Javascript

Introducción

Origen

JavaScript es un lenguaje interpretado, desarrollado por Netscape inicialmente para sus navegadores.

Actualmente es compatible con los navegadores más utilizados: Netscape Communicator, Microsoft Internet Explorer y Opera.

Conceptos Básicos

Se trata de un lenguaje de tipo script compacto, basado en objetos y guiado por eventos.

JavaScript es un lenguaje interpretado, es decir, no requiere compilación. Es utilizado principalmente en páginas Web con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

JavaScript es case sensitive, es decir, diferencia mayúsculas y minúsculas.

Aunque la similitud del nombre puede inducir a creer que JavaScript es equivalente a Java, la verdad es que no tienen prácticamente nada en común. Java ha sido desarrollado para construir grandes aplicaciones distribuidas con el objetivo de máxima fiabilidad. JavaScript se creó con el objetivo de ser un lenguaje sencillo de aprender y utilizar, ideal para desarrollar pequeños programas que corren en un navegador.

Dónde y cómo incluir Javascript

Existen distintos modos de incluir lenguaje JavaScript en una página.

La forma más frecuente de hacerlo es utilizando la directiva <script> en un documento HTML (se pueden incluir tantas directivas <script> como se quiera en un documento).

El formato es el siguiente:

```
<script language="Javascript 1.3">
```

El atributo lenguaje hace referencia a la versión de JavaScript que se va a utilizar en dicho script. Otro atributo de la directiva script es src, que puede usarse para incluir un archivo externo que contiene JavaScript y que quiere incluirse en el código HTML.

```
<script language="JavaScript" src ="archivo.js"> </script>
```

El archivo externo simplemente es un archivo del texto que contiene código JavaScript, cuyo nombre acaba con la extensión js.

Puede incluirse también código JavaScript como respuesta a algún evento:

<input type="submit" onclick="alert('Acabas de hacer click');return false;" value="Click">

Nota: Los scripts pueden incluirse como comentarios para asegurar que su código no es "visto" por navegadores viejos que no reconocen JavaScript y así evitar errores.

Tipos de Datos

Javascript reconoce seis tipos de valores diferentes: numéricos, lógicos, objetos, cadenas, nulos e indefinidos.

JavaScript, al igual que PHP, tiene la peculiaridad de ser un lenguaje débilmente tipado, esto es, una variable puede cambiar de tipo durante su vida. Por ejemplo, uno puede declarar una variable que ahora sea un entero y más adelante sea una cadena.

A diferencia de otros lenguajes, y como ya hemos visto, en Javascript no es necesario declarar las variables especificando el tipo de dato que contendrán. Será el propio intérprete el que le asignará el tipo apropiado.

Estructuras de Control de Flujo

Ciclos repetitivos

Javascript soporta los mismo ciclos repetitivos que PHP (for, while y do-while) y su sintaxis es exactamente la misma.

También soporta las sentencias break y continue, al igual que PHP.

Sentencias condicionales

Javascript soporta las mismas sentencias condicionales que PHP (if, else, else if, swith) y su sintaxis es exactamente la misma.

Ventanas del Navegador

Ventana Alert

Crea una caja de diálogo con un ícono de peligro amarillo, un botón 'Aceptar' y un texto definido por el parámetro enviado a la función.

```
<script>
alert("Ha ocurrido un error");
```

</script>

Las alertas serán útiles para transmitir información al usuario, tal como errores ocurridos en la navegación, problemas en el rellenado de un formulario, entre otros.

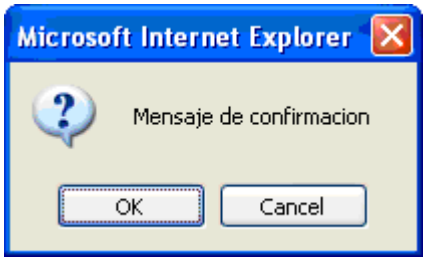
Ventana Confirm

Crea una caja de confirmación con un ícono de interrogación, botones “Aceptar” y “Cancelar” y un texto definido por el parámetro enviado a la función.

Devuelve 1 cuando el usuario abandona el diálogo pulsando “Aceptar” y 0 si lo hace pulsando “Cancelar” o el aspa de cerrar.

```
<script>
if(confirm('¿Seguro que ha leído las condiciones del contrato?'))this.form.submit();
</script>
```

Será útil para recibir información del usuario en tiempo de ejecución (al pulsar un botón o al pasar el mouse por un lugar)



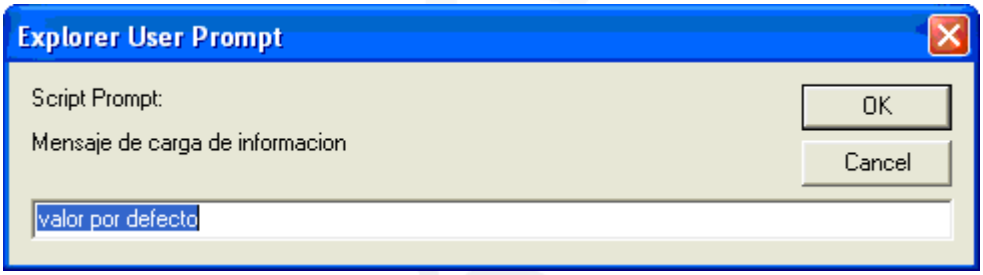
Ventana Prompt

Muestra un diálogo de campo de formulario con botones “Aceptar” y “Cancelar”, un texto definido por el primer parámetro enviado a la función y un input de texto con valor predeterminado, definido por el segundo parámetro.

La función devuelve el valor insertado en el campo de formulario si el usuario pulsa en “Aceptar” o null si pulsa “Cancelar” o el aspa de cerrar.

```
<script>
nombre = prompt('Introduce tu nombre','[ nombre del usuario ]');
</script>
```

Prompt será útil sobre todo para recoger datos del usuario para utilizar en nuestro script en tiempo de ejecución.



Funciones de Validación

Función parseInt()

Esta función recibe un número escrito como una cadena de caracteres y un número que indica una base. Puede recibir otros tipos de variables, dado que las variables no tienen tipo en Javascript, pero se suele utilizar pasándole un string para convertir la variable de texto en un número.

Las distintas bases que puede recibir la función son 2, 8, 10 y 16. Si no se le pasa ningún valor como base, la función interpreta que la base es decimal. El valor que devuelve la función siempre tiene base 10, de modo que si la base no es 10 convierte el número a esa misma base antes de devolverlo.

```
document.write (parseInt("34"))
```
Devuelve el numero 34

```
document.write (parseInt("101011",2))
```
Devuelve el numero 43

```
document.write (parseInt("34",8))
```
Devuelve el numero 28

```
document.write (parseInt("3F",16))
```
Devuelve el numero 63

Función parseFloat()

Convierte su argumento (la cadena) e intenta retornar un número de punto flotante. Si encuentra un caracter diferente a un signo (+ o -), un número (0-9), un punto decimal o un exponente, entonces retorna el valor hasta antes del punto e ignora este caracter y todos los sucesivos. Si el primer caracter no puede ser convertido a un número, retorna "NaN" (no es número).

```
parseFloat(cadena)
```

Función isNaN()

La función isNaN evalúa un argumento para determinar si éste no es un número.

Esta función devuelve un boleano dependiendo de si lo que recibe es un número o no. Lo único que puede recibir es un número o la expresión NaN.

La función suele trabajar en combinación con la función parseInt o parseFloat, para saber si lo que devuelven estas dos funciones es un número o no.

```
miInteger = parseInt("A3.6")
isNaN(miInteger)
```

Funciones definidas por el usuario

¿Qué es una función?

A la hora de hacer un programa existen determinados procesos que se pueden concebir de forma independiente y que son más sencillos de resolver que el problema entero. Además, estos procesos suelen ser realizados repetidas veces a lo largo de la ejecución del programa. Los mismos se pueden agrupar en una función definida para que no se tenga que repetir una y otra vez ese código en los scripts, sino que simplemente se llama a la función y ella se encarga de hacer todo lo que debe.

Así que se puede ver una función como una serie de instrucciones o como un bloque de código que se engloba dentro de un mismo proceso. Este se podrá ejecutar luego desde cualquier otro sitio con sólo llamarlo.

Las funciones se utilizan constantemente, no sólo las que se escriben, sino también las que ya están pre-definidas por el lenguaje como hemos visto en el punto anterior, pues todos los lenguajes de programación tienen un montón de funciones para realizar procesos habituales, como por ejemplo, obtener la hora, imprimir un mensaje en la pantalla o convertir variables de un tipo a otro.

¿Cómo se escribe una función?

La sintaxis es exacta a PHP.

Una función se debe definir con una sintaxis especial que vamos a conocer a continuación.

```
function nombrefuncion (){
  instrucciones de la función
  ...
}
```

Simplemente se escribe en la página un texto; es una función tan sencilla que el ejemplo no expresa suficientemente el concepto de función, pero ya veremos otras más complejas. Las etiquetas H1 no se escriben en la página, sino que son interpretadas como el significado de la misma (En este caso que escribimos un encabezado de nivel 1). Como se está escribiendo en una página web, al poner etiquetas HTML se interpretan como lo que son.

¿Cómo llamar a una función ?

Para ejecutar una función se la tiene que llamar en cualquier parte de la página. Con eso se conseguirá que se ejecuten todas las instrucciones que tiene la función entre las dos llaves. Para ejecutar la función se utiliza su nombre seguido de los paréntesis.

NombreDeLaFuncion()

Base de Datos MySQL

Entorno MySQL

¿Qué es MySQL?

MySQL es un sistema administrador para bases de datos relacionales basadas en la arquitectura cliente/servidor.

Transact-SQL es el lenguaje que emplea para enviar peticiones entre cliente y servidor.

Es un lenguaje exclusivo de MySQL, pero basado en el lenguaje SQL estándar, siendo utilizado por casi todos los tipos de bases de datos relacionales que existen.

MySQL es software de fuente abierta. Esto significa que es posible para cualquier persona usarlo y modificarlo. Se puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades.

MySQL usa el GPL (GNU General Public License) para definir qué puede y qué no puede hacer con el software en diferentes situaciones. Si no se ajusta al GLP o requiere introducir código MySQL en aplicaciones comerciales se pude comprar una versión comercial licenciada.

Bases de Datos

Una base de datos es una colección estructurada de datos.

Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto monto de información en un red corporativa.

Los administradores de bases de datos juegan un papel central como aplicaciones independientes o como parte de otras aplicaciones.

Tablas

Las tablas son objetos compuestos por una estructura (conjunto de columnas) que almacenan información interrelacionada (filas) acerca de algún objeto en general.

Las tablas tienen un sólo nombre y es único en toda la base datos.

Están compuestas por registros y columnas.

Los registros y columnas pueden estar en diferentes órdenes.

Una base de datos contiene muchas tablas. Cada tabla almacena información.

MySQL es un sistema de administración relacional de bases de datos

Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo.

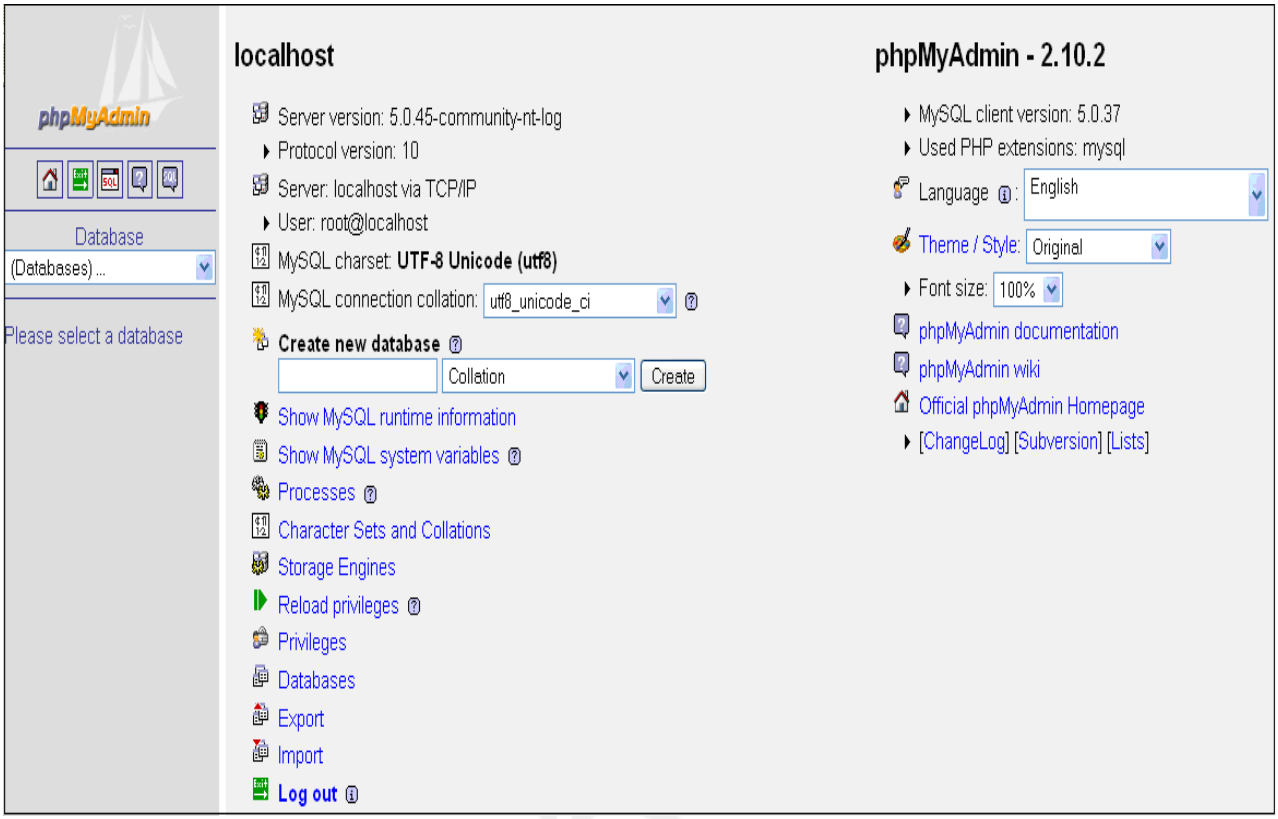
Esto permite velocidad y flexibilidad.

Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

PHPMyAdmin

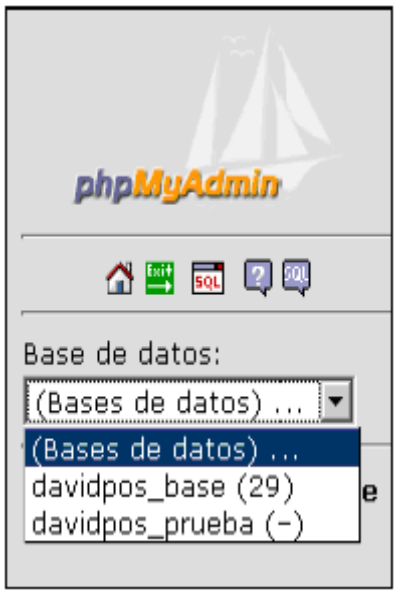
Introducción

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas Web utilizando Internet. Actualmente puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 50 idiomas. http://www.phpmyadmin.net/



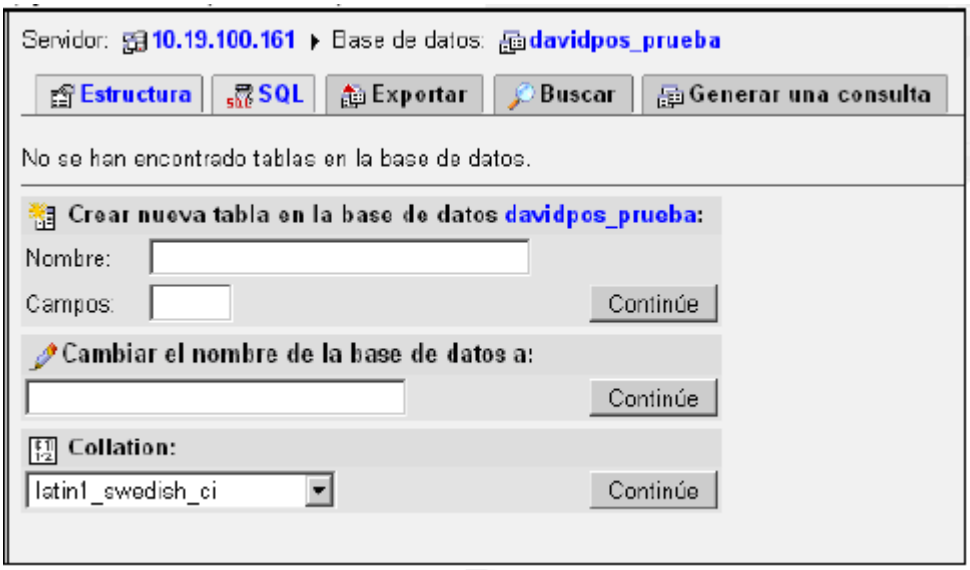
La pantalla inicial (imagen superior) otorga la opción tanto de seleccionar una base de datos ya creada desde el menú de la izquierda como de crear una nueva base de datos (Esta opción está deshabilitada, por lo que se deberá crear la base manualmente):

Selección de una base de datos



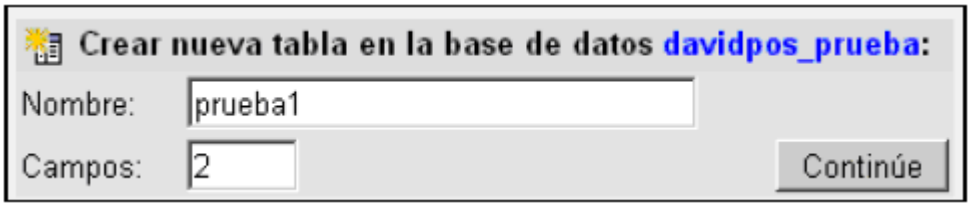
Una vez seleccionada se procederá a gestionar la base de datos. A partir de aquí se pueden crear tablas, sentencias SQL, insertar datos en una tabla ya creada anteriormente, buscar registros, modificarlos, etc.

Crear una Tabla

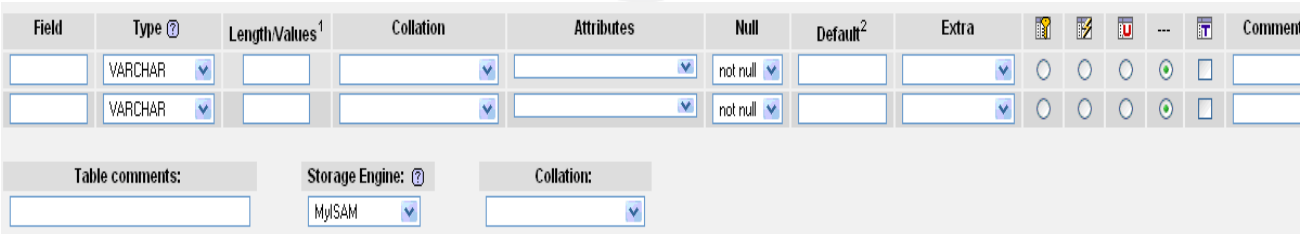


En la pestaña de estructura se podrán crear tablas dentro de la base de datos, modificar el nombre de la base, y el cifrado (collation).

Para crear una nueva tabla se introducirá el nombre que se le quiere asignar y el número de campos.



Una vez pulsado el botón de continuar, se accederá a la configuración de los campos de la tabla



Dispone las siguientes opciones para configurar cada campo:

1. Nombre del campo

2. Tipo de campo (texto, numérico, fecha/hora, etc.)

3. Longitud/Valores

4. Collation (cifrado)

5. Atributos

6. Nulo

7. Predeterminado (valor predeterminado)

8. Extras

9. Clave primaria / Índice / Único

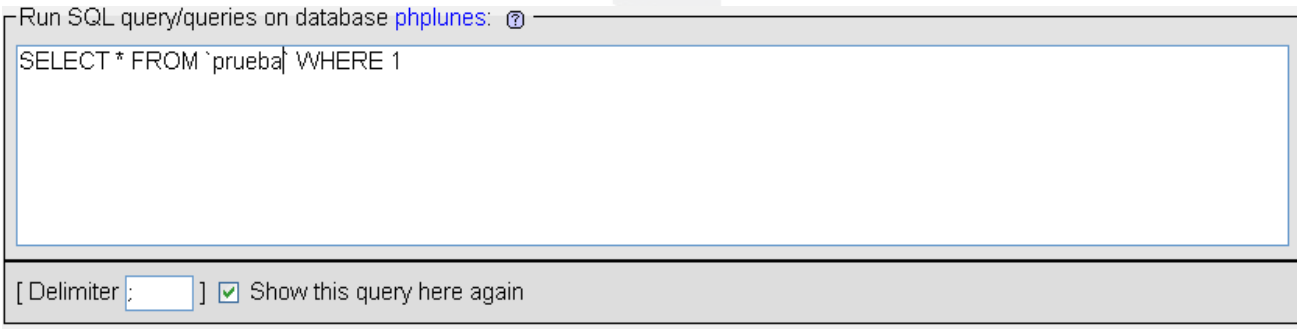
Como opciones de la tabla se tienen las siguientes:

1. Comentarios de la tabla.

2. Tipo de tabla

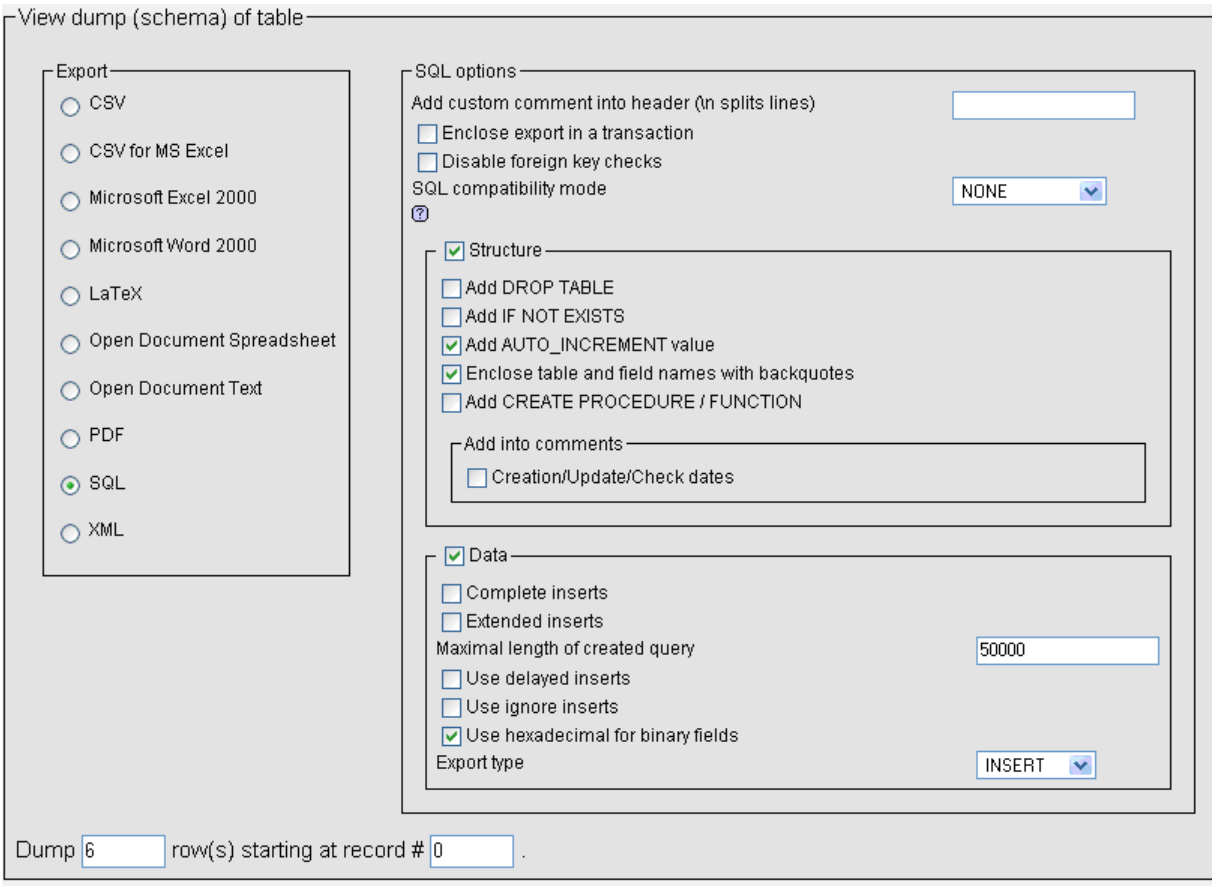
3. Collation

Consultas SQL



Mediante consultas SQL también se podrá realizar cualquier operación de las tablas, insertar, modificar o eliminar registros, realizar búsquedas, etc.

Desde esta pestaña se pueden exportar de una forma sencilla bases de datos.



El lenguaje SQL

Introducción

Existen dos lenguajes para el manejo de bases de datos:

DDL (Data Definition Language): Lenguaje de definición de datos. Es el lenguaje que se usa para crear bases de datos y tablas y para modificar sus estructuras, así como los permisos y privilegios.

DML (Data Manipilation Language): Lenguaje de manipulación de datos. Es el que se utiliza para modificar y obtener datos desde las bases de datos

¿Qué es DDL?

DDL: Data Definition Language

Involucra los comandos necesarias para crear y eliminar una tabla.

Permite crear claves primarias, índices y restricciones.

Los comandos mas conocidos son:

CREATE TABLE – crea una nueva base de datos

ALTER TABLE – modifica una base de datos existente

DROP TABLE – elimina una base de datos

CREATE INDEX – crea un índice

DROP INDEX – elimina un índice

Creación de una Tabla

La sentencia CREATE TABLE creará una tabla con las columnas que indiquemos.

Crearemos como ejemplo una tabla que permitirá almacenar nombres de personas y sus fechas de nacimiento.

CREATE TABLE gente (nombre VARCHAR(40), fecha DATE);

La sintaxis para definir columnas es:

nombre_col tipo [NOT NULL | NULL] [DEFAULT valor_por_defecto]

[AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']

[definición_referencia]

Al definir cada columna se podrá decidir si podrá o no contener valores nulos.

Ejemplo:

```
CREATE TABLE `alumnos` (
  `idAlumno` int(11) unsigned NOT NULL auto_increment,
  `nombre` varchar(40) NOT NULL default '',
  `apellido` varchar(100) NOT NULL default '',
  `sexo` enum('f','m') NOT NULL default 'm',
  `telefono` varchar(30) NOT NULL default '',
  `mail` varchar(50) NOT NULL default '',
  `fechaNac` date NOT NULL default '0000-00-00',
  PRIMARY KEY  (`idAlumno`) )
```

¿Qué es DML?

DML: Data Manipulation Language

Involucra los comandos necesarios para hacer consultas, inserciones, modificaciones y eliminaciones.

Los comandos utilizados son:

SELECT – obtiene información de una base de datos

UPDATE - actualiza información de una base de datos

DELETE – elimina información de una base de datos

INSERT INTO – inserta información en una base de datos

Eliminar una Tabla

DROP TABLE nombre_de_la_tabla

SELECT

Se utiliza para seleccionar información de una tabla.

Para seleccionar todas la columnas se utiliza el * (asterisco)

La cláusula WHERE se utiliza para establecer un criterio de búsqueda

Sintaxis #1:

SELECT *

FROM tabla_nombre

Sintaxis #2:

SELECT nombre_columna(s)

FROM tabla_nombre

Sintaxis #3:

SELECT nombre_columna(s)

FROM tabla_nombre

WHERE campo1 = valor1INSERT

Para insertar datos en una tabla utilizamos la orden INSERT.

Los valores que se van a introducir van en comillas simples (‘),

Código:
INSERT INTO nombre_de_la_tabla (
Columna1, columna 2, ….)
VALUES (
Valor1, valor2, ….);

INSERT INTO gente VALUES ('Fulano','1974-04-12');

INSERT INTO gente VALUES ('Mengano','1978-06-15');

UPDATE

Se pueden modificar valores de las filas de una tabla usando la sentencia UPDATE

UPDATE tbl_name

SET col_name1=expr1 [, col_name2=expr2 ...]

[WHERE where_definition]

[LIMIT row_count]

Se puede, del mismo modo, actualizar el valor de más de una columna, separándolas en la sección SET mediante comas.

La primera es mediante la cláusula WHERE. Usando esta cláusula se puede establecer una condición. Sólo las filas que cumplan esa condición serán actualizadas.

La cláusula LIMIT. Esta cláusula permite especificar el número de filas a modificar. Esta cláusula se puede combinar con WHERE, de modo que sólo las 'n' primeras filas que cumplan una determinada condición se modifiquen.

DELETE

Para eliminar filas se usa la sentencia DELETE.

DELETE FROM table_name

[WHERE where_definition]

[LIMIT row_count]

Para eliminar filas se usa la sentencia DELETE.

Consultas con SQL SELECT

Para extraer los datos y que nos los presente en pantalla se utiliza la orden SELECT

```
SELECT [ALL | DISTINCT | DISTINCTROW]
  expresion_select,...
  FROM referencias_de_tablas
  WHERE condiciones
  [GROUP BY {nombre_col | expresion | posicion}
     [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING condiciones]
  [ORDER BY {nombre_col | expresion | posicion}
     [ASC | DESC] ,...]
  [LIMIT {[desplazamiento,] contador | contador OFFSET desplazamiento}]
```

Mediante la sentencia SELECT es posible hacer una proyección de una tabla seleccionando las columnas de las que se quieren obtener datos. En la sintaxis que se ha mostrado la selección de columnas corresponde con la parte "expresion_select". En el ejemplo anterior se ha usado '*', que quiere decir que se muestran todas las columnas.

Las expresiones_select no se limitan a nombres de columnas de tablas, sino que pueden ser otras expresiones, incluso aunque no correspondan a ninguna tabla:

```
SELECT SIN(3.1416/2), 3+5, 7*4
```

También se pueden aplicar funciones sobre columnas de tablas y usarlas en expresiones para generar nuevas columnas:

```
SELECT nombre, fecha, DATEDIFF(CURRENT_DATE(),fecha)/365 FROM gente;
```

Es posible asignar un alias a cualquiera de las expresiones select. Esto se puede realizar utilizando la palabra AS, aunque esta palabra es opcional:

```
SELECT nombre, fecha, DATEDIFF(CURRENT_DATE(),fecha)/365 AS edad
```

La sentencia que se ha usado asume el valor por defecto (ALL) para el grupo de opciones ALL, DISTINCT y DISTINCTROW. En realidad sólo existen dos opciones, ya que las dos últimas (DISTINCT y DISTINCTROW) son sinónimos.

```
SELECT DISTINCT fecha FROM gente;
```

SELECT permite usar condiciones como parte de su sintaxis, es decir, para hacer selecciones. Concretamente mediante la cláusula WHERE.

```
SELECT * FROM gente WHERE nombre="Mengano";
```

En una cláusula WHERE se puede usar cualquier función disponible en MySQL.

También se puede aplicar lógica booleana para crear expresiones complejas. Se dispone de los operadores AND, OR, XOR y NOT.

Consultas Agrupadas - GROUP BY

Es posible agrupar filas en la salida de una sentencia SELECT según los distintos valores de una columna usando la cláusula GROUP BY. Esto, en principio, puede parecer redundante ya que se podría hacer lo mismo usando la opción DISTINCT. Sin embargo, la cláusula GROUP BY es más potente.

```
SELECT fecha FROM gente GROUP BY fecha
```

La primera diferencia que se observa es que si se utiliza GROUP BY la salida se ordena según los valores de la columna indicada. En este caso, las columnas aparecen ordenadas por fechas.

Otra diferencia es que se eliminan los valores duplicados aún si la proyección no contiene filas duplicadas.

```
SELECT nombre,fecha FROM gente GROUP BY fecha
```

La diferencia principal es que el uso de la cláusula GROUP BY permite usar funciones de resumen o reunión. Por ejemplo, la función COUNT(), que sirve para contar las filas de cada grupo

```
SELECT fecha, COUNT(*) AS cuenta FROM gente GROUP BY fecha;
```

Esta sentencia muestra todas las fechas diferentes y el número de filas para cada fecha.

Existen otras funciones de resumen o reunión como MAX(), MIN(), SUM(), AVG(), STD(), VARIANCE()...

Estas funciones también se pueden usar sin la cláusula GROUP BY siempre que no se proyecten otras columnas.

La cláusula HAVING permite hacer selecciones en situaciones en las que no es posible usar WHERE. Veamos un ejemplo completo:

Operadores Lógicos

A	B	A AND B
falso	falso	falso
falso	verdadero	falso
verdadero	falso	falso
verdadero	verdadero	verdadero
falso	NULL	falso
NULL	falso	falso
verdadero	NULL	NULL
NULL	verdadero	NULL

A	B	A OR B
falso	falso	falso
falso	verdadero	verdadero
verdadero	falso	verdadero
verdadero	verdadero	verdadero
falso	NULL	NULL
NULL	falso	NULL
verdadero	NULL	verdadero
NULL	verdadero	verdadero

A	B	A XOR B
falso	falso	falso
falso	verdadero	verdadero
verdadero	falso	verdadero
verdadero	verdadero	falso
falso	NULL	NULL
NULL	falso	NULL
verdadero	NULL	NULL
NULL	verdadero	NULL

A	NOT A
falso	verdadero
verdadero	falso

NULL	NULL

Para crear expresiones lógicas a las que podremos aplicar el álgebra de Boolean, se dispone de varios operadores de comparación. Estos operadores se aplican a cualquier tipo de columna: fechas,cadenas, números, etc., y devuelven valores lógicos: verdadero o falso (1/0).

Si uno o los dos valores a comparar son NULL, el resultado es NULL, excepto con el operador <=>, de comparación con NULL segura.

El operador <=> funciona igual que el operador =, salvo que si en la comparación una o ambas de las expresiones es nula el resultado no es NULL. Si se comparan dos expresiones nulas, el resultado es verdadero:

Operador	Descripción
<=	Menor o igual
<	Menor
>	Mayor
>=	Mayor o igual

Los operadores IS NULL e IS NOT NULL sirven para verificar si una expresión determinada es o no nula.

Entre los operadores de MySQL hay uno para comprobar si una expresión está comprendida en un determinado rango de valores. La sintaxis es:

BETWEEN mínimo AND máximo

NOT BETWEEN mínimo AND máximo

Los operadores IN y NOT IN sirven para averiguar si el valor de una expresión determinada está dentro de un conjunto indicado.

IN (<expr1>, <expr2>, <expr3>...)

NOT IN (<expr1>, <expr2>, <expr3>...)

El operador IN devuelve un valor verdadero (1) si el valor de la expresión es igual a alguno de los valores especificados en la lista. El operador NOT IN devuelve un valor falso en el mismo caso. Por ejemplo:

SELECT 10 IN(2, 4, 6, 8, 10);

El operador LIKE se usa para hacer comparaciones entre cadenas y patrones. El resultado es verdadero (1) si la cadena se ajusta al patrón y falso (0) en caso contrario. Esto ocurre tanto si la cadena como el patrón son NULL, el resultado es NULL. La sintaxis es:

LIKE <patrón> [ESCAPE 'carácter_escape']

Carácter	Descripción
%	Coincidencia con cualquier número de caracteres, incluso ninguno.
_	Coincidencia con un único carácter.

La comparación es independiente del tipo de los caracteres, es decir, LIKE no distingue mayúsculas de minúsculas salvo que se indique lo contrario (ver operadores de casting).

Como siempre que se usan caracteres concretos para crear patrones, se presenta la dificultad de hacer comparaciones cuando se deben buscar precisamente esos caracteres concretos. Esta dificultad se suele superar mediante secuencias de escape. Si no se especifica nada en contra, el carácter que se usa para escapar es '\'. De este modo, si se quiere que el patrón contenga los caracteres '%' o '_', se los hará escapar de este modo: '\%' y '_':

Como en cualquier otro lenguaje, los paréntesis se pueden usar para forzar el orden de la evaluación de determinadas operaciones dentro de una expresión. Cualquier expresión entre paréntesis adquiere mayor precedencia que el resto de las operaciones en el mismo nivel de paréntesis.

Consultas Anidadas – JOIN

Es posible hacer consultas usando varias tablas en la misma sentencia SELECT.

Esto permite realizar otras dos operaciones de álgebra relacional: el producto cartesiano y la composición.

El producto cartesiano de dos tablas son todas las combinaciones de todas las filas de las dos tablas. Cuando se utiliza una sentencia SELECT se hace proyectando todos los atributos de ambas tablas. Los nombres de las mismas se indican en la cláusula FROM, separados con comas.

La composición interna trata de un producto cartesiano restringido; las tuplas que se emparejan deben cumplir una determinada condición.

```
referencia_tabla, referencia_tabla

referencia_tabla [INNER | CROSS] JOIN referencia_tabla ON expresión_condicional
```

```
SELECT * FROM personas2, telefonos2

WHERE personas2.id=telefonos2.id;
```

La coma y JOIN son equivalentes y las palabras INNER y CROSS son opcionales.

La condición en la cláusula ON puede ser cualquier expresión válida para una cláusula WHERE. De hecho, en la mayoría de los casos, son equivalentes.

Las composiciones externas no proceden de un producto cartesiano. Por lo tanto, en estas pueden aparecer tuplas

que no aparecen en el producto cartesiano.

Para hacer una composición externa se toman las tuplas de una de las tablas una a una y se combinan con las tuplas de la otra.

Si no existe ninguna tupla en la segunda tabla que cumpla las condiciones, se combina la tupla de la primera con una nula de la segunda.

referencia_tabla LEFT [OUTER] JOIN referencia_tabla [join_condition]

referencia_tabla RIGHT [OUTER] JOIN referencia_tabla [condición]