1. **El gestor de base de datos por excelencia, MySQL.** 
   1. Introducción. Conceptos básicos.
   2. Instalación de MySQL y phpMyAdmin en Linux.
   3. Instalación en Windows.
   4. Creación y borrado de una base de datos.
   5. Gestión de usuarios.
   6. Creación de tablas.
   7. Utilizando la base de datos: introducir/eliminar/borrar y recuperar información.
   8. Procedimientos almacenados.
   9. Consultas multitabla.
   10. Modelo entidad relación. Paso a relacional con Ferrer.
   11. Backup y restauración de la base de datos.
   12. Una alternativa potente PostgreSQL y PgAdmin para facilitarnos el trabajo.
   13. Introducción. Conceptos básicos.

Vamos a entrar ahora en el estudio de uno de los sistemas gestores de bases de datos más potentes y eficientes del mercado. MySQL es un sistema gestor de base de datos, es decir, nos permite gestionar grandes volúmenes de información de forma que su acceso sea rápido y eficiente. MySQL sigue el modelo de base de datos relacional. Sin entrar en grandes detalles, la idea es organizar la información en tablas que se encuentran relacionadas.

phpMyAdmin es una herramienta escrita en PHP para facilitar las tareas de administración de MySQL con una interfaz tipo Web. Permite crear y borrar bases de datos, crear/borrar/modificar tablas, eliminar/editar/añadir campos, ejecutar cualquier consulta SQL, etc. Para más información visita su página Web, <http://www.phpmyadmin.net/>**.**

Otro concepto importante que debes conocer es SQL. SQL son las siglas de Structural Query Lenguage, lenguaje estructurado de consultas. Es el lenguaje de base de datos relacional estándar, contiene órdenes para definir la base de datos (tablas, índices, vistas, etc.), para manipularla (recuperar datos, insertar/borrar/modificar datos). Incluso existe el SQL inmerso, es decir, consultas SQL insertadas en programas escritos en lenguajes estructurados tipo C o Cobol.

* 1. Instalación de MySQL y phpMyAdmin en Linux.

Vamos a empezar por la instalación, sigue el siguiente procedimiento:

|  |  |
| --- | --- |
| Procedimiento  .procedimiento.tif  Si instalaste XAMPP desde Windows ya no tienes nada más que hacer, MySQL iba incluido en el lote.  En Linux instala los paquetes: *mysql-server*, *mysql-client*, *mysql-admin*, *php5*, *libapache2-mod-auth-mysql*, *php5-mysql y phpmyadmin*.  En la instalación te solicitará, como se observa en la figura, que teclees la contraseña o password del usuario root (Administrador) de MySQL. | mysql  .-mysql.tif  Te pedirá que repitas la contraseña, no la olvides. Para hacer clic en **Aceptar** puedes utilizar el tabulador. |
| Ahora debes elegir el servidor Web con el que se configurará phpMyAdmin, selecciona **apache2** utilizando la barra espaciadora, luego con el tabulador pulsa **Aceptar**. | *mysql2* |
|  | |
| Además tendrás que configurar phpMyAdmin. Teclea en un navegador **http://localhost/phpmyadmin/**, indica el idioma **Español-Spanish (utf-8)**, el usuario (**root**) y contraseña la que indicaste en el paso anterior y haz clic en **Continuar**.  Observa el resultado final en la figura adjunta. | *phpmyadmin*  .-phpmyadmin.tif |
| *phpmyadmin2*  .-phpmyadmin2.tif | |

* 1. Instalación en Windows.

|  |  |
| --- | --- |
| Si eres usuario de Windows, lo primero que vamos a realizar es darle una contraseña al administrador de MySQL.  Desde la página de tu navegador con la dirección de tu servidor local, <http://localhost>, selecciona **Español** si no lo has hecho todavía y haz clic dentro de **Herramientas** en **phpMyAdmin**.  Desde aquí puedes cambiar el idioma: **Español-Spanish** y el estilo o tema de phpMyAdmin. | phpmyadminWindows  .-phpmyadminwindows.tif  Observa en el panel inferior un cuadro de advertencia que te indica que debes proporcionar una palabra de paso para el usuario root o administrador. Así como los pasos a seguir: seleccione **Privilegios** y agregue la contraseña a root@localhost. Deberá escribir la misma contraseña en config.inc.php de phpMyAdmin. Por tanto haz clic en **Privilegios**. |
|  |  |
| Ahora puedes observar los usuarios que tiene tu base de datos, hay una línea por usuario. Desde esta ventana serás capaz de editar sus permisos, crear o eliminar usuarios, etc.  Vamos a asignar una contraseña para root (del Servidor localhost), así que edita sus privilegios haciendo clic en el icono phpmyadminWindows3  .- phpmyadminwindows3.tif  . | phpmyadminWindows2  .- phpmyadminwindows2.tif |
|  |  |
| En esta ventana puedes visualizar y editar los privilegios sobre la edición, estructura y administración de los datos, así como cambiar o asignar una contraseña.  Ahora tenemos que modificar el fichero *c:\xampp\phpMyAdmin\config.inc.php* la línea *$cfg[‘Servers’][$i][‘password’]=’****tuClaveFavorita****’*; donde en negrita figura lo que debes cambiar. | phpmyadminWindows4  8.-phpmyadminWindows4.tif  En el panel **Cambio de contraseña**, selecciona **Contraseña** e introdúcela, observa que debes escribirla dos veces. Cuando finalices haz clic en **Continuar**. |

A partir de aquí todo será idéntico en ambos sistemas operativos.

* 1. Creación y borrado de una base de datos.

|  |  |
| --- | --- |
| Desde la página principal, crear una base de datos es un juego de niños.  Observa el cuadro de texto rotulado **Crear nueva base de datos**, escribe el nombre deseado y haz clic en **Crear**. | crearBaseDatos  9.-crearBaseDatos.tif |
|  |  |
| Observa el resultado obtenido: Base de datos myBaseDatos se creó. Además fíjate dónde te encuentras, en la base de datos myBaseDatos del servidor localhost.  También puedes conocer la consulta SQL que generó tu petición: *CREATE DATABASE ‘myBaseDatos’;* | crearBaseDatos2  10.-crearBaseDatos2.tif |
| Imagina que te has equivocado y has escrito my en vez de myBaseDatos. Fácil, haz clic en el botón Eliminar.  Observa que te pide confirmación y que te informa de la consulta SQL que se generará: *DROP DATABASE ‘my’;* | eliminarBaseDatos  .-eliminarBaseDatos.tif |

Por supuesto todo esto se puede hacer desde la consola. Veamos ahora todo lo que hemos aprendido desde una interfaz Web realizado en el terminal. En Windows recuerda: **Inicio**, **Ejecutar**, teclea *cmd* para entrar en el intérprete de comandos de Windows.

Escribe *mysql –u root –p*. Ahora tendrás que teclear tu clave, observa que u indica con que usuario entras en MySQL (u es de **U**ser, p de **P**assword). Una vez conectados, podemos crear una base de datos, por ejemplo recuerda: *mysql>create database myOtraBaseDatos;*

* 1. Gestión de usuarios.

|  |  |
| --- | --- |
| Para crear un nuevo usuario desde la página principal de phpMyAdmin, selecciona **Privilegios** y dentro de esta **Agregar un nuevo usuario**.  Ahora rellena los campos típicos: Nombre de usuario, Servidor (típicamente locahost) y contraseña. Haz clic en **continuar**. | usuarios  .-usuarios.tif |
|  |  |
| Una vez creado el usuario podemos editar los privilegios para dicho usuario y darle todos los permisos (o sólo algunos) a cierta base de datos, por ejemplo myBaseDatos.  Escribe en la ventana de editar privilegios, en la sección de **Añadir privilegios a esta base de datos** tu base de datos, en el ejemplo *myBaseDatos*. | usuarios2  .-usuarios2.tif |
| Observa la ventana que te aparece, lo típico es darle al usuario “*joe*” todos los permisos sobre la base de datos “*mybasedatos*”. Haz clic en **Marcar todos/as** y luego en **Continuar**.  La respuesta será:  Ha actualizado los privilegios para ‘*joe’@’localhost*’ (nombreUsuario@nombreHost). | usuarios3  .-usuarios4.tif  Además te mostrará la consulta SQL equivalente: *GRANT ALL PRIVILEGES ON ‘mybasedatos’.\* to ‘joe’@’localhost’ WITH GRANT OPTION;* |

Por supuesto, todo esto se puede realizar desde la consola. Para crear usuarios teclea: *mysql>create user joe identified by ‘tuClaveFavorita’;*

Dichos usuarios no podrán hacer casi nada hasta que no les des permisos, el más típico es darle todos los permisos sobre una base de datos, en el ejemplo: *mysql>grant ALL on* mybasedatos*.\* to joe@localhost identified by ‘tuClaveFavorita’;* Por último, puedes salir de MySql: *mysql>quit.*

Una vez que el usuario ha sido creado, podrá entrar con su nombre y contraseña, para ello debe escribir:

mysql>mysql –u joe –p

Ahora podría ver las bases de datos del sistema: *mysql>show databases;*

Deberá ver dos: information\_schema y la recién creada myBaseDatos. Si quisiera ver cuantas tablas existen, deberá primero posicionarse en una base de datos sobre la que tenga permiso, en nuestro caso ejecutará: *mysql>use myBaseDatos*. Luego dentro de esta base de datos ejecuta: *mysql>show tables;* No hay todavía ninguna tabla, vamos a solucionarlo en el siguiente apartado del capítulo.

* 1. Creación de tablas.

|  |  |  |
| --- | --- | --- |
| Para crear una tabla, hemos de seleccionar en el panel izquierdo una base de datos (p.e. mybasedatos). Como puedes observar no hay todavía ninguna tabla dentro de dicha base de datos.  Basta con escribir el nombre de la tabla (Libros) y el número de campos (4); luego haz clic en **Continuar**, para definir los distintos campos que forman la tabla. | | *phpmyadmin9*  *15.-phpmyadmin9.tif* |
|  | |  |
| Definimos los cuatro campos: código (un entero, de tipo INT), nombre, autor y editorial.  Estos tres últimos campos son de tipo VARCHAR e indicamos la longitud máxima. Además código va a ser clave primaria, observa en la figura inferior que la llave (clave primaria) está seleccionada para el primer campo. | | *phpmyadmin10*  *16.-phpmyadmin10.tif* |
| |  |  | | --- | --- | | idea  .-idea.tif | Recuerda que MySQL es un sistema gestor de base de datos relacional. Una de las ideas clave de un modelo relacional es que los datos están organizados en tablas (p.e. Libros, Personas, Paciente, etc.). Cada tabla está formada por filas o tuplas y no pueden existir dos tuplas duplicadas. La clave es el conjunto de atributos mínimo en el que dos tuplas deben de diferir (p.e. código, dni, número seguridad social, etc.), es decir, no puede haber dos libros con el mismo código, dos personas con el mismo dni, etc. | | | |
| *phpmyadmin11*  *18.phpmyadmin11.tif* | Fíjate en el resultado final, indica que la tabla se creó y la consulta SQL generada.  En consola si quieres ver la descripción de dicha tabla ejecuta: *mysql>describe Libros;*  Observa también la pestaña **Insertar** para empezar a introducir datos en la tabla y la pestaña **Eliminar** para eliminar la tabla. | *phpmyadmin12*  *19.-phpmyadmin12.tif*  Observa lo tedioso de la consulta SQL que crea la tabla, la eliminación es bastante más sencilla: *drop table Libros;* |

También puedes modificar la tabla, recuerda siempre estar en la pestaña **Estructura**, porque quieres modificar la estructura o definición de la tabla. Por ejemplo, añadir un nuevo campo (en la parte inferior de la figura), fíjate donde dice **Añadir 1 campo**, decides en que posición: al final de la tabla, al comienzo de la tabla o después de algún campo (en nuestro ejemplo código, nombre o autor). Otra opción es borrar un campo, para ello selecciona un campo (haciendo clic en el checkbox a su izquierda) y elimínalo utilizando el aspa:eliminarCampos

.-eliminarCampos.tif

* 1. Utilizando la base de datos: introducir/eliminar/borrar y recuperar información.

|  |  |
| --- | --- |
| Ahora procedemos a insertar dos libros.  Fíjate que es bastante sencilla y amigable la introducción de datos en la tabla (recuerda como paso previo hacer clic en la pestaña **Insertar**). Cuando acabes haz clic en **Continuar**.  El resultado será una pantalla informándonos del número de filas insertadas (2) y de la consulta SQL que las insertó. | *phpmyadmin13*  *21.-phpmyadmin13.tif*  Una inserción en SQL tiene la forma: *INSERT INTO ‘myBaseDatos’.’Libros’ (‘codigo’,’nombre’,’autor’,’editorial’) VALUES (*  *‘1’, ‘Pásate a Linux’, ‘Máximo y Javier’, ‘Inforbooks’);*  Que debes leer cómo: inserta en la tabla Libros de la base de datos myBaseDatos la siguiente tupla (o tuplas): ‘1’ que sería el código del libro, ‘Pásate a Linux’ el nombre, etc. |
| |  |  | | --- | --- | | idea  .-idea.tif | Muchas veces tienes los datos en ficheros de texto. Por ejemplo, si tienes los libros en un fichero myLibros.txt con el siguiente formato:  3 Guía del perfecto Webmaster Máximo y Javier Inforbooks  4 Pásate a Ubuntu Máximo y Javier Inforbooks  Es decir, los campos separados por tabuladores y las tuplas por saltos de línea o retorno de carro, puedes insertarlas en tu tabla Libros con la siguiente sintaxis:  *Load data local infile “myLibros.txt” into table Libros;* | | |
|  |  |
| Observa que desde phpMyAdmin también podemos hacer consultas SQL, haz clic en la pestaña **Examinar**, **SQL** o **Buscar**.  La más básica será recuperar todas las tuplas o filas de la tabla, esta es la que el sistema crea por defecto cuando hacemos clic en SQL, sin embargo, puedes crear cualquier otra consulta SQL. | *phpmyadmin14*  *23.-phpmyadmin14.tif*  Observa que seleccionando las tuplas deseadas, puedes editarlas (y así modificarlas) con el lápiz o borrarlas con el aspa eliminarCampos.  .-eliminarCampos.tif |
| Esta es la consulta SQL más sencilla que puedes realizar: Muéstrame todos los libros, *SELECT \* FROM ‘Libros’;*  Veamos otras consultas y sus correspondientes en SQL:   * Muéstrame todos los libros ordenados alfabéticamente por nombre: SELECT *\* FROM ‘Libros’ ORDER BY nombre ASC;* * Quiero consultar todos los libros pero sólo me interesa el nombre y el autor: *SELECT nombre, autor FROM ‘Libros’;* Observa que hemos sustituido la \* por la enumeración de los campos que queremos visualizar. * Recupera el libro cuyo código es el 1*: SELECT \* FROM ‘Libros’ WHERE codigo=1;* La cláusula WHERE impone una condición sobre la búsqueda que debe satisfacerse. | |
| Por supuesto todo esto lo podemos realizar desde phpMyAdmin.  Realicemos otra consulta en phpMyAdmin a la base de datos, “Muéstrame todos los libros de Máximo y Javier”, primero haz clic en **Buscar**.  En la ventana que observas puedes seleccionar los campos a mostrar, el número de **registros por página** (p.e. si hubiéramos publicado más de 30 libros sólo mostraría 30 en cada página), que nos muestre los libros ordenados por nombre y que muestre sólo los que tengan como autores a ‘Máximo y Javier’. | *phpmyadmin15*  *25.-phpmyadmin15.tif* |
|  |  |
| En el resultado obtenido nos indica:   * Ha obtenido dos registros o filas en 0,0003 segundos. * La consulta SQL que se corresponde a lo que le habíamos solicitado gráficamente. * Las tuplas o filas ordenadas alfabéticamente por nombre (primero **A**prende en Libre, después **P**ásate a Linux). | *phpmyadmin16*  *26.-phpmyadmin16.tif* |
|  |  |
| Obsérvese ahora una sesión típica en modo consola en Linux.  Entramos con *mysql –u joe –p*, recuerda que debes restringir el uso de root para tareas administrativas.  Cuando accedes a mySQL, escribe *use myBaseDatos;* ya que vamos a trabajar sobre dicha base de datos.  Luego creamos una tabla con CREATE TABLE, insertamos dos tuplas o filas con la orden INSERT y extraemos información de MySql (en este caso, los dos libros que previamente habíamos insertados) con la cláusula SELECT. | *mysql*  *27.-mysqlConsola.tif*  Recuerda que habíamos ya creado, desde la consola, el usuario joe, la base de datos myBaseDatos y que habíamos dado todos los permisos a joe sobre dicha base de datos. |

Para borrar tuplas basta con utilizar la cláusula DELETE, por ejemplo, para eliminar el libro con código 3, ejecuta la siguiente sentencia: *mysql>delete from Libros where codigo=3;*

Finalmente te mostramos un cuadro resumen de la sintaxis básica del SQL para crear tablas, insertar o recuperar información:

|  |  |
| --- | --- |
| **Cláusula SQL** | **Significado** |
| CREATE TABLE nombreTabla (  nombreCampo1 Tipo  nombreCampo2 Tipo  ….  PRIMARY KEY (campoClavePrimaria)  ); | Crea una tabla llamada nombreTabla con los campos nombreCampo1, nombreCampo2, etc., cuya clave o llave primaria es campoClavePrimaria. |
| INSERT INTO nombreTabla VALUES (valor1, valor2,… valorN); | Inserta una tupla o fila en la tabla nombreTabla. |
| SELECT nombreCampo1, nombreCampo2 ó \*  FROM tabla1, tabla2  WHERE condición; | Extrae información de la base de datos. Podemos recuperar información de varias tablas (FROM), seleccionar los campos a mostrar (si indicamos \* significa que queremos que se visualicen todos los campos), indicar una condición de búsqueda (WHERE), etc. |

* 1. Procedimientos almacenados.

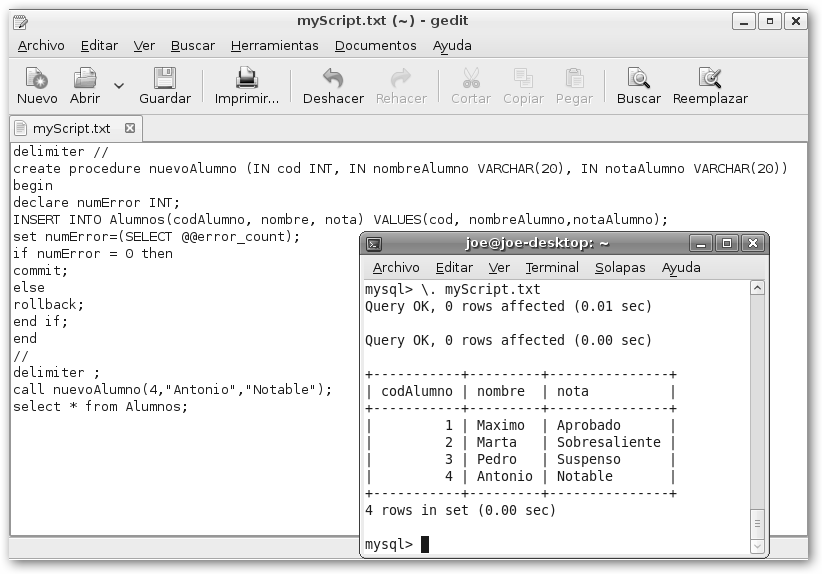
Un procedimiento almacenado es un programa que se almacena y ejecuta en el servidor de base de datos, el cual suele estar en una máquina distinta al servidor Web. El procesamiento en el servidor de las funciones típicas de tratamiento de los datos reduce la transferencia de información con los programas clientes de tal información. También proporciona más seguridad pues el servidor de base de datos suele estar en un ordenador privado de la Intranet. Proporciona además independencia de la lógica de los datos respecto a la de negocios o interfaz consiguiendo una mayor reutilización y mantenibilidad. Ejemplos de utilización de procedimientos almacenados son la validación de los datos o la “doble contabilidad” de las operaciones realizadas sobre la base de datos. Así, en grandes empresas se hace un registro de todas las operaciones que se han realizado para proveerle de mayor seguridad y poder auditar el sistema.

|  |  |
| --- | --- |
| No es el objetivo de esta obra describir con profundidad la sintaxis de los procedimientos almacenados. Sólo veremos algunos ejemplos.  Por ejemplo, en la figura creamos un procedimiento almacenado llamado listadoLibros que realiza una consulta para mostrar todos los libros de la tabla.  Para invocar el procedimiento almacenado desde el propio MySQL basta escribir *call* seguido del nombre del procedimiento. | *procedimientosAlmacenados*  .-procedimientosAlmacenados.tif  Para mayor información visita las direcciones: <http://dev.mysql.com/tech-resources/articles/mysql-storedprocedures.pdf> y [http://dev.mysql.com/doc/refman/6.0/en/index.html (Capítulo 17](http://dev.mysql.com/doc/refman/6.0/en/index.html%20(Capítulo%2017)**,** Stored Programs and Views). Te aconsejamos consultar la versión en inglés que suele estar más actualizada. |
| En un segundo ejemplo, insertamos un alumno en la tabla Alumnos con un procedimiento almacenado.  Lo más interesante a destacar es que este procedimiento almacenado presenta tres parámetros de entrada, obsérvese cómo están declarados: *IN nombreParámetro Tipo.* | *procedimientosAlmacenados2*  .- procedimientosAlmacenados.tif  La tabla Alumnos consta de tres campos: codAlumno (INT), nombre y nota (ambos VARCHAR(20)). |

Finalmente, veamos un ejemplo más útil de procedimientos almacenados:

* En vez de escribirlo desde la consola mysql, creamos y editamos (por ejemplo con gedit, kate, emacs, etc.) un fichero llamado myScript.txt y luego lo invocaremos mediante: *\. myScript.txt*. Observa que es mucho más cómodo y rápido escribir procedimientos almacenados mediante esta técnica.
* Fíjate ahora en el procedimiento almacenado. En primer lugar cambiamos el delimitador por defecto, en vez de ser el punto y coma “;” será ahora las dos barras verticales “//”. Esto lo realizamos porque nuestro procedimiento contendrá más de una sentencia SQL y el motor de MySql debe ser capaz de conocer cuando acaba nuestro procedimiento y no simplemente una sentencia SQL inmersa dentro de el. Se realiza con *delimiter //*.
* Declaramos el procedimiento de la forma habitual: *create procedure nombreProcedimiento (IN cod INT,…)*. Observa que el procedimiento contiene las palabras clave begin y end para delimitar el comienzo y el fin. Cuando acabamos el procedimiento lo señalamos con el delimitador que hemos definido **//** y luego cambiamos al separador por defecto “;” con *delimiter ;*
* Realizamos una llamada al procedimiento almacenado que inserta una nueva tupla en la tabla Alumnos y una consulta SQL para comprobar que todo funciona correctamente.
* Sin embargo el “meollo” viene ahora. Declaramos una variable de tipo entero numError, insertamos un alumno con los parámetros que hemos recibido y almacenamos en nuestra variable numError el número de errores que se han producido.
* Si el numero de errores es cero, es decir, no ha habido ningún problema entonces finalizamos la transacción con **COMMIT** (o hacemos los cambios pertinentes) sino la deshacemos con **ROLLBACK**.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | idea  .-idea.tif | Una transacción es un conjunto de órdenes que deben realizarse como una unidad, es decir, o se realizan todas o ninguna. Comienza con la sentencia BEGIN, realizamos las consultas SQL (inserción, actualización, eliminación) y cuando queramos guardar los cambios finalizamos la transacción con COMMIT, si hay algún problema lo deshacemos con ROLLBACK. Para ver los errores podemos utilizar *SHOW COUNT(\*) ERRORS;*  (<http://dev.mysql.com/doc/refman/5.0/es/show-errors.html>) y para conocer el número: *SELECT @@error\_count;* | |



.-procedimientosalmacenados3.tif

* 1. Consultas multitabla.

SQL permite también las consultas multitablas, es decir, las consultas sobre la base de datos que implican varias tablas.

Precisamos introducir primero unos conceptos básicos. Como hemos comentado una base de datos está formada por un conjunto de tablas. Estas tablas tienen un conjunto de propiedades o atributos. Uno de ellos es la clave primaria, una clave primaria es un atributo o conjunto de atributos que identifica unívocamente a una tupla. Pongamos un ejemplo, crearemos una tabla llamada books. Nuestra tabla tiene tres atributos: un código que será la clave primaria (codLibro), el nombre del autor o autores del libro (nombreAutor) y el nombre del libro (nombreLibro).

Compliquemos un poco más el escenario, imaginemos que tenemos bibliotecas, estas podrían definirse con una tabla (library) con los siguientes atributos: código de la biblioteca (codBiblioteca) que sería la clave primaria y el nombre de la biblioteca (nombreBiblioteca). Pero, ¿cómo modelamos que una biblioteca posee un número de libros? Podríamos agregar los atributos codLibro, nombreAutor, nombreLibro a la tabla Biblioteca. Sin embargo esta solución tiene dos problemas:

1. El primer problema se refiere a cual sería la clave primaria. Este problema se puede resolver haciendo que la clave primaria sea ahora dos atributos: codBiblioteca + codLibro.
2. El siguiente inconveniente es la redundancia de la información, es decir, existe mucha información repetida.

Una solución sería mantener dos tablas y crear en la tabla library una clave externa a la primera, sería como un puntero que lo vincula o relaciona a la tabla books.

|  |  |
| --- | --- |
| Observa en la figura adjunta que en la tabla library hay un atributo llamado codLibroCE de tipo entero y fíjate en la sentencia: *FOREIGN KEY (codLibroCE) REFERENCES books(codLibro)*, es decir, la clave externa codLibroCE referencia a la clave primaria codLibro de books. | *tablasRelacionadas*  .-tablasRelacionadas.tif  Así cuando realizas: *insert into library values (1, ‘Biblioteca de Málaga’, 3)* indicas que estas insertando en la Biblioteca de Málaga con código 1 el libro con código 3, más concretamente el titulado Guía del perfecto Webmaster. |
| Observa ahora como podemos consultar todos los libros que poseen las librerías:  select codBiblioteca, codLibro, nombreAutor, nombreLibro  from library, books (es una consulta de dos tablas: library y books)  where library.codLibroCE=books.codLibro;  La última sentencia es la clave, especifica la condición de búsqueda y vincula las dos tablas y se lee así: seleccióname todas las bibliotecas y libros donde el campo codLibroCE de la tabla library sea el mismo que algún código de algún libro (codLibro) de books. | |
| Fíjate la respuesta que hemos obtenido, una tupla por cada biblioteca y libro que posean dichas bibliotecas. | *tablasRelacionadas2*  .-tablasRelacionadas2.tif |
| Si quisiéramos saber cuantos libros hemos vendido en todas las librerías la consulta SQL sería: | *Select COUNT(codLibro)*  *From library, books*  *Where library.codLibroCE=books.codLibro;*  Nos devuelve 5. |

Esto es una consulta sumaria, es decir, SQL nos permite resumir los datos de la base de datos mediante un conjunto de funciones, a saber: SUM, COUNT, AVG, MAX y MIN. Estas funciones toman un conjunto de datos y producen un único resultado. En particular: SUM (suma), COUNT (cuenta cuantas instancias, en el ejemplo cuantos libros; OJO: recuerda que codLibro es una clave primaria, es decir, no puede haber dos libros con el mismo código de libro), AVG (nos proporciona la media), MAX (devuelve el valor máximo) y MIN (el valor mínimo).

Sin embargo, observa que nuestro ejemplo dista mucho de ser perfecto, en el ejemplo que hemos mostrado *insert into library values (1, ‘Biblioteca de Málaga’, 3)* tenemos que indicar que se trata de la Biblioteca de Málaga por cada libro que esta adquiera. Esto es muy redundante y puede producir inconsistencias, suponte que te confundes y escribes: *insert into library values (2, ‘Biblioteca de Málaga’, 3).* ¿Pero 2 no era el código de la Biblioteca de Barcelona?

El problema es que la solución requiere tres tablas: books para libros; library para las bibliotecas con sólo dos campos: codBiblioteca y nombreBiblioteca; y una que llamaremos Catálogo. Esta última tabla vincula ambas tablas con dos campos: codBiblioteca y codLibro. Estos dos campos son la clave primaria de Catálogo, así como dos claves externas: codBiblioteca a la tabla library y codLibro a books. Observa en la figura adjunta el resultado final después de toda esta discusión, esta si es una solución bastante satisfactoria.

|  |  |
| --- | --- |
| *crearBaseDatos3*  .-crearBaseDatos3.tif | |
| Fíjate en el resultado de las tres consultas que hemos solicitado.  En la primera solicitamos que nos muestre todos los libros en todas las bibliotecas.  La segunda cuenta todos los libros que hay en todas las bibliotecas. Imagina ahora que buscamos el número de libros en cada biblioteca, ¿Cómo lo haremos? Eso es una consulta agrupada, para lo que utilizaremos la cláusula **GROUP BY**, en este caso agrupamos por bibliotecas y contamos cuantos libros tenemos en cada una. | crearBaseDatos4  .-crearBaseDatos4.tif |

* 1. Modelo entidad relación. Paso a relacional con Ferrer.

Bueno para hacerlo todavía mejor cuando se realiza un proyecto software se realiza en fases: análisis, diseño, implementación, pruebas y mantenimiento. En particular en el análisis de los datos creamos un Modelo Entidad Relación. Este modelo es de muy alto nivel y nos permite representar los datos mediante dos conceptos:

* **Entidad**: cualquier cosa del mundo real con existencia propia, independiente (p.e. Alumno, Matrícula, Casa, Libro, etc.).
* **Relaciones** entre dichas entidades. Estas relaciones tienen distintas cardinalidades, es decir, pueden ser de 1 a 1 (Alumno, Matrícula, es decir, un alumno sólo puede tener una matrícula en la Universidad), 1 a N (Alumno, Libro, en otras palabras, un alumno tiene muchos libros, pero un libro sólo puede pertenecer a un alumno o propietario), N a M (Casa, Persona; una casa puede pertenecer a varias personas y una persona puede tener varias casas).

En la siguiente fase de desarrollo este modelo conceptual lo traducimos al modelo relacional que es el que realmente los sistemas gestores de base de datos (MySQL en particular) entienden. Existen herramientas CASE (herramientas de apoyo al desarrollo software) que nos simplifican este trabajo. Un ejemplo de herramienta CASE es Ferret.

Ferret es una herramienta de ingeniería libre, cuya página principal es <http://www.gnuferret.org/>, que nos permite modelar nuestro proyecto en Entidad Relación y luego pasar dicho modelo a relacional, proporcionándonos incluso las consultas SQL para crear las tablas.

|  |  |
| --- | --- |
| Se instala simplemente escribiendo en la consola *sudo apt-get install ferret*.  Lo siguiente que debes hacer es crear un nuevo proyecto haciendo clic en el botón **new**.  aviso  .-Aviso.tif  Ten en cuenta que la versión es la 0.6. | *ferret*  .-ferret.tif |
| El siguiente paso es indicar: el nombre del proyecto, el del fichero (haz clic en **select file** para indicar el nombre y su ubicación) así como el de los autores.  Luego haz clic en la pestaña **ER Diagram** para empezar a dibujar nuestro modelo entidad relación. | *ferret2*  .-ferret2.tif |
|  |  |
| Observa el diagrama, hemos creado dos entidades: books y library con una relación N:M entre ellas: Catalogo. Esta relación es N:M, pues un mismo libro puede estar en varias bibliotecas y una biblioteca puede tener muchos libros; ésta será una tabla en el modelo relacional así como las otras entidades: books y library.  En el panel inferior cambias el nombre de las entidades así como añades los atributos. En **New Attributes**, escribe el nombre en el cuadro de texto **Name**, elige el tipo de atributo: INT, VARCHAR 16, etc., indica si son claves o no (**is key**) y pulsa **Add.** | *ferret3*  .-ferret3.tif  Análogamente debes modelar las relaciones entre las entidades. En este pantallazo puedes observar que hemos definido el nombre de la relación e indicado su cardinalidad. |
| Finalmente, haz clic en la pestaña **Output** (Salida) e indica el formato de salida, en particular selecciona **mysql** (SQL para MySQL).  Comprueba como te ha generado las consultas SQL para crear tus tablas. Además es interactivo, si observas problemas, vuelve a la pestaña **ER Diagram**, realiza las modificaciones pertinentes, y haz clic en el botón **update** (actualizar) de esta ventana para crear las nuevas instrucciones SQL. | *ferret4*  .-ferret4.tif |

* 1. Backup y restauración de la base de datos.

|  |  |
| --- | --- |
| aviso  .-aviso.tif | Puedes tener todos los mecanismos de seguridad que quieras: servidores de bases de datos en ordenadores “privados” de la Intranet, un complejo mecanismo de seguridad con usuarios, privilegios, vistas, etc., protección contra incendios (extintores, detectores de humos), agua e inundaciones, protección física (sistemas antirrobos, acceso controlado por personal de seguridad). Pero antes o después todo petará en cuyo caso tu mejor defensa es la socorrida COPIA DE SEGURIDAD. |
| procedimiento  .-procedimiento.tif | Vamos a realizar los siguientes pasos para comprobar que todo funciona perfectamente:   1. Realizamos un backup de nuestra base de datos 2. La eliminamos 3. La restauramos y realizamos una consulta para comprobar que todo está como lo habíamos dejado. |

1. El primer paso es realizar un backup de nuestra base de datos, esto se realiza con la siguiente sentencia:

*joe@joe-desktop:$ mysqldump –u joe –p myBaseDatos > copia.sql*

Es decir, realiza un volcado de myBaseDatos en el fichero copia.sql con el usuario joe. Te solicitará la contraseña.

1. El segundo es bastante más fácil. Entra en mysql: *mysql –u joe –p;* Elimina la base de datos, **¿Quién dijo miedo habiendo hospitales?**: *drop database myBaseDatos*.
2. Luego hay que crear la base de datos nuevamente: *create database myBaseDatos*; en caso de desastre quizás tu usuario también haya sucumbido, agrega el usuario *create user joe;* ve a tu base de datos: *use myBaseDatos;* ejecuta el script SQL copia de seguridad: *mysql>\. copia.sql*, y comprueba que todo funciona: *select \* from books;* **Bueno, si no funciona, la culpa es de Javier Sánchez, su teléfono es 952…. y bla bla bla…**

Por supuesto esto es sólo una posibilidad, podrías querer hacer una copia de seguridad de todas las bases de datos que tengas en tu gestor MySQL, *--all-databases*, observa que dos guiones preceden a **all**. También puedes hacer un copia de seguridad de varias bases de datos en la misma sentencia (*mysqldump [opciones]- - databases myBaseDatos1 myBaseDatos2 …*), sólo de ciertas tablas de la base de datos (*mysqldump [opciones] myBaseDatos myTabla1 myTabla2 …* ), etc. La opción opt, significa que utilice las opciones por defecto, por ejemplo, puedes ejecutar: *sudo mysqldump - - opt* (dos guiones preceden a opt) *myBaseDatos –p > copia.sql*.

Por tanto, en vez de seleccionar sólo la base de datos myBaseDatos podrías optar por escribir - - all-databases para hacer un backup de todas las bases de datos o incluso si la copia de seguridad es muy voluminosa, podría serte útil comprimir el resultado con gzip:

*joe@joe-desktop:$ mysqldump –u joe myBaseDatos > copia.sql*

*gzip copia.sql*

O mejor aún, puedes hacer las dos sentencias en una sola*:*

*joe@joe-desktop:$ mysqldump –u joe –p myBaseDatos | gzip > copia.sql.bz*

Recuerda que después tendrás que descomprimirlo con *gzip –d copia.sql.*

* 1. Una alternativa potente PostgreSQL y PgAdmin para facilitarnos el trabajo.

PostgreSQL es un servidor open source (de código abierto) de base de datos relacional orientado a objetos, véase los enlaces siguientes para una información más detallada: <http://es.wikipedia.org/wiki/Postgresql> y <http://www.postgresql.org/>. PostgreSQL puede ser administrado por consola (psql) o más fácilmente mediante un entorno visual con PgAdmin.

La última versión 8.3 se encuentra en los repositorios, por lo que para instalarlo bastará con escribir en la consola la siguiente línea: *sudo apt-get install postgresql-8.3 pgadmin3*. Lo instalamos junto al entorno administrativo gráfico.

|  |  |
| --- | --- |
| idea  .-idea.tif | Una buena idea antes de instalar ningún paquete es comprobar la versión que disponemos en el repositorio, por ejemplo escribiendo en la consola *apt-cache search* *postgresql* y comprobando en la página Web que tenemos disponible la última versión estable. |

Una guía más detallada sobre la instalación la puedes encontrar en inglés en: <http://www.ubuntugeek.com/howto-setup-database-server-with-postgresql-and-pgadmin3.html>. Aquí vamos a seguir los pasos imprescindibles, pero para afinar más consulta la dirección indicada.

|  |  |  |
| --- | --- | --- |
| Lo primero es cambiar la contraseña del administrador del servidor de base de datos, observa la figura.  3. Luego, cambiamos la contraseña del usuario postgres a la misma que indicamos anteriormente, en el paso 2, en nuestro sistema Linux (Ubuntu). | 1. Deshabilitamos la contraseña que tuviera por defecto: *sudo passwd –d postgress* (como root) 2. Agregamos nuestra contraseña (supercoco): *su postgres –c passwd*.   *postgres*  .-postgres.tif | |
| Entramos ahora en el servidor (4) y crearemos una base de datos (5). | 1. *su postgres –c psql* 2. *createdb myBaseDatos* | |
| Podríamos seguir desde consola, pero vamos a simplificar y utilizaremos nuestra interfaz gráfica. Puedes lanzarla desde **Aplicaciones**, **Herramientas del Sistema**, **pgAdminIII**.  Desde **Archivo**, **Añadir Selección** o desde el primer icono en forma de enchufe, nos conectamos. Tenemos que suministrar los siguientes datos: **Nombre** (pequeña descripción), **Servidor** (localhost o dirección IP), **Nombre de Usuario** y **Contraseña**, el resto de campos puedes dejar las opciones por defecto. | postgres2  .-postgres2.tif  Hemos entrado en la base de datos por defecto, por supuesto esto es solo una iniciación. Además, si quieres acceder desde un cliente en otro ordenador distinto tendrás que cambiar el archivo de configuración: *sudo gedit /etc/postgresql/8.3/main/postgresql.conf*, y cambiar estas líneas a los valores que te indicamos: **listen\_addresses=’\*’, password\_encryption = on**, así como *sudo gedit /etc/postgresql/8.3/main/pg\_hba.conf* y en la última línea indica a quien vas a permitir el acceso 192.168.0.7 (sólo a una máquina) o 192.168.0.0 (a todas las máquinas de la red). | |
| Observa la interfaz que presenta pgAdminIII una vez que te has conectado al servidor.  Comprueba que en **Servidores** aparecen los servidores con los que te has conectado, lo habíamos llamado en el ejemplo **Conexión con mi servidor**.  Vamos a comenzar creando nuestra tabla Books, tal como hicimos con MySQL. | postgres3  .-postgres3.tif | |
| Para crear una nueva tabla conviene saber que ésta no es más que un objeto de un esquema público de postgres. Luego situándote en Bases de Datos > postgres > Catálogos > Esquemas > public haz clic en **Nuevo Objeto**, **Nueva Tabla…** tal como aparece en la figura.  Hemos utilizado la base de datos por defecto, podríamos haber creado una desde Bases de Datos. Posiciónate en Servidores>Conexión con mi servidor>Bases de Datos y haz clic en **Nueva Base de Datos…** | postgres4  .-postgres4.tif  Observa que el menú contextual mostrará un conjunto de opciones en función de donde estés posicionado, para crear una tabla tienes que situarte en los esquemas o definiciones de tu base de datos. | |
|  | | |
| Puedes ver en la figura adjunta el diálogo que te aparece para crear la nueva tabla.  Debes indicar el **Nombre** (Books), quien es el **Propietario** (por defecto, postgres) y luego en las distintas pestañas defines las columnas (los distintos atributos de la tabla), las restricciones (típicamente la claves primarias), los privilegios y en la última pestaña te indicará la sentencia SQL que se creará con tus indicaciones.  Cuando hayas escrito el nombre y el propietario, haz clic en la pestaña **Columnas** y luego en el botón **Añadir**. | | postgres5  .-postgres5.tif |
| Fíjate que estamos definiendo el código del libro (CodLibro), es de tipo entero (integer). Sigue añadiendo el resto de columnas o atributos, en nuestro ejemplo teníamos tres columnas: el código, el nombre del libro (nombreLibro) y el nombre del autor o autores (nombreAutor) que los hemos definido como de tipo text, texto.  Observa que además puedes indicar un valor por defecto, si aceptas o no valores nulos (una clave primaria no puede tener valores nulos), añadir comentarios descriptivos, etc.  Haz clic en **Aceptar** cuando hayas terminado de definir el atributo y luego en **Añadir** para terminar de definir los campos de la tabla. | | postgres6  .-postgres6.tif |
| Luego debemos de configurar las restricciones que vamos a definir en nuestra base de datos. Una de las más importantes es la restricción de clave primaria.  Haz clic en la pestaña **Restricciones**, puedes elegir que tipo de restricciones quieres añadir:   * **Clave Primaria**: selecciona esta. * **Clave Ajena**: para las claves externas. * **Único**: indica que un atributo no puede repetirse. * **Check**: para exigir que se satisfaga una determinada condición, por ejemplo, en una tabla de notas de tipo entero que no haya notas negativas: NotaParcial>=0.   Te aparecerá otro cuadro de dialogo con dos pestañas. En la primera, **Propiedades**, indicas el nombre que quieres utilizar para describir la restricción, por ejemplo escribe clave\_primaria. En la segunda pestaña **Columnas**, la que aparece en la figura, selecciona CodLibro que será la columna o atributo que forma nuestra clave primaria. | | postgres7  .-postgres7.tif |
| Finalmente observa en la pestaña **SQL** nuestra consulta SQL conforme a nuestros requerimientos.  Existen dos consultas SQL: **CREATE TABLE** que crea la base de datos y **ALTER TABLE** que la modifica para indicar que su propietario es postgres. Además podemos comprobar que existe una restricción **CONSTRAINT** relativa a nuestra clave primaria.  Haz clic en **Aceptar**, para que se haga efectiva la creación de dicha tabla y para empezar a introducir tuplas o datos en ella. | | postgres8  .-postgres8.tif |
| Sitúate en la tabla recién creada (Servidores, Conexión con mi servidor, Bases de Datos, postgres, Esquemas, public, Tablas, Books) y con el botón derecho selecciona: **Ver Datos**, **Ver Todas las Filas**.  Deberás ver sólo una tabla vacía, ahora puedes empezar a introducir tuplas.  También puedes en cualquier momento modificar cualquier objeto, por ejemplo, la tabla que hemos creado, selecciónala y haz clic en **Propiedades**. | | postgres9  .-postgres9.tif  Como has comprobado el propio entorno te facilita consultar las tablas sin escribir ningún código SQL. Observa además el embudo que te permite ordenar y filtrar los resultados de la búsqueda. |