Visualizar la estructura de una tabla

Es posible visualizar la estructura de una tabla de dos maneras: utilizando nuevas **funciones** de PHP, o aplicando nuevas *sentencias* MySQL con las funciones ya conocidas.

Lectura de resultados de sentencias MySQL

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(*sent, $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:

```
$r=mysql_query(inst,$c);
while($r=mysql_fech_row($r){
   foreach ($r as $valor){
     print $valor;
   }
}
```

o también

```
while($r=mysql_fech_row($r){
    $g[]=$t;
}
```

con lo que estaríamos creando un array bidimensional con el contenido de los resultados de cada línea.

En este caso el *primer índice* del array **$g** seguiría las normas de creación de arrays y se iría autoincrementando en cada ciclo del bucle *while*.

El ejemplo tiene desarrollados ambos métodos.

Existe una función alternativa que mejora las prestaciones de la

Ver la estructura de una tabla utilizando MySQL

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

SHOW FIELDS from *nombre de la tabla*

```
<?
# asignamos a una variable el nombre de la base de datos
$base="ejemplos";
# esta otra recoge el nombre de la tabla
$tabla="ejemplo1";
# establecemos la conexión con el servidor
$c=mysql_connect ("localhost","pepe","pepa");
# seleccionamos la base de datos
mysql_select_db ($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 que comentamos al margen
while($v=mysql_fetch_row ($resultado)){
foreach($v as $valor) {
      echo  $valor,"<br>";
     }
}


#########################################################################
#        REPETIMOS LA CONSULTA ANTERIOR USANDO AHORA LA                 #
#              función mysql_fecth_array                                 #
#########################################################################

#tenemos que VOLVER a EJECUTAR LA SENTENCIA MySQL
# porque el puntero está AL FINAL de la ultima línea
# de los resultados

$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>");
     }
}

################### la tercera posibilidad comentada
#######################################################

$resultado=mysql_query( "SHOW FIELDS from $tabla",$c);
# intentaremos explicar este doble bucle con calma
/* En los procesos anteriores a cada paso del bucle
   foreach leiamos un array, lo imprimiamos y sustituiamos
   ese array por uno nuevo

   Ahora trataremos de recoger todos esos resultados en array
   para ello usamos un array bidimensional que renueva
   el primer indice en cada ciclo del bucle while
   de ahi que pongamos el $contador para asignarle
   el primer indice
   Los segundos indices serán los valores de los indices
   del array $v que recogemos de la petición MySQL
   pero mysql_fetch_array genera dos indices para
   cada uno de los valores, uno numerico y otro asociativo
   asi que filtramos y si el indice es numérico
   guardamos en el array llamado numerico
   y si no lo es guardamos el llamado asociativo
   Tendremos separados en dos array ambos resultados
   ¿Complicado... ? Es cuestión de analizar con calma */

# pongamos el contador a cero para asegurar
# no hay variables anteriores con el mismo nombre
# y valores distintos
```

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 enlace.

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** correspondientes 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.

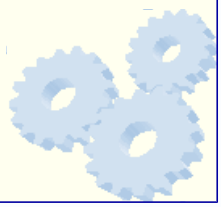
Liberando memoria

Si queremos *liberar* la parte de la memoria que contiene un identificador de resultado bastará con que insertemos la siguiente instrucción:

mysql_free_result(*$res*)

Este proceso debe ser **posterior** a la visualización de los datos.

```php
$contador=0;

while($v=mysql_fetch_array($resultado)){
    foreach ($v as $indice1=>$valor1){
        if(is_int($indice1)){
          $numerica[$contador][$indice1]=$valor1;
         }else{
           $asociativa[$contador][$indice1]=$valor1;
         }
    }
        $contador++;
  }
/* vamos a leer los array resultantes
   empecemos por el numerico que al tener
   dos indices requiere dos foreach anidados
   el valor del primero será un array
   que extraemos en el segundo */

foreach($numerica as $i=>$valor){
        foreach ($valor as $j=>$contenido){
         print ("numerico[".$i."][".$j."]=".$contenido."<br>");
            }
 }

foreach($asociativa as $i=>$valor){
        foreach ($valor as $j=>$contenido){
         print ("asociativo[".$i."][".$j."]=".$contenido."<br>");
            }
 }


# liberamos memoria borrando de ella el resultado

mysql_free_result ($resultado);

# cerramos la conexion con el servidor

mysql_close($c);
?>
```



ejemplo164.php

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:

Ver código fuente

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 existiera, 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.

Ver código fuente

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.

Ver código fuente