Consultas de Acción

¿Cuáles son?

Las consultas de acción son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir, borrar y modificar registros.

- DELETE
- INSERT
- UPDATE

DELETE

- Crea una consulta de eliminación que **borra los registros** de una **tabla listada en la cláusula FROM**, **que satisfagan la cláusula WHERE**. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Devuelve el número de registros borrados.
- Si se borra la fila que contiene el valor **máximo** para una columna **AUTO_INCREMENT**, ese valor **podrá ser usado** por una tabla **ISAM o BDB**, pero **no por una tabla MyISAM o InnoDB**. Si se borran todas las filas de una tabla **con DELETE FROM tbl_name** (sin un WHERE) en **modo AUTOCOMMIT**, la **secuencia comenzará de nuevo** para todos los motores de almacenamiento, **excepto para InnoDB** y (desde MySQL 4.0) **MyISAM.**
- Una vez que se han eliminado los registros utilizando una consulta de borrado, **no puede deshacer la operación**.

DELETE

Si se desea saber qué registros se eliminarán, **primero examinar los resultados de una consulta de selección que utilice el mismo criterio** y después ejecutar la consulta de borrado.

Debería mantenerse copias de seguridad de sus datos en todo momento. Si elimina los registros equivocados se podrá recuperar desde las copias de seguridad.

```
select id, last_name, first_name,title,dept_id,salary
from emp
where salary < 800;

delete
from emp
where salary < 800;
```

DELETE

Su *sintaxis* es:
DELETE [LOW_PRIORITY] FROM tbl_name
[WHERE where_definition]
[LIMIT rows]

La opción **LOW_PRIORITY** hace que la operación de borrado se demore hasta tanto no haya ningún otro cliente accediendo a los datos.
La opción **LIMIT** le especifica al servidor hasta cuantas filas debe borrar. Esto es útil para el caso de querer que el control pase al cliente enseguida. Luego, simplemente, se repite el **DELETE** hasta tanto el nro. de filas afectadas sea menor al valor especificado en el **LIMIT**.
Si se usa una cláusula **ORDER BY** las filas serán borradas en el orden especificado por la cláusula. Esto sólo será práctico si se usa en conjunción con **LIMIT**.
Por ejemplo, la siguiente sentencia encuentra filas que satisfagan la cláusula WHERE, las ordena por tiempos, y borra la primera (la más antigua)

```
DELETE FROM somelog
   WHERE user = 'jcole'
   ORDER BY timestamp
   LIMIT 1
```

DELETE MULTITABLAS

Se pueden especificar múltiples tablas en la sentencia **DELETE** para eliminar filas de una o más tablas dependiendo de una condición particular en múltiples tablas. Sin embargo, **no es posible usar ORDER BY o LIMIT en un DELETE multitabla.**

Su *sintaxis* se describe en .
DELETE t1,t2 FROM t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
Sólo se borran las filas coincidentes de las tablas listadas antes de la cláusula FROM.

Otra opción, *sintaxis*:
DELETE FROM t1,t2 USING t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
Estas sentencias usan las tres tablas cuando buscan filas para borrar, pero borran las filas coincidentes sólo de las tablas t1 y t2.

Si se usa una sentencia DELETE multitabla que afecte a **tablas InnoDB** para las que haya definiciones de **claves foráneas**, el optimizador MySQL procesará las tablas en un orden diferente del de la relación padre/hijo. En ese caso, la sentencia puede fallar y deshará los cambios (roll back). En su lugar, se debe borrar de una sola tabla y **confiar el las capacidades de ON DELETE** que proporciona InnoDB que harán que las otras tablas se modifiquen del modo adecuado.

INSERT

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipos: Insertar un único registro con valores explícitamente determinados, o Insertar en una tabla los registros contenidos en otra/s tabla/s.
Sintaxis:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    VALUES (\,...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expression, ...]

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    SET col_name=\, ...
    [ ON DUPLICATE KEY UPDATE col_name=expression, ... ]

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

INSERT

INSERT … VALUES e *INSERT … SET*, insertas filas basándose en los valores especificados explícitamente. El formato *INSERT … SELECT* inserta filas seleccionadas de otra tabla o tablas.

tbl_name es la tabla donde se insertarán las filas. Las columnas para las que la sentencia proporciona valores se pueden especificar de las siguientes formas:

- Las lista de nombres de columnas o la clausula *SET indican las columnas explícitamente.*
- Si no se especifica una lista de columnas para *INSERT … VALUES* o *INSERT … SELECT*, se deben *proporcionar valores para todas las columnas* en la lista *VALUES()* o por *SELECT*. Si no se conoce el orden de las columnas dentro de la tabla, usar tbl_name para encontrarlo.

INSERT con valores por defecto

- Si se especifica una lista de columnas que no contiene el nombre de todas las columnas de la tabla, a las columnas sin nombre le serán asignados sus valores por defecto.
- Se puede usar la palabra clave DEFAULT para poner una columna a su valor por defecto de forma explícita. Hace más sencillo escribir sentencias INSERT que asignan valores a casi todas las columnas, porque permite la escritura de una lista VALUES incompleta, que no incluye un valor para cada columna de la tabla.
- Si tanto la lista de columnas como la de VALUES están vacías, INSERT crea una fila en la que cada columna tendrá su valor por defecto.

  Ejemplo:

  ```
  INSERT INTO tbl_name () VALUES();
  ```

INSERT valores con expresión

Se puede especificar una expresión expr para proporcionar un valor de columna. Esto forzará complejas conversiones de tipo si el de la expresión no coincide con el tipo de la columna, y la conversión de un valor dado puede provocar diferentes valores insertados dependiendo del tipo de la columna.
Por ejemplo:
Insertar la cadena '1999.0e-2' en una columna INT, FLOAT, DECIMAL(10,6) o YEAR producirá los valores 1999, 19.9921, 19.992100 y 1999.
El motivo es que el valor almacenado en una columna INT y YEAR sea 1999 es que la conversión de cadena a entero mira sólo la parte inicial de la cadena que se pueda considerar un valor entero o un año válido.

Para columnas en punto flotante o punto fijo, la conversión de cadena a punto flotante tiene en cuenta la cadena completa como un valor válido en coma flotante.

Una expresión expr se puede referir a cualquier columna que se haya asignado previamente en la lista de valores. Por ejemplo, se puede hacer esto, ya que el valor de col2 se refiere a col1, que ya ha sido asignado:
INSERT INTO tbl_name (col1,col2) VALUES(15,col1*2);
Pero **no se puede hacer** esto, porque el valor para col1 se refiere a col2, que se asigna después de col1:
INSERT INTO tbl_name (col1,col2) VALUES(col2*2,15);
La excepción es para columnas que contengan valores autoincrementados. Esto es debido a que el valor AUTO_INCREMENT se genera después de la asignación de cualquier otro valor, así que la referencia a una columna AUTO_INCREMENT devolverá un 0.

INSERT - modificadores

- Si se especifica la palabra clave **DELAYED**, el **servidor coloca la fila** o filas a **insertar en un búfer**, y el cliente lanza la sentencia y puede continuar. Si la tabla está ocupada, el servidor guarda las filas. **Cuando la tabla queda libre, empieza a insertar filas**, verificando periódicamente para ver si hay nuevas peticiones de lectura para la tabla. Si las hay, **la cola de inserción retardada se suspende hasta que la tabla quede libre de nuevo**.

- Si se especifica la palabra clave **LOW_PRIORITY**, la ejecución de la sentencia **INSERT se retrasa hasta que no haya clientes leyendo de la tabla**. Esto incluye a otros clientes que empiecen a leer mientras existan clientes leyendo, y mientras la sentencia INSERT LOW_PRIORITY está esperando. Por lo tanto, es posible que un cliente que lance una sentencia INSERT LOW_PRIORITY permanezca **esperando por mucho tiempo en un entorno con muchas lecturas**. LOW_PRIORITY **no debe ser usado con tablas MyISAM** que **no permiten inserciones concurrentes.**

- Si se especifica la palabra clave **HIGH_PRIORITY**, **se anula el efecto de la opción --low-priority-updates** si el servidor fue arrancado con esa opción. Esto también hace que **no se puedan usar inserciones concurrentes.**

INSERT - modificadores

Si se especifica la palabra **IGNORE** en una sentencia **INSERT**, los errores que se produzcan mientras se ejecuta la sentencia se tratan con avisos.

Sin **IGNORE una fila que duplique** un valor de clave PRIMARY o UNIQUE existente en la tabla provocará un error y la sentencia será abortada. Con IGNORE, **el error se ignora y la fila no será insertada**.

Las conversiones de datos que produzcan errores abortarán la sentencia si no se especifica **IGNORE**. Con IGNORE, **los valores inválidos se ajustarán al valor más próximos y se insertarán**; se **producirán avisos pero la sentencia no se aborta**. Se puede determinar cuantas filas fueron insertadas mediante la función del API mysql_info.

Si se especifica la cláusula **ON DUPLICATE KEY UPDATE**, y se inserta una fila que puede provocar un valor duplicado en una clave **PRIMARY o UNIQUE**, se realiza un **UPDATE** (actualización) **de la fila antigua**. En general, **se debe intentar evitar el uso** de de la cláusula ON DUPLICATE KEY en tablas con múltiples claves UNIQUE. **Cuando se usa ON DUPLICATE KEY UPDATE**, la opción **DELAYED se ignora**.

INSERT – Devolución una vez hecha la inserción

Si se usa una sentencia INSERT ... VALUES con una lista de múltiples valores o INSERT ... SELECT, la sentencia devuelve una cadena de información con este formato:

Records: 100 Duplicates: 0 Warnings: 0

Records indica el número de filas procesadas por la sentencia. (No es necesariamente el número de filas insertadas. "Duplicates" puede ser distinto de cero.)

Duplicates indica el número de filas que no pudieron ser insertadas porque contienen algún valor para un índice único ya existente.

Warnings indica el número de intentos de inserción de valores de columnas que han causado algún tipo de problemas. Se pueden producir "Warnings" bajo cualquiera de las siguientes condiciones:

- Inserción de NULL en una columna declarada como NOT NULL. Para sentencias INSERT de varias filas o en sentencias INSERT ... SELECT, se asigna el valor por defecto apropiado al tipo de columna. Ese valor es 0 para tipos numéricos, la cadena vacía ('') para tipos cadena, y el valor "cero" para tipos de fecha y tiempo.
- Asignación de un valor fuera de rango a una columna numérica. El valor se recorta al extremo apropiado del rango.
- Asignación de un valor como '10.34 a' a un campo numérico. La parte añadida se ignora y se inserta sólo la parte numérica inicial. Si el valor no tiene sentido como un número, el valor asignado es 0.
- Inserción de un valor a una columna de cadena (CHAR, VARCHAR, TEXT o BLOB) que exceda la longitud máxima para la columna. El valor se trunca a la longitud máxima de la columna.
- Inserción de un valor dentro de una columna de fecha o tiempo que sea ilegal para el tipo de columna. Se asigna a la columna el valor apropiado de cero, según su tipo.

UPDATE

UPDATE actualiza columnas de filas existentes de una tabla con nuevos valores. La cláusula **SET** indica las columnas a modificar y los valores que deben tomar. La cláusula **WHERE**, si se da, especifica qué filas deben ser actualizadas. Si no se especifica, serán actualizadas todas ellas. Si se especifica la cláusula **ORDER BY**, las filas se modificarán en el orden especificado. La cláusula **LIMIT** establece un límite al número de filas que se pueden actualizar.

Sintaxis:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT row_count]
O
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT row_count]
```

UPDATE – Modificadores

La sentencia **UPDATE** soporta los modificadores siguientes:

- Si se usa la palabra **LOW_PRIORITY**, la ejecución de UPDATE se **retrasará hasta que no haya otros clientes haciendo lecturas de la tabla**.

- Si se especifica **IGNORE**, la sentencia UPDATE **no se abortará si se producen errores durante la actualización.** Las filas con conflictos de **claves duplicadas no se actualizarán**. Las filas para las que la actualización de columnas se puedan producir **errores de conversión se actualizarán con los valores válidos más próximo**s.

UPDATE

Si se accede a una columna de "tbl_name" en una expresión, UPDATE usa el valor actual de la columna. Por ejemplo, la siguiente sentencia asigna a la columna "edad" su valor actual más uno:

```
UPDATE persondata SET edad=edad+1;
```

Las asignaciones UPDATE se evalúan de izquierda a derecha. Por ejemplo, las siguientes sentencias doblan el valor de la columna "edad", y después la incrementan:

```
UPDATE persondata SET edad=edad*2, edad=edad+1;
```

- Si se asigna a una columna el valor que tiene actualmente, MySQL lo notifica y no la actualiza.
- Si se actualiza una columna que ha sido declarada como NOT NULL con el valor NULL, se asigna el valor por defecto apropiado para el tipo de la columna y se incrementa en contador de avisos. El **valor por defecto** es **0** para tipos **numéricos**, la **cadena vacía ('')** para **tipos de cadena**, y el valor "**cero" para tipos de fecha y tiempo.**
- UPDATE **devuelve el número de filas** que se han **modificado**.
- **LIMIT** se restringe al número de filas coincidentes. La sentencia se detiene tan pronto como se encuentran "row_count" filas que satisfagan la cláusula WHERE, tanto si se han modificado como si no.

UPDATE MULTITABLA

- Es posible realizar operaciones UPDATE que cubran múltiples tablas:

```
UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

  El ejemplo muestra una fusión interna usando el operador coma, pero un UPDATE multitabla puede usar cualquier tipo de fusión (join) permitido en sentencias , como un LEFT JOIN.

- **No** es posible usar ORDER BY o LIMIT con UPDATE multitabla.
- Si se usa una sentencia UPDATE multitabla que afecte a tablas InnoDB para las que haya definiciones de claves foráneas, el optimizador MySQL procesará las tablas en un orden diferente del de la relación padre/hijo. En ese caso, la sentencia puede fallar y deshará los cambios (roll back). En su lugar, se debe actualizar una tabla y confiar el las capacidades de ON UPDATE que proporciona InnoDB que harán que las otras tablas se modifiquen del modo adecuado.
- Actualmente, no se puede actualizar una tabla y seleccionar desde la misma en una subconsulta.