Servidores Linux en centros educativos

21 de mayo de 2005

Índice general

PHP y bases de datos. MySQL. Plataformas: instalación y configuración.

Capítulo 1

PHP.

1.1. Repaso de conceptos previos: HTML

Esta parte de los apuntes esta dirigida a aquellos que han realizado paginas web mediante editores (FrontPage, Composer,...) y que no se han preocupado en exceso por las etiquetas que estos programas generaban, para aquellos que deseen repasar las etiquetas de HTML y para aquellos que no saben que es el HTML. En cualquier caso no hace falta memorizar las etiquetas pero si saber utilizarlas para lo que nos viene despues : el php.

1.1.1. El Lenguaje HTML

1.1.1.1. Las Etiquetas

El lenguaje HTML es un lenguaje de marcas, estas marcas serán fragmentos de texto destacado de una forma especial que permiten la definición de las distintas instrucciones de HTML, tanto los efectos a aplicar sobre el texto como las distintas estructuras del lenguaje. A estas marcas las denominaremos etiquetas y serán la base principal del lenguaje HTML. En documento HTML será un fichero texto con etiquetas que variarán la forma de su presentación.

Una etiqueta será un texto incluido entre los símbolos menor que <y mayor que >.. El texto incluido dentro de los símbolos será explicativo de la utilidad de la etiqueta. Por ejemplo:

<B>Letra Negrita, del inglés Bold (negrita).

<TABLE>Definirá una tabla.

<IMG>Inclusión de una IMaGen.

Existe normalmente una etiqueta de inicio y otra de fin, la de fin contendrá el mismo texto que la de inicio añadiéndole al principio una barra inclinada /. El efecto que define la etiqueta tendrá validez para todo lo que este incluido entre las etiquetas de inicio y fin, ya sea texto plano o otras etiquetas HTML.

<ETIQUETA>Elementos Afectados por la Etiqueta</ETIQUETA>

Por ejemplo, con la etiqueta siguiente:

<B>Texto que será en negrita</B>.

Obtendremos:

Texto que será en negrita

Algunas etiquetas no necesitarán la de fin, serán aquellas en las que el final este implícito, por ejemplo <P>párrafo,
salto de línea ó <IMG>inclusión de una imagen. Definen un efecto que se producirá en un punto determinado sin afectar a otros elementos.

El uso de mayúsculas o minúsculas en las etiquetas es indiferente, se interpretarán del mismo modo en ambos casos, pero lo normal es expresarlas en mayúsculas para que destaquen con más nitidez del texto normal.

Las etiquetas pueden presentar modificadores que llamaremos atributos que permitirán definir diferentes posibilidades de la instrucción HTML. Estos atributos se definirán en la etiqueta de inicio y consistirán normalmente en el nombre del atributo y el valor que toma separados por un signo de igual. El

orden en que se incluyan los atributos es indiferente, no afectando al resultado. Si se incluyen varias veces el mismo atributo con distintos valores el resultado obtenido será imprevisible dependiendo de como lo interprete el navegador. Cuando el valor que toma el atributo tiene más de una palabra deberá expresarse entre comillas, en otro caso no será necesario.

Un ejemplo de atributo será:

<A HREF="http://www.cepalcala.org">Pagina principal del CEP</A>

En este caso la etiqueta A presenta un atributo HREF cuyo valor es http://www.cepalcala.org

Igualmente una etiqueta podría presentar varios atributos:

<HR ALIGN=LEFT NOSHADE SIZE=5 WIDTH=50 %>

En este caso la etiqueta HR presenta cuatro atributos. El segundo atributo NOSHADE es un caso especial que no presenta valor. El orden en que se especifiquen los atributos no afectarán al resultado final.

Todo texto que se encuentre entre los caracteres <y >se considerará una etiqueta, si la etiqueta no fuera una de las validas del lenguaje HTML no será tenida en cuenta, sin causar ningún tipo de error. Dejándose el texto o las etiquetas a las que afectaba como si no existiera la etiqueta extraña. Cuando se comete un error sintáctico al expresar una etiqueta o un atributo no se producirá ningún error, simplemente no de obtendrá el efecto que deseábamos.

El lenguaje HTML es un lenguaje que evoluciona muy rápidamente y cada nueva versión de los programas navegadores presenta etiquetas nuevas que causan efectos más espectaculares o atributos nuevos de las etiquetas ya existentes. Esto causa que los programas más antiguos no entiendan estas nuevas etiquetas y por tanto las considere erróneas y no realice la acción que deseábamos. Dándose el caso de atributos que son validos solo para un único navegador.

Cuando creemos código HTML hay que hacerlo lo más estándar posible para permitir que el documento pueda ser visto de forma efectiva por distintos navegadores en maquinas distintas. Por tanto debemos renunciar a efectos espectaculares que solo tienen validez en un navegador e intentar comprobar como se ve el documento en una variedad de navegadores, ya que las personas que se conectan a nuestras páginas no tendrán en la mayoría de los casos el mismo que nosotros. Tambien es interesante como se vería el documento en los distintos tamaños de la ventana del navegador, teniendo en cuenta que todos no tienen un monitor con la misma resolución.

En este manual se han tratado de incluir las características más estándar de HTML.

1.1.1.2. Estructura de un documento HTML

Un documento HTML está definido por una etiqueta de apertura <HTML>y una etiqueta de cierre </HTML>. Dentro de este se dividen dos partes fundamentales la cabecera, delimitada por la etiqueta <HEAD>y el cuerpo, delimitado por la etiqueta <BODY>. Por tanto la estructura de un documento HTML será:

<HTML>
<HEAD>
Definiciones de la cabecera
</HEAD>
<BODY>
Instrucciones HTML
</BODY>
</HTML>

Ninguno de estos elementos es obligatorio, pudiendo componer documentos HTML que se muestren sin ningún problema sin incluir estas etiquetas de identificación. Si se utilizan elementos que forzosamente deban ser incluidos en la cabecera (como la etiqueta de titulo), no serán reconocidos correctamente si no se incluyen entre las etiquetas de <HEAD>.

Para insertar comentarios dentro de un documento HTML utilizaremos la etiqueta especial <!–, definiéndose un comentario de la forma:

<!– Esto es un comentario –>

1.1.1.3. Atributos de BODY

La etiqueta BODY presenta algunos atributos que son de definición global para todo el documento, estos definirán los colores y el fondo del documento HTML.

Los atributos de BODY son:

<BODY BACKGROUND="URL" BGCOLOR=#rrvvaa TEXT=#rrvvaa LINK=#rrvvaa VLINK=#rrvvaa>

BACKGROUND="URL":

Definirá la imagen que se utilizará de fondo del documento HTML, la URL definida será el camino a una imagen. Esta se muestra debajo del texto y las imágenes del documento HTML. En el caso de que la imagen no rellene todo el fondo del documento esta será reproducida tantas veces como sea necesario.

BGCOLOR=#rrvvaa ó nombre del color:

Indicará el color del fondo del documento HTML, solo se utilizará si no se ha definido una imagen de fondo, o si esta imagen no puede obtenerse.

TEXT=#rrvvaa ó nombre del color:

Especificará el color del texto normal dentro del documento HTML. Por defecto será normalmente negro.

LINK=#rrvvaa ó nombre del color:

Indicará el color que tendrán los hiperenlaces que no han sido accedidos. Por defecto será azul.

VLINK=#rrvvaa ó nombre del color:

Color de los enlaces que ya han sido visitados. Por defecto es un color azul más oscuro.

1.1.1.4. El espaciado en HTML

En HTML no está permitido más de un elemento blanco (espacios, tabuladores, saltos de línea) separando cualquier elemento o texto, todos estos son convertidos a un único espacio blanco y el resto se omiten en la representación del documento. En el documento fuente podremos usar el espaciado que deseemos, y no deberá estar bien formateado, este se conseguirá con las etiquetas HTML.

Existen unas etiquetas especiales HTML para definir estos elementos de control de texto. A continuación se detallará cada una de ellas.

<P>Cambio de párrafo :

Definirá un párrafo, se usará al comienzo o al final de un párrafo de texto e introducirá un espaciado de separación (normalmente dos líneas) con el próximo texto que se exprese. Esta etiqueta se puede utilizar para introducir un espaciado entre cualquier elemento HTML y no solo servirá para separar texto.

El efecto se conseguirá introduciendo la etiqueta <P>en el punto en el que deseemos que se produzca el espaciado. La etiqueta de fin de párrafo </P>es opcional no siendo necesario incluirla.

Salto de línea :

Su utilidad es similar al anterior pero en este caso el espaciado del texto es menor, se pasará a la línea siguiente, sin dejar una línea de separación. En este caso será un cambio de línea y no de parrafo. Igualmente no es necesario usarlo tras los elementos que llevan implícitos un salto de línea, ni tampoco es necesaria la etiqueta de fin </BR>.

  Espacios en blanco :

Con esta secuencia de caracteres conseguiremos espacios en blanco que se mostrarán de forma efectiva, pudiendo mostrar más de un espacio en blanco de separación. Se incluirán tantas expresiones   como espacios en blanco se desee conseguir.

1.1.1.5. Tamaño y Color de las fuentes de caracteres

Con esta eqtiqueta podremos incluir texto resaltado en medio de una frase.

La etiqueta que permite esto se llama FONT y presenta atributos que nos permiten modificar el tamaño y color del texto incluido entre la etiqueta de inicio y fin.

<FONT SIZE=n>: Tamaño de la fuente

El atributo SIZE permite indicar el tamaño de la fuente, su valor puede estar entre 1 y 7. Incrementándose de tamaño progresivamente desde 1, que es la fuente de menor tamaño, hasta 7 que la fuente de

mayor tamaño. El texto normal equivale a la fuente de tamaño 3, por tanto los valores menores que 3 serán fuentes más pequeñas que el texto normal y las mayores que 3 serán de mayor tamaño.

El tamaño también puede indicarse de forma relativa, indicando el incremento o detrimento a partir de la fuente base. Por defecto la fuente base será 3, por tanto si se indica como valor +1 la fuente será de tamaño 4.

Otras etiquetas que se aplicarán a las fuentes son aquellas que nos permiten cambiar el aspecto de la misma por ejemplo poniendo el texto en negrita, subrayado,... El efecto se aplicará al texto expresado entre la etiquetas de inicio y fin. Las eqtiquetas son las siguientes:

<B>Negrita.
<I>Cursiva.
<TT>Maquina de escribir, muestra una fuente de caracteres de espaciado fijo.
<BLINK>Parpadeo.
<SUB>Subíndice. Para Netscape 2.0+
<SUP>Superíndice. Para Netscape 2.0+
<BIG>Texto grande, se utilizará el mayor tamaño de fuente. Para Netscape 2.0+
<SMALL>Texto pequeño, se utilizará la fuente de menor tamaño. Para Netscape 2.0+

1.1.1.6. Hiperenlaces

Es la utilidad básica del hipertexto, permite indicar zonas de texto o imágenes que si son seleccionados por el lector del documento nos traslada a otros documentos HTML o otras zonas del documento actual.

Para definir un hiperenlace podemos utilizar cualquier elemento HTML, no debe ser texto necesariamente, podemos usar, cabeceras (<Hn>), cualquiera de los estilos, una imagen, etc .... Un hiperenlace igualmente podrá definirse dentro de cualquier elemento HTML: listas, párrafos de texto, tablas, formularios.

La forma de indicarlo será:

<A HREF="URL a la que se accede">Texto del Hiperenlace</A>

El texto indicado entre las etiquetas de comienzo y de fin se presentará de forma resaltada y en el caso de seleccionar este texto el documento actual cambiará por el especificado en la URL.

1.1.1.7. Imágenes

Una de las características principales del lenguaje HTML y de la WWW es la introducción de elementos multimedia, en este apartado veremos como introducir gráficos y ficheros de imágenes en un documento HTML.

En un documento HTML se puede incluir cualquier imagen en alguno de los siguientes formatos gráficos: GIF, JPEG ó XBM. El formato más extendido y practico es el GIF.

La etiqueta encargada de mostrar imágenes en HTML es IMG y tiene el siguiente formato:

<IMG SRC="URL de la Imagen" ALT="Texto alternativo a la imagen">

El atributo SRC indica el fichero de imagen que se incluirá en el documento.

1.1.1.8. Tablas

Permite la representación de datos por filas y columnas, en forma tabular. La definición es muy flexible indicando solo los elementos que forman cada fila y columna, calculándose de forma automática el tamaño que deben tener las celdas. En una tabla podemos introducir todo tipo de elementos del lenguaje HTML como imagenes, enlaces, texto, listas, cabeceras, formularios, etc.

No es necesario definir inicialmente el número de filas o columnas ya que estas se calculan según se va definiendo la tabla. En el caso que una fila tenga más columnas que otra, en las otras filas las columnas se representarán vacias, no siendo necesario que todas las filas sean iguales.

<TABLE>Definición de la Tabla

Para definir una tabla usaremos la etiqueta <TABLE>, que tiene el siguiente formato:

<TABLE BORDER="tamaño del borde" >

... Definición de la tabla ...
</TABLE>
<TR>Fila de la tabla

Definirá cada una de las filas de la tabla y especificará los parámetros que afectarán a todas las celdas de la fila. Por cada elemento TR que se incluya se creará una fila de la tabla. No es necesario indicar la etiqueta de cierre </TR>. En caso de tablas anidadas será necesario incluir la etiqueta de cierre.

<TH>y <TD>Una celda de la tabla

Define cada una de las celdas de una fila de la tabla, TH se usará para definir una celda de tipo cabecera, en este caso se mostrarán destacados en negrita y TD para definir una celda de datos. Estos elementos deben aparecer tras los elementos TR para definir cada una de las columnas de la fila. Existirá una columna por cada elemento TD ó TH que se defina.

1.1.1.9. Formularios (FORM).

Los formularios son plantillas que permiten la creación de documentos HTML con peticiones de datos. La principal utilidad de los formularios es la posibilidad de crear cuestionarios, encuestas, páginas de comentarios o cualquier documento en la que se desee una interacción por parte del usuario.

Se podrán definir distintos tipos de recuadros de dialogo, botones de selección, menús de múltiples opciones, ... Para permitir obtener los datos de una manera más intuitiva.

<FORM>Definición de formularios

Existe una instrucción HTML para la creación de formularios esta es FORM, que tiene la siguiente estructura:

<FORM ACTION="fichero que trata el formulario" METHOD= POST | GET >
...
Elementos que forman el formulario
...
</FORM>

Dentro de la etiqueta de formulario se definirán los distintos elementos de petición de datos. Estas instrucciones que se explicarán a continuación definirán los tipos de botones, cajas de dialogo y ventanas para la introducción de datos. Y definirán las variables que almacenarán los datos introducidos por el usuario. Estas etiquetas se incluirán entre la de definición del formulario y la etiqueta de final de formulario.

Los atributos que presenta la etiqueta FORM son los siguientes:

ACTION:

Indica el programa que se encargará de tratar los datos del formulario. Este programa debe encontrarse en el servidor y estar escrito en algún lenguaje de programación. A este programa se pasará como parámetros los datos introducidos en el formulario y retornará un código HTML que se mostrará tras procesar el formulario. A este tipo de programas se les llama cgi-bin.

METHOD:

Indica el protocolo usado para el envío de los datos. Con POST envía los datos en la entrada estándar del programa que trata el formulario y con GET los datos se pasan por parámetro, en la línea de comandos, al programa. El usar uno o otro método vendrá determinado por como son tratados los parámetros en el formulario en el (CGI-BIN). El método de uso más normal será POST.

Una vez definidas las características globales del formulario incluiremos los distintos botones y cajas de dialogo que lo constituyen. Dentro de la instrucción del formulario podrán incluirse cualquier texto o instrucción HTML, siendo recomendado a fin de poder etiquetar las opciones de entrada y especificar cualquier dato importante relacionado con el formulario. Igualmente un formulario puede ser incluido en algunas instrucciones HTML como las listas, tablas, etc ...

<INPUT>Entrada básica de datos

La etiqueta INPUT se utiliza para definir gran variedad de tipos de campos de entrada de datos. Por lo general serán entradas de texto corto (a lo sumo una frase) o opciones. El formato básico es el siguiente:

<INPUT TYPE = (TEXT | PASSWORD | CHECKBOX | RADIO | HIDDEN | SUBMIT | IMAGE | RESET) NAME = "Variable que toma el valor" VALUE = "Valor de Inicialización" >

El atributo TYPE se usa para determinar el tipo de recuadro de dialogo de entrada que se está definiendo, a continuación se explicarán por separado cada una de las opciones. El atributo NAME especifica el nombre de la variable que se define. Este nombre será pasado al programa que trata el formulario junto con el valor que le asigno el usuario del formulario. El atributo VALUE suele especificar el valor de inicialización, que será el valor por defecto.

A continuación se relatan los distintos tipos de instrucciones de entrada.

<INPUT TYPE=TEXT...>Texto corto

Se utiliza para la entrada de cadenas de texto corto, como por ejemplo nombre de personas, números, fechas o diversos datos que se puedan expresar en una línea de texto.

Se mostrará un recuadro que ocupa una línea y la que será posible especificar este texto. El formato de la instrucción es el siguiente:

<INPUT TYPE=TEXT NAME="variable" VALUE="valor inicial" SIZE="tamaño" MAXLENGTH="longitud máxima" >

Otro tipo será el siguiente:

<INPUT TYPE=PASSWORD...>Palabras secretas

Es similar al anterior pero en este caso no se imprimen los caracteres según se van introduciendo, se muestra un asterisco en vez de los caracteres. Solo se puede ver el número de caracteres, pero no valor. Se usa para la introducción de claves de acceso (passwords) y datos que no deban ser vistos al introducirlos. El formato es:

<INPUT TYPE=PASSWORD NAME="variable" VALUE="valor inicial" SIZE="tamaño" MAXLENGTH="longitud máxima" >

Otra opción serán los Botones de selección

El checkbox es un botón que puede presentar dos estados activado o desactivado. El formato es el siguiente:

<INPUT TYPE=CHECKBOX NAME="variable" CHECKED>

Otra opción será la Selección entre múltiples opciones

Se usa cuando la opción puede tomar un valor simple de una serie de alternativas. En este caso se presentan unos valores opcionales de los que solo puede tomar un valor. Para especificar cada uno de estos valores se incluirá una etiqueta RADIO por cada una de las posibles alternativas, su estructura general será:

<INPUT TYPE=RADIO NAME="variable" VALUE="valor 1" CHECKED >
<INPUT TYPE=RADIO NAME="variable" VALUE="valor 2" >
. . .
<INPUT TYPE=RADIO NAME="variable" VALUE="valor n" >

Cada una de las etiquetas RADIO tendrá el mismos atributo NAME, y con un distinto atributo VALUE que será el valor que tome si se selecciona esta opción. Para inicializarlo se usa el atributo CHECKED que se indicará solo en la opción que se quiera especificar por defecto.

El botón para enviar datos.

Este botón se usa para enviar los datos del formulario, al pulsar el usuario este botón, se acaba la introducción del formulario y pasa el control al programa indicado en ACTION. En todo formulario debe existir al menos un botón de SUBMIT, si solo incluye un recuadro del tipo TEXT no será necesario incluirlo. El formato es:

<INPUT TYPE=SUBMIT VALUE="mensaje a mostrar" >

Elección entre múltiples opciones

Se usa para menús simple o múltiples. Define menús de tipo pop-up (menú de barras) y ofrece una alternativa más compacta al uso de botones RADIO o CHECKBOX. Su formato es el siguiente:

<SELECT NAME="variable">
<OPTION SELECTED VALUE=valor1>Opción Primera
<OPTION VALUE=valor2>Opción Segunda
. . .
<OPTION VALUE=valorn>Opción Enésima
</SELECT>

1.2. ¿Que es php?

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.

El lenguaje PHP es un lenguaje de programación de estilo clásico, con esto quiero decir que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones.... No es un lenguaje de marcas como podría ser HTML, XML o WML. Está mas cercano a JavaScript o a C, para aquellos que conocen estos lenguajes.

Pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

1.2.1. ¿Qué se puede hacer con PHP?

Si podemos realizar paginas Web ¿para que usar entonces php? La razones son muchas pero la más evidente es la comentada anteriormente aunque ahora citamos de nuevo **crear páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.**

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Existen tres campos en los que se usan scripts escritos en PHP.

*Scripts del lado del servidor. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó módulo), un servidor web y un navegador. Es necesario correr el servidor web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor web. Consultar la sección Instrucciones de instalación para más información.

*Scripts en la línea de comandos. Puede crear un script PHP y correrlo sin ningún servidor web o navegador. Solamente necesita el intérprete PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto. Consultar la sección Usos de PHP en la línea de comandos para más información.

*Escribir aplicaciones de interfaz gráfica. Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si conoce bien PHP, y quisiera utilizar algunas características avanzadas en programas clientes, puede utilizar PHP-GTK para escribir dichos programas. También es posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal. Si está interesado en PHP-GTK, puedes visitar las páginas web del proyecto.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercadoPHP y soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

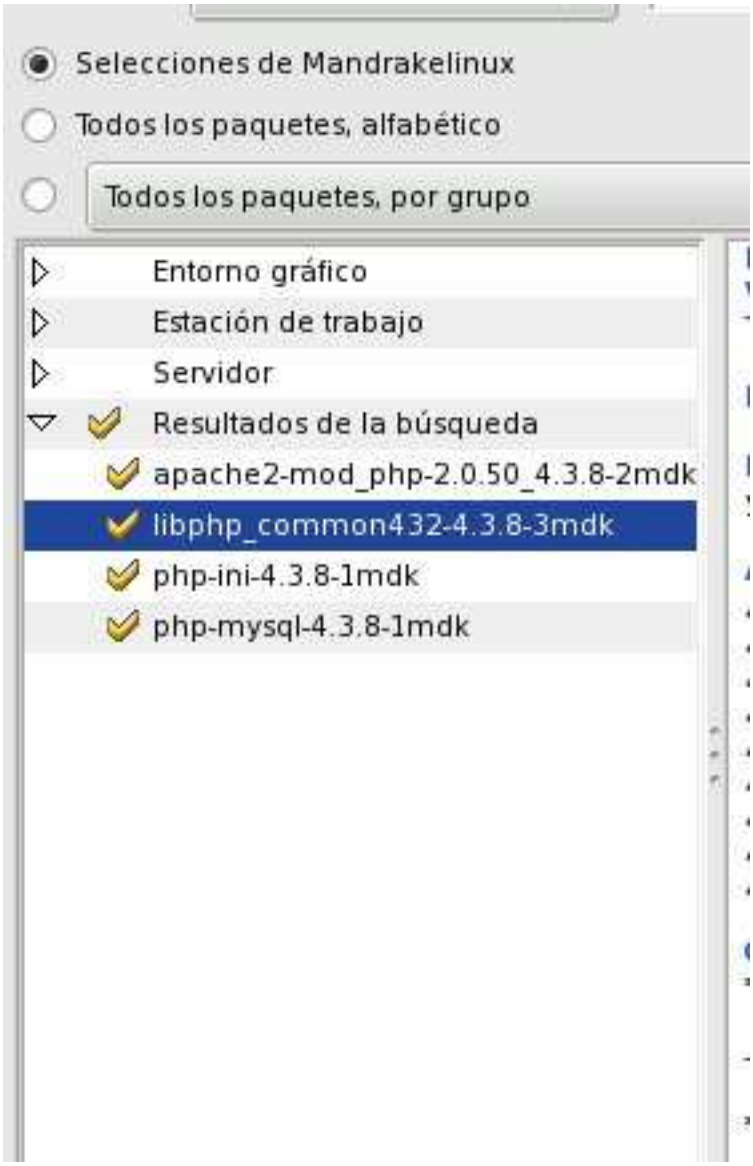
Adabas D Ingres Oracle (OCI7 and OCI8)dBase InterBase Ovrimos Empress FrontBase PostgreSQL FilePro (read-only) mSQL Solid Hyperwave Direct MS-SQL Sybase IBM DB2 MySQL Velocis Informix ODBC Unix dbm

1.2.2. Instalación de php

En cualquiera de los sistemas partimos que tenemos instalado un servidor Web Apache y MYSQL.

1.2.2.1. Instalación en Mandrake

La instalación se puede realizar tomando como base el Centro de control de Mandrake. En este caso bastaría con tomar los paquetes que se muestran a continuación e instalarlos.



En cualquier caso la instalación es bastante sencilla y para que todo funcione desde un principio deberemos de realizar muy pocas modificaciones. Como veis se esta instalando en este caso la versión 4 de php pero si los CDs que acompañan a vuestra distribución tuviesen una version inferior no existe ningun problema.[1]

1.2.2.2. Instalación en Guadalinex

La instalación desde Guadalinex se puede hacer usando el entorno gráfico (Synoptic) pero vamos a realizar la misma desde la consola y siempre como usuario root escribimos:

apt-get install libapache2-mod-php4 php4-pear php4-cgi phpdoc

Una vez instalado reiniciamos el servidor con

[1] Desde la consola mediante el comando rpm o urpmi se instalarán los paquetes siguientes (se listan los correspondientes a Mandrake 9):
php-cli-4.3.1-11mdk.i586.rpm
php-mysql-4.3.0-2mdk.i586.rpm
apache2-mod_php-2.0.50_4.3.8.2mdk.i586.rpm
php-ini-4.3.8-1.mdk.i586.rpm
En cualquier caso upmpi nos indicará las depencias que necesitemos

apache2ctl graceful
Ahora instalamos el módulo que enlaza php y mysql
apt-get install php4-mysql

1.2.2.3. El archivo de configuración.

El archivo de configuración (llamado php3.ini en PHP 3, y simplemente php.ini a partir de PHP 4) es leído cuando arranca PHP. Para las versiones de PHP como módulo de servidor esto sólo ocurre una vez al arrancar el servidor web. Para la versión CGI y CLI, esto ocurre en cada llamada. Este fichero se encuentra en el directorio /etc/

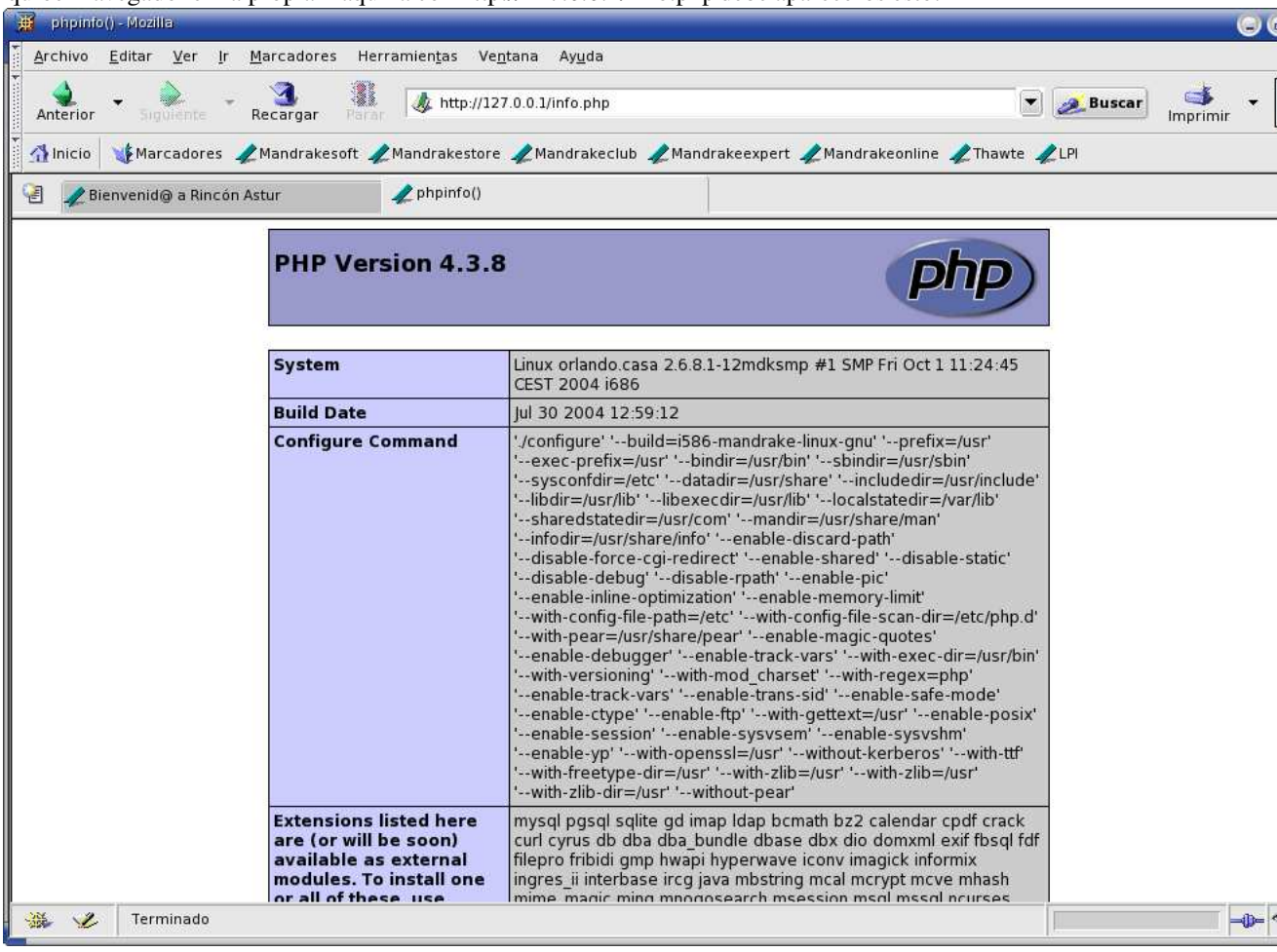
Cuando se usa PHP como un módulo de Apache, se pueden cambiar valores de la configuración usando directivas en los archivos de configuración de apache, httpd.conf y .htaccess. Necesitará de los privilegios "AllowOverride Options" o "AllowOverride All" para hacerlo.

En principio una vez instalado lo único que necesita es reiniciar el servidor apache para que se tomen los nuevos cambios o mejor dicho las nuevas configuraciones de apache + php + mysql.

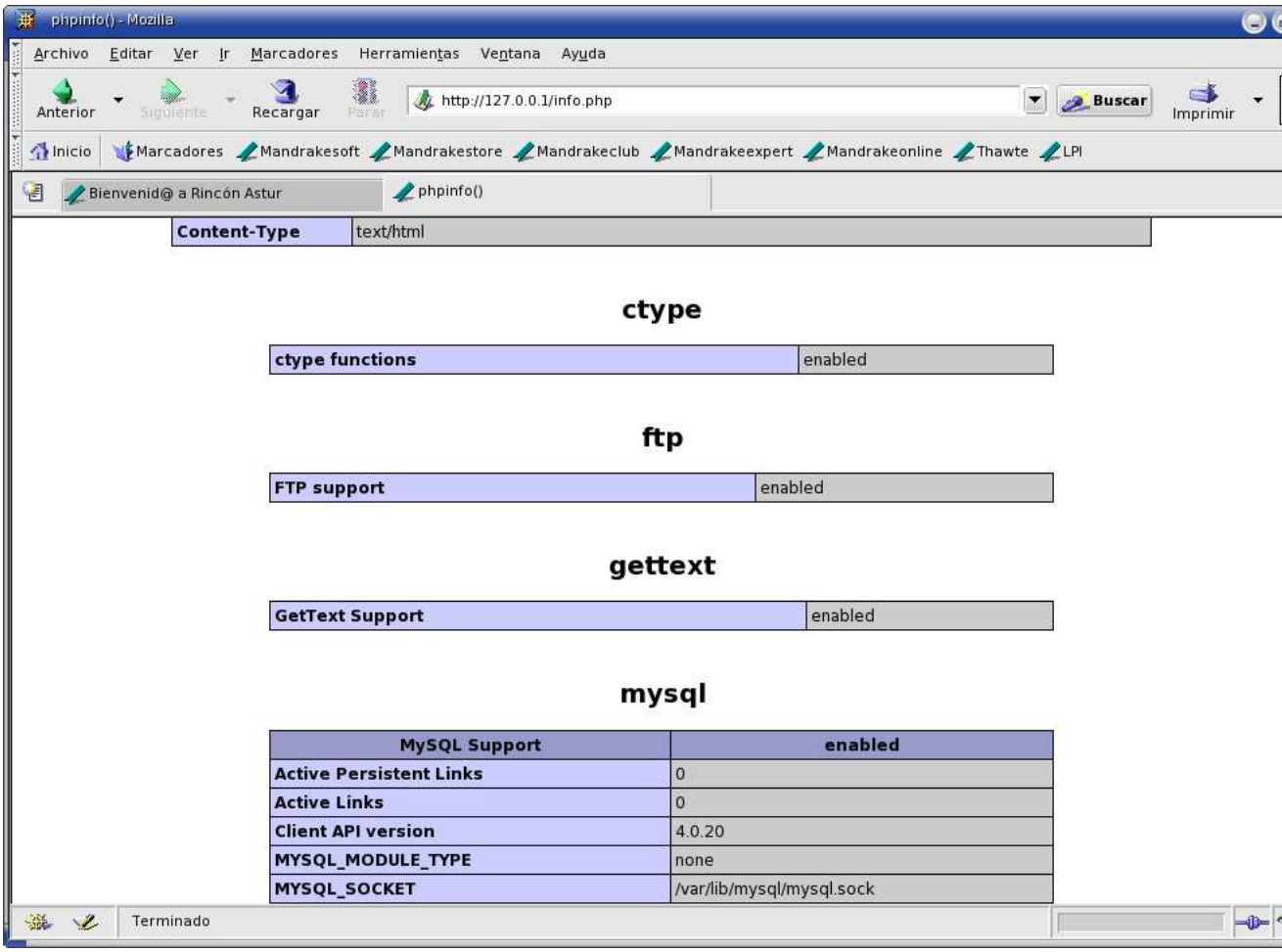
Antes que nada vamos a ver las funcionalidades de php que tenemos en nuestra máquina para ello nos crearemos un primer fichero que llamaremos info.php cuyo contenido debe ser este:

<? phpinfo(); ?>

y lo guardaís en un sitio donde sea accesible desde el navegador por ejemplo en /var/www/html. Desde cualquioer navegador en la propia máquina con http://127.0.0.1/info.php debe apareceros esto:



Es una página web con toda la información que se dispone sobre php en nuestro sistema, si bajamos y todo nos fué correctamente debemos observar como esta instalado el módulo de mysql:



1.2.3. Programación en php

En fin hasta ahora no lo he comentado pero todos los archivos que realicemos con php deben tener extensión php.[2]

1.2.3.1. Su primera página con PHP

Comienze por crear un archivo llamado hola.php y colocarle en el "directorio raíz" (DOCUMENT_ROOT) con el siguiente contenido:[3]

Ejemplo 1. Nuestro primer script PHP: hola.php

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php echo "Hola Mundo"; ?>
</body>
</html>
```

A traves de l navegador web debes acceder al archivo, con la URL terminando en "/hola.php". Si está programando localmente este URL funcionará algo como http://localhost/hola.php o http://127.0.0.1/hola.php

[2] Esta escrito en minúsculas y no en mayúsculas.

[3] o bien en cualquier directorio a partir del DOCUMENT_ROOT, para que tengamos acceso www.

pero esto depende de la configuración del servidor web. Si todo está configurado correctamente, el archivo será analizado por PHP y el siguiente contenido aparecerá en su navegador:

Hola mundo

El ejemplo anterior realiza lo mismo que:

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<p>Hola Mundo</p>
</body>
</html>
```

Este programa es extremadamente simple, y no necesita usar PHP para crear una página como ésta. Todo lo que hace es mostrar: Hola Mundo usando la sentencia echo().

Si ha intentado usar este ejemplo, y no produjo ningún resultado, preguntando si deseaba descargar el archivo, o mostró todo el archivo como texto, lo más seguro es que PHP no se encuentra habilitado en su servidor. Revisa la instalación de los paquetes.

El objetivo de este ejemplo es demostrar cómo puede usar las etiquetas PHP. En este ejemplo usamos <?php para indicar el inicio de la etiqueta PHP. Después indicamos la sentencia y abandonamos el modo PHP usando ?>. Puede salir de PHP y regresar cuantas veces lo desee usando este método.

¿Donde lo escribimos todo? en cualquier editor vi, kwrite, Quanta,.... el gusto de cada uno es libre como os sentaís mas comodos.

1.2.3.2. Comentarios

PHP soporta el estilo de comentarios de 'C', 'C++' . Por ejemplo:

```
<?php
echo "Prueba"; // Esto es un comentario como c++
/* Esto es un comentario
multilinea que sigue aqui*/
echo "Imprime";
echo "Se imprime esto"; # Esto es comentario
?>
```

Los estilos de comentarios de una linea actualmente sólo comentan hasta el final de la linea o el bloque actual de código PHP, lo primero que ocurra.

Hay que tener cuidado con no anidar comentarios de estilo 'C', algo que puede ocurrir al comentar bloques largos de código.

```
<?php
/*
echo "Texto"; /* Este comentario causa problemas*/
*/
?>
```

1.2.3.3. Tipos de datos

PHP soporta ocho tipos primitivos.

Cuatro tipos escalares:

*boolean
*integer
*float (número de punto-flotante, también conocido como 'double')
*string

Dos tipos compuestos:

*array
*object

Y finalmente dos tipos especiales:
*resource
*NULL

También puede encontrar algunas referencias al tipo "double". Considere al tipo double como el mismo que float, los dos nombres existen solo por razones históricas.

El tipo de una variable usualmente no es declarado por el programador; en cambio, es decidido en tiempo de compilación por PHP dependiendo del contexto en el que es usado la variable.

Enteros

Un integer es un número del conjunto Z = {..., -2, -1, 0, 1, 2, ...}.

Sintaxis

Los enteros pueden ser especificados en notación decimal (base-10), hexadecimal (base-16) u octal (base-8), opcionalmente precedidos por un signo (- o +).

Si usa la notación octal, debe preceder el número con un 0 (cero), para usar la notación hexadecimal, preceda el número con 0x.

Ejemplo. Literales tipo entero

```
<?php
$a = 1234; // numero decimal
$a = -123; // un numero negativo
$a = 0123; // numero octal (equivalente al 83 decimal)
$a = 0x1A; // numero hexadecimal (equivalente al 26 decimal)
?>
```

El tamaño de un entero es dependiente de la plataforma, aunque un valor máximo de aproximadamente dos billones es el valor usual (lo que es un valor de 32 bits con signo). PHP no soporta enteros sin signo.

Aviso Si un dígito inválido es pasado a un entero octal (p.ej. 8 o 9), el resto del número es ignorado.

Ejemplo. Curiosidad de valores octales

```
<?php
var_dump(01090); // 010 octal = 8 decimal
?>
```

Números de punto flotante

Los números de punto flotante (también conocidos como "flotantes", "dobles" o "números reales") pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

El tamaño de un flotante depende de la plataforma, aunque un valor común consiste en un máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales (lo que es un valor de 64 bits en formato IEEE).

Cadenas

Un valor string es una serie de caracteres. En PHP, un caracter es lo mismo que un byte, es decir, hay exactamente 256 tipos de caracteres diferentes. Esto implica también que PHP no tiene soporte nativo de Unicode. Vea utf8_encode() y utf8_decode() para conocer sobre el soporte Unicode.

Nota: El que una cadena se haga muy grande no es un problema. PHP no impone límite práctico alguno sobre el tamaño de las cadenas, así que no hay ninguna razón para preocuparse sobre las cadenas largas.

Sintaxis

Un literal de cadena puede especificarse en tres formas diferentes.

*comillas simples
*comillas dobles
*sintaxis heredoc

Comillas simples

La forma más simple de especificar una cadena sencilla es rodearla de comillas simples (el caracter ').

Para especificar una comilla sencilla literal, necesita escaparla con una barra invertida (\), como en muchos otros lenguajes. Si una barra invertida necesita aparecer antes de una comilla sencilla o al final de

la cadena, necesitará doblarla. Note que si intenta escapar cualquier otro caracter, ¡la barra invertida será impresa también! De modo que, por lo general, no hay necesidad de escapar la barra invertida misma.

```
<?php
echo 'esta es una cadena simple';
echo 'Tambi&eacute;n puede tener saltos de l&iacute;nea embebidos
en las cadenas de esta forma, ya que
es v&aacute;lido';
// Imprime: Arnold dijo una vez: "I'll be back"
echo 'Arnold dijo una vez: "I\'ll be back"';
// Imprime: Ha eliminado C:\*.*?
echo 'Ha eliminado C:\\*.*?';
// Imprime: Ha eliminado C:\*.*?
echo 'Ha eliminado C:\*.*?';
// Imprime: Esto no va a expandirse: \n una nueva linea
echo 'Esto no va a expandirse: \n una nueva linea';
// Imprime: Las variables no se $expanden $tampoco
echo 'Las variables no se $expanden $tampoco';
?>
```

Comillas dobles

Si la cadena se encuentra rodeada de comillas dobles ("), PHP entiende más secuencias de escape para caracteres especiales:

Los siguientes son los "Caracteres escapados y la secuencia de su significado"

\n alimentación de línea (LF o 0x0A (10) en ASCII)

\r retorno de carro (CR o 0x0D (13) en ASCII)

\t tabulación horizontal (HT o 0x09 (9) en ASCII)

\\ barra invertida

\$ signo de dólar

\" comilla-doble

1.2.3.4. Variables

Una variable es un contenedor de información, en el que podemos meter números enteros, números decimales, carácteres... el contenido de las variables se puede leer y se puede cambiar durante la ejecución de una página PHP.

En PHP todas las variables comienzan con el símbolo del dólar $ y no es necesario definir una variable antes de usarla. Tampoco tienen tipos, es decir que una misma variable puede contener un número y luego puede contener carácteres.

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
$a = 1;
$b = 3.34;
$c = "Hola Mundo";
echo $a,"<br>",$b,"<br>",$c;
?>
</body>
</html>
```

En este ejemplo hemos definido tres variables, $a, $b y $c y con la instrucción echo hemos impreso el valor que contenían, insertando un salto de línea entre ellas.

Existen 2 tipos de variables, las variables locales que solo pueden ser usadas dentro de funciones y las variables globales que tienen su ámbito de uso fuera de las funciones, podemos acceder a una variable global desde una función con la instrucción global nombre_variable;

1.2.3.5. Operadores Aritméticos

Los operadores de PHP son muy parecidos a los de C y JavaScript, si usted conoce estos lenguajes le resultaran familiares y fáciles de reconocer.

Estos son los operadores que se pueden aplicar a las variables y constantes numéricas.

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números
-	Resta	7 - 9	Resta dos números
*	Multiplicación	6 * 3	Multiplica dos números
/	División	4 / 8	Divide dos números
%	Módulo	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Suma 1	$a++	Suma 1 al contenido de una variable.
–	Resta 1	$a–	Resta 1 al contenido de una variable.

Ejemplo de php

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
$a = 8;
$b = 3;
echo $a + $b,"<br>";
echo $a - $b,"<br>";
echo $a * $b,"<br>";
echo $a / $b,"<br>";
$a++;
echo $a,"<br>";
$b–;
echo $b,"<br>";
?>
</body>
</html>
```

1.2.3.6. Operadores de Comparación

Los operadores de comparación son usados para comparar valores y así poder tomar decisiones.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
==	Igual	$a == $b	$a es igual $b
!=	Distinto	$a != $b	$a es distinto $b
<	Menor que	$a <$b	$a es menor que $b
>	Mayor que	$a >$b	$a es mayor que $b
<=	Menor o igual	$a <= $b	$a es menor o igual que $b
>=	Mayor o igual	$a >= $b	$a es mayor o igual que $b

Ejemplo de php

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
```

```
<body>
<?php
$a = 8;
$b = 3;
$c = 3;
echo $a == $b,"<br>";
echo $a != $b,"<br>";
echo $a <$b,"<br>";
echo $a >$b,"<br>";
echo $a >= $c,"<br>";
echo $b <= $c,"<br>";
?>
</body>
</html>
```

1.2.3.7. Operadores Lógicos

Los operadores lógicos son usados para evaluar varias comparaciones, combinando los posibles valores de estas.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
&&	Y	(7>2) && (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
and	Y	(7>2) and (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
||	O	(7>2) || (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
or	O	(7>2) or (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
!	No	! (7>2)	Niega el valor de la expresión.

Ejemplo de php

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
$a = 8;
$b = 3;
$c = 3;
echo ($a == $b) && ($c >$b),"<br>";
echo ($a == $b) || ($b == $c),"<br>";
echo !($b <= $c),"<br>";
?>
</body>
</html>
```

1.2.3.8. Sentencias Condicionales

Las sentencias condicionales nos permiten ejecutar o no unas ciertas instrucciones dependiendo del resultado de evaluar una condición. Las más frecuentes son la instrucción if y la instrucción switch.

Sentencia if ... else

```
<?php
if (condición)
{
Sentencias a ejecutar cuando la
condición es cierta.
}
else
```

```
{
Sentecias a ejecutar cuando la
condición es falsa.
}
?>
```

La sentencia if ejecuta una serie de instrucciones u otras dependiendo de la condición que le pongamos. Probablemente sea la instrucción más importante en cualquier lenguaje de programación.

Ejemplo

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
$a = 8;
$b = 3;
if ($a <$b)
{
echo "a es menor que b";
}
else
{
echo "a no es menor que b";
}
?>
</body>
</html>
```

En este ejemplo la condición no es verdadera por lo que se ejecuta la parte de código correspondiente al else.

Sentencia switch ... case

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
$posicion = "arriba";
switch($posicion) {
case "arriba": // Bloque 1
echo "La variable contiene";
echo " el valor arriba";
break;
case "abajo": // Bloque 2
echo "La variable contiene";
echo " el valor abajo";
break;
default: // Bloque 3
echo "La variable contiene otro valor";
echo " distinto de arriba y abajo";
}
?>
</body>
</html>
```

Con la sentencia switch podemos ejecutar unas u otras instrucciones dependiendo del valor de una variable, en el ejemplo anterior, dependiendo del valor de la variable $posicion se ejecuta el bloque 1 cuando el valor es "arriba", el bloque 2 cuando el valor es "abajo" y el bloque 3 si no es ninguno de los valores anteriores.

1.2.3.9. Sentencias Repetitivas o Bucles

Los bucles nos permiten iterar conjuntos de instrucciones, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición.

Sentencia while

```
<?php
while (condición)
{
intrucciones a ejecutar.
}
?>
```

Mientras la condición sea cierta se reiterará la ejecución de las instrucciones que están dentro del while.

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
Inicio<BR>
<?php
$i=0;
while ($i<10)
{
echo "El valor de i es ", $i,"<br>";
$i++;
}
?>
Final<BR>
</body>
</html>
```

En el siguiente ejemplo, el valor de $i al comienzo es 0, durante la ejecución del bucle, se va sumando 1 al valor de $i de manera que cuando $i vale 10 ya no se cumple la condición y se termina la ejecución del bucle.

Sentencia for

```
<?php
for (inicial ; condición ; ejecutar en iteración)
{
intrucciones a ejecutar.
}
?>
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
Inicio<BR>
<?php
for($i=0 ; $i<10 ; $i++)
{
echo "El valor de i es ", $i,"<br>";
```

```
}
?>
Final<BR>
</body>
</html>
```

La instrucción for es la instrucción de bucles más completa. En una sola instrucción nos permite controlar todo el funcionamiento del bucle.

El primer parámetro del for, es ejecutado la primera vez y sirve para inicializar la variable del bucle, el segundo parámetro indica la condición que se debe cumplir para que el bucle siga ejecutándose y el tercer parámetro es una instrucción que se ejecuta al final de cada iteración y sirve para modificar el valor de la variable de iteración.

foreach

PHP 4 (PHP3 no) incluye una construcción foreach, tal como perl y algunos otros lenguajes. Esto simplemente da un modo fácil de iterar sobre matrices. foreach funciona solamente con matrices y devolverá un error si se intenta utilizar con otro tipo de datos ó variables no inicializadas. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

foreach(expresion_array as $value) sentencia

foreach(expresion_array as $key =>$value) sentencia

La primera forma recorre el array dado por expresion_array. En cada iteración, el valor del elemento actual se asigna a $value y el puntero interno del array se avanza en una unidad (así en el siguiente paso, se estará mirando el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable $key en cada iteración.

Esta función tendrá mucha utlidad cuando trabajemos con las tablas de mysql.

1.2.3.10. Salida

Hasta ahora hemos usado la instrucción echo para realizar salida a pantalla, esta instrucción es bastante limitada ya que no nos permite formatear la salida. En esta página veremos la instrucción printf que nos da mucha más potencia.

Sentencia printf

```
<?php
printf(cadena formato, variable1, variable2...);
?>
```

La cadena de formateo indica cómo se han de representar las valores que posteriormente le indicaremos. La principal ventaja es que además de poder formatear los valores de salida, nos permite intercalar texto entre ellos.

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
printf("El numero dos con diferentes formatos: %d %f %.2f",2,2,2);
?>
</body>
</html>
```

1.2.3.11. Manejo de cadenas

Dado el uso del lenguaje PHP el tratamiento de cadenas es muy importante, existen bastantes funciones para el manejo de cadenas, a continuación explicaremos las más usadas.

* strlen(cadena). Nos devuelve el número de carácteres de una cadena.

* split(separador,cadena). Divide una cadena en varias usando un carácter separador.

* sprintf(cadena de formato, var1, var2...). Formatea una cadena de texto al igual que printf pero el resultado es devuelto como una cadena.

* substr(cadena, inicio, longitud). Devuelve una subcadena de otra, empezando por inicio y de longitud longitud.

* chop(cadena). Elimina los saltos de línea y los espacios finales de una cadena.

* strpos(cadena1, cadena2). Busca la cadena2 dentro de cadena1 indicándonos la posición en la que se encuentra.

* str_replace(cadena1, cadena2, texto). Reemplaza la cadena1 por la cadena2 en el texto.

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
echo strlen("12345"),"<br>";
$palabras=split(" ","Esto es una prueba");
for($i=0;$palabras[$i];$i++)
echo $palabras[$i],"<br>";
$resultado=sprintf("8x5 = %d <br>",8*5);
echo $resultado,"<br>";
echo substr("Devuelve una subcadena de otra",9,3),"<br><br>";
if (chop("Cadena \n\n ") == "Cadena")
echo "Iguales<br><br>";
echo strpos("Busca la palabra dentro de la frase", "palabra"),"<br><br>";
echo str_replace("verde","rojo","Un pez de color verde, como verde es la hierba."),"<br>";
?>
</body>
</html>
```

1.2.3.12. Funciones

El uso de funciones nos da la capacidad de agrupar varias instrucciones bajo un solo nombre y poder llamarlas a estas varias veces desde diferentes sitios, ahorrándonos la necesidad de escribirlas de nuevo.

```
<?php
function Nombre(parametro1, parametro2...)
{
instrucción1;
instrucción2;
instrucción3;
instrucción4;
return valor_de_retorno;
}
?>
```

Opcionalmente podemos pasarle parámetros a las funciones que se trataran como variable locales y así mismo podemos devolver un resultado con la instrucción return valor; Esto produce la terminación de la función retornando un valor.

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
function media_aritmetica($a, $b)
{
```

```
$media=($a+$b)/2;
return $media;
}
echo media_aritmetica(4,6),"<br>";
echo media_aritmetica(3242,524543),"<br>";
?>
</body>
</html>
```

1.2.3.13. Envío y recepción de datos

El lenguaje PHP nos proporciona una manera sencilla de manejar formularios, permitiéndonos de esta manera procesar la información que el usuario ha introducido.

Al diseñar un formulario debemos indicar la página PHP que procesará el formulario, así como en método por el que se le pasará la información a la página.

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
Introduzca su nombre:
<FORM ACTION="procesa.phtml" METHOD="GET">
<INPUT TYPE="text" NAME="nombre"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
```

Al pulsar el botón Enviar el contenido de cuadro de texto es enviado a la página que indicamos en el atributo ACTION de la etiqueta FORM.

En versiones anteriores a 4.2.0 PHP creaba una variable por cada elemento del FORM, esta variable creada tenía el mismo nombre que el cuadro de texto de la página anterior y el valor que habíamos introducido. Pero por razones de seguridad a partir de entonces para acceder a las variables del formulario hay que usar el array de parámetros $_POST[] o $_GET[] dependiendo del método usado para enviar los parámetros.

En este ejemplo se ha creado una entrada en el array $_GET[] con el índice 'nombre' y con el valor que haya introducido el "navegante".

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
El nombre que ha introducido es: <?php echo $_GET['nombre'] ?>
<br>
</body>
</html>
```

Method GET y POST

En la página anterior hemos comentado que los datos de un formulario se envía mediante el método indicado en el atributo METHOD de la etiqueta FORM, los dos métodos posibles son GET y POST.

La diferencia entre estos dos métodos radica en la forma de enviar los datos a la página, mientras que el método GET envía los datos usando la URL, el método POST los envía por la entrada estándar STDIO.

```
<html>
<head>
```

<title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
<FORM ACTION="procesa2.php" METHOD="GET">
Introduzca su nombre:<INPUT TYPE="text" NAME="nombre">
Introduzca sus apellidos:<INPUT TYPE="text" NAME="apellidos">
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
Listado de ejemplo usando el metodo POST
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
<FORM ACTION="procesa2.php" METHOD="POST">
Introduzca su nombre:<INPUT TYPE="text" NAME="nombre">
Introduzca sus apellidos:<INPUT TYPE="text" NAME="apellidos">
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
A continuación listamos el fichero procesa2.php
procesa2.php
<!– Manual de PHP de WebEstilo.com –>
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
El nombre que ha introducido por GET es: <?php echo $_GET['nombre']," ",$_GET['apellidos'] ?>
El nombre que ha introducido por POST es: <?php echo $_POST['nombre']," ",$_POST['apellidos'] ?>

</body>
</html>
El resultado final es el mismo, solo que con el método GET podemos ver los parámetros pasados ya que están codificados en la URL.

Capítulo 2

Bases de datos: MySQL.

2.1. Qué es MySQL.

MySQL es un Sistema Gestor de Bases de Datos Relacionales (SGBDR, o RDBMS en inglés), que posibilita gestionar grandes volúmenes de información (datos) estructurados siguiendo una determinada jerarquía, basada en el uso de tablas. Un SGBDR permite gestionar más de una Base de Datos, donde cada una de ellas contiene un conjunto de datos organizados a partir de tablas, de manera que cada BD dispone de al menos una tabla. Cada tabla posee unos atributos (también denominados campos o propiedades) que definen el conjunto de valores correspondiente a un elemento de información almacenado en la tabla. Cada elemento de información recibe el nombre de registro, de manera que un registro no es más que el conjunto de valores que caracteriza a un elemento de información de la tabla. Por ejemplo, una tabla que almacene los nombres y los logins (identificadores de acceso) que poseen los usuarios de un sistema, puede estructurarse de la siguiente forma:

- Nombre de la tabla: Usuarios.
- Nombre de los atributos que componen un registro (es decir, un usuario): Nombre del usuario, apellidos del usuario y login del usuario.

Evidentemente el tipo de información que se almacena en cada atributo puede ser diferente, por ejemplo, si se determina que el login es un número, tendríamos que el atributo **login del usuario** es de tipo *entero*, mientras que **nombre del usuario** y **apellidos del usuario** son de tipo *cadena de caracteres*.

Una base de datos puede componerse de varias tablas, y la información contenida en ella estar relacionada de alguna manera. Habrá atributos en cada una de estas tablas relacionadas que posibilitarán *saltar* desde un registro de una de ellas hasta uno o varios registros de otra. Más adelante profundizaremos más sobre esta cuestión. Ahora lo importante es entender que el sistema basado en tablas no es el único empleado por los sistemas de bases de datos, pero es el que caracteriza a uno de ellos, y que en la actualidad es de los más empleados: el sistema de bases de datos relacionales.

MySQL es un SGBDR, que sigue el modelo Cliente-Servidor, es decir, incluye un servidor que posibilita, mediante programas cliente, acceder a los datos contenidos en el conjunto de bases de datos relacionales que soporta o gestiona. MySQL, como su propio nombre indica, posibilita trabajar con las BDR mediante el lenguaje SQL. Aunque los comienzos de MySQL se remontan a 1994, es en 1996 cuando MySQL 3.11.1 es depositada en Internet mediante distribuciones binarias para Linux y Solaris.

Es preciso tener en cuenta que MySQL no es un proyecto *Open Source*, ya que bajo ciertas condiciones (por ejemplo si se venden servicios que requieren MySQL) es preciso disponer de licencia. Sin embargo, los términos de licencia no son demasiado restrictivos y por tanto es muy usado por la comunidad Open Source. En el caso que nos ocupa, la comunidad educativa no está obligada a pagar dinero alguno por el uso de MySQL, y esta es la razón de que en todas las distribuciones Linux actuales se incorpore de manera estándar este SGBDR.

MySQL además es portable ya que hay versiones para diferentes sistemas operativos (Windows incluido).

¿Qué hace de MySQL que sea tan usada en detrimento de otros sistemas a priori más potentes?

Esta pregunta tiene trampa, ya que damos por hecho que se escoge a sabiendas que no es tan potente como otros, a pesar de que este matiz está en función de qué entendemos por potente.

Para empezar, MySQL (en general) es gratuito. Oracle, por ejemplo, no lo es. Y ser gratuito ya es una primera e importante ventaja.

En segundo lugar, MySQL es **muy rápida**, posiblemente sea la más rápida que se pueda encontrar,lo que la hace especialmente idónea para aquellos casos en los que se precisa manejar un volumen muy alto de datos en procesos de consulta (algo habitual por ejemplo en las aplicaciones web de comercio electrónico). Entidades y empresas de gran prestigio emplean esta base de datos (The Associated Press, Yahoo, NASA, Sabre Holdings y Suzuki entre otras).

En tercer lugar está su facilidad de uso, ya que es mucho menos compleja de configurar y administrar que otras. Una base de datos MySQL no es más que un conjunto de archivos ubicados en un determinado directorio, así de simple.

Finalmente, el soporte del lenguaje SQL, la capacidad de dar acceso a múltiples cientes concurrentemente sobre el servidor, su capacidad para trabajar en red (bien intranet o bien extranet, como internet) incorporando control de acceso de usuarios y su portabilidad, la hacen casi imprescindible en determinados contextos de uso[1].

2.2. Cómo obtener MySQL.

Como con otras aplicaciones tratadas en este curso, hay dos formas básicas de obtener y/o instalar MySQL. Una es descargarla desde el sitio oficial de internet desde la que se ofrece (o alguno de sus *mirrors*), la otra es instalarla directamente desde el gestor de paquetes de la propia distribución si ésta la incorpora. Prácticamente todas las distribuciones Linux incorporan MySQL como paquete base. Sólo si esto no fuera así, o bien deseáramos instalar una versión de MySQL posterior a la que ofrece la distribución, optaríamos por descargarla de su website[2].

Si optamos por descargarla de su website, hemos de asegurarnos que bajamos todos los paquetes/archivos que necesitamos y que además no se trata de una versión *alpha*. Si optamos por descargar en formato *tar.gz* bastará con bajarse el archivo correspondiente a la distribución *Standard*, que son unos 30 Mb, y para plataforma x86. Si preferimos el formato de paquete *rpm*, deberemos bajar al menos los paquetes correspondientes al Servidor y a los programas clientes (unos 20 Mb aproximadamente). Actualmente la versión estable disponible es la 4.1.10[3].

En Mandrake 9.1 los paquetes necesarios son:

- MySQL-4.0.11a-5mdk (Servidor MySQL)
- MySQL-client-4.0.11a-5mdk (Cliente MySQL)
- MySQL-common-4.0.11a-5mdk (Archivos comunes necesarios para el servidor)
- php-mysql-4.3.0-2mdk (Conjunto de librerías dinámicas que posibilita añadir soporte MySQL a los programas escritos en PHP)

En Mandrake 10.0 los paquetes necesarios son:

- MySQL-4.0.18-1mdk (Servidor MySQL)
- MySQL-client-4.0.18-1mdk (Cliente MySQL)
- MySQL-common-4.0.18-1mdk (Archivos comunes necesarios para el servidor)

[1] Structured Query Language (lenguaje estructurado de consulta).

[2] http://www.mysql.com es su web oficial, pero para descargar MySQL y sus correspondientes herramientas accederemos a http://dev.mysql.com/).

[3] Marzo de 2005.

- php-mysql-4.3.4-1mdk (Conjunto de librerías dinámicas que posibilita añadir soporte MySQL a los programas escritos en PHP)

2.3. Instalar MySQL[4].

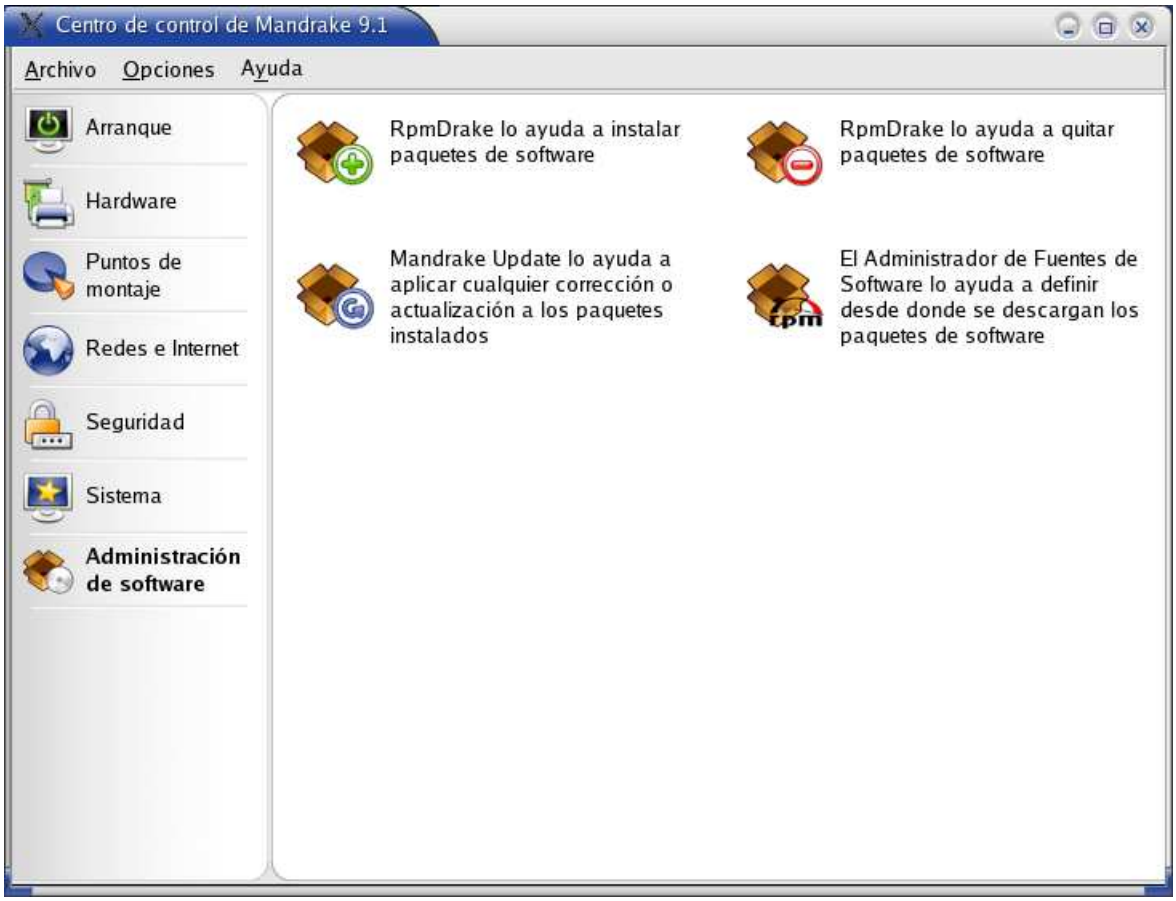
La instalación de MySQL puede realizarse desde el gestor gráfico de paquetes suministrado por la distribución, o bien mediante el comando **rpm**. Para hacerlo todo un poco más sencillo, y tratándose de la tendencia tomada por las actuales distribuciones Linux de llevar la mayoría de las tareas de administración del sistema a través del entorno gráfico, lo haremos mediante el gestor gráfico.

Antes será necesario asegurarse de que no tenemos ya instalado el servidor MySQL, ya que en ese caso no se hace necesario volver a instalarlo. Para ello bastará con abrir una consola Linux y ejecutar el comando **mysql**. Si aparece un mensaje indicando que el comando es desconocido entonces procederemos a su instalación. Otra forma es dirigirse a la herramienta gráfica de instalación/desinstalación de paquetes de la distribución y comprobar si ya está instalado o no el servidor. De todas formas se hace necesario tener en cuenta que no basta con instalar el servidor MySQL, si no que se precisan los paquetes correspondientes a MySQL Control Center y al módulo de integración de MySQL en PHP. Por tanto, no estaría de más leer los apartados 2.3.1 y 2.3.2 siguientes, y comprobar si todo está ya instalado o no, en cuyo caso se deberá proceder a instalar lo que falte.

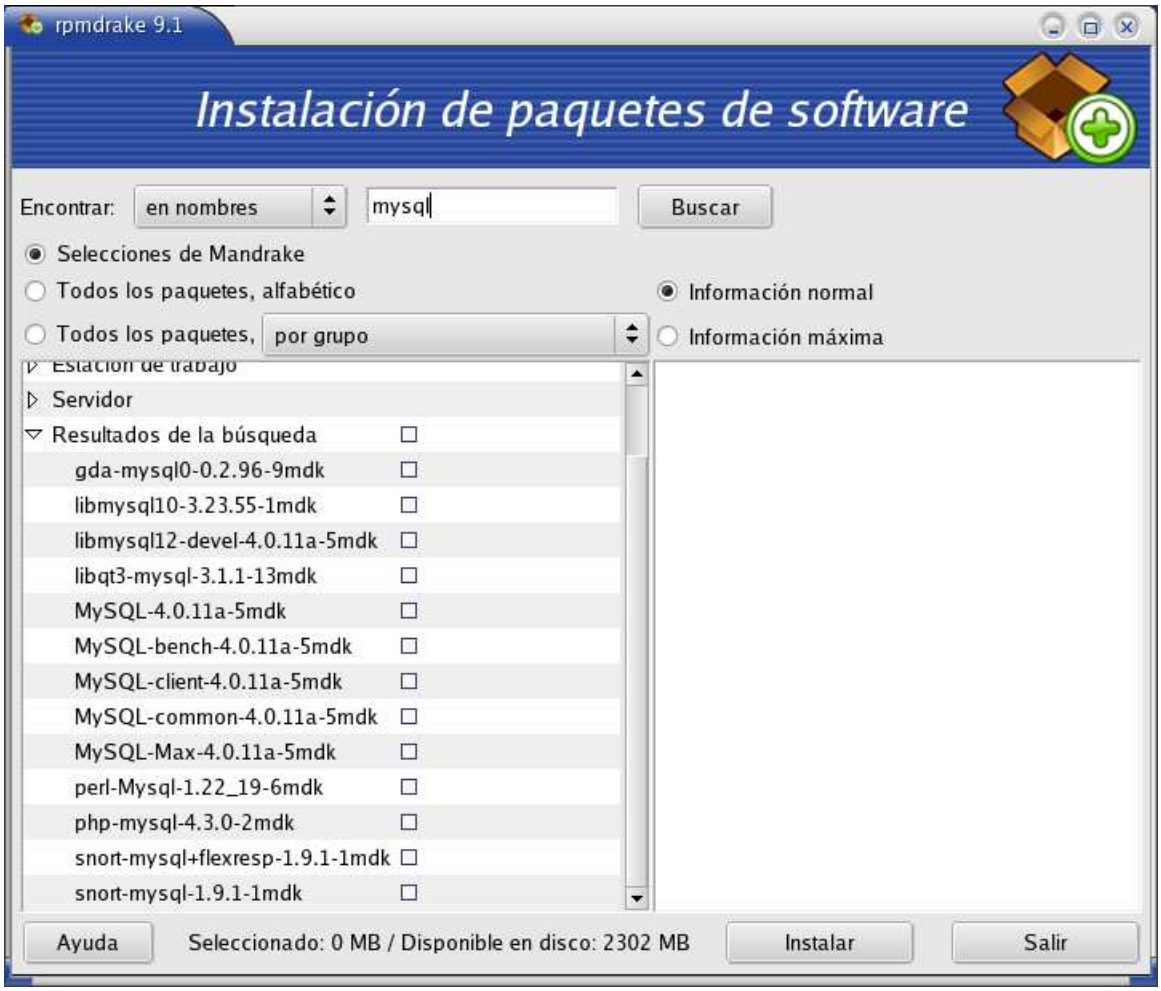
2.3.1. En Mandrake 9.1.

Accedemos al Centro de Control de Mandrake (CCM) y seleccionamos la opción **Administración de Software**. Y hacemos clic sobre el icono de instalación.

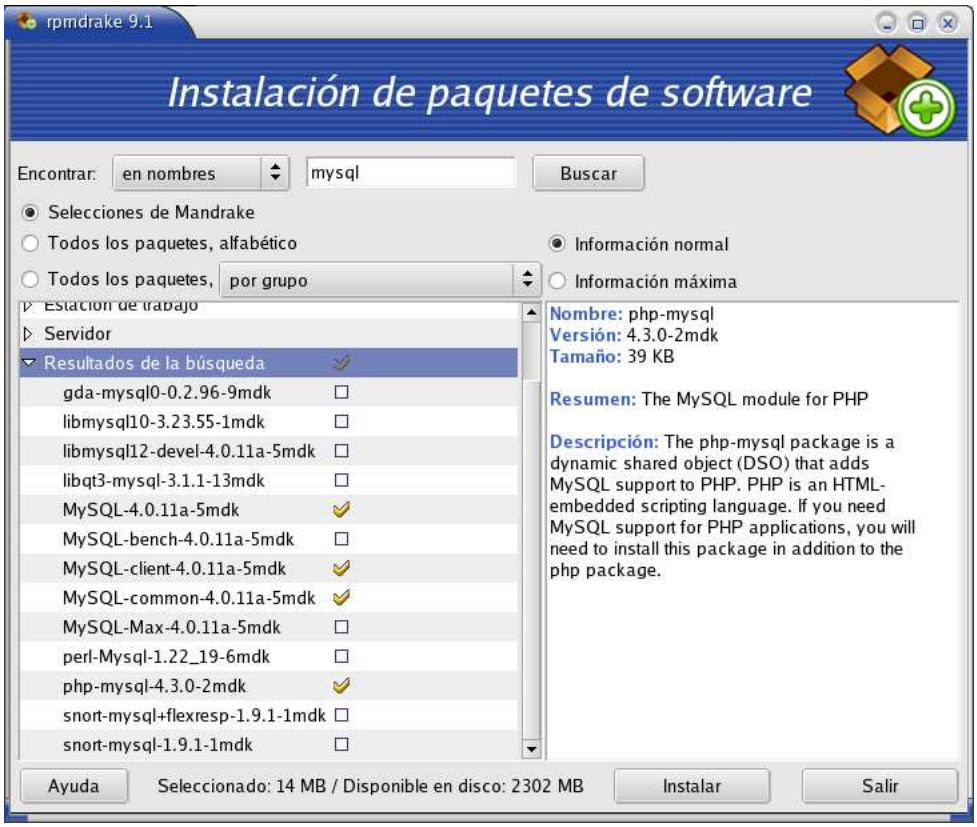
[4] Se asume que se instalará como usuario *root*.



Nos aparece una ventana en la que introduciremos la palabra **mysql** en el cuadro de texto dedicado a la búsqueda de paquetes. Finalizada la búsqueda el resultado será similar a éste:

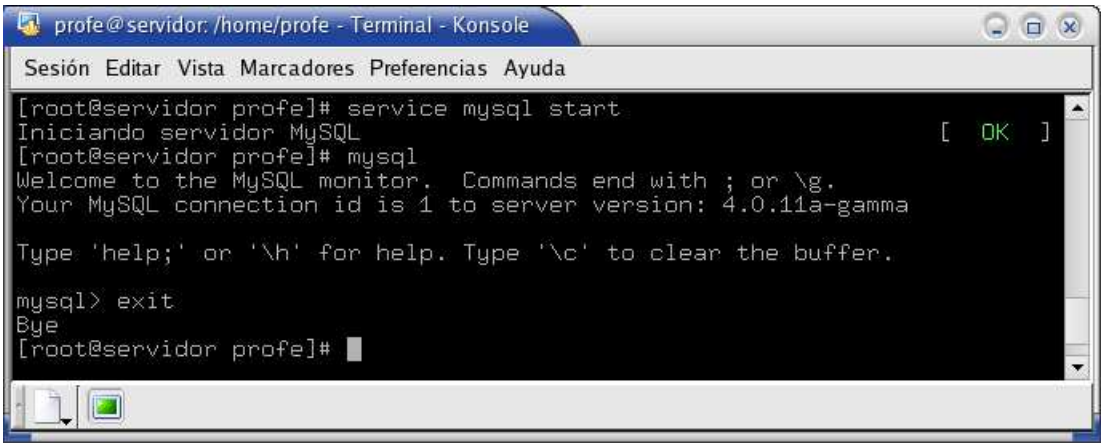


Y marcamos los paquetes necesarios:



Pulsamos **Instalar**.

Finalizada la ejecución procedemos a comprobar si ya está corriendo MySQL o no (lo normal es que aún no se haya lanzado). Para ello basta con ejecutar desde consola el comando **mysql**[5]. Si aparece algún mensaje de error, o la respuesta no es similar a la siguiente imagen, debermos lanzarlo ejecutando el comando **service mysql start**.



Para salir se introduce **exit**, tal como se observa en la imagen.

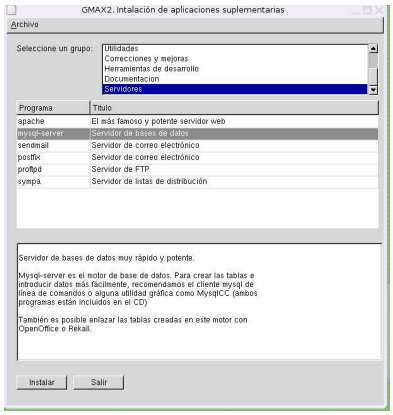
Para detener el servidor bastará con ejecutar **service mysql stop**.

La instalación en Mandrake 10 es similar.

[5]Este comando es precisamente una aplicación cliente en modo texto que posibilita acceder a las bases de datos del servidor. Se analizará más adelante.

2.3.2. En Guadalinex Edición Ciudadano 2004.

Hacemos clic en **Aplicaciones->Configuración->Gmax (instalador de suplementos)**. Nos pedirá la contraseña del root, se la damos. Insertamos el CD de Suplementos. Nos aparecerá una ventana en la que podremos seleccionar un grupo de paquetes. Seleccionamos Servidores. Y nuevamente, en un cuadro inferior, seleccionamos **mysql-server**.



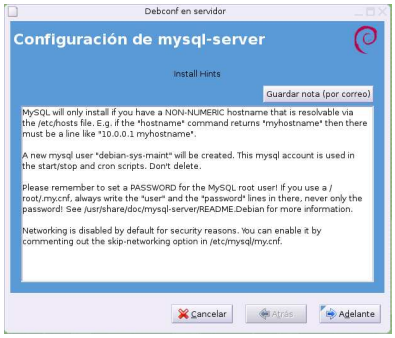
Pulsamos **instalar**.

Si aparece un asistente de configuración de Exim, pulse el botón **siguiente** en cada una de las fases que éste muestra, aceptando los valores por defecto ofrecidos en cada una de ellas.

Aparece el asistente de configuración del servidor MySQL. En la primera ventana dejamos marcada la opción de arrancar el servidor cada vez que arranque la máquina.

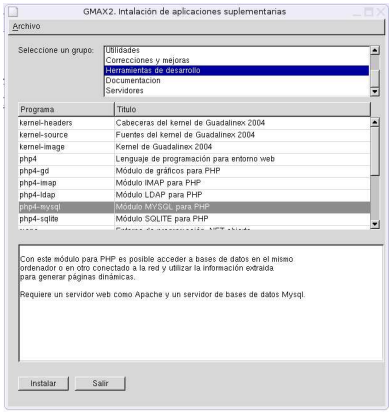
Se pulsa **Adelante**. Aparece la siguiente ventana de configuración, en la que se nos informa de diversas cuestiones:

- MySQL requiere que la máquina posea una IP y un nombre de host asociado. Esto debe estar especificado en /etc/hosts.
- Se creará un usuario denominado **debian-sys-main**, que se encargará de realizar procesos de arranque y parada del servidor, por lo que no debe borrarse del sistema.
- Nos recuerda que debemos establecer un password para el usuario **root** de MySQL.
- Para trabajar a través de la red con MySQL será necesario poner en comentarios la línea del archivo /etc/mysql/my.cnf que establece el valor de la optión **skip-networking**.



Pulsamos **Adelante**. El servidor se ha instalado. Si abrimos una consola y ejecutamos en comando **mysql**, aparecerá el intérprete de comandos mysql en espera de que introduzcamos una orden. Si tecleamos *exit* salimos.

Para poder emplear la herramienta de configuración y administración de bases de datos MySQL phpMyAdmin, será necesario disponer del lenguaje de programación PHP y de los módulos que se integran en Apache y en MySQL. Se supone ya instalado PHP4, Apache y el módulo de integración de PHP sobre Apache. Queda por tanto instalar el módulo **php4-mysql** que posibilitará emplear funciones de acceso a MySQL mediante el lenguaje PHP. Para ello procedemos de forma similar a como lo hemos hecho con MySQL, pero esta vez seleccionando el grupo **Herramientas de desarrollo** y el paquete **php4-mysql**.



2.4. Herramientas de configuración y administración de MySQL.

Tras instalar MySQL disponemos de:

- El servidor MySQL. Se trata del *corazón* del sistema gestor. Escucha por el puerto 3306 por defecto. Es recomendable no cambiarlo, ya que se trata de un puerto *asumido* por Linux como asociado a MySQL.
- Un conjunto de aplicaciones/comandos que permiten comunicarse con el servidor y ejecutar procesos de administración. Los más importantes son: *mysql*[6], *mysqladmin*[7], *isamchk*[8], *myisamchk* y *mysqldump*[9].
- Un directorio desde el que *cuelga* toda la información gestionada por el servidor. Las bases de datos se estructuran a base de directorios, conteniendo cada uno de ellos todo lo relacionado con cada una de las bases de datos que gestiona el servidor.

[6] Cliente MySQL para interactuar con las bases de datos del servidor.
[7] Herramienta de administración que posibilita arrancar y parar el servidor, crear y borrar bases de datos, gestionar usuarios, etc.
[8] Junto con myisamchk permite realizar optimizaciones de las tablas, así como restaurar tablas dañadas.
[9] Realiza copias de seguridad de las bases de datos.

- La potencia del lenguaje SQL, mediante el cual podrán realizarse todas las operaciones de administración y gestión de las bases de datos del servidor.

Las operaciones básicas de administración que son necesarias conocer se resumen en:

- **Arranque y parada del servidor**. Es lo primero que se debe aprender a hacer.
- **Analizar y ajustar el funcionamiento del servidor**. Se hace necesario saber si está funcionando correctamente o no. Si se lanza sin problemas o genera fallos de conexión desde los clientes.
- **Administración de usuarios**. El servidor MySQL posee su propio sistema de usuarios, diferente al empleado en Linux. Es por ello que se hace imprescindible controlar qué usuarios se emplean para administrar propiamente el servidor y qué usuarios y con qué restricciones se emplean para administrar, gestionar y consultar las bases de datos y tablas. MySQL posee un usuario administrador por defecto que es **root**, aunque no tiene nada que ver con el usuario root de Linux. De hecho, ambos pueden no disponer de la misma contraseña. Al instalarse MySQL su usuario root no tiene asignada contraseña. Será lo primero que deberá realizar un administrador del servidor, cambiarla.
- **Gestión de los archivos de registro**. Estos archivos almacenan el seguimiento de las operaciones realizadas sobre el servidor. Se hace necesario llevar un buen mantenimiento de los mismos y evitar así que *engorden* excesivamente.

Estructura de archivos del servidor

Hemos realizado la instalación, pero antes de empezar a trabajar con el servidor se hace necesario saber dónde están ubicados sus archivos y dónde se ubicarán las bases de datos que vayamos creando. Un primer paso puede ser acceder a las herramientas de gestión de paquetes que suministran las distribuciones Linux y comprobar qué estructura de directorio mantienen. Por ejemplo, en el caso de Mandrake, y para el paquete MySQL-4.0.11a-5mdk (el paquete que contiene al servidor propiamente dicho) de Mandrake 9.1, los archivos proporcionados (junto con su ubicación) son:

- /etc/rc.d/init.d/mysql[10]
- /usr/lib/mysql/mysqld.sym
- /usr/sbin/mysqld[11]

Es necesario advertir que diferentes distribuciones pueden ubicar estos archivos en directorios diferentes, así como añadir otros nuevos.

Respecto al *directorio de datos* (hay uno, desde el que cuelgan todas las bases de datos, ya que una base de datos es un conjunto de archivos reunidos bajo un mismo directorio), su ubicación está un poco en función de mediante qué vía se ha instalado MySQL. Si se ha hecho desde un paquete rpm lo más probable es que esté en /var/lib/mysql[12], si se ha realizado desde una distribución fuente[13], podemos tener a /usr/local/var como ubicación, y si hemos partido de un binario, quizás estén en /usr/local/mysql/data.

Cada base de datos se corresponde con un directorio que cuelga bajo el denominado *directorio de datos*. El nombre del directorio coincide con el de la base de datos. De hecho, cuando se crea la base de datos proporcionándole un nombre, el servidor crea dicho directorio[14].

Las tablas serán archivos ubicados dentro del directorio de la base de datos.

Además, en el directorio de datos se incluyen archivos que almacenan el estado de la base de datos, y los errores que se han producido sobre ella.

[10] En Guadalinex este archivo está ubicado en /etc/init.d.

[11] En Guadalinex está ubicado en este mismo directorio.

[12] Es precisamente éste el empleado por Mandrake 9.1. Obsérvese además que es en /var donde localizamos directorios empleados por servidores para depositar los datos que manipulan. Por ejemplo, Apache en /var/www, y ProFTPD en /var/ftp.

[13] Es decir, se han bajado los fuentes y se ha compilado todo el servidor.

[14] Hay ciertas restricciones cuando se nombran bases de datos y tablas en MySQL. Lo recomendable es usar caracteres alfabéticos estándar y prescindir de espacios, signos y caracteres específicos de un idioma. También se debe tener en cuenta que si bien Windows no va a distinguir entre mayúsculas y minúsculas al nombrar a las bases de datos y tablas en MySQL, Linux sí lo hace, por lo que se recomienda que siempre se empleen o bien minúsculas o bien mayúsculas.

El *demonio* mysqld

Se trata del núcleo del servidor, y es el encargado de canalizar todos los procesos sobre las diferentes bases de datos que gestiona bajo su correspondiente directorio de datos. Todas las operaciones solicitadas desde los clientes pasan por él. Digamos que es el intermediario entre estos clientes y los datos de las bases de datos. Habitualmente esto se hace asociando un puerto TCP/IP al servidor, aunque también, en Linux, puede hacerse mediante un archivo *socket*[15].

Normalmente no se ejecuta directamente, sino que se recurre al comando *service mysql start* o bien a ejecutar el script *mysqld_safe*, ubicado en /usr/bin[16]. Este script acepta los parámetros más importantes asociados al arranque del servidor, como por ejemplo *–datadir* y *–log*. Este script tiene además la ventaja de que reinicia el servidor si éste ha generado algún tipo de error.

Archivos de estado

MySQL proporciona varios archivos con información relativa a su estado. Estos son:

Nombre del archivo	Contenido
<hostname>.pid	ID del proceso asociado al demonio servidor (mysqld)
<hostname>.err	Informe de errores surgidos durante su ejecución. Registro de arranque y parada del servidor
<hostname>.log	Registro de conexiones y desconexiones al servidor. Registro de consultas. Opcional. Se requiere el parámetro *–log* al lanzar el servidor.
<hostname>.<nnn>	Sentencias SQL que se han ejecutado sobre el servidor y que modifican la estructura de tablas y contenido de las mismas. Opcional. Se requiere el parámetro *–log-update* al lanzar el servidor.

Como se observa, el nombre por defecto de estos archivos es el nombre del servidor. Están ubicados dentro del directorio de datos. Cada vez que se arranca el servidor, éste escribe el IP del proceso en el archivo <*hostname*>*.pid.* Mediante este archivo sabremos cómo, por ejemplo, eliminar el proceso servidor si éste se bloqueara.

El archivo de registro de errores, <*hostname*>*.log* se crea por medio del script *mysqld_safe* y por tanto es exclusivo de él. Informa (registra) quiénes están conectados, desde dónde y qué consultas realizan. El archivo de registro de actualizaciones, <*hostname*>.<*nnn*> registra las consultas a la base de datos que suponen modificación de su contenido.

Estos archivos pueden estar localizados en directorios diferentes al directorio de datos, salvo el registro de error (<hostname>.err). Para ello se emplean los siguientes parámetros[17]:

Archivo	Parámetro
<hostname>.pid	–pid-file=<ubicación/archivo>
<hostname>.log	–log=<ubicación/archivo>
<hostname>.<nnn>	–log-update=<ubicación/archivo>

En Guadalinex tenemos:

Archivo	Ubicación
mysqld.pid	/var/run/mysqld
mysql.log	/var/log/mysql
mysql.err	/var/log

[15] Este aspecto no lo trataremos en este manual.

[16] Es en este directorio donde están ubicados la mayoría de los archivos ejecutables de MySQL. Una excepción es el demonio *mysql*, que está en /usr/sbin, considerado como un ejecutable de *sistema*.

[17] Si en lugar de emplear rutas absolutas (empezando por /) se emplean rutas relativas, los archivos se ubican bajo el directorio de datos.

Copia y traslado del directorio de datos

Muchas veces nos veremos obligados a modificar la ubicación del directorio de datos de un servidor MySQL. Por ejemplo si deseamos llevarnos las bases de datos desde una máquina a otra. O desde la partición de un sistema Linux a otro. La manera más sencilla consiste en parar el servidor, realizar la copia, y posteriormente arrancar el servidor proporcionando el nuevo directorio de ubicación de datos a través del parámetro –datadir. Si sólo se desea copiar una base de datos, se deberá copiar esa base de datos más la base de datos mysql. Y deberán quedar bajo el directorio de datos especificado en –datadir.

2.4.1. MySQLAdmin

Se instala automáticamente al instalar el paquete MySQL-client y permite realizar labores de administración sobre el servidor MySQL. El comando es **mysqladmin**.

Operaciones básicas de administración

Arranque

mysqld_safe & (se acompaña de *&* con el propósito de que se lance en segundo plano o *background*, de manera que pueda cerrarse la terminal desde la que se ha ejecutado el comando). Debe ejecutarse como usuario root de Linux. Debe tenerse mucho cuidado con la cuenta de usuario con la que se arranca el servidor. El servidor puede arrancarse tanto manualmente como automáticamente[18]. Si se arranca manualmente, éste lo hace como el usuario con el que se está lanzando. Es decir, si se está conectado como root de Linux, al arrancar el servidor éste se estará ejecutando como un proceso asociado al usuario root. Esto puede suponer un problema si alguien consigue que el servidor actúe de manera *insegura*, ya que estaría teniendo acceso a todo el sistema. Una opción preferible es hacerlo como un usuario distinto. Normalmente, las diferentes instalaciones disponibles de MySQL crean un usuario y grupo específicos para poder hacer esto. Tanto en Mandrake como en Guadalinex es **mysql** (tanto para la cuenta de usuario como para el grupo). Se puede entrar en esa cuenta, arrancar el servidor, y salir de ella. Ahora la ejecución del servidor está a salvo de la cuenta root.

También puede arrancarse el servidor directamente mediante *mysqld*, pero no es aconsejable. La ejecución de *man mysqld* muestra las posibilidades que ofrece la ejecución directa del demonio.

Arranque usando un directorio de datos diferente al empleado por defecto

mysqld_safe –datadir=/home/datos (En este caso se determina que el directorio de datos está en /home/datos. Al menos la base de datos *mysql* debe estar en este directorio, ya que se trata de la base de datos que guarda información asociada al servidor, como son sus usuarios). Debemos tener presente que al instalarse MySQL se crea un usuario especial para el servidor, como se comentó en apartado anterior, que es *mysql* y el directorio de datos lo tiene como propietario. Si vamos a crear otro directorio alternativo para ubicar las bases de datos, debemos asegurarnos que éste tiene como propietario a *mysql*, y que también lo es para el diretorio que contiene la base de datos mysql (que también se denomina mysql).

Parada

mysqladmin shutdown
Si no estamos en Linux con la cuenta de root, podemos emplear esta sintaxis:
mysqladmin -u root shutdown

Comprobar que el demonio mysqld[19] está activo

mysqladmin ping

[18] Se deja al lector el estudio de este aspecto, aunque bastaría con recurrir a las herramientas gráficas de gestión de servicios para establecer el inicio del servidor al arrancar la máquina.

[19] Se trata del demonio asociado al servidor, y por tanto encargado de escuchar por el puerto 3306.

Determinar los procesos que están asociados a MySQL

Podemos optar por ejecutar *ps -ef | grep mysql*.

Obtener información de la ubicación del directorio de datos del servidor

Esta operación es importante, sobre todo si se emplean varios servidores MySQL en la misma máquina o se desea consultar esta información sobre un host remoto.

mysqladmin –port=3306 variables[20]

Si no se especifica el parámetro *–port*, se entiende como puerto el 3306.

Este comando muetra dicha información relativa al servidor MySQL ubicado en la máquina local que atiende por el puerto 3306. Se lista una tabla con los valores de las variables de configuración con las que trabaja el servidor. Por ejemplo, buscando la variable *datadir* sabremos dónde están las baes de datos del servidor. Si es esto lo que buscamos el comando puede filtrar esta información ejecutando lo siguiente:

mysqladmin variables | grep datadir

Si el servidor fuera remoto, bastaría con hacer mención a su nombre o IP.

mysqladmin –host=frolik variables

Cambiar la contraseña del usuario root de MySQL

Cuando se instala MySQL por primera vez en una máquina, el usuario *root* de MySQL queda sin asignar contraseña. Además se establecen otros aspectos relacionados con las restricciones de conectividad, que se resumen de la siguiente manera:

- El usuario *root* se puede conectar al servidor desde la máquina local sin contraseña. El usuario root posee todos los privilegios[21] y por tanto, puede hacer cualquier cosa sobre el servidor.
- Existe la posibilidad de acceder anónimamente al servidor si ésto se realiza desde la máquina local, para la base de datos denominada *test*[22]. No poseen privilegios de administración.

Para cambiar la contraseña del root, por ejemplo para establecerla a *root*[23], se ejecuta el siguiente comando:

mysqladmin -u root password "root"

Una vez ejecutado este comando, si decidimos, por ejemplo, parar el servidor y realizamos:

mysqladmin shutdown

comprobaremos que el sistema informa de que la conexión con el servidor ha fallado debido a que se ha denegado el acceso al usuario *root@localhost* a causa de que no se ha suministrado el password.

Desde el momento en que se ha asignado password al usuario root, será necesario incorporar los parámetros -u y -p al comando mysqladmin. Ahora para detener el servidor será necesario ejecutar:

mysqladmin -u root -p shutdown

El comando solicitará que le proporcionemos la contraseña de root, y una vez aceptada, procederá a detenerlo.

Si después de cambiar establecer por primera vez la contraseña del root, observa que al ejecutar, por ejemplo, el siguiente comando

mysqladmin -u root status

se sigue accediendo al servidor, se deben recargar las tablas de privilegios de MySQL mediante el comando *reload*, ejecutando

mysqladmin -u root reload.

[20] Es recomendable ejecutar este comando seguido de *more* ya que la salida es extensa. Basta con hacer *mysqladmin –port=3306 variables | more*.

[21] El concepto de *privilegio* es sumamente importante en el contexto de administración de bases de datos. Los privilegios son las autorizaciones que poseen los diferentes usuarios MySQL para poder ejecutar determinadas acciones sobre la información contenida en un servidor. Existen multitud de tipos de privilegios. Los hay que afectan al acceso de los datos de las tablas, a la capacidad de alterar la estructura de la base de datos, a modificar aspectos de la administración propia del servidor, etc.

[22] Se trata de una base de datos que no incorpora tablas. Sirve, inicialmente, para comprobar que se accede al servidor y se pueden hacer operaciones sobre él, partiendo ya de una base de datos creada.

[23] Evidentemente, esta es la última contraseña que debe ponerse al usuario root, sin embargo, por motivos nemotécnicos es preferible.

Finalmente, para eliminar la contraseña se ejecutará el comando
mysqladmin -u root -p password “”

Realizar copias de seguridad de las bases de datos

Existen dos formas de realizar copias de seguridad:

- Mediante la herramienta *mysqldump*. Es el más seguro, ya que al realizar la copia con la supervisión del servidor, nos aseguramos que mientras ésta se hace no se están modificando datos en ella. La desventaja estriba en que es un poco más lento que la alternativa descrita seguidamente. Esta herramienta no se limita a copiar los archivos, sino que genera un documento en formato texto que contiene las sentencias SQL necesarias para *recrear* la base de datos y su correspondiente información sobre otro servidor. Esto tiene el aliciente de que los archivos generados son transportables no sólo entre servidores MySQL de Linux, sino a otras plataformas e incluso máquinas con hardware diferente[24].
- Mediante copia directa de los directorios/archivos que forman la base de datos. Para ello se hace preciso asegurarse de que la base de datos no se está usando. Deberá pararse antes el servidor MySQL.

El uso de *mysqldump* no está restringido a las copias de seguridad, sino que si se desea copiar una base de datos de un servidor a otro, ésta es una buena manera de hacerlo.

Por ejemplo, para realizar una copia de la base de datos test de MySQL al directorio /home/profe, bastará con ejecutar:

mysqldump test >/home/profe/test.seg

El archivo *test.seg* contendrá las sentencias necesarias para crear las tablas e insertar los datos de la base de datos test actual. Si aún no posee tablas, el resultado será algo así como:

– MySQL dump 9.10
–
– Host: localhost Database: test
– ————————————————————
– Server version 4.0.18

Más adelante volveremos a retomar el comando, y procederemos no sólo a realizar la copia de seguridad, sino que la restauraremos sobre el servidor.

Y si lo que se pretende es volcar la base de datos desde un directorio de datos a otro, bastará con que en el directorio destino esté creada la base de datos, aunque esté vacía. Supongamos que deseamos volcar la base de datos *test* desde nuestro servidor local hacia la máquina identificada con nombre de host **servidor2**. La secuencia de comandos será la siguiente:

mysqladmin -u root -p -h servidor2 create test
mysqdump -u root -p test | mysql -u root -p servidor2 test

En el primer comando se crea la base de datos *test* en **servidor2**. En el segundo comando, se crea el archivo de copia *test* y se redirige como entrada al comando *mysql*, que tomará su contenido y lo asociará como instrucciones que deben ejecutarse sobre la base de datos *test* creada en el paso anterior. Se emplean los parámetros -u y -p suponiendo que ambos servidores poseen un usuario root con clave.

El comando *mysqldump* posee otras opciones. Pueden consultarse mediante *man mysqldump*.

Verificación y reparación de tablas[25]

Se dispone de dos herramientas que permiten realizar estas funciones: *myisamchk* e *isamchk*. La primera se emplea para tablas con formato MyISAM y la segunda para las tablas con formato ISAM[26].

[24] Esto precisa una matización. MySQL trabaja realmente con diferentes formatos internos de tablas. Si se está usando el formato MyISAM la transportabilidad entre máquinas con hardware diferente está asegurada.

[25] Antes de reparar una tabla hay que asegurarse de que se tiene copia de seguridad de la misma.

[26] Si observamos los archivos ubicados dentro de un directorio de una base de datos concreta, por ejemplo dentro del directorio mysql, veremos que se listan archivos que tienen la extensión .MYI y/o .ISM. Los primeros se corresponden con datos pertenecientes a tablas en formato MyISAM y los segundos a las tablas con formato ISAM. Actualmente el formato habitual es el MyISAM.

Ambas herramientas pueden trabajar con opciones que afectan a la verificación que se realizará sobre la tabla, sin embargo, el uso sin opciones puede ser válido en la mayoría de los casos.

Si ejecutamos *myisamchk *.MYI*[27] estaremos verificando las tablas de sistema del propio servidor MySQL.

Para reparar una tabla de forma automatizada y lo menos *traumática* posible, se ejecuta:
myisamchk –recover –quick <nombre de tabla>
Si esto no fuera suficiente, se procederá con:
myisamchk –recover <nombre tabla>
Finalmente, si aún hay errores, con:
myisamchk –safe-recover <nombre tabla>

2.4.2. MySQL Control Center, MySQL Administrator y MySQL Query Browser

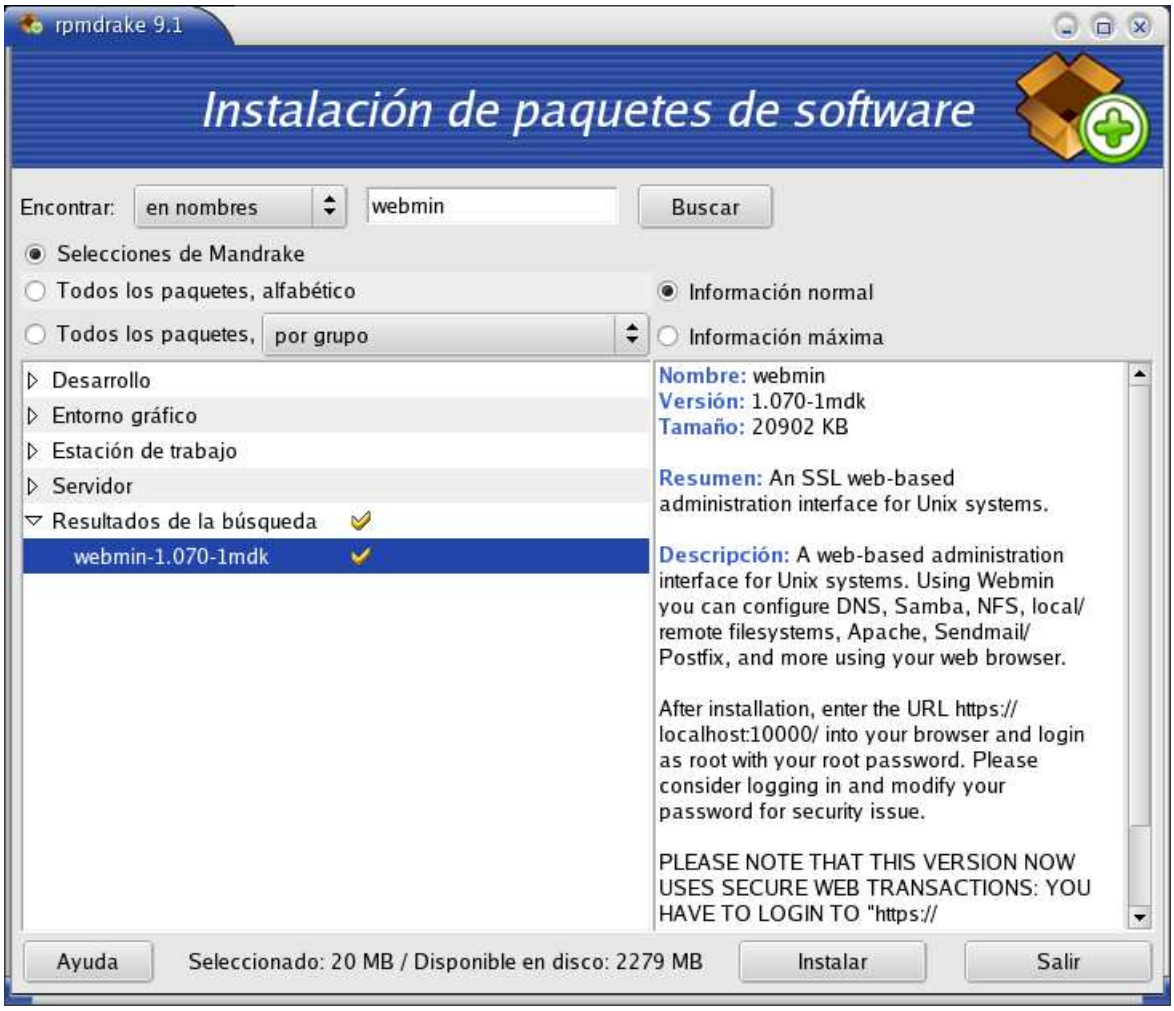
Se trata de herramientas que posibilitan administrar las bases de datos de un servidor MySQL, además de crear nuevas y de modificar las ya existentes, permitiendo incluso añadir datos a las tablas. Estas herramientas pueden obtenerse desde el propio website oficial de MySQL, concretamente en *http://dev.mysql.com/*, accediendo a sus correspondientes enlaces. Realmente MySQL Control Center ya no está disponible, ya que MySQL ha abandonado el desarrollo de esta aplicación siendo sustituida por una de administración (*MySQL Administrator*) y por otra que permite crear, modificar y consultar bases de datos y tablas (*MySQL Query Browser*). Guadalinex, sin embargo, incluye en su edición Ciudadano 2004 MySQL Control Center. Por otra parte, la herramienta MySQL Query Browser adolecía hace unos meses de algún *bug* que hacía que bajo ciertas circunstancias, la aplicación se cerrara inesperadamente. Por supuesto, existen otras herramientas que permiten realizar estas funciones, dos de ellas son Webmin y phpMyAdmin. Quizás esta última sea la más interesante debido, entre otros aspectos, por su independencia de plataforma, por la solidez de su diseño y por ser específica para configurar MySQL. Webmin[28] es una aplicación muy extendida que permite administrar la mayoría de los aspectos más importantes de un servidor Linux, incluyendo la configuración de la máquina y la de los servidores que corran en ella. Por supuesto, no todos los servidores pueden configurarse con Webmin, pero sí es cierto que la mayoría de los más extendidos sí. Procuraremos dar un repaso a varias de estas herramientas, dedicando un apartado especial a phpMyAdmin. Webmin la trataremos con Mandrake y MySQL Control Center con Guadalinex.

Mandrake

La distribución Mandrake 9.1 incorpora de *serie* la versión 1.070 de Webmin (la versión actual[29] es la 1.190). Se trata de una herramienta similar a phpMyAdmin en tanto que ambas son aplicaciones web, es decir, se ejecutan a través de un navegador web. Hace uso del protocolo HTTPS[30], y *atiende* por el puerto 10000. Por tanto, para acceder a ella basta con acceder a la dirección *https://localhost:10000*. Por supuesto, se puede acceder a la configuración de una máquina remota si en ella está instalado webmin con sólo hacer referencia al nombre de la máquina o a su IP. Por ejemplo, para acceder a la configuración de nuestra máquina servidora Linux desde cualquier otra (independientemente de la plataforma que se esté usando, ya sea Linux o Windows) bastará con introducir algo similar a *https://servidor:1000* o *https://192.168.0.1:10000*. Esto mismo es posible con otras herramientas que hagan uso de un navegador, como es el caso de phpMyAdmin o que posibiliten acceder a un servidor mediante protocolo TCP, como es el caso de MySQL Control Center.

Ejecutaremos rpmdrake para instalar el paquete webmin.

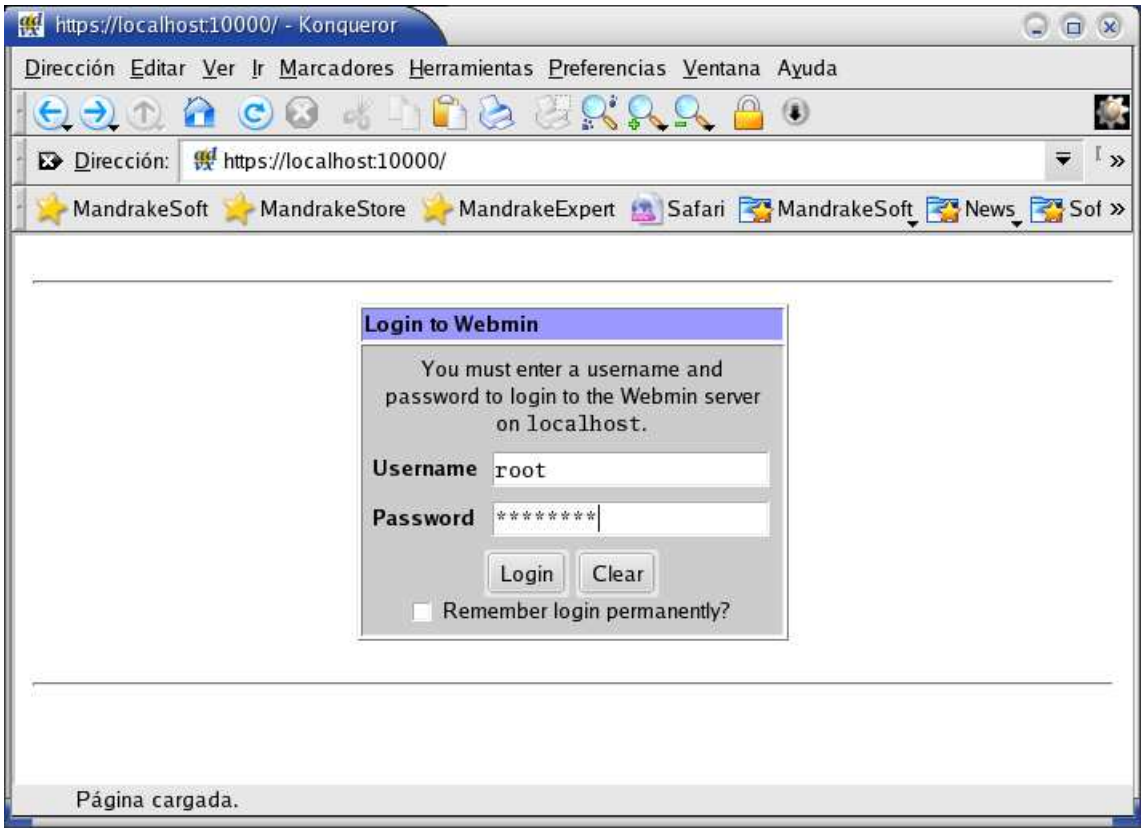
[27] Deberemos estar ubicados en /var/lib/mysql/mysql y asegurarnos que la extensión está en mayúsculas.
[28] Para más información, incluida la posibilidad de descarga, consultar http://www.webmin.com.
[29] Marzo de 2005.
[30] Protocolo para comunicaciones seguras de HTTP.



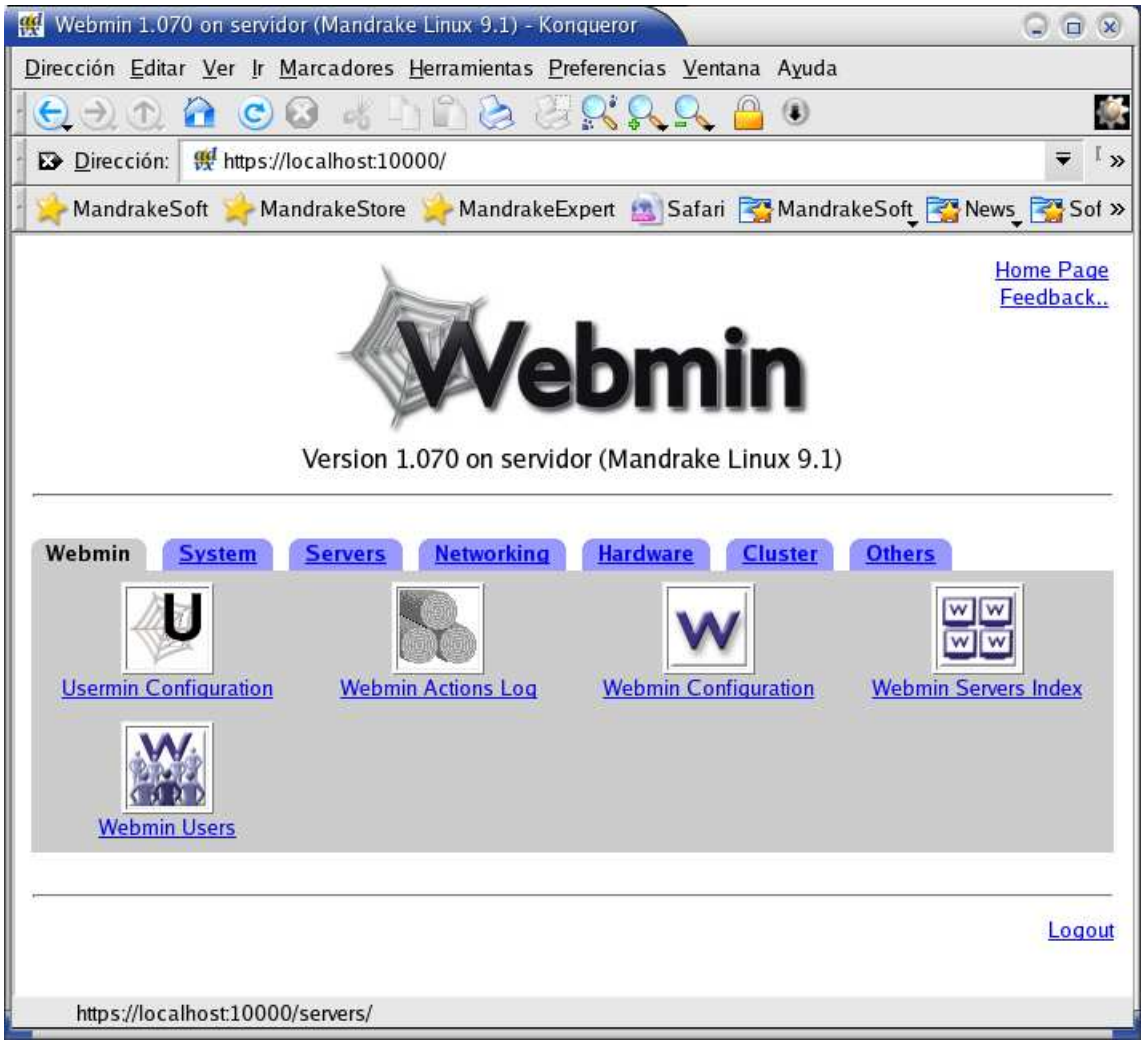
Y lo instalamos. Si al introducir la URL para administrar la máquina el navegador no la reconoce, abrimos una consola Linux y lanzamos el servicio webmin ejecutando **service webmin start**[31].

Al conectarnos a la aplicación web ésta solicita un nombre de usuario y una contraseña. Proporcionamos como nombre de usuario **root** y como contraseña la que tengamos establecida en el sistema para él.

[31] Si da problemas la ejecución de este comando, se puede probar a ejecutar directamente el comando webmin.



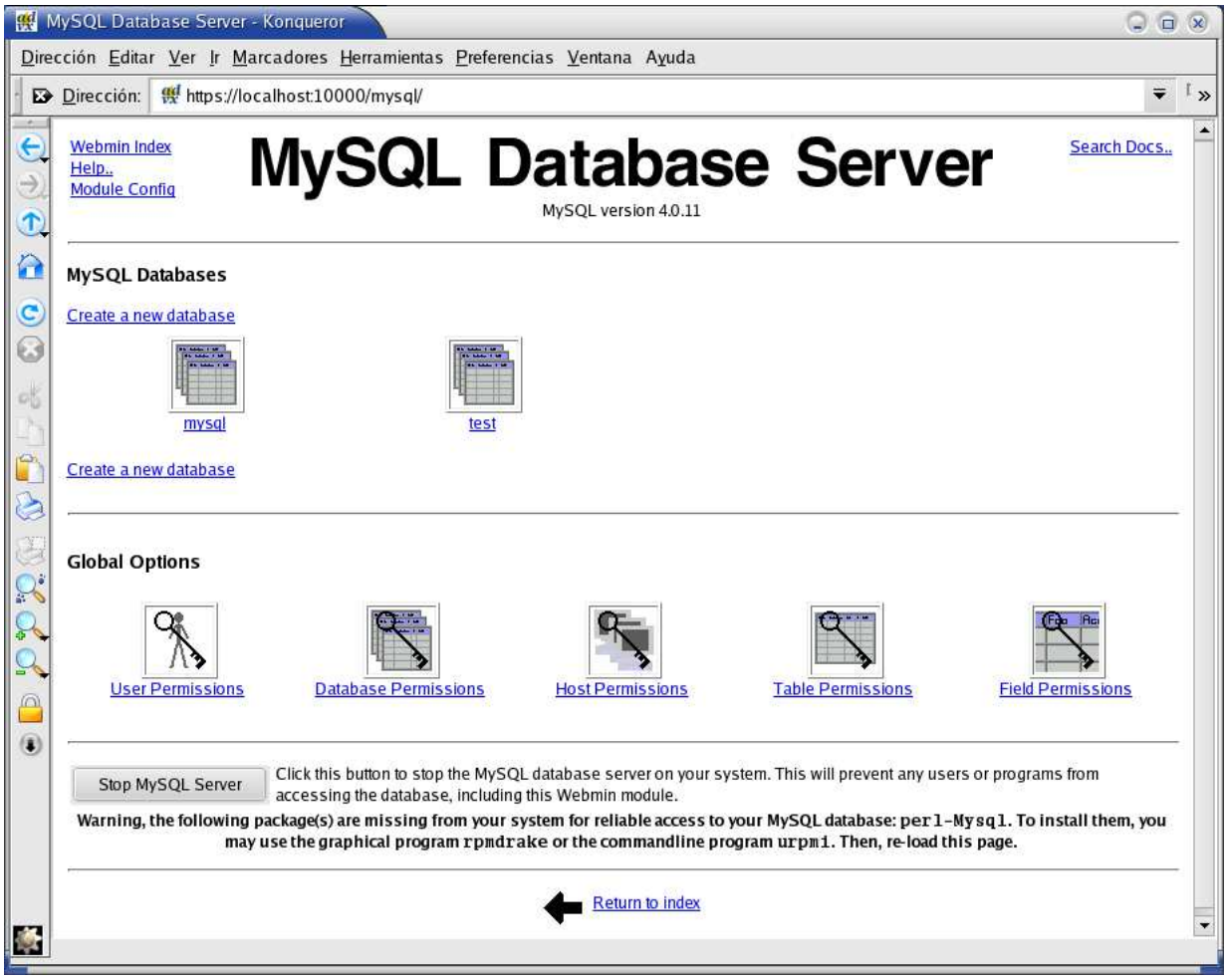
La interfaz principal de webmin es la siguiente:



Seleccionamos la *pestaña* **Servers** y buscamos el icono asociado a MySQL. Si nos posicionamos sobre él aparecerá una etiqueta emergente que nos indica la URL asociada a la administración desde webmin del servidor MySQL[32]. Hacemos clic sobre él[33].

[32] https://localhost:10000/mysql/.

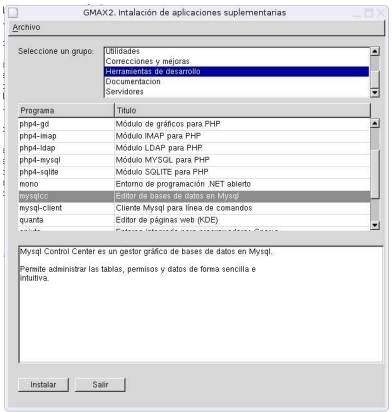
[33] Podría solicitarse que se introduzca además la contraseña del usuario **root** de MySQL. Por defecto este usuario carece de contraseña al instalarse el servidor MySQL.



Por ahora dejaremos el uso de esta herramienta para un apartado posterior.

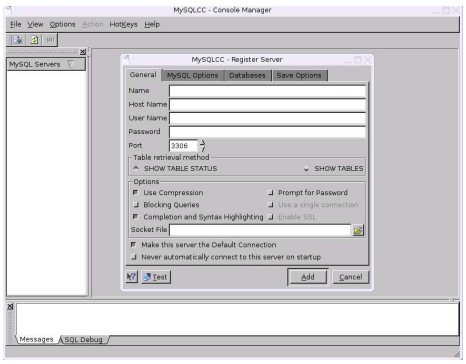
Guadalinex

En Guadalinex se instalará MySQL Control Center desde el mismo grupo al que pertenece PHP, es decir, **Herramientas de desarrollo**. Se selecciona el paquete **mysqlcc**.

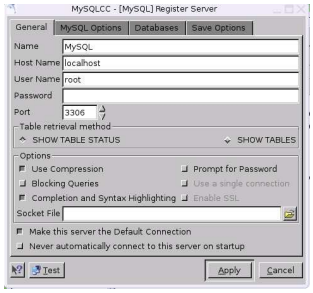


Una vez instalado, el Centro de Control de MySQL estará ubicado como una entrada en **Aplicaciones->Accesorios**.

Al ejecutarlo aparecerá una pantalla similar a la siguiente:



Procederemos a configurar el acceso a nuestro servidor MySQL recién instalado. Para ello rellenamos los campos de la ventana de configuración tal como se muestran:



Y entramos en la aplicación de administración. Dejamos el uso de esta herramienta para un apartado posterior.

2.4.3. PhpMyAdmin

phpMyAdmin es una herramienta que posibilita gestionar/administrar bases de datos MySQL mediante navegador haciendo uso del lenguaje PHP.

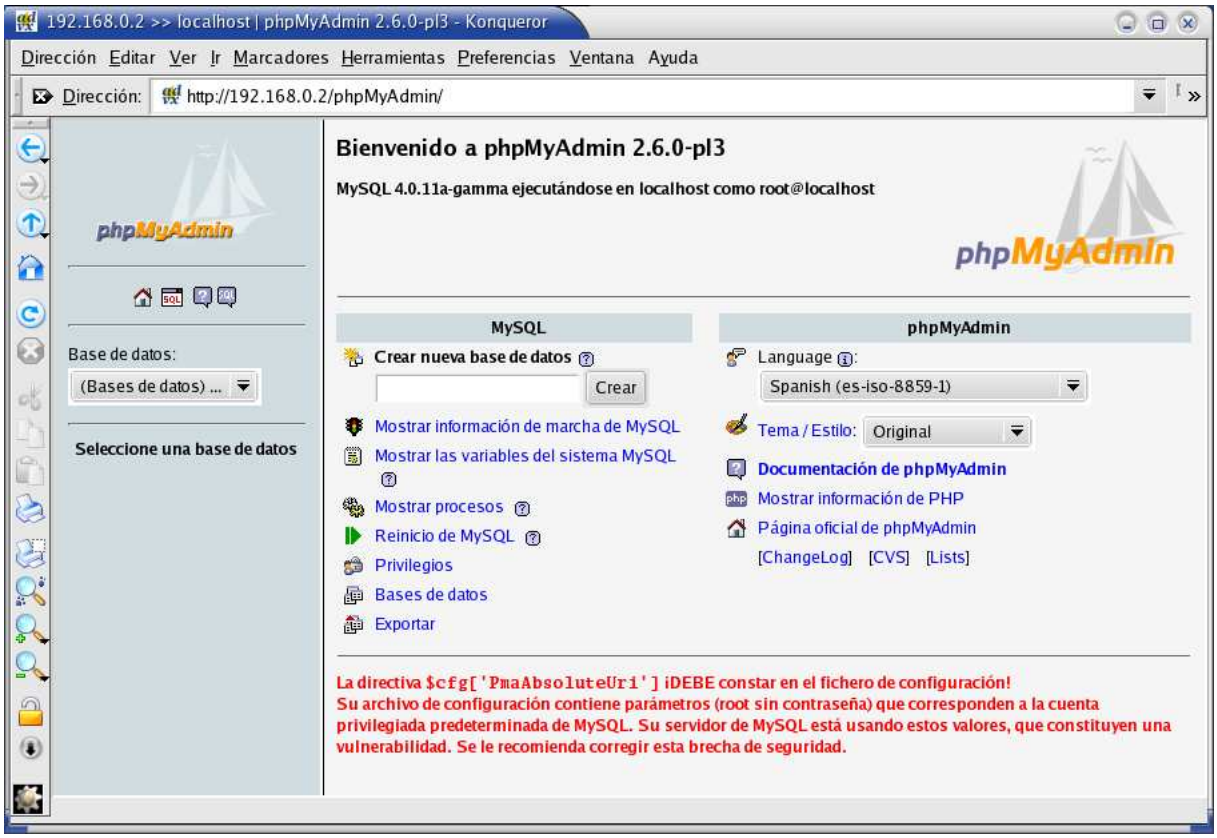
Para instalar phpMyAdmin basta con descargarse el archivo comprimido con la aplicación desde su website oficial, y descomprimirlo sobre el área de www que tiene asociada el servidor web. Si trabajamos con Apache, se descomprimirá en /var/www/html[34], creándose un directorio denominado **phpMyAdmin-X.Y.Z** donde X, Y y Z hacen referencia a la versión descargada. Puede renombrarse ese directorio para hacerlo de más fácil acceso desde el navegador. Por ejemplo, si se dispone del archivo comprimido **phpMyAdmin-2.6.0-pl3.zip**, lo descomprimiremos sobre /var/www/html, creándose del directorio **phpMyAdmin-2.6.0-pl3**. Si lo renombramos a **phpMyAdmin** mejor. Ahora para ejecutar la aplicación bastará con introducir la dirección **http://localhost/phpMyAdmin** en el navegador.

En apartado posterior se abordará el uso de esta herramienta.

Mandrake

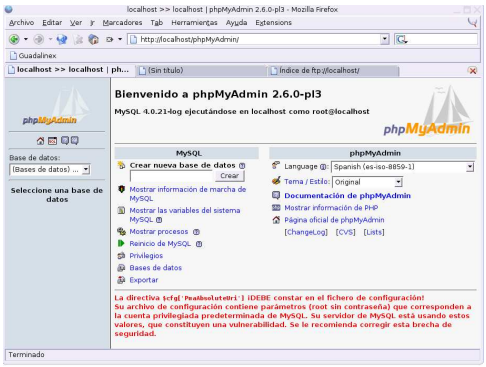
El aspecto ofrecido por la página principal de la aplicación se muestra seguidamente:

[34] /var/www si se trata de Guadalinex.



Guadalinex

El aspecto es prácticamente idéntico al ofrecido en Mandrake desde su navegador Konqueror.



2.5. Conceptos básicos de Bases de Datos Relacionales (BDR). La base de datos de alumnos de un curso on-line.

2.6. Creación de la base de datos Alumnos.

Dependiendo de la distribución de Linux sobre la que estemos trabajando optaremos por crear la base de datos con MySQL Control Center (caso de Guadalinex) o con Webmin (caso de Mandrake). Poste-

riormente, para introducir más datos o bien modificar la estructura de la base de datos, recurriremos a phpMyAdmin.

Supondremos que deseamos crear una base de datos que permita gestionar la información relacionada con los alumnos que están suscritos a los cursos on-line que se imparten en una determinada plataforma educativa.

Lo primero que debemos tener en cuenta es que el diseño de bases de datos es algo bastante más complejo de lo que puede parecer, y que hay infinidad de libros que abordan este tema, unos más teóricos y otros más prácticos. En este curso no vamos a tratar demasiado esta cuestión y nos limitaremos a plantear una base de datos bastante simple y cuya estructura surge casi de forma inmediata.

Para plantear el diseño de una base de datos debemos empezar por analizar el tipo de información que se va a gestionar, buscando en primer lugar las *entidades de información.* Por ejemplo, si vamos a gestionar información de alumnos que están suscritos a cursos, y de los alumnos hemos de almacenar ciertos datos y de los cursos también, está claro que al menos tendremos que gestionar información de dos entidades, *alumnos* y *cursos*. Que ambas identidades se caracterizan por una serie de datos que son específicos de cada una de ellas, y que cada conjunto de valores de esos datos determinan un alumno y un curso en concreto. Para determinar los datos que componen cada entidad se hace necesario tener muy claro qué información se va a gestionar de cada entidad. Para ello, cuanto más claro tengamos el *enunciado* del problema, mejor.

Supongamos que para cada alumno se precisa conocer su nombre, apellidos, edad, curso en el que está matriculado y calificación final obtenida. Y de cada curso, la descripción del mismo. Cada uno de estos tipos de datos recibe la denominación de *atributo*, de manera que un alumno o un curso se caracteriza por un conjunto de valores concretos de sus correspondientes atributos. Se denomina a ese conjunto de valores específicos como *instancia*. Cada entidad, en una base de datos relacional, será una tabla, y cada instancia se denomina registro. Los registros serán las filas, y los atributos las columnas de la tabla. Cada atributo almacenará un tipo de dato determinado (números, fechas, textos, etc.). Normalmente, tras crear las entidades y establecer sus atrubutos, comprobaremos que éstas están relacionadas entre sí. Entendemos que están relacionadas dado que partiendo de un registro de una de ellas podemos obtener uno o más registros pertenecientes a otra, y que están asociados o rlecionados con el primero. Supongamos que hemos establecido las siguientes entidades de nuestra base de datos como sigue:

Entidad Alumno:

Nombre	**Apellido1**	**Apellido2**	**Edad**	**Curso**	**Calificación final**

Entidad Curso:

Descripción curso

Observamos que cada alumno está matriculado en un curso[35], y por tanto debemos relacionar de alguna manera esta información con la almacenada en la entidad Curso. Alguno se empezará a preguntar porqué no está ya incluida la descripción del curso en la propia entidad Alumno. La respuesta es clara, si tenemos 200 alumnos en un mismo curso, habrá 200 registros cuyo atributo Curso posee el mismo texto. ¿Qué sucedería si tuviéramos que cambiar la descripción del curso? Nos veríamos obligados a modificar los 200 registros. ¿Y si nos equivocamos y sólo modificamos 199? Acabaríamos de dar con lo que se denomina en bases de datos como fallo de integridad. Además, normalmente, de un curso no sólo nos conformaremos con su descripción, quizás también se necesite almacenar información específica de cada curso como pueda ser la fecha de inicio y final, la url para acceder a él, etc. ¿Repetimos toda esta información en cada uno de esos 200 registros? Lo mejor será introducir en el atributo Campo de la entidad Alumno una *referencia* de curso, de manera que partiendo de ella, y localizando el registro que la posee en la entidad Curso, obtengamos toda la información del curso que posee dicha referencia. Esto es lo que entendemos por relacionar un registro (instancia) de una entidad con otro registro de otra. Esta referencia recibe el nombre especial de *clave,* en la entidad Alumno se denomina *clave ajena* o *clave externa*, y en la entidad Curso *clave principal.* Por tanto, podemos definir clave principal como aquel atributo cuyo valor permite distinguir de forma unívoca cada registro de la entidad, por ejemplo, un ejemplo claro de clave principal es el DNI[36]. Y como clave ajena, aquella que es principal en una entidad relacionada. Esta relación no sólo se da en un sentido, sino que podemos *saltar* desde la entidad Curso hasta la entidad Alumno. ¿Cómo? Pues haciéndonos la siguiente

[35] Adoptaremos el criterio que un alumno no puede estar matriculado en más de un curso. Más adelante se planteará este caso.

[36] Nos movemos en un mundo repleto de claves principales, aunque no seamos conscientes. Una matrícula de un vehículo, una dirección postal, una digitalización de nuestra retina, el ADN, una cuenta bancaria, son candidatos a ser claves principales en muchos procesos de tratamiento de la información.

pregunta ¿cuántos alumnos está matriculados en cada curso? La respuesta consiste en recorrerse la entidad Curso, tomar el valor de la clave principal, y localizar las instancias de Alumno cuya referencia de curso coincide con el valor de la clave principal seleccionada. Observamos que saber seleccionar adecuadamente la clave principal en una instancia es fundamental para posteriormente acceder a datos relacionados entre varias entidades. Por supuesto que esta elección depende en gran medida del enunciado del problema, pero una cosa debe estar clara, en el mundo de las bases de datos relacionales, cada entidad debe poseer su propia clave principal[37].

Así pues, tras este breve repaso sobre conceptos de bases de datos, procedemos a completar el diseño:

Entidad Alumno:

Nombre	**Apellido1**	**Apellido2**	**Edad**	**Código de curso**	**Calificación final**

Entidad Curso:

Código de curso	**Descripción del curso**

La entidad curso se convertirá en la tabla *cursos* y la entidad alumno en la tabla *alumnos*. Ahora bien, se ha determinado que cada tabla debe poseer una clave principal. ¿Qué clave principal adoptamos para la tabla Alumno? Hay varias posibilidades, una sería la unión de los atributos (también denominados *campos* cuando hacemos referencia a los términos de tabla y registro) Nombre, Apellido1 y Apellido2. Sin embargo esto no nos garantiza que no nos encontremos con dos alumnos con el mismo nombre e idénticos apellidos, además, los SGBDR suelen trabajar mejor con aquellas claves principales que son números o cadenas de caracteres de tamaño corto. Así pues una opción es emplear el DNI del alumno, o bien asignarle un código de alumno específico. Aquí nos encontramos con que nos surgen nuevos atributos (DNI, código de alumno), y esto es algo normal en el diseño de bases de datos.

Entidad Alumno:

Código alumno	**DNI**	**Nombre**	**Apellido1**	**Apellido2**	**Edad**	**Código de curso**	**Calificación final**

No sólo nos surgen nuevos atributos, algunos que no están especificados en el propio enunciado del problema (por ejemplo el código del alumno) si no que a veces surgen incluso nuevas tablas, que posibilitan relacionar los datos de otras dos. Por ejemplo. ¿Qué haríamos si un alumno puede estar matriculado en más de un curso y en cada curso puede haber más de un alumno? Comprobamos que ahora la relación es de muchos a muchos[38] (o varios a varios, es decir, dado un alumno varios cursos, y dado un curso, varios alumno). Pues bien, este tipo de relación no puede trasladarse al mundo de las bases de datos relacionales si no se define una tabla intermedia. Esta entidad intermedia tendrá la siguiente estructura:

Entidad AlumnoCurso:

Código alumno	**Código de curso**

Y observamos que está formada por las claves principales de las dos entidades relacionadas. Además, esta entidad permitiría almacenar toda la información que es específica de estos emparejamientos, por ejemplo, es aquí donde se guardarían las calificaciones de los alumnos, ya que para cada curso y alumno hay una calificación concreta.

Sabemos entonces que necesitaremos dos tablas (la posibilidad de contemplar alumnos que puedan matricularse en más de un curso queda como ejercicio para el lector). Al denominar las tablas procuraremos hacerlo siempre con letras minúsculas[39], sin usar espacios en blanco ni signos o caracteres propios de un

[37] Aquí se deben hacer dos matizaciones. En primer lugar, y de manera excepcional, podemos admitir entidades sin clave principal, pero muy excepcionalmente y siempre que se demuestre que no admite clave principal. Algunos teóricos demuestran que la existencia de entidades sin clave principal supone un error en el análisis de la información y su posterior reflejo en la estructura de la base de datos. En segundo lugar, una clave principal puede estar formada por un conjunto de atributo. Por ejemplo, si disponemos de una tabla que almacena productos, su clave principal puede estar formada por los atributos *clave-producto* y *clave-fabricante*, ya que dados dos o más fabricantes, éstos podrían disponer de productos con su misma referencia.

[38] Un aspecto importante en el diseño de bases de datos es determinar correctamente la cardinalidad de las relaciones existentes entre entidades. Esta cardinalidad puede ser de uno a uno, uno a muchos (muchos a uno es también lo mismo), muchos a muchos. Además, se debe contemplar si en la relación se contempla la posibilidad de que no haya en uno o en ambos extremos registro alguno relacionado. Tendremos entonces que en cada extremo de la relación existe una cardinalidad máxima y una mínima. Por ejemplo, pueden existir cursos en los que no hay alumnos matriculados. Decimos entonces que dado un curso, hay cero, o más alumnos matriculados. La cardinalidad mínima en el *lado* de la entidad alumno es 0, y la máxima es muchos. Por el contrario, dado un alumno, está claro que como mínimo está matriculado en un curso, y por tanto, la cardinalidad mínima en el lado de la entidad curso es 1, y la máxima, muchos.

[39] Puede adoptarse el criterio contrario, usar siempre mayúsculas, pero no es aconsejable mezclar ambos, ya que Linux distingue entre mayúsculas y minúsculas al trabajar con las tablas, ya que al fin y al cabo, éstas son archivos ubicados en un directorio y con extensiones determinadas.

idioma. Las llamaremos *alumnos* y *cursos*.

Cada registro de cada tabla se compone de una serie de campos. Cada campo es de un tipo de dato concreto. Y aquí empieza hacerse necesario analizar en dos fases esto. En primer lugar se debe determinar, para cada campo, qué tipo de dato *genérico* se va a emplear. Entendemos como tipo de datos genéricos los definidos como *número*, *carácter*, *cadena de caracteres*, *fecha*, *coma flotante*...etc. No son más que un punto de partida para posteriormente escoger el tipo concreto de entre los que nos ofrece el estándar ANSI SQL. Esto es así porque los tipos de datos SQL no coinciden necesariamente con los tipos de datos de los lenguajes de programación. Y no sólo eso, sino que diferentes SGBDR pueden soportar tipos diferentes. Por tanto, aquí se hace necesario informarse previamente de qué tipos son los disponible en el SGBDR con el que se vaya a trabajar, y a partir de ahí determinar los que se usarán en el diseño de la base de datos. La siguiente tabla muestra la mayoría de los tipos de datos que permite MySQL[40].

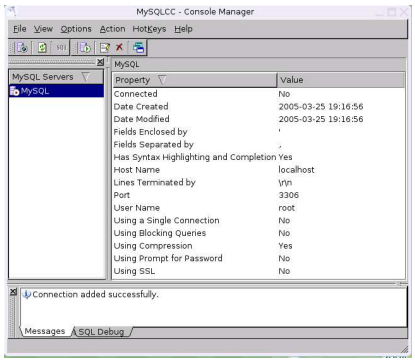
[40] Puede obtenerse mucha más información al respecto en http://dev.mysql.com/doc/mysql/en/column-types.html, o bien descargando el manual correspondiente en formato PDF desde http://dev.mysql.com/get/Downloads/Manual/manual-a4.pdf/from/pick.

Denominación	Descripción	Rango/Tamaño almacenaje
TINYINT	Numérico, entero muy pequeño.	con signo: -128 a 127, sin signo: 0 a 255
SMALLINT	Numérico, entero pequeño.	con signo: -32768 a 32767, sin signo: 0 a 65535
MEDIUMINT	Numérico, entero mediano.	con signo: -8388608 a 8388607, sin signo: 0 a 16777215
INT	Numérico, entero estándar.	con signo: -2 ^31 a (2 ^31) - 1, sin signo: 0 a (2^32) - 1
BIGINT	Numérico, entero largo.	con signo: -2^63 a (2^63)-1, sin signo: 0 a (2^64)-1
FLOAT	Numérico, precisión de coma flotante.	Valor mínimo distinto de 0: +/- 1.175494351E-38. Valor máximo distinto de 0: +/- 3.402823466E+38.
DOUBLE	Numérico, doble precisión de coma flotante.	Valor mínimo distinto de 0: +/- 2.2250738585072014E-308, Valor máximo distinto de 0: +/- 1.7976931348623157E+308
DECIMAL	Numérico, de coma flotante, representado como una cadena.	Varios, depende de la parte decimal que se represente.
CHAR	Cadena de caracteres, longitud fija.	Variable
VARCHAR	Cadena de caracteres, longitud variable.	Variable
TINYBLOB	Objeto binario muy pequeño.	$2^8 - 1$ bytes.
BLOB	BLOB pequeño.	$2^{16} - 1$ bytes.
MEDIUMBLOB	BLOB mediano.	$2^{24} - 1$ bytes.
LONGBLOB	BLOB largo.	$2^{32} - 1$ bytes.
TINYTEXT	Cadena de texto muy pequeña.	$2^8 - 1$ bytes.
TEXT	Cadena de texto pequeña.	$2^{16} - 1$ bytes.
MEDIUMTEXT	Cadena de texto mediana.	$2^{24} - 1$ bytes.
LONGTEXT	Cadena de texto larga.	$2^{32} - 1$ bytes.
ENUM	Enumeración.	65535 miembros
SET	Conjunto.	64 miembros
DATE	Fecha (AAAA-MM-DD)	
TIME	Hora (hh:mm:ss)	
DATETIME	Fecha y hora (AAAA-MM-DD hh:mm:ss)	
TIMESTAMP	Tiempo transcurrido (AAAAMMDD-hhmmss)	
YEAR	Año (AAAA)	1901 a 2155.

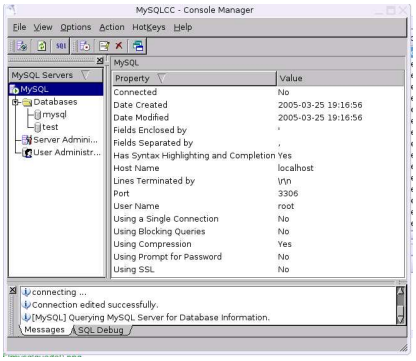
En los siguientes apartados, veremos más detenidamente qué tipos de datos asignaremos a cada campo de las tablas.

2.6.1. Creación de la base de datos Alumnos desde Guadalinex (MySQL Control Center)

Ejecutamos MySQL Control Center, y veremos lo siguiente:

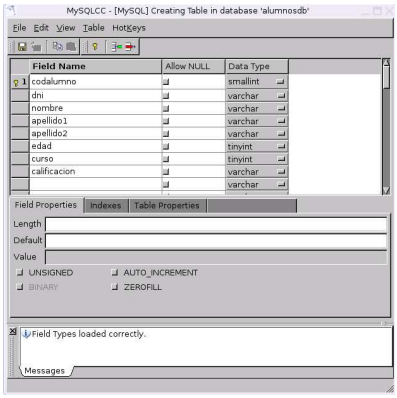


Si hicieramos doble clic sobre el icono asociado al servidor MySQL, obtendríamos un mensaje de error en la zona inferior de mensajes indicando que se impide el acceso al usuario *root@localhost*[41]. Esto es debido a que el root de MySQL ya posee contraseña. Para resolver esto bastará con colocar el puntero del ratón sobre el icono asociado al servidor MySQL y pulsar el botón derecho. Seleccionamos la opción *Edit*. Añadimos el password, pulsamos *Apply* (podemos antes pulsar *Test* para comprobar si el password es correcto) y cerramos. Hacemos doble clic sobre el icono del servidor MySQL. Obtendremos:



Ahora colocamos el puntero del ratón sobre la entrada *Databases* que cuelga del icono MySQL. Pulsamos botón derecho del ratón y seleccionamos la opción *New Database*. Nos solicita un nombre de base de datos. Le proporcionamos *alumnosdb*. Aparece la nueva base de datos en el arbol de jerarquía de bases de datos del servidor. Seleccionamos la base de datos *alumnosdb* y pulsamos de nuevo el botón derecho del ratón. Esta vez escogemos la opción *Connect*. Aparece una entrada nueva denominada *Tables*. Seleccionamos *Tables* y pulsamos nuevamente el botón derecho del ratón. Seleccionamos *New Table*. Nos aparece una ventana con el formulario que debemos rellenar para crear la tabla. Primero creamos su estructura para luego introducir los datos. Se rellenan los nombres de los campos y sus correspondientes tipos de datos tal como se observa en la siguiente imagen:

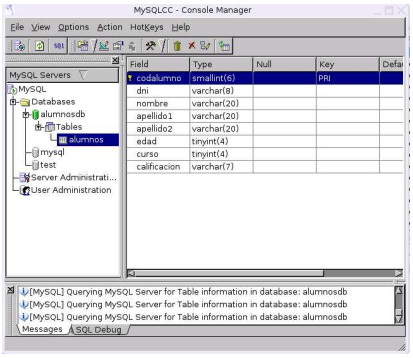
[41] Supondremos que se configuró el acceso tal como se indicó en apartado anterior al instalar MySQL Control Center.



Si observamos la imagen, el campo *codalumno* tiene asociada una llave amarilla. Es el indicador de que es la clave primaria o principal. Para asignársela bastará con tener seleccionado el campo y pulsar sobre el botón de la llave. Cada vez que se selecciona un campo podemos añadir información asociada a él mediante las pestañas *Field Properties*, *Indexes* y *Table Properties*. Será en el cuadro de texto *Length* de la primera de estas pestañas donde estableceremos los anchos de los campos siguientes:

Campo	Ancho
dni	8
nombre	20
apellido1	20
apellido2	20
calificacion	7

Pulsamos el botón del disquete e introducimos el nombre de la tabla cuando nos lo pida la aplicación. La denominaremos *alumnos*. Después de haber hecho esto cerramos la ventana de edición de tabla y volvemos a la ventana principal. Ahora la base de datos posee una tabla:



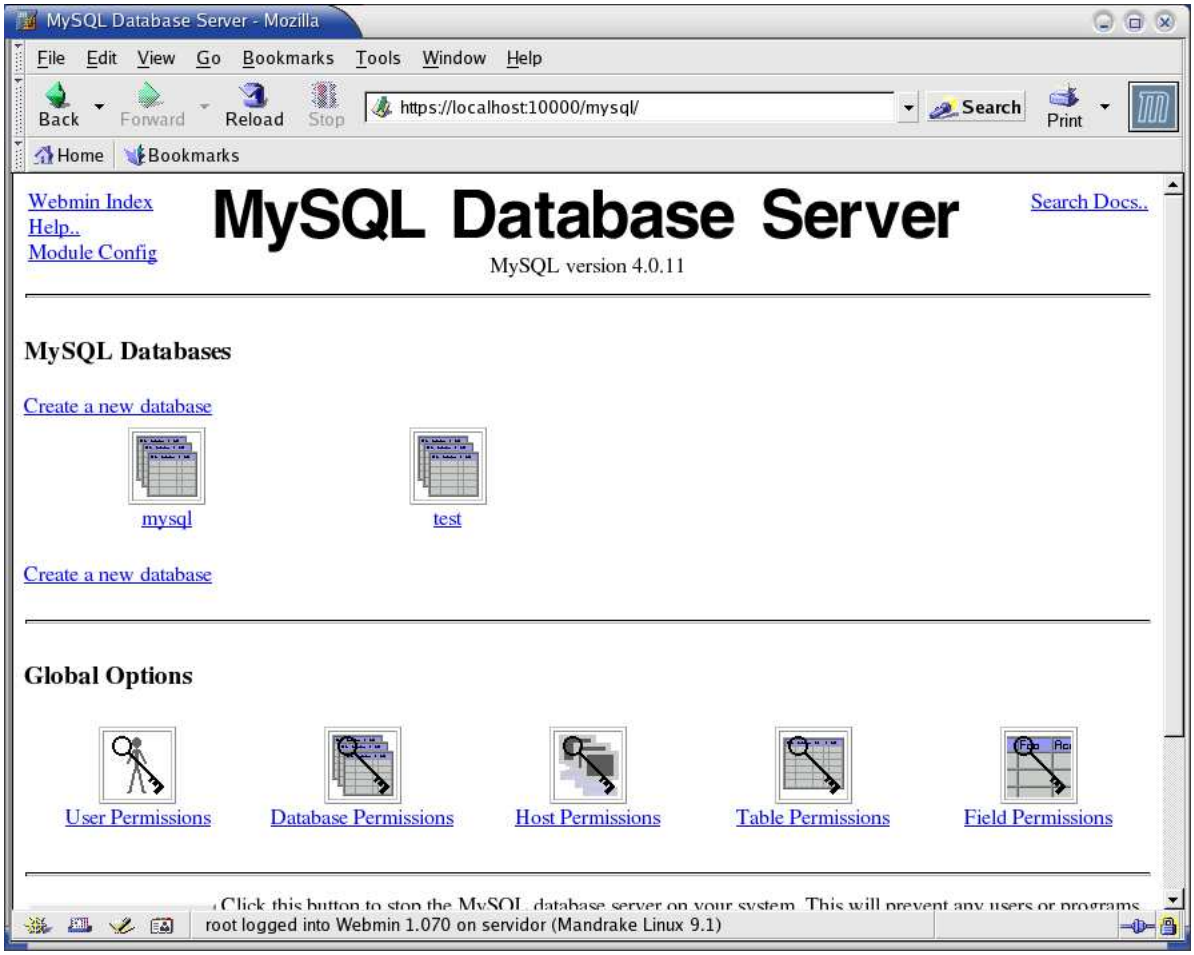
Hacemos lo propio con la tabla *cursos*, que tendrá la siguiente estructura:

Nombre campo	Tipo
codcurso (clave principal)	tinyint
descripcion	text

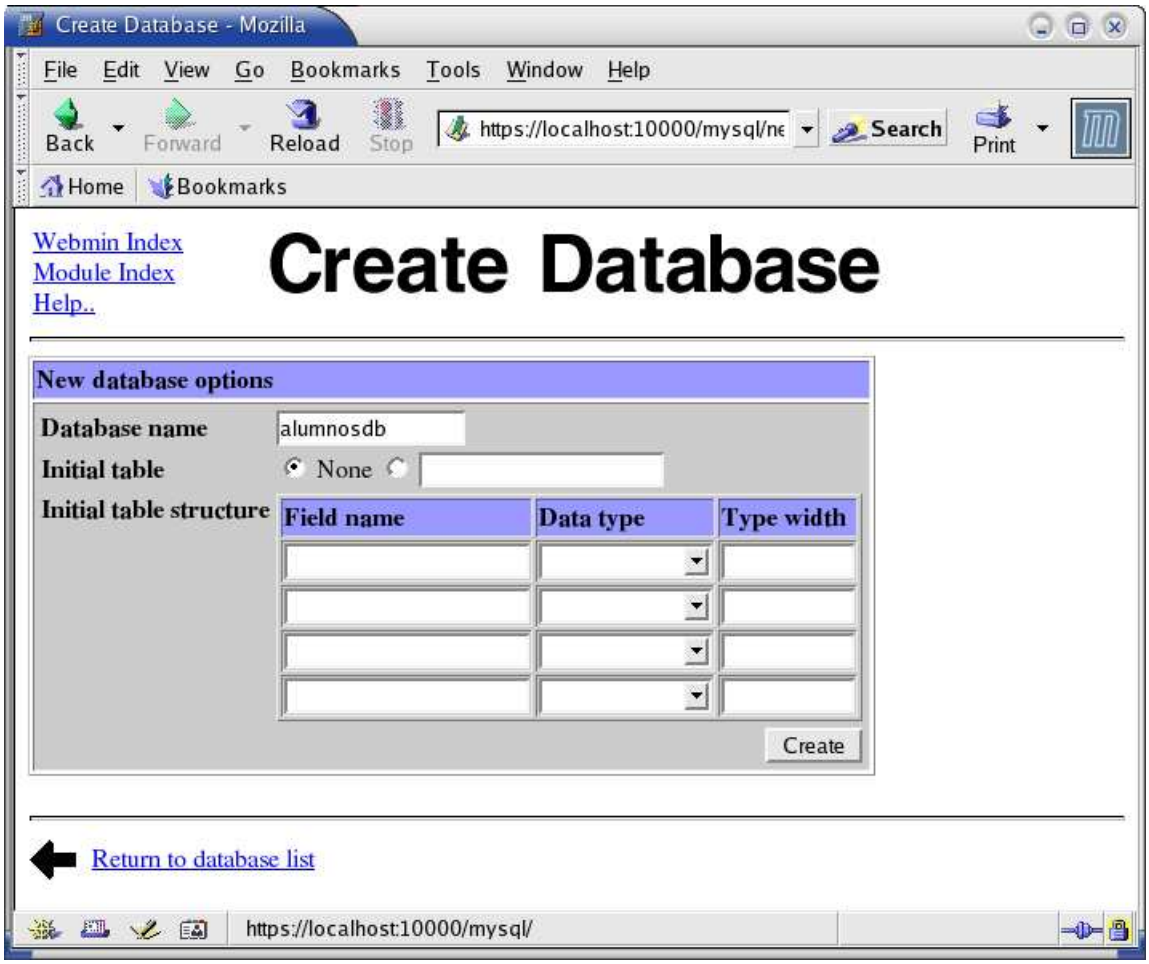
Para introducir datos, si no deseamos hacerlo con phpMyAdmin, se puede optar por hacer doble clic sobre la tabla a la que le queremos añadir los datos y seguidamente, pulsar el botón de inserción de registro.

2.6.2. Creación de la base de datos Alumnos desde Mandrake (Webmin)

Accedemos a webmin mediante la URL *https://localhost:10000* y seleccionamos sucesivamente, la pestaña *Servers* y el icono *MySQL Database Server*.

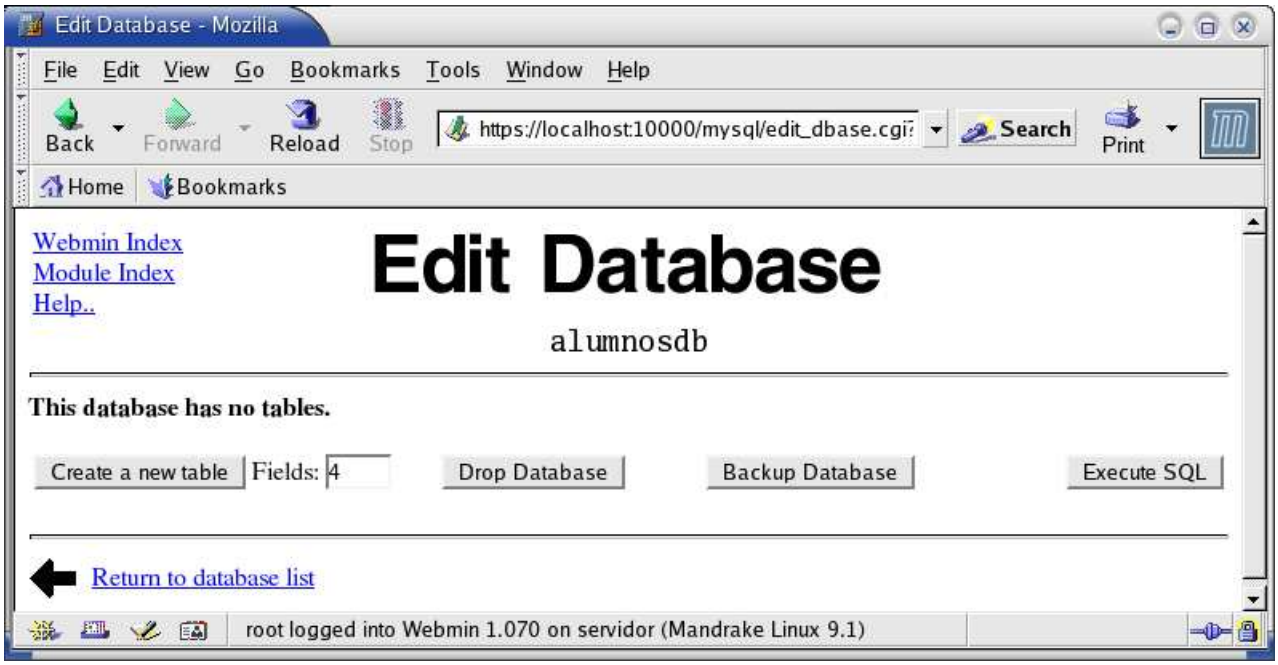


Se pulsa sobre el enlace *Create a new database* para crear la base de datos *alumnosdb*. Aparece la página de creación de base de datos. Introducimos el nombre de la base de datos, pero no creamos en este momento ninguna tabla:



Se pulsa el botón *Create*. Si no aparece el icono correspondiente a la base de datos recien creada pulsamos el botón de recarga de página del navegador[42]. Si no aparece la nueva base de datos volvemos a repetir el proceso. Para crear las tablas pulsamos sobre el icono de *alumnosdb*. Aparecerá la página de creación de tablas.

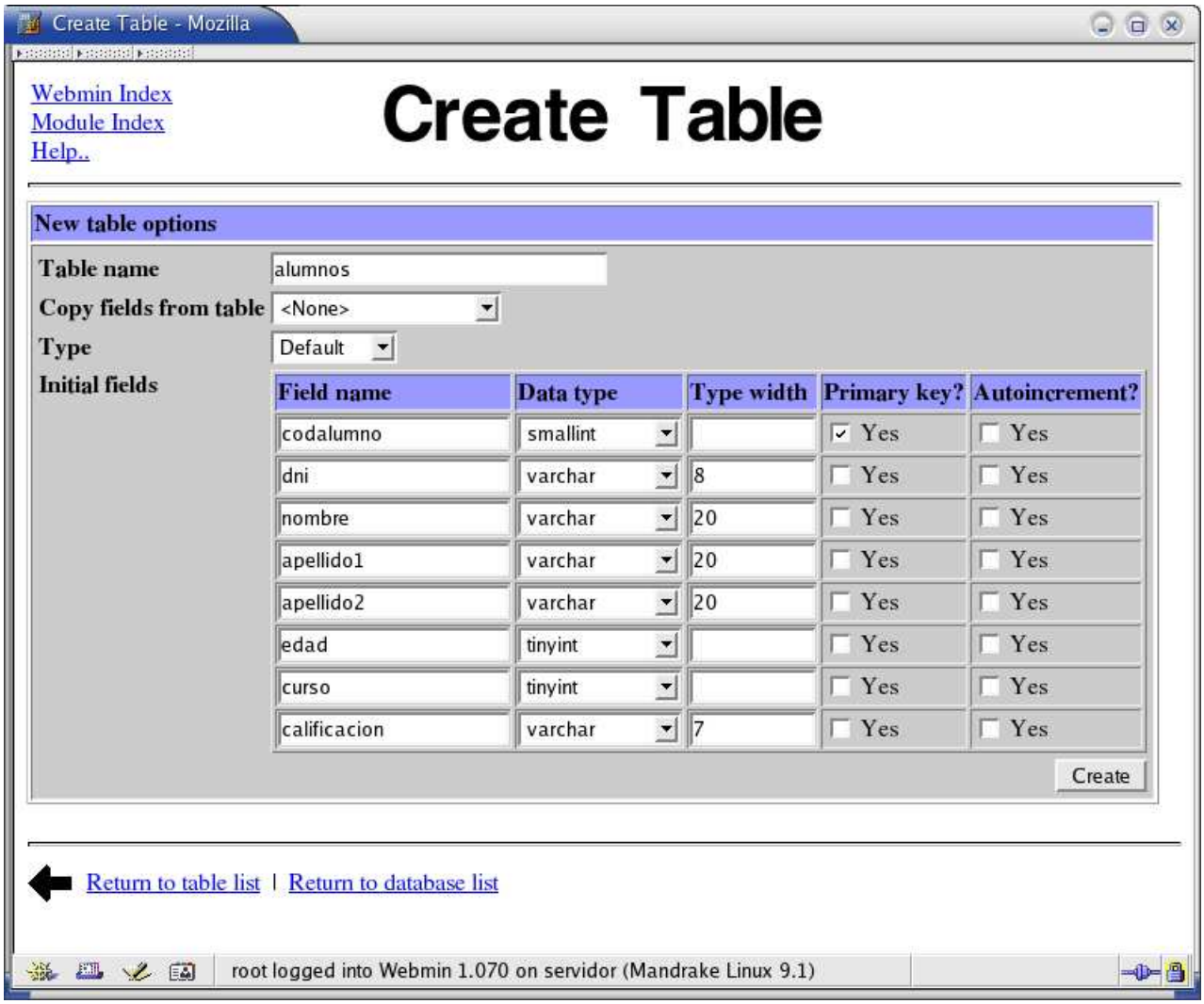
[42]En este punto puede suceder que webmin indique que se precisa instalar el paquete *perl-Mysql*. Lo instalamos.



Introduciremos en primer lugar la tabla *alumnos*. Para ello introducimos en el campo *Fields* el número de campos que tendrá la tabla (8) y pulsamos el botón *Create a new table*. Aparece un formulario donde introduciremos el nombre de la tabla, y para cada campo, su nombre[43], el tipo de dato, el ancho y si es clave principal o no[44]. La imagen siguiente muestra cómo quedaría la definición de la tabla *alumnos*:

[43] Procuraremos escribir siempre en minúsculas, sin espacios y con nombres más o menos identificativos.

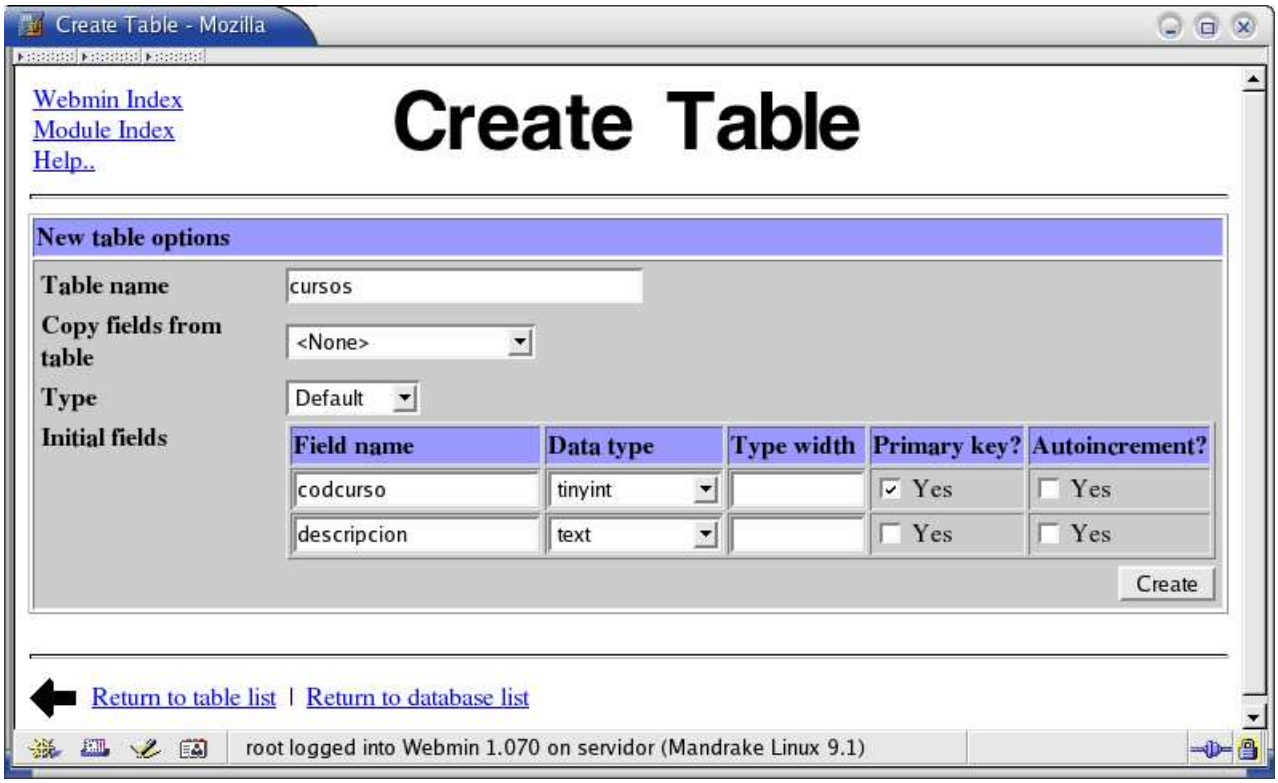
[44] La tabla alumnos posee como clave principal el campo *codalumno*. Suele denominarse también a la clave principal como *clave primaria*. La columna *Autoincrement* sólo tiene sentido si el campo almacenará valores secuenciales en incremento de 1 y asignados automáticamente por el SGBDR cada vez que se crea un nuevo registro. Este no es nuestro caso, aunque para un código de alumno podría plantearse. Es preciso advertir que si se borra un registro, el valor de los campos de tipo autoincremento no vuelven a utilizarse.



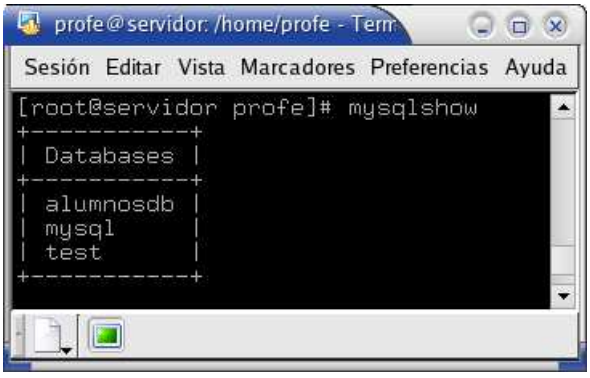
Observamos que establecemos ancho para aquellos tipos de datos que almacenan caracteres, para los numéricos no será necesario, salvo que no se desee utilizar más de un número determinado de dígitos. El campo *calificacion* tiene tamaño 7 ya que podrá contener dos posibles valores "APTO" y "NO APTO"[45]. Se pulsa el botón *Create*.

Volvemos a repetir el proceso, esta vez con la tabla *cursos*. Se crea la tabla partiendo de los datos siguientes:

[45]Este tipo de campo que admite un conjunto limitado de posibles valores es precisamente el tipo ENUM. Sin embargo, se deja al lector el uso del mismo.



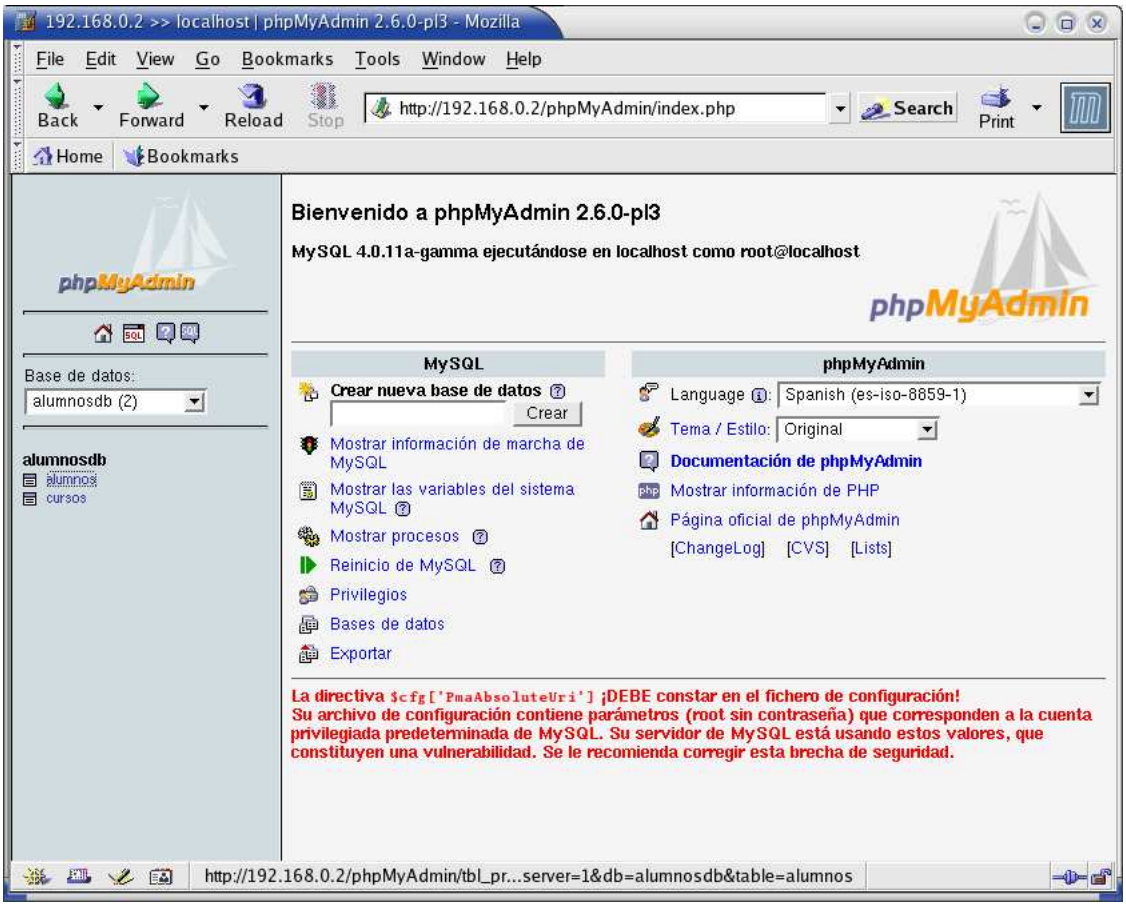
Llegados a este punto, y antes de introducir datos en las tablas, podemos ejecutar el comando *mysqlshow*[46] y comprobar que efectivamente ya dispondemos de una base de datos más. El resultado será similar a éste:



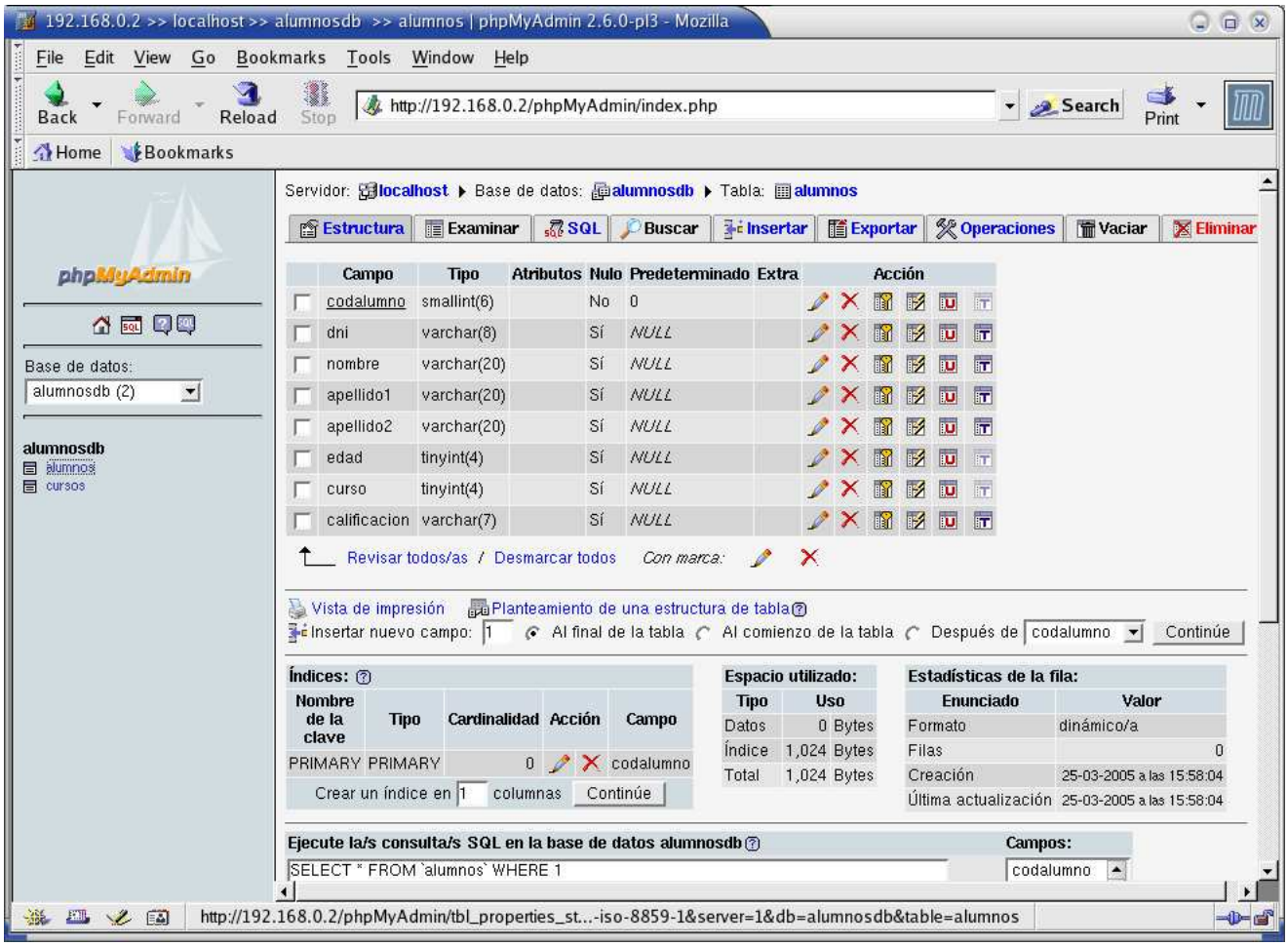
2.6.3. Introducción de una batería de datos en Alumnos (phpMyAdmin, ambas distribuciones)

Para introducir los datos procedemos a ejecutar la aplicación web phpMyAdmin mediante la URL *http://localhost/phpMyAdmin*. Antes de seleccionar la base de datos, desplegamos la lista de opciones de idiomas y seleccionamos el español. Tras escoger la base de datos *alumnosdb*, el navegador mostrará algo similar a lo siguiente[47]:

[46] Ejecutar *mysqlshow -u root -p* en caso de que ya se haya puesto contraseña al root de MySQL.
[47] No nos preocuparemos ahora por los mensajes de error que nos muestra la aplicación.

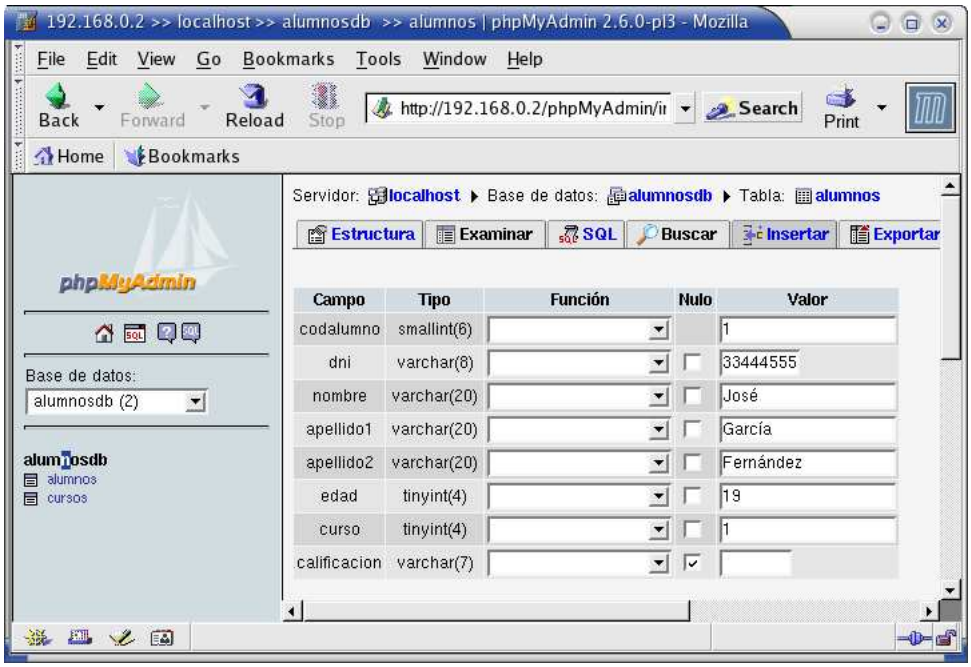


Seleccionamos la tabla *alumnos*. Aparecerá la siguiente pantalla:

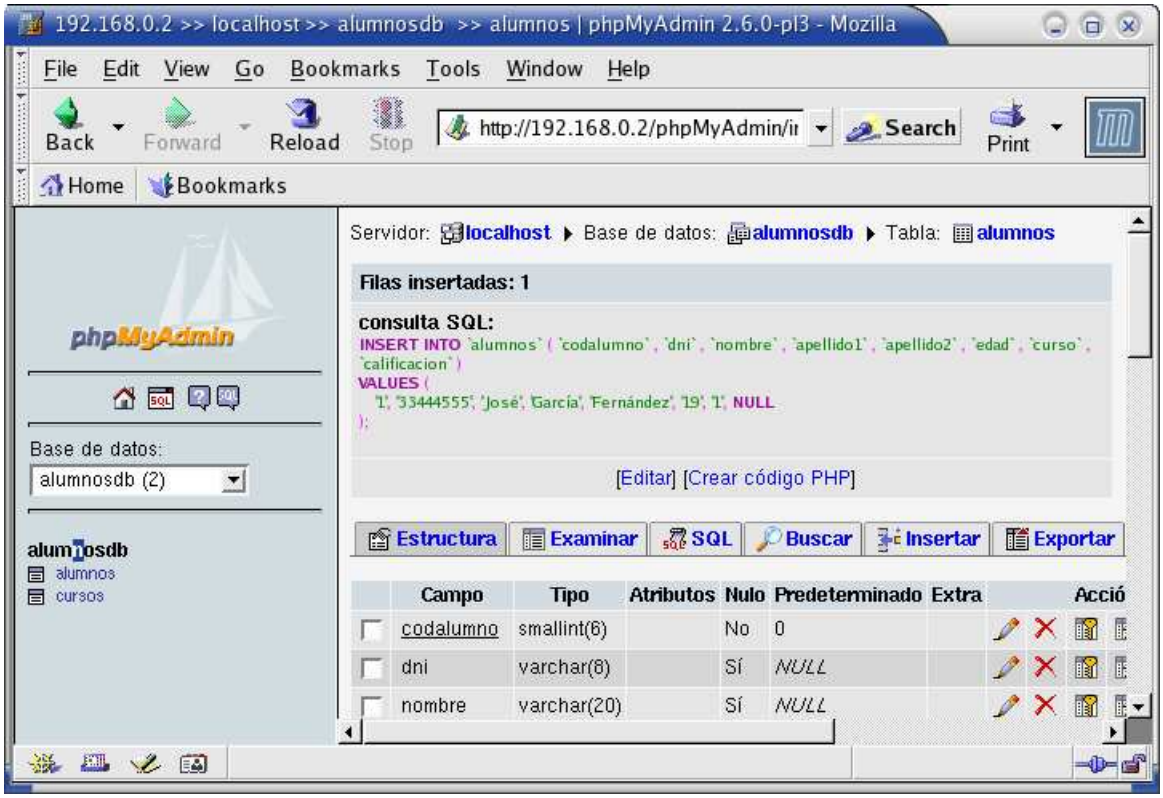


Observamos que la cantidad de información suministrada por phpMyAdmin es sensiblemente superior a la ofrecida por Webmin. Para insertar un registro pulsamos sobre la pestaña *insertar*. Aparece un formulario de inserción de registros. Introducimos los datos del primer alumno[48]:

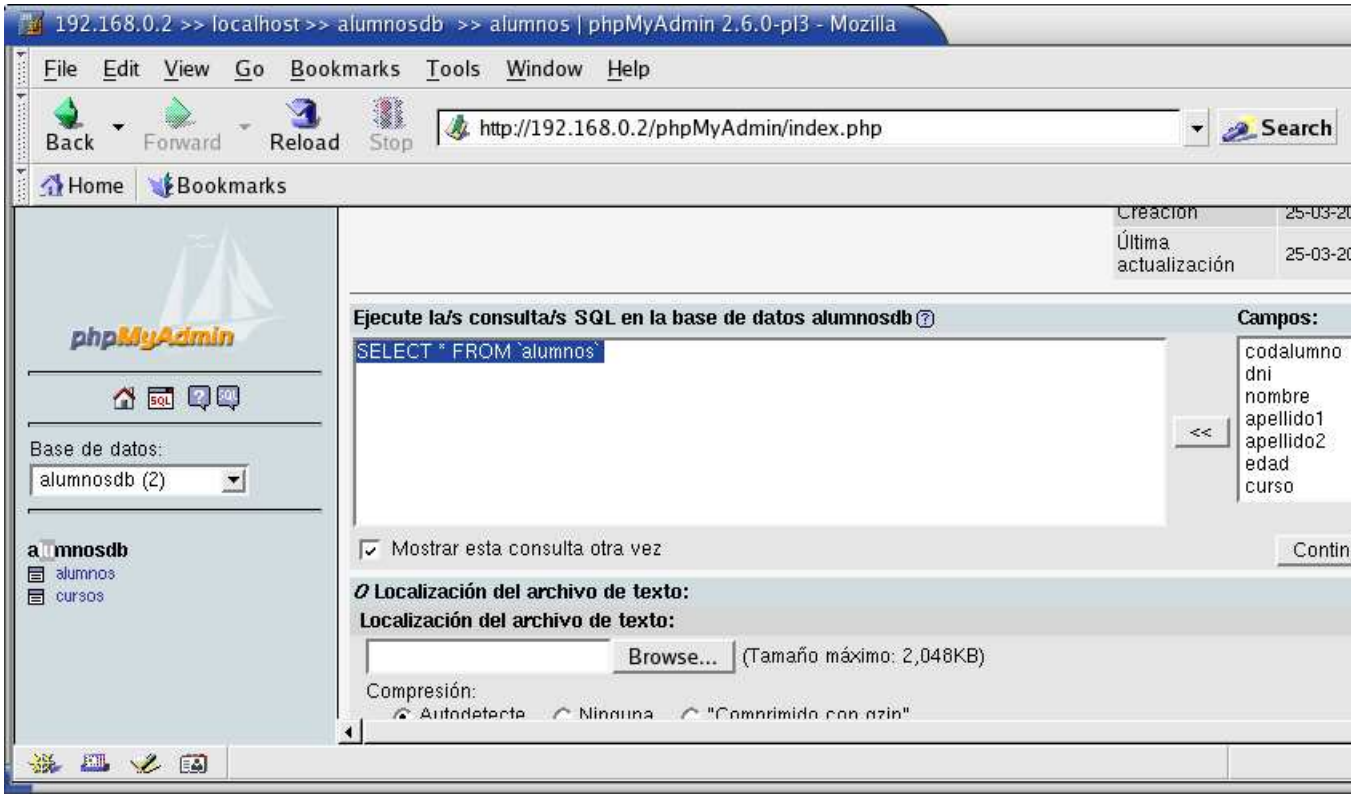
[48] La columna *Función* no la rellenamos.



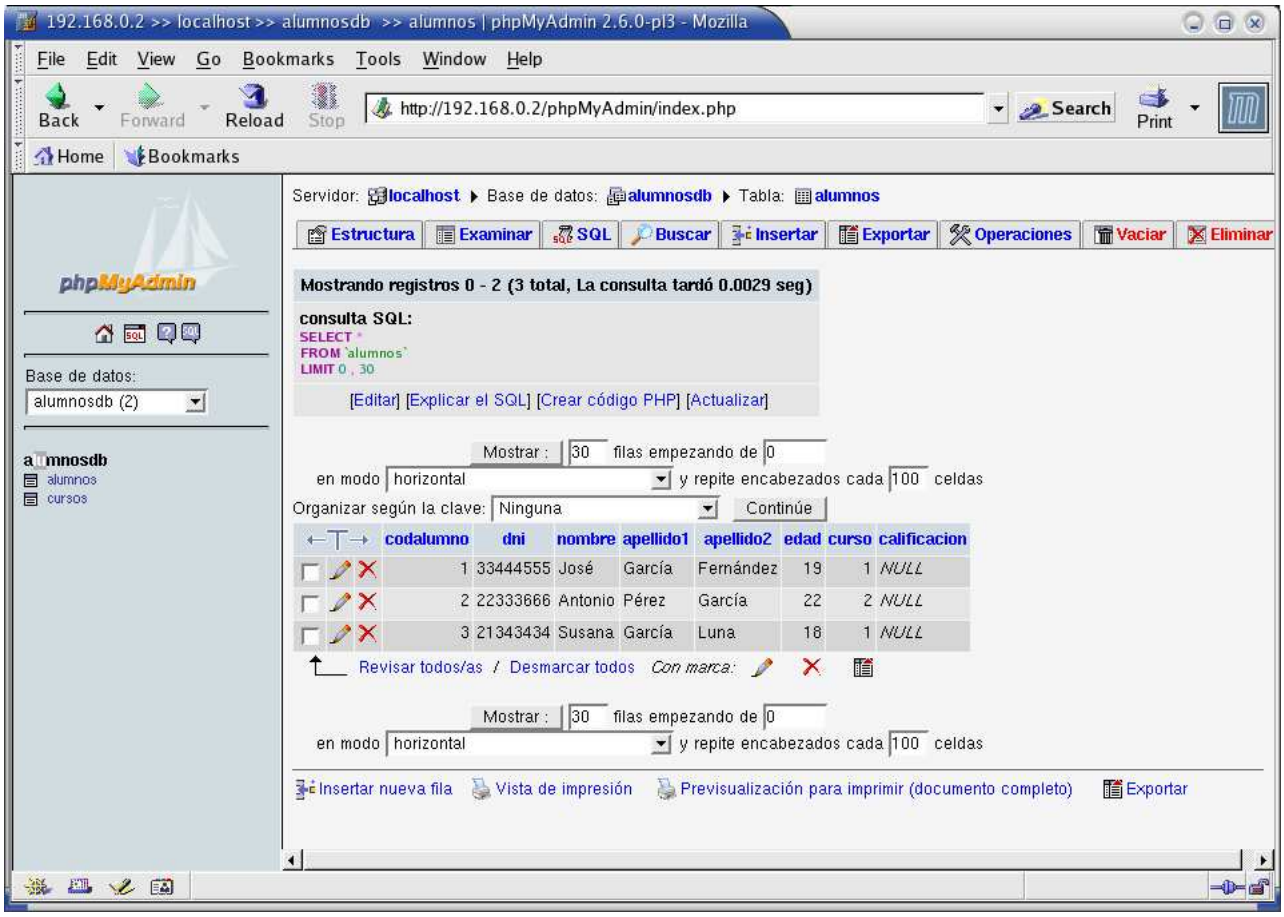
Dejamos en blanco el campo *calificacion* ya que hasta que no se finalice el curso no se rellenará. Pulsamos el botón *Continúe*. Al regresar a la página anterior observaremos que se ha añadido al principio un área denominada *consulta SQL:*. Ahí podremos observar la sentencia SQL que posibilita insertar un registro. Esta es una manera de ir aprendiendo el uso del lenguaje SQL.



Insertaremos dos o tres alumnos más. Para comprobar qué datos hemos introducido, ejecutamos la sentencia SELECT siguiente desde la página que contiene la información de la tabla *alumnos*:



Tras pulsar el botón *Continúe*, phpMyAdmin nos mostrará el siguiente resultado:



Después procederemos a insertar la información relativa a los cursos. Éstos serán los siguientes:

Código de curso	Descripción
1	Linux
2	Java

2.7. El lenguaje SQL. Operaciones básicas con el comando mysql.

Como se avanzó en apartado anterior, para trabajar con MySQL se recurre al lenguaje SQL[49], un estándar dentro de las bases de datos relacionales, y lenguaje en base al que se administran y gestionan los SGBDR más importantes de la actualidad.

SQL se estructura en base a un conjunto de instrucciones, que reciben la denominación de *sentencias*, cada una de ellas cubriendo una parte de la funcionalidad que ofrece el lenguaje.

Es importante dejar claro que cuando estamos usando este lenguaje están interviniendo en el proceso dos aplicaciones, una es el servidor de la base de datos y la otra el programa cliente. Es en el servidor donde se ubican las bases de datos, y desde el que se cursan las peticiones de los usuarios ubicados a lo largo de la red. Y los programas clientes son las aplicaciones que emplean esos usuarios para acceder a la información almacenado en los servidores. Estamos por tanto ante una típica estructura *cliente-servidor*.

Como se comentó en el apartado 2.4, la instalación de MySQL incluye, entre otras herramientas, un cliente en modo texto para acceder al servidor, se trata de *mysql*.

[49] Para obtener un tutorial sobre el SQL de MySQL: http://dev.mysql.com/get/Downloads/Manual/manual-a4.pdf/from/pick. Y para acceder a este mismo manual de forma online: http://dev.mysql.com/doc/mysql/en/index.html.

Si analizamos cómo hemos creado la base de datos y sus tablas, comprobaremos que ha sido empleado como usuario el *root* de MySQL, y su correspondiente contraseña. Además se ha realizado desde el mismo equipo en el que está ubicado el servidor MySQL. El usuario y el equipo desde donde se conecta el usuario son dos aspectos muy importantes a considerar cuando se gestionan bases de datos MySQL, ya que existe una completa gama de niveles o grados de restricción aplicables a las bases de datos y sus correspondientes tablas, que están en función de ellos. Se puede establecer que un determinado usuario pueda crear tablas en una base de datos si está intentando realizar esta opración desde la propia máquina donde se ubica el servidor MySQL que la contiene, pero impedirlo si el equipo es otro. Como esta restricción se pueden establecer otras muchas. Así pues, cuando se diseña un base de datos MySQL se hace imprescindible estudiar qué usuarios podrán acceder a ella, desde dónde y bajo qué restricciones. Toda esta política contribuye a fortalecer la seguridad de los procesos relacionados con la administración y gestión de las bases de datos.

Hasta ahora sólo se dispone de un usuario, *root*[50]. Crearemos un nuevo usuario, que llamaremos *cep*. Le asignaremos la contraseña *cep*. Y le asignaremos todos los privilegios posibles sobre la base de datos *alumnosdb* y sobre sus correspondientes tablas. Para ello ejecutaremos el comando *mysql*, proporcionando el nombre de usuario *root* y su contraseña. De esta manera, como *root* posee privilegios de administración, se podrá crear el nuevo usuario tal como deseamos. El comando es:

mysql -u root -p

El parámetro **-u** indica que el identificador que viene detrás hace referencia a un nombre de usuario, y el parámetro **-p** que se debe solicitar la contraseña de ese usuario para poder llevar a cabo la operación. El cliente *mysql* pedirá la contraseña, se la proporcionaremos, y entraremos en la aplicación para poder ejecutar de forma interactiva sentencias SQL sobre el servidor.

Como se adelantó párrafos atrás, las sentencias SQL atienden a diferentes tipos de funcionalidad, y una de ellas es la administración de los privilegios de los usuarios sobre las bases de datos.

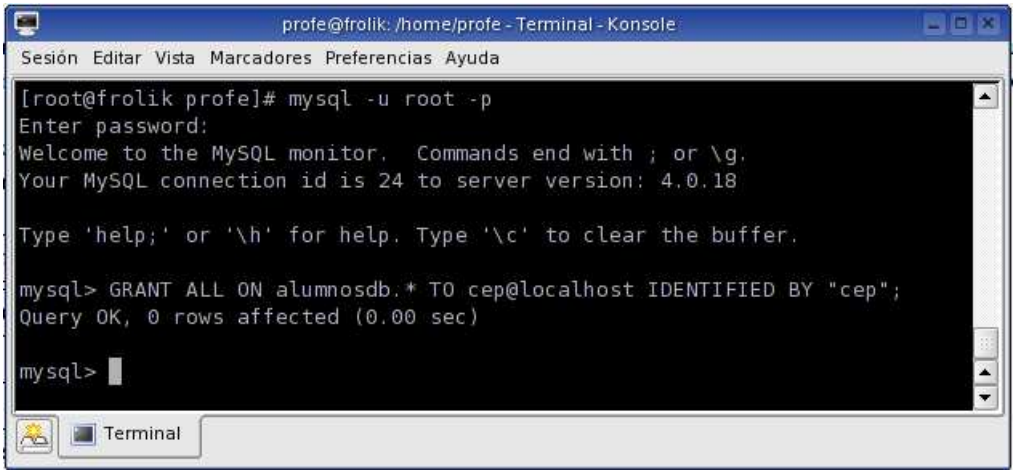
La siguiente tabla muestra los privilegios más importantes disponibles en MySQL:

Denominación	**Descripción**
ALTER	Modificar la estructura de las tablas
CREATE	Crear bases de datos y tablas
DELETE	Borrar registros de las tablas
DROP	Borrar bases de datos y tablas
INDEX	Crear y/o eliminar índices
INSERT	Añadir registros a las tablas
SELECT	Extraer registros de tablas (consultar)
UPDATE	Actualizar los datos de registros existentes en tablas
RELOAD	Recargar las tablas de concesión de privilegios
SHUTDOWN	Parar el servidor
ALL	Sinónimo de ALL PRIVILEGES, conceder todos los permisos
USAGE	Sinónimo de *sin privilegios*

Ejecutaremos la siguiente sentencia SQL desde el cliente de MySQL:

GRANT ALL ON alumnosdb.* TO cep@localhost IDENTIFIED BY “cep”;

[50]Esto es relativo, ya que existe la posibilidad de conectarse como usuario anónimo a la base de datos *test* o a cualquier otra que comience por *test_*. Los usuarios anónimos no poseen privilegios de administración.



Con esta orden estamos asignando (GRANT) todos los privilegios (ALL) sobre (ON) la base de datos *alumnosdb* y sus tablas (alumnosdb.*) para el usuario *cep* cuando se conecta desde el equipo donde está ubicado el servidor MySQL (TO cep@localhost) usando la contraseña *cep* (IDENTIFIED BY "cep").

Salgamos del cliente y volvamos a entrar de la siguiente manera:

mysql -u cep -p

Cuando nos pida el password le proporcionamos *cep*.

Ahora ejecutemos el comando que selecciona una base de datos[51], y vamos a intentar seleccionar la base de datos *mysql* que es la base de datos que contiene información específica del servidor (por ejemplo de los usuarios y los privilegios existentes para cada usuario):

use mysql;

Comprobamos que nos sale un mensaje **ERROR 1044: Access denied for user: 'cep@localhost' to database 'mysql'** . Este mensaje indica que el usuario *cep* cuando se conecta desde la misma máquina del servidor no tiene privilegios suficientes para poder acceder a esta base de datos. Intentémoslo con la base de datos *alumnosdb*.

use alumnosdb;

Aparece el mensaje **Database changed**. Es la manera que tiene MySQL para indicarnos que la base de datos ya la tenemos seleccionada.

Podemos también ejecutar una sentencia para comprobar qué bases de datos contiene el servidor MySQL.

show databases;

Veremos que aparecerán aquellas bases de datos sobre las que el usuario tiene algún tipo de privilegio concedido.

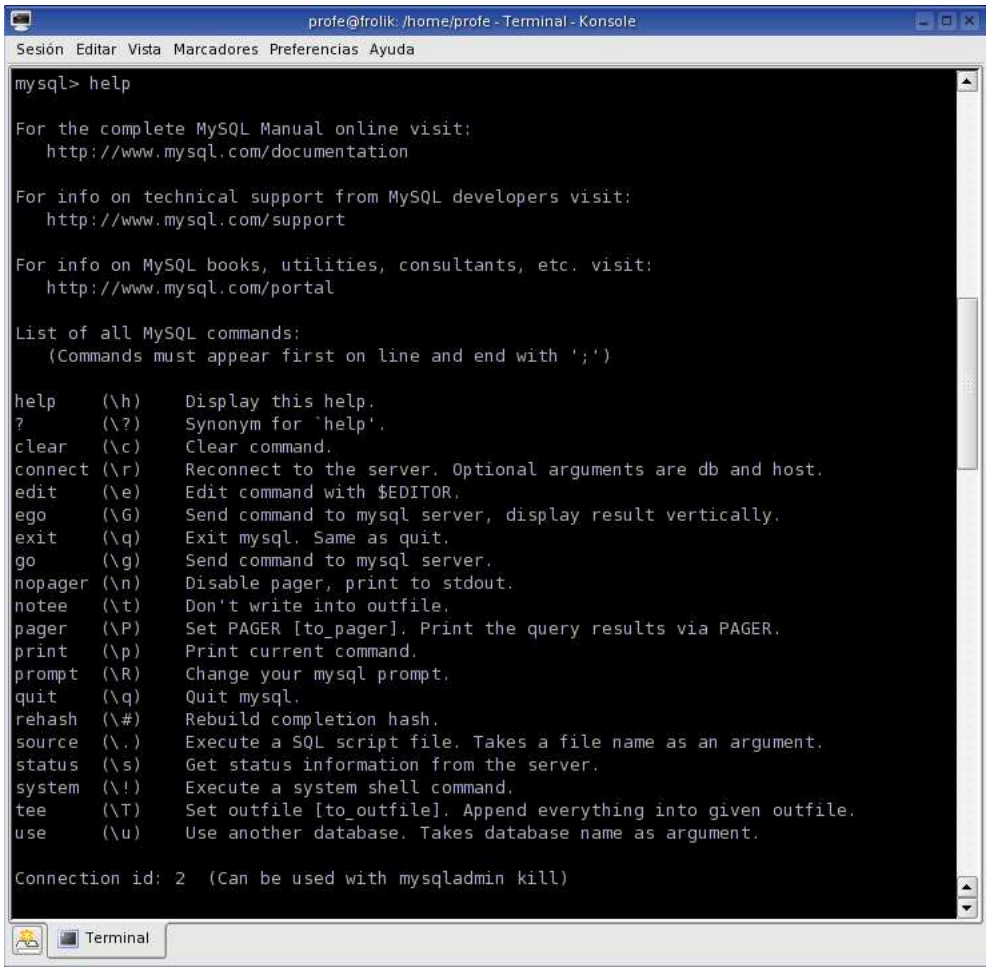
mysql>show databases;
+————+
| *Database* |
+————+
| *alumnosdb* |
+————+
1 row in set (0.01 sec)
mysql>

El usuario *cep@localhost*[52] sólo tiene acceso a *alumnosdb*, aunque sabemos que el servidor posee al menos otras dos bases de datos, *mysql* y *test*.

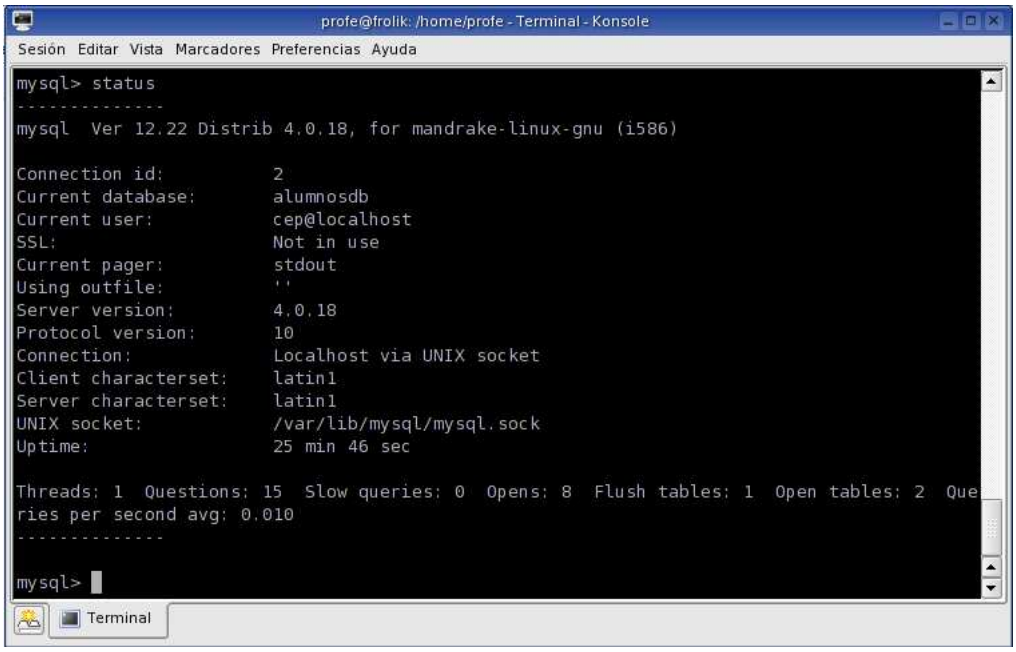
[51] Para poder *usar* una base de datos es preciso seleccionarla antes, es por ello que se precisa esta sentencia.

[52] Es habitual que cuando se hace mención a un usuario y sus privilegios, siempre se acompañe de la ubicación de la máquina desde la que accede. Existe una forma para conceder privilegios a un usuario desde cualquier máquina, la forma es *GRANT ALL ON alumnosdb.* TO cep@ % IDENTIFIED BY "cep"*. El símbolo **%** indica precisamente *desde cualquier máquina.* Pueden incluso convivir *cep@localhost* con *cep@ %*, aplicándose el segundo cuando el host sea diferente de localhost.

Podemos ejecutar el comando *help* para obtener una ayuda sobre los comandos que nos permite ejecutar el cliente, aparte del uso de las sentencias SQL. He aquí la lista de comandos posibles:



La mayoría de estos comandos no los utilizaremos en este curso, sin embargo hay uno verdaderamente útil, ya que muestra información relacionada con el servidor MySQL con el que se está conectado, se trata del comando *status*. El resultado de su ejecución será algo similar a lo que se muestra seguidamente:



Nos indica la base de datos con la que estamos conectados, con qué usuario y de qué versión de MySQL se trata.

Llegados a este punto vamos a proceder a realizar un breve recorrido por las sentencias SQL más importantes y cómo usarlas en el cliente MySQL.

La siguiente tabla muestra una relación de sentencias SQL soportadas por MySQL[53]:

[53] Puede haber variaciones respecto a las últimas versiones disponibles de MySQL.

Tipo	Nombre de la sentencia[54]
Creación, borrado y selección de base de datos	CREATE DATABASE
Creación, borrado y selección de base de datos	DROP DATABASE
Creación, borrado y selección de base de datos	USE
Creación, modificación y borrado de tablas e índices	ALTER TABLE
Creación, modificación y borrado de tablas e índices	CREATE INDEX
Creación, modificación y borrado de tablas e índices	CREATE TABLE
Creación, modificación y borrado de tablas e índices	DROP INDEX
Creación, modificación y borrado de tablas e índices	DROP TABLE
Obtención de información sobre bases de datos, tablas e índices	DESCRIBE
Obtención de información sobre bases de datos, tablas e índices	EXPLAIN
Obtención de información sobre bases de datos, tablas e índices	SHOW
Seleccion de datos almacenados en tablas	SELECT
Modificación de datos almacenados en tablas	DELETE
Modificación de datos almacenados en tablas	INSERT
Modificación de datos almacenados en tablas	LOAD DATA
Modificación de datos almacenados en tablas	OPTIMIZE TABLE
Modificación de datos almacenados en tablas	REPLACE
Modificación de datos almacenados en tablas	UPDATE
Sentencias de administración	FLUSH
Sentencias de administración	GRANT
Sentencias de administración	KILL
Sentencias de administración	REVOKE
Otras	CREATE FUNCTION
Otras	DROP FUNCTION
Otras	LOCK TABLES
Otras	SET
Otras	UNLOCK TABLES

2.7.1. Operaciones de consulta.

Supongamos que ya disponemos de datos en la base de datos *alumnosdb*[55].

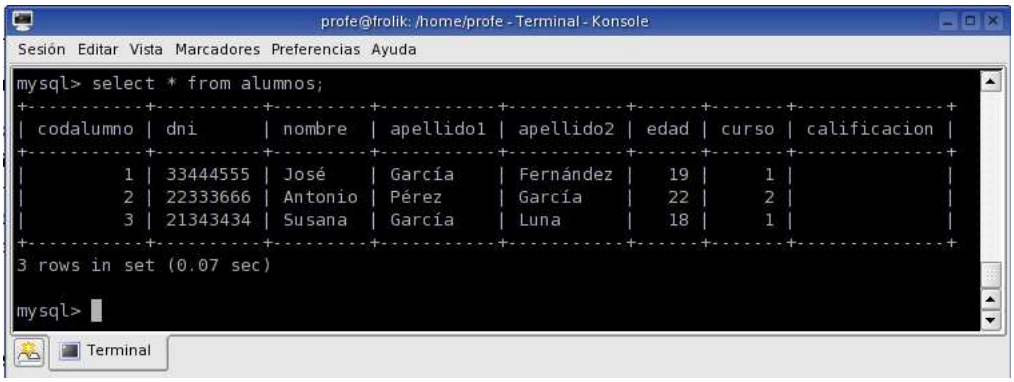
Empezamos por la sentencia más usada y quizás la que posee una sintaxis más compleja, se trata de *SELECT*[56]. Es la sentencia que permite extraer datos de las tablas. Consiste en preguntar al servidor sobre el conjunto de datos que se quieren obtener, de dónde y qué requisitos deben cumplir. La forma más simple es la siguiente:

*SELECT * FROM <TABLA>* donde *<TABLA>* es el nombre de una tabla de la base de datos.

Si ejecutamos esta sentencia desde nuestro cliente, obtenemos:

[55] Lo hemos hecho en el apartado 2.6.3, si aún no se han introducido datos, sería conveniente consultar ese apartado y añadirle los datos que se mencionan.

[56] No abordaremos la sintaxis completa ya que supondría tratar cuestiones que no se comentarán en este curso, y podría originar confusión en el lector. Por otra parte, aunque hasta ahora siempre se han escrito las sentencias SQL en mayúsculas, esto no es más que una convención, pues puede emplearse indistintamente mayúsculas/minúsculas.

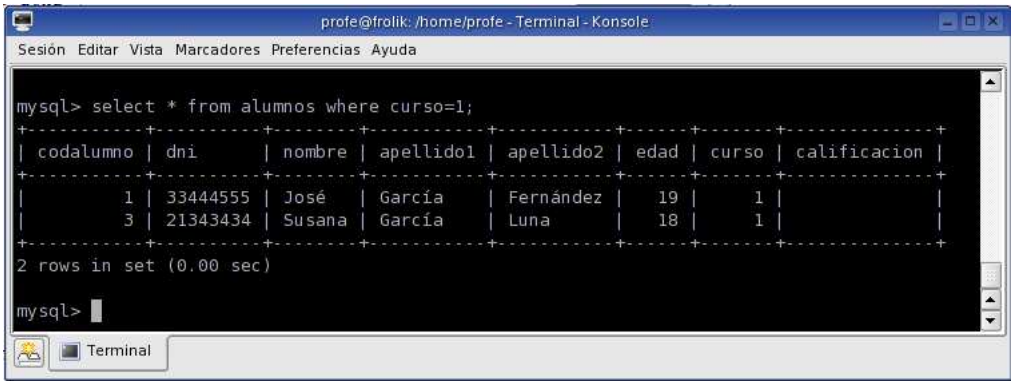


Comprobamos que esta sentencia *selecciona* (SELECT) *todos los campos* (*) *de los registros de la tabla alumnos* (FROM ALUMNOS) y como no se ha establecido ningún requisito de filtrado para esos registros, se obtienen **todos** los registros de esa tabla.

Supongamos ahora que deseamos seleccinar todos los registros de la tabla alumnos que tengan en el campo *curso* el valor 1. Ejecutaremos:

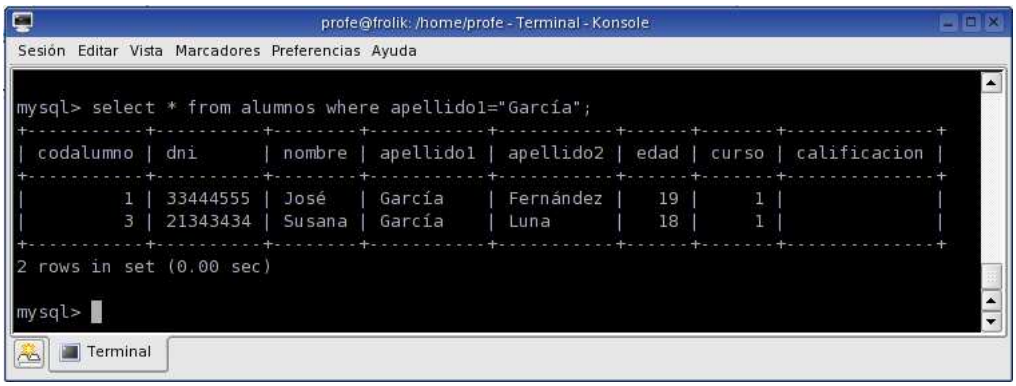
*SELECT * FROM alumnos WHERE curso=1;*

El resultado es el siguiente:



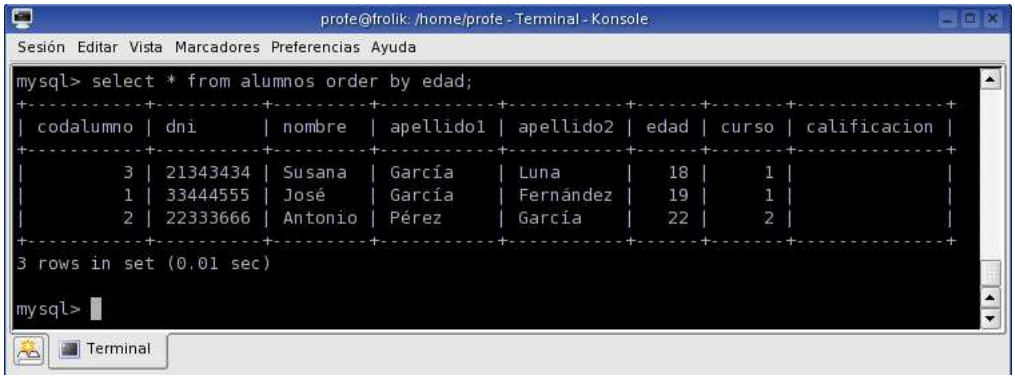
Esta consulta se podría haber interpretado como *Selecciona todos los datos de todos los alumnos que están matriculados en el curso de código 1*. Como observamos, se emplea *WHERE* seguida de una expresión que cumple el papel de condición que deben cumplir los registros que se van a seleccionar. Tanto WHERE como FROM reciben el nombre específico de *cláusula*, de manera que las sentencias SQL se componen de cláusulas.

Planteémosnos seleccionar todos los alumnos cuyo primer apellido sea García. La sentencia será:



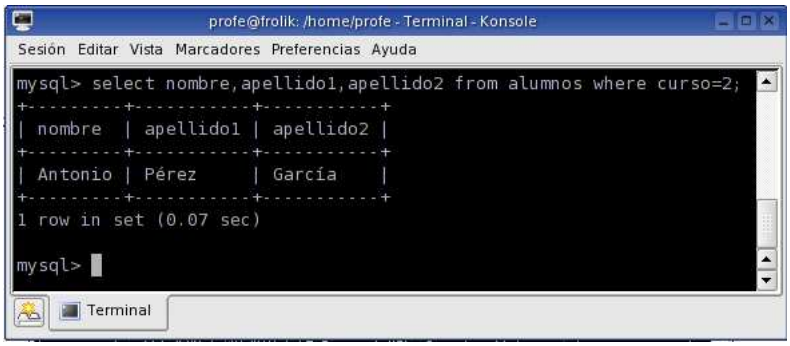
En este caso vemos que la condición de la cláusula *where* debe estar entre comillas dobles[57], ya que se trata de un campo de tipo cadena de caracteres.

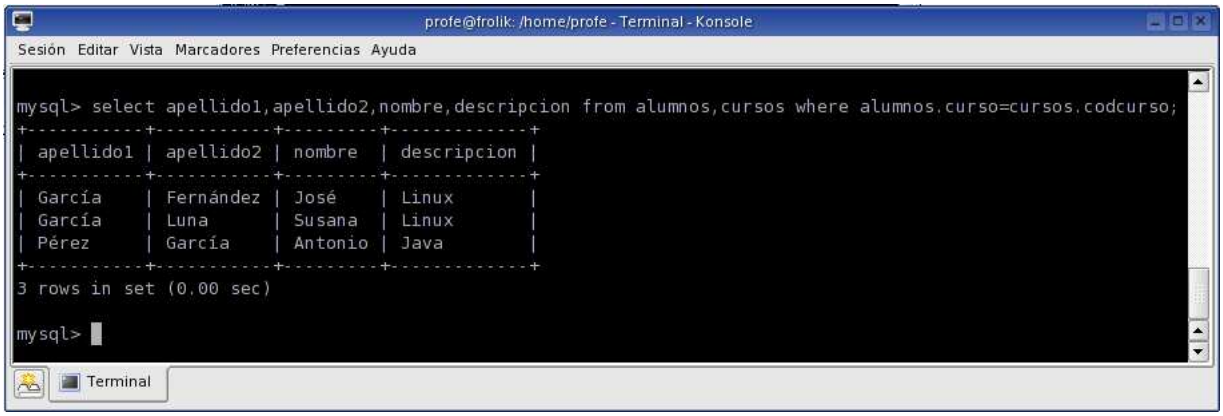
Podemos también sacar los datos ordenados por el campo que nos interese:



Observamos que la cláusula *ORDER BY* ordena ascendentemente los registros seleccionados. Si quisiéramos una ordenación descendente emplearíamos la siguiente sentencia: *SELECT * FROM alumnos ORDER BY edad DESC;*

Las siguientes sentencias SQL realizan diferentes tipos de selección:





En este último ejemplo se obtienen registros formados por datos relacionados en ambas tablas, y para ello se comparan los registros de la tabla alumno con los de la tabla curso mediante un campo común existente en ellas, el campo que almacena el código del curso. Este campo cumple con una condición y es

[57] Pueden emplearse también comillas simples.

que en una tabla es clave principal (tabla curso) y en la otra es clave ajena (tabla alumnos). No es necesario que ambos campos tengan el mismo nombre en cada tabla, basta con que almacenen el mismo tipo de información, ya que la cláusula *where* permitirá relacionarlos mediante sus respectivos nombres.

Resumiendo, la sentencia SELECT devuelve como resultado una nueva tabla obtenida a partir de campos específicos de una o varias tablas de la base de datos, permitiendo establecer los criterios que deben cumplir dichos registros en base a condiciones especificas impuestas a uno o varios campos de los registros de las tablas de la base de datos sobre los que se realiza la selección.

2.7.2. Operaciones de gestión de la base de datos.

El apartado anterior se ha centrado en la extracción de información almacenada en las tablas de la base de datos. En éste, nos centraremos en las operaciones encaminadas a alterar tanto el contenido de las tablas como su estructura.

Insertar registros

Hay varias formas de insertar registros en una base de datos. Puede hacerse mediante la sentencia SQL INSERT INTO, desde un archivo a través de la sentencia LOAD DATA o con la utilidad MySQL *mysqlimport.*

La sentencia INSERT INTO ofrece varias posibilidades en su sintaxis. Una de ellas es la siguiente:

INSERT INTO <nombre-tabla>VALUES (<expr>[,<expr>]...)

Ejemplo:

INSERT INTO curso VALUES (3,'PHP');

Con esta sentencia se inserta un registro nuevo en la tabla *curso*, con código de curso 3 y descripción 'PHP'. No se especifican los nombres de los campos en los que se insertan los valores si se hace en el mismo orden que guardan en la estructura de la tabla. Las cadenas deben introducirse con comillas simples, pero no los valores de tipo numérico.

Puede emplearse también esta sintaxis:

INSERT INTO <nombre-tabla>(<lista-columnas>) VALUES (<expr>[,<expr>]...)

Ejemplo:

INSERT INTO alumnos (codalumno, nombre, apellido1, apellido2) VALUES (4, 'Francisco', 'López', 'De la Hera');

En este caso, se inserta un nuevo registro en la tabla *alumnos* pero sólo se rellenan los campos que almacenan el código de alumno y su nombre completo. Quedan sin rellenar el resto de campos, adoptando valores por defecto si así se especificó al crear la estructura de la tabla.

La sentencia INSERT INTO, como en el caso de SELECT, dispone de una sintaxis más compleja que en este curso no se aborda.

Otra manera de cargar registros en una tabla es mediante la sentencia LOAD DATA. Con esta sentencia se puede insertar un gran volumen de registros de una vez desde un archivo en formato texto plano. Veámoslo con un ejemplo.

Si ejecutamos:

LOAD DATA LOCAL INFILE "alumnos.txt" INTO TABLE alumnos;

Se procederá a la lectura del contenido del archivo *alumnos.txt*, ubicado en el directorio actual del equipo desde donde el cliente está ejecutando la sentencia, y lo envía al servidor para que éste lo cargue en la tabla *alumnos*. Si no se emplea la cláusula LOCAL, el archivo debe situarse en la máquina desde donde se ejecuta el servidor MySQL, precisando además disponer ese usuario del privilegio de acceso FILE.

Por defecto, la sentencia LOAD DATA considera que los valores de los campos (columnas) están en el mismo orden en que se especificaron cuando se creó la tabla, que están separados por tabuladores y que cada linea del archivo de texto finaliza con un retorno de línea. Se pueden leer datos con otros formatos.

Si creamos el siguiente archivo[58]:

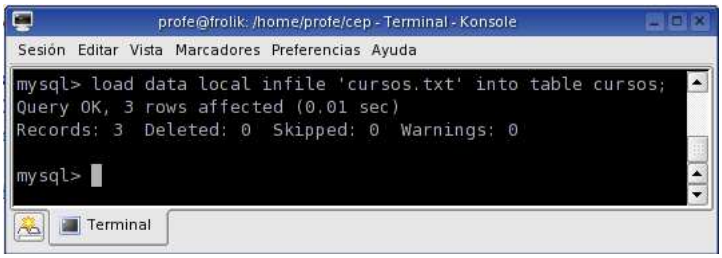
4 JavaScript

5 ASP

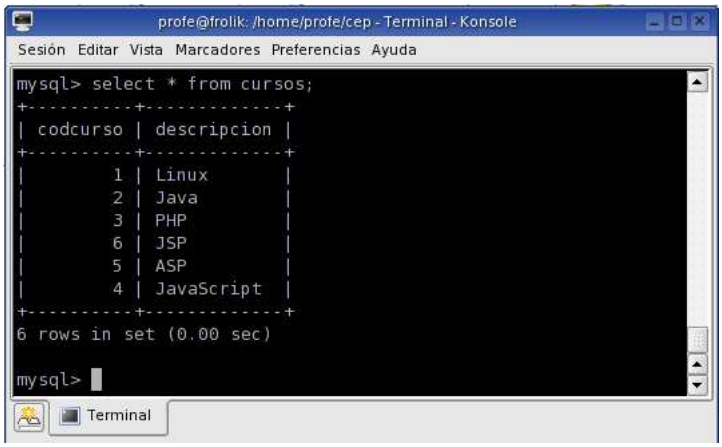
[58] Los valores de ambos campos deben estar separados por un tabulador, y cada línea finalizada en un *intro*.

6 JSP

Y lo almacenamos como *cursos.txt*, podremos ejecutar la siguiente sentencia:



Y al consultar los registros de la tabla cursos, obtendremos:



Finalmente, existe otra forma de insertar registros, en este caso mediante la herramienta *mysqlimport*[59]. Esta herramienta, al ejecutarse, crea una sentencia LOAD DATA partiendo del contenido del archivo de texto que se proporciona como argumento de entrada.

Si ejecutamos:

mysqlimport –local alumnosdb cursos.txt

Se cargarán los datos especificados en el archivo dentro de la tabla *cursos* de la base de datos *alumnosdb*. Sin embargo, este comando generará un error, ya que se precisa especificar el usuario y su contraseña para acceder a la base de datos. volvemos a ejecutar el comando, pero esta vez incluyendo la referencia al usuario:

mysqlimport -u cep -p –local alumnosdb cursos.txt

Esta vez sí se incluyen los registros[60].

Modificar registros

Para actualizar los datos ya existentes en las tablas se emplea la sentencia SQL UPDATE.

Sintaxis abreviada:

UPDATE <nombre-tabla>SET <nombre-columna>=<expresión>[, <nombre-columna>=<expresión>]... [WHERE <expresion-where>]

Modifica el contenido de la tabla donde los registros son seleccionado por la cláusula *WHERE* y los valores de los campos especificados por *<nombre-columna>* son establecidos por su correspondiente *<expresion>*.

[59] Esta herramienta dispone de un amplio conjunto de opciones/parámetros de entrada. Se recomienda ejecutar este comando sin opción alguna, y se mostrará una pequeña referencia de ayuda sobre estas opciones.

[60] Habrá que asegurarse que no se repiten los códigos de los cursos al insertarlos, ya que produciría un error por duplicidad de clave principal.

Supongamos que deseamos modificar la descripción del curso de código 1, cambiando la que tiene por 'Aplicación Web'. Ejecutaremos:

UPDATE cursos SET descripcion='Aplicación Web' WHERE codcurso=1;

También podemos modificar un conjunto de registros. Supongamos que se decide que todos los alumnos estarán matriculados en el mismo curso, por ejemplo el 3. Entonces se ejecuta:

UPDATE alumnos SET curso=3;

Esta sentencia seleccionará todos los registros de la tabla *alumnos* y almacenará en el campo *curso* el código 3.

Borrar registros

El borrado de registros se realiza mediante la sentencia SQL DELETE FROM.

Sintaxis abreviada:

DELETE FROM <nombre-tabla>[WHERE <expresion-where>]

Para borrar todos los alumnos cuyo segundo apellido es García ejecutaremos:

DELETE FROM alumnos WHERE apellido2='García';

Si se omite la cláusula *WHERE* se borran todos los registros de la tabla.

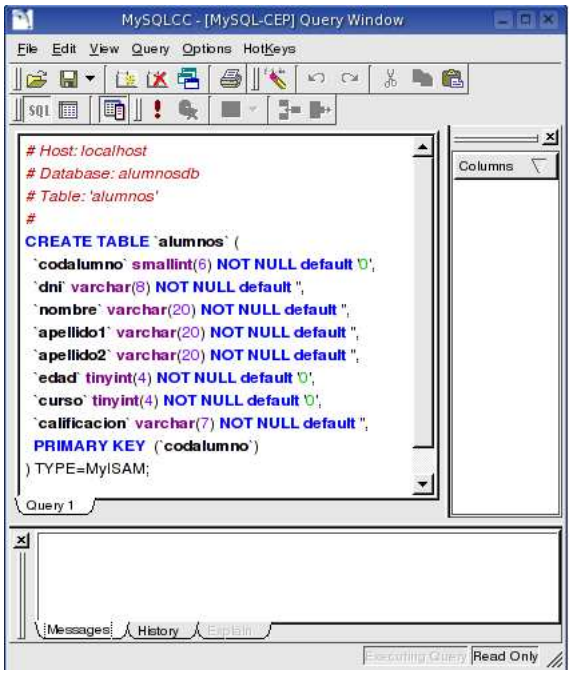
Crear y borrar tablas

Hasta ahora se ha visto cómo alterar el contenido de las tablas, pero existe un conjunto de sentencias SQL que están orientadas a modificar la estructura de las mismas, a crearlas y a borrarlas. En este apartado trataremos las dos últimas.

Para crear una tabla se emplea la sentencia SQL CREATE TABLE.

La sintaxis de esta sentencia es bastante compleja, y muchas de sus posibilidades no se han abordado en el manual. La mejor forma de aprender cómo funciona consiste en crear tablas mediante herramientas gráficas como MySQL Control Center[61]. Con esta herramienta, seleccionando la opción de menú *Action->Tools->Show Create* y tras escoger la tabla que nos interese pulsando en *Execute*, obtendremos la sentencia CREATE TABLE necesaria para crear la tabla tal como está definida hasta ese momento. Por ejemplo, podríamos obtener un resultado similar a éste:

[61] Con phpMyAdmin también puede hacerse, basta con seleccionar la ficha *Exportar*, luego escoger la tabla y pulsar el botón *Continúe*. si hemos dejado marcada la casilla de *Datos*, se incluirán también las sentencias INSERT necesarias para añadir los datos ya existentes en esa tabla. Es necesario advertir que esta ficha se emplea precisamente para crear archivos con las sentencias SQL necesarias para poder crear la tabla y/o insertar datos en ella.



Para borrar una tabla (datos y estructura) se emplea la sentencia SQL DROP TABLE.

Sintaxis:

DROP TABLE [IF EXISTS] <nombre-tabla>[,<nombre-tabla>]...

Por ejemplo, creamos una tabla denominada cursos2:

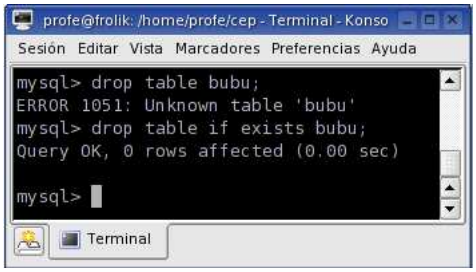
CREATE TABLE curso2 (codcurso SMALLINT NOT NULL, nombrecurso VARCHAR(30), PRIMARY KEY(codalumno));

Y ahora la borramos:

DROP TABLE IF EXISTS curso2;

Al indicar *IF EXISTS* se impide que se genere un error si la tabla especificada no existiera. Si no se emplea esa cláusula, el servidor genera un error si la tabla no existe en la base de datos en uso.

He aquí un ejemplo de ello:



Modificar la estructura de una tabla

Para modificar la estructura de una tabla ya creada se emplea la sentencia SQL ALTER TABLE.

La sintaxis es:

ALTER TABLE <nombre-tabla><lista-acciones>

Evidentemente la complejidad a esta sentencia la aporta la *<lista-acciones>* que puede ser muy variada.

Veremos un par de ejemplos, simplemente con carácter ilustrativo.

Para añadir un nuevo campo a la tabla cursos, por ejemplo, estableciendo la fecha de comienzo del mismo:

ALTER TABLE cursos ADD fechainicio DATE;

Añade el campo *fechainicio* detrás del último campo de la estructura de la tabla *cursos*, siendo de tipo *DATE*, que permite trabajar con fechas[62].

Si visualizamos el contenido de la tabla *cursos*, veremos que se ha añadido una nueva columna. Ahora todos los registros para la columna *fechainicio* muestran el valor NULL. NULL se emplea para indicar que no se ha introducido ningún valor para ese campo, y que no se había determinado un valor por defecto. La siguiente sentencia, rellenará el campo *fechainicio* con el valor 2005-04-01[63] para todos los registros.

UPDATE cursos SET fechainicio='2005-04-01';

Para renombrar la tabla *cursos* por *cursosonline*:

ALTER TABLE cursos RENAME AS cursosonline;

Para cambiar el tipo SMALLINT de la columna *codalumno* de la tabla *alumnos* a INT:

ALTER TABLE alumnos MODIFY codalumno INT;

Crear un archivo de script con las sentencias necesarias para crear una base de datos con registros incluidosw

Se emplea la herramienta *mysqldump*. Como se avanzó en un apartado anterior, esta herramienta posibilita la copia de seguridad de una base de datos o su copia hacia un servidor diferente, y además permite obtener archivos que contienen las sentencias necesarias para crear las tablas y/o insertar los datos en ellas. Por defecto, si no se especifica tabla alguna en esta herramienta, devolverá una sentencia CREATE TABLE por cada tabla de la base de datos, seguida por el conjunto de sentencias INSERT INTO necesarias que permitirían rellenarla de nuevo. Esta herramienta acepta la opción *–tab* que creará dos archivos por cada tabla, uno para la sentencia SQL CREATE TABLE, y otro que contendrá los datos en formato plano (similar al comentado en el subapartado *Insertar registros* del apartado 2.7.2).

Como ejemplos, primero obtendremos los scripts necesarios para volver a crear la base de datos *alumnosdb* por completo.

mysqldump -u cep -p alumnosdb >alumnosdb_copia

El contenido del archivo *alumnosdb_copia* será el siguiente:

```
————
– MySQL dump 9.10
–
– Host: localhost Database: alumnosdb
– ————————————————————
– Server version 4.0.18
–
– Table structure for table 'alumnos'
–
CREATE TABLE alumnos (
codalumno int(11) NOT NULL default '0',
dni varchar(8) NOT NULL default '',
nombre varchar(20) NOT NULL default '',
apellido1 varchar(20) NOT NULL default '',
apellido2 varchar(20) NOT NULL default '',
edad tinyint(4) NOT NULL default '0',
curso tinyint(4) NOT NULL default '0',
calificacion varchar(7) NOT NULL default '',
PRIMARY KEY (codalumno)
) TYPE=MyISAM;
–
– Dumping data for table 'alumnos'
```

[62] Las fechas pueden tener alguno de los siguientes formatos: 'AAAA-MM-DD', 'AA-MM-DD', 'AAAAMMDD', 'AAMMDD', AAAAMMDD o AAMMDD. Admiten por tanto que se introduzcan como cadenas (los cuatro primeros formatos) o como números (los dos siguientes).

[63] Se admite también escribir 2005-4-1.

–
INSERT INTO alumnos VALUES (1,'33444555','José','García','Fernández',19,3,");
INSERT INTO alumnos VALUES (3,'21343434','Susana','García','Luna',18,3,");
INSERT INTO alumnos VALUES (4,'','Francisco','López','De la Hera',0,3,");
–
– Table structure for table 'cursosonline'
–
CREATE TABLE cursosonline (
codcurso tinyint(4) NOT NULL default '0',
descripcion text NOT NULL,
fechainicio date default NULL,
PRIMARY KEY (codcurso)
) TYPE=MyISAM;
–
– Dumping data for table 'cursosonline'
–
INSERT INTO cursosonline VALUES (1,'Aplicación Web','2005-04-01');
INSERT INTO cursosonline VALUES (2,'Java','2005-04-01');
INSERT INTO cursosonline VALUES (3,'PHP','2005-04-01');
INSERT INTO cursosonline VALUES (6,'JSP','2005-04-01');
INSERT INTO cursosonline VALUES (5,'ASP','2005-04-01');
INSERT INTO cursosonline VALUES (4,'JavaScript','2005-04-01');
————

Como se observa, lo único que se precisa para poder *recrear* la base de datos es utilizar la sentencia *CREATE DATABASE*, y posteriormente emplear una de las vías que ofrece MySQL para procesar este archivo.

Se puede crear un archivo con las sentencias necesarias para crear una tabla, en lugar de la base de datos completa:

mysqldump -u cep -p alumnosdb cursosonline >alumnosdb.cursosonline.copia

El contenido del archivo *alumnosdb.cursosonline.copia* será similar al siguiente:

————-
– MySQL dump 9.10
–
– Host: localhost Database: alumnosdb
– ——————————————
– Server version 4.0.18
–
– Table structure for table 'cursosonline'
–
CREATE TABLE cursosonline (
codcurso tinyint(4) NOT NULL default '0',
descripcion text NOT NULL,
fechainicio date default NULL,
PRIMARY KEY (codcurso)
) TYPE=MyISAM;
–
– Dumping data for table 'cursosonline'
–
INSERT INTO cursosonline VALUES (1,'Aplicación Web','2005-04-01');
INSERT INTO cursosonline VALUES (2,'Java','2005-04-01');
INSERT INTO cursosonline VALUES (3,'PHP','2005-04-01');
INSERT INTO cursosonline VALUES (6,'JSP','2005-04-01');
INSERT INTO cursosonline VALUES (5,'ASP','2005-04-01');
INSERT INTO cursosonline VALUES (4,'JavaScript','2005-04-01');

———-

Podemos ejecutar el comando con la opción *–tab* pero antes debemos determinar en qué directorio se van a almacenar los archivos creados por la herramienta. Por ejemplo, creamos un directorio *copia* en /home/alumno. Ubicados en /home/alumno, ejecutamos el comando de la siguiente manera:

mysqldump -u cep -p –tab=copia alumnosdb cursosonline

Pero el servidor informa de que el usuario no posee privilegios suficientes. Esto es debido a que no posee el privilegio FILE. Podemos recurrir al usuario *root* para hacerlo, o bien asignar este privilegio al usuario *cep.* Optamos por la primera opción, que además es la más recomendable:

mysqldump -u root -p –tab=copia alumnosdb cursosonline

En el directorio *copia* localizamos dos archivos:

———-cursosonline.sql

– MySQL dump 9.10

–

– Host: localhost Database: alumnosdb

– ————————————————

– Server version 4.0.18

–

– Table structure for table ‘cursosonline‘

–

CREATE TABLE cursosonline (
codcurso tinyint(4) NOT NULL default ’0’,
descripcion text NOT NULL,
fechainicio date default NULL,
PRIMARY KEY (codcurso)
) TYPE=MyISAM;

———-

———-cursosonline.txt

1 Aplicación Web 2005-04-01
2 Java 2005-04-01
3 PHP 2005-04-01
6 JSP 2005-04-01
5 ASP 2005-04-01
4 JavaScript 2005-04-01

———-

Ahora tendremos una mayor flexibilidad, ya que el proceso de creación de la tabla puede abordarse separadamente del proceso de rellenado de datos.

Crear una base de datos partiendo de un archivo de script

Disponiendo de los archivos generados por *mysqldump* podremos crear tablas y rellenarlas, bien en el mismo servidor MySQL o en cualquier otro. Además, servirá para aprender un poco más sobre el lenguaje SQL.

Como ejemplo creamos una base de datos denominada *alumnosdb2*, y procederemos a crear las tablas y rellenarlas posteriormente a partir de los archivos generados en el apartado anterior. Emplearemos *alumnosdb.cursosonline.copia*.

Una manera es emplear la propia herramienta cliente *mysql*.

Primero creamos desde *mysql* la base de datos *alumnosdb2*.

Ahora podemos abandonar el cliente, y ejecutar desde consola:

mysql -u cep -p alumnosdb2 <alumnosdb.cursosonline.copia

Habremos creado y rellenado las tablas de *alumnosdb2*.

Si optamos por la opción *–tab* al usar *mysqldump*, podremos emplear el archivo .txt en el cliente *mysql* con la sentencia *LOAD DATA*.

Php - Mysql

2.8. ¿Qué es lo que pretendemos?

Hasta ahora hemos visto como crear páginas web dinámicas con Php, también hemos aprendido a trabajar con una base de datos típica del mundo Linux como Mysql. Demos un paso más y veamos como podemos acceder a nuestra base de datos desde una página web.

El proceso es sencillo, simplemente se abrirá la base de datos, se le mandarán peticiones SQL, y se tratarán los datos para mostrarlos o modificar la base de datos.

2.9. Conectar con el servidor de bases de datos.

Para poder acceder a las bases de datos tenemos que conectar con el servidor, se utiliza la siguiente instrucción:

```
$c = mysql_connect(servidor, login, password)
```

Siguiendo con nuestro ejemplo:

Una vez finalizadas las operaciones es necesario cerrar la conexión:

```
mysql_close($c)
```

donde $c es el nombre de la variable en la que se recogió el indentificador del enlace en el momento de la apertura.

2.10. Trabajar con bases de datos.

2.10.1. Listar las bases de datos.

Antes de crear y/o borrar una base de datos puede ser conveniente y útil comprobar si ya existe.

PHP dispone de herramientas para conocer el número de bases de datos existentes en el servidor, así como sus nombres. Todas ellas requieren que se haya establecido una conexión con el servidor.

```
$p = mysql_list_dbs($c)
```

La variable $p es un nuevo identificador imprescindible y previo a la determinación del número y los nombres de las bases de datos existentes en el enlace abierto (identificado por $c).

```
$n = mysql_num_rows($p)
```

Esta función devuelve el número de bases de datos existentes en el servidor.

Utiliza como parámetro ($p) el resultado obtenido mediante la función anterior.

Ese número puede recogerse en una variable (en este caso $n).

```
mysql_db_name($p, i)
```

Esta nueva función devuelve el nombre de una de las bases de datos, identificada por un número i que debe pertenecer al intervalo [0,$n).

Fíjate que i tiene que ser i<$n porque si, por ejemplo, $n=5 los cinco valores posibles de i serían: 0,1,2,3 y 4.

Una lista completa de todas las bases de datos existentes en el servidor podría hacerse mediante el siguiente proceso:

- Abrir la conexión.
- Invocar a mysql_list_dbs.
- Contar el número de bases de datos con mysql_num_rows
- Insertar un bucle:
  ```
  for ($i=0;$i<$num,$i++)
  ```
- Presentar la lista de nombres mediante un bucle que lea los diferentes valores de $i en:
  ```
  mysql_db_name($p,$i)
  ```

A continuación se muestra un ejemplo completo:

```
<?
if($c=mysql_pconnect ("localhost","root","")){
  echo "<h2>Conexión establecida con el servidor</h2><br>";

  # recoge en una nueva variable que hemos llamado $p
  # un nuevo identificador, $p
  $p=mysql_list_dbs($c);

  # utilizando ese nuevo identificador podemos extraer el número
  $numero=mysql_num_rows($p);
  echo "Hay ",$numero, " bases de datos en el servidor<br>" ;

  #este bucle (desde cero hasta el número nos irá listando los nombres correspondientes a
  #esos numeros de orden
  for ($i=0;$i<$numero;$i++) {
    echo mysql_db_name($p, $i),"<br>";
  }

  if(mysql_close($c)){
    echo "<h2>Conexión cerrada con exito</h2><br>";
  }else{
    echo "<h2>No se ha cerrado la conexión</h2>";
  };
}else{
  echo "<h2>NO HA SIDO POSIBLE ESTABLECER LA CONEXIÓN</h2>";
}
?>
```

2.10.2. Crear una base de datos.

La creación de una base de datos también requiere una conexión previa y utiliza la siguiente sintaxis:
mysql_query ("CREATE DATABASE nom")
donde nom es el nombre de la nueva base de datos.
Esta función devuelve TRUE si la base de datos es creada, y FALSE si no es posible hacerlo.
Si intentamos crear una base de datos con un nombre ya existente la función nos devolverá FALSE.
Realizamos un script que cree una base de datos de nombre otraBase;

```
<?
if($conexion=mysql_connect ("localhost","root","")){
  echo "<h2>Conexión establecida con el servidor</h2><br>";
  if(mysql_query("CREATE DATABASE otraBase")){
    echo "<h2>Base de datos creada</h2><br>";
  }else{
    echo "<h2>No ha sido posible crear la base de datos</h2><br>";
  };
  if(mysql_close($conexion)){
    echo "<h2>Conexión cerrada con exito</h2><br>";
    echo "El identificador de conexion es:",$conexion;
  }else{
    echo "<h2>No se ha cerrado la conexión</h2>";
  };
}else{
  echo "<h2>NO HA SIDO POSIBLE ESTABLECER LA CONEXIÓN</h2>";
}
?>
```

2.10.3. Borrar una base de datos

Para borrar una base de datos se requiere el uso de la siguiente función PHP:

```
mysql_query ("DROP DATABASE nom")
```

donde nom es el nombre de la base de datos y debiendo ponerse toda la cadena del paréntesis entre comillas.

Esta función devuelve TRUE cuando se ejecuta con éxito, y FALSE en el caso contrario.

Igual que ocurría al tratar de crearla, si intentamos borrar una base de datos inexistente la función nos devolverá FALSE.

Este es el código de un script que puede borrar la base creada anteriormente:

```
<?
if($c=mysql_connect ("localhost","root","")){
  echo "<h2>Conexión establecida con el servidor</h2><br>";
  if(mysql_query ("DROP DATABASE otraBase",$c)){
    echo "<h2>Base de datos borrada</h2><br>";
  }else{
    echo "<h2>No ha sido posible BORRAR la base de datos</h2><br>";
  };
  if(mysql_close($c)){
    echo "<h2>Conexión cerrada con exito</h2><br>";
    echo "El identificador de conexion es:",$conexion;
  }else{
    echo "<h2>No se ha cerrado la conexión</h2>";
  };
}else{
  echo "<h2>NO HA SIDO POSIBLE ESTABLECER LA CONEXIÓN</h2>";
}
?>
```

2.11. Trabajar con tablas.

2.11.1. Creación de tablas.

Las tablas son elementos de las base de datos. Por esa razón nos resultará imposible crear una tabla sin tener creada y seleccionada una base de datos.

Es por eso que para la creación de una tabla se necesitan los siguientes requisitos:

- Tener abierta una conexión con el servidor MySQL.
- Tener seleccionada una base de datos.

La conexión con el servidor ha de ser establecida antes de cualquier otra intervención relacionada con accesos a bases de datos y tablas.

Dado que podemos manejar bases de datos distintas es preciso decir a MySQL con qué base queremos trabajar.

```
mysql_select_db("n", $c);
```

donde n es el nombre de la base de datos (puede ser una cadena entrecomillada o el nombre de una variable previa que contenga ese nombre). En este último caso, como es habitual, el nombre de la variable no llevaría comillas.

El segundo parámetro $c es el identificador de conexión. Es decir, la variable creada al establecer la conexión con MySQL.

Este valor debe insertarse siempre. La razón es que MySQL permite mantener abiertas, de forma simultánea, varias conexiones (podríamos manejar más de un servidor de bases de datos) y en esas condiciones sería necesaria una conexión distinta para cada servidor.

La creación de tablas MySQL requiere una de estas dos sentencias:

```
CREATE TABLE IF NOT EXISTS tabla (campo1, campo2,... )
```

```
CREATE TABLE tabla (campo1, campo2,... )
```

La única diferencia entre ambas opciones es que la segunda daría un error si tratáramos de crear una tabla preexistente mientras que la primera no da ese mensaje de error.

Aunque no lo hemos indicado, CREATE DATABASE también permite esta sintaxis alternativa.

2.11.1.1. Definición de campos en una tabla MySQL.

Cada uno de los campos que vayamos a crear en una tabla requiere una definición que debe contener lo siguiente:

- **nombre del campo**

Es una palabra cualquiera, distinta para cada campo de la tabla y que normalmente suele elegirse aludiendo al contenido. Por ejemplo, fec_nac, nom_perro, etc.

No va entre comillas nunca y MySQL diferencia mayúsculas/minúsculas.

Para utilizar como nombres de campo palabras reservadas -por ejemplo, create- del lenguaje MySQL debemos escribirla entre ‘ ‘. Observa que no son comillas sino acentos graves. Lo más aconsejable es evitar esta situación.

- **tipo y dimensiones**

Los tipos de campos tienen que ajustarse a uno de los soportados por MYSQL. Se escriben a continuación del nombre sin otra separación que un espacio y requieren la sintaxis -estricta- correspondiente a cada tipo.

- **flags del campo** (son opcionales)

Puede utilizarse cualquiera de los permitidos para cada tipo de campo -puedes verlos encerrados entre corchetes- al lado de cada tipo de campo.

Ejemplo de creación de una tabla:

```
<?
# nos conectamos con el servidor recogiendo en $c el identificador de conexión
$c=mysql_connect ("localhost","root","");

# seleccionamos una base de datos existente de lo contrario nos daría un error
# pondremos como nombre ejemplos nuestra base de datos creada en la página anterior
# y usaremos $c, importante no olvidarlo
mysql_select_db ("ejemplos", $c);

/* ahora ya estamos en condiciones de crear la tabla podríamos escribir ya la instrucción
mysql_query y meter detro la sentencia MySQL pero, por razones de comodidad crearemos
antes una variable que recoja toda la sentencia y será luego cuando la ejecutemos.
Definiremos una varable llamada $crear e iremos añadiendo cosas */

# la primera parte de la instrucción es esta, espacio final incluido
$crear="CREATE TABLE IF NOT EXISTS ";

# añadiremos el nombre de la tabla que será ejemplo1
# fíjate en el punto (concatenador de cadenas) que permite ir añadiendo a la cadena anterior
$crear .="ejemplo1 ";

#ahora pongamos el paréntesis (con un espacio delante)
#aunque el espacio también podría detrás de ejemplo1
$crear .="( ";

# insertemos el primer campo y llamemoslo num1
# hagamoslo de tipo TINYINT sabiendo que solo permitira valores numéricos comprendidos entre
# -128 y 127
```

```
$crear .="num1 TINYINT , ";

# LOS CAMPOS SE SEPARAN CON COMAS por eso la hemos incluido al final de la instrucción anterior
# ahora num2 del mismo tipo con dimensión 3 y el flag UNSIGNED Y ZEROFILL que:
# cambiará los límites de valores al intervalo 0 - 255, y rellenará con ceros por la izquierda
# en el caso de que el número de cifras significativas sea menor de 3.
# Fijate que los flags van separado unicamente por espacios
$crear .="num2 TINYINT (3) UNSIGNED ZEROFILL, ";

# en num3 identico al anterior añadiremos un valor por defecto de manera que cuando se añadan
# registros a la tabla se escriba automaticamente el valor 13 en el caso de que no le asignemos
# ningun valor a ese campo. Por ser numérico 13 no va entre comillas
$crear .="num3 TINYINT (7) UNSIGNED ZEROFILL DEFAULT 13, ";

# ahora un número decimal num4 tipo REAL con 8 digitos en total de los cuales tres serán
# decimales y también rellenaremos con ceros.Pondremos como valor por defecto 3.14
$crear .="num4 REAL (8,3) ZEROFILL DEFAULT 3.14, ";

# añadamos una fecha
$crear .="fecha DATE, ";

# una cadena con un limite de 32 carácter con BINARY para que diferencie Pepe de PEPE
$crear .="cadena VARCHAR(32) BINARY, ";

# un ultimo campo -opcion- del tipo ENUM que solo admita como valores SI, NO, QUIZA
# fijate en las comillas y en el parentesis ¡¡cuidado...!! aqui no ponemos coma al final
# es el último campo que vamos a insertar y no necesita ser separado. Si la pones dará un ERROR
$crear .="opcion ENUM('Si','No','Quiza') ";

# solo nos falta añadir el paréntesis conteniendo toda la instrucción
$crear .=")";

# tenemos completa la sentencia MySQL solo falta ejecutarla mediante mysql_query
# ya que la conexión está abierta y la base de datos ya está seleccionada
# pongamos un condicional de comprobación
if(mysql_query($crear,$c)){
  print "Se ha creado la base de datos<br>";
  print "La sentencia MySQL podríamos haberla escrito asi:<br>";
  print "mysql_query(".""".$crear.""" , $c);";
}else{
  print "Se ha producido un error al crear la tabla";
}
?>
```

2.11.2. Ver la estructura de una tabla.

La sentencia MySQL que permiten visualizar la estructura de una tabla es la siguiente:

```
SHOW FIELDS from nombre de la tabla
```

La sentencia SHOW FIELDS - como prácticamente ocurre con todas las sentencias MySQL- no devuelve los resultados en formato legible.

Los resultados devueltos por estas sentencias requieren ser convertidos a un formato que sea interpretable por PHP.

Esa traducción se realiza de la siguiente forma:

El resultado devuelto por MySQL a través de una llamada mysql_query() es recogido en una variable, de la forma siguiente:

```
$r=mysql_query("SHOW FIELDS from nombre", $c)
```

El resultado recogido en la variable $r, está estructurado en líneas y la función:

```
$t =mysql_fetch_row ($r)
```

recoge en una variable ($t) el contenido de la primera línea y coloca su puntero interno al comienzo de la línea siguiente. Por esta razón la lectura completa del contenido de la variable $r requiere llamadas sucesivas a mysql_fetch_row hasta que haya sido leída la última línea del resultado de la llamada a MySQL.

La variable $t tiene estructura de array escalar siendo cero el primero de sus índices.

Cuando el puntero interno de mysql_fetch_row() alcance el final de la última línea del resultado devolverá FALSE.

Por esa razón, la visualización de los resultados de una sentencia MySQL suele requerir dos bucles. Este es el esquema de la lectura:

```
$r=mysql_query("SHOW FIELDS from nombre",$c);
while($r=mysql_fech_row($r){
  foreach ($r as $valor){
    print $valor;
  }
}
```

Veamos un ejemplo completo:

```
<?
# asignamos a una variable el nombre de la base de datos
$base="ejemplos";

# esta otra recoge el nombre de la tabla
$tabla="ejemplo1";

# estableceremos la conexión con el servidor
$c=mysql_connect ("localhost","root","");

# seleccionamos la base de datos
mysql_select_db ($base, $c);

#ejecutamos mysql_query llamando a la sentencia SHOW FIELDS
$resultado=mysql_query( "SHOW FIELDS from $tabla",$c);

# determinamos el número campos de la tabla
$numero=mysql_num_rows($resultado);

# Presentamos ese valor númerico
print "La tabla tiene $numero campos<br>";

# ejecutamos los bucles comentados
while($v=mysql_fetch_row ($resultado)){
  foreach($v as $valor) {
    echo $valor,"<br>";
  }
}
?>
```

Existe una función alternativa que mejora las prestaciones de la anterior. Se trata de:

```
$t =mysql_fetch_array($r)
```

es idéntica en cuanto a su funcionamiento y, además, añade una nueva posibilidad ya que los arrays que devuelve pueden ser leídos como escalares y también como asociativos. En este último caso incorporan como índice el nombre del campo de la tabla del que se han extraído cada resultado.

Respecto a la forma en la que sea asignan los índices a los array obtenidos mediante consultas a tablas, puedes verla en este ejemplo.

```
$resultado=mysql_query( "SHOW FIELDS from $tabla",$c);
print("<BR>Los resultados con mysql_fech_array<br>");
while($v=mysql_fetch_array($resultado)){
  foreach($v as $clave=>$valor) {
    print ("El indice es: ".$clave." y el valor es: ".$valor."<br>");
```

```
    }
  }
```

2.11.2.1. Otras funciones informativas

mysql_num_fields($res)

Está función -en la que $res es el identificador de resultado - devuelve el número de campos de la tabla.

mysql_num_rows($res)

Devuelve el número de registros que contiene la tabla. Si la tabla no contiene datos devolverá CERO.

mysql_field_table($res,0)

Devuelve el nombre de la tabla. Observa que se pasa con índice 0 ya que esta información parece ser la primera que aparece en la tabla.

mysql_field_type($rs,$i)

Nos devuelve el tipo de campo correspondiente a la posición en la tabla señalada por el índice $i. Dado que la información de primer campo está en el índice 0, el último valor válido de $i será igual al número de campos menos uno.

mysql_field_flags($res,$i)

Nos devuelve los flags del campo correspondientse a la posición en la tabla señalada por el índice $i. Se comporta igual que la anterior en lo relativo a los índices.

mysql_field_len($res,$i)

Nos devuelve la longitud del campo correspondiente a la posición en la tabla señalada por el índice $i. Igual que las anteriores en lo relativo a los índices.

mysql_field_name($rs,$i)

Nos devuelve el nombre del campo correspondiente a la posición en la tabla señalada por el índice $i. En lo relativo a los índices su comportamiento es idéntico a las anteriores.

2.11.3. Borrar una tabla.

Las sentencias MySQL que permite borrar una tabla son las siguientes:

```
DROP TABLE IF EXISTS from nombre de la tabla
DROP TABLE from nombre de la tabla
```

la diferencia entre ambas radica en que usando la primera no se generaría ningún error en el caso de que tratáramos de borrar una tabla inexistente.

Aquí tienes el código fuente de un ejemplo:

```
<?
# insertamos un nombre de tabla imaginaria para evitar riesgos de ejecución
$base="ejemplos";
$tabla="prueba1";

$borrar="DROP TABLE ";
$borrar .=$tabla;

$conexion=mysql_connect ("localhost","pepe","pepa");
mysql_select_db ($base, $conexion);
if(mysql_query ($borrar ,$conexion)) {
  echo "<h2>Tabla $tabla borrada con EXITO</h2><br>";
}else{
  echo "<h2>La tabla $tabla NO HA PODIDO BORRARSE</h2><br>";
};
mysql_close($conexion);
?>
```

2.11.4. Borrar uno de los campos de una tabla.

La sentencia MySQL que permite borrar uno de los campos de una tabla es la siguiente:
`ALTER TABLE nombre de la tabla DROP nombre del campo`
Es posible modificar la estructura de una tabla -en este caso borrar un campo- siguiendo un procedimiento similar a los anteriores.

La única diferencia estriba en utilizar la sentencia MySQL adecuada.

Resulta obvio que el campo debe existir para que pueda ser borrado y si no exitiera, es obvio también que se produciría un error.

Aquí tienes el código fuente de un script que borra uno de los campos de una tabla:

```
<?
# nombre de la base de datos
$base="ejemplos";

# nombre de la tabla en la que pretenderemos borrar el campo
$tabla="tabla1";

# nombre del campo a borrar
$campo="num1";

# definicion de una variable que contiene la sentencia MySQL de borrado
$borrar="ALTER TABLE ";
$borrar.=$tabla;
$borrar.=" DROP $campo";

$conexion=mysql_connect ("localhost","root","");
mysql_select_db ($base, $conexion);
if(mysql_query ($borrar ,$conexion)) {
  echo "<h2>A la tabla $tabla se le ha BORRADO el campo $campo</h2><br>";
}else{
  echo "<h2>No ha podido BORRAR</h2><br>";
};
mysql_close($conexion);
?>
```

2.11.5. Añadir un nuevo campo a una tabla.

Las sentencia MySQL que permite añadir un nuevo campo a una tabla es la siguiente:
`ALTER TABLE nombre de la tabla ADD nombre del campo tipo [flags]`
El procedimiento es similar a los casos anteriores utilizando esta nueva sentencia MySQL.

La sintaxis es similar a la de la creación de tablas: el nombre del campo debe ir seguido del tipo sin otra separación que el espacio

Aquí tienes el código fuente de un script que añade uno de los campos de una tabla:

```
<?
$base="ejemplos";
$tabla="tabla1";

$anadir="ALTER TABLE ";
$anadir.=$tabla;
$anadir.=" ADD nuevocampo TINYINT(12) ";

$conexion=mysql_connect ("localhost","root","");
mysql_select_db ($base, $conexion);
if(mysql_query ($anadir ,$conexion)) {
  echo "<h2>A la tabla $tabla se le ha añadido un campo</h2><br>";
}else{
  echo "<h2>No ha podido añadir</h2><br>";
```

```
};
mysql_close($conexion);
?>
```

2.12. Trabajar con registros.

2.12.1. Añadir registros a una tabla.

Las sentencias MySQL que permiten añadir registros a una tabla son las siguientes:
INSERT tabla (campo1,campo2,..) VALUES (valor1,valor2,..)
Vamos a desarrollar un ejemplo completo de inserción de registros en la tabla de alumnos utilizada anteriormente. Se utilizaran diversos métodos.

2.12.1.1. Añadir un registro.

El primer método, y menos util, insertará unos datos que se incluyen en el script:

```
<?
# recogemos en una variable el nombre de BASE DE DATOS
$base="alumnosdb";

# recogemos en una variable el nombre de la TABLA
$tabla="alumnos";

# estableceremos la conexion con el servidor
$conexion=mysql_connect("localhost","pepe","pepa");

#asignaremos la conexión a una base de datos determinada
mysql_select_db($base,$conexion);

# AÑADIMOS EL NUEVO REGISTRO
mysql_query("INSERT $tabla (codalumno, dni, nombre, apellido1, apellido2, edad, curso, calificacion)
             VALUES ('1', '25648923', 'Juan', 'Gómez', 'Pérez', '17', '1', '5')",$conexion);


#comprobamos el resultado de la insercion
# el error CERO significa NO ERROR
# el error 1062 significa Clave duplicada
# en otros errores forzamos a que nos ponga el número de error
# y el significado de ese error (aunque sea en ingles)....
if (mysql_errno($conexion)==0){
  echo "<h2>Registro AÑADIDO</b></H2>";
}else{
  if (mysql_errno($conexion)==1062){
    echo "<h2>No ha podido añadirse el registro<br>Ya existe un campo con este DNI</h2>";
  }else{
    $numerror=mysql_errno($conexion);
    $descrerror=mysql_error($conexion);
    echo "Se ha producido un error nº $numerror que corresponde a: $descrerror <br>";
  }
}

# cerramos la conexion
mysql_close();
?>
```

2.12.1.2. Añadir un registro a partir de datos contenidos en variables

También es posible añadir registros a partir de valores contenidos en variables PHP. Esta opción es, sin ninguna duda, la más utilizada ya que lo habitual será escribir el contenido a añadir en un form y después -a través del method (POST o GET)- pasar al script de inserción esos valores como variables PHP.

Aquí tienes el código fuente de un ejemplo con la tabla anterior:

```
<?
# recogemos en una variable el nombre de BASE DE DATOS
$base="alumnosdb";

# recogemos en una variable el nombre de la TABLA
$tabla="alumnos";

# recogemos en variables los valores que vamos a asignar a cada campo
$v1="2";
$v2="9876545";
$v3="Gonzalo";
$v4="Fernández";
$v5="del Campo";
$v6=18;
$v7="1";
$v8=8;

# estableceremos la conexion con el servidor
$c=mysql_connect("localhost","root","");
#asignamos la conexión a una base de datos determinada
mysql_select_db($base,$c);

# AÑADIMOS EL NUEVO REGISTRO
mysql_query("INSERT $tabla (codalumno, dni, nombre, apellido1, apellido2, edad, curso, calificacion)
             VALUES ('$v1','$v2','$v3','$v4','$v5','$v6','$v7','$v8','$v9')",$c);

#comprobamos el resultado de la insercion
# el error CERO significa NO ERROR
# el error 1062 significa Clave duplicada
# en otros errores forzamos a que nos ponga el número de error
# y el significado de ese error (aunque sea en ingles)....
if (mysql_errno($c)==0){
  echo "<h2>Registro AÑADIDO</b></H2>";
}else{
  if (mysql_errno($c)==1062){
    echo "<h2>No ha podido añadirse el registro<br>Ya existe un campo con este DNI</h2>";
  }else{
    $numerror=mysql_errno($c);
    $descrerror=mysql_error($c);
    echo "Se ha producido un error nº $numerror que corresponde a: $descrerror <br>";
  }
}

# cerramos la conexion
mysql_close();
?>
```

2.12.1.3. Añadir registros con datos de un formulario.

cuando se envía el contenido de un formulario mediante el method=POST y se indica como action un fichero PHP los valores enviados son recogidos en este último fichero en variables de PHP que tienen como

nombre $_POST['var'] -o $HTTP_POST_VARS['var']- donde cada una de los índices asociativos de los array (var) coindicen con los name de los diferentes campos del formulario.

A partir de ahí, bastaría con depurar los valores recibos, recoger en variables los valores depurados e incluirlos en la sentencia MySQL INSERT para añadirlos a la tabla correspondiente.

Primero se crea el formulario:

```
<html>
  <head>
    <title>Formulario para añadir datos a la tabla alumnos</title>
  </head>
  <body>
    <center><h2>Tabla alumnos<br>Formulario de altas<br></center>
    <!— creamos un formulario en el que recogeremos los valores
         a añadir a la base de datos
         utilizaremos los mismos nombres de variables que en ejemplo anterior
         – por razones de claridad– anteponiendoles p_ //—>
    <form name="altas" method="POST" action="insertar.php">
      <table bgcolor="#E9FFFF" align=center border=2>
        <tr>
          <td align="right">Código: </td>
          <td align="left"><input type="text" name="p_v1" value="" size=6></td>
        </tr>
        <tr>
          <td align="right">D.N.I.: </td>
          <td align="left"><input type="text" name="p_v2" value="" size=8></td>
        </tr>
        <tr>
          <td align="right">Nombre....: </td>
          <td align="left"><input type="text" name="p_v3" value="" size=20></td>
        </tr>
        <tr>
          <td align="right">Primer apellido....: </td>
          <td align="left"><input type="text" name="p_v4" value="" size=20></td>
        </tr>
        <tr>
          <td align="right">Segundo apellido...: </td>
          <td align="left"><input type="text" name="p_v5" value="" size=20></td>
        </tr>
        <tr>
          <td align="right">Edad...: </td>
          <td align="left"><input type="text" name="p_v6" value="" size=4></td>
        </tr>
        <tr>
          <td align="right">Curso...: </td>
          <td align="left"><input type="text" name="p_v7" value="" size=4></td>
        </tr>
        <tr>
          <td align="right">Calificación...: </td>
          <td align="left"><input type="text" name="p_v8" value="" size=4></td>
        </tr>
        <!—colocamos los botones de enviar y borrar //—>
        <td align=center><input type=submit value="Enviar"></td>
        <td align=center><input type=reset value="Borrar"></td>
      </form>
    </table>
  </body>
</html>
```

Y a continuación el script que deberá llamarse insertar.php:

```
<?
```

```
# recogemos en una variable el nombre de BASE DE DATOS
$base="alumnosdb";

# recogemos en una variable el nombre de la TABLA
$tabla="alumnos";

# recogemos en variables los valores que vamos a asignar a cada campo
/* estas variables intermedias podrían evitarse. El hecho de usarlas
obedece unicamente a un intento de mayor claridad en la interpretación
de este codigo fuente */
$v1=$_POST['p_v1'];
$v2=$_POST['p_v2'];
$v3=$_POST['p_v3'];
$v4=$_POST['p_v4'];
$v5=$_POST['p_v5'];
$v6=$_POST['p_v6'];
$v7=$_POST['p_v7'];
$v8=$_POST['p_v8'];

# estableceremos la conexion con el servidor
$c=mysql_connect("localhost","root","");

#asiganamos la conexión a una base de datos determinada
mysql_select_db($base,$c);

# AÑADIMOS EL NUEVO REGISTRO
mysql_query("INSERT $tabla (codalumno, dni, nombre,apellido1, apellido2, edad, curso, calificacion)
             VALUES ('$v1','$v2','$v3','$v4','$v5','$v6','$v7','$v8')",$c);

#comprobamos el resultado de la insercion
# el error CERO significa NO ERROR
# el error 1062 significa Clave duplicada
# en otros errores forzamos a que nos ponga el número de error
# y el significado de ese error (aunque sea en ingles)....
if (mysql_errno($c)==0){
  echo "<h2>Registro AÑADIDO</b></H2>";
}else{
  if (mysql_errno($c)==1062){
    echo "<h2>No ha podido añadirse el registro<br>Ya existe un campo con este DNI</h2>";
  }else{
    $numerror=mysql_errno($c);
    $descrerror=mysql_error($c);
    echo "Se ha producido un error nº $numerror que corresponde a: $descrerror <br>";
  }
}

# cerramos la conexion
mysql_close();
?>
```

2.12.2. Consultar los registros de una tabla.

Las consultas de los datos y registros contenidos en una tabla ofrecen un amplisímo abanico de posibilidades. Veamos algunas de las posibilidades.

2.12.2.1. La consulta más simple

Utilizamos la sentencia:
SELECT * FROM tabla
obtendremos información sobre todos los campos (*) y la salida estará en el mismo orden en el que fueron añadidos los datos. Realizamos un script con la consulta anterior:

```
<?
# recogemos en una variable el nombre de BASE DE DATOS
$base="alumnosdb";

# recogemos en una variable el nombre de la TABLA
$tabla="alumnos";

# estableceremos la conexion con el servidor
$c=mysql_connect("localhost","root","");

#asignaremos la conexión a una base de datos determinada
mysql_select_db($base,$c);

# estableceremos el criterio de SELECCION
# en este caso el comodin * indica que se seleccionen todos los campos
$resultado= mysql_query("SELECT * FROM $tabla" ,$c);

# CREAMOS UNA CABECERA DE UNA TABLA (codigo HTML)
echo "<table align=center border=2>";

# estableceremos un bucle que recoge en un array cada una de las LINEAS DEL RESULTADO DE LA CONSULTA
# utilizamos en esta ocasión «mysql_fetch_row» en vez de «mysql_fetch_array» para EVITAR DUPLICADOS
# recuerda que esta ultima función devuelve un array escalar y otro asociativo con los resultados
##########################################################
# INDICES DE LOS ARRAYS RECOGIDOS EN $REGISTRO
#
# EN EL CASO DE QUE SELECT VAYA MARCADO CON * (CONSULTAS DE TODOS LOS CAMPOS DE LA TABLA)
# LA CORRESPONDENCIA ENTRE INDICE DE ESTE ARRAY ESCALAR Y LOS CAMPOS SERÍAN LA SIGUIENTES:
# Tendría INDICE 0 el elemento del array que recoge el valor del PRIMER CAMPO
# según el orden en el que fue CREADA LA TABLA, el indice 1 CORRESPONDERÍA
# al segundo de los campos en el ORDEN DE CREACIÓN Y ASÍ SUCESIVAMENTE....
##########################################################
while ($registro = mysql_fetch_row($resultado)){
  # insertamos un salto de línea en la tabla HTML
  echo "<tr>";

  # estableceremos el bucle de lectura del ARRAY con los resultados de cada LINEA y encerramos
  # cada valor en etiquetas <td></td>para que aparezcan en celdas distintas de la tabla
  foreach($registro as $clave){
    echo "<td>",$clave,"</td>";
  }
}

echo "</table>";
# cerramos la conexion
mysql_close();
?>
```

2.12.2.2. Consultando sólo algunos campos.

Ahora utilizaremos la sentencia
SELECT campo1,campo2, ... FROM tabla

y tendremos como resultado una lista completa, por el mismo orden que la anterior, pero sólo mostrando los campos indicados

```
<?
$base="alumnosdb";
$tabla="alumnos";

$c=mysql_connect("localhost","root","");
mysql_select_db($base,$c);

# estableceremos el criterio de SELECCION
# en este caso los campos Nombre, Apellido1, Apellido2 unicamente
#######################################################
# INDICES DE LOS ARRAYS RECOGIDOS EN $REGISTRO
#
# EN ESTE CASO (consulta de algunos campos) LA CORRESPONDENCIA ENTRE INDICE DE ESTE ARRAY
# ESCALAR Y LOS CAMPOS SERÍAN LA SIGUIENTES:
# Tendría INDICE 0 el campo Nombre (primero de la consulta).
# INDICE 1 correspondería a Apellido1
# el indice 2 correspondería a Apellido 3
#######################################################
$resultado= mysql_query("SELECT Nombre, Apellido1, Apellido2 FROM $tabla" ,$c);

echo "<table align=center border=2>";
while ($registro = mysql_fetch_row($resultado)){
  echo "<tr>";
  foreach($registro as $clave){
    echo "<td>",$clave,"</td>";
  }
}
echo "</table>";

mysql_close();
?>
```

2.12.2.3. Consulta seleccionando registros.

Utilizaremos la sentencia MySQL de esta forma

`SELECT campo1, ... FROM tabla WHERE condición`

que nos devolverá la lista de registros que cumplen la condición indicada. Aquí tienes un ejemplo muy sencillo:

```
<?
$base="alumnosdb";
$tabla="alumnos";

$c=mysql_connect("localhost","root","");
mysql_select_db($base,$c);

# estableceremos el criterio de SELECCION
# añadimos un criterio de seleccion WHERE para que sólo muestre a los menores de 16 años
$resultado= mysql_query("SELECT Nombre, Apellido1, Apellido2 FROM $tabla WHERE (edad<16) ",$c);


echo "<table align=center border=2>";
while ($registro = mysql_fetch_row($resultado)){
  echo "<tr>";
  foreach($registro as $clave){
    echo "<td>",$clave,"</td>";
  }
```

```
}
echo "</table>";

mysql_close();
?>
```

2.12.3. Modificar los registros de una tabla

2.12.3.1. Modificar un campo en todos los registros de una tabla

La sentencia MySQL, que permite modificar uno o varios campos en todos los registros de una tabla, es la siguiente:

```
UPDATE tabla SET campo1=valor1, campo2=valor2
```

¡Cuidado con esta sentencia!. Hay que tener muy presente que con esta sentencia -en la que no aparece WHERE- se modificarán TODOS LOS REGISTROS DE LA TABLA y por lo tanto los campos modificados tendrán el mismo valor en todos los registros.

Veamos un ejemplo:

```
<?
$base="alumnosdb";
$tabla="alumnos";
$c=mysql_connect ("localhost","root","");
mysql_select_db ($base, $c);

# asignamos el valor a escribir en todos los registros a una varibale
$valor=7;

# hacemos la llamada a MySQL mediante la función mysql_query y le decimos que
# UPDATE (modifique) la tabla y que lo haga (SET) en el campo edad
# poniendo el valor que en este caso es 7
$resultado=mysql_query("UPDATE $tabla SET edad=$valor",$c);

# cerramos la conexión con la base de datos
mysql_close($c);

# escribimos un mensaje para que nos avise del final de proceso de actualización
echo "<h2>Proceso de actualización terminado</h2>";
?>
```

2.12.3.2. Selección y modificación de un solo registro

Es una de las opciones más habituales. Es el caso en el que -mediante un formulario- asignamos una condición a WHERE y simultáneamente asignamos los nuevos valor del campo o campos elegidos. Requiere la siguiente sintaxis:

```
UPDATE tabla SET campo1=valor1, campo2=valor2 WHERE condición
```

La condición es fundamental en esta opción y normalmente aludirá a un campo índice (clave principal o única), de modo que sea un solo registro el que cumpla la condición. Podría ser el caso, en nuestro ejemplo, del campo codalumno que por su unicidad garantizaría que la modificación solamente va a afectar a uno solo de los registros.

Primero se realiza un formulario para introducir los datos:

```
<html>
  <head><title>Formulario de modificaciones</title></head>
  <body>
    <h2><center>MODIFICACION DE EDAD<BR>DE LA PRUEBA Nº 2 </h2></center>
    <FORM name="modificar" method="GET" action="modifi.php">
      <table align=center border=2>
        <td>Escriba el Código del alumno ..:</td>
        <td><input type="text" name="alumno" value=""></td><tr>
```

```
        <td>Escriba aquí la edad..:</td>
        <td><input type="text" name="edad" value=""></td><tr>
        <td align=center><input type="submit" value="Modificar"></td>
        <td align=center><input type="reset" value="Borrar"></td>
    </form>
  </table>
  <br><HR>
  </body>
</html>
```

Se implementa el script para modificar el registro:

```
<?
#recogemos del formulario las variables alumno y edad
$alumno=$_GET['alumno'];
$edad=$_GET['edad'];

$base="alumnosdb";
$tabla="alumnos";
$c=mysql_connect ("localhost","root","");
mysql_select_db ($base, $c);

##########################################################          #######
# COMPROBACION DE LA EXISTENCIA DE UN REGISTRO CON ESE D.N.I. #
##########################################################          #######
# Es una operación necesaria para advertir al usuario de la correcta realización
# del proceso de modificación.
# Si introducimos un codalumno inexistente la función UPDATE no DARA MENSAJE DE ERROR
# aunque evidentemente NO LO ACTUALIZARA tampoco
#
# Para hacer esa comprobación tenemos múltiples opciones una de ellas sería
# contar los registros en los que el codalumno es igual al valor recibido en la variable
#
# Si existiera el codalumno devolvería UNO en el índice CERO DEL ARRAY
# recuerda que los indices de ese array se corresponden con el orden
# en el que han sido insertados los campos en la opcion SELECT
# en este caso solo ponemos uno... COUNT(codalumno) por lo que el índice del array
# ha de ser el primero de los posibles que como sabes es CERO
$resultado=mysql_query("SELECT codalumno FROM $tabla WHERE (edad=$edad)",$c);
$comprueba=mysql_num_rows($resultado);

#HACEMOS LA COMPROBACION
if($comprueba==0) {
  $avisar="<h2>No existe el alumno</h2>";
}else {
  # hacemos la llamada a MySQL mediante la función mysql_query
  # y le decimos que UPDATE (modifique) la tabla
  # y que lo haga (SET) en el campo edad
  # poniendo el valor que en este caso $edad
  $resultado=mysql_query("UPDATE $tabla SET edad=$edad WHERE (codalumno=$alumno)",$c);

  #colocamos la opcion de mensaje de error por si se produce alguna incidencia
  if (mysql_errno($c)==0){
    echo " ";
  }else{
    if (mysql_errno($c)==1062){
      echo "<h2>No ha podido añadirse el registro</h2>";
    }else{
      $numerror=mysql_errno($c);
```

```
        $descrerror=mysql_error($c);
        echo "Se ha producido un error nº $numerror que corresponde a: $descrerror <br>";
      }
    }
  }
  mysql_close($c);
  ?>
```

2.12.4. Borrar registros seleccionándolos de una lista

En el ejemplo siguiente tienes el código para utilizar la cláusula WHERE en un proceso de borrado de registros que presenta un formulario que contiene una lista con todos los registros actuales y una casilla de verificación por cada uno.

Al marcar las casillas y enviar el formulario el script que recibe los datos procede al borrado de todos los registros marcados en todas la tablas afectadas.

```
<html>
  <head>
    <title>Formulario para ELIMINAR REGISTROS de la tabla alumnos</title>
  </head>
  <body>
    <?
      $base="alumnosdb";
      $tabla="alumnos";
      $conexion=mysql_connect("localhost","root","");
      mysql_select_db($base, $conexion);

      #creamos una consulta de las bases
      $resultado=mysql_query("SELECT * FROM $tabla ",$conexion);

      #creamos una tabla
      echo "<table align=center border=2 bgcolor='#E0FFFF'>";

      #y su cabecera
      echo "<tr bgcolor='#ffffff'><td colspan=7 align=center>
            Para BORRAR marca la casilla correspondiente al registro a eliminar</td>
            <tr bgcolor='#ffffff'>";
      echo "<td align=center>DNI</td>";
      echo "<td align=center>Nombre</td>";
      echo "<td align=center>Apellido 1</td>";
      echo "<td align=center>Apellido 2</td>";
      echo "<td align=center>Edad</td>";
      echo "<td align=center>Calificación</td>";
      echo "<td align=center>Borrar</td><tr>";

      # escribimos la etiqueta de apertura de un formulario como method=post
      # como action pondremos la direccion de la página que realizará las actualizaciones
      # en este caso sera borra.php
      echo "<form name='modificar' method=post action='borra.php'>";

      while($salida = mysql_fetch_array($resultado)){
        for ($i=1;$i<7;$i++){
          echo "<td>",$salida[$i], "</td>";
        }

        # despues de recoger los datos añadimos un campo de formulario
        # identificado por un NAME que es una matriz (borra) cuyo indice es el codalumno
        # al ponerle como VALUE='Si' lo que estamos haciendo es que
        # cuando este "marcada" la casilla ese elemento de la matriz tome valor Si
```

```
        # cuando "no está marcada" tomará valor NULL y por lo tanto
        # NO SERÃ ENVIADA POR EL METHOD POST
        echo "<td align=center><input type=checkbox name=borra[$salida[0]] value='Si'></td><tr>";
      }

      mysql_close($conexion)
    ?>
  <td colspan=5 align=center><br>
  <input type=submit value='Eliminar registros marcados'> 
  <input type=reset value='Borrar el formulario'>
  </form></table>
</body>
</html>
```

Veamos ahora el fichero borra.php:

```
<?
$base="alumnosdb";
$conexion=mysql_connect ("localhost","root","");
mysql_select_db ($base, $conexion);

# recogemos del formulario la matriz borra[] que tiene como indices
# los codalumnos de todos los registros de las bases de datos
# en los que la variable contenga el valor Si (los marcados)
# los registros no marcados (en el checkbox) no son transferidos
# por lo que TODOS LOS ELEMENTOS DEL ARRAY CORRESPONDEN A REGISTROS ELEGIDOS PARA BORRAR
# leemos ese array completo usando el bucle foreach y
# recogemos el indice y el valor en $indice y $valor
foreach ($_POST['borra'] as $indice=>$valor){

  # ejecutamos la instruccion DELETE filtrada por WHERE
  # para que borre el registro en el que coincida codalumno con el indice
  mysql_query("DELETE FROM alumnos WHERE (codalumno=$indice)",$conexion);
  $num_borrados +=mysql_affected_rows();
}
print ("Se han borrado ".$num_borrados." registros");

mysql_close($conexion);
?>
```

Capítulo 3

Plataformas

3.1. Entorno virtual de aprendizaje: Moodle

La palabra Moodle era al principio un acrónimo de Modular Object-Oriented Dynamic Learning Environment (Entorno de Aprendizaje Dinámico Orientado a Objetos y Modular), lo que resulta fundamentalmente útil para programadores y teóricos de la educación. También es un verbo que describe el proceso de deambular perezosamente a través de algo, y hacer las cosas cuando se te ocurre hacerlas, una placentera chapuza que a menudo te lleva a la visión y la creatividad. Las dos acepciones se aplican a la manera en que se desarrolló Moodle y a la manera en que un estudiante o profesor podría aproximarse al estudio o enseñanza de un curso en línea. Todo el que usa Moodle es un Moodler.

Ven y ¡moodlea con nosotros!

http://moodle.org/doc

3.1.1. Introducción.

Moddle es una "plataforma educativa" que se desarrolla bajo licencia GPL, se trata de "un paquete de software para la creación de cursos y sitios Web basados en Internet. Es un proyecto en desarrollo diseñado para dar soporte a un marco de educación social constructivista."

Es sencillo de mantener y actualizar y, salvo el proceso de instalación, no necesita prácticamente de "mantenimiento" por parte del administrador.

3.1.2. Instalación

La Web principal del programa es http://moodle.org/. Desde allí (sección Downloads) podemos bajar la última versión (por ahora), se trata de moodle-1.4.4.tgz esta versión se va depurando continuamente para solucionar los bugs que se van encontrando, por lo que se ofrece también el paquete moodle-latest-14.tgz que es en el que se van depurando errores, aunque no incluye más funcionalidades.

Para instalar Moodle en nuestro sistema necesitamos tener en funcionamiento: Apache, php y MySQL, y partiremos de esto.

Tenemos que garantizarnos que la directiva de Apache AcceptPathInfo esté en On[1], así que revisemos el fichero de configuración de Apache y añadámosla[2]:

AcceptPathInfo on

una vez cambiada la línea ejecutemos

```
# apachectl restart
```

o

```
# httpd2 -k restart
```

Una vez en nuestra máquina, el proceso de instalación es muy simple y se encuentra muy bien guiado en las intrucciones de la web de moodle (http://moodle.org/doc/?file=install.html). Se resume en:

- Poner el paquete moodle-latest-14.tgz en el lugar adecuado, en general[3] será /var/www/html, y desempaquetarlo

  ```
  # cp moodle-latest-14.tgz /var/www/html
  ```

[1] Si no esta así, no se pueden ver las fotos ni los ficheros que se suban.

[2] Si trabajamos con Debian debemos cambiar otro parámetro más del archivo de configuración de php (/etc/php4/apache2/php.ini). Se trata de cambiar la ruta donde se guardan los archivos de datos:

```
;session.save_path = /var/lib/php4
session.save_path = /tmp
```

para activar los cambios

```
# apache2ctl restart
```

[3] Recordemos que en Debian, versión 2.0.49 de Apache es /var/www

```
# cd /var/www/html
# tar -xzvf moodle-latest-14.tgz
# cd moodle
```

- Crear el directorio moodledata y ajustarle los permisos de forma adecuada. Es preferible que no sea accesible directamente desde la web. Así que un lugar posible puede ser[4]:

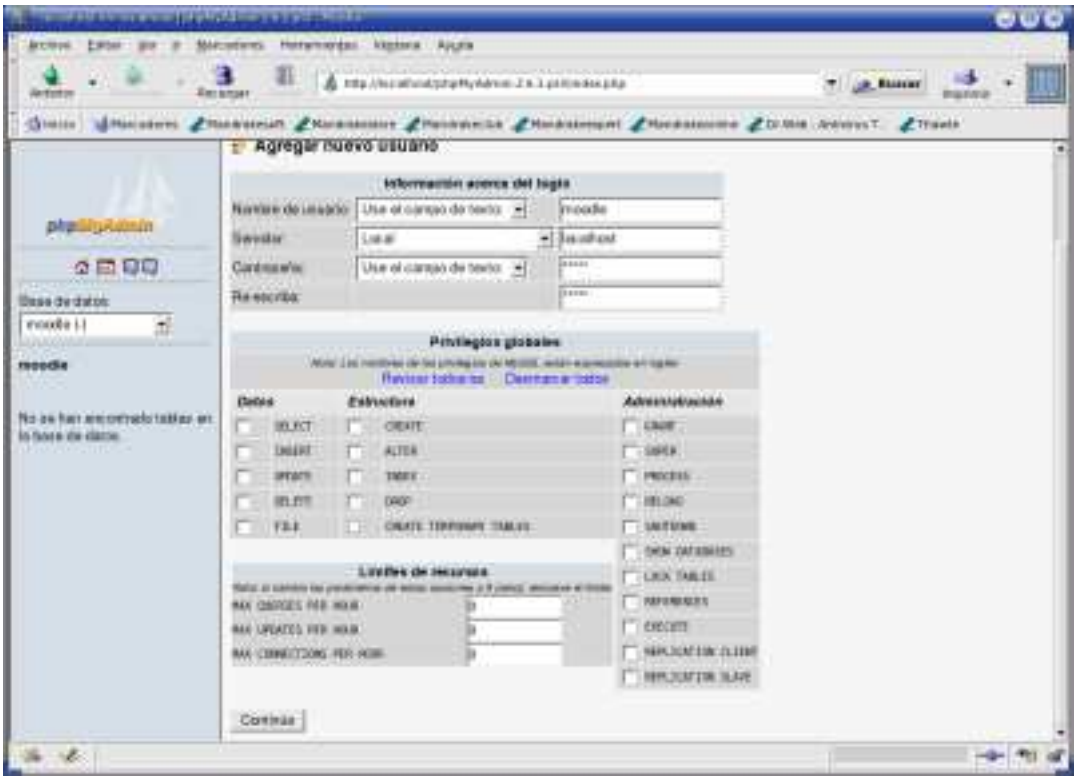
```
# mkdir /var/www/moodledata
# chown apache /var/www/moodledata
```

- Crear la base de datos[5] moodle:

```
# mysqladmin -u root -p create moodle
Enter password:
```

- De forma opcional, podemos optar porque un usuario (moodle por ejemplo) se conecte a esa base de datos. Para eso en la ventana principal de phpMyAdmin pulsemos sobre Privilegios Agregar nuevo usuario

  Introducimos el nombre de usuario, optamos porque las conexiones se realicen sólo desde Local y contraseña de acceso, y pulsamos sobre Continúe.



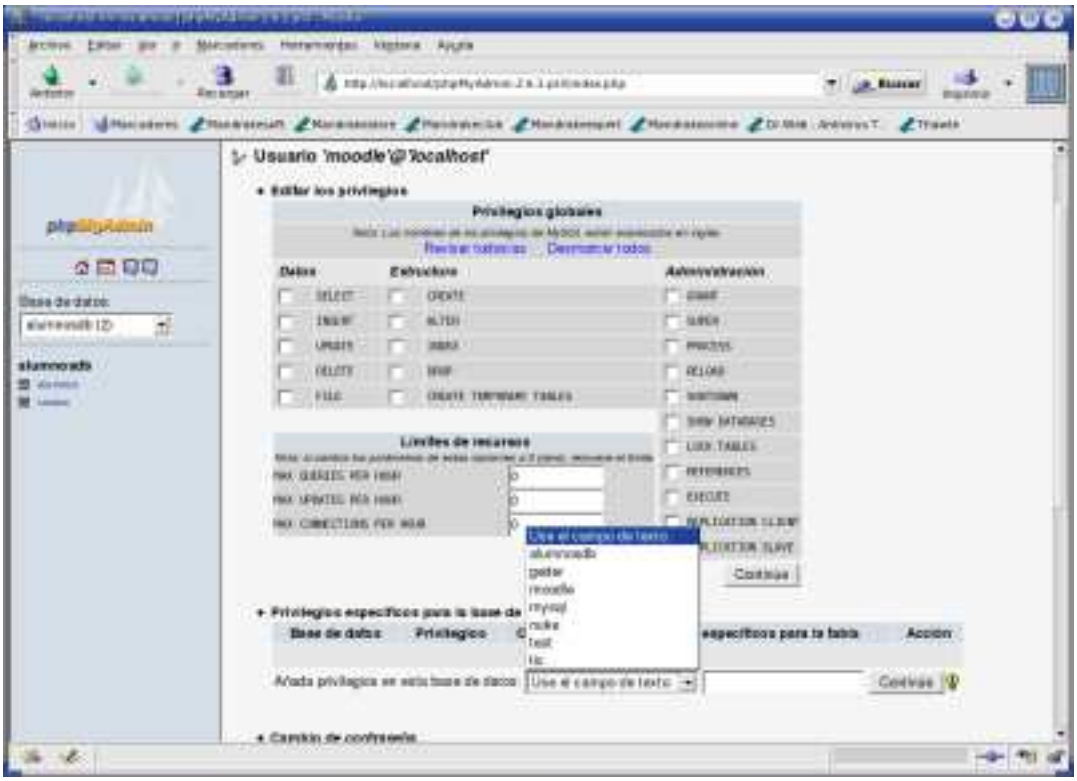
  Se nos abren más posibilidades y optamos por moodle en la lista desplegable Añada privilegios en esta base de datos.

[4] En Debian

```
# mkdir /var/moodledata y
# chown www-data /var/moodledata
```

[5] Si se desea se puede ejecutar:

```
# mysqladmin -u root -p
mysql>CREATE DATABASE moodle;
mysql>quit
```



Después, marcamos todos los privilegios para ella:

y pulsamos Continue.

Este mismo proceso se puede realizar desde la línea de comandos:

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 55 to server version: 4.0.18-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>GRANT ALL PRIVILEGES ON moodle.* TO moodle@localhost IDENTIFIED BY 'contraseña';
Query OK, 0 rows affected (0.00 sec)
```

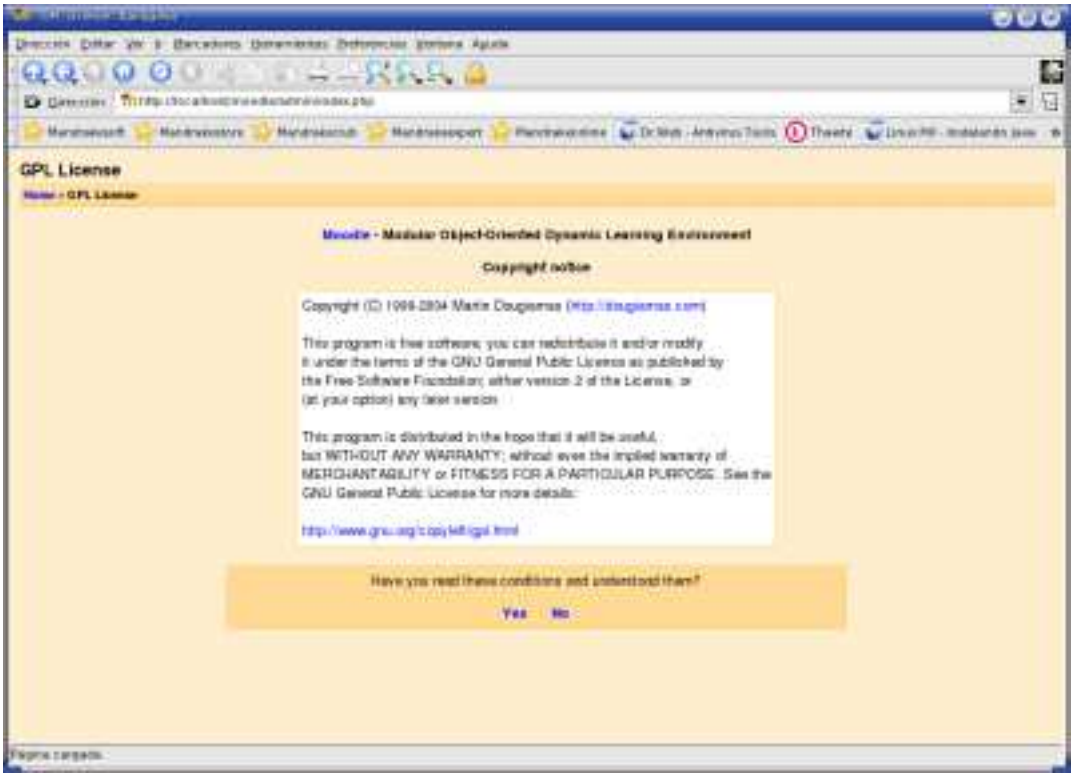
- Modificar el fichero config.php (manteniendo el original config-dist.php) para que pueda conectar (usuario y password).

  ```
  #cp config-dist.php config.php
  ```

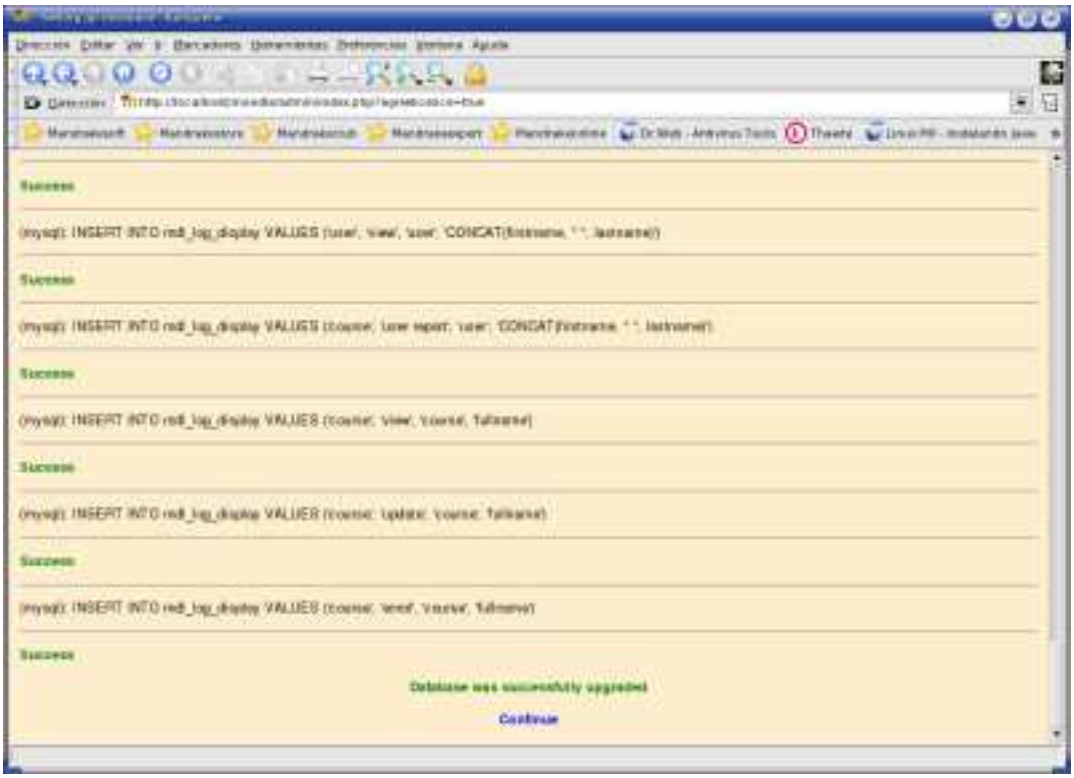
  Editamos el fichero y, además de ajustar los valores adecuados para poder conectar, hay que ajustar los path (por ejemplo) y mejor si restringimos un poco los permisos del directorio de datos[6]:

  ```
  $CFG->dbuser = 'moodle';
  $CFG->dbpass = 'password';
  $CFG->wwwroot = 'http://localhost/moodle';
  $CFG->dirroot = '/var/www/html/moodle';
  $CFG->dataroot = '/var/www/moodledata';
  $CFG->directorypermissions = 0750;
  ```

- Comienza la "moodlemanía". Escribamos en un navegador web: http://localhost/moodle.



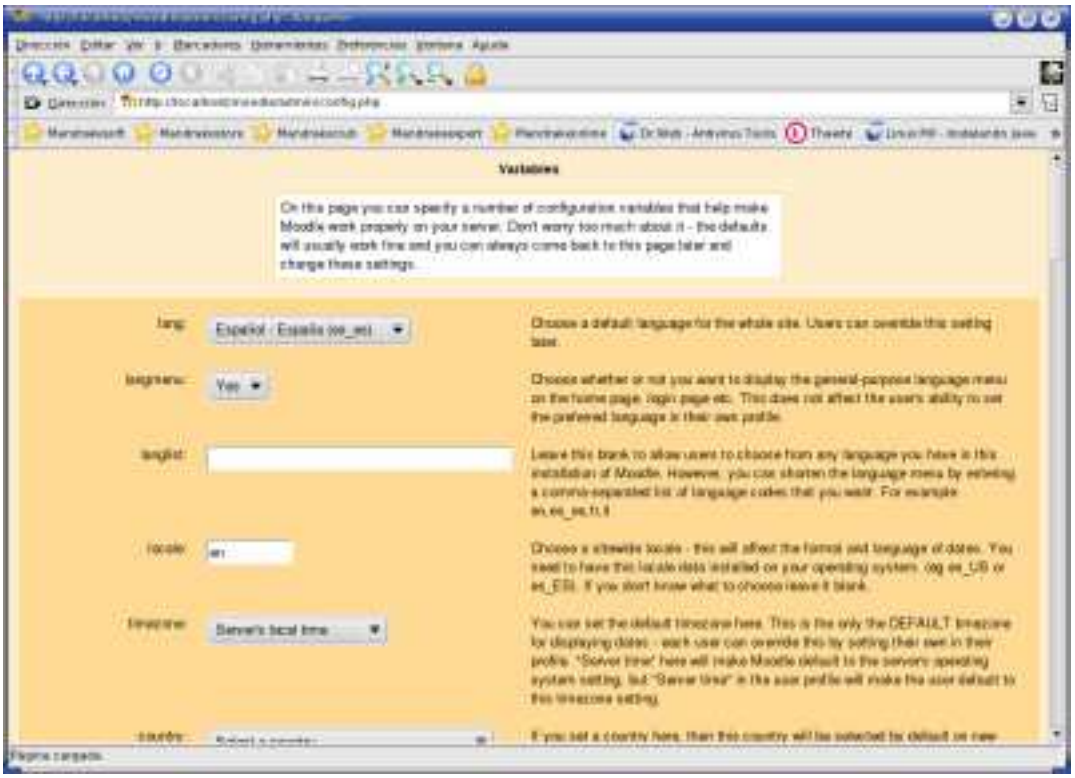
[6] Si tenemos un dominio escribiremos
$CFG->wwwroot = 'http://www.midominio.org/moodle';



Sólo tenemos que ir aceptando en las distintas pantallas que nos van a ir saliendo. Merece la pena pararse en la que nos informa de los cambios surgidos en esta versión. Después, cuando se han creado las tablas de la base de datos y directorios de datos, accederemos a la primera ventana de configuración propiamente dicha. Salvo que nos guste el inglés, lo mejor es seleccionar el castellano[7].

[7] Todo lo que cambiemos desde este momento podrá ser modificado después. Así que no hay ningún problema si nos equivocamos en algo ahora.

Aunque la captura aparece en castellano, incialmente estará en inglés. Hasta que no se opte por el idioma y se guarden los cambios no la veréis así.



La ayuda de contexto es muy buena[8], así que sólo comentaremos las variables susceptibles de ser cambiadas desde el principio:

[lang:] Español–España (es_es)

[locale:] optaremos por escribir es_ES

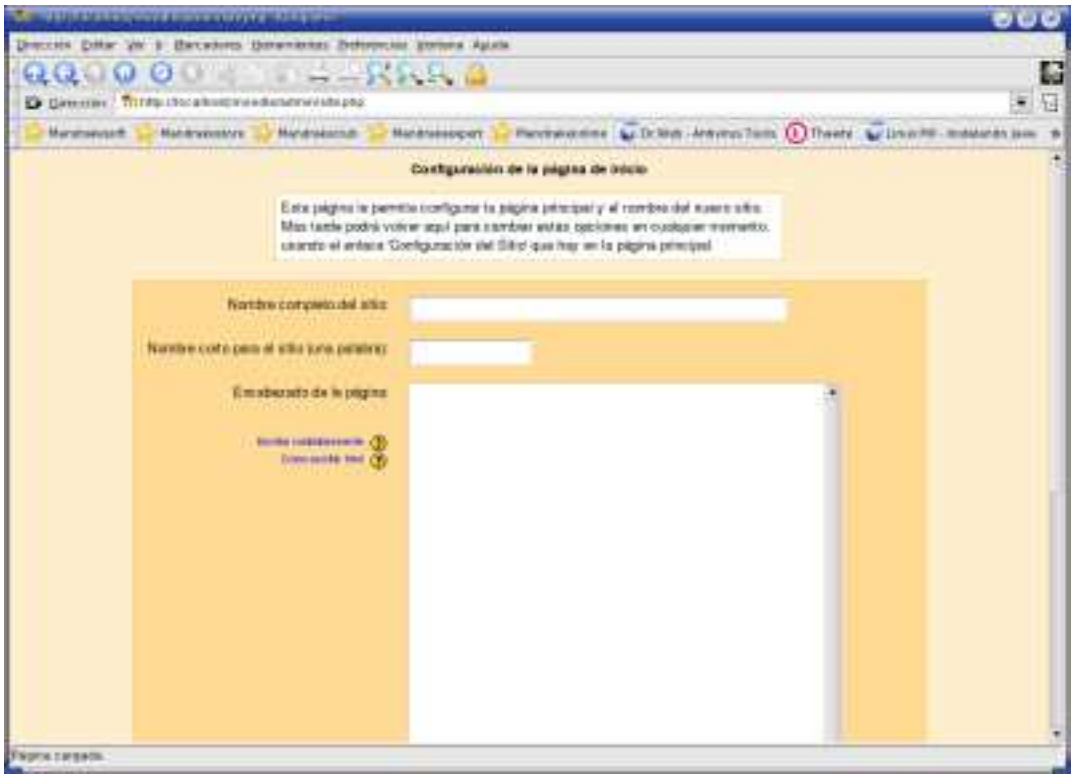
[zip:] en general será /usr/bin/zip

[unzip:] en general será /usr/bin/unzip

[country] deberíamos elegir el país por defecto para los nuevos usuarios .

Configuremos la página de inicio

[8] Y en castellano, sólo hay que optar por la variable del idioma, guardar y retroceder luego en el navegador.



Con la lista de forma de encabezado podemos optar por la forma que tendrá la página inicial de la aplicación, podemos elegir entre Mostrar items de noticia, Mostrar un listado de cursos o Mostrar un listado de categorias. A continuación debemos configurar la cuenta para el administrador principal. Debemos asegurarnos de darle un nombre de usuario y contraseña seguras además de una dirección de correo electrónico válida (y cómo no, la foto de rigor). Posteriormente podremos crear más cuentas de administración.

Listo, ya tenemos nuestro Moodle en funcionamiento. Si pulsamos sobre Admin, además de poder modificar todas la variables que definen el sitio, podremos acceder a la magnífca ayuda (en castellano) que acompaña al programa.

Además de la ayuda de contexto, desde http://moodle.org/mod/resource/index.php?id=11se puede bajar un completo manual para el profesor: manual_del_profesor.zip .

3.2. PHPNuke

3.2.1. Introducción

PHP-Nuke es un sistema automatizado de noticias especialmente diseñado para ser usado en Intranets e Internet. El Administrador tiene el control total de su sitio Web, sus usuarios registrados, y tendrá a la mano un conjunto de herramientas poderosas para mantener una página web activa y 100 % interactiva usando bases de datos.

Su autor es Francisco Burzi, que es el que mantiene el código y realiza todas las modificaciones que lleva el paquete original. Podemos encontrar su trabajo en http://www.phpnuke.org. Los requisitos para usar PHP-Nuke, que veremos posteriormente, no forman parte del sistema y han de instalarse independientemente. (Manual de referencia rápida para PHP-Nuke, Carlos Pérez Pérez http://www.forodecanarias.org/doc/nuke/html/node4.ht

3.2.2. Instalación de phpnuke

Necesitamos:

- Tener instalado Apache

- Tener instalado PHP
- Tener instalada la base de datos MySQL
- Instalar el módulo que permite a PHP disponer de soporte de base de datos MySQL

Empezamos:

Descomprimir el PHPNuke en /var/www (esto es opcional, pero así se queda ya puesto en su sitio):

```
# cp PHP-Nuke-7.2.zip /var/www
# cd /var/www
# unzip PHP-Nuke-7.2.zip
```

Una vez descomprimido y tras situarnos en el directorio /var/www/html/sql ejecutemos[9]:

```
# mysqladmin -u root -p create nuke
```

para crear la base de datos nuke, y

```
# mysql -u root -p nuke <nuke.sql
```

para crear las tablas de esta base de datos según se establece en el fichero nuke.sql.

Ajustar la contraseña de la base de datos en el fichero /var/www/html/config.php

```
$dbname = "root";
$dbpass = "contraseña";
```

Podemos tener varios Nukes instalados en nuestra máquina. Para eso sólo debemos descomprimirlos en directorios diferentes y crear bases de datos diferentes para cada uno de ellos, por ejemplo para el segundo nuke

```
# mysqladmin -u root -p create nuke2
# mysql -u root -p nuke2 <nuke.sql
```

y ajustar la variable adecuada a la base de datos para ese phpNuke, en concreto en el fichero config.php

```
$dbname = "nuke2";
```

Comprobar que todo está bien, apuntando con nuestro navegador a[10] http://localhost/

En primer lugar pongamos el portal en castellano para eso optemos porque el interfaz se muestre así, seleccionando el idioma español del bloque de la derecha .

Una vez pasado este trámite, vamos crear el super-usuario, no podemos usar ni espacios ni caracteres "extraños". Si los usamos no podremos acceder al portal.

En /var/www/html/docs se nos ha instalado la documentación que acompaña al portal. Más de una duda seguro que se puede resolver desde aquí.

[9] Véase el fichero /var/www/html/html/Install.txt para ampliar sobre los detalles de la instalación.
Podemos usar phpMyAdmin para hacerlo, pero en general es más rápido desde el modo comando.

[10] En Debian es mejor ajustar el DocumentRoot modificando el fichero /etc/apache2/sites-available/default y ajustarlo a DocumentRoot /var/www/html
Una vez cambiado:
apache2ctl restart
Se consigue administrar el portal apuntando a: http://localhost/admin.php