



# PHP & MySQL

*Anexo 7.- MySQL*



DISTRIBUIDO POR:

**CENTRO DE INFORMÁTICA PROFESIONAL S.L.**

C/ URGELL, 100  
08011 BARCELONA  
TFNO: 93 426 50 87

C/ RAFAELA YBARRA, 10  
48014 BILBAO  
TFNO: 94 448 31 33

[www.cipsa.net](http://www.cipsa.net)

**RESERVADOS TODOS LOS DERECHOS. QUEDA PROHIBIDO TODO TIPO DE REPRODUCCIÓN TOTAL O PARCIAL DE ESTE MANUAL, SIN PREVIO CONSENTIMIENTO POR EL ESCRITOR DEL EDITOR**

## ***Bases de Datos***

### **Fundamentos**

El almacenamiento y recuperación de datos desde un fichero es útil para la generación de copias de seguridad o el almacenamiento de ficheros de configuración, contenidos, imágenes... etc. Sin embargo, en la mayoría de las aplicaciones es necesario el almacenamiento de datos de diferentes tipos y que sean accesibles rápidamente.

Supóngase como ejemplo una aplicación web para un videoclub. Una aplicación así necesita almacenar las películas disponibles, los socios registrados, y los alquileres pendientes. Por otro lado; la aplicación debe poder acceder a los datos de manera relacionada respondiendo a consultas tales como: ¿qué películas no están alquiladas?, ¿qué películas están alquiladas?, ¿Quién las ha alquilado?, ¿Cuándo se realizó el alquiler de una determinada película?... etc. Resultaría difícil y poco eficiente almacenar esta información y realizar las consultas necesarias empleando ficheros. Para estos casos es necesario el uso de bases de datos.

Las bases de datos almacenan grandes cantidades de información de diferentes tipos permitiendo búsquedas complejas de forma rápida y eficaz.

### **Bases de datos relacionales.**

Las bases de datos relacionales almacenan los datos empleando un *modelo de entidad-relación* que estructura la información en **entidades** y **relaciones**.

Las *entidades* representan conjuntos de datos de un determinado tipo: *Personas*, *Ciudades*, *Vehículos*..., etc.

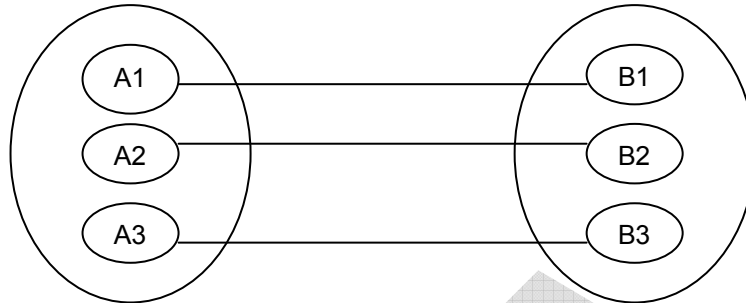
CLIENTE

Las *relaciones* representan una asociación entre dos entidades. Estas pueden clasificarse de las siguientes formas:

- *Por número*: Se clasifican las relaciones en función de la cantidad de elementos de una entidad que se relacionan con los de otra distinta.
- *Por grado*: Se clasifican las relaciones en función del número de entidades que se relacionan.

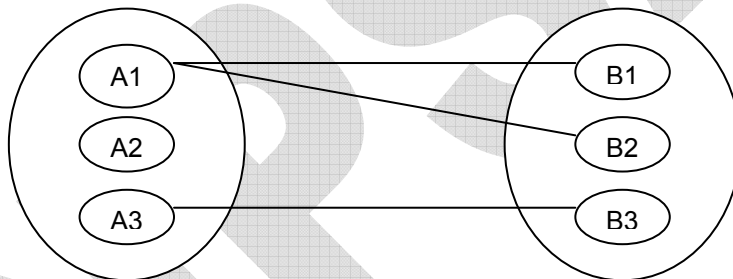
## Clasificación de relaciones por número

- **Relaciones 1:1** → Cada elemento de la entidad A se relaciona con un elemento de otra entidad B, y viceversa.



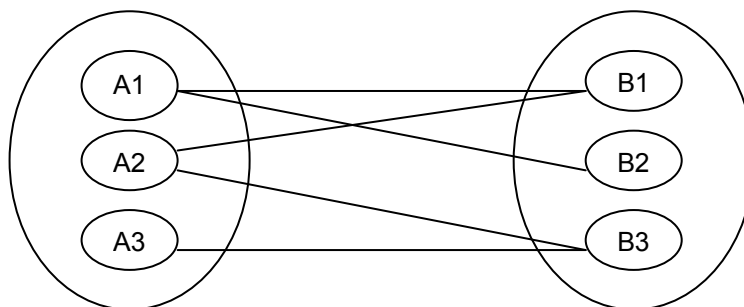
**Por ejemplo:** PERSONAS – RECETA ELECTRÓNICA. Una persona tiene una receta electrónica a su nombre, y cada receta electrónica pertenece a una persona.

- **Relaciones 1:n** → Cada elemento de una entidad A se relaciona 1 o varios elementos de la entidad B. Sin embargo; cada elemento de la entidad B únicamente se relaciona con un elemento de la entidad A.



**Por ejemplo:** DEPARTAMENTO – EMPLEADO. Un departamento tiene varios empleados, pero cada empleado pertenece a un único departamento.

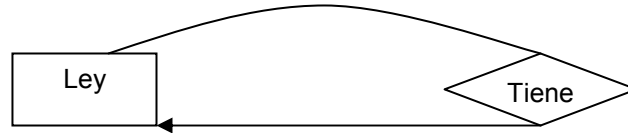
- **Relaciones n:n** → Cada elemento de una entidad A se puede relacionar con uno o varios elementos de la entidad B, y al mismo tiempo; cada elemento de la entidad B puede estar relacionado con uno o varios elementos de A.



**Por ejemplo:** CLIENTE – PELÍCULA. Un cliente puede alquilar varias películas, y una película puede ser alquilada por varios clientes.

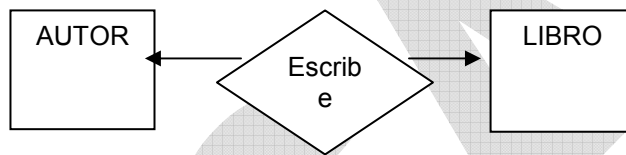
## Clasificación de relaciones por grado

- *Relación unaria:* Los elementos de una entidad se relacionan con otros elementos de la misma entidad:



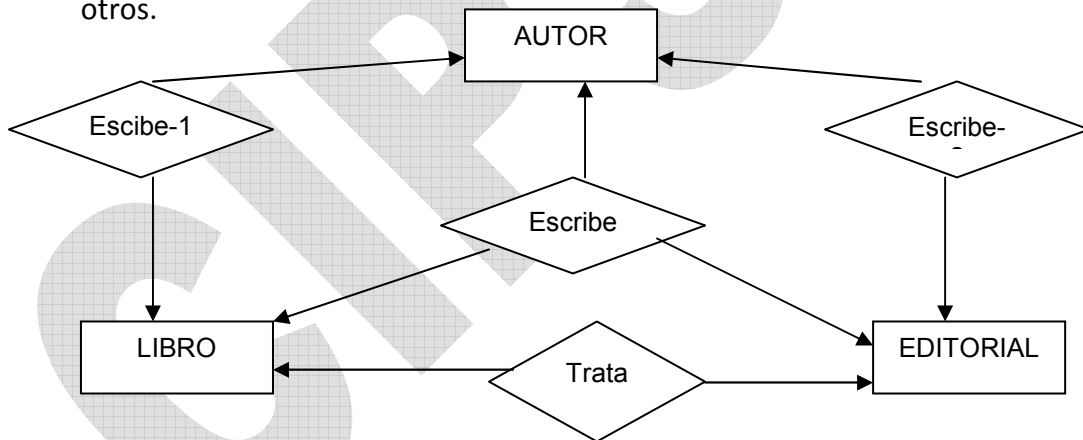
**Por ejemplo:** Una ley se compone de otras leyes.

- *Relaciones binarias:* Los elementos de una entidad se relacionan con los de otra entidad.



**Por ejemplo:** Un autor escribe un libro.

- *Relaciones ternarias:* Los elementos de tres entidades se relacionan unos con otros.



**Por ejemplo:** Un autor escribe un libro para una editorial.

## Reglas de integridad.

Las reglas de integridad establecen cómo se diferencian los elementos de una entidad, y cómo se relacionan con otros elementos de la misma entidad o de otra distinta.

### Regla de integridad de identidad:

Como ya se ha mencionado; cada entidad representa un conjunto de elementos: personas, coches, ciudades... etc. Cada elemento representa una determinada persona, coche, o ciudad, y almacena los datos necesarios:

#### Por ejemplo:

```
PERSONAS( NIF, NOMBRE, APELLIDOS, TELEFONO )
```

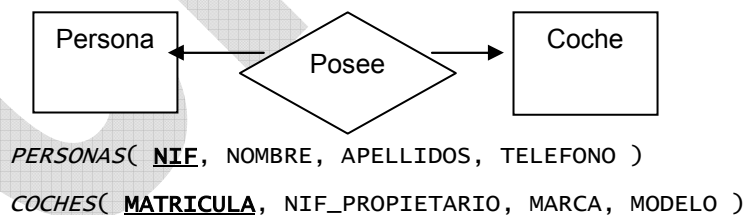
La *regla de integridad de identidad* determina: Todos los elementos de una entidad deben almacenar un valor único no nulo que no debe repetirse para ningún otro elemento de la entidad. Este valor recibe el nombre de *clave primaria* o *Primary Key*.

En el caso de la entidad PERSONAS, la clave primaria sería el DNI ya que todas las personas tienen un DNI exclusivo que no se repite.

### Regla de integridad referencial:

Las entidades se relacionan unas con otras. Esto implica que los elementos de una entidad se relacionan con los de otras.

#### Por ejemplo:



Una persona puede tener varios coches en propiedad, pero cada coche únicamente tiene un propietario. Para relacionar cada coche con la persona que lo conduce, cada elemento coche almacena un valor NIF\_PROPIETARIO, que contiene el NIF del elemento persona que es su propietario. El valor NIF\_PROPIETARIO recibe el nombre de *clave externa*, *extranjera*, o *Foreign Key*.

La *regla de integridad referencial* determina: Todo elemento con una clave externa debe relacionarse con la clave primaria de otro elemento el cual debe existir. En el caso del ejemplo; no podría existir un vehículo asociado a una persona inexistente.

## Diseño de una Base de Datos

Las bases de datos relacionales se crean siguiendo un esquema conceptual conforme a un modelo relacional. Este esquema establece que entidades de información almacena la base de datos y sus relaciones correspondientes.

La creación de una base de datos en base a un esquema conceptual se lleva a cabo siguiendo las siguientes directrices:

- Cada *Entidad* se convierte en una *Tabla*.

Una tabla se compone a su vez de una serie de *Registros* y *Campos*:

- Cada *Registro* es una fila que representa a su vez un elemento almacenado en la tabla. Si la tabla almacena ventas, cada registro representa los datos de una determinada venta.
- Cada *Campo* representa un dato que desea almacenarse para cada registro en la tabla y se corresponde con una columna. Cada columna lleva asociado un nombre y un tipo de dato asociado.
- Cada tabla posee una *clave primaria (pk)*. La clave primaria está compuesta por uno o varios campos cuyo valor(es) deben ser únicos para cada registro y no repetirse en ningún caso.

**Ejemplo:** Una tabla de **CLIENTES** podría almacenar los siguientes campos:

- **DNI**
- **Nombre**
- **Apellido**

CLIENTES		
DNI (pk)	NOMBRE	APELLIDO
79505906L	Pedro	Lopez
76096585W	Juan	Gonzalez
79605543L	Roberto	Esteban
73645635R	Luis	Ramirez

En el caso de la tabla **CLIENTES**, el **DNI** es la clave primaria puesto que todos los clientes tienen un número de identidad y es único para cada cliente.

- Las relaciones 1-N se representan como dos tablas cuyos registros se relacionan mediante una *clave externa (fk)*. Una clave externa es un campo cuyos valores son copia de la clave primaria de otra tabla.

**Ejemplo:** La tabla **CUENTAS** representa las cuentas bancarias de los clientes de un banco. Un cliente puede poseer muchas cuentas, y cada cuenta pertenece a un único cliente. Por ello, la tabla **CUENTAS** posee el campo **DNI** que hace de clave externa almacenando el número de identidad del cliente de la cuenta.

CLIENTES		
DNI (pk)	NOMBRE	APELLIDO
79505906L	Pedro	Lopez
76096585W	Juan	Gonzalez
79605543L	Roberto	Esteban
73645635R	Luis	Ramirez

CUENTAS		
NCUENTA (pk)	DNI (fk)	SALDO
29304	79505906L	1240,56
20395	76096585W	2024,50
23994	79605543L	22023,45
24569	73645635R	203,45
20495	79605543L	2304,45

- Las *relación N-M* no pueden representarse en una base de datos y debe descomponerse obligatoriamente en dos relaciones 1-N creando una nueva tabla intermedia.

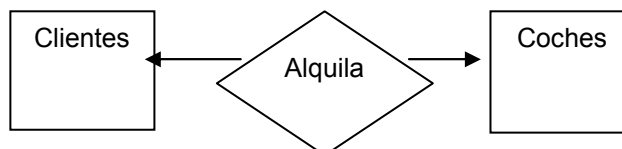
Supóngase una base de datos de un negocio de alquiler de coches. Cada coche es alquilado a muchos clientes, y cada cliente alquila muchos coches. Para descomponer esta relación es necesario crear una tercera tabla **ALQUILERES** junto con las tablas de **CLIENTES** y **COCHES**. La tabla **ALQUILERES** almacena cada alquiler indicando el coche alquilado y el cliente:

**CLIENTES**( DNI(PK), NOMBRE, APELLIDOS, EDAD )

**ALQUILERES**( DNI(FK), MATRICULA(FK), FECHA\_ALQUILER )

**COCHES**( MATRICULA(PK), MODELO, MARCA )

La tabla **ALQUILER** posee el campo **ID\_ALQUILER** con el nº de referencia de cada alquiler, y dos clave externas ( **DNI** y **MATRICULA** ) que almacenan el número de identidad del cliente y la matrícula del vehículo alquilado.





## Normalización de una base de datos

La normalización son un conjunto de normas que se aplican al diseño de una base de datos para optimizar su funcionamiento. Esto permite reducir el tamaño requerido para almacenar la misma información, evitar la duplicación de datos, y mantener más fácilmente la coherencia de los datos relacionados.

El proceso de normalización de una base de datos se ejecuta en niveles. Por norma general se emplean tres niveles de normalización, cada uno de los cuales implica el cumplimiento del nivel anterior más ciertas reglas adicionales.

### Primera forma normal. ( 1FN ). Eliminación de campos listados

La primera forma normal determina que cada columna sólo puede tener un valor para cada campo.

Supóngase la siguiente tabla:

```
ALUMNOS( DNI, NOMBRE, APELLIDO, CURSO1, CURSO2, CURSO3 )
```

Los campos Curso1, Curso2, y Curso3 constituyen lo que se denomina un *grupo de repetición*.

Los grupos de repetición se eliminan extrayéndolos en otra tabla:

```
ALUMNOS( DNI, NOMBRE, APELLIDO )  
CURSOS_CONTRATADOS( DNI_ALUMNO, CURSO )
```

La tabla **CURSOS\_CONTRATADOS** posee una clave primaria compuesta por el nombre del curso y el DNI del alumno que lo recibe, que a su vez hace de clave externa permitiendo relacionar curso y alumno.

### Segunda forma normal. ( 2FN ). Eliminación de redundancias

La segunda forma normal determina que todas aquellas columnas cuyos valores no dependan exclusivamente de la clave primaria deben colocarse en una tabla aparte con su propia clave primaria.

Supóngase la siguiente tabla:

```
CONDUCTORES( DNI, NOMBRE, EDAD, MATRICULA, MODELO, MARCA, NUM_PUERTAS )
```

Los campos *nombre* y *edad* dependen del campo **DNI** del conductor, no obstante; los campos *marca* y *número de puertas* dependen del *modelo* del vehículo y se repetirán en todos los usuarios con el mismo modelo de vehículo.

La solución es separar los datos del vehiculo en otra tabla.

*CONDUCTORES*( DNI, NOMBRE, EDAD, MATRICULA, MODELO )

*MARCAS*( MODELO, MARCA, NUMPUERTAS )

### Tercera forma normal ( 3FN ). *Asegurar dependencia funcional*

La tercera forma normal determina que no debe haber ningún campo en la tabla que no dependa directamente de la clave primaria de la misma.

Supóngase la siguiente tabla:

*TORNEOS*( NOMBRE\_TORNEO, AÑO, NOMBRE\_GANADOR, FECHA\_NACIMIENTO\_GANADOR )

El campo ***FECHA\_NACIMIENTO\_GANADOR*** está asociado al registro de torneo pero de modo indirecto, ya que es un dato que depende realmente de la persona que lo ha ganado. Se dice por tanto que el campo depende transitivamente de la clave primaria al hacerlo indirectamente a través del valor de otro campo.

La solución consiste en extraer el campo ***FECHA\_NACIMIENTO\_GANADOR***:

*TORNEOS*( NOMBRE\_TORNEO, AÑO, REF\_GANADOR )

*GANADORES*( REF\_GANADOR, GANADOR, FECHA\_NACIMIENTO )

La tabla Torneos posee una clave externa que permite asociar cada registro con el registro correspondiente del ganador del torneo en la tabla Ganadores.

\* \* \*

Existen más formas normales que permiten un diseño aún más optimizado. No obstante, en la mayoría de los casos suele ser suficiente con que el diseño cumpla las tres primeras formas normales.



## MySQL

MySQL es un software de gestión de bases de datos relacionales ( *SGBD* ) propiedad de Oracle. MySQL puede ser instalado en multitud de sistemas operativos y dispone de funciones de manejo de datos equiparables a las de otros *SGBD*'s como Microsoft SQL Server, Oracle, DB2... etc. MySQL ofrece buenos rendimientos con bases de datos grandes y es compatible con múltiples lenguajes y plataformas de desarrollo como .NET, Java, y PHP... etc.

### Descarga de MySQL

Aunque la mayoría de los paquetes de instalación de PHP y Apache incluyen también la instalación de MySQL y PHPMyAdmin; MySQL puede instalarse en un servidor por separado si va a ser empleado por varias aplicaciones y plataformas al mismo tiempo.

Existen varias versiones disponibles de MySQL: Estándar, Enterprise, Cluster y Community que es la única gratuita. Esta puede descargarse desde la web oficial: <http://www.mysql.com/downloads/mysql/>

**MySQL Community Server 5.5.8**

Select Platform:

Microsoft Windows Select

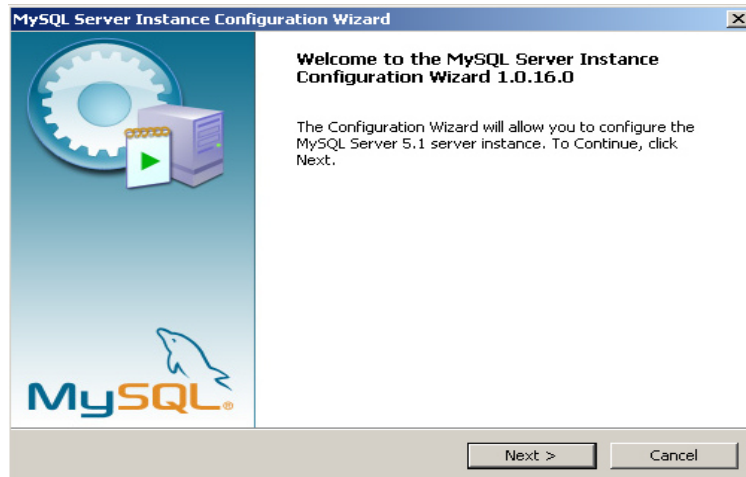
<b>Windows (x86, 32-bit), MSI Installer</b>	5.5.8	120.6M	<a href="#">Download</a>
(mysql-5.5.8-win32.msi) MD5: 491b624e86328ca16cb029aceb0c61fd   <a href="#">Signature</a>			
<b>Windows (x86, 64-bit), MSI Installer</b>	5.5.8	122.7M	<a href="#">Download</a>
(mysql-5.5.8-winx64.msi) MD5: b39111ae03bba4948cdf2ea9e5dea8a   <a href="#">Signature</a>			
<b>Windows (x86, 32-bit), ZIP Archive</b>	5.5.8	27.5M	<a href="#">Download</a>
(mysql-5.5.8.zip) MD5: 082ee988f6b1e852081a92d0ca3abd13   <a href="#">Signature</a>			
<b>Windows (x86, 32-bit), ZIP Archive</b>	5.5.8	132.5M	<a href="#">Download</a>
(mysql-5.5.8-win32.zip) MD5: b66785a4affbc88b73837bb6c6d777be   <a href="#">Signature</a>			
<b>Windows (x86, 64-bit), ZIP Archive</b>	5.5.8	134.7M	<a href="#">Download</a>
(mysql-5.5.8-winx64.zip) MD5: bd7612e41674578ecc02ff5cffe54288c   <a href="#">Signature</a>			

Existen múltiples versiones para diferentes sistemas operativos. Sólo es necesario escoger un sistema operativo y descargar el paquete deseado.

## Instalación independiente de MySQL

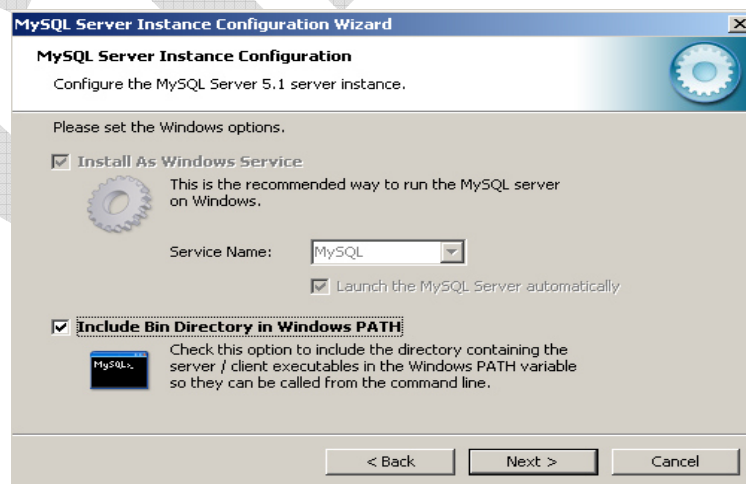
MySQL se instala de forma predeterminada como un servicio que se inicia al arrancar el ordenador, y que una vez en ejecución “escucha” cualquier petición de conexión realizada por parte de un usuario o aplicación. Este servicio recibe también el nombre de *instancia*.

La configuración de esta instancia se realiza durante la instalación mediante el *MySQL Instance Configuration Wizard*:



La configuración básica de la instancia de MySQL consta de las siguientes opciones:

→ **Configuración de servicio:** Permite indicar si se desea instalar MySQL como un servicio ( recomendado ), y el nombre de la instancia. Este nombre se corresponde con el identificador del servicio.



La segunda opción permite añadir la ruta de instalación de MySQL a la variable del sistema. Esto permite conectarse a MySQL y consultar o manipular las bases de datos desde la línea de comandos.

→ **Configuración de cuenta de administrador ( root ):** Permite indicar la cuenta de administrador inicial de MySQL.



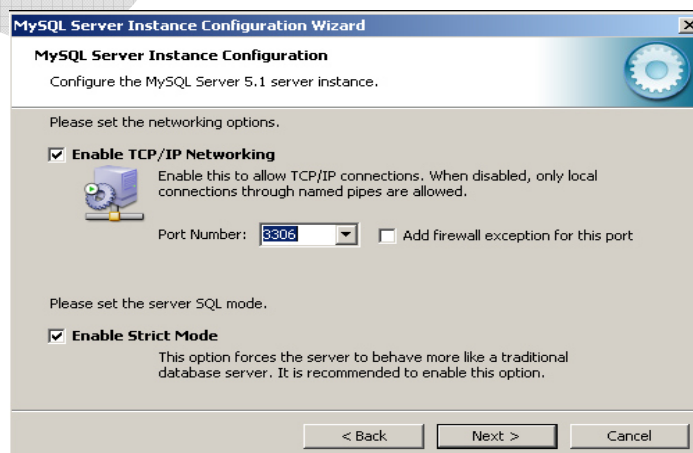
MySQL utiliza un sistema de cuentas de usuario para permitir el acceso a las bases de datos. La cuenta inicial se llama 'root' y tiene por defecto permisos para hacer cualquier cosa. Esta cuenta es muy delicada y debe estar protegida por contraseña para evitar accesos malintencionados.

De igual manera, por seguridad no se permite de forma predeterminada el acceso como "root" si no es desde la propia máquina en la que está instalado MySQL ( acceso local ). Para permitir la conexión desde cualquier otra máquina ( acceso remoto ) es necesario marcar la casilla de verificación "Enable root access from remote computer".

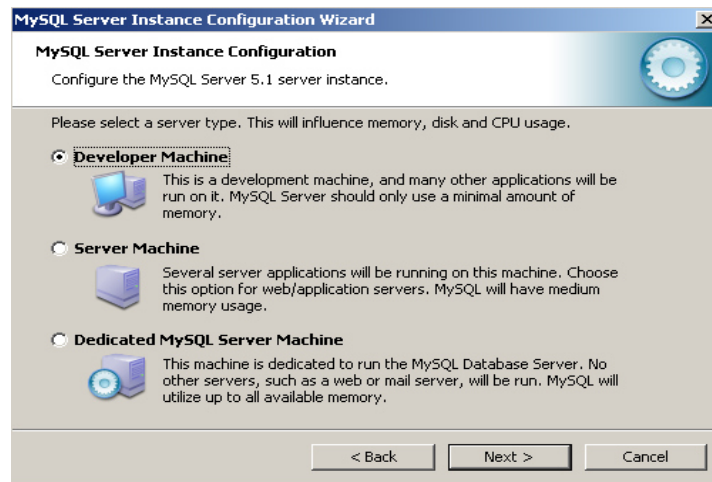
Opcionalmente, también es posible crear una cuenta anónima que permite el acceso a MySQL sin indicar contraseña con un mínimo de permisos.

### Opciones de configuración avanzada.

→ **Configuración de conexión remota:** La instancia de MySQL emplea el puerto 3306 del protocolo TCP/IP para recibir peticiones de acceso desde otros equipos. Por defecto esta opción está habilitada. Si se desactiva esta opción, MySQL únicamente acepta peticiones de conexión desde el propio equipo.



→ **Configuración de modo de servicio:** La instancia de MySQL puede configurarse con diferentes perfiles de funcionamiento según el uso que reciba el ordenador en el que está instalado:



- **Modo *Developer Machine*:** La opción recomendable para ordenadores de desarrollo en los que se programa y prueban aplicaciones contra una instancia de MySQL instalada localmente.
- **Modo *Server Machine*:** La opción recomendable para ordenador que funcionan como servidores Web con Apache, PHP y MySQL instalados.
- **Modo *Dedicated MySQL Server Machine*:** La opción recomendable para un ordenador que funciona exclusivamente como servidor de base de datos.

# PhpMyAdmin

*PhpMyAdmin* es un gestor web de bases de datos programado en PHP que permite la administración de las bases de datos y sus datos desde un navegador. Este puede descargarse gratuitamente de la web oficial:

[http://www.phpmyadmin.net/home\\_page/downloads.php](http://www.phpmyadmin.net/home_page/downloads.php)

*phpMyAdmin* viene integrado en la mayoría de los paquetes de instalación de PHP junto con *MariaDB*.

*MariaDB* es un SGBD de código abierto empleado como alternativa a MySQL, totalmente compatible con éste último. Puede obtenerse libremente de la web oficial:

<https://mariadb.org>

## Configuración de *phpMyAdmin*

La configuración de *phpMyAdmin* se realiza modificando directamente el archivo *config.inc.php* empleando un editor de texto sencillo. La configuración se almacena en una variable multidimensional denominada *cfg*:

```
/* Authentication type and info */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = '';
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['AllowNoPassword'] = true;
$cfg['Lang'] = '';

/* Bind to the localhost ipv4 address and tcp */
$cfg['Servers'][$i]['host'] = '127.0.0.1';
$cfg['Servers'][$i]['connect_type'] = 'tcp';

/* User for advanced features */
$cfg['Servers'][$i]['controluser'] = 'pma';
$cfg['Servers'][$i]['controlpass'] = '';
```

Las posiciones '*user*', '*password*' hacen referencia al usuario y clave empleados por *phpMyAdmin* para acceder al SGBD. Debe corresponderse con una cuenta con privilegios de administrador.

**Nota:** Por motivos de seguridad se recomienda modificar el nombre de la cuenta *root* tan pronto es posible para evitar posibles ataques.

La posición '*host*' indica la IP del equipo donde está instalado el SGBD. Por defecto ésta es *127.0.0.1*, o '*localhost*' indicando que está en el propio ordenador.

(\*) El archivo de configuración de *phpMyAdmin* puede encontrarse en la subcarpeta *phpMyAdmin* dentro de la carpeta de instalación de XAMPP.

## Configuración de MySQL mediante *PhpMyAdmin*

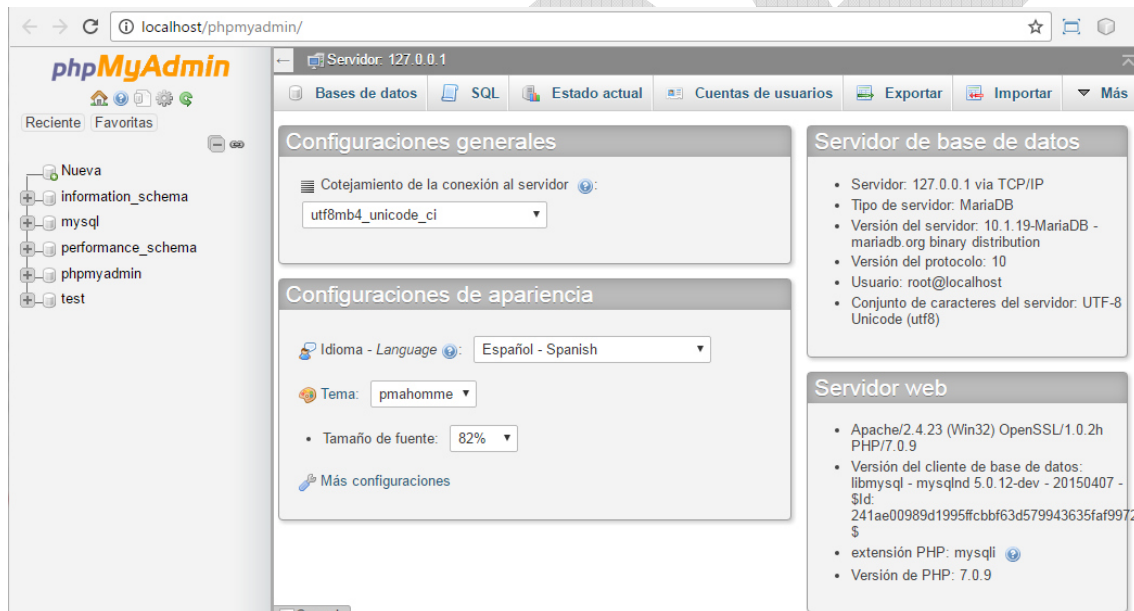
Una vez instalado phpMyAdmin, el acceso se realiza mediante un navegador accediendo a la siguiente dirección:

<http://<servidor>/phpmyadmin/>

Donde <servidor> se corresponde con el nombre o IP del servidor Web, o el nombre del dominio. En caso de estar trabajando localmente con el propio servidor Web bastaría indicar 'localhost':

<http://localhost/phpmyadmin>

Dependiendo de la configuración de seguridad establecida en el fichero de configuración *config.inc.php*, se solicita al usuario el nombre de usuario y contraseña de acceso a MySQL y se accede a la pantalla principal de phpMyAdmin:



Vista de la página web de inicio de PhpMyAdmin

La pantalla principal posee varias pestañas que permiten la realización de diferentes operaciones. Las más destacables son las siguientes:

- **Bases de Datos** → Muestra las bases de datos contenidas en el servidor MySQL.
- **SQL** → Permite escribir comandos y consultas en SQL para el SGBD
- **Estado Actual** → Muestra información sobre el estado actual de funcionamiento del SGBD.
- **Privilegios** → Permite gestionar las cuentas de usuario del servidor MySQL.
- **Procesos** → Muestra las conexiones establecidas con el servidor MySQL.
- **Exportar / Importar** → Permite exportar e importar datos procedentes de otras fuentes y bases de datos.



## Crear bases de datos.

*phpMyAdmin* permite tanto modificar, crear y eliminar bases de datos en el servidor, como los datos contenidos en las mismas.

**Ejemplo:** Se desea crear una base de datos para almacenar las cuentas y movimientos de una entidad bancaria. La base de datos va a llamarse 'banco' y se compone de las siguientes tablas:

- *Cuentas* → Almacena los datos de las cuentas registradas incluyendo el nº de cuenta, el nombre del titular, saldo, tasa de interés, si está bloqueada y la fecha de apertura.
- *Movimientos* → Almacena los movimientos de capital en las sueldas registrando el nº de cuenta, capital movido, concepto y fecha.

Para crear la base de datos lo primero es seleccionar la pestaña **Base de datos** de la pantalla principal de *phpMyAdmin* que muestra un listado con todas las bases de datos registradas:

Base de datos	Cotejamiento	
<input type="checkbox"/> information_schema	utf8_general_ci	<a href="#">Seleccionar privilegios</a>
<input type="checkbox"/> mysql	latin1_swedish_ci	<a href="#">Seleccionar privilegios</a>
<input type="checkbox"/> performance_schema	utf8_general_ci	<a href="#">Seleccionar privilegios</a>
<input type="checkbox"/> phpmyadmin	utf8_bin	<a href="#">Seleccionar privilegios</a>
<input type="checkbox"/> test	latin1_swedish_ci	<a href="#">Seleccionar privilegios</a>
<b>Total: 5</b>		latin1_swedish_ci

☐ Seleccionar todo    Para los elementos que están marcados: [Eliminar](#)

Vista de bases de datos registradas en el servidor

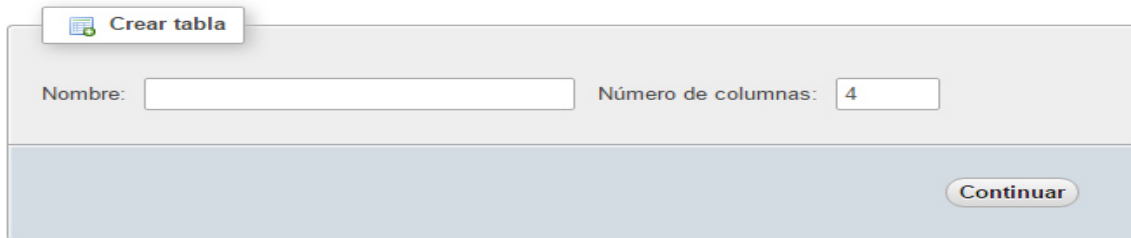
En la parte superior de la pantalla, aparece un cuadro de texto y una lista desplegable donde puede indicarse el nombre de una nueva base de datos y el conjunto de caracteres a utilizar. (se recomienda 'utf8\_unicode\_ci').

Pulsando el botón "Crear" se crea la base de datos con el nombre indicado.

## Crear tablas

Una vez creada la base de datos pueden crearse las tablas que la componen. Para ello se selecciona la nueva base de datos en la lista del margen izquierdo. Se muestra entonces la estructura de la base de datos con las tablas que la componen y un formulario en la parte inferior para crear nuevas tablas:

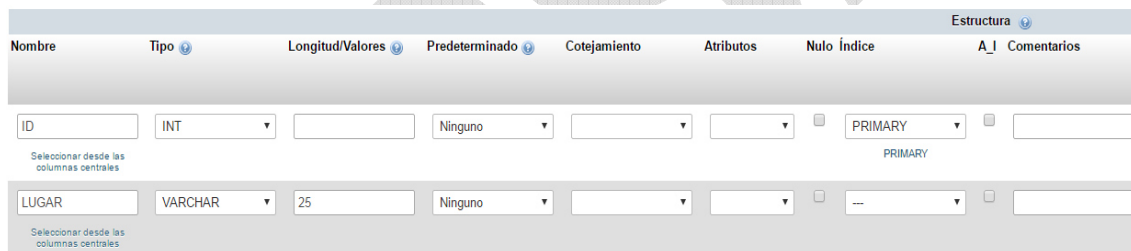
En el formulario debe indicarse el nombre de la nueva tabla y el nº de columnas que van a componerla:



Formulario de creación de una nueva tabla. Incluye un botón "Crear tabla" con un icono de documento y una lista. Campos para "Nombre:" (input vacío) y "Número de columnas:" (input con el valor 4). Un botón "Continuar" en la parte inferior derecha.

*Formulario de creación de una nueva tabla*

Al pulsar el botón continuar se muestra una página donde pueden configurarse el nombre y características de las columnas:



Formulario de definición de columnas. Encabezado: Estructura. Tabla de configuración:

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	Índice	A_I	Comentarios
ID	INT		Ninguno			<input type="checkbox"/>	PRIMARY	<input type="checkbox"/>	
<small>Seleccionar desde las columnas centrales</small>									
LUGAR	VARCHAR	25	Ninguno			<input type="checkbox"/>	---	<input type="checkbox"/>	
<small>Seleccionar desde las columnas centrales</small>									

*Formulario de definición de columnas*

Los campos mostrados tienen los siguientes significados:

- El campo **Nombre** indica el nombre de la columna
- El campo **Tipo** indica el tipo de dato almacenado en la columna.
- El campo **Longitud** permite indicar la longitud en caracteres de aquellas columnas con cadenas de texto.
- El campo **predeterminado** permite indicar un valor por defecto para la columna.
- El campo **Cotejamiento** permite indicar el conjunto de caracteres empleado por la tabla. Si no se indica, se emplea el definido por defecto por el servidor.
- La casilla de verificación **Nulo** permite indicar si la columna puede no tener valor.
- La lista desplegable **Índices**, permite indicar si la columna constituye clave primaria de la tabla (valor: **PRIMARY**).
- La casilla de verificación **AUTO\_INCREMENT** permite indicar si los valores de la columna son generados automáticamente por el servidor al insertar cada registro. ( autonumérico ).

Una vez definidas todas las columnas debe pulsarse el botón “*Guardar*” situado en la parte inferior del formulario.

Al definir cada columna de una tabla debe especificarse el tipo y tamaño de los datos que va a almacenar. Los tipos de datos disponibles son los siguientes:

### Números:

- **BIT** o **BOOL**, para un número entero que puede ser 0 ó 1
- **TINYINT** es un número entero con rango de valores válidos desde -128 a 127. Si se configura como unsigned (sin signo), el rango de valores es de 0 a 255
- **SMALLINT**, para números enteros, con rango desde -32768 a 32767. Si se configura como unsigned, 0 a 65535.
- **MEDIUMINT** para números enteros; el rango de valores va desde -8.388608 a 8388607. Si se configura como unsigned, 0 a 16777215
- **INT** para almacenar números enteros, en un rango de -2147463846 a 2147483647. Si configuramos este dato como unsigned, el rango es 0 a 4294967295
- **BIGINT** número entero con rango de valores desde -9223372036854775808 a 9223372036854775807. Unsigned, desde 0 a 18446744073709551615.
- **FLOAT** (m,d) representa números decimales. Podemos especificar cuantos dígitos (m) pueden utilizarse (término también conocido como ancho de pantalla), y cuantos en la parte decimal (d). Mysql redondeará el decimal para ajustarse a la capacidad.
- **DOUBLE** Número de coma flotante de precisión doble. Es un tipo de datos igual al anterior cuya única diferencia es el rango numérico que abarca
- **DECIMAL** almacena los números como cadenas.

### Cadenas de texto:

- **CHAR** Este tipo se utiliza para almacenar cadenas de longitud fija. Su longitud abarca desde 1 a 255 caracteres.
- **VARCHAR** Al igual que el anterior se utiliza para almacenar cadenas, en el mismo rango de 1-255 caracteres, pero en este caso, de longitud variable. Un campo CHAR ocupará siempre el máximo de longitud que le hallamos asignado, aunque el tamaño del dato sea menor (añadiendo espacios adicionales que sean precisos). Mientras que VARCHAR solo almacena la longitud del dato, permitiendo que el tamaño de la base de datos sea menor. Eso sí, el acceso a los datos CHAR es mas rápido que VARCHAR.
- **TINYTEXT** almacena cadenas de texto plano (sin formato) y case-insensitive (sin distinguir mayúsculas o minúsculas) para un máximo de 255 caracteres.
- **TEXT** se usa para cadenas con un rango de 255 - 65535 caracteres.
- **MEDIUMTEXT** textos de hasta 16777215 caracteres.
- **LONGTEXT** textos de hasta máximo de 4.294.967.295 caracteres

### Datos binarios

Estos tipos de datos almacenan datos binarios que pueden corresponderse con cualquier tipo de información tal como imágenes, archivos de sonido, videos... etc.

- **TINYBLOB** Almacena un máximo de 255 bytes de información
- **BLOB**: Almacena datos comprendidos entre los 255 y 64 KBytes.
- **MEDIUMBLOB**: Almacena datos de hasta 16 Mbytes.
- **LOB**: Almacena datos de hasta 4 Gbytes.

### Fechas

Estos tipos de datos almacenan datos temporales tales como fechas y horas en diferentes rangos:

- **DATE** para almacenar fechas. El formato por defecto es YYYY MM DD desde 0000 00 00 a 9999 12 31.
- **DATETIME** Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-día horas:minutos:segundos
- **TIMESTAMP** Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo
- **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'

**Ejemplo:** Crear en la base de datos “banco” las siguientes tablas con las columnas y tipos indicados:

**Tabla Cuentas:**

Columna	Tipo	Nulo	Predeterminado
NUMERO_CUENTA ( <i>Primaria</i> )	char(8)	No	
TITULAR	varchar(50)	No	
SALDO	decimal(10,2)	Sí	0.00
INTERES	decimal(3,2)	Sí	0.00
BLOQUEADA	tinyint(1)	No	0
APERTURA	datetime	No	

**Tabla Movimientos:**

Columna	Tipo	Nulo	Predeterminado
ID_MOVIMIENTO ( <i>Primaria</i> )	int(11)	No	
CANTIDAD	decimal(10,2)	No	
CONCEPTO	varchar(255)	No	
CUENTA	char(8)	No	
FECHA	datetime	No	

## Crear relaciones entre tablas.

Las relaciones en las tablas establecen el modo en que los registros de una tablas se relacionan con los de otras. Para ello deben definirse claves externas en las tablas dependientes y asociarlas a las claves primarias de las tablas principales.

En el caso de la base de datos *banco*, las tablas **CUENTAS** y **MOVIMIENTOS** están relacionadas ya que cada *movimiento* es referido a una *cuenta*, mientras que una *cuenta* posee múltiples *movimientos*. Existe por una relación 1-N entre ambas tablas:

- La tabla principal es **CUENTA**, cuya clave primaria es el campo **NUMERO\_CUENTA**.
- La tabla secundaria es **MOVIMIENTO**, cuya clave primaria es ID\_**MOVIMIENTO**, y a su vez posee el campo CUENTA que hace de clave externa asociada a la clave primaria de la tabla CUENTAS.

### Creación de índice en la columna clave externa

Para crear una clave externa, primero debemos declarar un índice en la columna que va a hacer de clave externa. Para ello seleccionamos la tabla **MOVIMIENTOS** en el margen derecho, y a continuación la pestaña “Estructura”.

Se muestra entonces las columnas que conforman la tabla **MOVIMIENTOS**. Bajo la columna “acción” puede verse un icono “Índice” para cada columna:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/>	1 ID_MOVIMIENTO	int(11)			No	Ninguna	AUTO_INCREMENT	
<input type="checkbox"/>	2 CANTIDAD	decimal(10,2)			No	Ninguna		
<input type="checkbox"/>	3 CONCEPTO	varchar(255)			No	Ninguna		
<input type="checkbox"/>	4 CUENTA	char(8)			No	Ninguna		
<input type="checkbox"/>	5 FECHA	datetime			No	Ninguna		

Para fijar un índice en la columna **CUENTA**, debemos hacer clic sobre su enlace “Índice” y aceptar el cuadro de diálogo de confirmación:

Confirmar

¿Realmente desea ejecutar "ALTER TABLE `movimientos` ADD INDEX(`CUENTA`);"?

OK

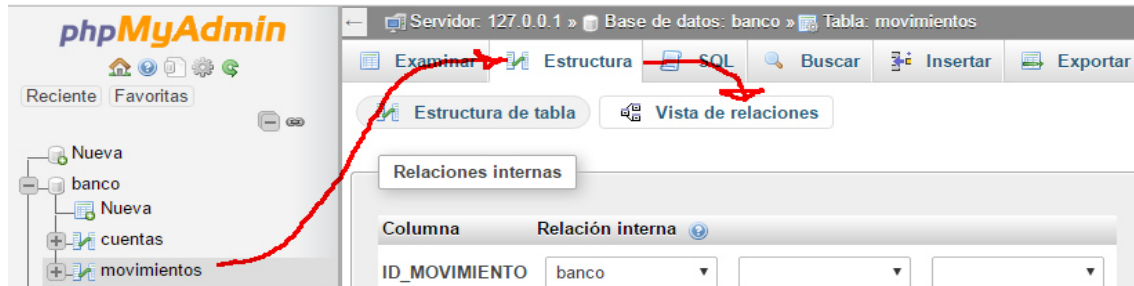
Cancelar

Una vez establecido el índice en la columna CUENTA, debe mostrarse una icono de una llave de color gris a la derecha del nombre de la columna:

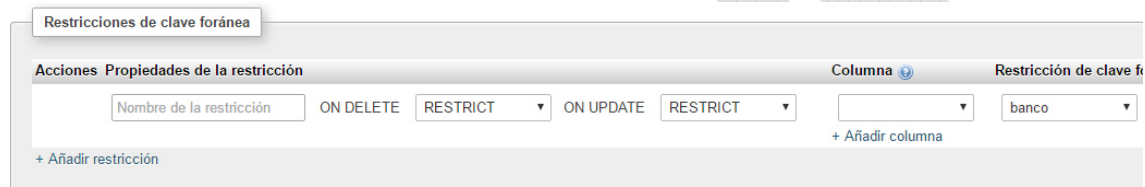
<input type="checkbox"/>	4 CUENTA	char(8)	No	Ninguna
--------------------------	----------	---------	----	---------

## Establecimiento de la clave externa

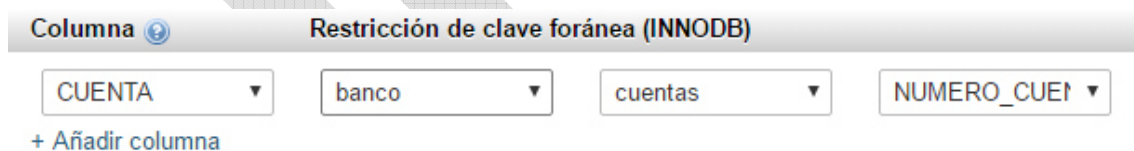
Para establecer la clave externa debemos seleccionar la tabla, a continuación la pestaña “Estructura” y la subpestaña “Vista de relaciones”:



En la parte inferior se muestra un formulario bajo el epígrafe “Restricciones de clave foránea”, donde pueden definirse las claves externas de la tabla haciendo clic sobre el enlace “+ Añadir restricción”:



Primero debe seleccionarse la columna de la propia tabla MOVIMIENTO que hace de clave externa. Para ello desplegamos la lista bajo el título “Columna”. Esta debe mostrar las columnas **ID\_MOVIMIENTO** y **CUENTA**. (Si no aparecerá **CUENTA**, es que no hemos creado el índice anterior). Seleccionamos la columna **CUENTA**.



**Selección de la clave primaria asociada a la clave externa**

Ahora debemos indicar la tabla y columna a la que se asocia la clave externa. En las listas desplegables a su derecha bajo el apartado “Restricción de clave foránea” debemos seleccionar a continuación la base de datos, la tabla y la columna clave primaria a la que se asocia la clave externa. En este caso, es el campo NUMERO\_CUENTA de la tabla CUENTAS.

Bajo el apartado “Restricciones” podemos indicar un identificador para la clave externa y las restricciones de integridad asociadas:

- **ON DELETE** → Indica qué sucede si se intenta eliminar un registro de CUENTAS que tiene MOVIMIENTOS asociados.
- **ON UPDATE** → Indica qué sucede si se intenta modificar el campo NUMERO\_CUENTA de un registro cuenta con registros MOVIMIENTO.

En ambos casos, el valor “RESTRICT” predeterminado indica que no se permite ni eliminar no modificar la clave primaria de un registro de LUGAR si tiene registros POSICION asociados.

## Examinar los datos de una tabla

Para ver los datos contenidos en una tabla debemos seleccionar primero la tabla correspondiente en la vista jerárquica del margen izquierdo. Si no se muestra, quizá debamos seleccionar primero la base de datos para que despliegue sus tablas.



Se muestran entonces por defecto los datos contenidos en la tabla. Estos pueden verse igualmente seleccionando la pestaña **“Examinar”** de la barra superior de opciones:

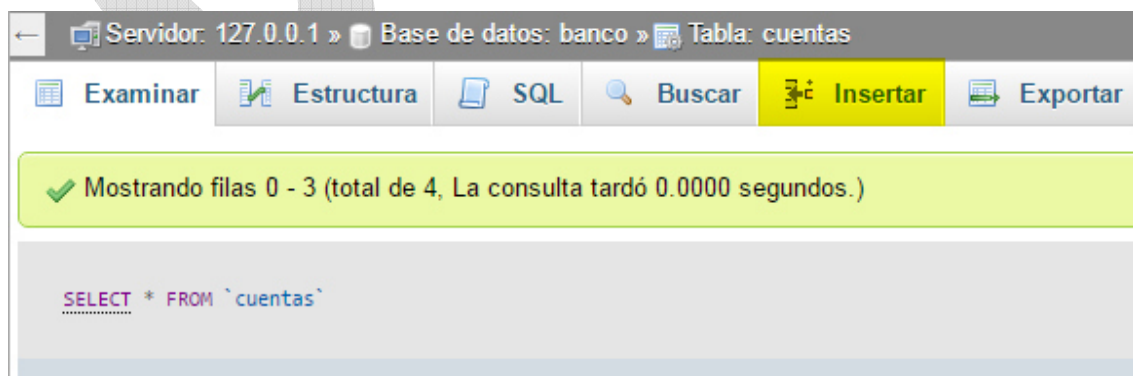
					ID_LUGAR	LUGAR
<input type="checkbox"/>		Editar		Copiar		Borrar
					1	Bilbao
<input type="checkbox"/>		Editar		Copiar		Borrar
					2	Vitoria

Vista de datos de la tabla 'lugares'

Las opciones **“Editar”** y **“Borrar”** permiten modificar o eliminar el registro correspondiente.

## Insertar registros

Para insertar registros en una tabla debemos seleccionar la pestaña **“Insertar”** situada en la barra superior de opciones. Se muestra entonces un formulario con tantas cajas de texto como campos de la tabla para introducir los valores de los nuevos registros:



Una vez introducidos los valores pulsamos el botón **“Continuar”** para registrarlos.

**(\*) No debe asignarse valor a aquellos campos configurados como auto-numéricos. El valor es generado automáticamente por la propia base de datos al insertar un nuevo registro.**



## Programación Web en PHP

En el caso de los campos clave externa, se muestra una lista en vez de una caja de texto donde puede seleccionarse un valor de la clave primaria asociada.

Por ejemplo, al insertar un registro de **MOVIMIENTOS**, el campo **CUENTA** muestra una lista con los valores de la columna **NUMERO\_CUENTA** de la tabla **CUENTAS**:

Columna	Tipo	Función	Nulo	Valor
ID_MOVIMIENTO	int(11)			
CANTIDAD	decimal(10,2)			
CONCEPTO	varchar(255)			
CUENTA	char(8)			AC129384 AC439586 AC490385 AC594045
FECHA	datetime			

Vista de cuadro de insertar registro para la tabla MOVIMIENTOS

## Modificar y eliminar registros

Para modificar el valor de algún campo de un registro sólo es necesario hacer clic sobre el vínculo **"Editar"** del registro que se quiere modificar.

ID	CANTIDAD	CONCEPTO	CUENTA	FECHA
11	100.00	PRUEBA MOVIMIENTO 1	AC439586	2017-03-30 20:06:38

Vista de un registro de la tabla MOVIMIENTOS con los vínculos "Editar" y "Borrar"

Esto muestra un formulario con el valor de todos sus campos donde éstos pueden modificarse:

Columna	Tipo	Función	Nulo	Valor
ID_MOVIMIENTO	int(11)			11
CANTIDAD	decimal(10,2)			100.00
CONCEPTO	varchar(255)			PRUEBA MOVIMIENTO 1
CUENTA	char(8)			AC439586
FECHA	datetime			2017-03-30 20:06:38

Formulario de edición de campos de registro de la tabla MOVIMIENTOS

Para eliminar un registro sólo es necesario hacer clic sobre el vínculo **"Borrar"**.

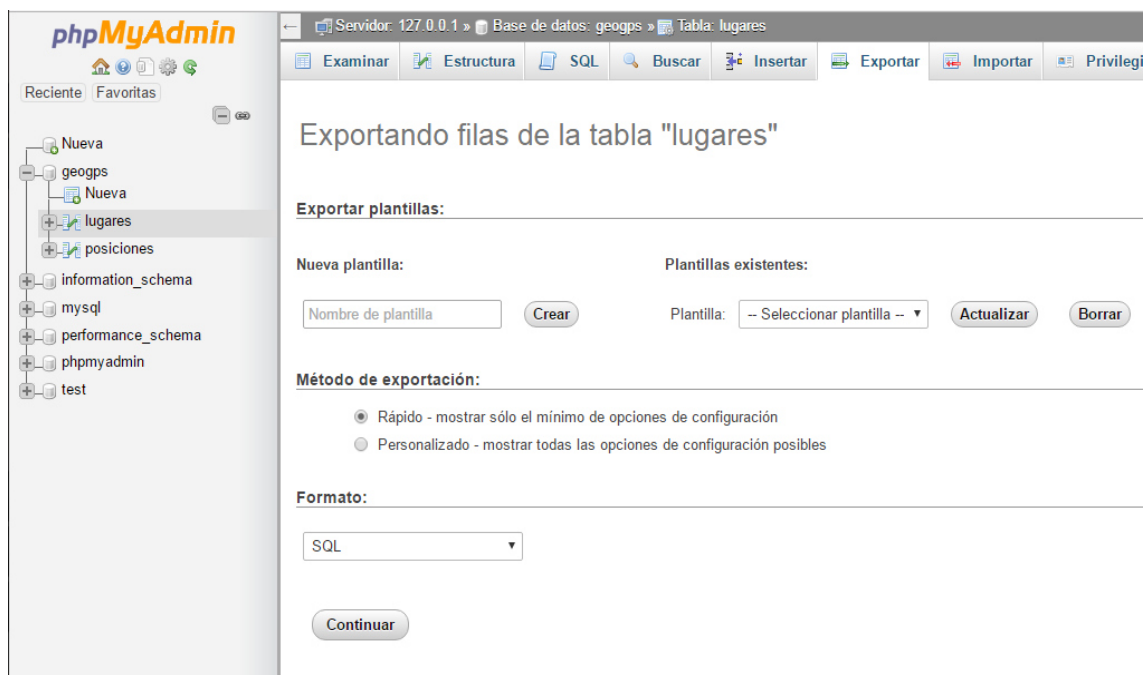


## Exportación de datos

Exportar consiste en copiar en un fichero los datos y estructura de una base de datos. La exportación puede emplearse para hacer copias de seguridad, transferir datos entre dos bases de datos, o transferir toda una base de datos de un servidor a otro.

### Exportación de una tabla

Exportar una tabla significa exportar sus datos y estructura. Para ello se selecciona la tabla en la vista jerárquica de la derecha y después seleccionar la pestaña “Exportar”:



Ventana de exportación de tabla.

Al darle al botón “Continuar”, se genera un archivo con el nombre de la tabla y extensión “.sql” que puede descargarse mediante el navegador.

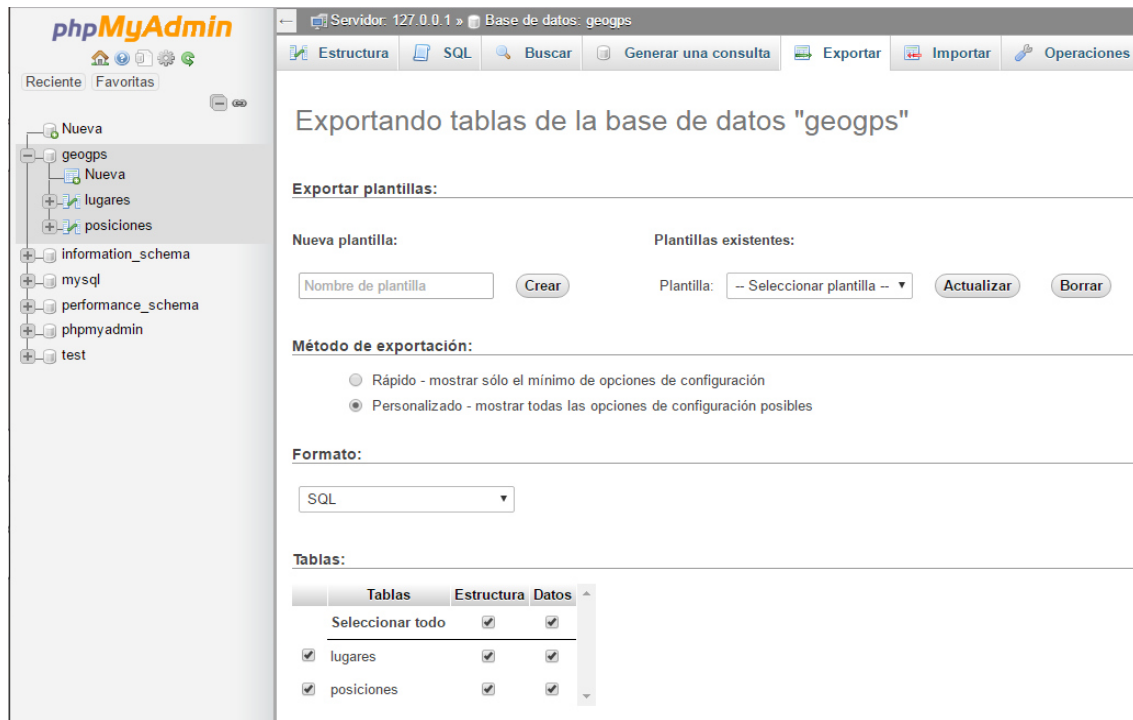
En el caso de exportar la tabla **LUGARES**, el fichero descargado sería “lugares.sql”. Dicho fichero contiene el código SQL para recrear e insertar los datos contenidos en la tabla en el momento de la exportación.

### Exportación de tablas

También es posible exportar la estructura y datos de varias o todas las tablas contenidas en una base de datos. Para ello debemos seleccionar la base de datos en la vista jerárquica de la derecha. A continuación seleccionamos la pestaña “Exportar” de la barra superior.

## Programación Web en PHP

De manera predeterminada esto exportará todas las tablas de la base de datos, pero si marcamos en la categoría **“Modos de exportación”** la opción **“Personalizado”**; se mostrará el apartado **“Tablas”** donde pueden marcarse las tablas a exportar:

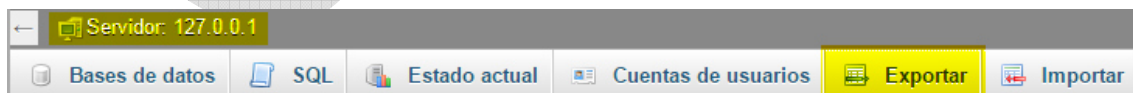


Ventana de exportación personalizada de tablas de una base de datos

Es importante destacar que éste modo exporta las tablas no la propia base de datos. Serviría por tanto para restaurar los datos y tablas de la base de datos, pero no para restaurar la base de datos en si, por ejemplo; en otro servidor.

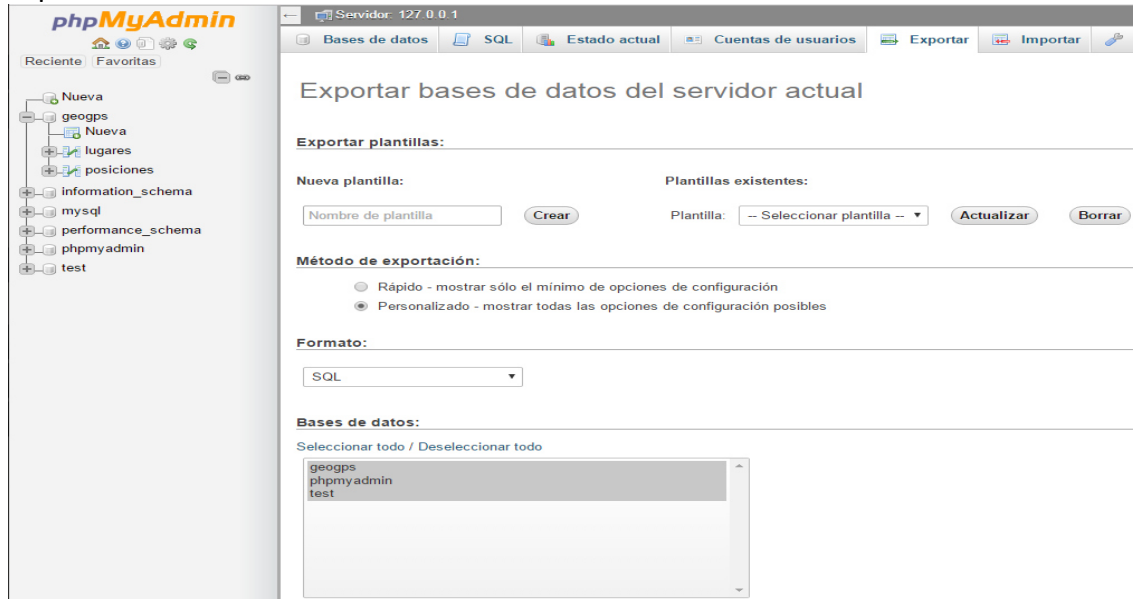
### Exportación de base de datos

Exportar la base de datos significa exportar no sólo sus tablas sino también la propia base de datos de modo que pueda restaurarse posteriormente o migrarla a otro servidor. Para ello debemos seleccionar el propio servidor en la barra superior y a continuación la pestaña “Exportar”.



De este modo podemos exportar de manera predeterminada todas las bases de datos existentes en el servidor.

Si marcamos en la categoría “**Modos de exportación**” la opción “**Personalizada**”, se mostrará el apartado “**Base de datos**” donde puede seleccionarse las bases de datos a exportar:



*Ventana de exportación personalizada de bases de datos de un servidor*

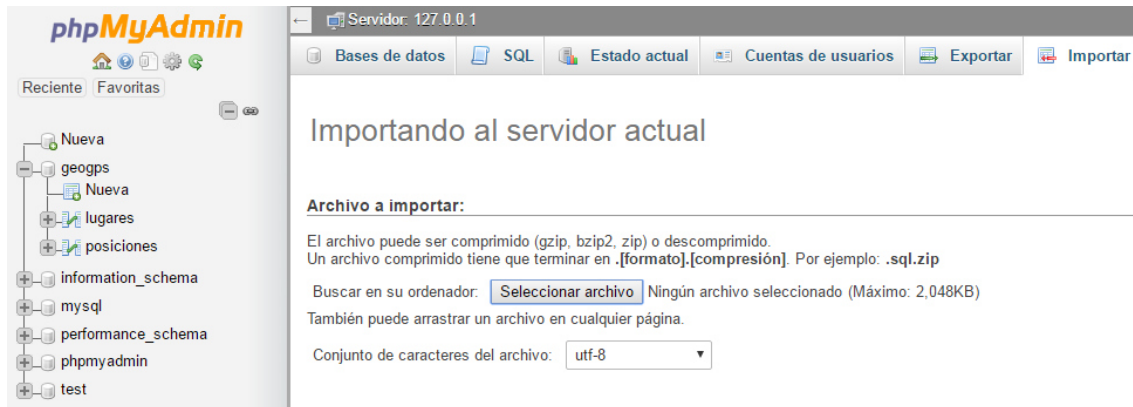
### Formatos de exportación

De manera predeterminada el formato de exportación de los datos es **SQL**. En este formato, los ficheros exportados contienen las sentencias SQL necesarias para crear la base de datos, sus tablas y los datos contenidos. Existe no obstante la posibilidad de exportar los datos en otros formatos como JSON, Código PHP, CSV, PDF, Microsoft Excel..., etc; con el fin de importarlos en otras plataformas y aplicaciones.

### Importación de datos

La importación es el proceso inverso de la exportación en la que se recuperan los datos, tablas, o bases de datos previamente exportados en un fichero.

Si el fichero contiene una o varias bases de datos debe seleccionarse el servidor en la barra superior y a continuación la pestaña **“Importar”**. Se muestra entonces un formulario donde debe pulsarse el botón **“Seleccionar archivo”** para seleccionar y subir al servidor web el fichero a importar.



Si el fichero contiene la exportación de una o varias tablas, debe seleccionarse primero la base de datos donde se desean importarlas y seleccionar la pestaña **“Importar”**.

# SQL

SQL es un lenguaje de consulta y manipulación de bases de datos compatible con la mayoría de los sistemas de bases de datos relacionales actuales tales como MySQL, Microsoft Access, SQL Server, IBM DB2, Oracle..., etc.

Las sentencias de SQL pueden dividirse en dos categorías principales:

- **Sentencias de definición de datos ( DDL )** → Son sentencias que permiten definir, modificar y eliminar elementos de la estructura de las bases de datos tales como tablas, índices, vistas, restricciones..., etc; además de permitir crear y eliminar las propias bases de datos en los servidores.
- **Sentencias de manipulación de datos ( DML )** → Son sentencias que permiten insertar, modificar y eliminar datos de las tablas.

## Ejecución de sentencias SQL en phpMyAdmin

Para introducir y ejecutar cualquier sentencia SQL en phpMyAdmin debe seleccionarse la pestaña “SQL”. Esto muestra un cuadro de texto donde pueden introducirse las sentencias para su ejecución.

*Ventana de introducción de sentencia SQL a nivel de servidor*


Para ejecutar la sentencia debe pulsarse el botón “**Continuar**”. El botón “**Formato**” aplica un formato a la ser más clara.

## Programación Web en PHP

Es importante introducir las sentencias en el ámbito correcto para su ejecución.

- Las sentencias de creación o eliminación de bases de datos deben ejecutarse a nivel de servidor.
- Las sentencias de creación, eliminación o modificación de las tablas de una base de datos, o consultas que combinen los datos de varias tablas deben realizarse seleccionando primeramente la base de datos y después la pestaña “SQL”.
- Las sentencias de consulta, inserción, modificación o eliminación de registros de una tabla concreta deben introducirse seleccionando primeramente la tabla y después la pestaña “SQL”.

Las sentencias de consulta (**SELECT**) muestran tras ejecutarse los resultados solicitados:



Mostrar ventana de consultas SQL

✓ Mostrando filas 0 - 1 (total de 2, La consulta tardó 0.0000 segundos.)

```
select * from lugares
```

[ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas:

Ordenar según la clave: Ninguna

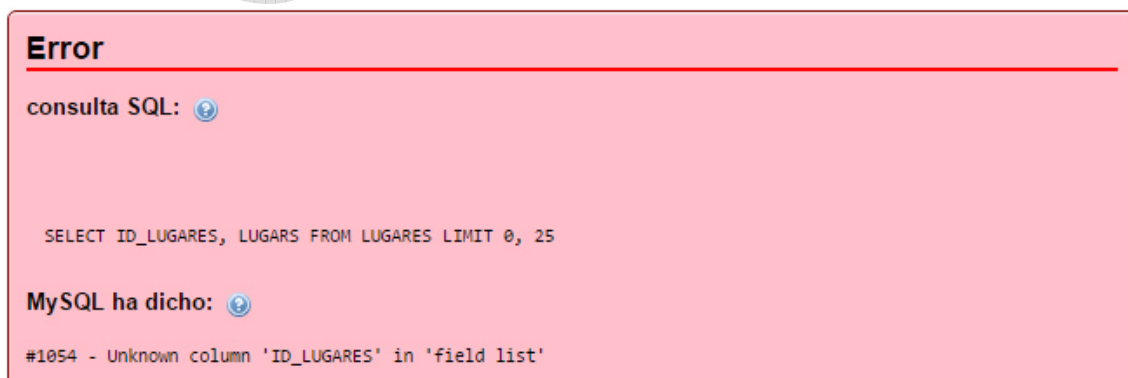
+ Opciones

	ID_LUGAR	LUGAR
<input type="checkbox"/> Editar <input type="button" value="Copiar"/> <input type="button" value="Borrar"/>	1	Bilbao
<input type="checkbox"/> Editar <input type="button" value="Copiar"/> <input type="button" value="Borrar"/>	2	Vitoria

Ventana de resultados de una consulta sobre la tabla LUGARES

Las sentencias de inserción (**INSERT**), modificación (**UPDATE**) y eliminación (**DELETE**) de datos muestran un mensaje indicando el nº de registros afectados.

Si la sentencia no es correcta se muestra como resultado un cuadro de error:



**Error**

consulta SQL:

```
SELECT ID_LUGARES, LUGARS FROM LUGARES LIMIT 0, 25
```

MySQL ha dicho:

#1054 - Unknown column 'ID\_LUGARES' in 'field list'

## Sentencias de definición de datos ( DDL )

### Sentencia **CREATE**

La sentencia **CREATE** se emplea para crear una base de datos, y definir sus tablas.

```
CREATE DATABASE <nombre_basedatos>
CREATE TABLE <nombre_tabla> (
    <columna_1> <tipo> <opciones>,
    ...
    <columna_n> <tipo> <opciones>
);
```

**Ejemplo:** Supóngase la siguiente tabla:

CLIENTES( NIF, NOMBRE, APELLIDOS, EDAD )

```
CREATE TABLE CLIENTES (
    NIF VARCHAR(9) NOT NULL PRIMARY KEY,
    NOMBRE VARCHAR(15),
    APELLIDOS VARCHAR(25),
    EDAD INT
);
```

### Sentencia **DROP**

La sentencia DROP se emplea para eliminar una tabla y todos los datos que contiene. El formato es:

```
DROP TABLE <nombre_tabla>
```

**Ejemplo:** Eliminar la tabla CLIENTES.

```
DROP TABLE CLIENTES;
```

### Sentencia **ALTER**

La sentencia ALTER se emplea para modificar la estructura de una tabla ya existente indicando la tabla que se modifica y el cambio que se realiza, el cual puede consistir en añadir, modificar o eliminar una columna o relación.

**Ejemplo:** Renombrado de una tabla.

```
ALTER TABLE tabla1 RENAME AS tabla2;
```

**Ejemplo:** Añadir una columna a una tabla.

```
ALTER TABLE table1 ADD COLUMN nueva_columna VARCHAR(50) NOT NULL;
```

**Ejemplo:** Eliminar una columna de una tabla.

```
ALTER TABLE table1 DROP COLUMN columna;
```

**Habitualmente el diseño y modificación de la estructura de la base de datos se realiza empleando diseñadores, por lo que no es necesario el empleo de estas sentencias.**

## Sentencias de manipulación de datos ( *DML* )

### Sentencia *SELECT*

Esta es la sentencia principal de consulta para obtener cualquier tipo de información de una base de datos. El formato básico es:

```
SELECT <columna_1>, <columna_2>,..., <columna_n>
FROM <tabla1> INNER JOIN <tabla2> ON <relación>
WHERE <condición_filtrado>
GROUP BY <campo_agrupamiento_1>,<campo_agrupamiento_2>...
HAVING <condición_agrupamiento>
ORDER BY <campo_ordenamiento_1>,<campo_ordenamiento_2>...
```

La partícula *FROM* indica la tabla de la que provienen los registros a devolver, y la partícula *SELECT* los campos que deben mostrarse de cada registro devuelto.

**Ejemplo:** Recupera el nº de cuenta, titular y saldo de todas las cuentas registradas en la tabla **CUENTAS**:

```
SELECT NUMERO_CUENTA, TITULAR, SALDO
FROM CUENTAS
```

### Filtrado

La partícula *WHERE* permite indicar una condición que determine qué registros deben mostrarse. Si se omite, se devuelven todos los registros.

**Ejemplo:** Recupera el nº de cuenta, titular y saldo de todas las cuentas registradas en la tabla CUENTAS con saldo superior a 500€.

```
SELECT NUMERO_CUENTA, TITULAR, SALDO
FROM CUENTAS
WHERE SALDO > 500.0
```

Las condiciones pueden realizarse empleando los operadores de comparación típicos tales como: >, <, >=, <=, !=, =. Sólo los dos últimos operadores son aplicables a datos de tipo cadena de texto.

Para la comparación de cadenas de texto puede resultar útil el uso de la cláusula LIKE, que permite comparaciones parciales del tipo “¿empieza la cadena por ‘BA’?, ¿Termina en ‘H’?... etc. Para ello se emplean también los denominados caracteres comodines:

- El guión bajo ‘\_’ representa un único carácter.
- El porcentaje ‘%’ representa uno o varios caracteres.



**Ejemplo:** Recupera el nº de cuenta, titular y saldo de todas las cuentas registradas en la tabla **CUENTAS** cuyo nombre sea “ROGER” independientemente del apellido.

```
SELECT NUMERO_CUENTA, TITULAR, SALDO
FROM CUENTAS
WHERE TITULAR LIKE '%ROGER%'
```

También es posible componer condicionales compuestas combinando varias condicionales simples mediante operadores lógicos:

- **AND:** ( Conjunción ). La condición es cierta si todas las condiciones son ciertas.
- **OR:** ( Disyunción ). La condición es cierta si una o ambas condiciones son ciertas.
- **NOT:** ( Negación ). Niega el resultado de una condición.

**Ejemplo:** Recupera el nº de cuenta, titular y saldo de todas las cuentas registradas en la tabla **CUENTAS** cuyo nombre de titular sea “ROGER” y el saldo sea superior a 2000€.

```
SELECT NUMERO_CUENTA, TITULAR, SALDO
FROM CUENTAS
WHERE TITULAR LIKE '%ROGER%'
AND SALDO > 2000
```

### Ordenamiento

Los resultados devueltos pueden mostrarse ordenados por el valor de uno o varias columnas, de manera ascendente, o descendente. Para ello se añade la cláusula **ORDER BY** seguida de los campos por los que ordenar y la palabra clave **ASC** para orden ascendente y **DESC** para descendente.

**Ejemplo:** Recupera el nº de cuenta, titular y saldo de todas las cuentas registradas en la tabla **CUENTAS** ordenadas por el saldo en sentido creciente.

```
SELECT NUMERO_CUENTA, TITULAR, SALDO
FROM CUENTAS
ORDER BY SALDO
```

**Ejemplo:** Recupera el nº de cuenta, titular, saldo y fecha de apertura de todas las cuentas registradas en la tabla **CUENTAS** ordenadas por fecha de apertura en sentido decreciente, es decir; primero las más modernas y por último las más antiguas.

```
SELECT NUMERO_CUENTA, TITULAR, SALDO, APERTURA
FROM CUENTAS
ORDER BY APERTURA DESC
```

### Combinaciones

Las tablas en una base de datos se relacionan entre sí mediante la presencia de claves externas. Esto permite realizar consultas combinando los registros de varias tablas.

**Ejemplo:** Deseamos obtener los datos de todos los movimientos realizados sobre las cuentas del titular 'ROGER' junto con el saldo e interés de la cuenta a la que se refiere cada movimiento.

Dado que los datos de cada movimiento están en la tabla MOVIMIENTOS, pero los de sus cuentas se almacenan en el registro de la tabla CUENTAS referenciados por el campo clave externa CUENTA; es necesario realizar una consulta combinando ambas tablas.

```
SELECT MOVIMIENTOS.CONCEPTO,  
       MOVIMIENTOS.CANTIDAD,  
       MOVIMIENTOS.FECHA,  
       CUENTAS.SALDO,  
       CUENTAS.INTERES  
FROM MOVIMIENTOS INNER JOIN CUENTAS  
      ON MOVIMIENTOS.CUENTA = CUENTAS.NUMERO_CUENTA  
WHERE TITULAR LIKE '%ROGER%'
```

La partícula **INNER JOIN** indica la tabla con la que se combinan los registros de la tabla indicada en **FROM**. La partícula **ON** indica los campos de ambas tablas que emparejan sus registros. La cláusula final WHERE restringe los empleados que deben mostrarse.

**(\*) Es costumbre cuando se escriben consultas que combinan varias tablas, indicar el nombre de cada campo anteponiendo el nombre de la tabla al que pertenece.**

### Agrupamiento

El agrupamiento permite obtener resultados a partir del agrupamiento de registros por el valor de uno o varios de sus campos indicados mediante la cláusula **GROUP BY**:

**Ejemplo:** Supóngase que se desea obtener el sumatorio del saldo de todas las cuentas de cada titular.

En este caso, los registros deben agruparse por el campo **TITULAR**, obteniendo el sumatorio de sus saldos mediante la función **SUM(SALDO)**:

```
SELECT TITULAR, SUM(SALDO)  
FROM CUENTAS  
GROUP BY TITULAR
```

**Ejemplo:** Supóngase que se desea conocer el nº de movimientos por cada cuenta bancaria registradas en el sistema.

En este caso, la consulta se hace sobre la tabla **MOVIMIENTOS** agrupando por el campo clave externa **CUENTA** y contabilizando los registros existente por cada cuenta mediante la función **COUNT()**.

```
SELECT CUENTA, COUNT(*)  
FROM MOVIMIENTOS  
GROUP BY CUENTA
```

MySQL define una serie de funciones que permiten obtener resultados a partir de agrupamientos. Algunas de las más comunes son:

- **COUNT()** → Devuelve el nº de registros agrupados
- **MAX()** → Devuelve el valor mayor de los agrupados.
- **MIN()** → Devuelve el valor menor de los agrupados
- **SUM()** → Devuelve el sumatorio de todos los valores agrupados.
- **AVG()** → Devuelve la media.

*(\*) Cuando se agrupa registros por el valor de un campo, sólo pueden mostrarse los campos por los que se agrupa, y funciones escalares que muestren un valor calculado en base al resto de campos.*

### Filtrado de agrupamiento

El agrupamiento por defecto muestra resultados para todos los valores del campo o campos por los que se agrupa. SQL permite filtrar los grupos a mostrar indicando una condición en la cláusula **HAVING**.

**Ejemplo:** Obtener el titular y el sumatorio del saldo de todas sus cuentas de aquellos clientes cuyo saldo total supere los 1000€.

```
SELECT TITULAR, SUM(SALDO) AS TOTAL  
FROM CUENTAS  
GROUP BY TITULAR  
HAVING TOTAL > 1000.0
```

### Sentencia **INSERT**

Esta sentencia permite insertar un registro en una tabla indicando el nombre de la tabla y los campos para los que se desea indicar un valor, seguido de los propios valores en el mismo orden tras la cláusula **VALUES**.

```
INSERT INTO <tabla>(
    <columna_1>, ..<columna_n>)
VALUES (
    <valor_para_columna_1>, ...<valor_para_columna_n> );
```

**Ejemplo:** Sentencia SQL que da de alta una nueva cuenta en la tabla CUENTAS.

<pre>INSERT INTO CUENTAS(     NUMERO_CUENTA, TITULAR, SALDO, INTERES, BLOQUEADA, APERTURA) VALUES     ('AC059603', 'ARTYOM DEMIDOV', 750.00, 0.04, 0, '2017-04-12 10:10:15')</pre>
<p>(*) Los campos para los que no se indique valor tomarán el valor predeterminado indicado en la definición de la tabla, o el valor nulo suponiendo que lo admitan. En caso de no ser así se producirá un error.</p>
<p>(*) NUNCA DEBE indicarse valor para un campo autonumérico en una sentencia INSERT. El valor lo asigna la propia base de datos.</p>

### Sentencia **UPDATE**

Esta sentencia permite modificar el valor de una o varias columnas de aquellos registros de una determinada tabla que cumplan cierta condición. El formato básico es:

```
UPDATE <table>
SET <columna_1> = <valor_1>, ... , <columna_n> = <valor_n>
WHERE <condicion>
```

La partícula **SET** indica los campos y los valores que se desean asignar a aquellos registros de la tabla indicada tras la cláusula **UPDATE** que cumplan la condición de la cláusula **WHERE**.

**Ejemplo:** Aumento de 250€ al valor del campo **SALDO** del registro cuenta con **NUMERO\_CUENTA** = 'AC059603':

<pre>UPDATE CUENTAS SET SALDO = SALDO + 250 WHERE NUMERO_CUENTA = 'AC059603'</pre>
<p>(*) La modificación del valor de un campo puede no ser posible si se trata de una clave primaria y tiene registros asociados, o si es una clave externa y se asigna un valor no existente en el campo clave primaria referenciado.</p>

### Sentencia **DELETE**

Esta sentencia elimina aquellos los registros de la tabla indicada que cumplan la condición indicada en la cláusula **WHERE**. El formato básico es:

```
DELETE FROM <tabla>  
WHERE <condicion>
```

El campo condición determina qué registros van a eliminarse.

**Ejemplo:** Eliminar el registro de la cuenta con *NUMERO\_CUENTA* 'AC059603'

```
DELETE FROM CUENTAS  
WHERE NUMERO_CUENTA = 'AC059603'
```

*(\*) El borrado de registros de una tabla puede no ser posible si el registro en cuestión tiene registros asociados en otras tablas. En este caso, si la cuenta indicada tiene registros asociados en la tabla MOVIMIENTOS, la eliminación no será posible por violar la regla de integridad referencial. Sería preciso en tal caso eliminar primeramente todos sus movimientos.*

## Prácticas

Crea una base de datos para una aplicación bancaria donde los empleados de una sucursal pueden registrar y manipular las cuentas bancarias de los clientes.

La base de datos debe constar de las siguientes tablas.

### **CLIENTES**

- *DNI* -> DNI del usuario
- *NOMBRE* -> Nombre de usuario

### **CUENTAS**

- *N\_CUENTA* -> Identificador de la cuenta ( tipo cadena )
- *DNI* -> DNI del titular de la cuenta
- *SALDO* -> Saldo de la cuenta
- *INTERESES* -> Interés aplicado
- *APERTURA* -> Fecha de apertura

### **USUARIOS**

- *USUARIO* → Cadena con el identificador del usuario
- *CLAVE* → Cadena con la clave del usuario
- *ADMINISTRADOR* → Indicador si el usuario es ordinario (0) o administrador(1)

Debe decidirse los tipos de datos apropiados para cada uno de los campos y crear la base de datos en MySQL empleando *phpMyAdmin*.

## ***Prácticas***

Crea una base de datos para una aplicación de tipo FORO donde los usuarios pueden publicar mensajes o responder a mensajes existentes.

La base de datos debe constar de las siguientes tablas.

### ***USUARIOS***

- *ID\_USUARIO* -> Autonumerico
- *LOGIN* -> Nombre de usuario
- *PASSWORD* -> Contraseña de acceso
- *MAIL* -> Dirección de correo electrónico

### ***FOROS***

- *ID\_FORO* -> Autonumérico
- *NOMBRE* -> Nombre del foro
- *ID\_USUARIO* -> ID\_USUARIO que lo creo
- *FECHA* -> Fecha de creación

### ***MENSAJES***

- *ID\_MENSAJE* -> Autonumérico
- *ID\_FORO* -> Foro al que pertenece el mensaje
- *ID\_USUARIO* -> Usuario que publicó el mensaje
- *MENSAJE* -> Cuerpo del mensaje
- *FECHA* -> Fecha de publicación
- *ID\_MENSAJE\_RESPUESTA* -> Mensaje al que este mensaje responde.

Debe decidirse los tipos de datos apropiados para cada uno de los campos y crear la base de datos en MySQL empleando phpMyAdmin.