Unidad 3: Acceso a bases de datos desde PHP

Índice

El mayor beneficio de trabajar con páginas dinámicas es que puede tomar la información que se mostrará de una base de datos, en lugar de tener que tenerla dentro del código o en un fichero externo. El uso de MySQL con PHP es muy popular en el desarrollo de aplicaciones web.

1. Gestores de bases de datos más usados

Existen diferentes bases de datos en el mercado que puede utilizar para crear su página web con bases de datos. Es importante conocer las más populares para poder decidir en cada caso cuál será la óptima para vosotros.

- **MySQL:** es una plataforma gratuita y muy adecuada para aprender. Es posiblemente una de las bases de datos más rápidas que podemos encontrar, además de consumir pocos recursos de la máquina donde está instalada. Se trata de un sistema relativamente fácil de instalar y administrar frente a otros productos del mercado. Se recomienda para iniciarse en el mundo de las bases de datos, ya que dispone de muchas utilidades, manuales y documentación que la inmensa comunidad de usuarios ha encargado de realizar desinteresadamente. La gestión de la base de datos utiliza el SQL (Structured Query Language), además de utilizar protocolos de comunicación de bases de datos desarrolladas por Microsoft, mediante ODBC o ADO.

  ORDBMS (Object-Relational Database Management System, sistema de gestión de bases de datos objeto-relacionales) es un sistema gestor de bases de datos similar a los relacionales, pero con un modelo de bases de datos objeto-relacionales con objetos, clases y herencia.

- **PostgreSQL:** es un gestor de bases de datos usado principalmente para grandes organizaciones en aplicaciones críticas o importantes. PostgreSQL es la segunda base de datos más popular implantada en el mercado GNU / Linux. PostgreSQL se diseñó como una base de datos orientada a objetos, es decir, una ORDBMS, con la que el concepto de tablas lo sustituimos por objetos, y se permite crear nuevos tipos de datos, hacer herencias entre objetos, etc. PostgreSQL es, sin duda, una de las bases de datos que ha aportado más implementaciones para los expertos. Dispone de lo que a MySQL le falta, pero también le falta todo lo que el MySQL tiene. PostgreSQL tiene transacciones, integridad referencial, vistas y un gran número de funcionalidades, a cambio de convertirse en un motor más lento y pesado que MySQL.

- **Oracle:** es uno de los sistemas de gestión de bases de datos relacional más importantes. Está disponible en una variedad de configuraciones, desde pequeñas versiones personales hasta versiones empresariales.

- **Microsoft Access:** es un sistema de pago que permite realizar un diseño de la base de datos mediante una herramienta gráfica. Tiene ciertas limitaciones importantes (como el número de conexiones concurrentes que puede tratar) y hay pocos sistemas operativos sobre los que puede funcionar.

- **Microsoft SQL Server:** está diseñado para crear webs y sistemas de bases de datos empresariales de escritorio. Permite almacenar grandes cantidades de información comparado con Access.

Utilizaremos MySQL porque: es ideal para aplicaciones pequeñas, aunque al ser escalable también se puede utilizar para aplicaciones grandes; es muy rápido, confiable y fácil de usar; utiliza SQL estándar; compila una gran cantidad de plataformas; y, finalmente, porque es gratuito de descargar y usar.

2. Gestión de bases de datos con PHPMyAdmin

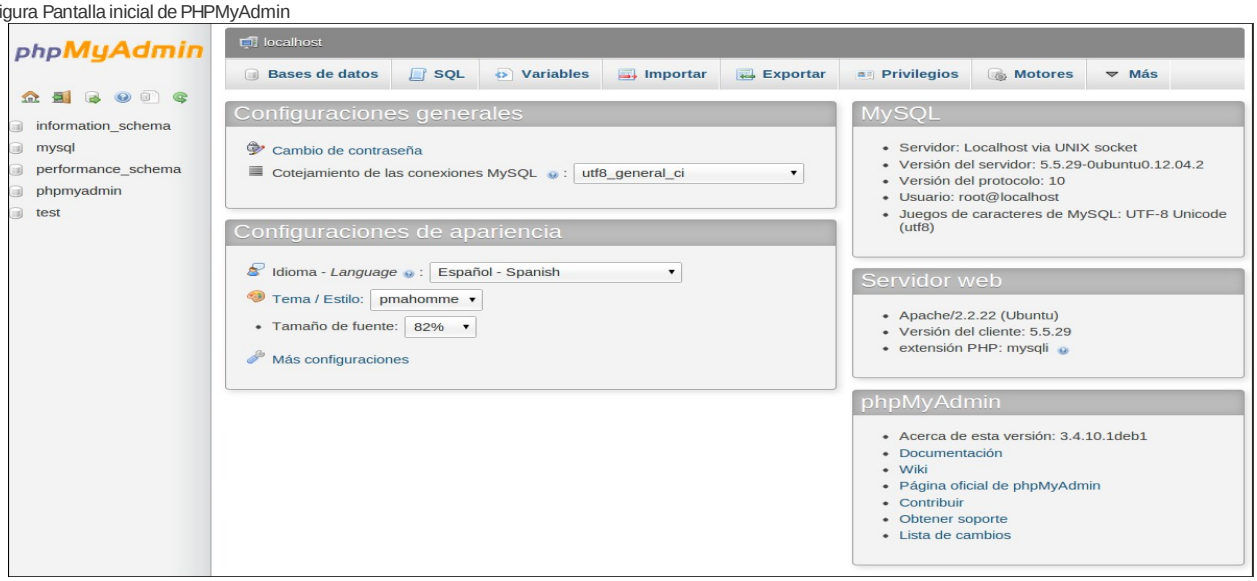
PHPMyAdmin es una herramienta libre y gratuita escrita en PHP que permite controlar la administración de MySQL desde un navegador web. Permite realizar diferentes tareas tales como crear, modificar o borrar bases de datos, tablas, campos o filas ... ejecutando órdenes SQL.

Si se accede con un usuario que sea administrador de MySQL, el phpMyAdmin permite gestionar todas las bases de datos existentes del servidor. De otro modo, sólo se muestran las bases de datos a las que se tiene acceso. Los usuarios administradores tienen acceso a una base de datos llamada mysql (base de datos de control) que sirve para configurar el motor de bases de datos. Es muy importante no tocarla ni hacer cambios a menos que se sepa perfectamente lo que está haciendo. En caso contrario, el MySQL podría dejar de funcionar o quedar inoperativo.

Para acceder a ella hay que entrar desde el navegador a la dirección http://localhost/phpmyadmin.

Una vez introducidos los datos de acceso (nombre y contraseña) debe ver una pantalla como la de la figura1.

Figura Pantalla inicial de PHPMyAdmin

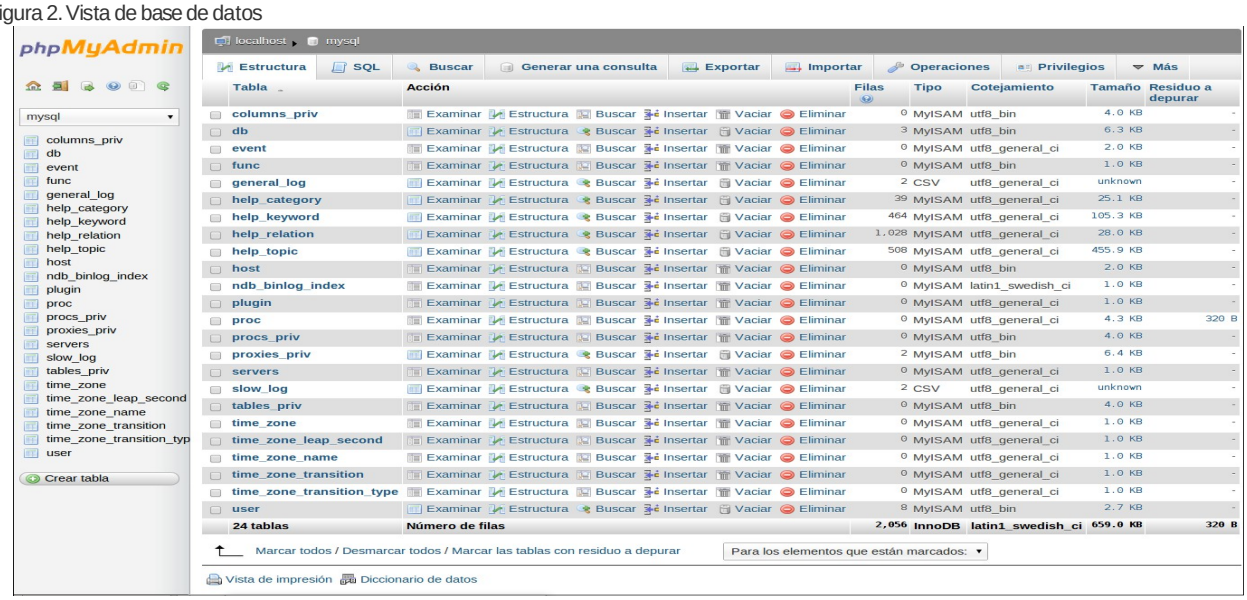


El phpMyAdmin divide la ventana en dos marcos. El de la izquierda ofrece un menú desplegable con las bases de datos a las que se tiene acceso (si son más de una) y las tablas de la base de datos seleccionada. El marco de la derecha es el lugar en el que se muestra la información de la navegación. Inicialmente muestra una pantalla de bienvenida con varios enlaces.

Al hacer clic en el nombre de las tablas que se muestran en el marco de la izquierda se puede ver la estructura (los campos) en el marco de la derecha y aparece un menú horizontal que permite acceder a las diversas funcionalidades del aplicación.

Si selecciona una base de datos existente (por ejemplo mysql) verá por defecto un listado de todas las tablas que existen en la base de datos, tal como se puede ver en la figura.2.

Figura 2. Vista de base de datos



Si hace clic en cualquiera de las tablas, puede ver la estructura de la tabla, examinar su registros o insertar uno nuevo.

Haciendo clic en la pestaña SQL puede insertar consultas directamente utilizando sintaxis SQL. Esto es muy útil cuando está programando, ya que puede comprobar si las consultas que está realizando durante el programa son correctas o no. Simplemente copie el texto de la consulta en la caja y haga clic en Continuar. Puede ver los registros resultantes de la consulta que ha hecho.

Finalmente, con la opción Exportar puede extraer los datos de la base de datos a un archivo de texto plano que puede importar posteriormente, con la opción Importar, a una base de datos MySQL.

3. Conexión con el servidor MySQL

La estructura para acceder a una base de datos es muy similar al protocolo para acceder a un archivo:

1. Abrir la base de datos.
2. Enviar el pedido SQL en la base de datos.
3. Devolver el resultado de la consulta.
4. Cerrar la conexión de la base de datos.

Antes de poder acceder a los datos en una base de datos, se debe crear una conexión con la base de datos. En PHP para conectar con la base de datos MySQL esto se hace con la función *mysql_connect().*

```
recurso = mysql_connect (nombre del servidor, nombre de usuario, contraseña, nuevo enlace, señales de cliente);
```

La función ***mysql_connect()*** devuelve una conexión con el servidor MySQL Server en caso de éxito, o FALSE y un error en caso de fallo. Puede ocultar la salida de error mediante la adición de una @ delante del nombre de la función.

Esta es la descripción de los parámetros:

- Nombre del servidor: es opcional. Especifica el servidor MySQL al que conectar. Puede incluir un número de puerto, por ejemplo Servidor: puerto o una ruta, por ejemplo,*: /camino/al/directorio* para el localhost. Si la directiva de PHP *mysql.default_host* no está definida (por defecto), entonces el valor por defecto es *localhost: 3306*. En el modo SQL seguro, este parámetro se ignora y *localhost: 3306* se utiliza siempre.

- Nombre de usuario: es opcional. Especifica el nombre de usuario para iniciar la sesión. El valor por defecto es el nombre del usuario propietario del proceso del servidor que está definido por la directiva *mysql.default_user*. En el modo SQL seguro, este parámetro se ignora y se usa el nombre del usuario propietario del proceso del servidor.

- Contraseña: es opcional. Especifica la contraseña para iniciar sesión. El valor predeterminado es definido por la directiva *mysql.default_password*. En el modo SQL seguro, este parámetro no se tiene en cuenta y se usa la contraseña vacía.

- Nuevo enlace: es opcional. Si se hace una segunda llamada a *mysql_connect ()* con los mismos argumentos, no se establece un nuevo enlace, en su lugar, se devolverá el identificador de enlace del enlace ya abierto. El parámetro nuevo enlace modifica este comportamiento y hace que *mysql_connect ()* siempre abre un nuevo vínculo, aunque se haya llamado a *mysql_connect ()*, con los mismos parámetros. En el modo SQL seguro, este parámetro se ignora.

- Señales de cliente: es opcional. El parámetro señales de cliente puede ser una combinación de las siguientes constantes: 128 (habilita el control de LOAD DATA LOCAL), MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE o MYSQL_CLIENT_INTERACTIVE. En el modo SQL seguro, este parámetro se ignora.

En el siguiente ejemplo se almacena la conexión en una variable ($conexion) para su uso posterior. Si la conexión falla ejecutará el código dentro del *if* y el programa saldrá.

```
 <? php
$conexion = mysql_connect("localhost", "admin", "12345");
   if (! $conexion)
   {
      exit('No se puede conectar:'. mysql_error ());
   }
  echo "Conexión correcta!";
?>
```

La función *exit()* muestra un mensaje y sale de la ejecución del código. La función *mysql_error()* devuelve un texto con un mensaje de error de la última operación MySQL que se ha ejecutado.

4. Cerrar la conexión a la base de datos

La función de PHP *mysql_close()*, cierra una conexión MySQL. Esta función devuelve TRUE si tiene éxito, o FALSE en caso de fallo.

Tiene un único argumento opcional que es la conexión MySQL para cerrar. Si no se especifica, se utiliza la última conexión abierta por *mysql_connect().*

El uso de *mysql_close()* no es imprescindible, ya que las conexiones no persistentes son cerradas automáticamente al final de la ejecución del script PHP.

Este es un ejemplo de utilización:

```
<?php
$conexion= mysql_connect("localhost", "admin", "12345");
if (!$conexion)
{
exit('No se puede conectar:'. mysql_error());
}
echo "Conexión correcta!<br>";
 mysql_close($conexion);
 echo "Conexión cerrada";
?>
```

5. Selección de la base de datos

Antes de empezar a trabajar con una base de datos del servidor, debemos seleccionar. Para ello, existe la función *mysql_select_db($nombre, $conexion*) que toma dos argumentos que especifican el nombre de la base de datos que queremos escoger, y la conexión al servidor que hemos hecho con anterioridad, respectivamente. La función devuelve TRUE en caso de éxito, y FALSE en caso de error.

He aquí un ejemplo de utilización de la función:

```
$conexion = mysql_connect("localhost", "usuari", "12345");
if (!$conexion)
{
exit('No se puede conectar:'. mysql_error());
}

if(!mysql_select_db("tabla_equipos", $conexion))
exit("Error al conectar con la base de datos". mysql_error());
```

6. Trabajo con la base de datos

Con una base de datos se pueden hacer varias operaciones: **consultar datos, insertar datos, actualizar datos y borrar datos.**

Para estas operaciones la estructura del código de programación es exactamente igual. Siempre que se programa una conexión a una base de datos MySQL se realizarán los siguientes pasos:

1. Conexión al servidor MySQL: para hacer esta operación se utiliza la función *mysql_connect($servidor, $usuario, $contraseña)*. La función devuelve un valor del tipo *resource (recurso)*, que representa la conexión con la base de datos.

2. Selección de la base de datos con la que se trabaja: con *mysql_select_db ($nom_BD, $conexion)*. Se debe indicar el nombre de la base de datos con la que se quiere trabajar y de la conexión con el servidor.

3. Hacer la consulta/operación necesaria con la función *mysql_query ($consulta).*

4. Puede comprobar cuántas filas ha devuelto un SELECT o han sido insertadas modificadas. Para consultar cuántas filas ha devuelto un consulta, puede utilizar la función *mysql_num_rows($resultado)* que devuelve el número de filas que se han seleccionado con un **SELECT o SHOW**. Para consultar cuántas filas han sido insertadas, modificadas o borradas con INSERT, UPDATE, REPLACE o DELETE, puede utilizar la función *mysql_affected_rows().*

5. Si es una operación de consulta de datos (como un SELECT), falta leer los datos que devuelve la consulta SQL. Para ello puede utilizar la función *mysql_fetch_array($resultado)* que devuelve los valores en forma de matriz asociativa en la que las claves son las columnas de la tabla. Devuelve FALSE si no hay más filas.

6.1 Inserción de información

Para añadir registros en la base de datos debe utilizar una consulta MySQL de tipo INSERT que tiene la siguiente estructura:

```
INSERT INTO nom_taula [(columna1, columna2, ...)]
    VALUES (valor1, valor2,...);
```

Puede consultar la sintaxis completa en:

http://dev.mysql.com/doc/refman/5.5/en/insert.html

Por ejemplo:

```
"INSERT INTO discos(titulo, autor, any) VALUES ('Axis: Bold as love', 'The Jimi
Hendrix Experience', '1967')";
```

El siguiente ejemplo muestra un código entero de inserción de datos en una base de datos:

```
$usuari = "admin";
$contrasenya = "admin";
// Connexió amb el servidor de base de dades i selecció de la base de dades
$connexio = mysql_connect("localhost", $usuari, $contrasenya);
if (!$connexio)
exit('No es pot connectar:'. mysql_error());
if(!mysql_select_db("llibreria", $connexio))
exit("Error al connectar amb la base de dades". mysql_error());
//Com que no inserirem totes les dades (id és autoincremental) s'han d'especificar els camps
$consulta = "INSERT INTO llibres (titol, autor, any) VALUES ('Norwegian Wood', 'Haruki Murakami', '2000')";
// Fem la consulta al servidor i obtenim la resposta
$resultat = mysql_query($consulta, $connexio);
if (!$resultat)
{
$missatge = 'Query incorrecta: ' . mysql_error() . "\n";
$missatge .= 'Query sencera: ' . $consulta;
exit($missatge);
}
echo "Dades introduïdes correctament!!";
```

Puede comprobar posteriormente que se ha insertado los datos (con una consulta MySQL o desde PHPMyAdmin, por ejemplo).

La gracia de las páginas dinámicas es que la información que se ha de insertar en la base de datos no es estática, como en este ejemplo, sino que son valores que introducen los usuarios. Esta tarea se hace mediante el uso de formularios. La secuencia de acciones es la siguiente:

1. El usuario introduce datos con un formulario.
2. Cuando valida el formulario con el botón, se envían los datos utilizando el método POST o GET.
3. Recibidos los datos del formulario se hace la conexión a la base de datos y la inserción de los datos.

6.2 Consulta de información

Para realizar una consulta en la base de datos para obtener información, debe utilizar una sentencia SELECT, que tiene la siguiente forma:

```
SELECT
    expressió
    FROM taula
    WHERE condició
    [GROUP BY {columna | expressió | posició}
    [ASC | DESC]]
    [HAVING condició]
    [ORDER BY {columna | expressió | posició}
    [ASC | DESC]]
    [LIMIT {[desplaçament,] num_files | num_files OFFSET offset}]
```

Puede consultar la sintaxis completa a:

http://dev.mysql.com/doc/refman/5.6/en/select.html

Por ejemplo:

```
SELECT nom, dni FROM usuaris WHERE uid=1
```

Una vez creada la consulta y almacenada en una variable, puede ejecutarla en base de datos con la función *mysql_query($consulta, $conexion).*

www.php.net/manual/en/function.mysql-query.php

Esta función:

- Para consultas del tipo SELECT, SHOW, Describe, Explain devuelve una variable de tipo *resource(recurso)* en caso de éxito y FALSE en caso de error.
- Para consultas INSERT, UPDATE, DELETE, DROP, retorna TRUE en caso de éxito y FALSE en caso de error.

Una vez que haya hecho la consulta SELECT, puede averiguar cuántas filas ha devuelto la consulta con la función *int mysql_num_rows ($resultado)* que devuelve el número de filas que ha devuelto el SELECT o FALSE en caso de error.

Para obtener los datos del resultado de la consulta utiliza la función *array mysql_fetch_array ($resultado, $tipo)* que devuelve un array asociativo que corresponde a una fila del resultado de la consulta (y avanza el puntero que indica la fila a mostrar).

El argumento $ tipo indica el tipo del array asociativo que se obtiene. Es una constante que puede coger los siguientes valores: MYSQL_ASSOC, MYSQL_NUM o MYSQL_BOTH (que es el valor

por defecto). Si deja este parámetro en blanco o seleccione MYSQL_BOTH obtendrá un array con el que puede acceder a los elementos tanto por número de la columna, como por su nombre.

La función devolviendo las diferente filas resultantes del SELECT según como es llamada, hasta que llega a la última fila resultante, en la que devuelve FALSE. Por lo tanto, para obtener todos los datos de retorno de un SELECT, debe llamar a *mysql_fetch_array* dentro de un bucle hasta que la función le devuelva FALSE. Por ejemplo:

```
$resultado = mysql_query("SELECT id, nom FROM taula");


        while ($fila = mysql_fetch_array($result, MYSQL_BOTH))
        {
                printf("ID: %s Nom: %s", $fila[0], $fila[1]);
                //com fem servir MYSQL_BOTH, és igual a
                printf("ID: %s Nom: %s", $fila["id"], $fila["nom"]);
        }
```

En este otro ejemplo se muestran los contenidos dentro de una tabla.

```
// Conexión con el servidor de base de datos y selección de la base de datos
$connexio = mysql_connect("localhost", $usuari, $contrasenya);

if (!$connexio)
exit('No es pot connectar:'. mysql_error());

if(!mysql_select_db("llibreria", $connexio))
exit("Error al connectar amb la base de dades". mysql_error());

//Definim la consulta que recuperarà totes les dades de la taula llibres
$consulta="SELECT * FROM llibres";

//Fem la consulta al servidor i obtenim la resposta
$resultat = mysql_query( $consulta, $connexio);

//Si tenim resultats els mostrarem
if( mysql_num_rows($resultat) > 0)
{

//Creem una taula per posar els resultats
echo "<table border = '1'>";
echo "<tr>";
echo "<td>id</td>";
echo "<td>titol</td>";
echo "<td>autor</td>";
echo "<td>editorial</td>";
echo "</tr>";

// Mentre hi hagi resultats els afegim a la taula
while($fila = mysql_fetch_array($resultat))
{

//Hi afegim una filera i recuperem els valors utilitzant la matriu associativa que guardem a $fila a cada iteració
echo "<tr>";
echo "<td>".$fila['id']."</td>";
echo "<td>".$fila['titol']."</td>";
echo "<td>".$fila['autor']."</td>";
echo "<td>".$fila['editorial']."</td>";
echo "</tr>";
}

//fi de la taula
echo "</table>";
```

```
}
else
{
echo "No hi ha cap resultat";
}
```

6.3 Modificación de información

Para codificar la información de la base de datos debe utilizar la consulta UPDATE de MySQL que tiene la siguiente forma:

```
UPDATE taula
        SET nomColumna1={expressió1|DEFAULT},
        nomColumna2={expressió2|DEFAULT}...

        WHERE condició
        [LIMIT numero_files]
```

Puede ver la sintaxis completa a:

http://dev.mysql.com/doc/refman/5.6/en/update.html

Por ejemplo:

```
UPDATE taula SET id='200' WHERE id=2;


o
UPDATE taula1 SET col1 = col1 + 1, col2 = col1;
```

No hay que poner las comillas a la condición WHERE.

La utilizará mediante la función *mysql_query($consulta)*. Posteriormente puede utilizar la función *mysql_affected_rows()* para obtener la cantidad de filas modificadas con la sentencia.

En el siguiente ejemplo se modifican los datos en una base de datos en función de los valores enviados por un formulario HTML.

```
//Connexió amb la base de dades i selecció base dades llibreria
        $enllac = connexioBaseDades();

        if ( isset($_POST['actualitza']) )
        {
        //L'usuari ha decidit enviar al servidor les noves dades actualitzades
        $consulta = "UPDATE llibres SET titol='".$_POST['titol']."', autor='" . $_POST['autor']. "', editorial='".
        $_POST['editorial']."' WHERE id=".$_POST['id'];

        $resultat = mysql_query( $consulta, $enllac);

        $err = mysql_error();
        if( $err != "" )
        echo "error=$err <br>";

        printf("Registres modificats: %d\n", mysql_affected_rows());
        }
```

6.4 Borrar registros

Para borrar registros de una tabla debe utilizar una consulta DELETE que tiene la siguiente forma:

```
DELETE FROM taula
        [WHERE condició]
        [ORDER BY ...]
        [LIMIT numero_files]
```

Puede encontrar la sintaxis entera en:

http://dev.mysql.com/doc/refman/5.6/en/delete.html

Por ejemplo:

```
DELETE FROM taula WHERE id < 10
```

La utilizará mediante la función mysql_query ($ consulta). Posteriormente podemos utilizar la función mysql_affected_rows () para obtener la cantidad de filas modificadas con la sentencia.

En el siguiente ejemplo se borra un elemento especificado por un formulario HTML de la tabla libros:

```
if( isset($_POST['id']) )
        {
        //Definim la consulta que esborra el registre seleccionat
        $consulta="DELETE FROM llibres WHERE id =".$_POST['id'];
        //Fem la consulta al servidor i obtenim la resposta
        $resultat = mysql_query($consulta, $connexio);

        //Mireu una forma alternativa de veure si hi ha hagut un error
        $err = mysql_error();
        if( $err != "" )
        echo "error=$err <br>";

        printf("Registres esborrats: %d\n", mysql_affected_rows());
        }
```

La función *mysql_error($conexion)* devuelve un texto con la descripción del error de la última función MySQL que se ha ejecutado. Por tanto, la debe ejecutar justamente después de la función de la cual desea saber el error.

En el caso de que no haya ningún error, devuelve una cadena vacía ("").

El parámetro conexión es opcional. Especifica la conexión MySQL. Si no se especifica, se utiliza la última conexión abierta por *mysql_connect ().*

La función *mysql_errno* devuelve el número de error de la sentencia SQL anterior.

En este ejemplo puede ver cómo tratar una serie de errores trabajando con bases de datos:

```
<?php
        $connexio = mysql_connect("localhost", "usuari", "contrasenya");
        mysql_select_db("bd_inexistent", $connexio);
        echo mysql_errno($connexio) . ": " . mysql_error($connexio). "\n";
        mysql_select_db("coses", $connexio);
        mysql_query("SELECT * FROM bd_inexistent", $connexio);
        echo mysql_errno($connexio) . ": " . mysql_error($connexio) . "\n";
?>
```

7. Creación de una base de datos

Hay varias formas de crear una base de datos. Generalmente la creaban con una herramienta como puede ser PHPMyAdmin, aunque lo puede hacer también desde el código.

Para crear una base de datos en MySQL se utiliza la sentencia:

```
mysql> CREATE DATABASE Nom_base_de_dades;
```

En ambientes Unix, los nombres de las bases de datos son case sensitive (al contrario que las palabras clave), por lo que siempre debe referirse a la base de datos exactamente igual (y no *nom_base_de_dades o NOM_BASE_DE_DADES*). Esto también se aplica a los nombres de tablas. Esta restricción no existe en Windows, aunque puede utilizar el mismo esquema de mayúsculas cuando se refiera a bases de datos y tablas en una consulta dada.

El pedido *mysql_query()* de PHP envía una sentencia en la base de datos. Por lo tanto puede utilizarla para crear una base de datos en el servidor:

```
<?php
        $sql = 'CREATE DATABASE LaMevaBD';

        //$con conté la connexió a la BD
        if (mysql_query($sql, $con))
        {
        echo "S'ha creat la Base de dades LaMevaBD\n";
        } else {
        echo 'Error al crear la base de dades: ' . mysql_error() . "\n";
        }
        ?>
```

Al crear una base de datos, ésta no se selecciona para su uso automáticamente, se debe hacer explícitamente. La sintaxis para hacerlo a MySQL es

```
mysql> USE Nom_base_de_dades
```

Las bases de datos sólo necesitan ser creadas una sola vez, pero deben ser seleccionadas cada vez que se inicia una sesión de MySQL. Se puede hacer a través del pedido USE como se muestra en el ejemplo. Cuando acceda a MySQL desde consola lo puede indicar en la línea de comandos al ejecutar MySQL. Simplemente se debe indicar el nombre de la base de datos después de los parámetros que necesita para ingresar. Por ejemplo:

```
shell> mysql -h host -o user -p LaMevaBD
```

Adviértase que en el pedido anterior LaMevaBD no es la contraseña. Si se quisiera suministrar la contraseña en la línea de comandos, después de la opción-p, debe hacerse sin dejar espacios en blanco (por ejemplo, -pcontrasenya, no -p contraseña). De todas formas, escribir la contraseña en la línea de comandos no es recomendable porque lo expone a la vista de otros usuarios.

Para eliminar una base de datos en PHP, al igual que para la creación de una base de datos, utilizará el comando *mysql_query()* para ejecutar la instrucción DROP DATABASE, tal como se puede ver en el siguiente ejemplo:

```
<? php
 $ query = 'DROP DATABASE LaMevaBD';
 $ result = mysql_query ($ query, $ con);
?>
```

Puede ver el contenido de las tablas con el orden de MySQL SHOW TABLES directamente sobre MySQL:

```
mysql> SHOW TABLES;
Empty septiembre (0.00 sec)
```

En un principio la base de datos recién creada está vacía.

7.1 Creación de tablas

En MySQL puede utilizar la sentencia CREATE TABLE para especificar la estructura de una tabla:

```
CREATE TABLE nombre_tabla
(
 nom_columna1 tipus_dades,
 nom_columna2 tipus_dades,
 nom_columna3 tipus_dades,
)
```

Siempre que desea ejecutar órdenes al MySQL puede utilizar la función de PHP mysql_query (). En este caso se debe ejecutar la instrucción de MySQL CREATE TABLE, como se puede ver en el siguiente ejemplo:

```
<?php
        // Seleccionareu la Base de Dades
        mysql_select_db("LaMevaBD", $con);
        $sql = "CREATE TABLE nom_taula
        (
        nom varchar(15),
        cognom varchar(15),
        anys int
        )";

        // Executareu la consulta
        mysql_query($sql, $con);
        ?>
```

Después de crear la tabla , desde MySQL con el comando SHOW TABLES debería producir la siguiente salida :

```
mysql> SHOW TABLES;
+ ---------------------+
| Tables in LaMevaBD |
+ ---------------------+
nom_taula |
+ ---------------------+
```

Para verificar que la tabla ha sido creada en la forma esperada , utilice la sentencia Describe a MySQL :

```
mysql> DESCRIBE nom_taula;
+ ----- + ----- + ---- -+ –-- + –-------- + –---- +
| Field | Type | Null | Key | Default | Extra |
+ ----- + ----- + ---- -+ –-- + –-------- + –---- +
| col1  | tipus | ca   |      | NULL   |       |
| col2  | tipus | ca   |      | NULL   |       |
| col3  | tipus | ca   |      | NULL   |       |
+ ----- + ----- + ---- -+ –-- + –-------- + –---- +
```

La sentencia *DESCRIBE* puede ser utilizada en cualquier momento, por ejemplo, si olvida los nombres o el tipo de datos de las columnas de la tabla.

Para terminar verá un ejemplo donde se crea la base de datos *Alumnes_IOC* y se añade una tabla alumnos con tres columnas nombre , apellido y años.

```php
<?php

$dbhost = 'localhost';
$dbusuari = 'root';
$dbcon = 'contrasenya';
$con =mysql_connect($dbhost, $dbusuari,$dbcon);

if (!$con)
{
exit('Error al connectar amb la base de dades de mysql: ' . mysql_error());
}

$sql = 'CREATE DATABASE Alumnes_IOC';

if (mysql_query($sql, $con))
{
echo "S'ha creat la Base de dades Alumnes_IOC\n";
}
else
{
echo 'Error al crear la base de dades: ' . mysql_error() . "\n";
}

// Seleccionareu la Base de Dades
mysql_select_db("Base_de_Dades", $con);
$sql = "CREATE TABLE alumnes
(
nom varchar(15),
cognom varchar(15),
anys int
) ";

// Executareu la consulta
mysql_query($sql,$con);

// Tancareu la connexió
mysql_close($con);

?>
```

8. Mecanismos de seguridad

La seguridad es un tema complejo que requiere una planificación a todos los niveles en que se está trabajando la página web y la base de datos. Por un lado usted utilizará contraseñas que lo más adecuado es que guarde de forma encriptada en la base de datos.

Los usuarios únicamente deben poder acceder a las páginas de la web a las que tienen permisos. Por ejemplo, un usuario alumno del IOC no tiene acceso a la edición del aula.

Por otra parte, los usuarios de la base de datos, deben tener únicamente la capacidad de acceder a los datos estrictamente necesarios. Los usuarios dentro de la base de datos también tienen un acceso limitado a ella.

8.1 Codificación de la información

Puede utilizar el algoritmo MD5 para codificar los datos confidenciales dentro de la base de datos. Puede encontrar la especificación aquí:

www.faqs.org/rfcs/rfc1321.html

Afortunadamente MD5 está implementado directamente en PHP, por eso puede calcular el MD5 de una cadena simplemente llamando a una función.

La función string *md5($cadena)* encripta la cadena utilizando el algoritmo MD5 y la devuelve en forma de número hexadecimal de 32 caracteres.

El uso del algoritmo SHA1 o MD5 no es estrictamente seguro para almacenar los datos, tal como se puede ver en este artículo de PHP.net *(www.php.net/manual/en/faq.passwords.php#faq.passwords .fasthash*), pero para hacer nuestras pruebas tendrá una seguridad suficiente y servirá para aprender a trabajar con claves encriptadas.

El siguiente es un ejemplo muy sencillo en el que se ve el funcionamiento de la función:

```
<?php

        $str = 'apple';

        if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f')
        {
        echo "Tens un ordinador Apple";
        }
        ?>
```

En el siguiente ejemplo se ve cómo crear una consulta donde se comprueba si el valor de nombre y contraseña obtenido de un formulario está en la base de datos :

```
$consulta = sprintf('select * from usuaris where nom="%s" and passwd = "%s"', $_POST["nom"], $_POST["passwd"]);
```

En caso de que las llaves estuvieran encriptados en la base de datos, se podría crear la consulta así :

```
$consulta = sprintf('select * from usuaris where nom="%s" and passwd = "%s"', $_POST["nom"],
md5($_POST["passwd"]));
```

La función *sprintf( )* de PHP funciona de forma similar a la versión de C. Es como un *printf*, pero en lugar muestra el resultado por pantalla, lo devuelve a un string . Os puede resultar muy útil para crear consultas. Puede ver aquí la especificación completa de la función :

www.php.net/manual/en/function.sprintf.php

En este caso , recuerde que al crear un usuario , su clave también deberá estar encriptada. por ejemplo

```
$consulta = sprintf('INSERT into usuaris (nom, passwd, administrador) VALUES ("%s","%s","%s
```

8.2 Acceso restringido a páginas

Una característica muy habitual de las webs actualmente es la de presentar diferentes contenidos de la web dependiendo de cuál sea el perfil del usuario. También hay ciertas páginas que pueden estar protegidas para impedir el acceso a los usuarios que no tienen los privilegios necesarios.

Por ejemplo, un usuario que no está registrado en www.amazon.com puede navegar y ver los productos que están a la venta . Pero no podrá realizar compras a menos que esté registrado, y que haya iniciado una sesión. De igual manera, un usuario registrado no podrá acceder a las páginas de administración de la web si no tiene los permisos necesarios.

Para implementar estos permisos de acceso puede utilizar sesiones o almacenar en la base de datos la estructura de páginas de la web y qué usuarios tienen acceso .

8.3 Seguridad en la base de datos

Las listas de control de acceso (ACL en inglés Access Control List) son el mecanismo de seguridad que utiliza MySQL para todas las conexiones y consultas que los usuarios pueden intentar realizar. Una lista de control de acceso controla si un usuario tiene permisos para acceder a un objeto. Las sentencias GRANT y REVOKE se utilizan para controlar el acceso de los usuarios a MySQL. Los permisos pueden darse en varios niveles:

- Nivel global: los permisos globales se aplican a todas las bases de datos de un servidor dado. Se guardan en la tabla mysql.user.
- Nivel de base de datos: los permisos de base de datos se aplican a todos los objetos en una base de datos especificada. Se guardan en las tablas mysql.db y mysql.host.
- Nivel de mesa: los permisos de mesa se aplican a todas las columnas en una tabla dada. Se guardan en la tabla mysql.tables_priv.
- Nivel de columna: los permisos de columna se aplican a columnas en una tabla dada. Se guardan en la tabla mysql.columns_priv.
- Nivel de rutina: los permisos de rutina se aplican a rutinas almacenadas. Se guardan en la tabla mysql.procs_priv.

Por ejemplo, para dar el permiso SELECT sobre toda la base de datos al usuario existente francesc, escribirás el siguiente comando:

```
GRANT SELECT ON base_dades.* TO francesc@'localhost';
```

Para dar más de un permiso a la vez, se deben separar las opciones con una coma. Así que para otorgar los permisos SELECT, INSERT y DELETE al usuario francesc , escribirás el siguiente comando :

```
GRANT SELECT, INSERT, DELETE ON base_dades.* TO francesc@'localhost';
```

Una vez que haya terminado debe refrescar los permisos de la memoria ejecutando el comando flush privileges .

Aquí puede encontrar toda la sintaxis del funcionamiento de la sentencia GRANT :

http://dev.mysql.com/doc/refman/5.1/en/grant.html

A continuación se muestra un ejemplo completo de gestión de permisos de acceso:

Primero entrar en la consola MySQL como root escribiendo el siguiente comando ya continuación introduzca la contraseña :

```
# mysql -p -u root

Enter password:
```

Una vez en la consola de comandos de MySQL , creará la base de datos base_de_dades escribiendo la siguiente sentencia :

```
mysql> CREATE DATABASE base_de_dades CHARACTER SET utf8 COLLATE utf8_bin;

Query OK, 1 row affected (0.01 sec)
```

Si ahora pedimos al MySQL que nos muestre las bases de datos, base_de_dades aparecerá en la lista.

```
mysql> SHOW DATABASES;

+--------------------+
| DATABASE |
+--------------------+
| information_schema |
| base_de_dades |
| mysql |
| phpmyadmin |
+--------------------+
4 rows IN SET (0.00 sec)
```

Ahora crearéis un usuario llamado ***usuari1*** con la contraseña *123456*.

```
 mysql> CREATE USER 'usuari1'@'%' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.00 sec)
```

Primero dan el permiso de SELECT a la base de datos *base_de_dades* para el usuario *usuari1*.

```
GRANT SELECT ON base_de_dades.* TO 'usuari1'@'%';
Query OK, 0 rows affected (0.00 sec)
```

Para comprobar los permisos de un usuario se utiliza el comando ***show grants for usuari1.***

```
mysql> SHOW grants FOR usuari1
    -> ;
+-----------------------------------------------------------------------------------------+
| Grants FOR usuari1@%
|
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'usuari1'@'%' IDENTIFIED BY PASSWORD
'*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9' |
| GRANT SELECT ON `base_de_dades`.* TO 'usuari1'@'%'
|
+-----------------------------------------------------------------------------------------+
2 rows IN SET (0.00 sec)
```

Se comprueba como aparece el permiso de SELECT, ahora añadimos los permisos de DELETE y INSERT al usuario *usuari1*.

```
mysql> GRANT DELETE,INSERT ON base_de_dades.* TO 'usuari1'@'%';
Query OK, 0 rows affected (0.00 sec)
```

Se comprueba los privilegios con el comando ***show grants*** y se ve como el usuario *usuari1* tiene los privilegios SELECT, INSERT, DELETE en la base de datos *base_de_dades*.

```
mysql> SHOW grants FOR usuari1;
+-------------------------------------------------------------------------------
------------------------+
| Grants FOR usuari1@%
|
+-------------------------------------------------------------------------------
------------------------+
| GRANT USAGE ON *.* TO 'usuari1'@'%' IDENTIFIED BY PASSWORD
'*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9' |
| GRANT SELECT, INSERT, DELETE ON `base_de_dades`.* TO 'usuari1'@'%'
|
+-------------------------------------------------------------------------------
------------------------+
2 rows IN SET (0.00 sec)
```

Para acabar y refrescar los permisos de la memoria es necesario ejecutar el comando ***flush privileges:***

```
mysql> FLUSH privileges;
Query OK, 0 rows affected (0.00 sec)
```

9. Verificación y pruebas

Para verificar el funcionamiento de MySQL puede utilizar software de monitorización que controla la disponibilidad de servicios. Para monitorizar el servicio de base de datos MySQL debe instalar conectores en el software. Entre las herramientas de software de monitorización libres que existen hablaremos de :

- Nagios
- Pandora FMS

9.1 Nagios

Nagios (www.nagios.com) es un software muy popular de monitorización de redes, ordenadores y aplicaciones. Está licenciado bajo la GNU (General Public License Version 2 publicada por la Free Software Foundation) y es de código abierto.

Se puede configurar para que monitorice los equipos (hardware) y servicios (software) que el

usuario quiera, alertando cuando el comportamiento de un recurso no sea correcto o supere los márgenes definidos por el administrador de red. Las alertas se pueden enviar entre otros por correo electrónico y mensajes SMS.

Entre sus características principales figuran la monitorización de servicios de red (SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH); de sistemas de hardware (carga del procesador, uso de disco, memoria, estado de los puertos, archivos de log, temperatura ...); remota mediante túneles SSL cifrados o SSH.

También ofrece la posibilidad de programar complementos específicos para cualquier parámetro de interés de un sistema.

Puede encontrar diferentes plugins para monitorizar bases de datos MySQL y servidores. Puede encontrar un listado aquí:

```
http://exchange.nagios.org/directory/Plugins/Databases/MySQL
```

Aquí tiene un listado de algunos de los más populares:

- Elemento de lista de *picscheck_mysql* (http://oss.isg.inf.ethz.ch/nagiosplug): comprueba y muestra el estado de las conexiones a un servidor MySQL.
- *check_mysqld* (http://william.leibzon.org/nagios): permite establecer umbrales para las variables del pedido ***SHOW STATUS*** de MySQL.
- *check_mysql_health* (http://labs.consol.de / nagios / check_mysql_health): permite controlar muchos parámetros de una base de datos, tales como el tiempo de conexión, número de conexiones, índices, la tasa de éxito de la caché , las consultas lentas, etc.

9.2 Pandora FMS

Pandora FMS (en inglés pandora Flexible Monitoring System, Sistema de Monitorización Flexible Pandora, *http://pandorafms.com*) es un software de código abierto publicado bajo licencia GPL2 GNU (General Public License), que sirve para monitorizar y medir todo tipo de elementos. Monitoriza de forma visual a lo largo del tiempo el estado de sistemas, aplicaciones o dispositivos.

Pandora FMS puede recoger información de cualquier sistema operativo, con software específico para cada plataforma: GNU / Linux, AIX, Solaris, HP-UX, BSD / IPSO y Windows 2000, XP y 2003.

Pandora FMS también puede monitorizar cualquier tipo de servicio de red tales como: TCP / IP, balanceadores de carga, routers, switches, sistemas operativos, aplicaciones o impresoras.

Para verificar el funcionamiento de MySQL existen módulos del propio software y conectores desarrollados. Entre los módulos encontramos:

- MySQL Monitoring: realiza un seguimiento general de MySQL.
- Mysql Cluster manager proccess: permite hacer un seguimiento de los procesos de MySQL.
- Mysql Active Connections: permite verificar las conexiones activas de MySQL.

9. 3 Pruebas de rendimiento

Algo muy importante en cualquier desarrollo web son las pruebas de rendimiento. Con ellas podrá evaluar atributos cuantitativos y cualitativos de su sistema como la fiabilidad o la escalabilidad, ya sean a nivel de software o hardware. Dentro de las pruebas, las bases de datos deben ser algo fundamental.

Mysqlslap es una herramienta de diagnóstico de MySQL disponible desde la versión 5.1.4 que le permitirá realizar estas pruebas. *Mysqlslap* emula la carga de un gran número de clientes accediendo a un servidor MySQL y ofrece informes una vez terminada la prueba.

Mysqlslap puede generar sentencias SQL automáticamente si no se pueden utilizar las sentencias que ejecuta el sistema.

Para invocar mysqlslap puede hacerlo de esta manera:

```
shell> mysqlslap [opciones]
```

Puede encontrar aquí un listado de las diferentes opciones de *mysqlslap:*

http://dev.mysql.com/doc/refman/5.1/en/mysqlslap.html

Mysqlslap se ejecuta en tres etapas: preparación, ejecución y limpieza. En la preparación se crean las tablas y las consultas para la prueba. En la ejecución se usan más de una conexión de clientes para ejecutar la prueba. En la limpieza se eliminan las tablas si se especifica y se desconectan los clientes.

Un ejemplo lo tenemos en:

Especifique la consulta SQL (SELECT * FROM tabla) y cree la tabla (CREATE TABLE tabla (clave int); INSERT INTO tabla VALUES (365)), especifique 50 clientes y 200 consultas seleccionadas para cada uno:

```
shell> mysqlslap -uroot -p --delimiter=";" \

  --create="CREATE TABLE taula (clau int);INSERT INTO taula VALUES
(365)" \
  --query="SELECT * FROM taula" --concurrency=50 --iterations=200
```

Una vez entrada la contraseña del usuario root mostrará unas estadísticas con los tiempos (medio, mínimo y máximo) de ejecución de todas las consultas:

```
Benchmark

        Average number of seconds TO run ALL queries: 0.021 seconds
        Minimum number of seconds TO run ALL queries: 0.009 seconds
        Maximum number of seconds TO run ALL queries: 0.900 seconds
        Number of clients running queries: 50
        Average number of queries per client: 1
```

Especifique que *mysqlslap* cree la instrucción de consulta SQL a una tabla de dos columnas INT y tres columnas VARCHAR. Utilice cinco clientes de consulta 20 veces cada uno. No cree la mesa

con la consulta para insertar los datos (es decir, se usará el esquema de la prueba anterior y los datos) :

```
shell> mysqlslap -uroot -p --concurrency=5 --iterations=20 \

        --number-int-cols=2 --number-char-cols=3 \
        --auto-generate-sql
```

Una vez entrada la contraseña del usuario *root*:

```
Enter password:

Benchmark

   Average number of seconds TO run ALL queries: 0.016 seconds

  Minimum number of seconds TO run ALL queries: 0.009 seconds

  Maximum number of seconds TO run ALL queries: 0.106 seconds

  Number of clients running queries: 5

 Average number of queries per client: 0
```

Creará dos archivos uno de creación (*creacio.sql*) con las sentencias (CREATE TABLE y INSERT INTO) y otro de consultas (*consultes.sql*) con sentencias (SELECT) . Primero ejecutará el archivo de creación y luego se ejecutará el archivo con las consultas. Específicos cinco clientes, cinco veces cada uno:

```
mysqlslap -uroot -p --concurrency=5 \

  --iterations=5 --query=query.sql --create=create.sql \
  --delimiter=";"
```

Resultado:

```
Enter password:
Benchmark
  Average number of seconds TO run ALL queries: 0.037 seconds
        Minimum number of seconds TO run ALL queries: 0.037 seconds
    Maximum number of seconds TO run ALL queries: 0.037 seconds
    Number of clients running queries: 5
   Average number of queries per client: 0
```