Teoría: Acceder con PHP a las bases de datos

1 - Objetivos

Objetivos

- **Saber conectarse** al servidor que contiene el gestor de bases de datos MySQL.
- **Aprender a crear una base de datos, a conectarse a ella y a borrarla** con sus tablas e información, sabiendo utilizar, además, las funciones que controlan los **posibles errores**.
- **Aprender a consultar una base de datos** utilizando *queries* en las que intervengan las principales funciones de PHP para acceder a bases de datos de tipo MySQL y tratar su información.
- **Saber insertar**, **actualizar** y **eliminar** registros de una tabla, haciendo uso, además, de formularios y de celdas de tablas para mostrar su información, preguntarla o seleccionar campos.
- **Aprender a instalar el controlador MyODBC** para poder conectar bases de datos de tipo MS Access con las de tipo MySQL, así como saber **importar o exportar** tablas entre ambos gestores de bases de datos.
- **Saber mantener una base de datos** realizando a través de funciones de usuario las principales operaciones de este tipo: altas, bajas, consultas, búsquedas, actualizaciones, etcétera.

2 - Funciones de PHP para acceder a las bases de datos

Conexión con una base de datos

Ya hemos comentado en las Unidades anteriores que PHP permite acceder y tratar la información de las principales bases de datos que hay en el mercado actualmente: Oracle, MySQL, PostgreSQL, SysBase, Informix, etcétera. En la Unidad anterior también hemos comentado que puede hacerlo tanto a través de los programas nativos de la propia base de datos (API) como mediante una conexión genérica de tipo OBDC.

En esta Unidad vamos a exponer y explicar, precisamente, cómo accede PHP a MySQL y las

funciones que permiten realizar operaciones con las bases de datos, sus tablas y los datos que contengan: crearlas, introducir información, actualizar ésta, eliminarla, elaborar informes a partir de la misma, etcétera. Así pues, en esta Unidad se estudiarán las principales funciones de PHP relacionadas con el servidor de datos MySQL.

En el capítulo **LXIV. Funciones MySQL** del Manual de PHP se listan las principales funciones de acceso al servidor de bases de datos MySQL y se permite acceder a la explicación de cada una mediante los enlaces correspondientes. Haremos, pues, uso de las APIs originales de esta base de datos.

En esta misma Unidad, más adelante, estableceremos también una conexión OBDC que nos permitirá acceder a las bases de datos de tipo MS Access.

Si los alumnos y alumnas del curso han comprendido y asimilado bien los conceptos fundamentales sobre las bases de datos relacionales, explicados en la Unidad anterior, y aprenden a manejar en esta Unidad las bases de datos de tipo MySQL, estarán preparados sin duda para poder aplicar las funciones correspondientes de otros tipos de bases de datos, que también incorpora el lenguaje PHP.

En esta Unidad vamos a seguir un proceso similar al de la Unidad anterior, llevando a cabo más o menos las mismas operaciones con las bases de datos, pero en este caso usando las funciones de PHP.

Establecer una conexión con el servidor de bases de datos MySQL

Para realizar una conexión con una base de datos creada con MySQL hay que utilizar la función **`mysql_connect()`**.

La función **`mysql_connect()`** establece una conexión con el servidor de bases de datos MySQL. Tiene la siguiente sintaxis:

```
mysql_connect ([nombre del servidor [:puerto]
               [:/camino/al/socket]
               [, string usuario
               [, string password]]]
                   )
```

Esta función devuelve un identificador de enlace (número entero positivo) si se establece la conexión o *False* si no se consigue conectar.

Todos los argumentos son opcionales. Si no los hay, se asumen los valores por defecto, que son 'localhost' como nombre del servidor, usuario propietario del proceso del servidor, en nuestro caso 'root', y *password* vacía.

Como hemos visto, para establecer una conexión con el servidor MySQL, pueden utilizarse hasta cinco datos:

1. **Servidor**: nombre del equipo o dirección IP donde está la base de datos. Por defecto, si no se pone nada, se asume el servidor "localhost".

2. **Puerto**: vía por donde se accede a la base de datos. Si no se pone, por defecto se asume el valor por defecto que es el "3306".

3. **Camino al socket** (**path_to_socket**): nombre del subdirectorio donde está el programa que está usando el servidor para escuchar las peticiones de los clientes. Sólo tiene efecto en el entorno Unix. Si no se pone, por defecto se asume "/tmp/mysql.sock".

4. **Usuario**: nombre del usuario que accede a la base de datos. Por defecto es el que tenga el propietario del servidor, que suele ser "root".

5. **Clave**: es la password de acceso que tiene el usuario. Si no se pone, se asume una cadena vacía.

Si ya hemos establecido una conexión y hacemos una nueva llamada a la función **`mysql_connect()`** con los mismos argumentos, no se establece un nuevo identificador de enlace, sino que se devuelve el mismo de la conexión ya abierta.

La conexión con el servidor se cierra cuando acaba de interpretarse el *script* PHP. Puede cerrarse antes indicándolo explícitamente con la función **`mysql_close()`**. Esta función tiene la sintaxis siguiente:

```
mysql_close ([identificador de conexión])
```

Devuelve *True* si se cierra la conexión y *False* si no se hace.

Así pues, **`mysql_close()`** cierra la conexión con la base MySQL que esté asociada con el identificador de enlace especificado. Si no se especifica el identificador de enlace, se asume por defecto el último enlace abierto.

Puede ocurrir que necesitemos mantener una conexión persistente, es decir, que no se cierre cuando acabe de interpretarse el *script* PHP, de forma que siga establecida cuando se ejecuten otros *scripts*. En este caso, debemos establecer una conexión persistente, que se crea con la función **`mysql_pconnect()`**. Esta función tiene la misma sintaxis que la función **`mysql_connect()`**, con la sola diferencia de que establece una conexión persistente, que no se acaba con la interpretación del fichero *script* donde se ha establecido, sino que se mantiene al interpretarse otros. Una conexión persistente no se cierra con la función **`mysql_close()`**.

Por ejemplo, si queremos establecer una conexión con el servidor "mihost" utilizando el nombre de usuario "joseja" y la clave "garcia", debemos escribir

```
mysql_connect("mihost", "josefa", "garcia");
```

Hay que tener en cuenta que, para que estos datos sean aceptados como válidos, es necesario que el servidor se denomine así. Además, el nombre del usuario y de la clave deben haber sido dados de alta en el servidor MySQL antes de pretender usarlos, ya que el servidor procede a autentificar estos datos antes de permitir la conexión.

Ahora bien, como esta instrucción, si se consigue, devuelve un identificador de conexión que después hemos de utilizar en otras funciones, lo mejor es asignar a una variable el nombre de este identificador, de esta forma:

```
$id_conexion = mysql_connect("mihost2", "josefa", "garcia");
```

Por otra parte, puede ser que no se logre conectar, en cuyo caso se produciría un error con un mensaje en inglés. Para optimizar un poco la salida en este caso, lo mejor es utilizar el

operador `@`, para prescindir del mensaje del sistema, y la función **`die()`**, para mostrar nuestro mensaje, como hemos hecho en las Unidades anteriores, de esta forma:

```
$id_conexion = @mysql_connect("mihost","josefa","garcia")
            or die("No se pudo establecer la conexión");
```

Además, como lo más probable es que necesitemos usar al menos estos tres datos en sucesivas conexiones, lo mejor es asignar sus valores a tres variables, guardarlas en un fichero de texto y, luego, con la sentencia `include` o `require` traerlas al *script* correspondiente cuando se necesiten para establecer una conexión.

Por ejemplo, creamos el fichero `datos_conexion.php` y guardamos en el mismo este contenido:

```
<?
     $servidor="mihost";
     $usuario="josefa";
     $clave="garcia";
 ?>
```

Después, ya podemos establecer la conexión en el *script* correspondiente, de esta forma:

```
<?
   include("datos_conexion.php");
   $id_conexion=@mysql_connect($servidor,$usuario,$clave)
                  or die("No se pudo establecer la conexión");
?>
```

En el **Ejemplo 1** de esta Unidad puede verse cómo hemos usado la función **`mysql_connect()`** para establecer una conexión con el servidor MySQL instalado para la realización del curso. Para poder ejecutarlo, es preciso arrancar el servidor MySQL antes de pretender establecer una conexión. Ahora ya no es preciso arrancar la utilidad MySQL-Front ni conectarse mediante la misma.

Además, en los *scripts* del modelo de **Proyecto final** del curso (carpeta `c:\cursoPHP5\curso\curso_ini\capitulos\11proyecto_final\proyecto_final`) se usan también todas las funciones que se van a explicar en esta Unidad. Conviene que, ya desde este momento, el alumno o alumna vaya fijándose en esta aplicación que pretendemos le sirva como modelo para realizar el propio proyecto final que ha de enviar al tutor o tutora antes de hacer la prueba final presencial del curso.

Conexión a servidores que contengan otros tipos de bases de datos

Aunque en este curso sólo se usa el gestor de base de datos MySQL, ponemos a continuación las funciones con la que se conectaría a otros tipos de bases de datos de las que pudiera disponer particularmente el alumno. En todo caso, consultando el Manual de PHP puede obtenerse información detallada sobre estas funciones y conocer su sintaxis.

Base de datos	Función para conectarse
PostgreSQL	`pg_connect("host=$servidor,username=$usuario, password=$clave,port=5432,dbname=$base");`

Informix `ifx_connect($base,$usuario,$clave);`

En estos dos tipos de bases de datos, se realizan a la vez dos operaciones: conectarse al servidor y a la base de datos.

ODBC `odbc_connect(%dsn,$usuario,$clave);`

En el último apartado de esta Unidad aprenderemos a establecer conexiones mediante el estándar ODBC y a acceder a los datos de bases de datos de tipo Access.

SQL Server `mssql_connect($servidor,$usuario,$clave);`

Oracle `ocilogon($usuario,$clave[,$basedatos]);`

Para establecer una conexión persistente, hay que poner `pconnet` en lugar de `connect`. En Oracle se pone `ociplogon`.

3 - Crear, seleccionar y destruir una base de datos

PHP permite también crear, seleccionar y borrar bases de datos asociadas a un identificador de conexión dentro del servidor al que se ha conectado.

Queremos hacer aquí una advertencia importante. Las operaciones de crear y eliminar bases de datos y tablas dentro de éstas son más bien propias del administrador del servidor y, por tanto, no es habitual que un usuario o cliente pueda crearlas y destruirlas. En cambio, son operaciones más propias de un cliente hacer solicitudes que permitan consultar y, como máximo, ampliar, modificar o eliminar el contenido de una base de datos desde su navegador.

Es importante saber que si se borra una base de datos, ésta no sólo pierde el contenido, sino que, después, no se pueden recuperar sus datos.

En este apartado vamos a estudiar las maneras de crear bases de datos y destruirlas. Asimismo, veremos cómo podemos seleccionar una entre las diferentes base de datos que haya en el servidor.

Crear una base de datos

En versiones anteriores a PHP 5 estaba disponible la función `mysql_create_db()` que permitía crear una base de datos de tipo MySQL asociada a un identificador de conexión. No obstante, en las últimas versiones de PHP se ha omitido esta función para simplificar el código y hay que crear las bases de datos directamente usando una **query**, es decir, ejecutando una consulta.

Por ejemplo, si queremos crear la base de datos “pruebas” asociada a la conexión del apartado anterior, podemos escribir

```
mysql_query('CREATE DATABASE pruebas', $id_conexion);
```

Para asimilar bien la función `mysql_query()`, hay que estudiarla detenidamente en el apartado siguiente, donde se aborda.

Como puede producirse algún error, es conveniente utilizar una estructura condicional de la forma siguiente:

```
if (mysql_query('CREATE DATABASE pruebas', $id_conexion))
     print("La base de datos \"pruebas\" se ha creado correctamente <P>");
else printf("No se ha podido crear la base de datos:%s <P>",'Error nº
            '.mysql_errno().'.-'.mysql_error());
```

Hemos aprovechado la estructura condicional anterior para incluir dos nuevas funciones que pueden servirnos de control de errores cuando se produzcan éstos en cualquier tipo de operación con una base de datos. Son las siguientes:

La función `mysql_errno()` devuelve el número de error asignado por MySQL y `mysql_error()` devuelve el mensaje en inglés correspondiente al error que se ha producido. Por ejemplo, si ya existe la base de datos "pruebas" por haber sido creada antes, se indicará lo siguiente al ejecutarse la cláusula `else` de la estructura anterior:

Error nº 1007.-Can't create database 'pruebas'. Database exists

Si empleamos acertadamente estas dos funciones para manejar y controlar los errores, se pueden conocer y detectar, antes de solucionarlos, posibles problemas al realizar diferentes operaciones con las bases de datos de tipo MyQSL.

Eliminar una base de datos

En versiones anteriores a PHP 5 estaba disponible la función `mysql_drop_db()` que permitía eliminar una base de datos de tipo MySQL asociada a un identificador de conexión. No obstante, en las últimas versiones de PHP se ha omitido esta función para simplificar el código y hay que borrar las bases de datos directamente usando una **query**, es decir, ejecutando una consulta.

Por ejemplo, si queremos borrar la base de datos "pruebas" asociada al identificador de conexión actual, podemos escribir

```
mysql_query('DROP DATABASE pruebas', $id_conexion);
```

Recordamos que para asimilar bien la función `mysql_query()`, hay que estudiarla detenidamente en el apartado siguiente, donde se aborda.

Es importante tener mucho cuidado al utilizar esta función, ya que, si se ejecuta, se perderá la base de datos con sus tablas y toda la información que contengan, sin que sea posible recuperarlas posteriormente.

También en este caso conviene utilizar una estructura de control condicional y las funciones de control de errores, como hemos hecho y explicado al crear esta base de datos.

Seleccionar una base de datos

La función `mysql_select_db()` permite elegir la base de datos de tipo MySQL con la que se va a trabajar entre las diferentes bases que haya en el servidor. Su sintaxis es ésta:

```
mysql_select_db(base de datos[, identificador de conexión])
```

Por ejemplo, si queremos trabajar con la base de datos "pruebas" asociada al identificador de conexión actual, podemos seleccionarla escribiendo:

```
mysql_select_db("pruebas",$id_conexion);
```

Si no se especifica el identificador de conexión, que es opcional, se asume como tal el último creado.

Al igual que en las dos operaciones anteriores, conviene utilizar una estructura condicional para comprobar si la operación se ha llevado a cabo correctamente, conocer los posibles problemas y poder arreglarlos.

En el **Ejemplo 1** de esta Unidad puede verse cómo hemos usado las funciones de creación y de borrado de base de datos y `mysql_select_db()`. Recordamos que, para poder ejecutarlo, es preciso arrancar el servidor MySQL antes de crear una base de datos, destruirla o seleccionarla, y que no es preciso utilizar la utilidad Cliente MySQL.

Creación de tablas

Como hemos dicho, PHP no dispone de funciones para crear tablas dentro de las bases de datos. Ésta es más bien una operación del administrador del servidor que contiene la información. Por ello, quien necesite crear tablas dentro de una base de datos ha de recurrir a *scripts* de tipo `sql` que puede ejecutar desde una ventana del DOS con el programa gestor de datos **mysql.exe** o bien mediante la utilidad Cliente MySQL. De todo ello hemos hablado y realizado suficientes prácticas en la Unidad 6.

No obstante, en el apartado siguiente veremos que se pueden crear tablas mediante una **query**, es decir, ejecutando una consulta. Por ejemplo, si dentro de la base de datos "pruebas" queremos incluir la tabla "agenda", podemos hacerlo así:

```
$consulta = "create table agenda (registro INT NOT NULL
             AUTO_INCREMENT, nombre CHAR(50),direccion CHAR(100),
             telefono CHAR(15), email CHAR(50), KEY (registro) )";

$datos= @mysql_query($consulta,$id_conexion) or
       die("No se ha podido ejecutar la consulta. Compruebe si la
             sintaxis de la misma es correcta.");
```

En la primera de las dos instrucciones anteriores asignamos a la variable `$consulta` la sentencia `sql` de creación de la tabla "agenda" indicando el nombre de ésta, así como el nombre de los campos, su tipo y longitud. Además, fijamos como clave de índice el campo "registro".

En la segunda instrucción ejecutamos la **query**, gracias a lo cual dentro de la base de datos "pruebas" se crea la tabla "agenda" con la estructura especificada. En el apartado siguiente encontrarás toda la información necesaria sobre la función `mysql_query()`.

También en el **Ejemplo 1** de esta Unidad aparece el código donde se crea la tabla indicada. Las nuevas funciones utilizadas se explican en el apartado siguiente de esta misma Unidad.

4 - Realizar consultas en una base de datos

PHP permite también consultar la información que haya en las tablas de una base de datos emitiendo con ella diferentes tipos de informes.

En este apartado vamos estudiar las funciones que permiten realizar consultas y elaborar informes a partir de la información contenida en las tablas de una base de datos. Concretamente, nos vamos a conectar a la base de datos "biblioteca", ya creada en el servidor MySQL al instalarse el curso, y a servirnos de su información para elaborar los informes y realizar las *queries* (consultas) de este apartado.

Ahora necesitamos conocer y recordar las sentencias SQL que vamos a poner como argumentos de las funciones de PHP. En la Unidad 6 se explicaron las más importantes. Si el alumno tiene alguna duda sobre la interpretación de las mismas, puede recurrir a los contenidos de la Unidad anterior o bien consultar algún manual propio sobre SQL o buscar información en las direcciones de Internet, indicadas en la Presentación del curso, que hacen referencia a este lenguaje.

Consultar una base de datos

La función **`mysql_query()`** envía una sentencia SQL al servidor para que éste la interprete. Su sintaxis es ésta:

```
mysql_query(consulta[,identificador de conexión])
```

Lo más recomendable es crear una variable y asignarle el texto de la consulta. Después, pasamos esta variable como primer argumento de la función `mysql_query()`.

Por ejemplo, si queremos enviar al servidor una consulta de la base de datos "biblioteca" en la que se muestren el título y el autor de los libros de la tabla "libros", debemos escribir

```
$consulta="select titulo,autor from libros";
mysql_query($consulta,$id_conexion);
```

Si no se especifica el identificador de conexión, que es opcional, se asume como tal el último identificador creado. Si no hay ninguno, de forma automática la función intenta establecer un enlace como si se utilizara la función `mysql_connect()` sin argumentos, y lo utiliza.

Hay que advertir que la sentencia SQL, asignada como texto de la variable `$consulta`, no puede terminar en punto y coma.

La función `mysql_query()` devuelve *True* para indicar que la sentencia es correcta y se ha ejecutado sin problemas y *False* en caso contrario. Además, no indica el número de filas devueltas (en el ejemplo anterior, registros de la tabla), aunque la sentencia SQL se ejecute correctamente.

Como puede producirse algún error, es conveniente utilizar alguno de los procedimientos ya conocidos: una estructura condicional; la combinación del operador `@` con la función `die()`, como ya hemos explicado anteriormente; o las funciones `mysql_errno()` y `mysql_error()`, que comentaremos después.

Como puede verse en el **Ejemplo 2**, hemos preferido la siguiente sintaxis:

```
$consulta="select titulo,autor from libros";
$datos= @mysql_query($consulta,$id_conexion) or
       die("No se ha podido ejecutar la consulta.
             Compruebe si la sintaxis de la misma
             es correcta.");
```

Conviene observar que la función `mysql_query()` ejecuta la consulta si es correcta, pero no devuelve por sí sola ninguna información al navegador del cliente. Por eso, hemos asignado su resultado a la variable `$datos`, que vamos a usar como identificador de la consulta dentro de la función `mysql_result()`.

Así pues, debemos combinar la primera función (`mysql_query()`) con la segunda (`mysql_result()`) para poder asignar la información de la consulta a diferentes variables y, luego, poder mostrar éstas en la pantalla del cliente.

La función **`mysql_db_query()`** envía una sentencia SQL al servidor especificando el nombre de la base de datos sobre la que se ejecuta la sentencia. Tiene, pues, el mismo comportamiento que la función **`mysql_query()`**, si bien su sintaxis es más completa, como puede verse:

```
mysql_db_query(nombre de la base de datos, consulta
               [,identificador de conexión])
```

En este caso no es necesario, en consecuencia, seleccionar previamente la base de datos con la función `mysql_select_db()`.

Por ejemplo, si queremos usar esta función para enviar al servidor una consulta de la base de datos "biblioteca" en la que se muestren el título y el autor de los libros de la tabla "libros", sin necesidad de seleccionar previamente la base de datos, debemos escribir

```
$consulta="select titulo,autor from libros";
mysql_db_query("biblioteca",$consulta,$id_conexion);
```

La función **`mysql_result()`** devuelve los datos solicitados (registros de datos) de un identificador generado por la función `mysql_query()`. Su sintaxis es ésta:

```
mysql_result(identificador de la consulta, número de
             fila, número de columna o nombre de campo)
```

En la Unidad anterior ya explicamos que una tabla está estructurada en filas (registros) y columnas (campos). Pues bien, al producirse la consulta, el identificador devuelto por la misma es una especie de matriz bidimensional que tiene en la primera dimensión tantas filas como registros se hayan visto afectados y en la segunda, tantas columnas como se haya indicado. Por ejemplo, en nuestra consulta anterior, que ejecuta la sentencia SQL `select titulo,autor from libros`, sabemos que hay 28 filas (todos los registros de la tabla "libros") y 2 columnas (los campos indicados).

Podemos acceder a cualquiera de esos 56 datos indicando simplemente el número de fila y el de columna, contando siempre desde 0 en adelante. En el **Ejemplo 2** de esta Unidad puede verse que mostramos los dos campos (título y autor) de los dos primeros registros de la forma siguiente:

```
$resul_00=mysql_result($datos,0,0);
$resul_01=mysql_result($datos,0,1);
echo $resul_00.' de '.$resul_01;
$resul_10=mysql_result($datos,1,0);
$resul_11=mysql_result($datos,1,1);
echo $resul_10.' de '.$resul_11;
```

En lugar del número de columna, podemos indicar directamente el nombre del campo, como cadena, o bien el alias que le hayamos asignado. Por ejemplo, podemos escribir también así la primera instrucción de asignación:

```
$resul_00=mysql_result($datos,0,"autor");
```

o bien, si hubiéramos definido la consulta como `select titulo,autor as a from libros`, podíamos haber escrito

```
$resul_00=mysql_result($datos,0,"a");
```

En las explicaciones anteriores hemos utilizado el acceso individualizado a cada uno de los datos de la consulta por motivos didácticos, para que el alumno comprenda bien cómo se produce la consulta y el resultado que genera. Cabe destacar que no conocemos el número de registros de una base de datos MySQL hasta que hacemos una consulta sobre la misma.

A continuación, estudiaremos otras funciones que permiten un tratamiento más adecuado y una presentación más eficaz de la información resultante de una consulta. Son las siguientes:

La función **`mysql_fetch_row()`** devuelve el contenido de una sentencia SQL en un array identificando cada elemento con un campo de la tabla y mueve el puntero al registro siguiente, devolviendo una matriz de índices numéricos con los datos contenidos en los distintos campos:

```
while($fila = mysql_fetch_row($datos))
{
   echo $fila[0]." de ".$fila[1];
}
```

Si la función `mysql_fetch_row()` llega al final de los registros resultantes de la consulta SQL entonces devuelve `FALSE` en lugar de la matriz con los datos.

La función **`mysql_fetch_array()`** devuelve el contenido de una sentencia SQL en un array identificando cada elemento con un campo de la tabla. Extrae sólo una fila del resultado como una matriz asociativa y mueve el puntero al registro siguiente. Su sintaxis es ésta:

```
mysql_fetch_array()(identificador de la consulta, [,tipo de resultado])
```

La función `mysql_fetch_array()` es una versión extendida de la función `mysql_fetch_row()`, estudiada anteriormente, pues no sólo guarda los datos en el índice numérico de la matriz, sino que también guarda los datos en los índices asociativos, usando el nombre del campo como clave (distinguiendo entre minúsculas y mayúsculas).

Si la función `mysql_fetch_array()` llega al final de los registros resultantes de la consulta SQL entonces devuelve `FALSE` en lugar de la matriz con los datos.

Si dos o más campos del resultado tienen el mismo nombre, el último tiene prioridad. Para acceder a los otros campos con el mismo nombre, hay que especificar el índice numérico o

definir un alias para esos campos. Por ejemplo, en nuestra base de datos "biblioteca" las tres tablas tiene el campo homónimo "registro". Para que en la matriz resultante de la consulta se distingan los tres, deberíamos escribir así la sentencia SQL:

```
select l.registro as lr, u.registro as ur, p.registro as pr
from libros l,usuarios u, prestamos p;
```

De esta forma, en la matriz resultante tendríamos una columna para lr (registro de libros), otra para ur (registro de usuarios) y una tercera para pr (registro de préstamos).

El segundo argumento, que es opcional, es una constante y puede tener los valores siguientes: MYSQL_ASSOC, MYSQL_NUM, y MYSQL_BOTH (valor por defecto) que indican los tipos de índices que se deben guardar en la matriz: asociativos, numéricos o ambos.

Por razones evidentes preferimos utilizar la función `mysql_fetch_array()`.

En el **Ejemplo 2** de esta Unidad puede verse cómo aplicamos la función explicada incluyéndola dentro de una bucle para recorrer todos los registros de la tabla. Éste es el código que hemos empleado:

```
while($fila = mysql_fetch_array($datos))
{
   echo $fila["titulo"]." de ".$fila["autor"];
}
mysql_free_result($datos);
```

La función **`mysql_free_result()`** libera de la memoria el resultado de la consulta. Conviene utilizarla siempre que tengamos dudas sobre la capacidad de la memoria del ordenador en el que ejecutamos la consulta.

La función **`mysql_fetch_objetc()`** devuelve el contenido de una sentencia SQL como un objeto. Extrae sólo una fila del resultado como un objeto con sus propiedades. Tiene la misma sintaxis que la función `mysql_fetch_array()`, pero se diferencia de ella en la forma en que debemos referirnos a cada dato. Por ejemplo, si usamos esta función en el caso del **Ejemplo 2**, debemos escribir:

```
while($fila = mysql_fetch_object($datos))
{
   echo $fila->titulo." de ".$fila->autor;
}
mysql_free_result($datos);
```

La función **`mysql_num_rows()`** devuelve el número de filas que se han vista afectadas al ejecutarse una sentencia SQL. Una vez que conocemos este valor, podemos utilizarlo, por ejemplo, para hacer un bucle que recorra todos los registros resultantes de la consulta y mostrar sus datos. Por ejemplo, podemos mostrar el título, la editorial, el año de publicación, el número de páginas y el precio de todos los libros que sean de la editorial "Alfaguara", ordenados por título, con el bucle siguiente:

```
$consulta="select titulo,editorial,anno_publica,paginas,precio_euros
           from libros
           where editorial='Alfaguara'
           order by titulo";
$datos= @mysql_query($consulta,$id_conexion)
        or die("No se ha podido ejecutar la consulta.Compruebe
```

```
                si la sintaxis de la misma es correcta.");

$filas=mysql_num_rows($datos);
$campos=mysql_num_fields($datos);
for ($i=0;$i<$filas;$i++)
{
          $datos_fila=mysql_fetch_array($datos);
          print "Título: ".$datos_fila["titulo"].".".
                " Editorial: ".$datos_fila["editorial"].".".
                " Año de publicación:".$datos_fila["anno_publica"].".".
                " Número de páginas:".$datos_fila["paginas"].".".
                " Precio en euros".$datos_fila["precio_euros"].".<P>";
}
echo "En la consulta anterior se ha visto afectados $campos campos.";
```

La función **`mysql_data_seek()`** mueve el puntero interno a la fila especificada del resultado. Tiene la siguiente sintaxis:

```
mysql_ data_seek(identificador de la consulta,número de fila)
```

Una vez leído un resultado con la función `mysql_fetch_array()`, podemos llevar el puntero a la última fila del resultado con la orden

```
mysql_data_seek($datos,mysql_num_rows($datos)-1);
```

Debemos tener en cuenta que en la matriz los elementos (filas del resultado) se cuentan desde 0 hasta `mysql_num_rows()-1`.

Después de ejecutarse la función `mysql_data_seek()`, la próxima llamada a la función `mysql_fetch_rows()` devuelve los datos de la fila a la que se ha llevado el puntero interno.

De forma similar a `mysql_num_rows()`, la función **`mysql_num_fields()`** devuelve el número de campos del resultado de ejecutar una sentencia SQL. Puede observarse cómo hemos usado esta función en el código anterior asignando a la variable `$campos` el número de campos afectados por esta consulta.

En el **Ejemplo 2** puede verse el código completo de la consulta anterior.

Con la función **`mysql_fetch_field()`** podemos conocer las características siguientes de un campo (columna de la consulta), que se devuelven como propiedades de un objeto:

- **Name:** Nombre de la columna.
- **Table:** Nombre de la tabla a la que pertenece la columna.
- **Max_length:** Longitud máxima de la columna.
- **Not_null:** Devuelve 1 si la columna no puede contener un valor nulo.
- **Primary_key:** Devuelve 1 si la columna es la clave primaria.
- **Unique_key:** Devuelve 1 si la columna es una clave única.
- **Multiple_key:** Devuelve 1 si la columna es una clave no única.
- **Numeric:** - Devuelve 1 si la columna es numérica.
- **Blob:** Devuelve 1 si la columna es un BLOB.
- **Type:** Tipo de la columna.
- **Unsigned:** Devuelve 1 si la columna es *unsigned*.

- **Zerofill:** Devuelve 1 si la columna es *zero-filled*.

Su sintaxis es la siguiente:

```
mysql_fetch_field(identificador de la consulta,[,salto de campo])
```

El segundo argumento, que es opcional, indica el número de campo, contando desde 0, al que nos referimos. Si no se indica y es la primera vez que se ejecuta esta función, se devuelven las propiedades del campo 0 (en la tabla "libros" es "autor"). Si no se indica y se ejecuta la función sucesivas veces, cada vez se devuelven las propiedades del campo siguiente. Si se indica, se devuelven las del campo que se especifique.

La función anterior puede combinarse con la función `mysql_field_seek()`, que asigna el puntero del resultado al número de campo especificado. Por ejemplo, la instrucción

```
mysql_field_seek($datos,3);
```

lleva el puntero al campo 3, contando el primero como 0, de la tabla "libros", que es "anno_publica". Si ahora ejecutamos la función `mysql_fetch_field()` sin especificar el salto de campo, se devuelven las propiedades de este campo.

El conocimiento de estos datos puede ser necesario en ocasiones. En el **Ejemplo 2** (opción **Propiedades y flags de algunos campos**) puede verse cómo mostramos en la pantalla las propiedades del campo "sueldo_euros" de la tabla "usuarios".

Hay otras funciones más sencillas que permiten conocer sólo algunas características particulares de un campo. Son éstas:

La función **`mysql_field_flags()`** devuelve los *flags* asociados a un campo determinado de la tabla. Cada *flag* es devuelto como una palabra. Los *flags* están separados por un espacio. La función `explode()` permite dividir los nombres de los *flags* devueltos. La versión de MySQL usada en el curso soporta los siguientes flags: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

La función **`mysql_field_len()`** devuelve la longitud del campo especificado de la tabla.

La función **`mysql_field_name()`** devuelve el nombre del campo especificado de la tabla.

La función **`mysql_field_table()`** devuelve el nombre de la tabla donde aparece el campo especificado.

La función **`mysql_field_type()`** devuelve el tipo del campo especificado de la tabla.

La función **`mysql_fetch_lengths()`** devuelve en un array la longitud de todos los campos de la tabla sobre la que se ha ejecutado una sentencia SQL. Extrae el resultado de la última fila devuelta por `mysql_fetch_row()`, `mysql_fetch_array()` y `mysql_fetch_object()`.

Su sintaxis es ésta:

```
mysql_fetch_lengths(identificador de la consulta)
```

Con las explicaciones dadas creemos que los alumnos y alumnas pueden utilizar correcta y

eficazmente las funciones abordadas y servirse del Manual de PHP para obtener la información que precisen sobre las mismas.

Por otra parte, en el **Ejemplo 2** de esta Unidad puede estudiarse el código donde se aplican las principales funciones de este tipo y se realizan abundantes consultas SQL.

Otras funciones que muestran informaciones sobre las bases de datos

PHP incorpora en la API de MySQL algunas funciones más que permiten obtener información sobre las bases de datos y sus tablas. Son las siguientes:

La función `mysql_list_fields()` permite obtener los campos de una tabla de una base de datos. Tiene la siguiente sintaxis:

```
mysql_list_fields(base de datos, tabla [,identificador de conexión])
```

Por ejemplo, la instrucción

```
mysql_list_fields("biblioteca","usuarios",$id_conexion);
```

devuelve un puntero (número entero positivo) que puede ser usado después como argumento de las funciones `mysql_field_len()`, `mysql_field_name()` y `mysql_field_type()` para conocer la longitud, el nombre y el tipo, respectivamente, de los campos de la tabla "usuarios" de la base de datos "biblioteca". Si se produce un error, esta función devuelve el valor –1.

En el **Ejemplo 3** hemos utilizado así la función `mysql_list_fields()` combinándola con las funciones `mysql_num_fields()`, `mysql_field_len()`, `mysql_field_name()` y `mysql_field_type()`.

```
$campos_tabla=mysql_list_fields("biblioteca", "usuarios",$id_conexion);
$numero_campos=mysql_num_fields($campos_tabla);

for ($i=0;$i<$numero_campos;$i++)
{
          $nombre_campo=mysql_field_name($campos_tabla,$i);
          $tipo_campo=mysql_field_type($campos_tabla,$i);
          $longi_campo=mysql_field_len($campos_tabla,$i);
          print ("<B>Nombre:</B>$nombre_campo <B>Tipo:</B>
                 $tipo_campo <B>Longitud: </B>$longi_campo <P>");
}
```

La función `mysql_list_dbs()` permite obtener la lista de las bases de datos disponibles en el servidor MySQL. Tiene la siguiente sintaxis:

```
mysql_list_dbs([identificador de conexión])
```

Por ejemplo, la instrucción

```
mysql_list_dbs($id_conexion);
```

devuelve un puntero (número entero positivo) que puede ser usado después como argumento de la función `mysql_tablename()` para mostrar el resultado.

En el **Ejemplo 3** hemos utilizado así la función `mysql_list_dbs()` combinándola con la función `mysql_tablename()`.

```
$id_bases=mysql_list_dbs($id_conexion);
$no_bases=mysql_num_rows($id_bases);
for ($i=0;$i<$no_bases;$i++)
{
   $nombre_base=mysql_tablename($id_bases,$i);
   print ("<B>Nombre:</B>$nombre_base<P>");
}
```

Para mostrar las tablas que contiene una base de datos debemos escribir la **query** directamente una vez seleccionada la base de datos, podemos escribir

```
mysql_query('SHOW TABLES', $id_conexion);
```

Como hemos indicado, en el **Ejemplo 3** de esta Unidad aparece el código completo de este *script*, en el que se utilizan las funciones explicadas.

Cómo utilizar ventanas, botones y tablas

En los ejemplos anteriores hemos ofrecido la información de las bases de datos “a pelo”, es decir, mostrando sus datos tal cual. En este caso nuestra intención ha sido que los alumnos y alumnas asimilen bien las funciones explicadas y su sintaxis. Por ello, no nos hemos preocupado demasiado de la presentación en la pantalla de su información.

Pero, habitualmente, se utilizan etiquetas HTML para formatear la salida de la información, seleccionar algún dato o introducirlo desde la pantalla de nuestro navegador usando formularios. Esto se consigue rellenando etiquetas: ventanas desplegables (**Select**) y botones (**Input**) con los datos de uno o de varios campos de una tabla de datos. Además, la información puede hacerse aparecer dentro de las celdas de alguna tabla (**Table**), de forma que los registros y campos se incluyan en el interior de éstas.

Veamos algunos casos prácticos que hemos incluido en el **Ejemplo 4**. Este ejercicio es bastante complejo. Por ello, si encuentras dificultad en comprender el código de algunas opciones, abórdalo en su totalidad cuando acabes el estudio completo de esta Unidad.

```
<?
  require("../uni7_var.php");
   /* Recuperamos las variables globales de la conexión.*/

  /********** Funciones auxiliares de la página *******/
  function cabecera()
  {
     header("Cache-Control:no-cache");
     header("Pragmal: no-cache");
     echo "<HTML><HEAD><TITLE>Curso PHP 5 - Unidad 7 - Ejemplo
4</TITLE></HEAD><BODY>";
     $acciones=array("Nuevo","Buscar","Mostrar_todos");
     echo "<CENTER><H1>Usuarios</H1>";
     for ($i=0;$i<count($acciones);$i++)
         echo "<A
HREF=".$_SERVER["PHP_SELF"]."?op=$acciones[$i]>$acciones[$i]||</A>";
     echo "</CENTER><P>";
  }

  function pie()
```

```
  {
    echo "</BODY></HTML>";
  }

  function formulario_datos($registro, $dni,$nombre,$apellidos,
                        $domicilio,$localidad,$provincia,$telefono,$tipo
)
  {
     echo "
     <FORM ACTION=".$_SERVER["PHP_SELF"]."?op=$tipo METHOD=POST>
     <CENTER>
     <TABLE>
       <TR>
         <TD>DNI</TD>
            <TD><INPUT NAME=dni VALUE=\"$dni\" size=9></TD>
       </TR><TR>
            <TD>Nombre</TD>
            <TD><INPUT NAME=nombre VALUE=\"$nombre\" size=10></TD>
       </TR><TR>
            <TD>Apellidos</TD>
            <TD><INPUT NAME=apellidos VALUE=\"$apellidos\"size=25></TD>

       </TR><TR>
         <TD>Dirección</TD>
         <TD><INPUT NAME=domicilio VALUE=\"$domicilio\" size=35></TD>
       </TR><TR>
         <TD>Localidad</TD>
         <TD><INPUT NAME=localidad VALUE=\"$localidad\" size=20></TD>
       </TR><TR>
         <TD>Provincia</TD>
         <TD><INPUT NAME=provincia VALUE=\"$provincia\" size=20></TD>
       </TR><TR>
         <TD>Teléfono</TD>
         <TD><INPUT NAME=telefono VALUE=\"$telefono\" size=20></TD>
       </TR>
     </TABLE>
       <INPUT TYPE=HIDDEN NAME=registro VALUE=$registro>
       <INPUT TYPE=SUBMIT VALUE=Aceptar></CENTER>
     </FORM><P>";
  }

  function formulario_busqueda()
  {
     $campos=array( array("dni","DNI"),
                    array("nombre","Nombre"),
                    array("apellidos","Apellidos"),
                    array("domicilio","Dirección"),
                    array("localidad","Localidad"),
                    array("provincia","Provincia"),
                    array("telefono","Teléfono"));

     for ($i=0;$i<count($campos);$i++)
        echo "<FORM ACTION=".$_SERVER["PHP_SELF"]."?op=Buscar
METHOD=POST>
             <INPUT TYPE=hidden NAME='campo_busqueda'
VALUE=".$campos[$i][0].">
             <TABLE>
             <TR>
                     <TD width=90>".$campos[$i][1]."</TD>
                     <TD><INPUT NAME=buscar_txt></TD>
                     <TD><INPUT TYPE=SUBMIT NAME=boton VALUE=\"Buscar
por '".$campos[$i][0]."'\"></TD>
             </TR></TABLE></FORM>";
```

```
  }

  function listado($los_datos)
  {
     $filas=mysql_num_rows($los_datos);
     echo "<TABLE BORDER=1>";
     echo "<TR><TD>Registro</TD><TD>DNI</TD>
                 <TD>Nombre</TD><TD>Apellidos</TD>
                 <TD>Dirección</TD><TD>Localidad</TD><TD>Provincia</TD>
                 <TD>Teléfono</TD><TD colspan=2>Operación</TD>
               </TR>";
     for ($i=0;$i<$filas;$i++)
     {
         list($registro,$dni,$nombre,$apellidos,$domicilio,$localidad,
              $provincia,$telefono)=mysql_fetch_array($los_datos);
         echo "<TR>
                   <TD>$registro</TD><TD>$dni</TD>
                   <TD>$nombre</TD><TD>$apellidos</TD>
                   <TD>$domicilio</TD><TD>$localidad</TD>
                   <TD>$provincia</TD><TD>$telefono</TD>
                   <TD><A
HREF=".$_SERVER["PHP_SELF"]."?op=editar&registro=$registro>
                            Editar</A></TD>
                   <TD><A
HREF=".$_SERVER["PHP_SELF"]."?op=borrar&registro=$registro>
                            Borrar</A></TD>
                </TR>";
     }
     echo "</TABLE><P>";
   }


  /******* END Funciones auxiliares de la página *******/


//Aquí empieza la funcionalidad de la página
cabecera();

$id_conexion =@mysql_pconnect($DBHost, $DBUser, $DBPass) or
     die("<CENTER><H3>No se ha podido establecer la conexión.<P>
          Compruebe si está activado el servidor de bases de
          datos MySQL.</H3></CENTER>");
/* Intentamos establecer una conexión persistente con el servidor.*/

if (!mysql_select_db("biblioteca"))
    printf("<CENTER><H3>No se ha podido seleccionar la base de
            datos \"biblioteca\": <P>%s",'Error nº '.
            mysql_errno().'.-'.mysql_error());

/* Intentamos seleccionar la base de datos "biblioteca". Si no
   se consigue, se informa de ello y se indica cuál es el
   motivo del fallo con el número y el mensaje de error.*/

if (isset($_GET["op"])) {
  if ($_GET["op"]=="Nuevo")
  {
       formulario_datos("","","","","","","","","inserta");
  }
  else
  if ($_GET["op"]=="inserta")
  {  $bien = ( (!empty($_POST["dni"])) && (!empty($_POST["nombre"])) &&
               (!empty($_POST["apellidos"])) &&
               (!empty($_POST["domicilio"])) &&
```

```
                (!empty($_POST["localidad"])) &&
                (!empty($_POST["provincia"])) &&
                (!empty($_POST["telefono"])) );
     if ($bien)
     {
        $consulta="insert into usuarios values
                ('".$_POST["nombre"]."','".$_POST["apellidos"]."','".
                    $_POST["dni"]."','1962-09-10',
                '".$_POST["domicilio"]."','".$_POST["localidad"]."',
                '".$_POST["provincia"]."',2000.000,
                '".$_POST["telefono"]."','','',NULL)";
        $datos=@mysql_query($consulta,$id_conexion) or
                die("<CENTER><H3>No se ha podido ejecutar la
                    consulta.<P>Compruebe si la sintaxis de
                    la misma es correcta.<P></H3></CENTER>");
        echo "<CENTER><H3>El registro ha sido dado de alta
                    correctamente</H3></CENTER>";
     }
     else
       formulario_datos(0,$_POST["dni"],$_POST["nombre"],$_POST["apelli
dos"],
                        $_POST["domicilio"],$_POST["localidad"],
                        $_POST["provincia"],$_POST["telefono"],"inser
ta");
  }//end if inserta
  else
  if ($_GET["op"]=="Mostrar_todos")
  {
     $consulta="select registro,dni,nombre,apellidos,
                domicilio,localidad,provincia,telefono
                from usuarios order by registro";
     $datos=@mysql_query($consulta,$id_conexion) or
             die("<CENTER><H3>No se ha podido ejecutar la consulta.
                 <P>Revise la sintaxis de la orden.</H3></CENTER>");
     listado($datos);

  }//end Mostrar_todos
  else
  if ($_GET["op"]=="editar")
  {
     $consulta="select registro,dni,nombre,apellidos,
                domicilio,localidad,provincia,telefono
                from usuarios where registro=".$_GET["registro"];
     $datos=@mysql_query($consulta,$id_conexion) or
             die("<CENTER><H3>No se ha podido ejecutar la consulta.
                 <P>Revise la sintaxis de la orden.</H3></CENTER>");
     list($registro,$dni,$nombre,$apellidos,$domicilio,$localidad,
          $provincia,$telefono)=mysql_fetch_array($datos);

     formulario_datos($registro,$dni,$nombre,$apellidos,$domicilio,
                      $localidad,$provincia,$telefono,"actualiza");
  }//end editar
  else
  if ($_GET["op"]=="actualiza")
  {
     $consulta ="update usuarios set dni='".$_POST["dni"]."',
                     nombre='".$_POST["nombre"]."',
                     apellidos='".$_POST["apellidos"]."',
                     domicilio='".$_POST["domicilio"]."',
                     localidad='".$_POST["localidad"]."',
                     provincia='".$_POST["provincia"]."',
                     telefono='".$_POST["telefono"]."'
                  where registro=".$_POST["registro"];
```

```
    $datos=@mysql_query($consulta,$id_conexion) or
            die("<CENTER><H3>No se ha podido ejecutar la consulta.
                <P>Revise la sintaxis de la orden.</H3></CENTER>");
    echo "<CENTER><H3>El registro ha sido modificado
                correctamente</H3></CENTER>";
  } //end actualiza
  else
  if ($_GET["op"]=="borrar")
  {
    $consulta="select registro,dni,nombre,apellidos,
                domicilio,localidad,provincia,telefono
                from usuarios where registro=".$_GET["registro"];
    $datos=@mysql_query($consulta,$id_conexion) or
            die("<CENTER><H3>No se ha podido ejecutar la consulta.<P>
                 Revise la sintaxis de la orden.</H3></CENTER>");
    list($registro,$dni,$nombre,$apellidos,$domicilio,$localidad,
         $provincia,$telefono)=mysql_fetch_array($datos);
    echo "Registro $registro<BR>DNI $dni<BR>nombre $nombre<BR>
          Apellidos $apellidos<BR>Domicilio $domicilio<BR>
          Localidad $localidad<BR>Provincia $provincia<BR>
          Teléfono $telefono.";
    $consulta="delete from usuarios where
registro=".$_GET["registro"];
    $datos=@mysql_query($consulta,$id_conexion) or
            die("<CENTER><H3>No se ha podido ejecutar la consulta.<P>
                 Revise la sintaxis de la orden.</H3></CENTER>");
    echo "<CENTER><H3>El registro ha sido borrado
                correctamente</H3></CENTER>";
  }//end borrar
  else
  if ($_GET["op"]=="Buscar")
  {
    if (!isset($_POST["campo_busqueda"])) formulario_busqueda();
    else
    {
        $consulta="select registro,dni,nombre,apellidos,domicilio,
                   localidad,provincia,telefono from usuarios
                      where ".$_POST["campo_busqueda"].
                  " like   '%".$_POST["buscar_txt"]."%'";
        $datos=@mysql_query($consulta,$id_conexion) or
                die("<CENTER><H3>No se ha podido ejecutar la
                        consulta.<P>Revise la sintaxis de la orden.
                        </H3></CENTER>");
        listado($datos);
    }
  }//end if buscar

}//end if isset($_GET["op"])


pie();

?>
```

5 - Modificar la información de una base de datos

PHP permite también modificar la información que haya en una base de datos actualizando los registros de sus tablas, introduciendo nuevos registros o eliminando algunos de los que tengan éstas. Las operaciones mencionadas se llevan a cabo también mediante consultas en las que se usan sentencias SQL.

En este apartado vamos estudiar las funciones que permiten realizar consultas que modifican la información de las bases de datos. Nos serviremos, también en esta ocasión, de la base de datos "biblioteca", la misma que hemos consultado en el apartado anterior

Insertar, actualizar y borrar registros de una tabla

La función **`mysql_insert_id()`** devuelve un identificador del registro insertado cuando en el mismo hay un campo de tipo AUTO_INCREMENTED, cada vez que se ejecuta una sentencia **insert**. Su sintaxis es la siguiente:

```
mysql_insert_id([identificador de conexión])
```

Lo más recomendable es crear una variable y asignar a ésta el valor devuelto por la consulta, en la que necesariamente debe aparecer la sentencia **insert**.

Por ejemplo, si necesitamos conocer el identificador del registro insertado, debemos enviar al servidor una consulta de la base de datos "biblioteca" en la que se añada un registro a los que haya en la tabla "libros". Puede se así:

```
$consulta="insert into libros values('Branchet, Bob',
           'Microsoft SQL Server &.5','Prentice Hall',
           '1997',648,70.75,'N','INF','84-98660-99-9',
           'Buen manual','Algo antiguo',0)";

$datos= @mysql_query($consulta,$id_conexion) or
         die("<CENTER><H3>No se ha podido ejecutar
              la consulta.<P> Compruebe si la sintaxis
              de la misma es correcta.<P></H3></CENTER>");
```

A continuación, ya podemos asignar a una variable el identificador del registro insertado, que será el mismo número que el del campo "registro", que es el campo de la tabla que tiene el tipo AUTO_INCREMENTED, de esta forma:

```
$id_insertar=mysql_insert_id($id_conexion);
```

Una vez obtenido éste, ya podemos utilizarlo en una nueva consulta, por ejemplo para ver sus datos. Se hace así:

```
$consulta= "select * from libros where registro=$id_insertar";
```

Si no se especifica el identificador de conexión, que es opcional, se asume como tal el último identificador de este tipo creado. Si no hay ninguno, de forma automática la función intenta establecer un enlace como si se utilizara la función `mysql_connect()` sin argumentos, y lo utiliza.

La función `mysql_affected_rows()` devuelve el número de filas (registros) que se han visto afectadas al ejecutarse una sentencia **insert**, **update** o **delete**.

Su sintaxis es la siguiente:

```
mysql_affected_rows[identificador de conexión])
```

Lo más recomendable es crear una variable y asignar a ésta el valor devuelto por la consulta, en la que necesariamente debe aparecer una sentencia **insert**, **update** o **delete**.

Si la última sentencia ejecutada fue **delete** sin cláusula WHERE y todos los registros quedaron borrados de la tabla, esta función devuelve 0.

Esta función no es efectiva para las sentencias SELECT, sino sólo para las sentencias que modifican registros.

Las operaciones de actualizar y borrar registros se llevan a cabo igualmente mediante consultas SQL en las que se incluyen las sentencias **update** y **delete**, respectivamente, tal como se han explicado en la Unidad anterior.

En el **Ejemplo 5** de esta Unidad aparece el código completo de estas operaciones (insertar, actualizar y borrar registros de una tabla). En el mismo puede verse cómo hemos utilizado las funciones explicadas en este apartado.

En los **Ejercicios** de esta Unidad hemos procurado incluir sencillas aplicaciones de mantenimiento de tablas, de forma que en ellas se incorporen y utilicen las principales funciones abordadas en este apartado y en el anterior. Estudiando su código, leyendo los comentarios y tratando de reproducir programas similares, el alumno o alumna del curso puede asimilar adecuadamente los contenidos de esta Unidad.

También el modelo de **Proyecto final** se sirve, de una forma integrada, de estas funciones con las que PHP nos permite conectarnos a un servidor, acceder a una base de datos y tratar correctamente la información de las tablas. Ya desde este momento, cada alumno o alumna debe estudiar este proyecto e ir elaborando uno propio, que debe remitir al tutor o tutora antes de presentarse a la prueba final presencial.

6 - Instalación y uso de MyODBC

Instalación y configuración

En este apartado se explica el proceso de instalación del controlador (*driver*) MyODBC 32 bits para Windows. Esto nos permitirá acceder desde un cliente Windows que disponga de MS Access a un servidor MySQL.

Suponemos que el gestor de datos MS Access está ya instalado en el ordenador donde se realiza el curso. Si no lo estuviera, sería imposible llevar a cabo los pasos explicados a continuación.

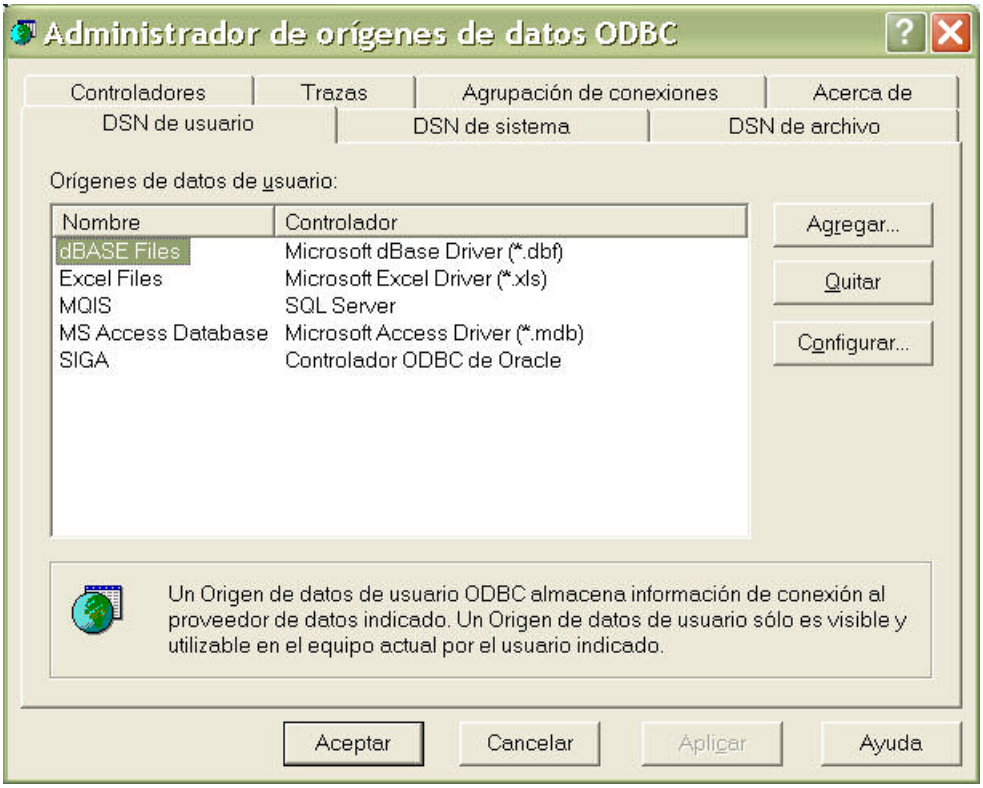
En el CD-ROM curso hemos incluido el driver MyODBC más reciente para Windows. Si el alumno o alumna precisa el controlador de versiones superiores de este entorno o quiere disponer en su día de una versión más actualizada de este controlador, puede recogerse en la

dirección http://www.mysql.com/. El fichero del controlador incluido en el curso se denomina `MyODBC-3.51.exe` y está en el directorio `C:\cursoPHP5\bin\MySQL4.1\MyODBC`.

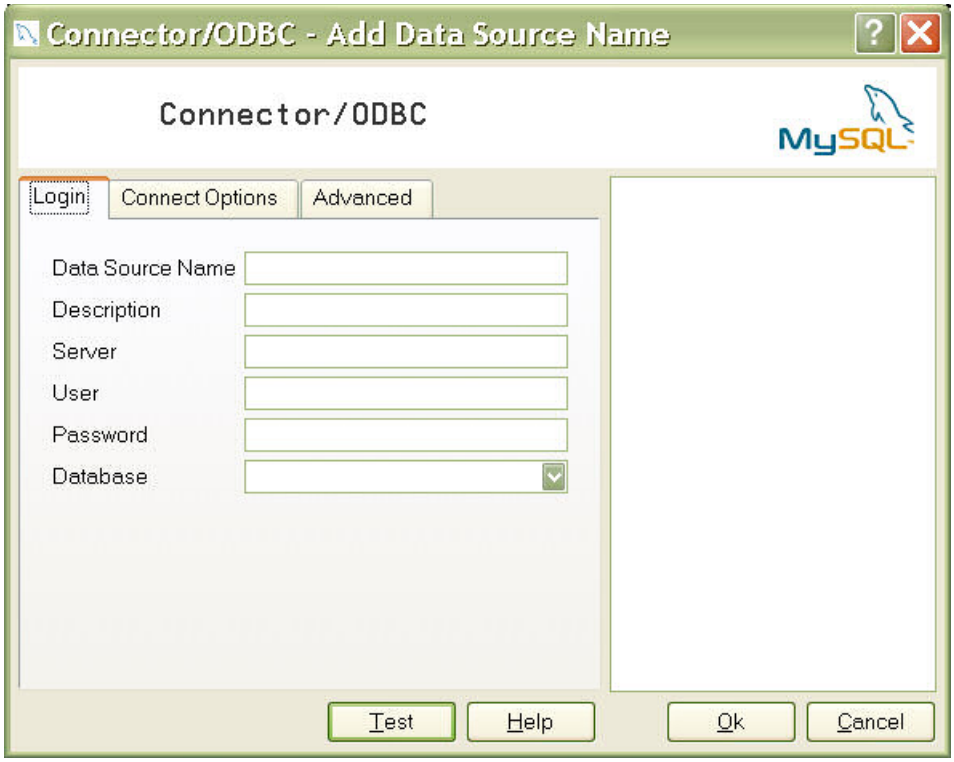
Queremos advertir que el controlador MyODBC no se instala con el curso y, por tanto, el alumno no lo hallará ya funcionando en el ordenador donde trabaje. Si quiere conectar una base de datos de tipo MySQL desde MS Access, debe instalar previamente el controlador tal como indicamos a continuación.

Para convertir nuestro MS Access en un cliente Windows de MySQL, en primer lugar debemos instalar el controlador ejecutando el fichero `MyODBC-3.51.exe`.

Una vez instalado el *driver* ODBC para MySQL, debemos ejercutar la utilidad Administrador de Orígenes de Datos ODBC que se encuentra en el Panel de Control (si dispones de Windows 2000 ó XP estará dentro de la la carpeta Herramientas Administrativas). Se mostrará la siguiente ventana



A continuación, dentro de la pestaña DNS de usuario , pulsamos sobre el botón Agregar, para añadir un nuevo origen de datos y seleccionamos el *driver MySQL ODBC 3.51 Driver* correspondiente a la base de datos MySQL y pulsamos el botón Finalizar apareciendo la siguiente ventana



Seguidamente, escribimos los valores siguientes, suponiendo, por ejemplo, que vamos a conectarnos con la base de datos "biblioteca" del servidor MySQL y pulsamos el botón OK:

• **Data Source Name**: nombre que daremos a la conexión. En nuestro ejemplo, ponemos CursoPHP5.

• **Server**: es el nombre o dirección IP del servidor al que nos queremos conectar. En nuestro ejemplo, ponemos localhost.

• **User**: nombre de usuario. En nuestro ejemplo, ponemos root.

• **Password**: clave del usuario. En nuestro ejemplo, lo dejamos en blanco.

• **Database**: nombre de la base de datos MySQL a la que queremos conectarnos. En nuestro ejemplo, ponemos biblioteca.

Conexión remota a MySQL desde MS Access

Veamos cómo podemos conectarnos a una base de datos MySQL que está en un servidor remoto usando un cliente Windows a través de MS Access. Para poder hacerlo, es preciso tener instalado en nuestro cliente Windows el *driver* MyODBC, así como haber arrancado el servidor MySQL.

Debemos seguir estos pasos:

1. Arrancar MS Access.

2. En el menú Archivo seleccionamos Abrir.

3. En la ventana de diálogo Abrir, apartado Tipo de archivo, seleccionamos ODBC

Databases().

4. Para seleccionar el origen de los datos al que nos vamos a conectar, en la pestaña Origen de datos de equipo escogemos en la columna Nombre del origen de datos el nombre que pusimos en la opción Windows DNS name, que en nuestro ejemplo es CursoPHP5, que creamos en la instalación de MyODBC o desde el Panel de control de Windows.

5. Una vez que hemos realizado la conexión a nuestra base de datos remota MySQL, en la ventana Vincular tablas, debemos escoger las tablas que queremos vincular. Si las tablas vinculadas no tienen una clave primaria, MS Access nos pedirá que elijamos una.

Queremos recordar que para poder modificar las tablas desde MS Access hay que tener los permisos pertinentes como usuarios de MySQL.

Exportar un tabla de tipo MS Access a otra de tipo MySQL

Ahora vamos a ver cómo podemos exportar una tabla de una base de datos MS Access a otra base de datos remota MySQL mediante ODBC. Para poder hacerlo es preciso tener instalado en nuestro cliente Windows el *driver* MyODBC, así como haber arrancado el servidor MySQL.

Debemos seguir estos pasos:

1. Arrancamos MS Access y abrimos una tabla de una base de datos.

2. En el menú Archivo seleccionamos Exportar.

3. En la ventana de diálogo, apartado Guardar como tipo, seleccionamos ODBC Databases().

4. En la ventana que aparece se indica el nombre que va tener la tabla exportada en la base de datos MySQL.

5. Para seleccionar el destino de los datos nos conectamos y, en la pestaña Origen de datos de equipo, escogemos en la columna Nombre del origen de datos el nombre que pusimos en la opción Windows DNS name, que en nuestro ejemplo anterior era CursoPHP5.

Una vez que ya tenemos la tabla original en formato MySQL, seguramente necesitaremos desde el servidor MySQL, con la aplicación Cliente MySQL, arreglar el nombre o el tipo de algún campo, los índices, claves, etcétera.

Importar una tabla de tipo MySQL a otra de MS Access

Ahora vamos a ver cómo desde MS Access podemos importar una tabla de tipo MySQL mediante ODBC. Para poder hacerlo es preciso tener instalado en nuestro cliente Windows el *driver* MyODBC, así como haber arrancado el servidor MySQL.

Debemos seguir estos pasos:

1. Arrancamos MS Access y abrimos la base de datos donde deba incorporarse la tabla MySQL.

2. En el menú Archivo seleccionamos Obtener datos externos, opción Importar.

3. En la ventana de diálogo Importar, apartado Tipo de archivo, seleccionamos ODBC DataBases().

4. Para seleccionar el origen de los datos MySQL, en la pestaña Origen de datos de equipo, escogemos en la columna Nombre del origen de datos el nombre que pusimos en la opción Windows DNS name, que en nuestro ejemplo anterior era CursoPHP5.

5. Una vez que hemos realizado la conexión a nuestra base de datos remota MySQL, en la ventana Importar tablas, debemos escoger las tablas que queremos importar.

Conviene tener en cuenta que en la operación **vincular** sólo se da acceso a las tablas MySQL desde MS Access, para poder tratar su información a través del controlador ODBC. En cambio, en las operaciones **importar** y **exportar**, las tablas afectadas cambian de formato de tipo de bases de datos, convirtiéndose en bases de datos de tipo MySQL, en el primer caso, o de MS Access en el segundo.

7 - Resumen

Hay que saber al final de esta unidad

- **Conectarse** al servidor que contiene el gestor de bases de datos MySQL.
- **Crear bases de datos, conectarse a ellas** y **destruirlas,** con sus tablas e información.
- **Utilizar las funciones que controlan los posibles errores** al realizar las operaciones anteriores.
- **Consultar una base de datos** utilizando *queries* en las que intervengan las principales funciones de PHP para acceder a bases de datos de tipo MySQL.
- **Insertar, actualizar** y **eliminar** registros de una tabla.
- **Usar formularios** y **celdas de tablas** para mostrar información de una tabla, preguntarla o seleccionar alguno de sus campos.
- **Instalar el controlador MyODBC** para poder conectar bases de datos de tipo MS Access con las de tipo MySQL.
- **Importar o exportar** tablas entre ambos gestores de bases de datos.
- **Mantener las tablas** de una base de datos realizando a través de funciones, tanto estándar de PHP para MySQL como de usuario, las principales operaciones

```
de este tipo: altas, bajas, consultas, búsquedas,
actualizaciones, etcétera.
```