Modificar registros

Sintaxis MySQL de modificación de registros

Las sentencias MySQL que permiten la modificación de registros en las tablas pueden incluir algunas de las siguientes *cláusulas* que, al igual que ocurría en casos anteriores, pueden tener categoría de **obligatorias** u **opcionales**.

El orden en que deben estar indicadas ha de seguir la misma secuencia en la que están descritas aquí.

UPDATE

Tiene carácter **obligatorio**, debe ser la **primera palabra de la sentencia** e indica a MySQL que vamos realizar una **modificación**.

[LOW_PRIORITY]

Es **opcional** e indica a MySQL **espere** *a que se terminen de hacer* las consultas que en ese momento pudiera haber en proceso antes realizar la actualización.

[IGNORE]

Es **opcional**. Cuando se incluye en una sentencia el proceso de actualización *no se interrumpe* si aparece un conflicto de clave duplicada en uno de los registros en proceso. Simplemente ignora ese registro y continúa con los siguientes

Si no se incluye, el proceso de modificación **se interrumpe** en el momento en que encuentre **un conflicto de clave duplicada**.

Tanto con **ignore** como sin esa *cláusula*, en el caso de duplicidad de clave NUNCA se efectúan las modificaciones.

tabla

Es **obligatoria** y contiene el **nombre** de la tabla que pretendemos modificar.

SET

Tiene carácter **obligatorio** y debe estar *delante* de las definiciones de *campo y valor*.

campo1 = valor1

Es **obligatoria** al menos una definición. Indica el **nombre del campo** a modificar (*campo1*) y el **valor** que se asignará a ese campo.

Si se pretende modificar **más de un campo** se repetirá esta definición tantas veces como sea necesario, separando cada una de ellas por **una coma**.

WHERE

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

ORDER BY

Tiene idéntica funcionalidad a la descrita al referirnos a consultas

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
.

Algunas consideraciones sobre la sintaxis

Siempre que manejes PHP y MySQL debes tener muy presente lo siguiente:

- MySQL requiere **SIEMPRE** que los valores tipo **cadena** que incluyen campos de fecha vayan **entre comillas**. Por el contrario, los **numéricos no deben llevar comillas**.
- Presta mucha atención a esto cuando **escribas** los valores directamente en la sentencia MySQL
- Cuando **pases valores** desde una variable PHP debes tener muy en cuenta las consideraciones anteriores y si el contenido de la variable es una cadena que va a ser tratada como tal por MySQL tienes dos opciones para evitar el error:
  - Definir la variable así: **$*variable* ="'*valor*'"** (comillas dobles, comilla simple *al principio* y comilla simple, comilla doble *al final*) y poner en la sentencia MySQL el nombre de la variable sin entrecomillar, o
  - Definir la variable PHP así: **$*variable* ="*valor*"** y al escribir el nombre de esa variable en la sentencia MySQL escribirlo entre **comillas sencillas**, es decir, así: **'$*variable*'**
- No pienses que es caprichoso el *orden* que hemos puesto en las comillas. Recuerda que al llamar a la sentencia MySQL, el contenido de la sentencia **va entre comillas** (que por costumbre son **comillas dobles**, por esa razón **todo *entrecomillado*** que vaya dentro de esa sentencia ha de usar **comillas simples** para evitar un **error seguro**).

  De ahí que al definir una variable PHP en la forma **$*variable* ="'*valor*'"** las comillas dobles exteriores indican a PHP que se trata de una cadena, por lo que, al pasar la variable a MySQL éste recibirá el contenido de la cadena que es, logicamente: **'*valor*'** y en este caso las comillas forman parte del valor, razón por el que no es necesario escribir -en la sentencia MySQL- el nombre de la variable entrecomillado.

En este primer ejemplo, hemos incluido una actualización de tablas que pondrá
puntuación 7
en la primera de las pruebas a todos los
aspirantes a astronautas
de nuestro ejemplo.

Es un caso de actualización sin la condición WHERE y tiene el código comentado.

Ver código fuente	Ejecutar la modificación

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 DNI que por su unicidad garantizaría que la modificación solamente va a afectar a uno solo de los registros.

El ejemplo siguiente nos permitirá hacer modificaciones de este tipo en la tabla **deomodat2**. Observa el código fuente y verás que mediante un simple recurso JavaScript, el script que realiza la modificación nos reenvía al formulario con un mensaje de confirmación de la modificación.

Ver código «formulario»	Ver código del «script»	Ejecutar la modificación

Modificación simúltanea de un campo en cualquiera de los registros

Aquí tienes un ejemplo que permite visualizar **el valor actual** de todas las puntuaciones de la *prueba 2* de los *astronautas* así como sus nombres y apellidos y DNI y en la cual se pueden modificar **ninguno**, **uno**, **varios** o todos los valores y posteriormente actualizarlos todos con los nuevos valores.

Ver código «formulario»	Ver código del «script»	Ejecutar la modificación

Ejercicio nº 43

Crea los formularios y scripts necesarios para poder elegir un alumno cualquiera mediante su DNI –en la **tabla1**– y modificar cualquiera de sus datos personales.