**Sintaxis de UPDATE**

**<http://dev.mysql.com/doc/refman/5.7/en/update.html>**

Sintaxis para una tabla:

UPDATE [LOW\_PRIORITY] [IGNORE] ***tbl\_name***

SET ***col\_name1***=***expr1*** [, ***col\_name2***=***expr2*** ...]

[WHERE ***where\_definition***]

[ORDER BY ...]

[LIMIT ***row\_count***]

Sintaxis para múltiples tablas:

UPDATE [LOW\_PRIORITY] [IGNORE] ***table\_references***

SET ***col\_name1***=***expr1*** [, ***col\_name2***=***expr2*** ...]

[WHERE ***where\_definition***]

El comando **UPDATE** actualiza columnas en registros de tabla existentes con nuevos valores. La cláusula **SET** indica qué columna modificar y los valores que puede recibir. La cláusula **WHERE** , si se da, especifica qué registros deben actualizarse. De otro modo, se actualizan todos los registros. Si la cláusula **ORDER BY** se especifica, los registros se actualizan en el orden que se especifica. La cláusula **LIMIT** es el límite de registros a actualizar.

El comando **UPDATE** soporta los siguientes modificadores:

* Si usa la palabra clave **LOW\_PRIORITY**, la ejecución de **UPDATE** se retrasa hasta que no haya otros clientes leyendo de la tabla.
* Si usa la palabra clave **IGNORE**, el comando de actualización no aborta incluso si ocurren errores durante la actualización. Los registros que presenten conflictos de clave duplicada no se actualizan. Los registros cuyas columnas se actualizan a valores que provocarían errores de conversión de datos se actualizan al valor válido más próximo.

Si accede a una columna de ***tbl\_name*** en una expresión, **UPDATE** usa el valor actual de la columna. Por ejemplo, el siguiente comando pone la columna **age** a uno más que su valor actual:

mysql> UPDATE persondata SET age=age+1;

Las asignaciones **UPDATE** se evalúan de izquierda a derecha. Por ejemplo, el siguiente comando dobla la columna **age** y luego la incrementa:

mysql> UPDATE persondata SET age=age\*2, age=age+1;

Si pone en una columna el valor que tiene actualmente, MySQL se da cuenta y no la actualiza.

Si actualiza una columna declarada como **NOT NULL** con un valor **NULL**, la columna recibe el valor por defecto apropiado para el tipo de la columna y se incrementa el contador de advertencias. El valor por defecto es **0** para tipos numéricos, la cadena vacía (**''**) para tipos de cadena, y el valor “cero” para valores de fecha y hora.

**UPDATE** retorna el número de registros que se cambian. En MySQL 5.0, la función **mysql\_info()** de la API de C retorna el número de registros coincidentes actualizados y el número de advertencias que ocurren durante el **UPDATE**.

Puede usar **LIMIT *row\_count*** para restringir el alcance del **UPDATE**. Una cláusula **LIMIT** es una restricción de registros coincidentes. El comando para en cuanto encuentra ***row\_count*** registos que satisfagan la cláusula **WHERE**, tanto si han sido cambiados como si no.

Si un comando **UPDATE** incluye una cláusula **ORDER BY**, los registros se actualizan en el orden especificado por la cláusula.

Puede realizar operaciones **UPDATE** que cubran varias tablas. La parte ***table\_references*** lista las tablas involucradas en el join. Su sintaxis se describe ampliamente en [Sección 13.2.7.1, “Sintaxis de **JOIN**”](http://dev.mysql.com/doc/refman/5.0/es/join.html). Aquí hay un ejemplo:

UPDATE items,month SET items.price=month.price

WHERE items.id=month.id;

Este ejemplo muestra un inner join usando el operador coma, pero los comandos **UPDATE** de múltiples tablas pueden usar cualquier tipo de join permitido en comandos **SELECT** tales como **LEFT JOIN**.

***Nota***: *No* puede usar **ORDER BY** o **LIMIT** con un **UPDATE** de múltiples tablas.

En MySQL 5.0, necesita el permiso **UPDATE** sólo para columnas referenciadas en un **UPDATE** de múltiples tablas que se actualizan realmente. Necesita sólo el permiso **SELECT** para algunas columnas que se leen pero no se modifican.

Si usa un comando **UPDATE** de múltiples tablas que involucren tablas **InnoDB** con restricciones de claves foráneas, el optimizador de MySQL puede procesar tablas en un orden distinto al de la relación padre/hijo. En este caso, el comando fall y hace un roll back. En su lugar, actualice una única tabla y confíen en las capacidades de **ON UPDATE** que proporciona **InnoDB** para que el resto de tablas se modifiquen acórdemente. Consulte[Sección 15.6.4, “Restricciones (constraints) **FOREIGN KEY**”](http://dev.mysql.com/doc/refman/5.0/es/innodb-foreign-key-constraints.html).

Actualmente, no puede actualizar una tabla y seleccionar de la misma en una subconsulta.

[UPDATE](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Fupdate.html) `curso` [SET](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Fset.html) `Aula`= 11 WHERE `Curso` =1

[UPDATE](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Fupdate.html) `calendario` [SET](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Fset.html) `IdTipoDia` = 5 WHERE `IdMes`=8 [and](http://localhost/phpmyadmin/url.php?url=http%3A%2F%2Fdev.mysql.com%2Fdoc%2Frefman%2F5.5%2Fen%2Flogical-operators.html#operator_and) `NumeroDia`=15