**Sintaxis de DELETE**

**http://dev.mysql.com/doc/refman/5.7/en/delete.html**

Sintaxis para una tabla:

DELETE [LOW\_PRIORITY] [QUICK] [IGNORE] FROM ***tbl\_name***

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

[ORDER BY ...]

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

Sintaxis para múltiples tablas:

DELETE [LOW\_PRIORITY] [QUICK] [IGNORE]

***tbl\_name***[.\*] [, ***tbl\_name***[.\*] ...]

FROM ***table\_references***

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

O:

DELETE [LOW\_PRIORITY] [QUICK] [IGNORE]

FROM ***tbl\_name***[.\*] [, ***tbl\_name***[.\*] ...]

USING ***table\_references***

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

**DELETE** borra los registros de ***tbl\_name*** que satisfacen la condición dada por ***where\_definition***, y retorna el número de registros borrados.

Si realiza un comando **DELETE** sin cláusula **WHERE** se borran todos los registros. Una forma más rápida de hacerlo, cuando no quiere saber el número de registros borrados, se usa **TRUNCATE TABLE**. Consulte [Sección 13.2.9, “Sintaxis de **TRUNCATE**”](http://dev.mysql.com/doc/refman/5.0/es/truncate.html).

Si borra el registro conteniendo el máximo valor para una columna **AUTO\_INCREMENT** , el valor se reúsa para una tabla **BDB** , pero no para tablas **MyISAM** o **InnoDB** . Si borra todos los registros en la tabla con **DELETE FROM*tbl\_name*** (sin cláusula **WHERE** ) en modo **AUTOCOMMIT** , la secuencia comienza para todos los tipos de tabla excepto para **InnoDB** y **MyISAM**. Hay algunas excepciones para este comportamiento para tablas **InnoDB** , como se discute en [Sección 15.6.3, “Cómo funciona una columna **AUTO\_INCREMENT** en **InnoDB**”](http://dev.mysql.com/doc/refman/5.0/es/innodb-auto-increment-column.html).

Para tablas **MyISAM** y **BDB** , puede especificar una columna **AUTO\_INCREMENT** secundaria en una clave de múltiples columnas. En este caso, el reúso de valores borrados del inicio de la secuencia se realiza incluso para tablas **MyISAM** . Consulte [Sección 3.6.9, “Utilización de **AUTO\_INCREMENT**”](http://dev.mysql.com/doc/refman/5.0/es/example-auto-increment.html).

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

* Si especifica **LOW\_PRIORITY**, la ejecución de **DELETE** se retarda hasta que no hay más clientes leyendo de la tabla.
* Para tablas **MyISAM** , si usa la palabra **QUICK** , el motor de almacenamiento no mezcla las hojas del índice durante el borrado, que puede acelerar algunos tipos de operaciones de borrado.
* En MySQL 5.0, la palabra clave **IGNORE** hace que MySQL ignore todos los errores durante el proceso de borrar registros. (Los errores encontrados durante la etapa de parseo se procesan de la forma habitual.) Los errores que se ignoran debido al uso de esta opción se retornan como advertencias.

La velocidad de las operaciones de borrado pueden verse afectadas por factores discutidos en [Sección 7.2.16, “Velocidad de sentencias **DELETE**”](http://dev.mysql.com/doc/refman/5.0/es/delete-speed.html).

En tablas **MyISAM** , los registros borrados se mantienen en una lista enlazada y las operaciones **INSERT**siguientes reúsan antiguas posiciones de registro. Para reclamar espacio no usado y reducir tamaño de fichero, use el comando **OPTIMIZE TABLE** o la utilidad **myisamchk** para reorganizar las tablas. **OPTIMIZE TABLE** es más sencillo, pero **myisamchk** es más rápido. Consulte [Sección 13.5.2.5, “Sintaxis de **OPTIMIZE TABLE**”](http://dev.mysql.com/doc/refman/5.0/es/optimize-table.html) y[Sección 5.8.3.10, “Optimización de tablas”](http://dev.mysql.com/doc/refman/5.0/es/optimization.html).

El modificador **QUICK** afecta si las hojas del índice es mezclan en operaciones de borrado. **DELETE QUICK** es más útil para aplicaciones en que los valores del índice para registros borrados se replazan con valores similares de registros insertados posteriormente. En este caso, los agujeros dejados por los valores borrados se reúsan.

**DELETE QUICK** no es útil cuando los valores borrados conducen a bloques de índices no rellenos con un rango de valores índice para el que vuelven a ocurrir nuevas inserciones. En este caso, el uso de **QUICK** puede conducir a un gasto de espacio que queda sin reclamar. Aquí hay un ejemplo de este escenario:

1. Cree una tabla que contenga una columna **AUTO\_INCREMENT** indexada.
2. Inserta varios registros en la tabla. Cada inserción resulta en un valor índice que se añade al final del índice.
3. Borra un bloque de registros al final del rango de la columna usando **DELETE QUICK**.

En este escenario, los bloques de índice asociados con los valores de índice borrado quedan sin rellenar pero no se mezclan con otros bloques de índice debido al uso de **QUICK**. Quedan sin rellenar cuando hay nuevas inserciones, ya que los nuevos registros no tienen valores índice en el rango borrado. Además, quedan sin rellenar incluso si luego usa **DELETE** sin **QUICK**, a no ser que algunos de los valores de índice borrados estén en los bloques de índice dentro o adyacentes a los bloques no rellenos. Para reclamar el espacio de índice sin usar bajo estas circunstancias use **OPTIMIZE TABLE**.

Si va a borrar varios registros de una tabla, puede ser más sencillo usar **DELETE QUICK** seguido por **OPTIMIZE TABLE**. Esto reconstruye el índice en lugar de realizar varias operaciones de mezcla de bloques de índice.

La opción de MySQL **LIMIT *row\_count*** para **DELETE** le dice al servidor el máximo número de registros a borrar antes de retornar el control al cliente. Esto puede usarse para asegurar que un comando **DELETE** específico no tarada demasiado tiempo. Puede simplemente repetir el comando **DELETE** hasta que el número de registros afectados sea menor que el valor **LIMIT** .

Si el comando **DELETE** incluye una cláusula **ORDER BY** , los registros se borran en el orden especificado por la cláusula. Esto es muy útil sólo en conjunción con **LIMIT**. Por ejemplo, el siguiente ejemplo encuentra registros coincidentes con la cláusula **WHERE** ordenados por **timestamp\_column**, y borra el primero (el más viejo).

DELETE FROM somelog

WHERE user = 'jcole'

ORDER BY timestamp\_column

LIMIT 1;

Puede especificar múltiples tablas en un comando **DELETE** para borrar registros de una o más tablas dependiendo de una condición particular en múltiples tablas. Sin embargo, no puede usar **ORDER BY** o **LIMIT** en un **DELETE** de múltiples tablas.

La parte ***table\_references*** lista las tablas involucradas en el join. Esta sintaxis se describe en[Sección 13.2.7.1, “Sintaxis de **JOIN**”](http://dev.mysql.com/doc/refman/5.0/es/join.html).

Para la primera sintaxis, sólo los registros coincidentes de las tablas listadas antes de la cláusula **FROM** se borran. Para la segunda sintaxis, sólo los registros coincidentes de las tablas listadas en la cláusula **FROM** (antes de la cláusula **USING** ) se borran. El efecto es que puede borrar registros para varias tablas al mismo tiempo y tienen tablas adicionales que se usan para buscar:

DELETE t1, t2 FROM t1, t2, t3 WHERE t1.id=t2.id AND t2.id=t3.id;

O:

DELETE FROM t1, t2 USING t1, t2, t3 WHERE t1.id=t2.id AND t2.id=t3.id;

Estos comandos usan las tres tablas al buscar registros a borrar, pero borrar los registros coincidentes sólo para las tablas **t1** y **t2**.

Los ejemplos anteriores muestran inner joins usando el operador coma, pero comandos **DELETE** de varias tablas pueden usar cualquier tipo de join permitido por comandos **SELECT** tales como **LEFT JOIN**.

La sintaxis permite **.\*** tras los nombres de tabla para compatibilidad con **Access**.

Si usa un comando **DELETE** de varias tablas incluyendo tablas **InnoDB** para las que hay restricciones de clave foránea, el optimizador MySQL puede procesar tablas en un orden ditinto del de su relación padre/hijo. En este caso, el comando falla y se deshace. En su lugar, debe borrar de una tabla úncia y confiar en la capacidad de **ON DELETE** que proporciona **InnoDB** para hacer que las otras tablas se modifiquen correctamente.

***Nota***: En MySQL 5.0, debe usar el alias (si se dió) al referirse a un nombre de tabla:

En MySQL 4.1:

DELETE t1 FROM test AS t1, test2 WHERE ...

Borrados cruzados entre bases de datos se soportan para borrados de varias tablas, pero en este caso, debe referirse a las tablas sin usar alias. Por ejemplo:

DELETE test1.tmp1, test2.tmp2 FROM test1.tmp1, test2.tmp2 WHERE ...

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