Practicar con la modificación de registros

Cuando tenemos que modificar datos que ya están grabados en una tabla, tenemos que utilizar siempre UPDATE y tener en cuenta:

* Lo primero dónde están los datos a modificar, esto me dará el destino, ojo, sólo puede ser una tabla, si queremos cambiar los valores almacenados en varias tablas hay que hacer un UPDATE por cada tabla.

* Luego qué columnas va a cambiar, esto me dirá cuántas asignaciones tendré en la cláusula SET, una por cada columna a cambiar.

* A continuación para cada columna tengo que preguntarme qué valor hay que asignarle. Recuérda que cualquier asignación se escribe siguiendo el formato: columna=valor a asignar y consiste en sustituir el valor que tenía la columna por el valor que se asigna.

Este sería el paso más delicado, dependiendo del valor a asignar habrá que hacer una cosa u otra:

- Si es un valor fijo, sólo asegúrate de asignar un valor del mismo tipo que la columna donde lo insertas.

- Si es el resultado de una expresión que utiliza columnas de la tabla a modificar, no hay problema, pero ten en cuenta que se utiliza el valor antes de la actualización de la fila.

- Si es el resultado de una expresión que utiliza un valor almacenado en UNA fila de otra tabla entonces añade la cláusula FROM de tal forma que en el resultado de esa FROM tengas en la misma fila la columna a modificar y la columna de la otra tabla que necesitas para el cálculo.

- Si el resultado es un valor calculado a partir de varias filas de otra tabla tendrás que asignar seguramente una subconsulta escalar que sea de resumen.

* Por último si queremos modificar todas las filas de la tabla o sólo algunas, si la respuesta es *sólo algunas* hay que indicar esas filas añadiendo un TOP o un WHERE dependiendo del caso.

Veamos todo esto con ejemplos.

1.- Poner a cero las ventas y la cuota de todos los empleados.

*La tabla a modificar es Empleados, ya tenemos el destino:

```
UPDATE Empleados
```

*Hay que modificar dos campos de esa tabla, ventas y cuota luego tendré en la SET dos asignaciones:

```
UPDATE Empleados
SET ventas = ¿? , cuota = ¿?
```

*Los valores a asignar en los dos casos son fijos (es el valor cero) luego no tengo más que asegurarme del tipo de dato de los campos, los campos son numéricos, asigno valores numéricos.

```
UPDATE Empleados
SET ventas = 0 , cuota = 0
```

*¿Quiero modificar todos los empleados? Sí, luego he terminado.

2.- Poner a cero las ventas y la cuota de todos los empleados de la oficina 12.

El razonamiento es el mismo que en el ejercicio 1, sólo cambia la respuesta a la última pregunta, no quiero actualizar todos los empleados sino los de la oficina 12, pues añado un WHERE:

```
UPDATE Empleados
SET ventas = 0 , cuota = 0
WHERE oficina = 12
```

3.- Poner a cero las ventas y la cuota de todos los empleados de las oficinas del Este.

El razonamiento es el mismo que en el ejercicio 1, sólo cambia la condición del WHERE pero en este caso la condición sería region='Este' y el campo region no está en empleados, pues utilizo una subconsulta:

```
UPDATE Empleados
SET ventas = 0 , cuota = 0
WHERE oficina IN (SELECT oficina FROM Oficinas WHERE region = 'Este'
```

4.- Quiero poner a cero las ventas y cuota de los dos primeros empleados.

El razonamiento es el mismo que en el ejercicio 1, sólo cambia la respuesta a la última pregunta, no quiero actualizar todos los empleados sino los dos primeros, pues añado un TOP:

```
UPDATE TOP (2) Empleados
SET ventas = 0 , cuota = 0
```

¡Ojo! Después de la palabra UPDATE el número del TOP debe ir siempre entre paréntesis.

5.- Con la solución del ejercicio anterior actualizo las dos primeras filas pero no sé cuáles son, vimos que con la cláusula TOP en muchos casos era conveniente combinarla con un ORDER BY. Pero el ORDER BY no está permito en un UPDATE. La forma de solucionar este inconveniente es utilizando una tabla derivada.

En este ejemplo vamos a poner a cero las ventas y cuota de los dos empleados con mejores ventas (con empates):

```
UPDATE aliast set ventas = 0
FROM (SELECT TOP 2 WITH TIES * FROM empleados ORDER BY ventas DESC) AS aliast
```

En la tabla derivada puedo utilizar el ORDER BY, luego selecciono las filas a actualizar con la tabla derivada y esa tabla es la que actualizo con el UPDATE.

6.- Subir un 5% las cuotas de los empleados.

*La tabla a modificar es Empleados, ya tenemos el destino:

```
UPDATE Empleados
```

*Hay que modificar un campo de esa tabla (cuota) luego tendré en la SET una asignación:

```
UPDATE Empleados
SET cuota = ¿?
```

*El valor a asignar no es fijo, la nueva cuota depende de la cuota que tenía el empleado anteriormente → en la expresión se utilizará el campo cuota.

¿A qué será igual la nueva cuota? A la cuota que había antes más un 5% de esa cuota, esto puesto a modo de fórmula sería: Cuota= cuota + cuota*0.05.

Cuando aumentamos en un tanto porcien el valor de un campo esta fórmula se puede simplificar de la siguiente forma: cuota= cuota*1.05. Se multiplica el campo por uno coma el tanto porcien a sumar. Del mismo modo si queremos aplicar un descuento a un campo se multiplica el campo por uno menos el tanto porcien a descontar, si en vez de aumentar en un 5% la cuota la quisiéramos reducir en un 5% la fórmula sería cuota=cuota * 0.95.

```
UPDATE Empleados
SET cuota = cuota*1.05
```

*¿Quiero modificar todos los empleados? Sí, luego he terminado.

7.- Actualizar el importe de los pedidos de este año utilizando el precio del producto.

*La tabla a modificar es Pedidos, ya tenemos el destino:

```
UPDATE Pedidos
```

*Hay que modificar un campo de esa tabla (importe) luego tendré en la SET una asignación:

```
UPDATE Pedidos
SET importe = ¿?
```

*El valor a asignar no es fijo, el nuevo importe se tiene que calcular a partir de la cantidad de producto que se ha vendido en el pedido (campo cant de Pedidos) y del precio del producto (campo precio de la tabla Productos).

La expresión sería: Importe= cant * Precio

Como Precio está en Productos tengo que utilizar una FROM para juntar cada pedido con su producto y así tener en una misma fila del origen los campos que intervienen en la expresión:

```
UPDATE Pedidos
SET importe = cant * precio
FROM Pedidos INNER JOIN Productos ON fab=idfab AND producto=idproducto
```

*¿Quiero modificar todos los pedidos? No, sólo los de este año → Añado un WHERE.

```
UPDATE Pedidos
SET importe = cant * precio
FROM Pedidos INNER JOIN Productos ON fab=idfab AND producto=idproducto
WHERE YEAR(fechapedido) = YEAR(GETDATE())
```

8.- Queremos volver a calcular las ventas de todas las oficinas a partir de las ventas de sus empleados.

*La tabla a modificar es Oficinas, ya tenemos el destino:

```
UPDATE Oficinas
```

*Hay que modificar un campo de esa tabla (ventas) luego tendré en la SET una asignación:

```
UPDATE Oficinas
SET ventas = ¿?
```

*El valor a asignar no es fijo, el nuevo valor se tiene que calcular a partir de varias filas de otra tabla, las ventas de todos los empleados de esa oficina, luego tengo que asignar una subconsulta escalar: la asignación (sin tener en cuenta la sintaxis de TSQL) sería: ventas de la oficina= SUMA(ventas de sus empleados).

Esa SUMA(ventas de sus empleados) escrita como una subconsulta escalar sería:

```
(SELECT SUM(ventas) FROM Empleados WHERE empleados.oficina= oficinas.oficina)
```

Pues ya tenemos el valor a asignar:

```
UPDATE Oficinas
SET ventas = (SELECT SUM(ventas) FROM Empleados WHERE empleados.oficina=
oficinas.oficina)
```

*¿Quiero modificar todas las oficinas? Sí, luego he terminado.

Estas son básicamente el tipo de actualizaciones que os encontraréis, luego todo se puede combinar.