MANIPULACIÓN DE REGISTROS: INSERT, DELETE Y UPDATE

I.- INSERT

EJERCICIO 1: Crear una tabla AMIGOS dentro de la BD Empleados2 (Cargar el fichero Empleados2.sql) que posea un ID que sea entero not null y los campos para almacenar el nombre, el teléfono y el trabajo que serán cadenas de longitud 30, 10 Y 15 respectivamente:

```
CREATE TABLE IF NOT EXISTS Amigos
( IdAmigo int not null,
nombre varchar(30),
telefono varchar(10),
 trabajo varchar(15));
```

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

INSERTAR REGISTROS: INSERT INTO

Agrega un registro en una tabla. Esta consulta puede ser de dos tipos:

- Insertar un único registro
- Insertar en una tabla los registros contenidos en otra tabla.

Debe especificarse cada uno de los campos del registro al que se le va a asignar un valor así como el valor para dicho campo. Cuando no se especifica dicho campo, se inserta el valor predeterminado o Null. Los registros se agregan al final de la tabla.

- ***Para insertar un único Registro:***

En este caso la sintaxis es la siguiente:

INSERT INTO Tabla (campo1, campo2, ..., campoN) VALUES (valor1, valor2, ..., valorN)

Esta consulta graba en el campo1 el valor1, en el campo2 y valor2 y así sucesivamente.

EJERCICIO 2: Realice esta inserción: 1,'GOMEZ, JUAN','JUEZ'

```
INSERT INTO AMIGOS (IdAmigo,NOMBRE, TRABAJO) VALUES (1,'GOMEZ, JUAN','JUEZ');
```

Al hacer un SELECT * veremos que el resto de campos estas vacío, es decir a null.

- ***Si se van a insertar todos los campos el formato es:***

```
INSERT INTO Tabla VALUES (valor1, valor2, ..., valorN)
```

EJERCICIO 3: Realice esta inserción: 2, 'SANCHEZ, LUIS','607564321','ESTUDIANTE'

```
INSERT INTO AMIGOS VALUES   (2, 'SANCHEZ, LUIS','607564321','ESTUDIANTE');
```

EJERCICIO 4: Pruebe a insertar un registro sin incluir el ID ¿Qué ocurre?

Con el último formato dará error, ya que hay que introducir todos los campos.

Con el formato del Ejercicio 2:

Si el campo IdAmigo no es primary key insertará 0 en dicho campo, ya que está definido con not null

Si es primary key el primer registro se introducirá como 0, pero los siguientes dará error ya que no puede duplicarse el valor. Si se define como auto_increment se incrementará automáticamente, incluso si se eliminan los registros introducidos en un momento determinado, la siguiente inserción continuará con el siguiente valor al anterior (aunque haya sido borrado)

Para eliminar la clave primaria de un campo que es autoincrementado, primero hay que quitar el auto_increment y luego eliminar la definición de clave primaria:

```
Alter table amigos modify idAmigo int;
Alter table amigos drop primary key;
```

EJERCICIO 5: Inserte 4 registros más CON EL ID CORRELATIVO (3,4,5,6...10) , en los que algunos tengan el teléfono a null. Si no lo ha hecho modifique el campo IdAmigo para que sea auto_increment

```
Alter table amigos modify idAmigo int auto_increment;
```

Si ya hay registros introducidos y estos tienen un idAmigo repetido será necesario borrarlos para poder crear la clave primaria.

```
delete from amigos;
Alter table amigos add constraint primary key (idamigo);
```

Haga un SELECT * de los amigos cuyo teléfono no es null.

```
select * from amigos where telefono IS NOT NULL;
```

Modifica la tabla amigos, añadiendo la restricción que el nombre sea único y obligatorio.

```
Alter table amigos modify nombre varchar(30) unique not null;
```

Elimina la restricción que acabas de introducir:

```
alter table amigos drop index nombre;
```

Para insertar Registros de otra Tabla:

Para insertar todos los campos:

INSERT INTO TablaDestino SELECT * FROM TablaOrigen [WHERE...]

De esta forma los campos de Tabla Origen se grabarán en Tabla, para realizar esta operación es necesario que todos los campos de Tabla Origen estén contenidos en TablaDestino, es decir, que TablaDestino posea todos los campos de TablaOrigen (coincidan en orden y tipo, no es necesario que el nombre de los campos sea igual).

En este tipo de inserción de datos hay que tener especial cuidado con los campos contadores o auto_incrementados puesto que al insertar un valor en un campo de este tipo se escribe el valor que contenga su campo homólogo en la tabla origen, no incrementándose como le corresponde.

EJERCICIO 6a: Cree la tabla amigos2 que sea una copia de amigos, pero sin registros:

create table if not exists amigos2 like amigos;

EJERCICIO 6b: Inserte todos los registros de amigos a amigos2:

Insert into amigos2 select * from amigos;

Para insertar sólo algunos campos

Se puede también especificar los campos que deseo copiar, y de esta forma que tabla destino sólo albergue los campos que me interesan.

En este caso la sintaxis es:

INSERT INTO TablaDestino [(campo1, campo2, , campoN)] SELECT [campo1, campo2, , campoN] FROM TablaOrigen [WHERE...]

EJERCICIO 6c: Inserte todos los registros de amigos a amigos2, pero sólo los campos nombre, teléfono y trabajo:

insert into amigo2 (nombre,tel,trabajo) select nombre,telefono,trabajo from amigos;

Se podrá realizar si no tiene establecida la primary key o no se repite el campo idAmigo

En este caso se seleccionarán los campos 1,2,..., n de la tabla destino y se grabarán en los campos 1,2,.., n de la Tabla origen. La condición SELECT puede incluir la cláusula WHERE para filtrar los registros a copiar. Si Tabla destino y Tabla Origen poseen la misma estructura podemos simplificar la sintaxis.

EJERCICIO 7: Fijándonos en la descripción de AMIGOS insertemos desde la tabla EMPLE solo los campos nombre (nombre y apellido de emple), trabajo (oficio de emple) en amigos2.

```
insert into amigo2 (nombre,trabajo) select concat(nombre,',',apellido),oficio from emple;
```

Igual que en el anterior, si no está establecida la clave primaria el idAmigo tomará valor 0, si está establecida y es auto_increment se incrementará automáticamente.

Compruebe que se ha producido la copia.

También podemos insertar registros nuevos en una tabla especificando alguno de los campos con un select y otros como valores fijos.

EJERCICIO 8: Hagamos la misma inserción anterior, pero en teléfono insertemos el valor '9111111' en todos los registros.

```
insert into amigos2 (nombre,telefono,trabajo) select concat(nombre,',',apellido),'9111111',oficio from emple;
```

EJERCICIO 9: Fijándonos en la descripción de AMIGOS insertemos desde la tabla AGENDA y EMPLE. Los campos nombre (nombre y apellido de emple), trabajo (oficio de emple) y teléfono (de agenda), pero sólo de los que pertenecen al departamento 20.

```
insert into amigo2 (idamigo,nombre,telefono,trabajo) select distinct  emp_no, concat(nombre,',',apellido), telefono,oficio from emple,agenda where emp_no=num_emp and dept_no=20;
```

Compruebe que se ha producido la copia

También podemos insertar registros nuevos en una tabla especificando alguno de los campos con un select y otros como valores fijos.

```
INSERT INTO TablaDestino SELECT DISTINCT (VALOR FIJO1, CAMPO1,....) FROM TablaOrigen [WHERE]
```

EJERCICIO 10: Insertemos al Presidente de la empresa en nuestra agenda con el select anterior, pero en oficio pongamos el valor fijo "JEFAZO"

```
insert into amigo2 (nombre,tel,trabajo) select distinct concat(nombre,',',apellido),telefono,'JEFAZO' from emple,agenda where emp_no=num_emp and oficio='Presidente' ;
```

EJERCICIO 11: ¿QUÉ HACE ESTA SENTENCIA?

```
INSERT INTO AMIGO2
SELECT null, 'PEREZ, RAQUEL', TELEFONO, TRABAJO
FROM AMIGOS WHERE TELEFONO=(SELECT MAX(TELEFONO) FROM AGENDA);
```

Inserta tantos registros como tenga amigos con trabajos distintos que tengan el número de teléfono mayor de agenda y como idamigo le asignará el siguiente.

Modificar las tablas anteriores, de forma que en la agenda sólo tendremos registros de empleados y en amigos también. Establecer las claves primarias y ajenas para que si se elimina un registro de empleado se elimine en cascada todos los registros correspondientes en las tablas amigos y agenda.

alter table emple add primary key (emp_no);
alter table agenda modify num_emp int(4) primary key;
alter table agenda add foreign key (num_emp) References Emple (emp_no) on delete cascade on update cascade;
Para establecer la clave ajena en amigo2, primero habrá que eliminar los registros que no tengan correspondencia con un empleado existente:
delete from amigo2 where idAmigo not in(select emp_no from emple);
Cambiar el tipo de dato, para que sea el mismo en ambos campos:
alter table amigo2 modify idamigo int(4);
alter table amigo2 add foreign key (idAmigo) References Emple (emp_no) on delete cascade on update cascade;

EJERCICIO 12: Vuelve a realizar la inserción del ejercicio anterior, pero teniendo en cuenta que el idAmigo ya no es un campo auto_incrementado y las restricciones que se acaban de añadir.

INSERT INTO AMIGO2
SELECT emp_no, 'PEREZ, RAQUEL', TELEFONO, oficio
FROM AGENDA,emple WHERE num_emp=emp_no and TELEFONO=(SELECT MAX(TELEFONO) FROM AGENDA);

EJERCICIO 13: Inserta en EMPLE al empleado 1111, 'GARCIA' de oficio ANALISTA, con fecha de alta la actual (SYSDATE), en el departamento 30, con salario y comisión iguales al que tiene el sueldo mínimo del departamento 30. El resto de valores a nulo.

insert into emple (emp_no, apellido, oficio, fecha_alt, salario, comision, dept_no) select 1111, 'Garcia', 'Analista', sysdate(), salario,comision,30 from emple where salario=(select min(salario) from emple where dept_no=30);

EJERCICIO 14: Inserta en EMPLE al empleado 1221, 'PARADA' de DIRECTOR, del departamento 40, con fecha de alta la actual(SYSDATE), con salario igual al máximo sueldo de la empresa que no sea el del

presidente. El resto de valores a nulo.

```
insert into emple (emp_no, apellido, oficio, fecha_alt, salario, dept_no) select 1221, 'Parada' ,'Director', sysdate(),salario, 40 from emple where salario=(select max(salario) from emple where oficio not like 'presidente');
```

- Daría error si hubiera más de un registro que cumpliera la condición, ya que la clave primaria no se puede repetir.

Si la clave fuera un campo auto_increment se podría hacer:

```
insert into emple (emp_no, apellido, oficio, fecha_alt, salario, dept_no,) select null,'Parada','Director', sysdate(),salario ,40  from emple where salario=(select max(salario) from emple where oficio not like 'presidente');
```

EJERCICIO 15: Inserta en EMPLE al empleado 9999, 'CLEMENTE' de 'VENDEDOR', en el departamento con menos empleados, con fecha de alta la actual(SYSDATE), y salario y comisión a 0. El resto de valores a nulo.

```
insert into emple (emp_no, apellido, oficio, fecha_alt, salario, comision, dept_no) select 9999, 'Clemente', 'Vendedor', sysdate(),0, 0 ,dept_no from emple group by  dept_no having count(*) <=all(select count(*) from emple group by  dept_no);
```

- **CREAR UNA TABLA NUEVA (Sólo la estructura)**

SINTAXIS 1

```
CREATE TABLE  nueva_tabla LIKE TablaOrigen;
```

Crea la tabla con la misma estructura, sin registros, pero con las restricciones.

- **CREAR UNA TABLA NUEVA INSERTANDO UNA SELECCIÓN DE REGISTROS**

SINTAXIS 2

```
CREATE TABLE nueva_tabla AS SELECT campos FROM tabla_origen [WHERE];
```

Crea la tabla con la misma estructura y copia todos los registros pero sin las restricciones. Habría que añadirle las restricciones posteriormente.

EJERCICIO 16: Vamos a crear una tabla TRABAJADORES sólo con los amigos cuya profesión no es nula. Añadir el código y comprobar el resultado.

```
create table TRABAJADORES AS SELECT *  FROM AMIGOS WHERE TRABAJO IS NOT NULL;
ALTER TABLE TRABAJADORES ADD PRIMARY KEY (IDAMIGO);
```

```
create table if not exists trabajadores like amigos;
insert into trabajadores select * from amigos where trabajo is not null;
```

EJERCICIO 17: Vamos a crear una tabla MOVILES sólo con los campos nombre y Teléfono de los amigos cuyo teléfono comience por 6

```
create table moviles AS SELECT IDAMIGO,NOMBRE,TELEFONO  FROM AMIGOS WHERE  telefono like '6%';
 ALTER TABLE MOVILES ADD PRIMARY KEY (IDAMIGO);
```

EJERCICIO 18: Crear una tabla DIRECTORES sólo con los campos apellido y departamento de los directores de emple. Añadir el código y comprobar el resultado.

```
mysql> create table DIRECTORES as select apellido,dept_no from emple where oficio like 'Director';
mysql> alter table directores add emp_no int(4) primary key auto_increment;
mysql> alter table directores modify emp_no int(4) first;
```

II.- DELETE

Una consulta de eliminación suprime los registros de una o más de las tablas listadas 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. Su sintaxis es:

DELETE FROM Tabla WHERE condición

Una vez que se han eliminado los registros utilizando una consulta de borrado, no puede deshacer la operación, a no ser que use rollback. Si desea saber qué registros se eliminarán, primero examine los resultados de una consulta de selección que utilice el mismo criterio y después ejecute la consulta de borrado. Mantenga copias de seguridad de sus datos en todo momento.
Si elimina los registros equivocados podrá recuperarlos desde las copias de seguridad.

EJERCICIO 1:

Cree una tabla EMPLE2 que sea una copia de EMPLE con los comandos aprendidos anteriormente
Cree una tabla DEPART2 que sea una copia de DEPART con los comandos aprendidos anteriormente

```
create table EMPLE2 LIKE EMPLE;
```

```
INSERT INTO EMPLE2 SELECT * FROM EMPLE;
create table DEPART2 LIKE DEPART;
INSERT INTO DEPART2 SELECT * FROM DEPART;
```

EJERCICIO 2:

Antes de ejecutar, compruebe los registros a borrar con una sentencia SELECT * de la condición dada ¿QUÉ REGISTO BORRA ESTA SENTENCIA?.

```
delete FROM  EMPLE2 WHERE OFICIO='DIRECTOR' AND DEPT_NO=40;
```

```
Borra los empleados de Emple2 que sean del dpto 40 y directores.
```

EJERCICIO 3:

Borre de EMPLE2 los empleados que sean ANALISTAS y no del departamento 20

```
DELETE FROM  EMPLE2 WHERE OFICIO='analista' AND DEPT_NO!=20;
```

EJERCICIO 4:

Borre de EMPLE2 los empleados que sean VENDEDORES y del departamento 30

```
delete from EMPLE2 WHERE OFICIO='vendedor' AND DEPT_NO=30;
```

~~EJERCICIO 5~~(*NOTA):

Borre de EMPLE2 los empleados cuyo salario sea igual al sueldo menor de su departamento

1/ Visualizar el sueldo menor de cada departamento

select dept_no,salario from emple2 e where salario = (select min(salario) from EMPLE2 where e.dept_no=dept_no group by dept_no);

```
mysql> delete from emple2 where salario = (select min(salario) from EMPLE2 e where e.dept_no=dept_no
group by dept_no);
ERROR 1093 (HY000): You can't specify target table 'emple2' for update in FROM clause
```

/* Menor a cualquier salario de la empresa*/

mysql> select * from emple2 where salario<=ANY(select min(salario) from emple2);

EJERCICIO 5(*):

Borre de EMPLE2 los empleados cuyo salario sea menor al sueldo medio de su departamento

1/ Visualizar dentro de cada departamento los salarios menores a la media

```
mysql> Select salario, dept_no from emple2 e where  salario < (select avg(salario) from emple2  where
```

e.dept_no=dept_no group by dept_no);

mysql> delete from EMPLE2 WHERE salario < (select avg(salario) from emple2 e where dept_no=e.dept_no group by dept_no);

ERROR 1093 (HY000): You can't specify target table 'EMPLE2' for update in FROM clause

OTRO ENUNCIADO: Borrar el empleado que tenga el salario menor al salario medio de la empresa

Hay que utilizar una variable para realizar la consulta y después eliminar el resultado

```
set @SalarioMedio=(select avg(salario) from emple2);
delete from EMPLE2 WHERE salario <@SalarioMedio;
```

EJERCICIO 6(*):

Borre de EMPLE2 el empleado con la máxima comisión

```
mysql> delete from emple2 where comision=(select max(comision) from emple2);
ERROR 1093 (HY000): You can't specify target table 'emple2' for update in FROM clause
mysql> set @maxComision=(select max(comision) from emple2);
Query OK, 0 rows affected (0.02 sec)
mysql>  delete from EMPLE2 WHERE comision >= @maxComision;
Query OK, 1 row affected (0.04 sec)
```

EJERCICIO 7(*):

Borre de EMPLE2 los empleados que pertenecen al departamento con más número de personas

1/ Departamento con mayor número de personas

```
mysql> select dept_no, count(*)  from emple2 group by dept_no having count(*) >= ALL(select count(*) from emple2 group by dept_no );
mysql> delete from emple2 where dept_no in (select dept_no from emple2 group  by dept_no having count(*) >= ALL(select count(*) from emple2 group by dept_no ));
ERROR 1093 (HY000): You can't specify target table 'emple2' for update in FROM clause
```

Si la subconsulta devuelve un solo resultado, es decir, si sólo hay un departamento que tenga el mayor número de personas, se podría hacer:

```
mysql> set @DptomasPersonas:=(select dept_no from emple group by dept_no having count(*)>=ALL (select count(*)  from emple group by dept_no));
mysql> delete from emple2 where dept_no=@DptomasPersonas;
```

EJERCICIO 8:

Borre de DEPART2 los departamentos con menos de 4 empleados

1/ Visualizar los departamentos con menos de 4 empleados:

```
mysql> select d.dept_no,d.dnombre, count(*) from depart2 d,emple2 e where e.dept_no=d.dept_no group by e.dept_no having count(*)<4;
mysql> select dept_no,  dnombre from depart2 where dept_no in (select dept_no from emple2 group by dept_no having count(*) <4);
2/ Eliminar
mysql> delete from depart2 where dept_no in (select dept_no from emple2 group by dept_no having count(*) <4);
```

EJERCICIO 9:

Borre de DEPART2 el departamento 40 sólo si no tiene empleados

```
mysql> delete from depart2 where dept_no=40 and dept_no NOT IN(select dept_no from emple2);
```

EJERCICIO 10:

Borre de AGENDA los empleados del departamento 20 cuyo teléfono es nulo

```
mysql> delete from agenda where telefono is NULL and num_emp IN(select emp_no from emple where dept_no=20);
```

EJERCICIO 11:

Borre los empleados que pertenecen a la Sede Central

Primero podemos realizar la consulta:

```
mysql> select e.nombre,c.nomce from emple3 e, depart d, centros c where e.dept_no=d.dept_no and d.numce=c.numce and nomce like 'central';

mysql> delete from emple3 where dept_no =(select dept_no from depart2 where  numce = (select numce from centros where nomce like 'central'));
```

Ejercicio 12:

El anterior pero solo si está en Madrid

```
mysql> select e.nombre, nomce, ciudad from emple e, depart d, centros c where e.dept_no=d.dept_no and d.numce=c.numce and nomce like 'central' and ciudad like Madrid';

mysql> delete from emple2 where dept_no =(select dept_no from depart2 where numce = (select numce
```

```
from centros where nomce like 'central' and ciudad like 'madrid'));
```

Ejercicio 13:

Borre los centros que tengan empleados viviendo en Madrid

```
mysql> select nombre, nomce,ciudad,d.dept_no from centros c, emple e, depart d where d.numce=c.numce and e.dept_no=d.dept_no and ciudad='Madrid';
mysql> select e.nombre,e.emp_no,c.nomce from emple e, depart d, centros c where e.dept_no =d.dept_no and d.numce=c.numce and nomce like 'central' and .ciudad like 'Madrid';
```

III.- UPDATE

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico. Su sintaxis es:

```
UPDATE Tabla SET Campo1=Valor1, Campo2=Valor2, CampoN=ValorN WHERE Criterio
```

UPDATE es especialmente útil cuando se desea cambiar un gran número de registros o cuando éstos se encuentran en múltiples tablas. Puede cambiar varios campos a la vez.

UPDATE no genera ningún resultado. Para saber qué registros se van a cambiar, hay que examinar primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecutar la consulta de actualización.

Si en una consulta de actualización suprimimos la cláusula WHERE todos los registros de la tabla señalada serán actualizados.

EJERCICIO 1:

Cree una tabla EMPLE3 que sea una copia de EMPLE con los comandos aprendidos anteriormente

Cree una tabla DEPART3 que sea una copia de DEPART con los comandos aprendidos anteriormente

```
mysql> create table emple3 like emple;
mysql> create table depart3 like depart;
mysql> insert into emple3 select * from emple;
mysql> insert into depart3 select * from depart;
```

EJERCICIO 2:

Antes de ejecutar, compruebe los registros a actualizar con una sentencia SELECT * de la condición dada ¿QUÉ CAMBIOS HACE ESTA SENTENCIA?.

```
UPDATE Emple3 SET APELLIDO='Ruiz' WHERE APELLIDO='SALA';
```

Modifica todos los apellidos Sala por Ruiz.

¿Y ESTA?

UPDATE Emple2 SET APELLIDO='RUIZ';

Modifica todos los apellidos a Ruiz

Ejecute ROLLBACK para recuperar la situación anterior. Comente los resultados obtenidos.

Si la variable autocommit está activa (ON/1/true) la actualización última no se puede deshacer, si la variable autocommit estuviera desactivada volveríamos a la situación anterior.

EJERCICIO 3:

Aumente el salario a todos los empleados un 10%

```
update emple3 set salario=salario*1.1;
```

EJERCICIO 4:

Cambie las comisiones nulas por 0

```
update emple3 set comision=0 where comision is null;
```

EJERCICIO 5:

Cambie los oficios a mayúsculas y la primera letra de todos los apellidos a mayúscula y las siguientes a minúsculas

```
update emple3 set oficio = upper(oficio), apellido= concat(upper(substr (apellido,1,1)), lower(substr(apellido,2)));
```

EJERCICIO 6:

Cambie la fecha de alta a la actual a todos los DIRECTORES

```
update emple3 set fecha_alt=sysdate() where oficio='Director';
```

EJERCICIO 7:

Cambiar el centro del departamento de INVESTIGACION a la sede que está en Sevilla.

```
mysql> update depart3 set numce=(select numce from centros where ciudad='Sevilla') where dnombre
```

```
like 'INVESTIGACION';
```

LOS VALORES PARA LOS CAMPOS TAMBIEN PUEDEN OBTENERSE DE UN SUBSELECT

```
UPDATE TABLA SET CAMPO= (SELECT CAMPO ……..) [WHERE]
```

EJERCICIO 8(*):

Cambie la fecha de alta del empleado 'GIL' a la fecha de alta de 'NEGRO'

```
mysql> update emple3 set fecha_alt=(select fecha_alt from emple3 where apellido='Negro') where apellido='Gil';
ERROR 1093 (HY000): You can't specify target table 'emple3' for update in FROM clause
```

```
mysql> set @fecha=(select fecha_alt from emple3 where apellido='Negro');
Query OK, 0 rows affected (0.01 sec)
mysql> update emple3 set fecha_alt=@fecha where apellido='Gil';
Query OK, 1 row affected (0.39 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

EJERCICIO 9(*):

Cambie el departamento de 'GIL' al departamento con menos empleados.

```
mysql> update emple3 set dept_no=(select dept_no from emple3 group by dept_no having count(*) <=ALL (select count(*) from emple3 group by dept_no)) where apellido='Gil';
ERROR 1093 (HY000): You can't specify target table 'emple3' for update in FROM clause

La solución podría estar en utilizar una variable para guardar el departamento que tenga menor número de empleados, pero de nuevo, esto no servirá si hay más de un departamento que tenga el menor número de empleados.
Suponiendo que sólo hay un departamento que tiene el menor número de empleados, se podría hacer:

mysql> set @deptMenosEmple=(select dept_no from emple3 group by dept_no having count(*) <=ALL (select count(*) from emple3 group by dept_no));
Query OK, 0 rows affected (0.01 sec)

mysql> update emple3 set dept_no=@deptmenosEmple where apellido='Gil';
```

```
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

EJERCICIO 10:

Doblar el salario de los que trabajan en CONTABILIDAD o en VENTAS

```
mysql> update emple3 set salario=2*salario where dept_no IN (select dept_no from depart3 where dnombre like 'Contabilidad' or dnombre like 'Ventas');
```

Nota (*) Los ejercicios marcados con * pueden no ejecutarse en mysql, se escribirá la sentencia en el fichero, aunque para probarla en vez de realizar un delete/update se realizará una select.

http://www.mundoracle.com/funciones-sql.html?Pg=sql_plsql_3.htm