MANIPULANDO DATOS 1

DAW 1 – Bases de Datos

Práctica 3.1

Introducción y manipulación de datos

Marco Valiente Rodríguez
Mvalienter2501@educantabria.es

Índice

Contenido

Planteamiento del Problema

Este nuevo ejercicio comienza ya que una vez que hemos conseguido implementar de forma correcta dentro de MySQL una gran variedad de bases de datos. Con este contenido que hemos conseguido hacer anteriormente, ahora vamos a centrarnos en rellenar nuestras bases de datos, en introducirles contenido. Esto se va a empezar a través de un ejercicio sencillo que servirá de introducción en esta misma práctica.

a. Visión de nuestro modelo

Nuestro modelo debe seguir la siguiente lógica:

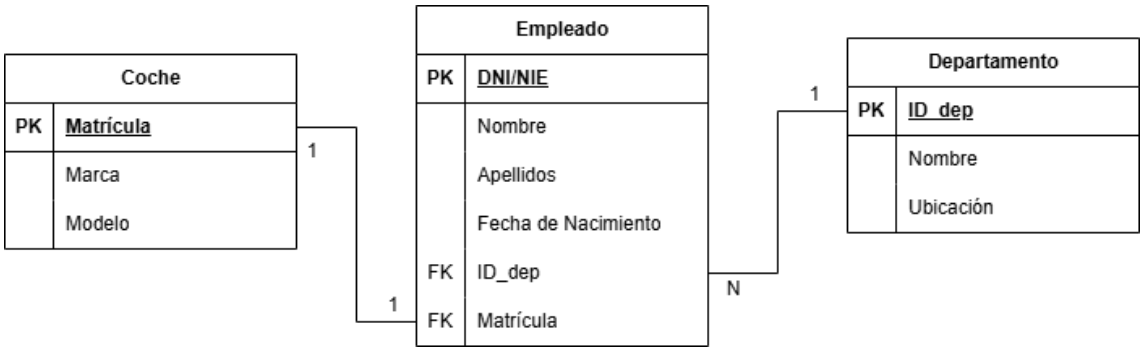
• Contamos con varios empleados. Cada empleado se identifica por su DNI/NIE, nombre, apellidos y fecha de nacimiento.

• Cada empleado trabaja en un departamento. Puede haber varios empleados trabajando en el mismo departamento, pero un empleado sólo puede trabajar en un departamento. Un departamento se identifica por su id, nombre y ubicación.

• Algunos empleados tienen un coche de la empresa, que sólo es utilizado por ese empleado. Un coche de empresa se identifica por su matrícula, su marca y su modelo.

b. Modelo relacional

De este modelo que hemos visualizado en el apartado anterior, ahora vamos a definir de forma directa el modelo relacional, directamente, sin pasar por el modelo entidad relación.



A la hora de resolver esta relación, no hay mucho que añadir, a excepción de que el ID de departamento se llama ID_dep para darle más luz y claridad sobre qué es ese dato, y con esto, nos centramos a darle únicamente los atributos mínimos requeridos.

Otro dato a comentar que sí puede ser relevante es que tenemos una relación de 1 a 1. Un Coche es de un único empleado. Se opta por poner la FK sobre Empleado, ya que puede ser más frecuente que un empleado sea despedido o se cambie de empresa, y que el coche se quede en la nuestra, y que pase a ser de otro empleado a futuro. Como este modelo tiene como intención ser sencillo para introducir datos, nos vamos a limitar a que la relación sea de 1 a 1 por designio.

c. Creación del código en MySQL

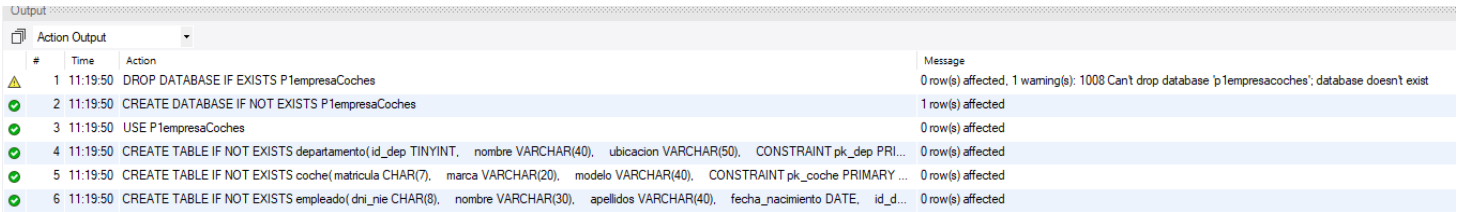
Una vez que tenemos el diagrama claro, nos queda dar el salto a MySQL.

```
DROP DATABASE IF EXISTS P1empresaCoches;
CREATE DATABASE IF NOT EXISTS P1empresaCoches;
USE P1empresaCoches;

CREATE TABLE IF NOT EXISTS departamento(
    id_dep TINYINT,
    nombre VARCHAR(40),
    ubicacion VARCHAR(50),
    CONSTRAINT pk_dep PRIMARY KEY (id_dep)
);
CREATE TABLE IF NOT EXISTS coche(
    matricula CHAR(7),
    marca VARCHAR(20),
    modelo VARCHAR(40),
    CONSTRAINT pk_coche PRIMARY KEY (matricula)
);
CREATE TABLE IF NOT EXISTS empleado(
    dni_nie CHAR(8),
    nombre VARCHAR(30),
    apellidos VARCHAR(40),
    fecha_nacimiento DATE,
    id_dep TINYINT,
    matricula CHAR(7),
    CONSTRAINT pk_empleado PRIMARY KEY (dni_nie),
    CONSTRAINT coche_fk_empleado FOREIGN KEY (matricula) REFERENCES
coche(matricula),
    CONSTRAINT dep_fk_empleado FOREIGN KEY (id_dep) REFERENCES
departamento(id_dep)
);
```

Se define al ID de Departamento como TINYINT y a matrícula como un CHAR de 7 como son en España, y también el DNI/NIE como otro CHAR de 8, aunque sean de 9 en realidad, en este ejemplo no posee tanta relevancia.

Tenemos que el código se ejecutó correctamente a la primera también.


Output

Action Output

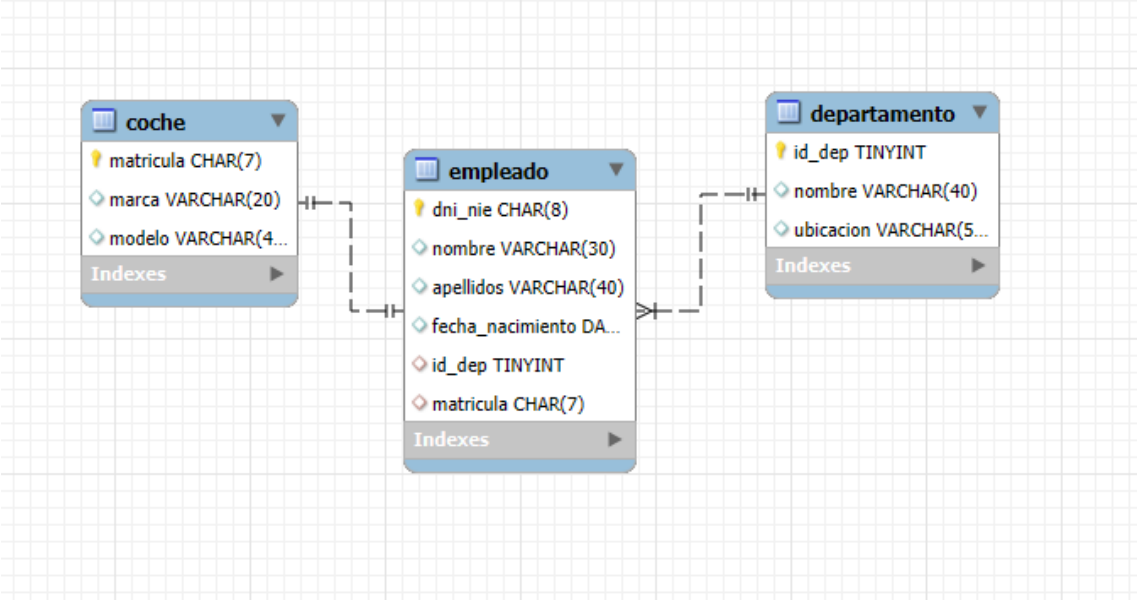
#	Time	Action	Message
1	11:19:50	DROP DATABASE IF EXISTS P1empresaCoches	0 row(s) affected, 1 warning(s): 1008 Can't drop database 'p1empresacoches'; database doesn't exist
2	11:19:50	CREATE DATABASE IF NOT EXISTS P1empresaCoches	1 row(s) affected
3	11:19:50	USE P1empresaCoches	0 row(s) affected
4	11:19:50	CREATE TABLE IF NOT EXISTS departamento(id_dep TINYINT, nombre VARCHAR(40), ubicacion VARCHAR(50), CONSTRAINT pk_dep PRI...	0 row(s) affected
5	11:19:50	CREATE TABLE IF NOT EXISTS coche(matricula CHAR(7), marca VARCHAR(20), modelo VARCHAR(40), CONSTRAINT pk_coche PRIMARY ...	0 row(s) affected
6	11:19:50	CREATE TABLE IF NOT EXISTS empleado(dni_nie CHAR(8), nombre VARCHAR(30), apellidos VARCHAR(40), fecha_nacimiento DATE, id_d...	0 row(s) affected

El aviso estando ahí debido a que la tabla no existía de antes.

d. Ingeniería reversa

Ahora seguimos con la ingeniería reversa.

Debido a que no se puede especificar en una propia relación 1 a 1, se ajustará manualmente una vez dibujada la relación 1 a 1.



e. Inserción de datos

Con las tablas ya diseñadas, se puede dar el salto a la inserción de datos.

Se pide lo siguiente:

1. Crea al empleado #1, al departamento #1 y al coche de empresa #1.
2. Crea 3 empleados más, utilizando una única sentencia.
3. Crea el coche de empresa #2, el cual sólo se identifica por su matrícula.
4. Cambia el nombre del empleado #2 a "Carlos".
5. Cambia el nombre del departamento #1 a "IT".
6. Cambia la marca a "Seat" y el modelo a "Ibiza" en el coche de empresa #2.
7. Elimina al empleado #4.
8. Elimina coche de empresa #2.
9. Elimina todos los registros.

Nuestro objetivo irá por partes, a crear individualmente las ordenes de creación.

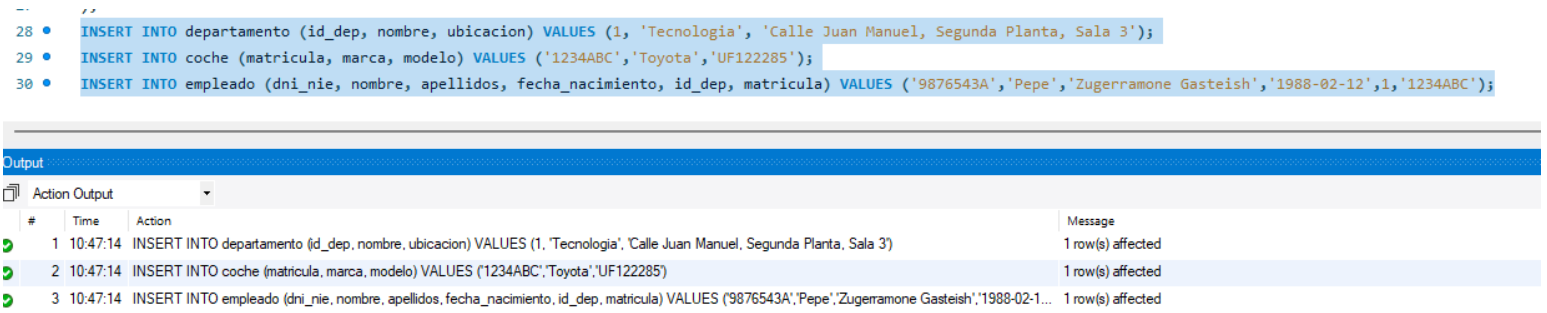
f. Código de las inserciones de datos

1. Código primera adición

Se nos requiere crear al primer departamento, coche y empleado.

```
INSERT INTO departamento (id_dep, nombre, ubicacion) VALUES (1,
'Tecnologia', 'Calle Juan Manuel, Segunda Planta, Sala 3');
INSERT INTO coche (matricula, marca, modelo) VALUES
('1234ABC','Toyota','UF122285');
INSERT INTO empleado (dni_nie, nombre, apellidos, fecha_nacimiento,
id_dep, matricula) VALUES ('9876543A','Pepe','Zugerramone
Gasteish','1988-02-12',1,'1234ABC');
```

Este código logra añadir las siguientes personas:

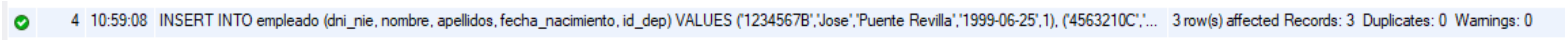


2. Código adición múltiple

Se nos requiere añadir a 3 usuarios más, con la restricción de que debemos usar una única sentencia. Vamos a probar no dándoles a los empleados una FK de coche, y dejando ese campo vacío.

```
INSERT INTO empleado (dni_nie, nombre, apellidos, fecha_nacimiento,
id_dep) VALUES ('1234567B','Jose','Puente Revilla','1999-06-25',1),
('4563210C','Pablo','Emanuel Marisco','2001-03-30',1),
('6554987D','Enricacio','Goycojea','1977-12-12',1);
```

Y consigue ejecutarse exitosamente:

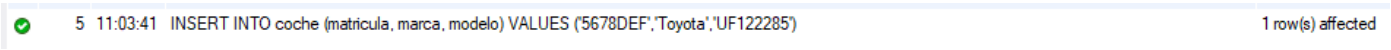


3. Código adición con solo PK diferencia

Al pedir que sólo se identifique por matrícula, sabemos que el resto de datos, por ejemplo, pueden ser exactamente iguales al primero de estos.

```
INSERT INTO coche (matricula, marca, modelo) VALUES
('5678DEF','Toyota','UF122285');
```

Se consigue hacer de forma exitosa:



4. Código primera modificación

Ahora nos toca hacer las primeras modificaciones. Para esto, vamos a cambiar el nombre de nuestro segundo empleado, a que sea simplemente “Carlos”.

```
UPDATE empleado SET nombre = 'Carlos' WHERE dni_nie = '1234567B';
```

Y tal como fue propuesto, tenemos tal que:

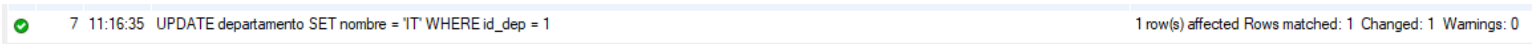
Hay un solo cambio en una sola columna, por lo que se confirma que se ha realizado exitosamente.

5. Código modificación de otros datos

Ahora empezamos intentando cambiar otros datos, como el nombre del departamento. Al solo haber un dato, nos podemos permitir cambiar todos los datos a la vez, pero no nos interesa hacerlo, ya que puede que ese código de problemas a futuro.

```
UPDATE departamento SET nombre = 'IT' WHERE id_dep = 1;
```

Nuestra resolución funcionó:



6. Código de modificación de varias columnas de una tabla

El cambio de los datos del coche #2 comienzan ahora, donde tenemos que conseguir editar de forma que los datos se cambien y se vuelvan un Seat Ibiza.

```
UPDATE coche SET marca = 'Seat', modelo = 'Ibiza' WHERE matricula =
'5678DEF';
```

Este código devuelve lo siguiente en la consola:

Al solo haber modificado uno, podemos intuir que la única columna modificada es la columna del coche #2, por lo tanto, se ha ejecutado correctamente.

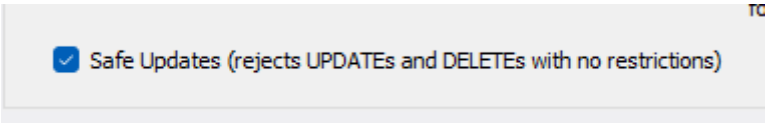
7. Código primera eliminación

Ahora empezamos con la eliminación de datos. Lo primero de todo es desarrollar que el código esté bien, y después, lo vamos a ejecutar dos veces. La primera vez, va a ser sin hacer ningún cambio a las medidas de seguridad, lo cual significa que nos debe salir una advertencia, y la segunda vez es una vez que hayamos habilitado la potestad de poder eliminar datos.

```
DELETE FROM empleado WHERE dni_nie = '6554987D';
```

Primera ejecución:

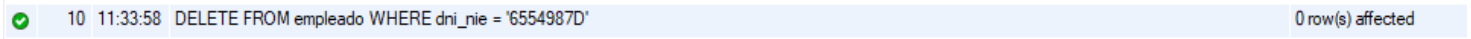
IMPORTANTE:



Ha conseguido borrarlo de forma exitosa a pesar de que tiene activado el modo Safe Updates.

Para poder borrar el dato de nuevo, se necesita volver a reintroducirlo ya que no existe. Para eso, hacemos la prueba, de intentar borrar el dato que no existe una segunda vez con esto desactivado.

Segunda ejecución:



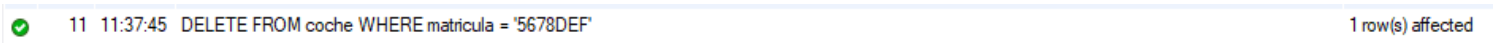
No se han borrado columnas porque ya existe. Y efectivamente esta vez estaba desactivado. De ahora en adelante, vamos a proceder con el ejercicio con el modo seguro activado de nuevo.

8. Código eliminación de datos de nuevo

Vamos a repetir la eliminación de datos, con el modo seguro siguiendo activo, y vamos a comprobar si salta algún fallo o si sigue dando permisos para las eliminaciones de datos.

```
DELETE FROM coche WHERE matricula = '5678DEF';
```

Ejecución:



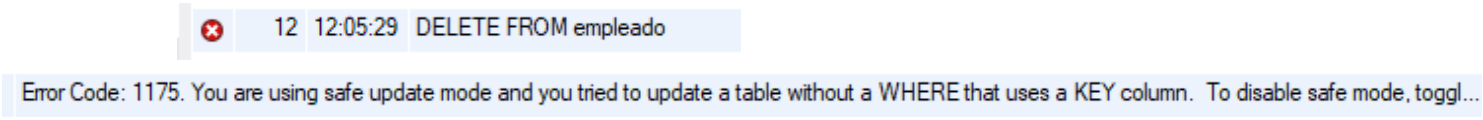
Sigue funcionando a pesar de que el modo seguro está activado.

9. Código eliminación de todos los datos

Se nos pide ahora la eliminación de todos los registros. El modo seguro sigue estando activo, así que en esta de aquí vamos a comprobar definitivamente si el modo seguro activo de prohíbe o no realizar esa actividad.

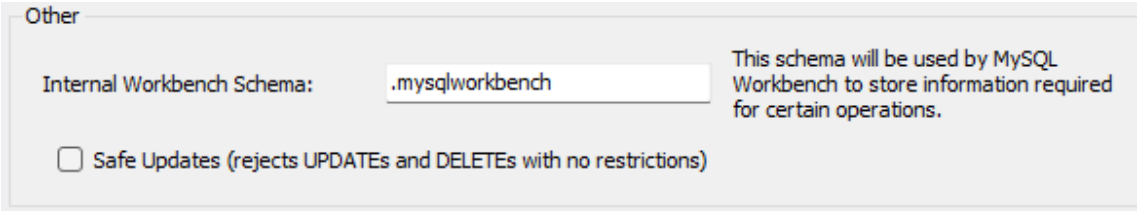
```
DELETE FROM empleado;
DELETE FROM coche;
DELETE FROM departamento;
```

Primera Ejecución:



Error. Finalmente. El modo seguro permite borrar datos siempre y cuando se use el WHERE y la condición sea una Primary Key o Foreign Key, por lo que se puede leer en el mensaje de consola. Es decir, solo puede modificar datos concretos, no te permite cambiar más datos de los necesarios.

Segunda Ejecución:



⊗	13	12:29:27	DELETE FROM empleado	Error Code: 1175. You are using safe update mode and you tried to update
⊗	14	12:29:52	DELETE FROM empleado	Error Code: 1175. You are using safe update mode and you tried to update
⊗	15	12:30:52	DELETE FROM empleado	Error Code: 1175. You are using safe update mode and you tried to update

Incluso cuando está desactivado, este te prohíbe borrarlos sin restricción. El archivo indica que sigue activado, aunque en ajustes muestra que no está, a pesar de que se ejecuta. Esto se debe a que se ha de reiniciar.

Action Output

	#	Time	Action	Message
✓	1	13:27:35	USE P1empresaCoches	0 row(s) affected
✓	2	13:27:38	DELETE FROM empleado	3 row(s) affected
✓	3	13:27:38	DELETE FROM coche	1 row(s) affected
✓	4	13:27:38	DELETE FROM departamento	1 row(s) affected

Y al reiniciar, se puede ejecutar correctamente, habiendo llegado de forma satisfactoria al final del ejercicio.

g. Revisión y Validación

Tras una jornada de pruebas y experimentaciones con el código en MySQL, hemos conseguido lograr introducir los datos dentro de la propia práctica sin mayores complicaciones.

Centrándonos en hacer una tabla sencilla a propósito debido a esto, y por eso, algunas de las relaciones se podrían trabajar más, como, por ejemplo, con las relaciones de coches y empleados, pero al ser el centro de esta actividad la inserción de datos, vamos a omitir y simplificar nuestra base de datos.

En cuanto a los datos introducidos, al intentar probar a ver si da lugar a fallos, se han conseguido expandir los conocimientos en cuanto a lo que realmente protege la funcionalidad del modo seguro.

Leyendo la consola, también se puede deducir que todos los comandos realizados han funcionado correctamente en su funcionalidad.

Bibliografía

- Recurso del curso de Desarrollo de Aplicaciones Web, Introducción al Tema 3.