PARTIENDO DEL MODELO CONCEPTUAL

DAW 1 – Bases de Datos

Práctica 2

Esta práctica es una con menos contenido que otras prácticas, cuyo objetivo es la práctica con el código y la aplicación de MySQL.

Marco Valiente Rodríguez
Mvalienter2501@educantabria.es

Índice

Contenido

Planteamiento del Problema

Disponemos de la siguiente figura:

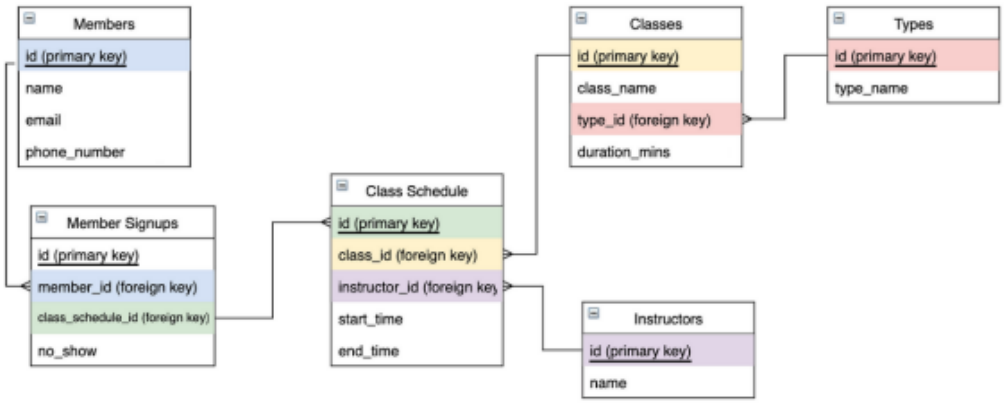


Figura 1. Modelo relacional

Y el objetivo de esta práctica es la de manejar SQL a intentar replicar y hacerla cambios.

a. Planteamiento previo a la replicación a través de Código en MySQL

Empezamos con el diseño de la base de datos. Para esto, vamos a empezar abriendo MySQL y escribiendo el código, intentando tener la menor cantidad de fallos posibles.

Al no pedir especificación a los datos, sino que sólo sean adecuados, se van a escoger datos genéricos y medianamente razonables sin pensar o necesitar mucho en qué pensar para estos.

Para los ID, se toma un valor arbitrario asumiendo un flujo moderado de miembros y de profesores. Las duraciones son TINYINT. Las fechas son DATETIME. VARCHARS para los nombres que tengan un rango de 30 caracteres, excepto a personas que se les otorgará 50. Tomamos a no_show como BOOLEAN (Es decir, TINYINT (1)).

Tomaremos la asunción que no vamos a depender de fuentes de datos externos y que estos datos han de ser únicamente basados en nuestro modelo proporcionado, a la hora de escoger el tamaño de estos datos.

b. Código en MySQL

Con lo descrito anteriormente tenemos el siguiente código:

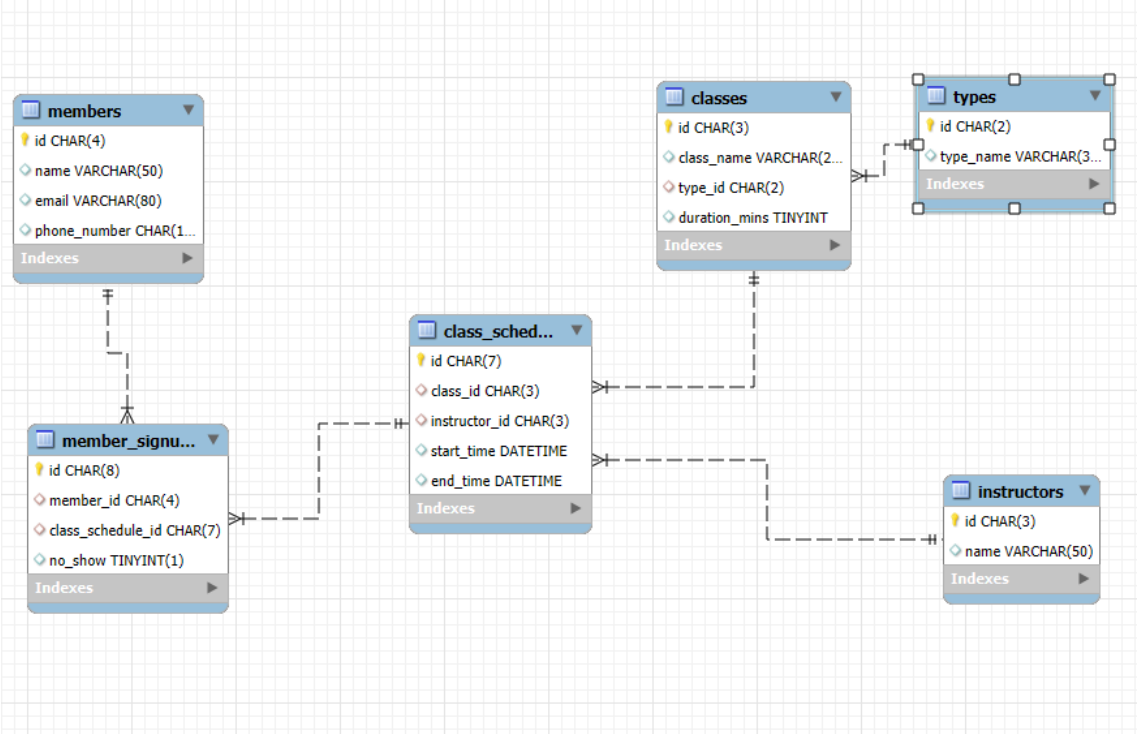
```sql
DROP DATABASE IF EXISTS ReplicaPractica;
CREATE DATABASE IF NOT EXISTS ReplicaPractica;
USE ReplicaPractica;

CREATE TABLE IF NOT EXISTS members(
    id CHAR(4),
    name VARCHAR(50),
    email VARCHAR(80),
    phone_number CHAR(12),
    CONSTRAINT id_member PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS types(
    id CHAR(2),
    type_name VARCHAR(30),
    CONSTRAINT id_type PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS instructors(
    id CHAR(3),
    name VARCHAR(50),
    CONSTRAINT id_instructors PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS classes(
    id CHAR(3),
    class_name VARCHAR(25),
    type_id CHAR(2),
    duration_mins TINYINT,
    CONSTRAINT id_classes PRIMARY KEY (id),
    CONSTRAINT type_id_fk_classes FOREIGN KEY (type_id) REFERENCES types(id)
);
CREATE TABLE IF NOT EXISTS class_schedule(
    id CHAR(7),
    class_id CHAR(3),
    instructor_id CHAR(3),
    start_time DATETIME,
    end_time DATETIME,
    CONSTRAINT id_class_schedule PRIMARY KEY (id),
    CONSTRAINT class_id_fk_class_schedule FOREIGN KEY (class_id) REFERENCES classes(id),
    CONSTRAINT instructor_id_fk_class_schedule FOREIGN KEY (instructor_id) REFERENCES instructors(id)
);
CREATE TABLE IF NOT EXISTS member_signups(
    id CHAR(8),
    member_id CHAR(4),
    class_schedule_id CHAR(7),
    no_show BOOLEAN,
    CONSTRAINT id_member_signups PRIMARY KEY (id),
    CONSTRAINT member_id_fk_member_signups FOREIGN KEY (member_id) REFERENCES members(id),
    CONSTRAINT class_schedule_id_fk_member_signups FOREIGN KEY (class_schedule_id) REFERENCES
class_schedule(id)
);
```

Y tenemos el siguiente output después de usarlo:

#	Time	Action	Message
4	10:39:58	CREATE TABLE IF NOT EXISTS members(id CHAR(4), name VARCHAR(25), email VARCHAR(80), phone_number CHAR(12), CONSTRAIN...	0 row(s) affected
5	10:39:58	CREATE TABLE IF NOT EXISTS types(id CHAR(2), type_name VARCHAR(30), CONSTRAINT id_type PRIMARY KEY (id))	0 row(s) affected
6	10:39:58	CREATE TABLE IF NOT EXISTS instructors(id CHAR(3), name VARCHAR(30), CONSTRAINT id_instructors PRIMARY KEY (id))	0 row(s) affected
7	10:39:58	CREATE TABLE IF NOT EXISTS classes(id CHAR(3), class_name VARCHAR(25), type_id CHAR(2), duration_mins TINYINT, CONSTRAINT ...	0 row(s) affected
8	10:39:58	CREATE TABLE IF NOT EXISTS class_schedule(id CHAR(7), class_id CHAR(3), instructor_id CHAR(3), start_time DATETIME, end_time DA...	0 row(s) affected
9	10:39:58	CREATE TABLE IF NOT EXISTS member_signups(id CHAR(8), member_id CHAR(4), class_schedule_id CHAR(7), no_show BOOLEAN, CON...	0 row(s) affected

c. Ingeniería Reversa del Código

Como método de valoración y visualización, podemos ver cómo queda tras la ingeniería reversa el código:



Efectivamente, es prácticamente calcada. Por lo que se puede valorar que efectivamente esté bien ejecutado.

d. Actualización del Código

Se nos pide ahora añadir a instructor, una nueva tabla para el apellido, surname, y para la tabla de members se desean borrar todos los números de teléfono.

A la hora de plantear esta solución, el primer pensamiento que se podría pasar por la cabeza es el de editar el código y rehacerlo desde cero, pero esto va a dar problemas a futuro, ya que significaría que todos los datos previamente subidos se tendrían que reactualizar.

Por esto, el ejercicio indica que se tendrá que usar ALTER. Para esto, se tendrá que añadir solo líneas sueltas que serán ejecutadas individualmente con el símbolo del rayo.

Otro dato a cambiar, va a ser el tamaño del nombre. Será reducido de 50 a 30 debido a que ahora existe surname.

e. Modificaciones con ALTER

Tras estas actualizaciones, se necesitan solo ejecutar estas. Enseñaremos las modificaciones particulares:

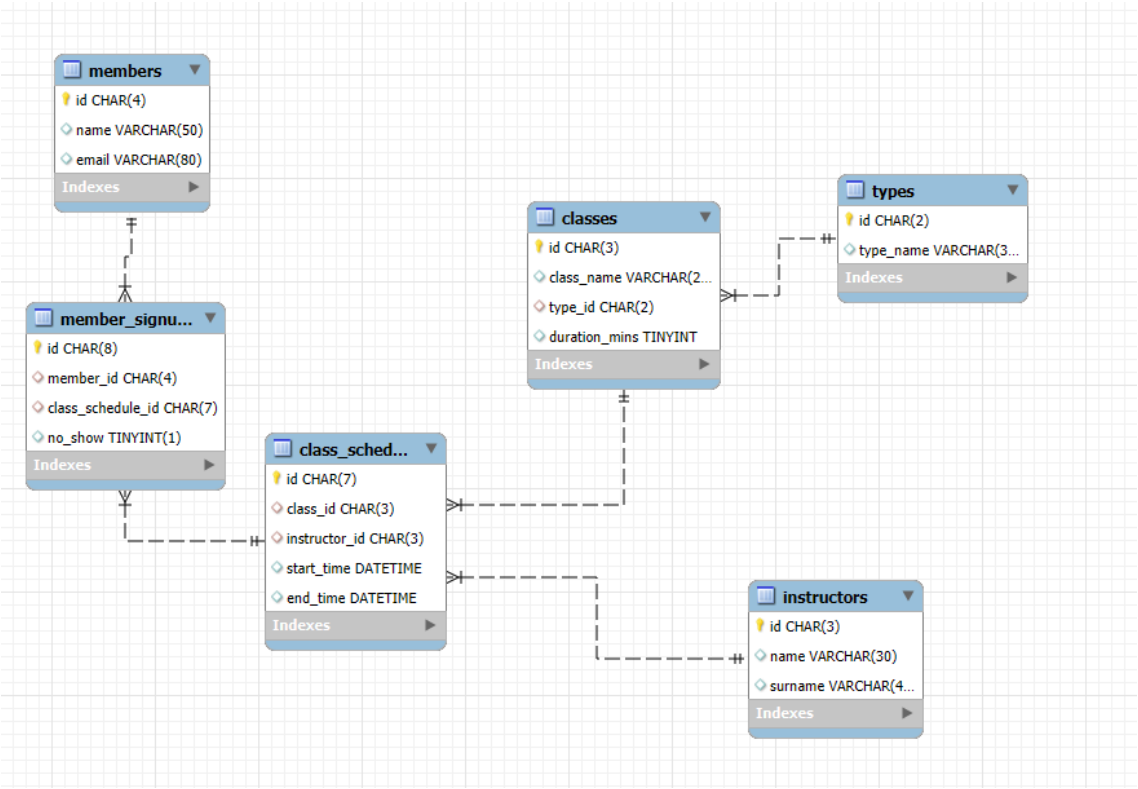
```
49 •  ALTER TABLE instructors ADD COLUMN surname VARCHAR(40);
50 •  ALTER TABLE instructors MODIFY COLUMN name VARCHAR(30);
51 •  ALTER TABLE members DROP COLUMN phone_number;
```

Y todo el código:

```
1 DROP DATABASE IF EXISTS ReplicaPractica;
2 CREATE DATABASE IF NOT EXISTS ReplicaPractica;
3 USE ReplicaPractica;
4
5 CREATE TABLE IF NOT EXISTS members(
6     id CHAR(4),
7     name VARCHAR(50),
8     email VARCHAR(80),
9     phone_number CHAR(12),
10    CONSTRAINT id_member PRIMARY KEY (id)
11 );
12 CREATE TABLE IF NOT EXISTS types(
13    id CHAR(2),
14    type_name VARCHAR(30),
15    CONSTRAINT id_type PRIMARY KEY (id)
16 );
17 CREATE TABLE IF NOT EXISTS instructors(
18    id CHAR(3),
19    name VARCHAR(50),
20    CONSTRAINT id_instructors PRIMARY KEY (id)
21 );
22 CREATE TABLE IF NOT EXISTS classes(
23    id CHAR(3),
24    class_name VARCHAR(25),
25    type_id CHAR(2),
26    duration_mins TINYINT,
27    CONSTRAINT id_classes PRIMARY KEY (id),
28    CONSTRAINT type_id_fk_classes FOREIGN KEY (type_id) REFERENCES types(id)
29 );
30 CREATE TABLE IF NOT EXISTS class_schedule(
31    id CHAR(7),
32    class_id CHAR(3),
33    instructor_id CHAR(3),
34    start_time DATETIME,
35    end_time DATETIME,
36    CONSTRAINT id_class_schedule PRIMARY KEY (id),
37    CONSTRAINT class_id_fk_class_schedule FOREIGN KEY (class_id) REFERENCES classes(id),
38    CONSTRAINT instructor_id_fk_class_schedule FOREIGN KEY (instructor_id) REFERENCES instructors(id)
39 );
40 CREATE TABLE IF NOT EXISTS member_signups(
41    id CHAR(8),
42    member_id CHAR(4),
43    class_schedule_id CHAR(7),
44    no_show BOOLEAN,
45    CONSTRAINT id_member_signups PRIMARY KEY (id),
46    CONSTRAINT member_id_fk_member_signups FOREIGN KEY (member_id) REFERENCES members(id),
47    CONSTRAINT class_schedule_id_fk_member_signups FOREIGN KEY (class_schedule_id) REFERENCES
   class_schedule(id)
48 );
49 ALTER TABLE instructors ADD COLUMN surname VARCHAR(40);
50 ALTER TABLE instructors MODIFY COLUMN name VARCHAR(30);
51 ALTER TABLE members DROP COLUMN phone_number;
```

f. Ingeniería reversa y revisión final

Veamos, pues, cómo queda tras la ingeniería reversa de MySQL:



Como se puede ver, esto ha sido un éxito.

Bibliografía

- Recurso del curso de Desarrollo de Aplicaciones Web, Los siguientes documentos.
- https://auladecroly.com/pluginfile.php/49482/mod_resource/content/4/2.3%20Creaci%C3%B3n%20de%20bases%20de%20datos%20II.pdf
- https://auladecroly.com/pluginfile.php/47045/mod_resource/content/4/2.1%20Introducci%C3%B3n%20y%20tipos%20de%20datos.pdf