

Sintaxis MySQL para borrado de registros

La sintaxis MySQL para las sentencia de borrado de registros de una tabla puede contener las siguientes *cláusulas* que, al igual que ocurría en casos anteriores, pueden tener categoría de **obligatorias** u **opcionales**.

La secuencia en la que deben estar indicadas en la sentencia es idéntica al orden en que están descritas aquí.

DELETE

Tiene carácter **obligatorio**. Debe ser la **primera palabra de la sentencia** e indica a MySQL que tratamos de **borrar** uno o más *registros*.

LOW_PRIORITY

Es **opcional** e indica a MySQL que **espere** para realizar la actualización **a que terminen** las consultas del fichero (en el caso de haber alguna en proceso).

FROM

Tiene carácter **obligatorio** y debe *preceder* a la definición de la *tabla* en la que se pretende eliminar registros.

tabla

Es **obligatoria** e indica el **nombre** de la tabla en la que pretendemos efectuar el borrado o eliminación de los registros.

WHERE

Es un campo **opcional** y su comportamiento es idéntico al señalado en al mencionar el proceso de *consultas*.

LIMIT n

La opción **LIMIT** es **opcional** y propia de MySQL.

Su finalidad es **limitar** el tiempo de ejecución del comando **DELETE** ya que cuando está activada devuelve el control al *potencial cliente* después de borrar **n** registros, con lo que en procesos de borrados **muy largos** (ficheros de gran tamaño) no obliga a esperar a borrado total para proceder a la consulta de la tabla.

Cuando se utiliza esta opción, la sentencia **DELETE** *debe repetirse* hasta que el número de registros

Borrar todos los registros de una tabla

[Ver índice](#)[Búsqueda rápida](#)[Página anterior](#)[Página siguiente](#)

Ten muy presente que con esta sentencia -en la que no aparece **WHERE**- se **BORRARÁN TODOS LOS REGISTROS DE LA TABLA**.

Respecto a otras posibles opciones no difiere en nada de lo indicado en la página anterior. Simplemente habría que sustituir en aquellos script **UPDATE** por **DELETE**. Borrar un registro no es otra cosa que un caso particular de modificación.

Integridad referencial tras el borrado de una tabla

¿Recuerdas el *ejemplo* de las *pruebas de selección de astronautas*? ¿Recuerdas que las **tres tablas de puntuaciones** habían sido creadas a partir de la tabla de *datos* de los *aspirantes*? ¿Qué ocurriría si **borrásemos** uno o varios registros de una de ellas? ¿Qué ocurriría se después de crear esas tablas **añadiésemos** nuevos aspirantes a la *lista* de candidatos?

Es obvio que si **no hacemos algo para evitarlo** se perdería la **integridad referencial** - la relación **uno a uno** - entre los registros de esas tablas.

Ocurriría que **no todos** los individuos que están incluidos en una de esas tablas lo **estarían** en las demás y por tanto, al ejecutar consultas o modificaciones posteriores correríamos el riesgo de que se produjeran errores.

Esa situación **es fácilmente evitable** modificando ligeramente los *scripts* con los que se realizan los **procesos** de *altas* y *bajas*.

Bastaría con **añadirles** algunas sentencias que **cada vez que se efectúa un alta o baja** en el fichero de **datos personales** **efectúen el mismo proceso** en **todos** los demás ficheros **relacionados** con aquel.

Aquí tienes comentado el *código fuente* de la modificación añadida al script que registra los nuevos aspirantes en el fichero de *altas* de la tabla **demo4**. Con esta modificación se actualizarían automáticamente los ficheros **demodat1**, **demodat2** y **demodat3** cada vez que se añadiera un nuevo aspirante.

El formulario no requiere ninguna modificación, los cambios sólo es necesario realizarlos en el *script* que realiza la **inserción**.

[Ver código fuente](#)[Añadir un nuevo aspirante](#)

Hecho este pequeño inciso -creemos que importante y necesario - continuaremos con la referencia al **borrado de registros**.

En este ejemplo, tienes el *código fuente* de un *script* que realiza el borrado de **un registro** -mediante un formulario en el que se inserta el DNI- tanto en la tabla **demo4** como **demodat1**, **demodat2** y **demodat3** manteniendo la **integridad referencial** entre los cuatro ficheros.

[Ver script](#)[Borrar un registro](#)

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 las tablas afectadas.

pendientes de borrado sea inferior al valor de n .

Optimización de tablas

Cuando se ejecuta la sentencia DELETE -pese a que son eliminados los valores de los campos- se **conservan** las **posiciones** de los registros borrados, con lo cual **no se reduce** el **tamaño** de la tabla.

Esas *posiciones de registro* serán **utilizadas** por MySQL para *escribir* los registros que se vayan **añadiendo** después del proceso de borrado.

Para eliminar esos *registros vacíos* y **reducir** el tamaño de una tabla, MySQL dispone de una sentencia que es la siguiente:

OPTIMIZE TABLE *tabla*

Esta sentencia -que debe usarse después de un proceso de borrado **amplio**- *depura* la tabla eliminando los *registros inutilizados* por el proceso DELETE, con lo que logra una *reducción* del tamaño de la tabla a su *dimensión óptima*.

Los arrays de la sentencia SELECT

Aunque están comentados en los *códigos fuente* de los scripts queremos reiterar aquí -aprovechando este espacio que la maquetación nos concede- para hacer algunas precisiones sobre los resultados de las consultas de tablas.

Se trata de los índices de los arrays que se obtienen mediante las funciones:

`mysql_fetch_array()`
y
`mysql_fetch_row()`

Los índices escalares, en ambos casos, cuanto tratan información obtenida mediante una sentencia SELECT coinciden con el orden en el que han sido establecidos los campos en esa instrucción concreta. De modo que el primer de esos nombres de campos sería asociado con el índice cero de estos array, el segundo con el índice 1 y así sucesivamente.

En el caso del array asociativo devuelto por la primera de estas funciones, los índices coinciden siempre con los nombres de los campos de los que han sido extraídos los datos.

En el caso de que la consulta afecte a varias tablas (recuerda que los campos se asignan poniendo *tabla.campo* (nombre de la tabla y nombre del campo) el índice del

[Ver formulario](#)[Ver script](#)[Ejecutar ejemplo](#)

Guardar y recuperar bases de datos y o tablas

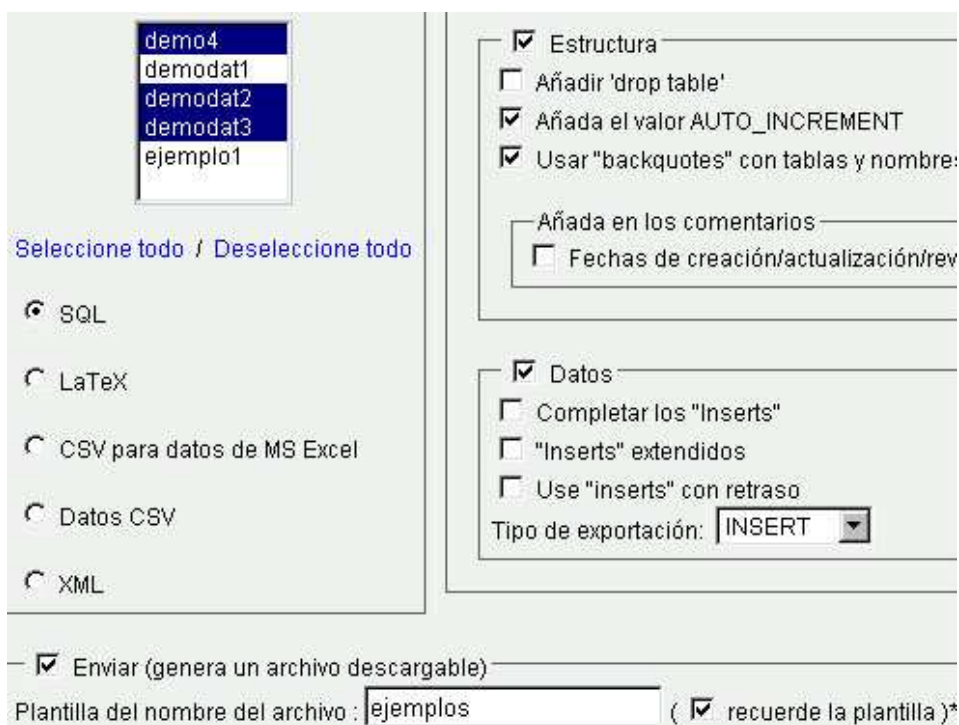
Aunque es perfectamente factible desarrollar scripts propios que permitan guardar y recuperar tanto las estructuras como los datos de una tabla ó de la base de datos completa, mencionaremos aquí una de las posibilidades más cómodas de hacerlos.

PhpMyAdmin es una magnífica herramienta para hacer y recuperar copias de seguridad.

Si abrimos esta utilidad <http://localhost/myadmin/> podremos ver los dos enlaces que ves en la imagen -**SQL** y **Exportar**- que permiten *importar* y *exportar* tanto estructuras como datos y estructuras.

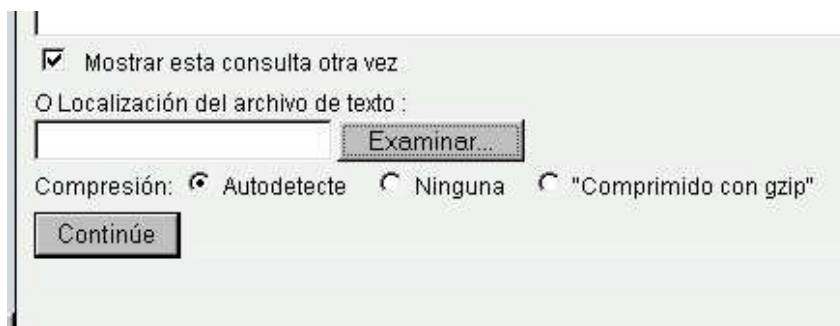


Al pulsar sobre *Exportar* nos aparecerá una página como esta:



donde podremos elegir una, varias o todas la tablas y que según la opciones elegidas nos permite exportar estructuras y/o datos, según las casillas de verificación que tengamos marcadas. Además nos permite elegir el formato en el que queremos guardar la copia -en nuestro caso elegiríamos SQL- y también según esté o no activada la casilla de verificación *Enviar* visualizar el fichero generado o guardarlo con el nombre que hayamos consignado en la caja de texto *Plantilla del nombre del archivo*.

array asociativo sería esa expresión con el punto incluido.



A screenshot of a MySQL backup restoration dialog box. It features a checkbox labeled 'Mostrar esta consulta otra vez' which is checked. Below it is a label 'Localización del archivo de texto :' followed by a text input field and an 'Examinar...' button. Underneath is a 'Compresión:' section with three radio button options: 'Autodetecte' (selected), 'Ninguna', and '"Comprimido con gzip"'. At the bottom is a 'Continúe' button.

Para restaurar datos y/o estructuras desde un fichero de seguridad creado mediante el proceso anterior usaríamos la opción SQL de la primera imagen. A través de ella accederíamos a una página cuyo contenido estamos visualizando en esta última imagen.

Bastaría pulsar en examinar, buscar el fichero de seguridad y pulsar continúe. MySQL se encargaría de restaurar –en la base de datos a la que pertenezcan– todas las tablas contenidas en esa copia.

