PHP y MySQL

Aplicaciones Web/ Sistemas Web

Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid

Material bajo licencia Creative Commons

PHP

Breve introducción a MySQL con phpMyAdmin

Persistencia de datos

- La información se guarda para volver a utilizarla
- Mecanismos
  - Ficheros
    - Almacenamiento básico
    - Acceso secuencial o aleatorio
  - Bases de datos
    - Información estructurada
    - Relaciones
    - Búsquedas
    - Acceso concurrente
    - Control de acceso a la información
    - Integridad

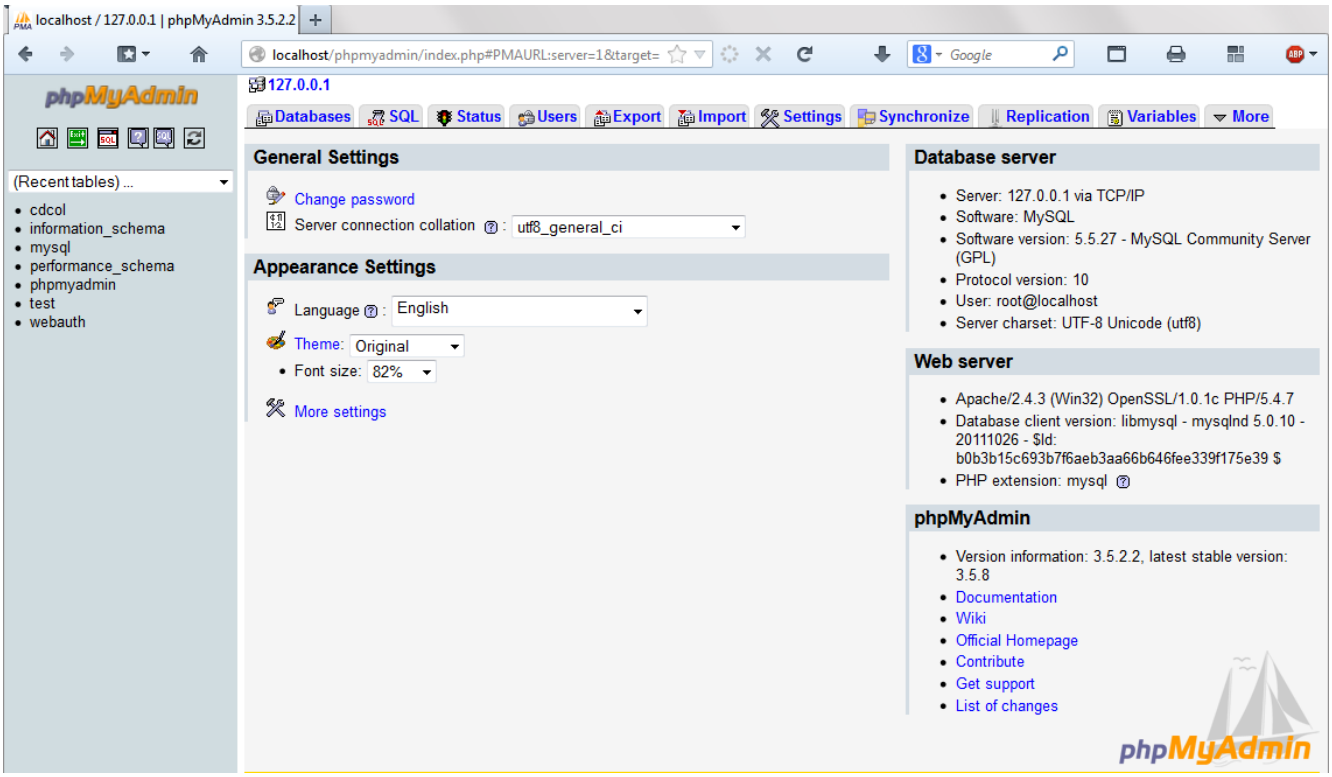
MySQL

- Sistema de Gestión de Base de Datos Relacional
  - La información se guarda en tablas
    - Una tabla es una colección de datos relacionados
    - Una tabla consta de columnas (campos) y filas (registros)
    - Las tablas se enlazan por relaciones entre columnas
- Implementa casi todo el estándar SQL (Structured Query Language)
- Código abierto
  - Actualmente de Oracle, que adquirió Sun, que tenía MySQL AB
- Escalable
  - Aplicaciones pequeñas y grandes (millones de registros)
- Transacciones, Multiusuario
- Eficiente: Multihilo, varias técnicas de hash, b-tree, etc.
- Conexión al servidor MySQL con sockets TCP/IP
  - Esto permite conectarla con casi cualquier plataforma

phpMyAdmin

- Herramienta que ofrece una interfaz gráfica para la administración del servidor MySQL
  - Configuración del servidor y las bases de datos
  - Gestionar (crear, modificar, borrar) las bases de datos, tablas, campos, relaciones, índices, etc.
  - Consultas con SQL, y mediante ejemplos (query by example)
  - Definir usuarios y asignar permisos
  - Realizar copias de seguridad
  - Crear gráficos (PDF) del esquema de la base de datos
  - Exportar a muchos formatos (documentos de texto, hojas de cálculo)
- En XAMPP se puede invocar en **`http://localhost/phpmyadmin/`**
- Configuración
  - Fichero **`config.inc.php`** (en el directorio raíz de phpMyAdmin)
  - Pero más recomendable a través de la interfaz web de phpMyAdmin en **`http://www.dominio.com/phpMyAdmin/setup`**
  - Usar el password del root de mySQL, que se aplica también a ese mismo usuario en phpMyAdmin (efectivo tras rearrancar mySQL)

phpMyAdmin – Página inicial

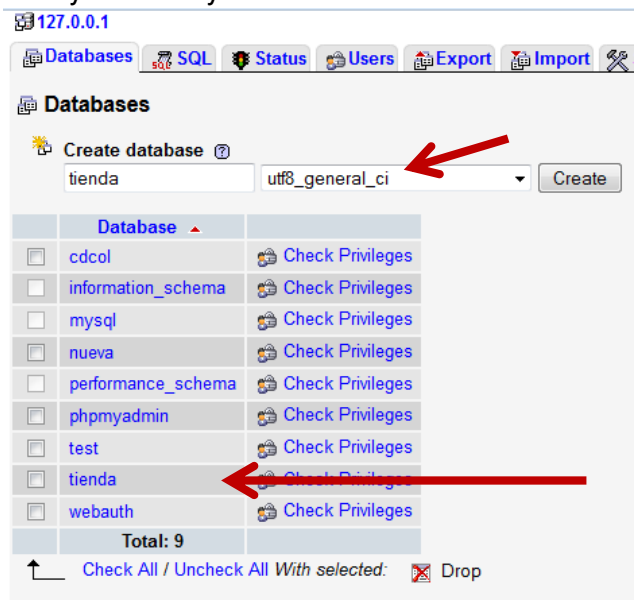


Usuarios de MySQL

- Conviene crear un nuevo usuario para cada sitio web
  - Cada sitio web tendrá sus propias bases de datos
  - El usuario root solo se debe usar para administración
- Entrar en phpMyAdmin como usuario root
  - En principio no tiene password, pero habrá que ponerlo, por seguridad
  - A continuación crear un nuevo usuario
    - por ejemplo para el sitio del ejemplo a continuación: **`tienda`**
    - Pestaña Users-> Add user
    - En la ventana que aparece indicar
      - Nombre de usuario: **`tienda`**
      - Host (local si estáis desarrollando con XAMPP en vuestro PC)
      - Password: **`tienda`**
      - Crear una base de datos para el nuevo usuario
        - Marcar la casilla *`Create database with same name and grant all privileges`*
      - No activar privilegios globales
  - Salir de la sesión como root
- Entrar con el nuevo usuario
  - Se puede trabajar con la nueva base de datos

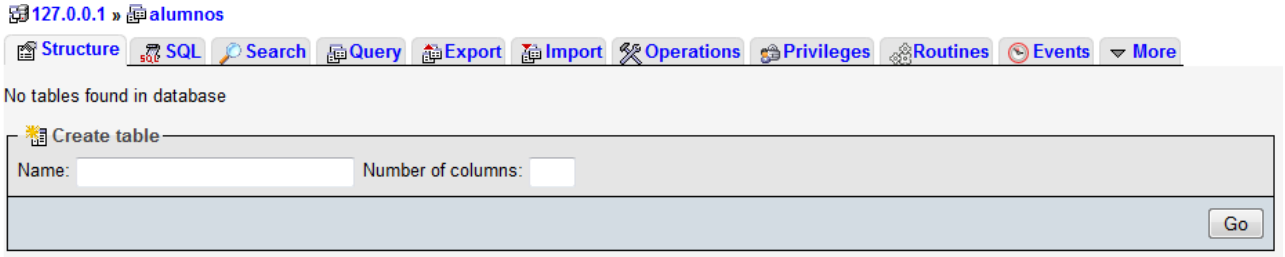
Creación de una base de datos con phpMyAdmin

- Como root: Create database en la pestaña Database
  - Indicar un nombre para la base de datos
  - La opción "collation" indica el conjunto de reglas de comparación y ordenación del texto en la base de datos, que dependerá del idioma
    - Por ejemplo, utf8_general_ci que vale para muchos idiomas y no es sensible a mayúsculas y minúsculas

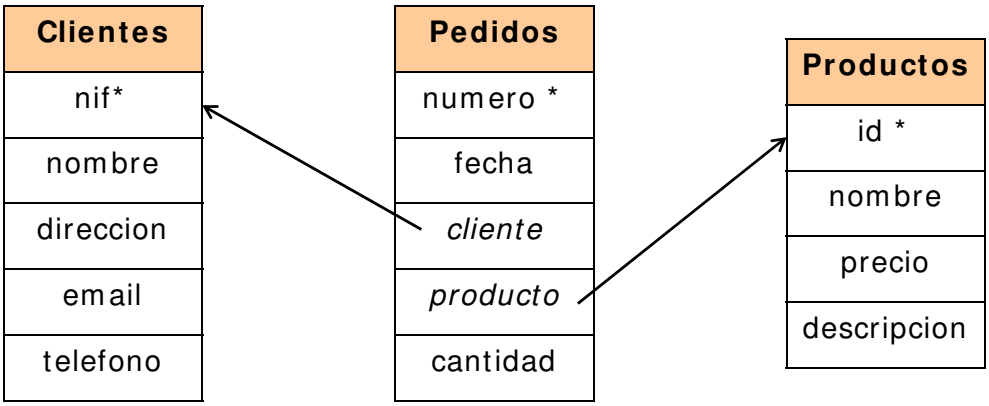


Manejo de la base de datos con phpMyAdmin

- Al seleccionar la base de datos creada aparecen las operaciones que se pueden realizar con ella
- Se pueden añadir permisos (pestaña *Privileges*) para que otros usuarios puedan usar la base de datos
- En Structure se pueden crear las tablas que definen el esquema de la base de datos



Ejemplo sencillo de base de datos: Tienda



* **Clave primaria** (los objetos en esta columna son únicos y no nulos). Será indexada.

** Se pueden definir también **índices** para mejorar la eficiencia de las búsquedas

*** Las **claves foráneas** (***foreign keys***) identifica una columna (o grupo de columnas) en una tabla que se refiere a otra columna (o grupo de columnas) en otra tabla, generalmente la clave primaria en la tabla referenciada.
Contribuyen a gestionar la *integridad de la base de datos*: no se puede crear un pedido de un cliente o un producto que no existan.
Las claves foráneas deberían indexarse porque se usarán para seleccionar registros con frecuencia.

Creación de tablas

- La base de datos consta de tablas
  - Cada una con una serie de columnas (campos)
  - Cada campo tendrá asociado un tipo:
    - Enteros: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT
    - Números reales: DECIMAL, DOUBLE, FLOAT, REAL
    - Booleanos: BOOLEAN
    - Fecha: DATE, TIME, YEAR
    - Strings: VARCHAR (hasta 256 caracteres), TEXT
  - Como Storage Engine conviene usar `InnoDB` para poder gestionar relaciones entre tablas
  - Como Collation conviene usar `utf8_general_ci`

Creación de tablas

- La primera tabla es la de clientes, con cinco campos
  - nif: servirá como primary key (el nif es único)
  - nombre: de empresa o de persona (sería nombre + apellidos)
    - Se puede indexar para hacer búsquedas por este campo
  - direccion
  - email
  - telefono: como string para permitir uso de caracteres no numéricos

Table name: clientes Add 1 column(s) Go

Structure

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I
nif	VARCHAR	10	None				PRIMARY	
nombre	VARCHAR	30	None				INDEX	
direccion	VARCHAR	50	None				---	
email	VARCHAR	30	None				---	
telefono	VARCHAR	15	None				---	

Table comments: Storage Engine: InnoDB Collation: utf8_general_ci

Creación de tablas

- Crear dos tablas más:
  - productos
  - pedidos

autoindex

Table name: productos Add 1 column(s) Go

Structure

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I
id	INT		None		UNSIGNED		PRIMARY	☑
nombre	VARCHAR	20	None				INDEX	
descripcion	VARCHAR	200	None				---	
precio	DECIMAL	6,2	None				---	

Table comments: | Storage Engine: InnoDB | Collation: utf8_general_ci

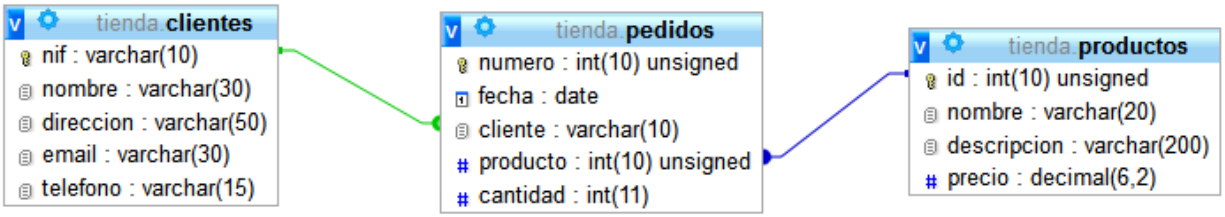
Table name: pedidos Add 1 column(s) Go

Structure

Name	Type	Length/Values [1]	Default [2]	Collation	Attributes	Null	Index	A_I
numero	INT		None		UNSIGNED		PRIMARY	☑
fecha	DATE		None				---	
cliente	VARCHAR	10	None				INDEX	
producto	INT		None		UNSIGNED		INDEX	
cantidad	INT		None				---	

Definir relaciones entre tablas

- Usar la pestaña Designer para ver gráficamente las tablas
  - Se pueden recolocar las tablas
  - Para añadir una relación seleccionar el botón Create relation
    - Seleccionar la primary key de la tabla clientes: nif
    - Seleccionar la foreign key en la tabla pedidos: cliente
    - Aparece una ventana para seleccionar qué hacer para preservar la integridad de las referencias, con las siguientes operaciones:
      - DELETE: seleccionar RESTRICT
      - UPDATE: seleccionar CASCADE
        - La restricción más adecuada en la mayoría de los casos es evitar realizar borrados en cascada y actualizar en cascada
    - La relación queda establecida y aparece en el gráfico
  - Para salvar el diagrama, usar el botón Save



Definir relaciones entre tablas

- Si falla el establecimiento de una relación, comprobar que:
  - Ambas usan tablas InnoDB como storage engine
  - No hay registros en las tablas
    - Si los hubiera hay que comprobar que concuerdan para mantener la integridad de las referencias
  - El campo en la primera tabla es una clave primaria
  - El campo correspondiente en la segunda tabla está indexado
  - Ambos campos tienen exactamente los mismos tipos de datos
    - Si son numéricos, ambos deben ser unsigned (o ninguno)
- Una vez que se haya establecido la relación no será posible introducir registros en la segunda tabla a menos que se correspondan en el campo de la relación con el de la primera tabla
  - ¿Qué ocurre en la segunda tabla cuando se borra un registro de la primera tabla?

Introducir datos en tablas

- Seleccionar la tabla
  - Seleccionar la pestaña Insert
  - Introducir datos para los campos correspondientes en Value
- También se pueden introducir a través del programa PHP que recupera la información que un usuario haya introducido en un formulario de una página web
- En la pestaña Browse se pueden ver los registros de la base de datos y modificar campos de los mismos

Backup de la base de datos

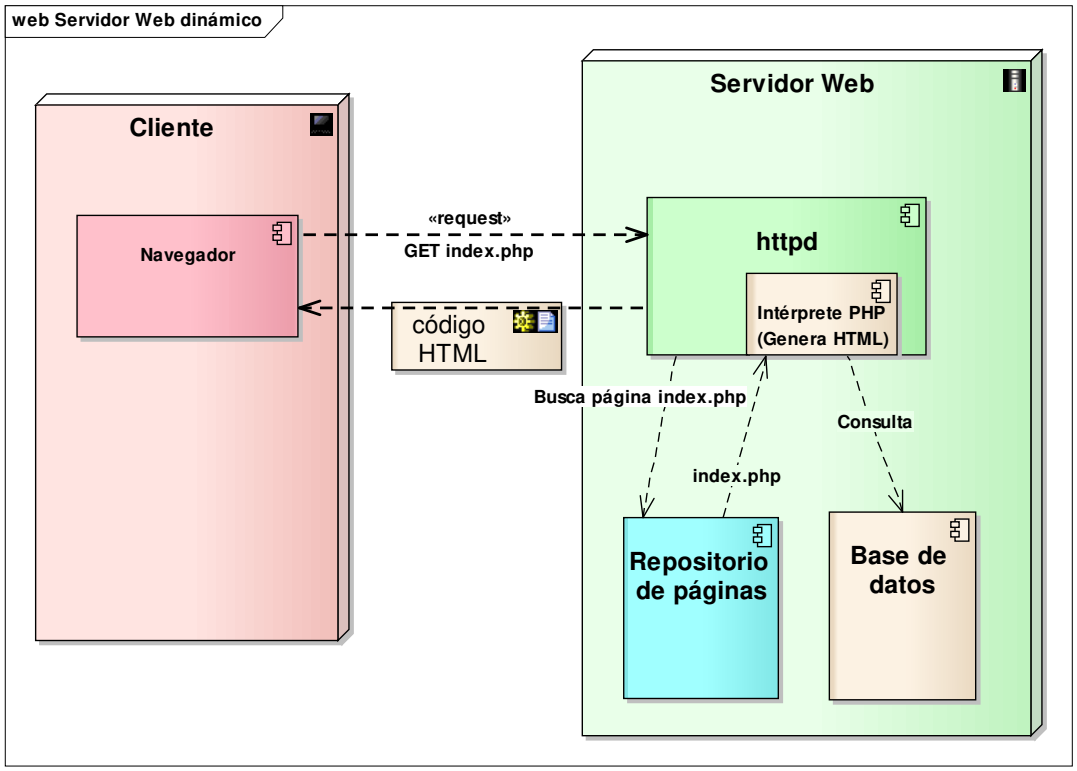
- **Export**
  - Conveniente de forma regular
    - Especialmente si se hacen muchos cambios
  - Opciones (seleccionar *Custom* en *Export Method*)
    - Qué se guarda
      - El servidor completo
      - Una base de datos entera
      - Una tabla
    - Estructura o datos, o ambos
    - Compresión: ninguna, zipped, gzipped, bzipped
    - Formato
      - SQL, CSV, Word, Latex, Excel, OpenDoc, PDF, XML, JSON, etc.
- El proceso inverso es posible con **Import**

Ejercicios con phpMyAdmin

- Instalar phpMyAdmin
- Crear la base de datos tienda con las tablas clientes, productos y pedidos tal como se han definido previamente
- Insertar elementos en las tres tablas, primero en clientes y productos y luego en pedidos
  - Comprueba qué ocurre si se intenta introducir un pedido para un cliente que no existe
- Intenta eliminar un cliente que tiene algún pedido
  - Observa el efecto de haber definido la política DELETE: RESTRICT cuando se estableció la FOREIGN KEY
- Intenta cambiar el nombre de un producto que tiene algún pedido
  - Observa el efecto de haber definido la política UPDATE: CASCADE cuando se estableció la FOREIGN KEY

Uso de MySQL con PHP

Acceso a la base de datos en PHP



Uso de una base de datos MySQL desde PHP

- Con PHP5 se recomienda utilizar la extensión **MySQLi** (*Mysql improved*) en vez de la tradicional **Mysql**
  - Permite utilizar las mejoras de las últimas versiones del servidor MySQL
  - Interfaz orientada a objetos
- Alternativa: PHP Data Objects (**PDO**)
  - Interfaz ligera para acceso a bases de datos, con soporte para MySQL y otros sistemas de gestión de bases de datos
    - Un driver específico para cada SGBD
  - Proporciona una capa de abstracción para el acceso a datos
    - Independiente del tipo de SGBD
      - No usa la sintaxis SQL
    - Orientado a objetos

Uso de una base de datos MySQL desde PHP

- Normalmente comprende los siguientes pasos:

1. **Conexión** con el servidor de bases de datos y selección de una base de datos
   - Se obtiene un objeto para operar con la base de datos
2. **Uso de la base de datos**
   - Envío de operación SQL a la base de datos
   - Recepción y tratamiento de los resultados
   - Liberar memoria de resultados
3. **Desconexión**

Conexión con la base de datos

- Para utilizar una base de datos hay que indicar el servidor y la base de datos que se quiere utilizar, con un usuario

```
$mysqli = new mysqli($hostname, $usuario, $password,$basededatos);
if ( mysqli_connect_errno() ) {
    echo "Error de conexión a la BD: ".mysqli_connect_error();
    exit();
}
```

  - Devuelve un objeto sobre el que operar con la base de datos
  - Si hubiera un error se comprueba con el método `mysqli_connect_errno()`

- Cuando se deja de utilizar la base de datos conviene cerrar la conexión al servidor para liberar recursos ordenadamente

```
$mysqli->close();
```


Operaciones SQL en una base de datos MySQL

- Las **queries SQL** se pasan con el método **query**

```
$mysqli->query("SQL query");
```

  - Devuelve un objeto que permite tratar los resultados
  - Devuelve FALSE si hay algún error
  - Si se ponen variables PHP en la query, se ponen entre comillas simples para que la función mysql_query las reemplace por su valor

```
$empresa="Empresa%";
$query="SELECT * FROM clientes WHERE nombre LIKE '$empresa'";

$resultado=$mysqli->query($query)
        or die ($mysqli->error. " en la línea ".(__LINE__-1));

$numregistros=$resultado->num_rows;
echo "<p>El número de clientes con nombre Empresa* es: ",$numregistros,".</p>";
```

Operaciones SQL en una base de datos MySQL

- Varios atributos y métodos de la clase **mysqli_result** facilitan el tratamiento de los registros obtenidos
  - **$num_rows**: Número de registros (filas)

    `$numfilas=$resultado->num_rows;`
  - **fetch_array()** o **fetch_all()**: Devuelve todas las filas en un array asociativo, numérico, o en ambos
    - **fetch_assoc()**: Lo mismo pero como array asociativo

    `$registro=mysqli->fetch_array([modo])`
    - Argumento opcional para indicar cómo se accede a los registros
      - Usando el nombre del campo como índice: MYSQL_ASSOC
      - Usando la posición como índice: MYSQL_NUM
      - Usando tanto el nombre de campo como la posisicón: MYSQL_BOTH
  - **free()**: Libera la memoria asociada al resultado

    `$resultado->free();`

Operaciones SQL en una base de datos MySQL

- Ejemplo: listado de la tabla clientes

```
$query="SELECT * FROM clientes";

$resultado=$mysqli->query($query)
        or die ($mysqli->error. " en la línea ".(__LINE__-1));

$numregistros=$resultado->num_rows;
echo "<p>El número de clientes con nombre Empresa* es: ",$numregistros,".</p>";

echo "<table border=2><tr><th>NIF</th> <th>Nombre</th> <th>Dirección</th>
<th>Email</th> <th>Teléfono</th></tr>";
while ($registro = $resultado->fetch_assoc()) {
        echo "<tr>";
        foreach ($registro as $campo)
                echo "<td>",$campo, "</td>";
        echo "</tr>";
}
echo "</table>";
$resultado->free();
```

SQL

- **SELECT**
  - Recupera elementos de una tabla o conjunto de tablas (con JOIN)

```
SELECT campos FROM tabla WHERE campo = valor
```

    - Si se quieren todos los campos, usar *
    - Si se omite la cláusula WHERE se tienen todos los campos de la tabla
    - Para la condición WHERE se pueden usar varios operadores:
      - = <> != < <= > >=
      - AND OR NOT
  - Se pueden recuperar campos de varias tablas

```
SELECT tabla1.campo1 tabla2.campo2 FROM tabla1, tabla2
   WHERE campo3=valor3 AND tabla1.campo1 = tabla2.campo2
```

  - También se pueden usar patrones para las condiciones
    - % indica cualquier subcadena

```
SELECT campos FROM tablas WHERE campo3 LIKE patron
```

    - Ejemplo: `SELECT nombre FROM clientes WHERE nombre LIKE Juan%`
  - Ordenar: ORDER BY
  - Para no tener registros duplicados: DISTINCT

```
SELECT DISTINCT campos FROM tablas WHERE ...
```


SQL

- **INSERT**
  - Inserta nuevos elementos en una tabla
    - Crea un nuevo cliente

```
INSERT INTO clientes (nif, nombre, direccion, email, telefono)
VALUES ("M3885337J", "Empresa Uno", "Calle Uno, Madrid",
"jefe@empresauno.com", "91 2347898")
```

- **UPDATE**
  - Actualiza campos de una tabla
    - Modifica el importe del producto "Producto1"

```
UPDATE productos SET precio = 399.99 WHERE nombre="Producto1"
```

- **DELETE**
  - Elimina registros de una tabla
    - Elimina pedidos con más de 30 días de antigüedad

```
DELETE FROM pedidos WHERE fecha < CURDATE()-10
```

Ejercicio

- Listar la información de la tabla pedidos indicando el nombre y NIF del cliente, y el coste de cada pedido (cantidad* precio del producto)
- Definir una función que liste todos los pedidos de un cliente dado por su nombre

Bibliografía

- Manual PHP oficial http://www.php.net/manual/es/index.php
- Sitio oficial de phpMyAdmin: http://www.phpmyadmin.net
- Introducción sencilla al uso de phpMyAdmin y php con MySQL
  - http://www.yourwebskills.com/dbbackground.php
  - http://www.yourwebskills.com/beginningmysql.php
- Libros
  - S. Suehring, T. Converse, J. Park. *PHP6 and MySQL Bible*. Wiley Pub. 2009
  - O. Heurtel. PHP y MySQL. *Domine el desarrollo de un sitio Web dinámico e interactivo*. Ediciones ENI 2009
  - M. Delisle. Dominar phpMyAdmin para una administración efectiva de MySQL. Packt Publishing (2007)

Apéndice: **Interfaz PHP Mysql tradicional**

- Normalmente comprende los siguientes pasos:
  1. Conexión con el servidor de bases de datos
     `mysql_connect()`
  2. Selección de una base de datos
     `mysql_select_db()`
  3. Uso de la base de datos
     - Envío de operación SQL a la base de datos
       `mysql_query()`
     - Recepción y tratamiento de los resultados
       `mysql_num_rows()`
       `mysql_fetch_array()`
       `mysql_free_result()`
  4. Desconexión del servidor de bases de datos
     `mysql_close()`

Conexión con la base de datos

- Para utilizar una base de datos hay que:
  - **Conectarse con el servidor** que la gestiona
    `$conexion = mysql_connect($servidor, $usuario, $password)`
    - Devuelve como resultado un recurso de tipo enlace (la conexión a la base de datos)
    - Si hubiera un error devuelve FALSE
      - En caso de error, si se quiere acabar el script se puede poner:

```
$conexion = mysql_connect($servidor, $usuario, $password) or
die("Error en conexión al servidor MySQL: ".mysql_error());
```

  - **Seleccionar la base de datos** en el servidor
    `mysql_select_db($basedatos, $conexion)`
    - La base de datos se especifica en la variable `$basedatos`
    - Devuelve TRUE si todo va bien, FALSE si hay algún error

```
mysql_select_db($bd, $conexion) or
die ("Error: No se puede usar la base de datos. ".mysql_error());
```

Operaciones SQL en una base de datos MySQL

- Las **queries SQL** se pasan con la función

  mysql_query("SQL query");
  - Devuelve un identificador o TRUE (dependiendo de la instrucción) si la instrucción se ejecuta correctamente
  - Devuelve FALSE si hay algún error
  - Si se ponen variables PHP en la query, se ponen entre comillas simples para que la función mysql_query las reemplace por su valor

```
$query="SELECT * FROM clientes";

$resultado=mysql_query($query) or die ("Error en la query: ".mysql_error());

$numregistros=mysql_num_rows($resultado);
echo "<p>El número de registros de clientes es: ",$numregistros,".</p>";
```


Operaciones SQL en una base de datos MySQL

- El conjunto de registros resultado del query se guarda en una variable
- Varias funciones facilitan el tratamiento de los registros obtenidos
  - **mysql_num_rows**: Devuelve el número de registros (filas)

    `$numfilas=mysql_num_rows($resultado)`
  - **mysql_fetch_array**: Devuelve el siguiente registro o FALSE si no hay más

    `$registro=mysql_fetch_array($resultado, [modo_de_acceso])`
    - Argumento opcional para indicar cómo se accede a los registros
      - Usando el nombre del campo como índice: MYSQL_ASSOC
      - Usando la posición como índice: MYSQL_NUM
      - Usando tanto el nombre de campo como la posisicón: MYSQL_BOTH
  - **mysql_free_results**: Libera la memoria una vez procesado el conjunto de registros

    `$registro=mysql_fetch_array($resultado)`

Operaciones SQL en una base de datos MySQL

- Ejemplo: listado de la tabla clientes

```
$query="SELECT * FROM clientes";

$resultado=mysql_query($query) or die ("Error en la query: ".mysql_error());

$numregistros=mysql_num_rows($resultado);
echo "<p>El número de registros de clientes es: ",$numregistros,".</p>";

echo "<table border=2><tr><th>NIF</th> <th>Nombre</th> <th>Dirección</th>
<th>Email</th> <th>Teléfono</th></tr>";

while ($registro = mysql_fetch_array ($resultado, MYSQL_ASSOC)) {
        echo "<tr>";
        foreach ($registro as $campo)
                echo "<td>",$campo, "</td>";
        echo "</tr>";
}
echo "</table>";

mysql_free_result($resultado);
```


Desconexión de la base de datos

- Cuando se deja de utilizar la base de datos conviene cerrar la conexión al servidor para liberar recursos ordenadamente

```
mysql_close($conexion);
```