Base de Datos
Boletín 4.3: SQL LMD

Utilizando la base de datos *Empresa*, realizar las siguientes consultas:

1. Añadir una nueva oficina para la ciudad de Madrid, con el número de oficina 30, con un objetivo de 10.000 euros y región 'Centro'. Suponer que conocemos el orden de los campos en la tabla.

   ```
   INSERT INTO oficinas
   VALUES (30, 'centro', 'Madrid', null, 100000,0) ;
   ```

   En este caso como no especificamos una lista de columnas tenemos que poner los valores en el mismo orden que las columnas en vista diseño de la tabla.

2. Igual que el ejercicio anterior suponiendo que no sabemos cual es el orden de los campos en la tabla Oficina.

   ```
   INSERT INTO oficinas (oficina,region,ciudad,dir,objetivo,ventas)
   VALUES (30, 'Madrid', 'centro', null, 100000,0) ;
   ```

3. Insertar tus datos como nuevo empleado. Utilizar como numemp los tres últimos dígitos del dni, la oficina 23, el puesto "Programador", sin jefe, con una cuota de 1000 y las ventas a 0. En el contrato la fecha de hoy.

   ```
   INSERT INTO Empleados
   VALUES (999, "Pepito", 25, 23, "Programador", '2017-10-11', null,
                                                              1000, 0);
   ```

   En lugar de escribier la fecha de hoy podemos utilizar NOW().

4. Insertar un nuevo cliente con tu nombre y utilizar como numclie 999. Dejar el resto de campos a su valor por defecto.

   ```
   INSERT INTO Clientes (numclie, nombre)
   VALUES (999, "Pepito");
   ```

   Otra opción es:

   ```
   INSERT INTO Clientes (numclie, nombre, resp, limitecredito)
   VALUES (999, "Pepito", DEFAULT, DEFAULT);
   ```

5. Insertar los empleados que no tienen jefes como clientes. Como número de clientes utilizaremos el mismo número de empleado, ellos mismo serán sus propios responsable y el límite de crédito será 0.

6. De la base de datos Robótica, insertar los dependientes como clientes de la base de datos Empresa. Para generar el numclie utilizaremos la suma del año, mes y día de la fecha de nacimiento de los dependientes. El resp será el 108 y el límite de crédito 1000 euros.

7. Subir un 5 % el precio de todos los productos del fabricante 'ACI'.

   ```
   UPDATE Productos
   SET precio = precio * 1.05
   WHERE idfab = 'ACI';
   ```

8. Incrementar en uno la edad de los empleados. WorkBench tiene un sistema de protección que impide realizar UPDATES o DELETE sin WHERE. Para quitar esta opción, desde workBench, en Edit - Preferences en la solapa SQL Queris, desmarcamos la opción “Safe Update” y reiniciamos workbench.

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

9. Cambiar los empleados de la oficina 21 a la oficina 30.

```
UPDATE Empleados
SET oficina = 30
WHERE oficina = 21;
```

   Si ejecutamos esta sentencia antes de haber creado la oficina 30, el sistema nos devuelve un error. Solo en caso de tener implementada la integridad referencial.

10. De los empleados que trabajan en Cádiz, disminuir su cuota en un 10 %.

```
UPDATE Empleados INNER JOIN Oficinas
       ON Empleados.oficina = Oficinas.oficina
SET cuota = cuota*0.9
WHERE ciudad = "Cádiz";
```

   Otra forma:

```
UPDATE Empleados
SET cuota = cuota*0.9
WHERE oficina IN (SELECT oficina
                  FROM Oficinas
                  WHERE ciudad = "Cádiz");
```

11. Bajar 100 euros el precio de los productos de los que no se han realizado ningún pedido. Hay que tener cuidado que no queden precios negativos.

```
UPDATE Productos
SET precio = GREATEST(precio-100,0)
WHERE NOT EXISTS (SELECT * FROM Pedidos
                  WHERE Pedidos.fab = Productos.idfab AND
                        Pedidos.producto = Productos.idproducto);
```

   Otra forma sin la función:

```
UPDATE Productos
SET precio = precio-100 > 0
WHERE NOT EXISTS (SELECT * FROM Pedidos
                  WHERE Pedidos.fab = Productos.idfab AND
                        Pedidos.producto = Productos.idproducto);
```

12. Modificar el nombre de los empleados para eliminar el segundo nombre o apellido (apellidos) de su nombre.

```
UPDATE Empleados
SET nombre = LEFT(nombre, INSTR (nombre, "␣")-1);
```

13. Cambiar la cuota de todos los empleados a 1000 euros.

```
UPDATE Empleados
SET cuota = 1000;
```

14. Eliminar los pedidos cuyo responsable es el empleado 105.

```
DELETE
FROM pedidos
WHERE resp = 105;
```

15. Eliminar los tres clientes con menor límite de crédito.

```
DELETE
FROM Clientes
ORDER BY limitecredito ASC
LIMIT 3
```

16. En un ejercicio anterior hemos insertado un nuevo empleado con nuestros datos. Eliminar dicho registro.

```
DELETE
FROM Empleados
WHERE nombre = "Pepito";
```

17. Eliminar las oficinas que no tengan empleados. Solución 1

```
DELETE
FROM oficinas
WHERE NOT EXISTS (SELECT *
                  FROM empleados
                  WHERE empleados.oficina = oficinas.oficina);
```

Si la oficina no tiene empleados asignados, no existe ningún empleado con el número de esa oficina.

Solución 2

```
DELETE
FROM oficinas
WHERE oficina NOT IN (SELECT oficina
                      FROM empleados
                      WHERE oficina IS NOT NULL) ;
```

También se puede ver como las oficinas cuyo número no se encuentra entre las oficinas asignados a los empleados. Si hay un empleado sin oficina, no funciona, al devolver una tupla con valor NULO, en el campo oficina la subconsulta.

Solución 3

```
DELETE oficinas.*
FROM oficinas LEFT JOIN empleados
     ON oficinas.oficina= empleados.oficina
WHERE empleados.numemp IS NULL ;
```

Otro planteamiento sería unir los empleados con sus oficinas y que también salgan las oficinas que no tienen empleados (por eso LEFT en vez de INNER) a partir de ahí seleccionamos las filas que no tienen valor en el campo numemp, estas son las no tienen ningún empleado relacionado. Como además el origen está basado en dos tablas es obligatorio poner oficinas.* para indicar que se tienen que borrar las filas de la tabla oficinas y no de empleados.

18. Elimiar cualquier rastro del cliente 2103 (datos y pedidos).

```
DELETE Pedidos.*, Clientes.*
FROM Pedidos RIGHT JOIN Clientes ON Pedidos.clie = Clientes.numclie
WHERE numclie = 2103;
```

    Utilizamos dos tablas en el FROM (JOIN) y borramos de las dos tablas. Se puede utilizar RIGHT por si el cliente 2103 no ha hecho ningún pedido.

19. Eliminar los empleado que han realizado al menos un pedido del fabricante 'ACI'. Cabe destacar que aunque utilizamos dos tablas en el FROM solo eliminamos de la tabla Empleados. Los pedidos no mantienen la integridad referencial.

```
DELETE Empleados.*
FROM Empleados JOIN Pedidos ON Empleados.numemp = Pedidos.resp
WHERE fab = "ACI";
```

    Otra posible solución utilizando subconsulta:

```
DELETE
FROM Empleados
WHERE EXISTS (SELECT *
              FROM PEDIDOS
              WHERE fab=``ACI'' AND Empleados.numemp = Pedidos.resp);
```