

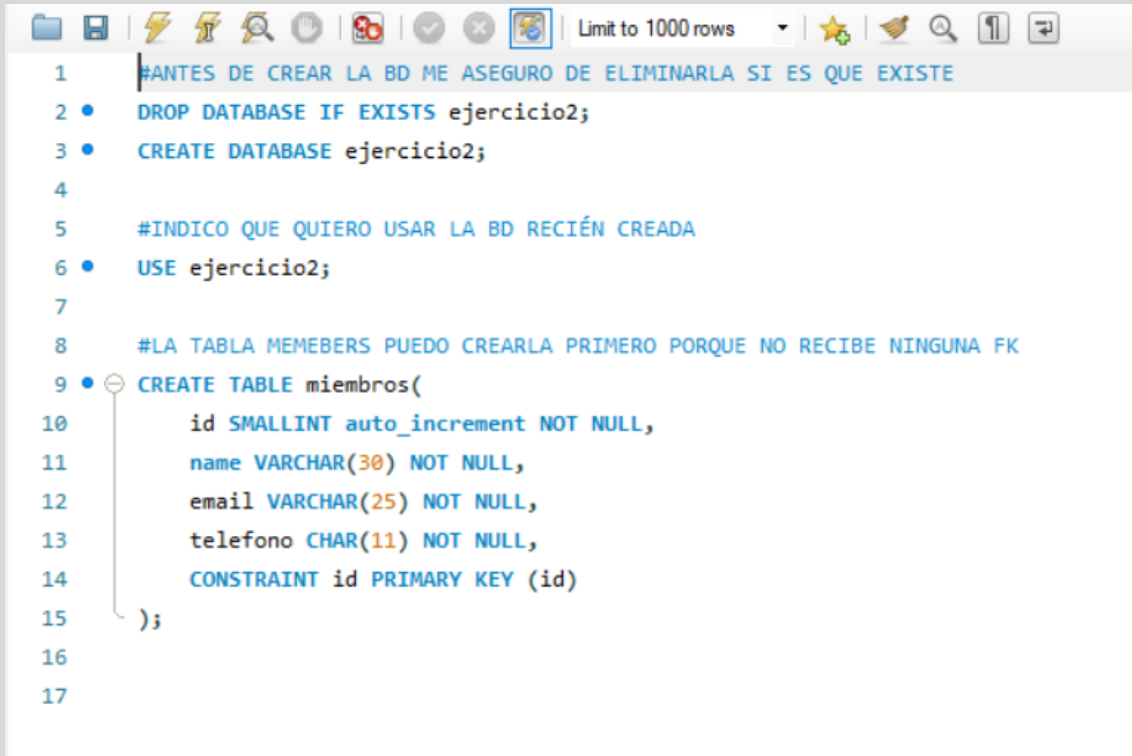
PARTIENDO DEL MODELO RELACIONAL

Trabajo de Alejandro Sainz Sainz

bd-aCTIVIDAD
2.2

PRIMEROS PASOS.....	2
TABLAS DEPENDIENTES	5
DIAGRAMA FINAL	10
MODIFICACIONES	11
CODIGO SQL	13

PRIMEROS PASOS



```
1 #ANTES DE CREAR LA BD ME ASEGURO DE ELIMINARLA SI ES QUE EXISTE
2 • DROP DATABASE IF EXISTS ejercicio2;
3 • CREATE DATABASE ejercicio2;
4
5 #INDICO QUE QUIERO USAR LA BD RECIÉN CREADA
6 • USE ejercicio2;
7
8 #LA TABLA MEMEBERS PUEDO CREARLA PRIMERO PORQUE NO RECIBE NINGUNA FK
9 • CREATE TABLE miembros(
10     id SMALLINT auto_increment NOT NULL,
11     name VARCHAR(30) NOT NULL,
12     email VARCHAR(25) NOT NULL,
13     telefono CHAR(11) NOT NULL,
14     CONSTRAINT id PRIMARY KEY (id)
15 );
16
17
```

Nada más empezar, y como se sugirió en clase lo que hago es preparar la creación de la BD. En este caso me aseguro, con el primero comando, que en caso de existir la elimine, para poder empezar de nuevo.

El comando en cuestión es `DROP DATABASE IF EXIST nombreBD`.

Acto seguido, introduzco un comando para crear la BD, e indico que es esta base de datos la que voy a utilizar.

Una vez insertados esos dos comandos, analizando el esquema que proporciona el ejercicio, comienzo a crear las tablas que creo que se pueden crear desde el inicio, ya que no contienen ninguna dependencia de otras tablas, en este caso la tabla miembros, es una de las que puedo crear.

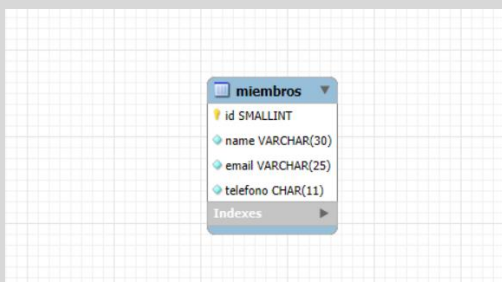
```
#LA TABLA MEMEBERS PUEDO CREARLA PRIMERO PORQUE NO RECIBE NINGUNA FK
CREATE TABLE miembros(
  id SMALLINT auto_increment NOT NULL,
  name VARCHAR(30) NOT NULL,
  email VARCHAR(25) NOT NULL,
  telefono CHAR(11) NOT NULL,
  CONSTRAINT id PRIMARY KEY (id)
);
```

Como se ve en esta imagen, voy añadiendo los campos como indica el ejercicio, con su tipo de dato correspondiente y su longitud, en caso de ser necesario indicarla. Añado NOT NULL a todos los campos pues considero que es necesario que sea obligatorio que tengan contenido.

Por último, en la línea final, indico cual es la PK de esta tabla, en este caso, id.

#	Time	Action
1	17:27:54	Could not connect, server may not be running.
2	18:04:28	DROP DATABASE IF EXISTS ejercicio2
3	18:04:28	CREATE DATABASE ejercicio2
4	18:04:28	USE ejercicio2
5	18:04:28	CREATE TABLE miembros(id SMALLINT auto_increment NOT NULL, name VARCHAR(30) NOT NULL, email VARCHAR(25) NOT NULL, telefono CHAR(11) NO...

Para asegurarme de que las cosas van bien hasta ahora, ejecuto el script para asegurarme de que no hay errores.



Como última comprobación, me aseguro de que se genera la parte del diagrama correspondiente.

```
#SIGO CREANDO OTRAS TABLAS, AQUELLAS QUE NO TIENEN DEPENDENCIAS  
CREATE TABLE tipos(  
    id smallint auto_increment NOT NULL,  
    type_name varchar(15),  
    CONSTRAINT id primary key (id)  
);
```

Creo la siguiente tabla que no tiene ninguna dependencia, en este caso la tabla tipos. Esta es una tabla mucho más sencilla, con solo dos campos, que se crea rápidamente.

```
#CREACION DE LA TABLA MONITORES  
create table monitores(  
    id smallint auto_increment not null,  
    name varchar (20),  
    CONSTRAINT id primary key (id)  
);
```

Por último, creo la tabla monitores, que en el caso del ejemplo se llama instructores. Defino sus dos campos y el tipo de dato que creo que es más adecuado.

TABLAS DEPENDIENTES

A partir de ahora, hay que crear las tablas que reciben PK de otras tablas como FK. Estas se crean a partir de este momento, pues en un inicio, si no se han creado las tablas de las que se reciben las PK como FK el programa lo reconoce como un error.

```
# EMPIEZO A CREAR LAS TABLAS DEPENDIENTES
# LA PRIMERA DE ELLAS ES CLASES

create table clases(
    id smallint auto_increment not null,
    class_name VARCHAR(20),
    duration_mins smallint,
    type_id SMALLINT,
    CONSTRAINT id primary key (id),
    CONSTRAINT type_id foreign key (type_id) REFERENCES tipos (id)
);
```

La primera tabla con dependencias que voy a crear es clases, que depende en alguna medida de la tabla tipos.

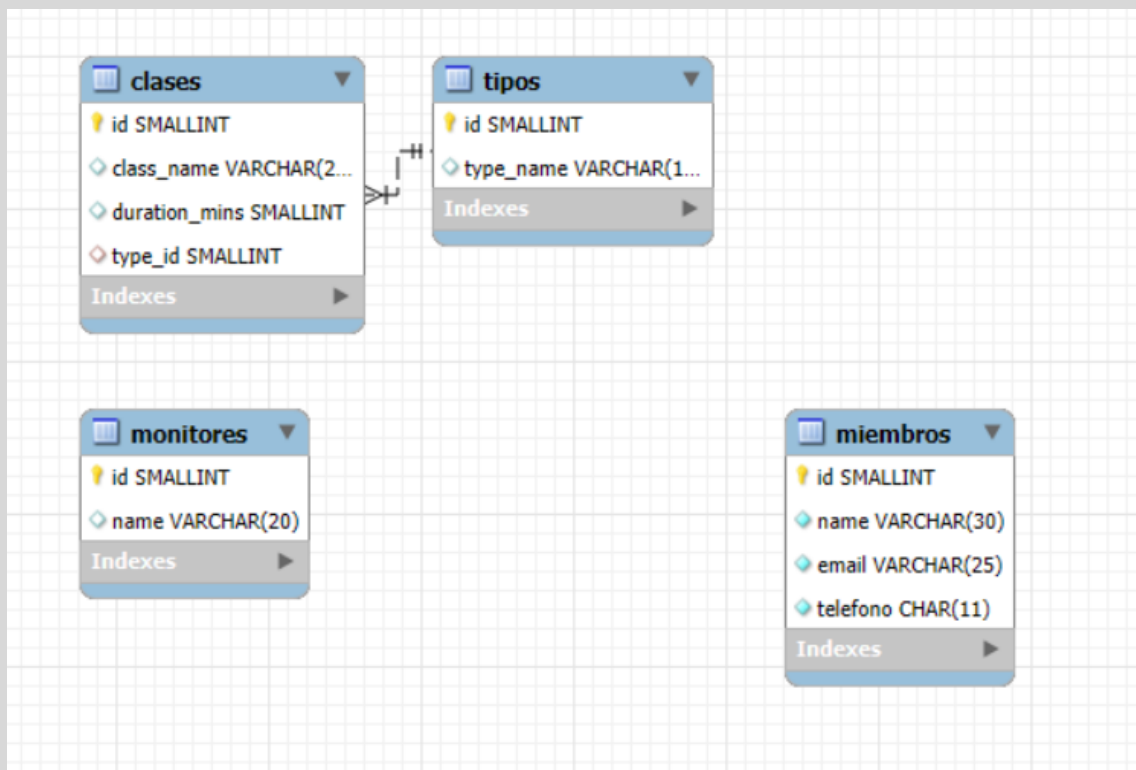
Voy añadiendo sus atributos, definiendo también el tipo de dato que creo que más le conviene. En las dos últimas líneas indico cual es la clave primaria y cual es la clave foránea. En este caso creo la FK, llamándola type_id, que coincide con el atributo dentro de la tabla, y que hace referencia a la tabla tipos y a su PK id.

```

7 18:30:12 CREATE DATABASE ejercicio2
8 18:30:12 USE ejercicio2
9 18:30:12 CREATE TABLE miembros(id SMALLINT auto_increment NOT NULL, name VARCHAR(30) NOT NULL, email VARCHAR(25) NOT NULL, telefono CHAR(11) N...
10 18:30:12 CREATE TABLE tipos(id smallint auto_increment NOT NULL, type_name varchar(15), CONSTRAINT id primary key (id))
11 18:30:12 create table monitores(id smallint auto_increment not null, name varchar (20), CONSTRAINT id primary key (id) )
12 18:30:12 create table clases(id smallint auto_increment not null, class_name VARCHAR(20), duration_mins smallint, type_id SMALLINT, CONSTRAINT id primary key...

```

De nuevo vuelvo a hacer una prueba intermedia, para asegurarme de que hasta este momento todo va bien.



También vuelvo a hacer la ingeniería inversa para comprobar como va quedando el diagrama.

Al añadir una nueva tabla, con una dependencia, el diagrama también lo va mostrando.

```
• create table horarios_clases(  
    id smallint auto_increment not null,  
    start_time datetime,  
    end_time datetime,  
    class_id smallint,  
    instructor_id smallint,  
    CONSTRAINT id primary key (id),  
    CONSTRAINT class_id foreign key (class_id) REFERENCES clases (id),  
    CONSTRAINT instructor_id foreign key (instructor_id) REFERENCES monitores (id)  
);
```

Esta nueva tabla se complica un poco más. En este caso es la tabla de horarios de las clases. Esta tabla tiene una id, que quizá no sea muy correcto que sea incremental, pero yo las claves primarias las considero mejor así, para que siempre sean únicas.

Le siguen un par de campos de tipo fecha y hora, que son las horas de inicio y de fin de las clases, seguidas de dos campos que son FK que recibe esta tabla.

Después de identificar en el script a id como PK, indico las dos claves foráneas y las tablas y atributos a los que referencian.

Alejandro Sainz Sainz

Una vez creada la tabla horarios, puedo pasar a crear la última tabla del modelo, ya que recibe una FK de la tabla horarios y otra de la primera tabla creada, miembros.

```
#CREO LA ÚLTIMA TABLA, EN ESTE CASO LA VOY A LLAMAR INSCRIPCIONES MIEMBROS
create table inscripciones_miembros(
  id smallint auto_increment not null,
  no_show boolean,
  member_id smallint not null,
  class_schedule_id smallint not null,
  CONSTRAINT id primary key (id),
  CONSTRAINT member_id foreign key (member_id) REFERENCES miembros (id),
  CONSTRAINT class_schedule_id foreign key (class_schedule_id) REFERENCES horarios_clases (id)
);
```

En la imagen vemos el código que he añadido en el script para crear la última tabla. De nuevo una PK que se llama id, un campo llamado no_show, que supongo que se refiere a no asistencia, por lo tanto, será de tipo boolean.

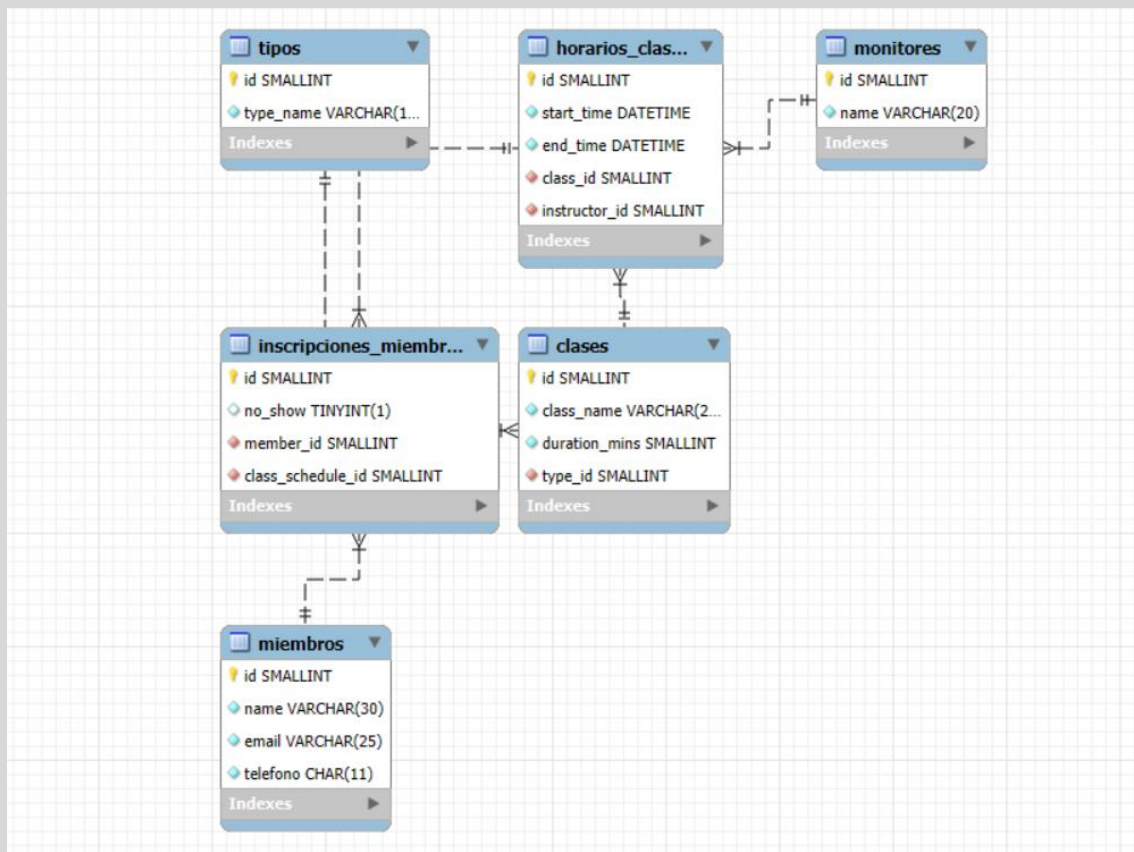
En las últimas líneas indico cuál es la Primary key y cuales son las Foreign Keys, una de ellas hace referencia al campo id de la tabla miembros, mientras la otra lo hace al campo id de la tabla horarios_clases.

Como último paso después de la creación de esta tabla, voy a ejecutar el script para que cree todas las tablas y después generaré el diagrama mediante el proceso de ingeniería inversa.

13	18:54:30	DROP DATABASE IF EXISTS ejercicio2	4 row(s) affected
14	18:54:30	CREATE DATABASE ejercicio2	1 row(s) affected
15	18:54:30	USE ejercicio2	0 row(s) affected
16	18:54:30	CREATE TABLE miembros(id SMALLINT auto_increment NOT NULL, name VARCHAR(30) NOT NULL, email VARCHAR(25) NOT NULL, telefono CHAR(11) N...	0 row(s) affected
17	18:54:30	CREATE TABLE tipos(id smallint auto_increment NOT NULL, type_name varchar(15) not null, CONSTRAINT id primary key (id))	0 row(s) affected
18	18:54:30	create table monitores(id smallint auto_increment not null, name varchar(20) not null, CONSTRAINT id primary key (id))	0 row(s) affected
19	18:54:30	create table clases(id smallint auto_increment not null, class_name VARCHAR(20) not null, duration_mins smallint not null, type_id SMALLINT not null, CON...	0 row(s) affected
20	18:54:30	create table horarios_clases(id smallint auto_increment not null, start_time datetime not null, end_time datetime not null, class_id smallint not null, instructor_i...	0 row(s) affected
21	18:54:30	create table inscripciones_miembros(id smallint auto_increment not null, no_show boolean, member_id smallint not null, class_schedule_id smallint not null, C...	0 row(s) affected

Al ejecutar el script, si hacemos captura de pantalla, estos serían todos los pasos que se ejecutan. Primero borra la BD en caso de que exista. Luego la crea, luego indica que va a usar esa BD y, ya por último, crea todas las tablas en el orden indicado en el script.

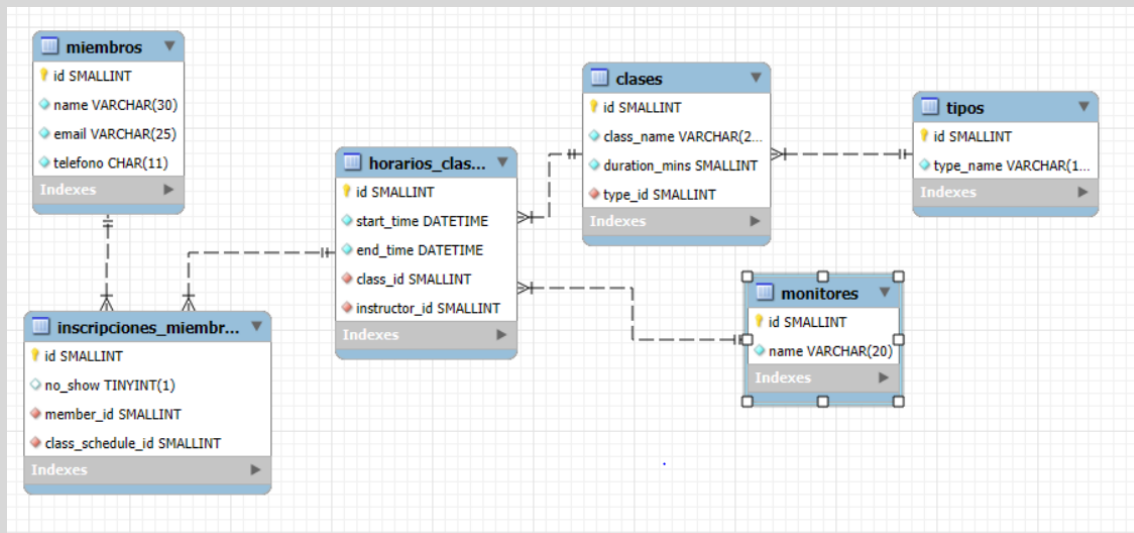
Ahora después de esto generamos el diagrama:



Alejandro Sainz Sainz

Este es el diagrama que me queda. Voy a intentar recolocarlo para que quede como en el enunciado del ejercicio.

DIAGRAMA FINAL



Una vez colocadas las tablas en el diagrama este es el resultado, Bastante parecido al del enunciado.

Cuando terminé me di cuenta de que no indiqué NOT NULL en muchos de los atributos, cosa que considero que tengo que hacer, así que lo fui añadiendo después. Lo que tampoco indiqué, y que también tendría que haber hecho, es añadir UNIQUE a los campos que son PK, pero en este caso no los he añadido, quizá porque al ser un ejemplo no lo consideré oportuno.

MODIFICACIONES

Según el enunciado debo de realizar dos modificaciones:

La primera es eliminar el campo teléfono de miembros:

- `ALTER TABLE miembros DROP COLUMN telefono;`

Este sería el comando introducido en el script.

La segunda modificación es añadir el campo surname a instructores (en mi caso monitores):

```
ALTER TABLE monitores ADD surname VARCHAR(20);
```

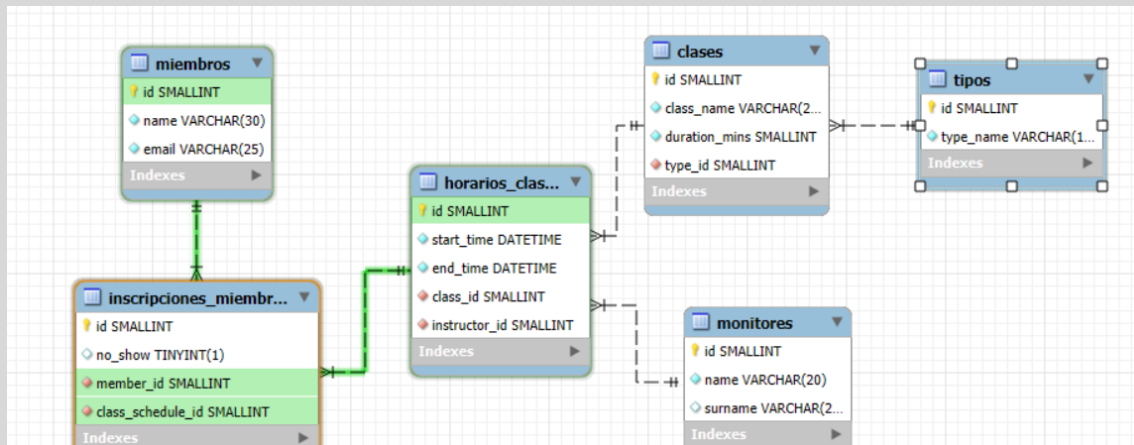
Y este es el comando que utilizo para realizarlo.

Vamos a ejecutar de nuevo el script:

✓	31	19:07:53	ALTER TABLE miembros DROP COLUMN telefono
✓	32	19:07:53	ALTER TABLE monitores ADD sumame VARCHAR(20)

En principio, al ejecutar el script, todo en verde y OK.

Vamos a obtener el diagrama y vemos si se ha modificado:



Como se puede observar, el campo teléfono ha desaparecido, y el campo surname se ha añadido a la tabla monitores.

CODIGO SQL

En este apartado añado el código que he usado en la aplicación para crear el modelo que se pedía.

#ANTES DE CREAR LA BD ME ASEGURO DE ELIMINARLA SI ES QUE EXISTE

DROP DATABASE IF EXISTS ejercicio2;

CREATE DATABASE ejercicio2;

#INDICO QUE QUIERO USAR LA BD RECÍÉN CREADA

USE ejercicio2;

#LA TABLA MEMEBERS PUEDO CREARLA PRIMERO PORQUE NO RECIBE NINGUNA FK

```
CREATE TABLE miembros(  
  id SMALLINT auto_increment NOT NULL,  
  name VARCHAR(30) NOT NULL,  
  email VARCHAR(25) NOT NULL,  
  telefono CHAR(11) NOT NULL,  
  CONSTRAINT id PRIMARY KEY (id)  
);
```

#SIGO CREANDO OTRAS TABLAS, AQUELLAS QUE NO TIENEN DEPENDENCIAS

```
CREATE TABLE tipos(  
  id smallint auto_increment NOT NULL,  
  type_name varchar(15) not null,  
  CONSTRAINT id primary key (id)  
);
```

Alejandro Sainz Sainz

#CREACION DE LA TABLA MONITORES

```
create table monitores(  
  id smallint auto_increment not null,  
  name varchar (20) not null,  
  CONSTRAINT id primary key (id)  
);
```

EMPIEZO A CREAR LAS TABLAS DEPENDIENTES

LA PRIMERA DE ELLAS ES CLASES

```
create table clases(  
  id smallint auto_increment not null,  
  class_name VARCHAR(20) not null,  
  duration_mins smallint not null,  
  type_id SMALLINT not null,  
  CONSTRAINT id primary key (id),  
  CONSTRAINT type_id foreign key (type_id) REFERENCES tipos (id)  
);
```

#VOY A AÑADIR OTRA TABLA, EN ESTE CASO LA MÁS COMPLICADA, QUE SON LOS

#HORARIOS DE CLASES

```
create table horarios_clases(  
  id smallint auto_increment not null,  
  start_time datetime not null,  
  end_time datetime not null,  
  class_id smallint not null,  
  instructor_id smallint not null,  
  CONSTRAINT id primary key (id),  
  CONSTRAINT class_id foreign key (class_id) REFERENCES clases (id),  
  CONSTRAINT instructor_id foreign key (instructor_id) REFERENCES  
monitores (id)  
);
```

#CREO LA ÚLTIMA TABLA, EN ESTE CASO LA VOY A LLAMAR INSCRIPCIONES MIEMBROS

```
create table inscripciones_miembros(  
  id smallint auto_increment not null,  
  no_show boolean,  
  member_id smallint not null,  
  class_schedule_id smallint not null,  
  CONSTRAINT id primary key (id),  
  CONSTRAINT member_id foreign key (member_id) REFERENCES miembros  
(id),  
  CONSTRAINT class_schedule_id foreign key (class_schedule_id)  
REFERENCES horarios_clases (id)  
);
```

ALTER TABLE miembros DROP COLUMN telefono;

ALTER TABLE monitores ADD surname VARCHAR(20);