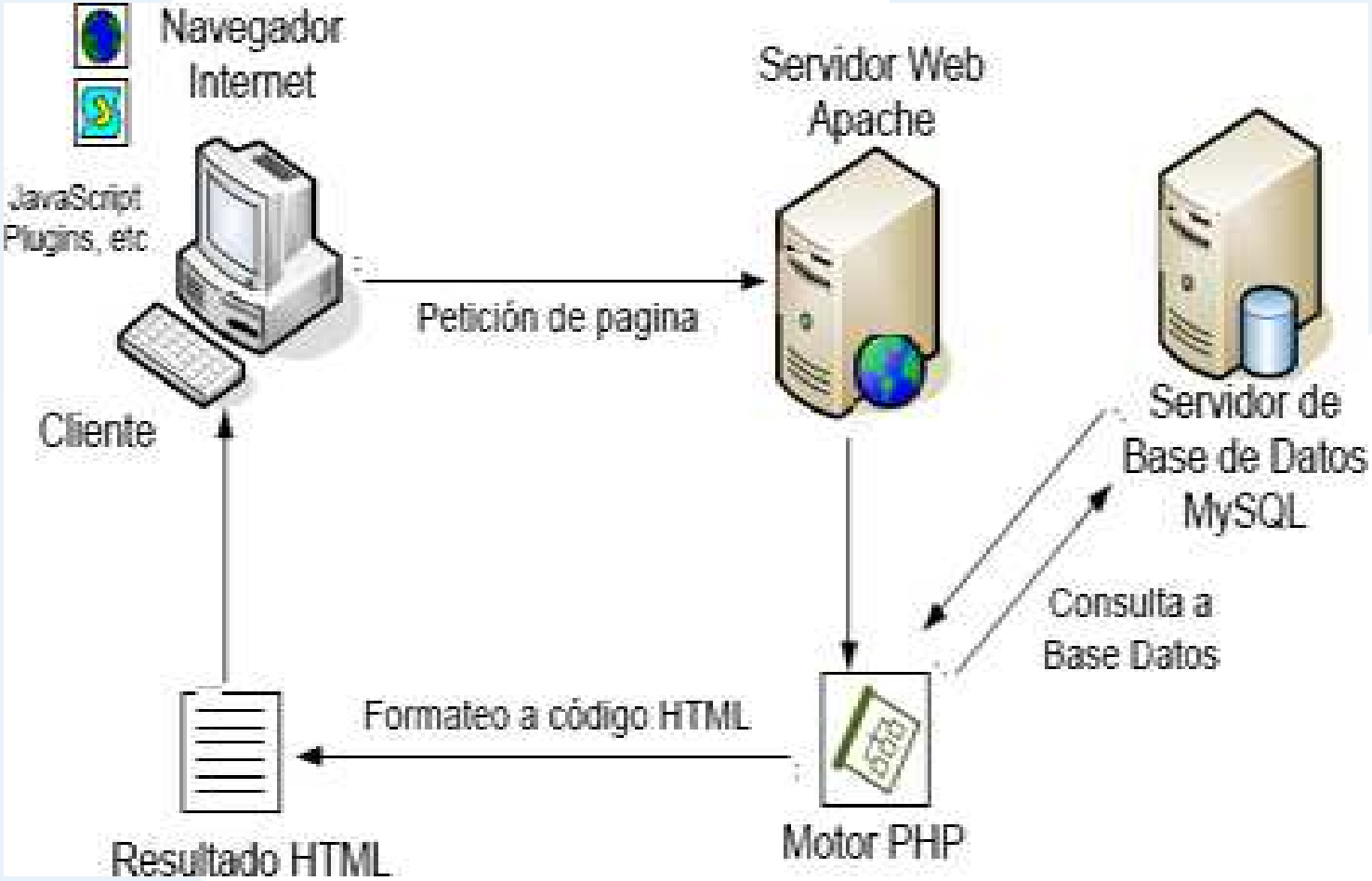
- **Uso muy extendido**, que permite encontrar muchísima documentación en la red sobre el mismo, así como tutoriales, solución a incidencias, etc.

# Introducción III

- Como se verá más adelante, MySQL junto con PHP proporcionan una gran cantidad de información mostrada por pantalla con un mínimo bloque de código.

Por ejemplo:
- Listados de productos.
- Listados de usuarios.

# Introducción IV



# Instalación MySQL

Podemos obtener el servidor MySQL directamente desde la página web oficial (requiere registro):

http://www.mysql.com

También es muy recomendable descargar desde la misma página el **MySQL Workbench**, aplicación que proporciona una interfaz gráfica para interactuar con la base de datos. De lo contrario, tendremos que acceder a la misma a través de la consola.

http://www.mysql.com/downloads/workbench/

Para su uso en el desarrollo de aplicaciones web, lo ideal es descargar un paquete del tipo **XAMPP**, que incluyen Apache, PHP y MySQL en una misma instalación.

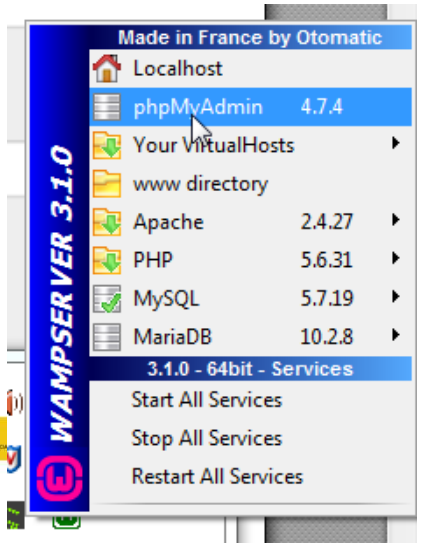
http://www.apachefriends.org/es/xampp.html

# Instalación: PHPMyAdmin

Si uso wampserver (y no tengo usuario dado de alta)

- PHPMyAdmin es una herramienta muy potente para l s bases de datos dentro de PHP.
- La mayoría de servidores web ya vienen con este programa incluido.
- Si no fuese así, lo podremos obtener en:

  http://www.phpmyadmin.net

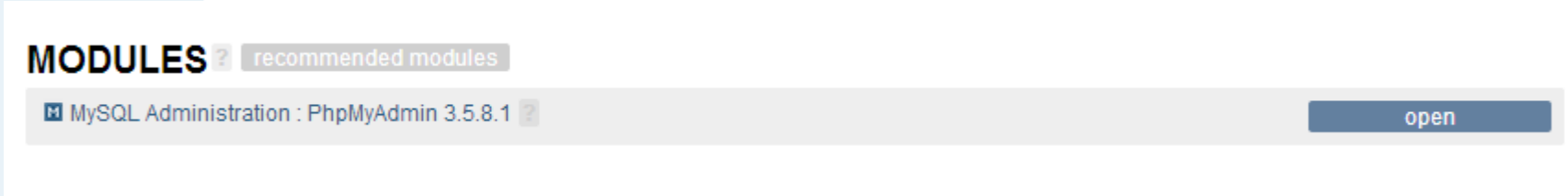



# Ejecución: PHPMyAdmin

- Con el servidor iniciado (EasyPHP) accedemos al navegador y vamos a:

http://localhost/home/

-En modules clicaremos en open para abrir el administrador, normalmente PHPMyAdmin:



# Ejecución: PHPMyAdmin

- Con el servidor iniciado (XAMPP, MAMP,…) accedemos al navegador y vamos a:

  http://localhost/phpmyadmin

# ESTRUCTURA DE UNA BASE DE DATOS

- Trataremos **bases de datos** relacionales que contienen:

  -**Tablas**: donde se almacenan los datos.

  -**Columnas** (atributos): propiedades de la relación.

  -**Filas**: cada uno de los registros que son contenidos en las tablas.

- Una **bases de datos** relacional significa que podremos establecer conexiones (relaciones) entre los datos que estén guardados en las tablas.

# ESTRUCTURA DE UNA BASE DE DATOS

- Por ejemplo:
  - Tabla **PERSONA**: contiene las características de una persona.
  - Columnas (atributos) de **PERSONA**.
    - Nombre, apellidos, edad.
  - Filas (registros) de **PERSONA**.
    - JUAN, MARTINEZ, 43.
    - MARIA, MONTE, 32.
    - MARTA, GONZALEZ, 12.

# ESTRUCTURA DE UNA BASE DE DATOS

- PROPIEDADES DE UNA TABLA:

  •Las columnas están ordenadas por orden de inserción.

  •Las filas están ordenadas por orden de inserción.

  •Cada **registro** debe tener un elemento característico diferente al resto que nos sirve para identificarlo.

  -P.E: en el caso anterior deberíamos añadir un **DNI**.

  - Este elemento se llama **CLAVE** o **IDENTIFICADOR**.

# ESTRUCTURA DE UNA BASE DE DATOS

- **CLAVES**:

  Una clave puede ser un atributo (p. e: DNI) o un conjunto de atributos (p.e: nombre y apellidos).

Tambien pueden ser claves compuestas usando varios campos

**•CLAVES PRIMARIA:**

- Identifica los registros.
- No puede ser nulo!

**•CLAVE FORÁNEA :**

-Relaciona un registro de una **TABLA A** con uno de una **TABLA B.**

**-** Este valor en **TABLA A** coincidirá con la clave primaria en **TABLA B**!

# ESTRUCTURA DE UNA BASE DE DATOS

## REPRESENTACIÓN DE UNA TABLA:

| PERSONA | |
|---|---|
| **DNI** | **CLAVE PRIMARIA** STRING |
| **NOMBRE** | STRING |
| **APELLIDOS** | STRING |
| **EDAD** | INT |

# ESTRUCTURA DE UNA BASE DE DATOS

## REPRESENTACIÓN DE UNA TABLA:

| COCHE | |
|---|---|
| **MATRICULA** | **CLAVE PRIMARIA**<br>STRING |
| **NUM_PUERTAS** | INT |

# ESTRUCTURA DE UNA BASE DE DATOS

**Ejercicio 1: Definir la tabla de vehículo**

- DNI DE PERSONA

| VEHÍCULO | |
|---|---|
| Matricula | CLAVE PRIMARIA<br>string |
| País | CLAVE PRIMARIA<br>string |
| Tipo | string |
| Puertas | int |
| Ruedas | int |
| Combustible | string |
| Marca | string |
| Modelo | string |
| Comentarios | text |
| DNI | CLAVE FORANEA<br>String |

# ESTRUCTURA DE UNA BASE DE DATOS

## RELACIÓN DE 2 TABLAS 1:1

| PERSONA | |
|---|---|
| DNI | CLAVE PRIMARIA STRING |
| NOMBRE | STRING |
| APELLIDOS | STRING |
| EDAD | INT |
| MATRICULA_COCHE | CLAVE FORÁNEA STRING |

| COCHE | |
|---|---|
| MATRICULA | CLAVE PRIMARIA STRING |
| NUM_PUERTAS | INT |
| DNI_PERSONA | CLAVE FORÁNEA STRING |

# ESTRUCTURA DE UNA BASE DE DATOS

- **TIPOS DE RELACIONES:**

-Se pueden tener relaciones de **1: 1**

Como en el ejemplo anterior:

- Una persona **SOLO** tendría un coche.
- Un coche **SOLO TIENE** un dueño( persona).
- Una persona puede **NO** tener un coche.
- Un coche puede **NO** tener dueño (persona).

# ESTRUCTURA DE UNA BASE DE DATOS

## RELACIÓN DE 2 TABLAS 1:1

| PERSONA | |
|---|---|
| DNI | CLAVE PRIMARIA<br>STRING |
| NOMBRE | STRING |
| APELLIDOS | STRING |
| EDAD | INT |
| CASADO_CON | CLAVE FORÁNEA<br>STRING |

# ESTRUCTURA DE UNA BASE DE DATOS

- **TIPOS DE RELACIONES:**

-Se pueden tener relaciones de **1: 1**

Como en el ejemplo anterior:

- Una persona **SOLO** esta casada con otra persona
- Una persona puede **NO** estar casada.

# ESTRUCTURA DE UNA BASE DE DATOS

## RELACIÓN DE 2 TABLAS 1:N

| PERSONA | |
|---|---|
| DNI | CLAVE PRIMARIA<br>STRING |
| NOMBRE | STRING |
| APELLIDOS | STRING |
| EDAD | INT |
| MATRICULA_<br>COCHE | CLAVE FORÁNEA<br>STRING |

| COCHE | |
|---|---|
| MATRICULA | CLAVE PRIMARIA<br>STRING |
| NUM_PUERTAS | INT |

# ESTRUCTURA DE UNA BASE DE DATOS

- **TIPOS DE RELACIONES:**

-Se pueden tener relaciones de **1: N**

Como en el ejemplo anterior:

- Una persona **SOLO** tendría un coche.
- Un coche **PUEDE TENER** muchos dueños( persona).
- Una persona puede **NO** tener un coche.
- Un coche puede **NO** tener dueño (persona).

# ESTRUCTURA DE UNA BASE DE DATOS

- **TIPOS DE RELACIONES:**

-Se pueden tener relaciones de **N: N**

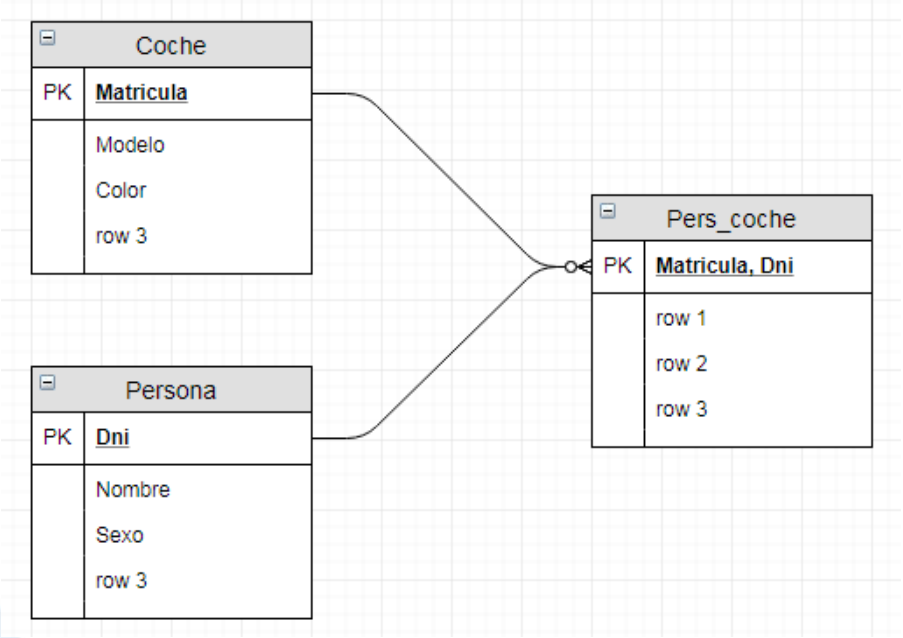
Esto requiere el uso de **una tercera tabla** para controlar esta relación.

- Una persona **PUEDE TENER MÁS DE UN** un coche.
- Un coche **PUEDE TENER MÁS DE UN** dueño( persona).
- Una persona puede **NO** tener un coche.
- Un coche puede **NO** tener dueño (persona).

# ESTRUCTURA DE UNA BASE DE DATOS

- **EJERCICIO:**

  - Representad la relación N:N entre coche y persona.




Multiples coches con multiples personas

# ESTRUCTURA DE UNA BASE DE DATOS

- **SIMPLIFICAR DATOS:**

-Una vez se tiene el conjunto de tablas definidas, en algunas ocasiones se pueden simplificar algunos de las columnas ya introducidas.

| PERSONA | |
|---|---|
| **DNI** | **CLAVE PRIMARIA** STRING |
| **NOMBRE** | STRING |
| **APELLIDOS** | STRING |
| **CIUDAD** | STRING |
| **COD_POST** | STRING |

| CIUDAD | |
|---|---|
| **COD_POST** | **CLAVE PRIMARIA** STRING |
| **NOMBRE** | STRING |

# ESTRUCTURA DE UNA BASE DE DATOS

- **SIMPLIFICAR DATOS:**

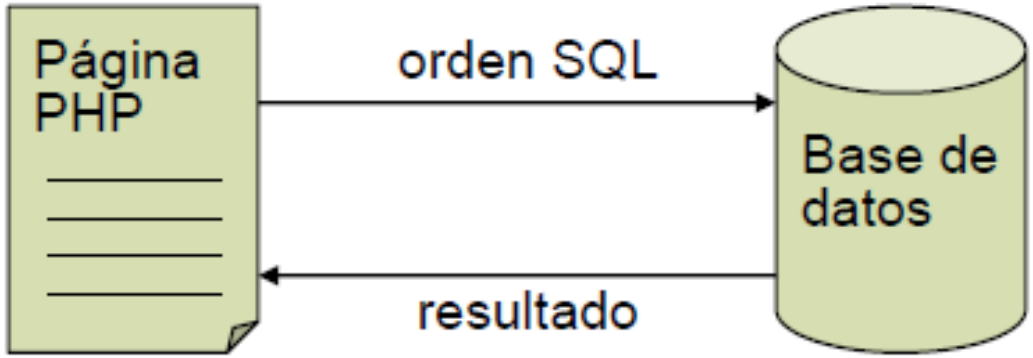
-En este caso, repetíamos datos en ambas tablas. Mediante la clave foránea de Código postal podremos acceder al nombre de la ciudad sin problemas!

| PERSONA | |
|---|---|
| **DNI** | **CLAVE PRIMARIA** STRING |
| **NOMBRE** | STRING |
| **APELLIDOS** | STRING |
| **COD_POST** | **CLAVE FORÁNEA** STRING |

| CIUDAD | |
|---|---|
| **COD_POST** | **CLAVE PRIMARIA** STRING |
| **NOMBRE** | STRING |

# SQL

- SQL (Structured Query Language) es el lenguaje que se utiliza para tratar con las bases de datos MySQL



# SQL

- TIPOS DE DATOS: existen muchos tipos de datos en MySQL, esto es debido a que se prevé que se almacenen muchos registros.

- Seleccionando el tipo de dato que más se adapte se puede ahorrar espacio en un futuro.

# SQL

- TIPOS NUMÉRICOS:
  - **TinyInt**: tipo entero. Del -128 al 127 con signo.
    Del 0 a 255 sin signo.
  - **Bit**: tipo entero. Del 0 al 1.
  - **Integer:** tipo entero. Del 2147483648 a 2147483647 con signo.
    Del 0 a 429.4967.295 sin signo.
  - **BigInt:** tipo entero. Del -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 con signo.
    Del 0 a 18.446.744.073.709.551.615. sin signo.

# SQL

- TIPOS NUMÉRICOS:
  - **Float**: tipo coma flotante. -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.

  - **Double**: tipo coma flotante. -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308

# SQL

- TIPOS NUMÉRICOS:

| Tipo de Campo | Tamaño de Almacenamiento |
|---|---|
| TINYINT | 1 byte |
| SMALLINT | 2 bytes |
| MEDIUMINT | 3 bytes |
| INT | 4 bytes |
| INTEGER | 4 bytes |
| BIGINT | 8 bytes |
| FLOAT(X) | 4 ú 8 bytes |
| FLOAT | 4 bytes |
| DOUBLE | 8 bytes |
| DOUBLE PRECISION | 8 bytes |
| REAL | 8 bytes |
| DECIMAL(M,D | M+2 bytes sí D > 0, M+1 bytes sí D = 0 |
| NUMERIC(M,D) | M+2 bytes if D > 0, M+1 bytes if D = 0 |

# SQL

- TIPOS DE FECHA:
  - **DateTime**: combinación de fecha más estándar. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-dia

  - **TimeStamp**: combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037.

  - **Time:** almacena una hora. El rango de horas va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos. El formato de almacenamiento es de 'HH:MM:SS'.

# SQL

- TIPOS DE FECHA:

| Tipo de Campo | Tamaño de Almacenamiento |
|---|---|
| DATE | 3 bytes |
| DATETIME | 8 bytes |
| TIMESTAMP | 4 bytes |
| TIME | 3 bytes |
| YEAR | 1 byte |

# SQL

- TIPOS DE CADENA:
  - **Char**: almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.
  - **VarChar**: almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.
  - **Text:** un texto con un máximo de 65535 caracteres.

# SQL

- TIPOS DE CADENA:

| Tipo de campo | Tamaño de Almacenamiento |
|---|---|
| CHAR(n) | n bytes |
| VARCHAR(n) | n +1 bytes |
| TINYBLOB, TINYTEXT | Longitud+1 bytes |
| BLOB, TEXT | Longitud +2 bytes |
| MEDIUMBLOB, MEDIUMTEXT | Longitud +3 bytes |
| LONGBLOB, LONGTEXT | Longitud +4 bytes |
| ENUM('value1','value2',...) | 1 ó dos bytes dependiendo del número de valores |
| SET('value1','value2',...) | 1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores |

# SQL

- Al igual que los lenguajes de programación, tiene una lista de operadores:

-**Aritméticos**

\+ - * /

-**Comparación**:

= <> <= < >= >

IS NULL IS NOT NULL

-**Lógicos**:

NOT (negación) AND OR

# SQL

- TIPOS MÁS UTILITZADOS:

**-Bit (1)**

**-Tinyint:** Tinyint(1) = Boolean

**-INT**

**-VarChar (255)**

**-Text**

**-Date**

**-Float**

# SQL

- Las instrucciones que se usaran una o pocas veces son de creación de tablas y modificación de su estructura CREATE, o borrar tablas TRUNCATE, DROP….

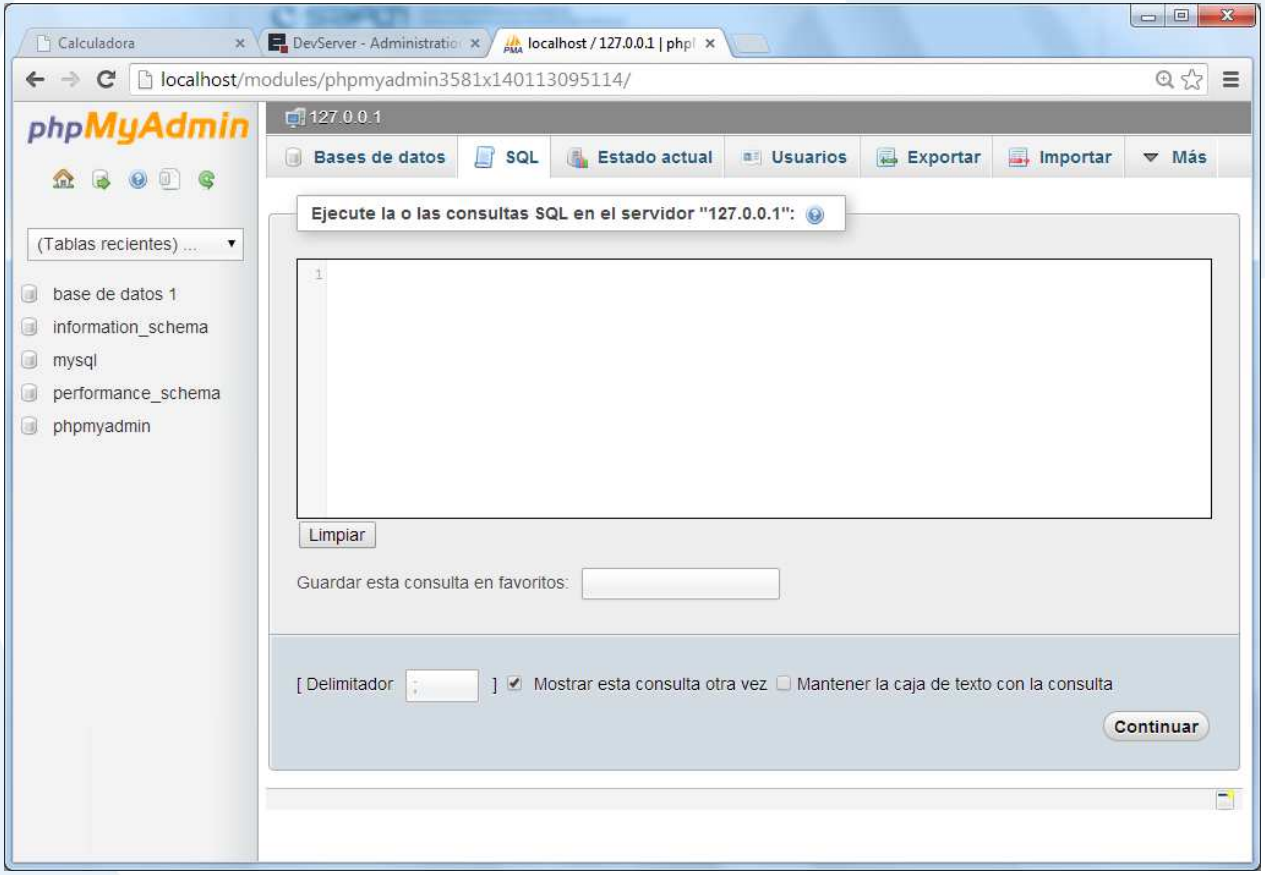
- Las instrucciones más habituales son SELECT, INSERT, UPDATE, DELETE que afectan a las filas o registros como tal.

TRUNCATE TABLE: Vacia el contenido de una tabla

DROP DATABASE: Borra la base de datos que contiene las tablas

# SQL

- Para probar todas estas instrucciones navegaremos hasta PHPMyAdmin y abriremos la opción SQL:



# SQL: DEFINIR DATOS

- **CREATE DATABASE [IF NOT EXISTS] nombre_base_datos;**

  - Nos crea una base de datos si no existe ya.

  CREATE DATABASE IF NOT EXISTS empresa;

# SQL: DEFINIR DATOS

- **DROP DATABASE [IF EXISTS] nombre_base_datos;**

  - Nos elimina base de datos si existe.

  DROP DATABASE IF EXISTS empresa;

# SQL: DEFINIR DATOS

- **CREATE TABLE [IF NOT EXISTS] nombretabla**

  **(nombre_campo1 tipo_datos (longitud) [características especiales] ,**
  **nombre_campo2 tipo_datos(longitud) [características especiales] );**

- Nos crea una tabla si no existe.

- Deberemos estar ya dentro de una base de datos.

- Las partes obligatorias son el nombre de la tabla y para cada columna su nombre y tipo de datos.

# SQL: DEFINIR DATOS

```
CREATE TABLE EMPLEATS(
        codi_emp CHAR(4),
        dni CHAR(9),
        nom VARCHAR(30),
        cog1 VARCHAR(30),
        cog2 VARCHAR(30),
        sou FLOAT(8),
        oficina CHAR(4)
);
```

# SQL: DEFINIR DATOS

- Para definir una clave primaria hace falta usar **primary key**.

- Si esta formada por una única columna puede especificarse así directamente en la definición de la tabla:

  **<columna> <tipo_dato> PRIMARY KEY**

  PRIMARY KEY ocupa el espacio de características especiales antes mostrado.

- Si esta formada por dos o más columnas debe especificarse al final de la creación de la tabla:

  **primary key (col1,col2,...)**

# SQL: DEFINIR DATOS

```
CREATE TABLE EMPLEATS(
        codi_emp CHAR(4) PRIMARY KEY,
        dni CHAR(9),
        nom VARCHAR(30),
        cog1 VARCHAR(30),
        cog2 VARCHAR(30),
        sou FLOAT(8),
        oficina CHAR(4)
);
```

# SQL: DEFINIR DATOS

```
CREATE TABLE EMPLEATS(
        codi_emp CHAR(4),
        dni CHAR(9),
        nom VARCHAR(30),
        cog1 VARCHAR(30),
        cog2 VARCHAR(30),
        sou FLOAT(8),
        oficina CHAR(4),
        PRIMARY KEY(nom, cog1, cog2)

);
```

# SQL: DEFINIR DATOS

- Otras características especiales:

  -**NOT NULL** si el valor en esa posición no puede ser nulo.

  <columna> <tipo_dato> NOT NULL

  **-UNIQUE** si el valor en esa posición debe ser único y no puede repetirse. Si el valor es una **PRIMARY KEY** no hace falta indicar por **UNIQUE** ya que tendrá esta propiedad por defecto.

  <columna> <tipo_dato> UNIQUE

# SQL: DEFINIR DATOS

- Otras características especiales:

  -**AUTO-INCREMENT:** incrementara el valor para cada registro añadido de manera automática. Se le puede asignar el valor por el cual va a empezar al final de la creación de la tabla.

```
CREATE TABLE IF NOT EXISTS `cliente1` (
`id_cliente` int(11) NOT NULL AUTO_INCREMENT,
`nombre` varchar(255) NOT NULL,
PRIMARY KEY (`id_cliente`)
)  AUTO_INCREMENT=1 ;
```

# SQL: DEFINIR DATOS

## CREACIÓN DE INDICES

**Ventajas**:

-Nos permiten acceder de manera más rápida a la información.

**Desventajas**:

-Nos ralentiza un poco la inserción y modificación de datos (insert, update, delete).

-En tablas pequeñas no mejoran mucho el tiempo de respuesta.

Por ello se debe decidir con calma que campos serán **índices**. Como mínimo, si la tabla tiene una clave foránea, esta será un índice.

# SQL: DEFINIR DATOS

## CREACIÓN DE INDICES

CREATE INDEX <nombre_indice> ON <nombre_tabla> ( nom_columna_indice [ASC | DESC] , ….);

-Si no se especifica el criterio de ordenación, se ordena de forma ascendente.

# SQL: DEFINIR DATOS

## ELIMINACIÓN DE INDICES

DROP INDEX <nombre_indice> ON <nombre_tabla>;

# SQL: DEFINIR DATOS

**INTEGRIDAD REFERENCIAL** (soportada a partir de MySQL 4.0).

Esta referencia nos ayuda a tratar con **CLAVES FORANEAS ENTRE TABLAS.**

Para usar este tipo de referencias las tablas deben crearse como ENGINE=INNODB ;

# SQL: DEFINIR DATOS

## INTEGRIDAD REFERENCIAL

- En la tabla con la clave foranea se tendrá que crear esta propiedad con:

  **FOREIGN KEY(campo_fk) REFERENCES nombre_tabla (nombre_campo)**

- Al igual que con claves foráneas normales, los campos deben ser del mismo tipo.Mismo tipo y tamaño

# SQL: DEFINIR DATOS

## INTEGRIDAD REFERENCIAL

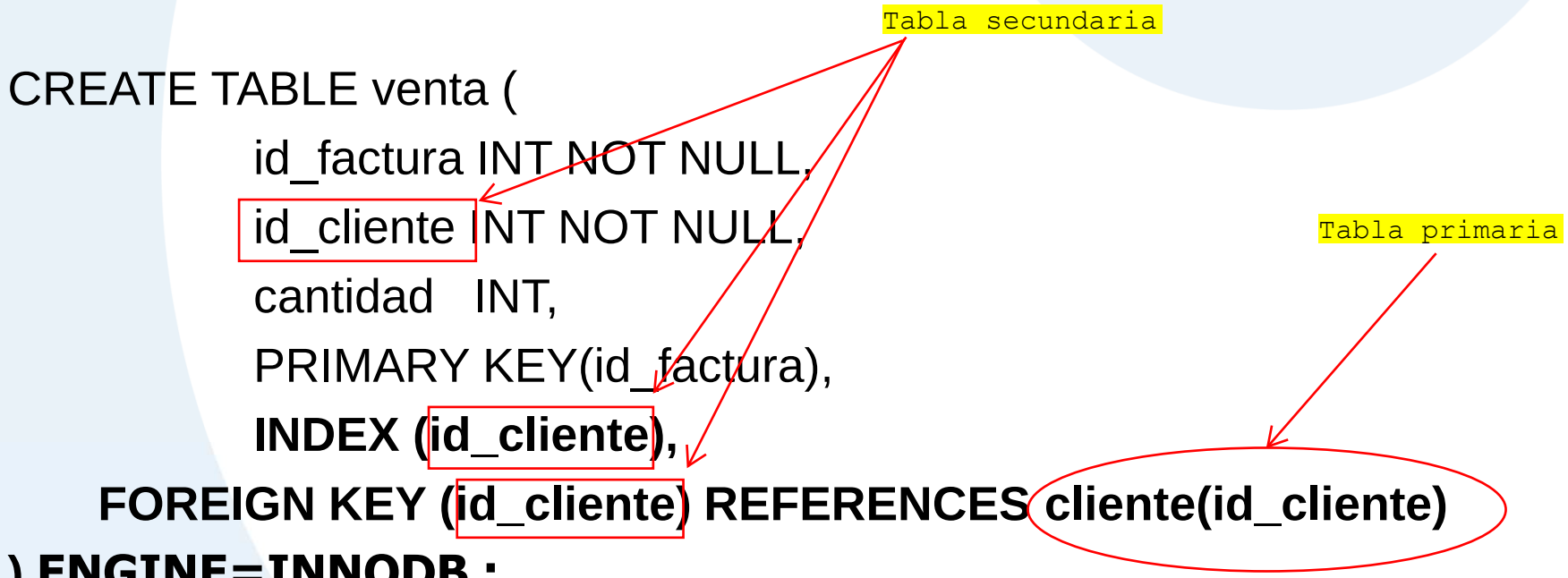
Creamos una tabla cliente con clave primaria cliente INT

```
CREATE TABLE cliente(
        id_cliente INT NOT NULL,
        nombre VARCHAR(30),
        PRIMARY KEY (id_cliente)
) ENGINE=INNODB ;
```

# SQL: DEFINIR DATOS

## INTEGRIDAD REFERENCIAL

Creamos una tabla venta que esta relacionada con cliente mediante un campo id_cliente.



# SQL: DEFINIR DATOS

## INTEGRIDAD REFERENCIAL

- En el caso anterior, la segunda tabla (venta) debe también tener como **índice la clave foránea.**

- **A pesar de ello, podemos crear claves foráneas a nivel lógico, pero en tal caso algunas acciones se deberán controlar a nivel de código.**

# SQL: DEFINIR DATOS

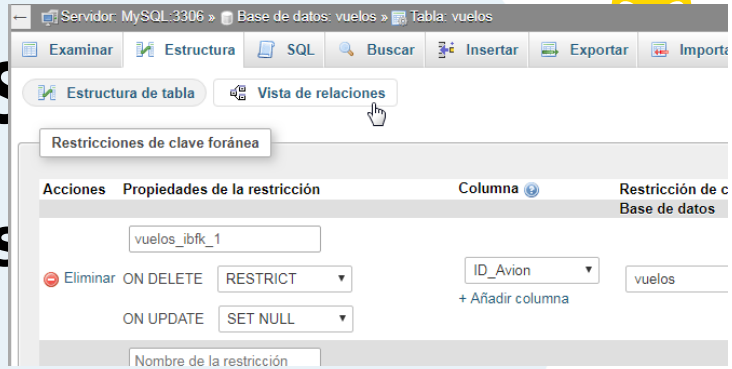
## INTEGRIDAD REFERENCIAL

Existen 4 alternativas para las acciones que se llevarán a cabo cuando se intente borrar o modificar el valor de la clave primaria asociada:

- **CASCADE**: Al cambiar el valor de la clave primaria automáticamente se actualiza el valor de la clave foránea asociada. Si borramos una clave primaria, entonces se eliminan todos los registros con claves ajenas que hagan referencia a la clave primaria que se ha borrado.

Si modificamos: Se modifica
Si borramos: Se borra

# SQL: DEFINIR DATOS




- **Creando CLAVES FORANEAS Y RELACIONES**

- **SET NULL**: Si se cambia o borra el valor de una clave primaria, entonces las claves foráneas cambiarán su valor a NULL.

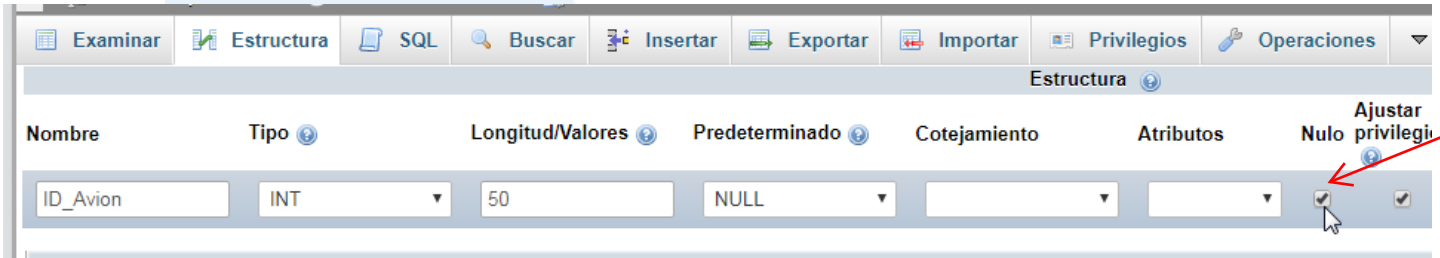
- **RESTRICT**: No se permite la eliminación o actualización de las claves primarias que tengan claves foráneas que le hagan referencia.

Si modificamos: No deja modificar la primera base
Si borramos: No deja borrar la primera base

- **NO ACTIONS**: Si se modifica o elimina una clave primaria las claves ajenas no sufren ninguna modificación.

Si modificamos: No afecta la tabla secundaria
Si borramos: No afecta la tabla secundaria

Definir Set Null



# SQL: DEFINIR DATOS

- **Creando CLAVES FORANEAS Y RELACIONES**

```
CREATE TABLE venta (
        id_factura INT NOT NULL,
        id_cliente INT NOT NULL,
        cantidad   INT,
        PRIMARY KEY(id_factura),
        INDEX (id_cliente),
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
) ENGINE=INNODB ;
```

# SQL: DEFINIR DATOS

- ALTER TABLE nombre_tabla
  <modificación_de_la_tabla>;

  - Nos modifica la tabla que le pasamos.

Existen varios tipos de modificaciones que se pueden dar en una tabla.

# SQL: DEFINIR DATOS

Tipos de modificaciones de una tabla:

1- Para añadir una columna:

add <nombre_columna><definición_de_la_columna>

***Alter table DEPT add any int(2);***

2-Para eliminar una columna

drop column <nom_columna>  o  drop <nom_columna>

***Alter table DEPT drop column any;***

# SQL: DEFINIR DATOS

Tipos de modificaciones de una tabla:

3- Para modificar la estructura de una columna. No hace falta añadir las características antiguas: Modifica caracteristicas: tipo, longitud, not null,...

modify <nom_col><nuevas_características> [FIRST]|[AFTER|BEFORE NombreColumna]

***Alter table DEPT modify any int(4) not null;***

4- Para modificar la estructura o canviar el nombre de una columna:

Change <nombre_vieja_columna> <nombre_nueva_columna> <características>

Columna Existente

***Alter table DEPT change any any_ent int(4) not null;***

Columna Nueva

# SQL: DEFINIR DATOS

Tipos de modificaciones de una tabla:

5- Añadir o eliminar el valor por defecto de una columna:

**alter column <nom_col> SET DEFAULT <valor_defecto>**
**alter column <nom_col> DROP DEFAULT**

6- Para añadir restricciones (primary key, foreign key, unique):

Add constraint <nombre_restricción> <restricción>

***Alter table DEPT add constraint DEPT_UNICO unique(any);***

Tabla

ID nuevo

Alter table **PERSONA** add **index(Id_coche)**;
Alter table **PERSONA** add constraint **fk_person.coche** foreing key (**id_coche**) reference **coche(id)**

Tabla secundaria

Tabla secundaria

Tabla primaria

# SQL: DEFINIR DATOS

Tipos de modificaciones de una tabla:

7- Eliminar restricciones

**drop primary key**
**drop index <nombre_restricción>**
**drop foreign key <nombre_restricción>**

**Alter table DEPT drop index DEPT_UNICO;**

# SQL: DEFINIR DATOS

Tipos de modificaciones de una tabla:

8- Para inicializar un campo autoincrementable:

AUTO_INCREMENT = x

**Alter table DEPT AUTO_INCREMENT = 0;**

# SQL: DEFINIR DATOS

- Por suerte, todas las operaciones SQL vistas hasta ahora no se realizan de manera común:

  P.E.: No es habitual generar un código que nos cree una base de datos.

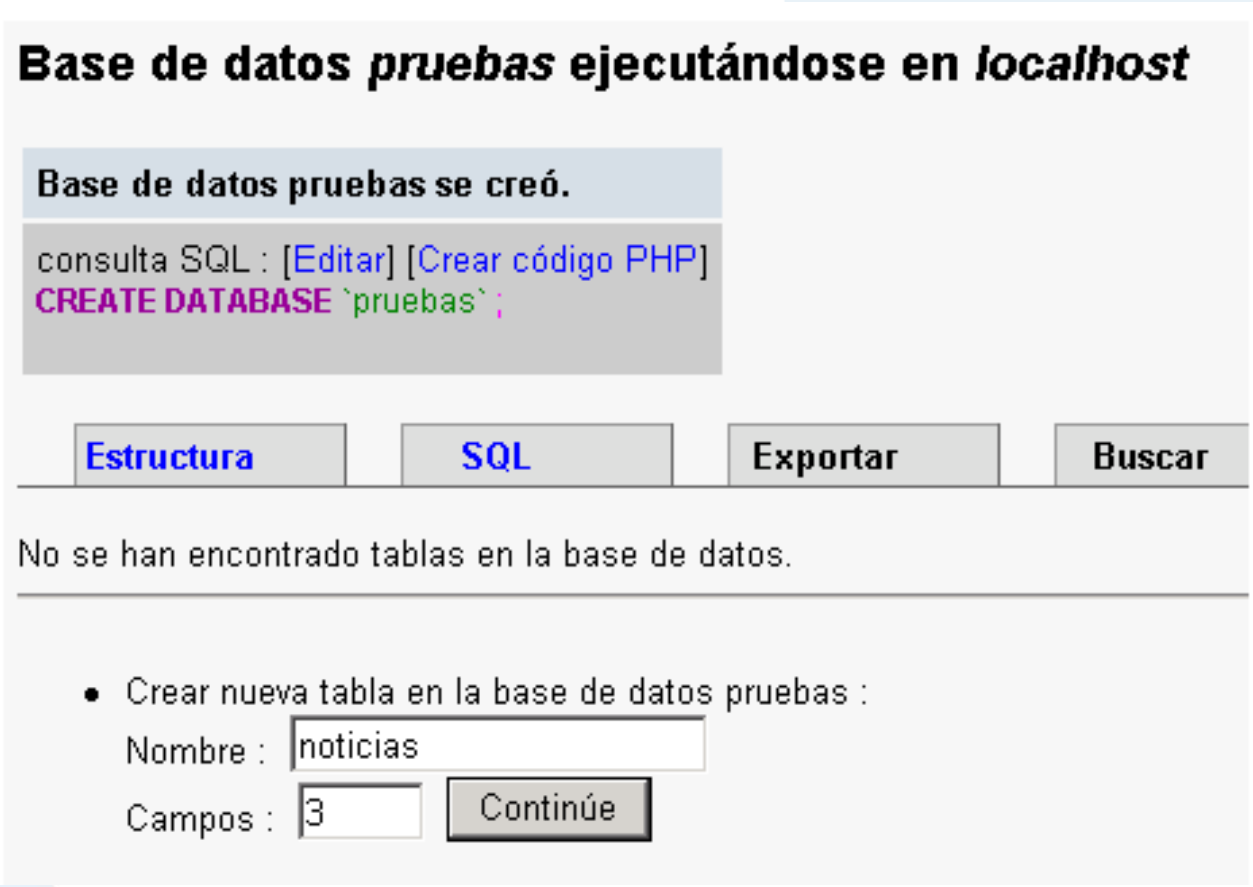
  -Para facilitar estas tareas, que son más de gestión de la estructura de la base de datos, se nos proporcionan herramientas útiles como **PHPMyAdmin.**

# PHPMyAdmin: estructura de la bbdd

- Creando una base de datos con PHPMyAdmin

# PHPMyAdmin: estructura de la bbdd

- Creando tablas con PHPMyAdmin: previa selección a la base de datos.

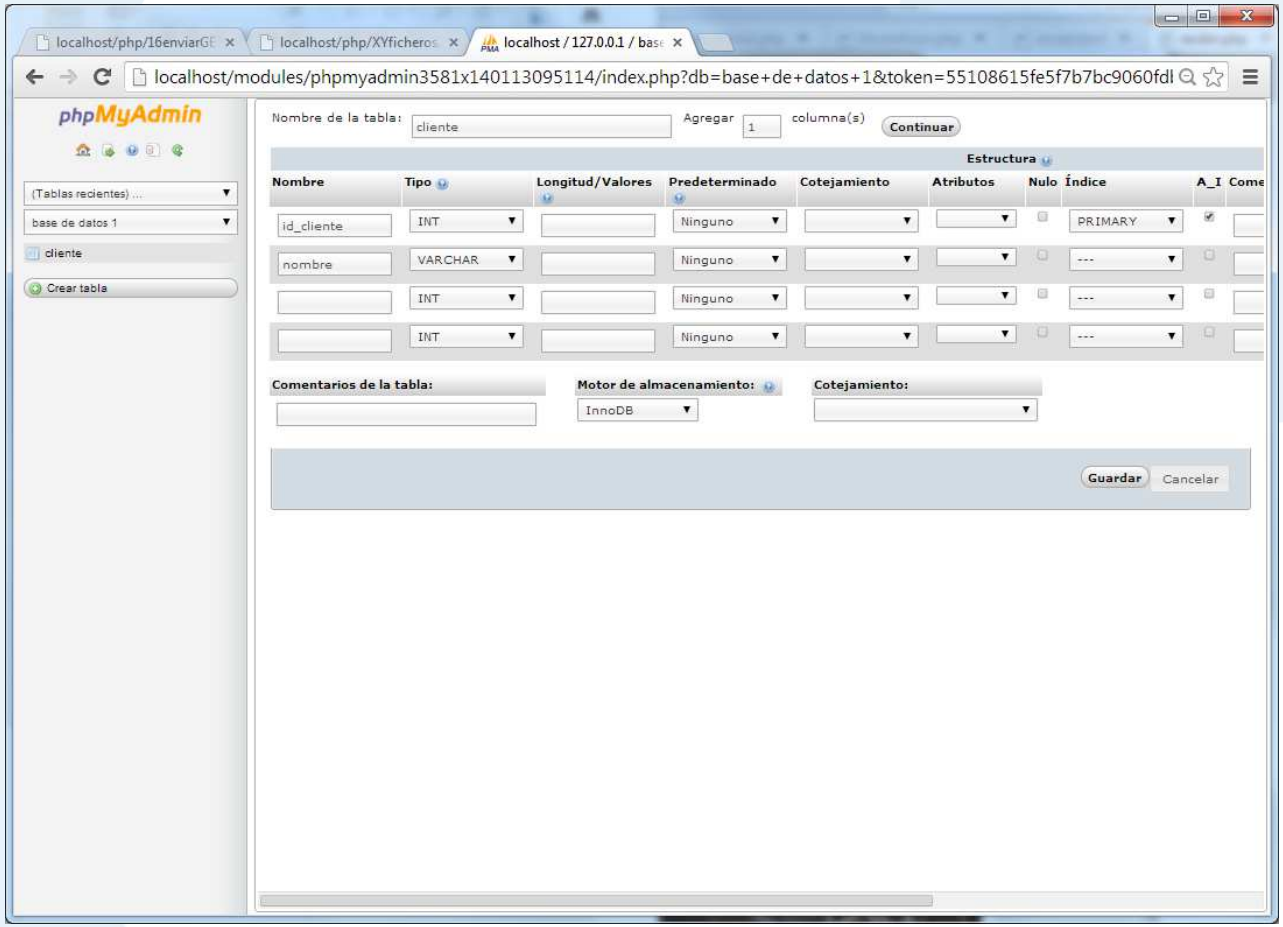


# PHPMyAdmin: estructura de la bbdd

- Creando campos y modificando con PHPMyAdmin

| Campo | Tipo | Longitud | Extra | Primaria |
|---|---|---|---|---|
| id | INT | 11 | auto_increment | Sí |
| titulo | VARCHAR | 250 | | |
| texto | LONG TEXT | | | |

# PHPMyAdmin: estructura de la bbdd



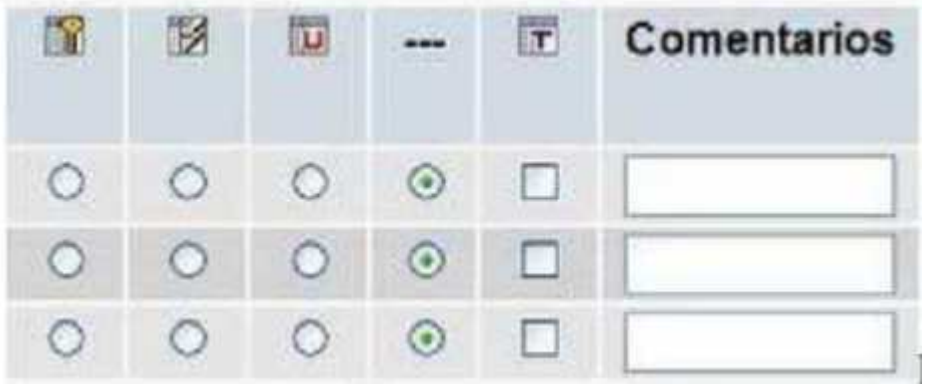
# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES PRIMARIAS y otras características**

- El primer icono hace referencia a la clave primaria de la tabla. Se puede seleccionar más de un campo.

- El segundo icono indica si el campo es o no un índice.

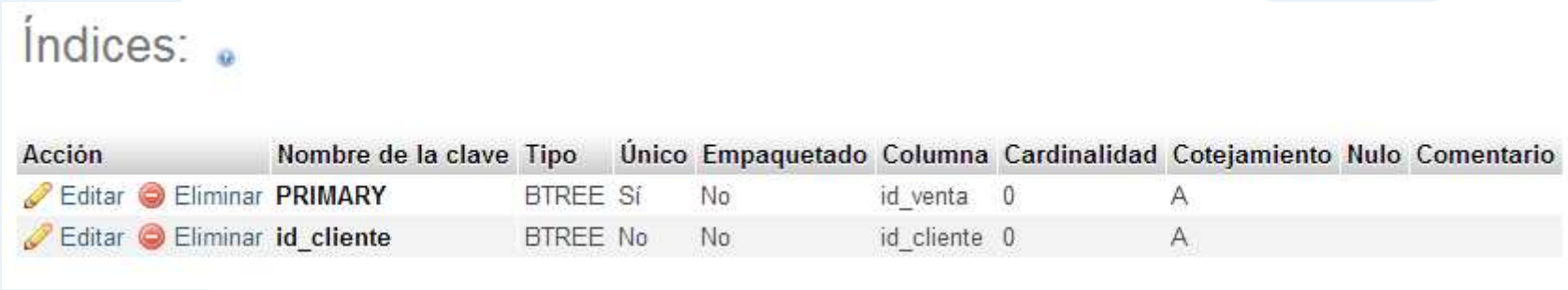
- El tercer icono indica si el campo permite valores duplicados (UNIQUE)



# PHPMyAdmin: estructura de la bbdd

- **Creando o eliminando CLAVES E ÍNDICES**

- En la tabla que queramos gestionar estas claves o índices veremos un subapartado correspondiente.



Índices:

| Acción | Nombre de la clave | Tipo | Único | Empaquetado | Columna | Cardinalidad | Cotejamiento | Nulo | Comentario |
|---|---|---|---|---|---|---|---|---|---|
| Editar Eliminar | PRIMARY | BTREE | Sí | No | id_venta | 0 | A | | |
| Editar Eliminar | id_cliente | BTREE | No | No | id_cliente | 0 | A | | |

# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS**

- A la hora de crear relaciones debemos tener en cuenta que los campos que van a relacionarse deben tener el mismo tipo.

- Los campos que queramos que sean **clave foránea** han de ser índices, pueden formar parte de un índice de varios campos o de una clave de varios campos.

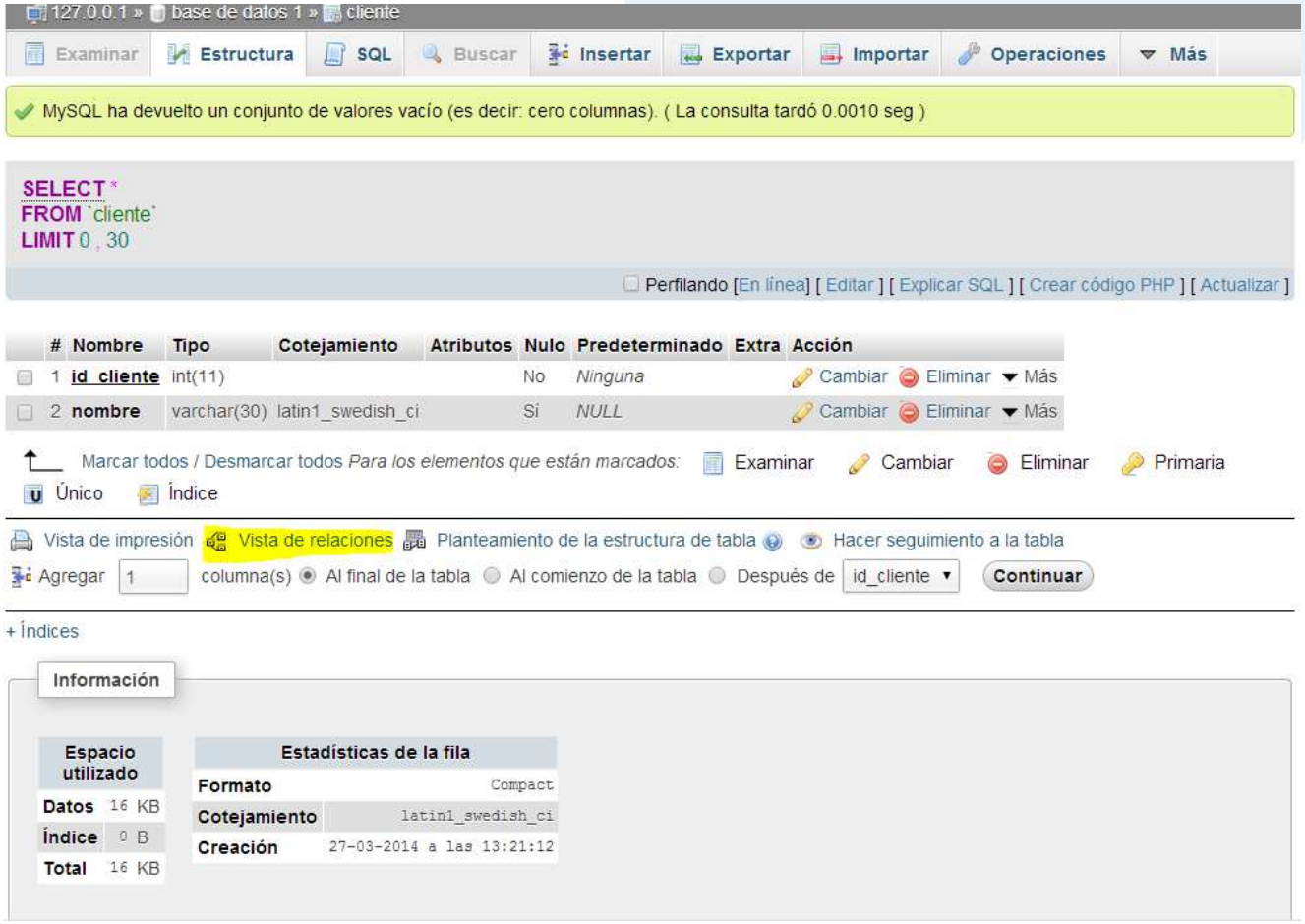
# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**

- Para crear este tipo de relaciones se tiene que seleccionar la tabla que contenga este tipo de clave y seleccionar "VISTA DE RELACIONES":

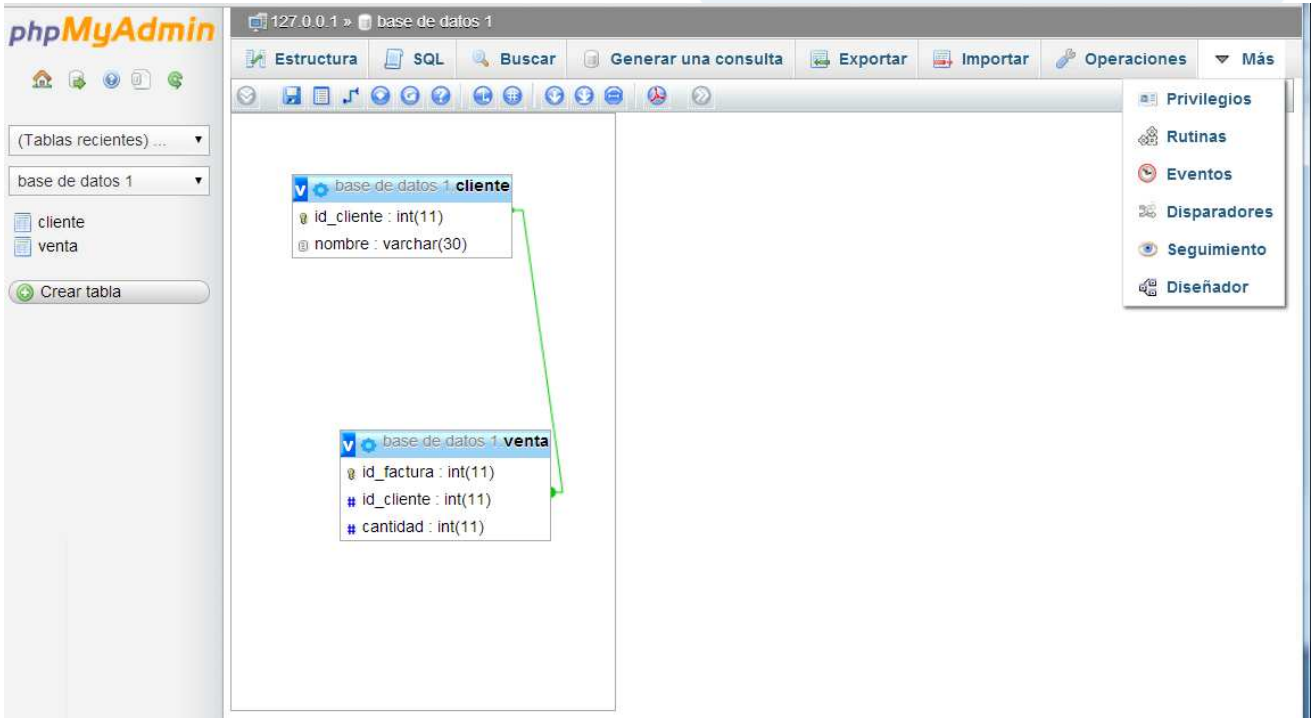
# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**



# PHPMyAdmin: DIAGRAMAS

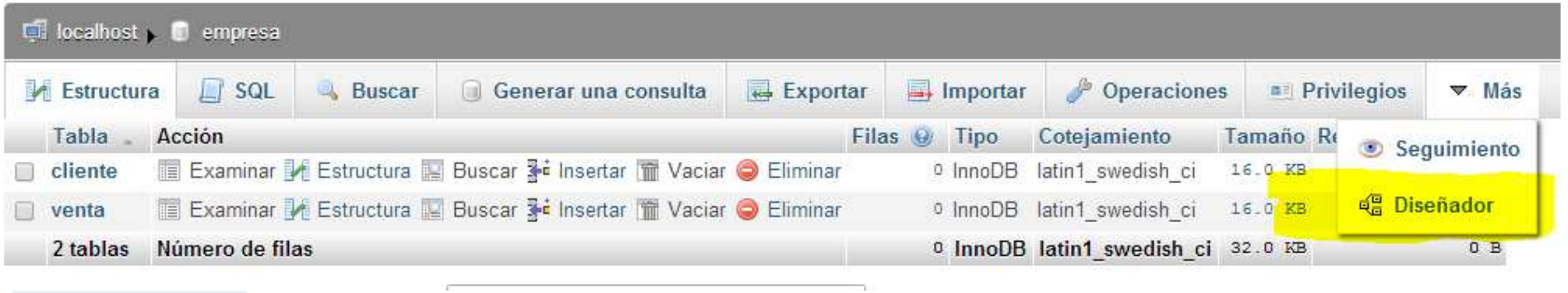
- A partir de PhpMyAdmin 3.5 se nos muestra una opción muy interesante a la hora de trabar con muchas tablas y poder consultar de un vistazo campos, claves primarias y claves foráneas, la opción **DISEÑADOR**.



# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**

- Podemos crear relaciones con el **DISEÑADOR** en la base de datos**.**




-De esta forma también se mantienen los mismos requisitos que anteriormente, es decir: Los campos que queramos que sean **clave foránea** han de ser índices, pueden formar parte de un índice de varios campos o de una clave de varios campos**.**

# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**

- Veremos un esquema muy visual de nuestras tablas.

- Primero de todo se debe seleccionar la herramienta de relación:

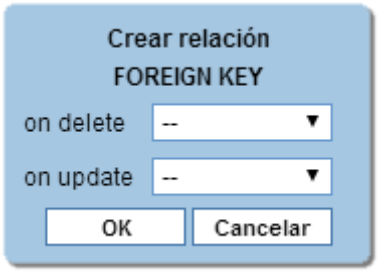
-Se nos indica que seleccionemos la **clave referenciada**. Esto es la clave primaria de la primera tabla.

- Se nos indica que seleccionemos la **clave foránea** de la segunda tabla.

# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**

- Una vez finalizada una relación, en el diseñador podremos ver esta ventana:




- Si aun no tenemos claro que acciones podremos tomar en este apartado por ahora podremos pulsar OK.

# PHPMyAdmin: estructura de la bbdd

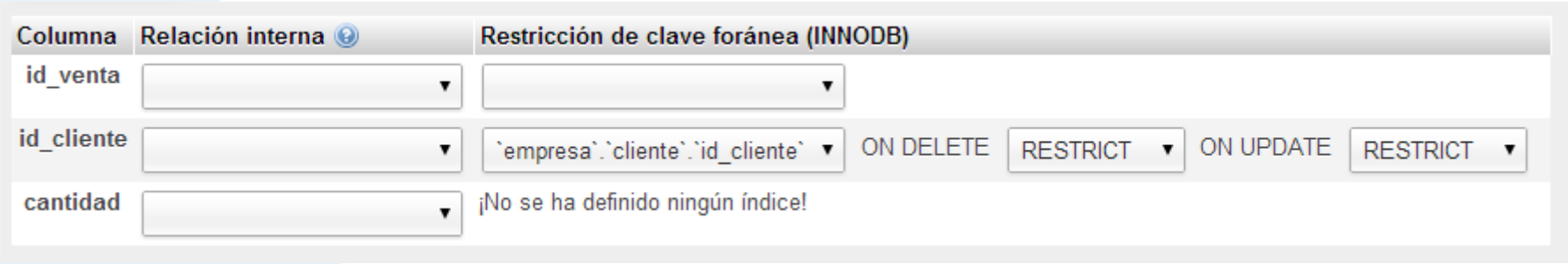
- **Creando CLAVES FORANEAS Y RELACIONES**

# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**

Además, se pueden escoger las acciones que se llevarán a cabo cuando se intente borrar o modificar el valor de la clave primaria asociada.

- Si vamos al apartado **Vista de relaciones** de la tabla con las **claves foráneas** podremos seleccionar acciones en caso de eliminar o actualizar la clave primaria:



# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**

Existen 4 alternativas:

- **CASCADE**: Al cambiar el valor de la clave primaria automáticamente se actualiza el valor de la clave foránea asociada. Si borramos una clave primaria, entonces se eliminan todos los registros con claves ajenas que hagan referencia a la clave primaria que se ha borrado.

# PHPMyAdmin: estructura de la bbdd

- **Creando CLAVES FORANEAS Y RELACIONES**

- **SET NULL**: Si se cambia o borra el valor de una clave primaria, entonces las claves foráneas cambiarán su valor a NULL.

- **RESTRICT**: No se permite la eliminación o actualización de las claves primarias que tengan claves foráneas que le hagan referencia.

- **NO ACTIONS**: Si se modifica o elimina una clave primaria las claves ajenas no sufren ninguna modificación.

# PHPMyAdmin: estructura de la bbdd

- Borrando tablas: