	RAMA:	Informática	CICLO:	DAM	
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial			CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:
	UNIDAD	COMPETENCIA	PHP. Conexión con bases de datos		

Índice

1.	Introducción a las bases de datos	2
2.	Conexión a MySQL.....	3
2.1.	Crear y seleccionar una base de datos	3
2.2.	Para crear y seleccionar una base de datos:	3
2.3.	Crear una tabla	5
2.4.	Insertar datos en una base de datos	6
2.5.	Asegurar datos de las consultas	7
2.6.	Recuperar datos de una base de datos	9
2.7.	Eliminar datos de una base de datos.....	10
2.8.	Actualizar datos en una base de datos.....	12

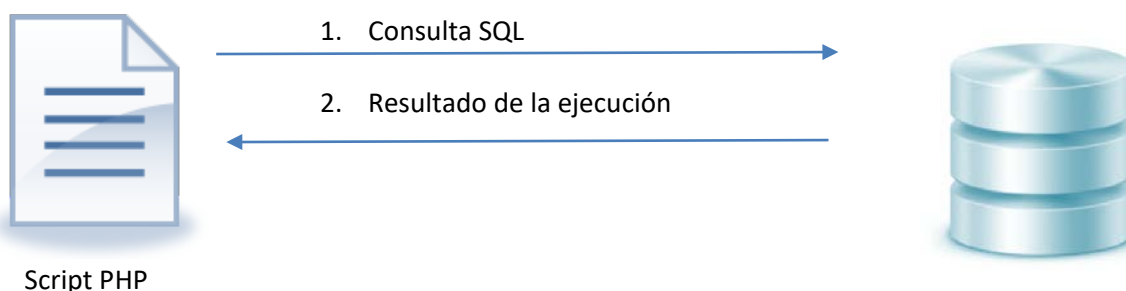
COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM	
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial			CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:
	UNIDAD COMPETENCIA	PHP. Conexión con bases de datos			

1. Introducción a las bases de datos

Una base de datos es una colección de tablas que almacenan información. Las bases de datos se pueden crear, actualizar y leer utilizando SQL (*Structured Query Language*). Los comandos más importantes que vamos a utilizar en la utilización de las bases de datos son:

Comando	Propósito
CREATE	Crea una base de datos o tabla
DELETE	Elimina registros de una tabla
DROP	Elimina una base de datos o tabla
INSERT	Agrega registros a una tabla
SELECT	Recupera registros de una tabla
UPDATE	Actualiza registros en una tabla

Cuando incorporamos una base de datos al diseño de páginas web, se utiliza PHP para enviar sentencias SQL a la aplicación de bases de datos, donde son ejecutadas. El resultado de esa ejecución ya sea la creación de una tabla, la inserción de un registro, la recuperación de algún/os registro/s o incluso un error se devuelven por la base de datos al script PHP.



Se utilizará PHP para enviar una sentencia SQL a MySQL. MySQL ejecutará esa sentencia y devolverá el resultado al script PHP.

Las herramientas que vamos a utilizar para conectarnos a una base de datos son:

1. El cliente MySQL. Una herramienta de línea de comandos para interactuar con MySQL

```

C:\WINDOWS\system32\cmd.exe - mysql
C:\>cd xampp\mysql\bin
C:\xampp\mysql\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.5.32 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

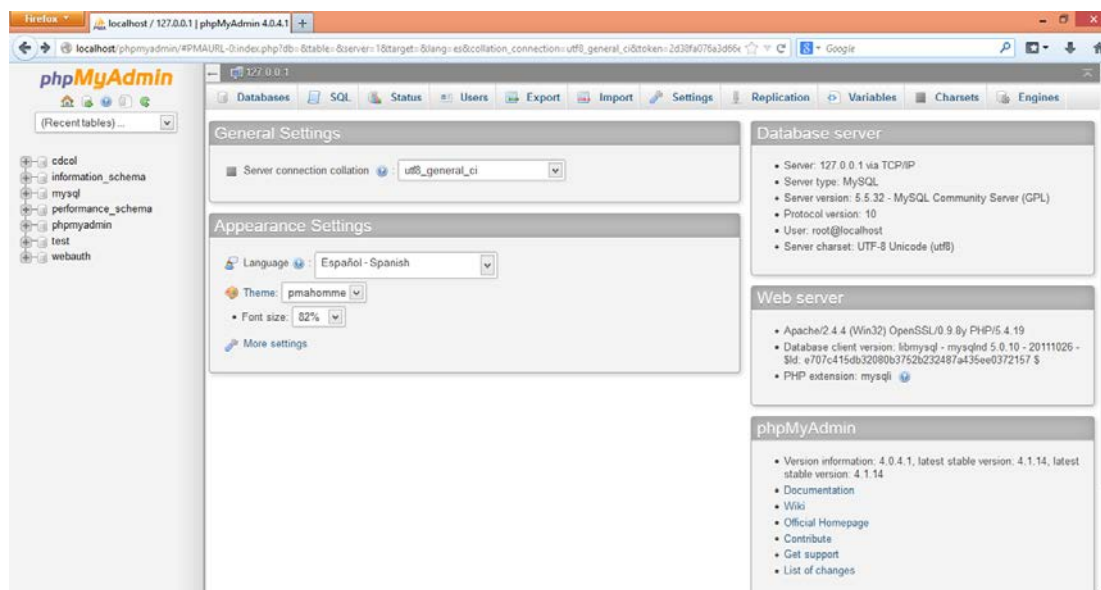
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
  
```

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial			CURSO:	1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	
	UNIDAD	COMPETENCIA	PHP. Conexión con bases de datos			

2. PHP MyAdmin. Una interfaz MySQL basada en PHP



2. Conexión a MySQL

Lo primero que tenemos que hacer es establecer una conexión al servidor de bases de datos (en este caso MySQL). Esta conexión se utilizará como punto de acceso para futuros comandos. La sintaxis para conectarse al servidor de bases de datos es:

```
$dbc=mysqli_connect(hostname, username, password);
```

La conexión a la base de datos (\$dbc) se establece utilizando tres argumentos:

- nombre del servidor. En nuestro caso será el servidor local localhost
- nombre de usuario
- contraseña para ese nombre de usuario (opcional)

Una vez que hayamos terminado de trabajar con la base de datos, es una buena idea considerar cerrar la conexión, para ello teclearemos:

```
mysqli_close($dbc);
```

2.1. Crear y seleccionar una base de datos

Antes de que un script pueda interactuar con una base de datos, primero se debe seleccionar la base de datos y ésta debe existir.

Para crear una base de datos con PHP utilizaremos la función `mysqli_query()` con el comando SQL `CREATE DATABASE (databasename)`:

```
mysqli_query('CREATE DATABASE basedatos');
```

Una vez hecho esto podemos seleccionar la base de datos utilizando `mysqli_select_db()`:

La base de datos sólo se creará una vez y deberá ser seleccionada antes de ejecutar cualquier consulta sobre ésta.

2.2. Para crear y seleccionar una base de datos:

1. Nos conectamos al servidor de bases de datos
2. Después de la sentencia `print()`, crearemos la nueva base de datos.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial		
	PROTOCOLO:	Apuntes clases	AVAL:	2
	UNIDAD	COMPETENCIA	DATA:	
	PHP. Conexión con bases de datos			
	CURSO: 1			

3. Por último seleccionamos la base de datos


```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Creación de la base de datos</title>
  </head>
  <body>
    <?php
      //Conexión a mysql
      if ($dbc = mysqli_connect('localhost', 'root')) {
        print '<p>Conexión exitosa </p>';
        //Creación de la base de datos
        if (mysqli_query($dbc, 'CREATE DATABASE myblog')) {
          print '<p>La base de datos ha sido creada!</p>';
        } else {
          print '<p style="color:red;">No ha sido posible crear la base de datos porque: <br/>' .
            mysqli_error($dbc) . '</p>';
        }

        //Seleccionamos la base de datos
        if (mysqli_select_db($dbc,'myblog')) {
          print '<p> La base de datos ha sido seleccionada </p>';
        } else {
          print '<p style="color:red;">No ha sido posible seleccionar la base de datos porque: <br/>' .
            mysqli_error($dbc) . '</p>';
        }
        mysqli_close($dbc);
      } else {
        print '<p style="color:red;">No ha sido posible realizar la conexión: <br/>' .
          mysqli_error($dbc) . '</p>';
      }
    ?>
  </body>
</html>
```

Para crear la base de datos, si es necesario (generalmente no crearemos una base de datos utilizando un script PHP), ejecutaremos utilizando la función `mysqli_query()` la consulta `$dbc, 'CREATE DATABASE myblog'`. Este paso no es necesario si la base de datos ha sido creada con anterioridad.

Para seleccionar la base de datos utilizamos la función `mysqli_select_db($dbc,'myblog')`.

Si PHP puede seleccionar la base de datos se muestra un mensaje, de lo contrario se muestra el error específico MySQL. Cada script que ejecuta consultas sobre una base de datos debe estar conectado a la misma y debe seleccionar la base de datos para trabajar con ella.

	RAMA:	Informática	CICLO:	DAM			
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial				CURSO:	1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:		
	UNIDAD COMPETENCIA		PHP. Conexión con bases de datos				

2.3. Crear una tabla

Una vez que la base de datos está creada y seleccionada, podemos comenzar a crear tablas. Para este ejemplo se creará una tabla individual y se almacenarán datos en la misma.

La consulta SQL adecuada para crear una tabla será:

```
CREATE TABLE nombretabla (columna1 definicion, columna2 definicion, etc)
```

Para cada columna, separada por comas, se indica primero el nombre de la columna y después el tipo de columna. Los tipos comunes son TEXT, VARCHAR, DATETIME e INT (entero). Se recomienda principalmente crear una primera columna que actúe como clave primaria. Ejemplo:

```
CREATE TABLE mi_tabla( id INT PRIMARY KEY, información TEXT)
```

Para el ejemplo vamos a crear una tabla que almacenará la información enviada a través de un formulario HTML.

La tabla que se creará en este ejemplo se describe en el cuadro siguiente:

Nombre de columna	Tipo de columna
Entry_id	Positivo, no nulo, entero incrementado automáticamente
title	Texto hasta 100 caracteres
entry	Texto de cualquier longitud
date_entered	Una marca de tiempo que incluye la fecha y hora en la que se agregó una fila

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Crear tabla</title>
  </head>
  <body>
    <?php
//Conectarse y seleccionar la base de datos
if ($dbc = mysqli_connect('localhost', 'root')) {
  if (!mysqli_select_db($dbc,'myblog')) {
    print '<p style="color:red;">No ha sido posible seleccionar la base de datos porque: <br/>' .
mysqli_error($dbc) . '</p>';
    mysqli_close($dbc);
    $dbc = FALSE;
  }
} else { //fallo en la conexión
  print '<p style="color:red;">No ha sido posible realizar la conexión: <br/>' .
    mysqli_error($dbc) . '</p>';
}
if ($dbc) {
  $query = 'CREATE TABLE entries(
    entry_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    entry TEXT NOT NULL,
    date_entered DATETIME NOT NULL
  )';
```

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial		
	PROTOCOLO:	Apuntes clases	AVAL:	2
	UNIDAD	COMPETENCIA	DATA:	
		PHP. Conexión con bases de datos		
			CURSO:	1

```
//Ejecutar la consulta
if (mysqli_query($dbc, $query)) {
    print '<p>La tabla ha sido creada</p>';
} else {
    print '<p style="color:red;">No ha sido posible crear la tabla porque: <br/>'.
        mysqli_error($dbc) . '</p>';
}
mysqli_close($dbc);
}
?>
</body>
</html>
```

2.4. Insertar datos en una base de datos

En este ejemplo se utilizará la base de datos como un blog. Las entradas del blog consistirán en un título y texto. Se agregarán a la base de datos utilizando una página y luego se mostrarán en otra página.

En el apartado anterior hemos creado una tabla que tenía cuatro columnas: `entry_id`, `title`, `entry` y `date_entered`. Para insertar registros en una tabla utilizaremos el comando `INSERT SQL` de cualquiera de las siguientes formas:


```
INSERT INTO nombretabla VALUES (valor1, valor2, valor3, etc)
```

```
INSERT INTO nombretabla (nombre_col1, nombre_col2) VALUES (valor1, valor2)
```

La consulta empieza con `INSERT INTO nombretabla`, después se pueden especificar las columnas que se desean insertar y las que no.

Los valores se colocan entre paréntesis, con cada valor separado por una coma. Los valores no numéricos, cadenas y fechas necesitan ir entre comillas, los números no.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insertar Entrada</title>
  </head>
  <body>
    <?php
      /* Este script agrega una entrada a una base de datos */
      if (isset($_POST['submitted'])) {
        //Conectar y seleccionar la base de datos
        $dbc = mysqli_connect('localhost', 'root');
        mysqli_select_db($dbc, 'myblog');
        //Validamos los datos del formulario
        $problema = FALSE;
        if (!empty($_POST['title']) && !empty($_POST['entry'])) {
            $titulo = trim($_POST['title']);
            $entrada = trim($_POST['entry']);
        } else {
            print '<p style="color:red;">Por favor introduzca un título y una entrada </p>';
            $problema = TRUE;
        }
      }
    }
  </body>
</html>
```

	RAMA:	Informática	CICLO:	DAM
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial		
	PROTOCOLO:	Apuntes clases	AVAL:	2
	UNIDAD	COMPETENCIA	DATA:	
PHP. Conexión con bases de datos				
CURSO: 1				

```

if (!$problema) {
    //Definimos los datos a introducir
    $query = "INSERT INTO entries (entry_id, title, entry, date_entered) VALUES (0, '$titulo', '$entrada',
NOW())";
    //Ejecutar la consulta
    if (mysqli_query($dbc, $query)) {
        print '<p> Entrada añadida</p>';
    } else {
        print '<p style="color:red;">No ha sido posible añadir la entrada porque: <br/>' .
            mysqli_error($dbc) . '</p><p>El query que se estaba ejecutando era' . $query . '</p>';
    }
} //todo OK
mysqli_close($dbc);
} //Fin envío formulario
?>
<form action="InsertarEntrada.php" method="post">
    <p>Título de la entrada: <input type="text" name="title" size="40"/></p>
    <p>Texto: <textarea name="entry" cols="40" rows="5"></textarea></p>
    <input type="submit" name="submit" value ="Añadir Entrada"/>
    <input type="hidden" name="submitted" value="true"/>
</form>
</body>
</html>

```

2.5. Asegurar datos de las consultas

En versiones antiguas de mysql el código del ejemplo anterior tendría una brecha de seguridad. Tal y como está, si alguien envía un texto que contiene un apóstrofe, ese dato interrumpirá la sentencia SQL. El resultado obviamente no es el deseado, pero, ¿por qué es inseguro?

Si usuarios mal intencionados saben que pueden interrumpir una consulta escribiendo un apóstrofe, pueden intentar ejecutar consultas utilizando esta brecha de seguridad. Si alguien envía `'DROP TABLE entries;` como título del blog la consulta resultante sería:


```
INSERT INTO entries (entry_id, title, entry, date_entered) VALUES (0, ' 'DROP
TABLE entries;', '<entry text>' NOW())
```

El apóstrofe inicial proporcionado, completa esa parte de la consulta haciendo que la consulta completa sea sintácticamente incorrecta. La brecha aquí es la nueva consulta proporcionada `DROP TABLE entries` que se ejecutará cuando falle la consulta original `INSERT`. A esto se le llama un **ataque de inyección SQL**, y son fáciles de prevenir.

Para prevenir estos ataques, se envían los datos potencialmente inseguros que se utilizarán en una consulta a través de la función `mysqli_real_escape_string()`. Esta función saldrá de cualquier carácter potencialmente dañino haciendo que los datos sean seguros para utilizarlos en una consulta. La sintaxis es:

```
$var=mysqli_real_escape_string($var);
```

Aplicándolo al código anterior queda:

	RAMA:	Informática	CICLO:	DAM
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial		
	PROTOCOLO:	Apuntes clases	AVAL:	2
	UNIDAD	COMPETENCIA	DATA:	
PHP. Conexión con bases de datos				
CURSO: 1				

```

<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insertar Entrada</title>
  </head>
  <body>
    <?php
      /* Este script agrega una entrada a una base de datos */
      if (isset($_POST['submitted'])) {
        //Conectar y seleccionar la base de datos
        $dbc = mysqli_connect('localhost', 'root');
        mysqli_select_db($dbc, 'myblog');
        //Validamos los datos del formulario
        $problema = FALSE;
        if (!empty($_POST['title']) && !empty($_POST['entry'])) {
          $titulo = mysqli_real_escape_string($dbc, trim($_POST['title']));
          $entrada = mysqli_real_escape_string($dbc, trim($_POST['entry']));
        } else {
          print '<p style="color:red;">Por favor introduzca un título y una entrada </p>';
          $problema = TRUE;
        }
        if (!$problema) {
          //Definimos los datos a introducir
          $query = "INSERT INTO entries (entry_id, title, entry, date_entered) VALUES (0, '$titulo', '$entrada',
NOW())";
          //Ejecutar la consulta
          if (mysqli_query($dbc, $query)) {
            print '<p> Entrada añadida</p>';
          } else {
            print '<p style="color:red;">No ha sido posible añadir la entrada porque: <br/>' .
              mysqli_error($dbc) . '</p><p>El query que se estaba ejecutando era' . $query . '</p>';
          }
        }
      } //todo OK
      mysqli_close($dbc);
    } //Fin envío formulario
  ?>
  <form action="InsertarEntrada.php" method="post">
    <p>Título de la entrada: <input type="text" name="title" size="40"/></p>
    <p>Texto: <textarea name="entry" cols="40" rows="5"></textarea></p>
    <input type="submit" name="submit" value="Añadir Entrada"/>
    <input type="hidden" name="submitted" value="true"/>
  </form>
</body>
</html>

```

Nota: Si se observa que la entrada del blog tiene barras invertidas adicionales antes de los apóstrofes, puede ser causado por una versión antigua de PHP. Si este es el caso se necesitará aplicar la función `stripslashes()` para eliminar las barras diagonales de los valores enviados. Ejemplo:

```
$titulo = mysqli_real_escape_string(stripslashes(trim($_POST['title'])));
```


COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial			CURSO:	1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	
	UNIDAD	COMPETENCIA	PHP. Conexión con bases de datos			

2.6. Recuperar datos de una base de datos

Para recuperar datos de una base de datos debemos asignar la información recuperada a una variable con el fin de utilizarla.

La sintaxis básica para recuperar datos es la consulta SELECT:

```
SELECT campos_a_consultar FROM nombre_tabla
```

La consulta más sencilla para leer datos de una tabla es:

```
SELECT * FROM nombre_tabla
```

El asterisco es equivalente a decir “todas los campos”. Si solamente queremos ciertos campos podemos limitar la consulta de la siguiente forma:

```
SELECT title, entry FROM entries
```

Esta consulta solicita solamente la información de los campos title y entry (título y entrada). Hay que tener en cuenta que no limita qué registros serán devueltos. Podemos restringir la consulta utilizando la cláusula WHERE:

```
SELECT * FROM usuarios WHERE (nombre='Carmen')
```

En esta sentencia indicamos que queremos la información de todas las columnas de la tabla, pero solamente de las filas donde el contenido de la columna nombre sea igual a Carmen.

Otra opción es añadir la cláusula LIKE.

```
SELECT * FROM usuarios WHERE nombre LIKE '%per';
```

Si añadimos LIKE, busca los campos que contienen la cadena ‘per’ en su valor

- ‘per%’: busca las cadenas que **comiencen** por ‘per’
- ‘%per’: busca las cadenas que **terminen** por ‘per’
- ‘%per%’: busca todas las cadenas que **contengan** ‘per’ en cualquier punto.

Para recuperar los datos insertados en una base de datos podemos asignar los resultados de una consulta a una variable:


```
$resultado=mysqli_query($dbc, $query);
```

Para acceder a múltiples filas de información recuperada, deberemos ejecutar la variable \$resultado a través de un bucle:

```
while($fila=mysqli_fetch_array($resultado){
    //Hacer algo con $fila
}
```

Con cada iteración del bucle, se devuelve la siguiente fila de información de la consulta (referenciada por \$resultado), en una matriz llamada \$fila. Este proceso continua hasta que no se encuentran más filas.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ver mi blog</title>
  </head>
  <body>
    <?php
      // Este script recupera las entradas de blog de una base de datos
      //Conectar y seleccionar la base de datos
```

	RAMA:	Informática	CICLO:	DAM			
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial				CURSO:	1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:		
	UNIDAD COMPETENCIA		PHP. Conexión con bases de datos				

```

$dbc = mysqli_connect('localhost', 'root');
mysqli_select_db($dbc,'myblog');
//Definir la consulta
$query = 'SELECT * FROM entries ORDER BY date_entered DESC';
if ($r = mysqli_query($dbc, $query)) { //Ejecutamos la consulta
    //Recuperar y mostrar cada registro
    while ($fila = mysqli_fetch_array($r)) {
        print "<p><h3>{$fila['title']}</h3> {$fila['entry']} <br/>
            <a href = \"editar_entrada.php?id={$fila['entry_id']}\">Editar </a> &nbsp;
            <a href=\"borrar_entrada.php?id={$fila['entry_id']}\">Borrar </a>
        </p> \n";
    }
} else { //No se ha ejecutado la consulta
    print "<p style='color:red;'>No ha sido posible recuperar la entrada porque: <br/>' .
        mysqli_error($dbc) . '</p><p>El query que se estaba ejecutando era' . $query . '</p>';
}
mysqli_close($dbc);
?>
</body>
</html>

```

2.7. Eliminar datos de una base de datos

Para eliminar datos de una base de datos utilizaremos la consulta DELETE . La sintaxis para una consulta de eliminación es:

```
DELETE FROM nombre_tabla WHERE columna=VALOR
```

No hace falta la cláusula WHERE, pero si es omitida eliminarán todos los registros de la tabla, una vez eliminado el registro, no hay forma de recuperarlo (a menos que haya una copia de seguridad de la base de datos).

Como medida de seguridad se quiere eliminar solamente un registro de la tabla, se puede agregar la cláusula LIMIT a la consulta:

```
DELETE FROM nombre_tabla WHERE columna=VALOR LIMIT 1
```

Como ejemplo vamos a ver un script borrar_entrada.php vinculado a la página RecuperarEntrada.php. Esta página recibe el ID del registro de la base de datos en la URL, para cada entrada confirma que el usuario desea eliminarla.

```

<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Borrar una entrada del blog</title>
  </head>
  <body>
    <?php
      //Este script elimina una entrada del blog
      //Conectar y seleccionar la base de datos


```

RAMA:	Informática	CICLO:	DAM
MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial		
CURSO:	1	PROTOCOLO:	Apuntes clases
AVAL:	2	DATA:	
UNIDAD	COMPETENCIA	PHP. Conexión con bases de datos	

```

$dbc = mysqli_connect('localhost', 'root');
mysqli_select_db($dbc,'myblog');
if (isset($_GET['id']) && is_numeric($_GET['id'])) {
    //Definimos la consulta
    $query = "SELECT title, entry FROM entries WHERE entry_id={$_GET['id']}";
    if ($r = mysqli_query($dbc, $query)) { //Ejecutamos la consulta
        $fila = mysqli_fetch_array($r); //Recuperamos la informacion
        //Creamos el formulario
        print '<form action="borrar_entrada.php" method="post">
            <p>¿Seguro que quieres borrar esta entrada?</p>
            <p><h3>' . $fila['title'] . '</h3>' . $fila['entry'] . '<br/>
            <input type = "hidden" name = "id" value = "' . $_GET['id'] . '" />
            <input type = "submit" name = "submit" value = "Borrar entrada"/>
            </form>';
    } else { //No se ha ejecutado la consulta
        print '<p style="color:red;">No ha sido posible recuperar la entrada porque: <br/>' .
            mysqli_error($dbc) . '</p><p>El query que se estaba ejecutando era' . $query . '</p>';
    }
} elseif (isset($_POST['id']) && is_numeric($_POST['id'])) {
    //Definir la consulta
    $query = "DELETE FROM entries WHERE entry_id={$_POST['id']} LIMIT 1";
    $r = mysqli_query($dbc,$query); //Ejecutamos la consulta
    //Resultado
    if (mysqli_affected_rows($dbc) == 1) {
        print '<p>La entrada del blog ha sido eliminada</p>';
    } else { //No se ha ejecutado la consulta
        print '<p style="color:red;">No ha sido posible borrar la entrada porque: <br/>' .
            mysqli_error($dbc) . '</p><p>El query que se estaba ejecutando era' . $query . '</p>';
    }
}
else {
    print '<p style="color:red;">Error de acceso a la página </p>';
}
mysqli_close($dbc);
?>
</body>
</html>

```

	RAMA:	Informática	CICLO:	DAM			
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial				CURSO:	1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:		
	UNIDAD COMPETENCIA		PHP. Conexión con bases de datos				

2.8. Actualizar datos en una base de datos

Para actualizar datos utilizaremos la cláusula UPDATE. Su sintaxis es:

```
UPDATE nombre_tabla SET columna1_nombre=valor, columna2_nombre=valor WHERE alguna_columna=valor
```


Si los valores son cadenas, deberán ir entre comillas simples

```
UPDATE usuarios SET nombre='Carmen' edad=18 WHERE user_id=142
```

Al igual que en la sentencia DELETE, utilizamos la cláusula WHERE para limitar las filas que se verán afectadas, si no se hace se actualizarán todos los registros de la base de datos.

Como ejemplo vamos a escribir una página que permita editar las entradas del blog.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Modificar entrada</title>
  </head>
  <body>
    <?php
      //Este script permite modificar la entrada del blog
      //Conectar y seleccionar la base de datos
      $dbc = mysqli_connect('localhost', 'root');
      mysqli_select_db($dbc, 'myblog');
      if (isset($_GET['id']) && is_numeric($_GET['id'])) {
        //Definimos la consulta
        $query = "SELECT title, entry FROM entries WHERE entry_id={$_GET['id']}";
        if ($r = mysqli_query($dbc, $query)) { //Ejecutamos la consulta
          $fila = mysqli_fetch_array($r); //Recuperamos la información
          //Creamos el formulario
          print<form action="editar_entrada.php" method="post">
            <p>Título de la entrada: <input type="text" name="title" size="40" value=""
htmlentities($fila['title']) . "'/></p>
            <p>Texto: <textarea name="entry" cols="40" rows="5">' . htmlentities($fila['entry'])
'</textarea></p>
            <input type="hidden" name="id" value="" . $_GET['id'] . "'/>
            <input type="submit" name="submit" value="Actualizar Entrada"/>
          </form>;
        } else { //No se puede recuperar la informacion
          print '<p style="color:red;">No ha sido posible recuperar la entrada porque: <br/>' .
            mysqli_error($dbc) . '</p><p>El query que se estaba ejecutando era' . $query . '</p>';
        }
      } elseif (isset($_POST['id']) && is_numeric($_POST['id'])) {
        $problema = FALSE;
        if(!empty($_POST['title'])&& !empty($_POST['entry'])) {
          $titulo = mysqli_real_escape_string($dbc, trim($_POST['title']));
          $entrada = mysqli_real_escape_string($dbc, trim($_POST['entry']));
        }
        else {
          print '<p style="color:red;">Por favor introduzca un título y una entrada <br/>';
          $problema = TRUE;
```

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Lenguajes de marcas y sistemas de gestión empresarial				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	
	UNIDAD	COMPETENCIA	PHP. Conexión con bases de datos			

```

}
if (!$problema) {
    //Definimos y ejecutamos la consulta
    $query = "UPDATE entries SET title='$titulo', entry='$entrada' WHERE entry_id={$_POST['id']}";
    $r = mysqli_query($dbc,$query);
    //Mostramos el resultado
    if (mysqli_affected_rows($dbc) == 1) {
        print '<p>La entrada del blog ha sido actualizada</p>';
    } else {
        '<p style="color:red;">No ha sido posible actualizar la entrada porque: <br/>' .
            mysqli_error($dbc) . '</p><p>El query que se estaba ejecutando era' . $query . '</p>';
    }
} else { //no se establece el ID
    print '<p style="color:red;">Error de acceso a la página </p>';
}
}
mysqli_close($dbc);
?>
</body>
</html>

```