implementando nuestro propio proyecto

Trabajo de Alejandro Sainz Sainz

BD-ACTIVIDAD 2.3

PRIMEROS PASOS 2

TABLAS DEPENDIENTES 5

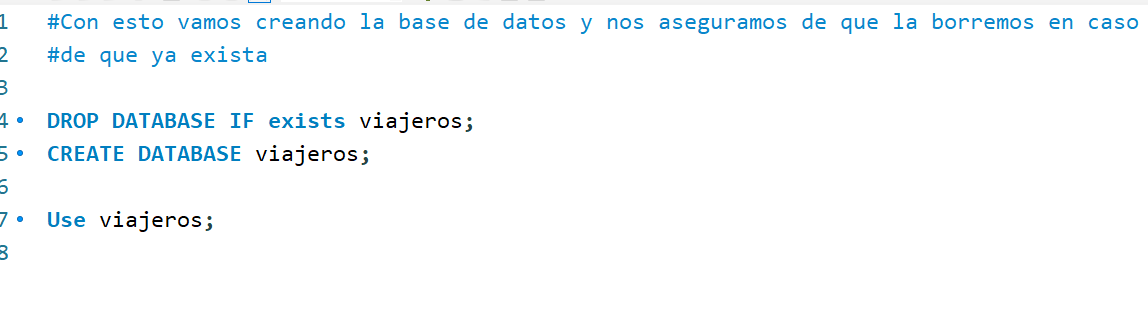
DIAGRAMA FINAL 10

MODIFICACIONES 11

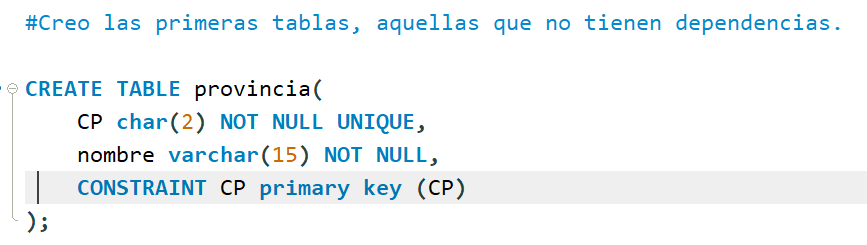
CODIGO SQL 13

# CREANDO EL SCRIP DE NUESTRO PROYECTO

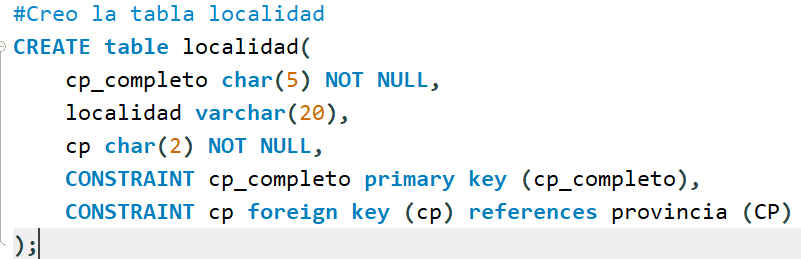
Lo primero es ir creando la BD en el script, e ir fijándonos en cuales son las tablas que no poseen dependencias para ir creándolas primero.



Lo primero que voy a hacer es crear la tabla provincia, que no tiene dependencias.

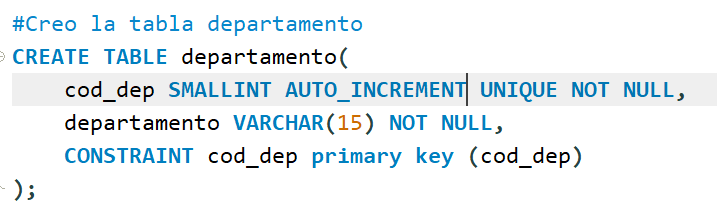


Ahora puedo crear la tabla localidad, ya que solo depende de la tabla provincia, así puedo dejar esa parte cerrada y seguir con otras tablas que no tengan dependencia.

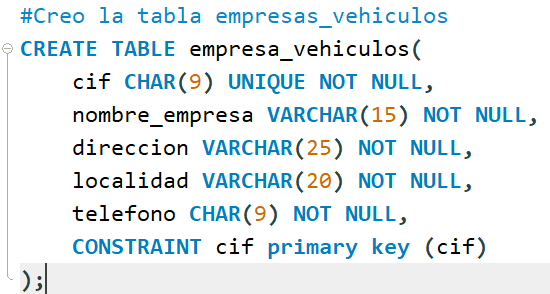


Al tener completadas estas dos, como luego necesito localidad para otras dos tablas como FK, ya puedo despreocuparme por ellas.

Voy a seguir con otras dos tablas independientes, que son la tabla departamentos y la tabla empresa vehículos.

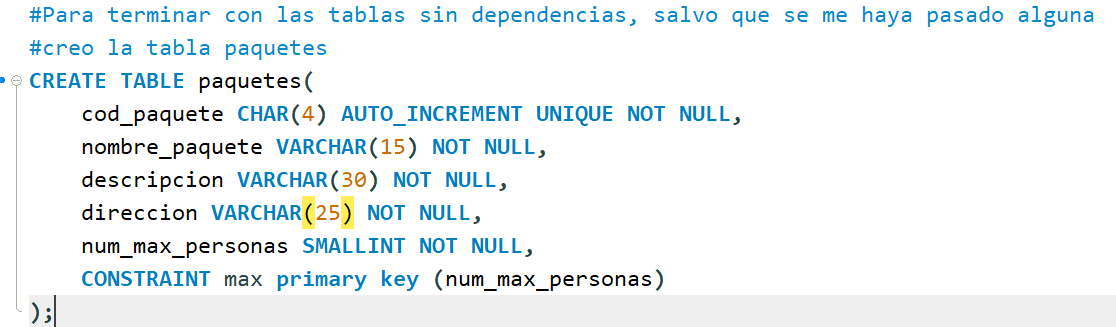


Esta es la parte del script correspondiente a la tabla departamento.



Con esta parte del script ya tengo terminada la tabla de empresas de vehículos de alquiler.

Ahora, si no me equivoco, creo que la única tabla que me quede de añadir que no tiene dependencias es la tabla paquetes, aquella que recoge los paquetes y su descripción.

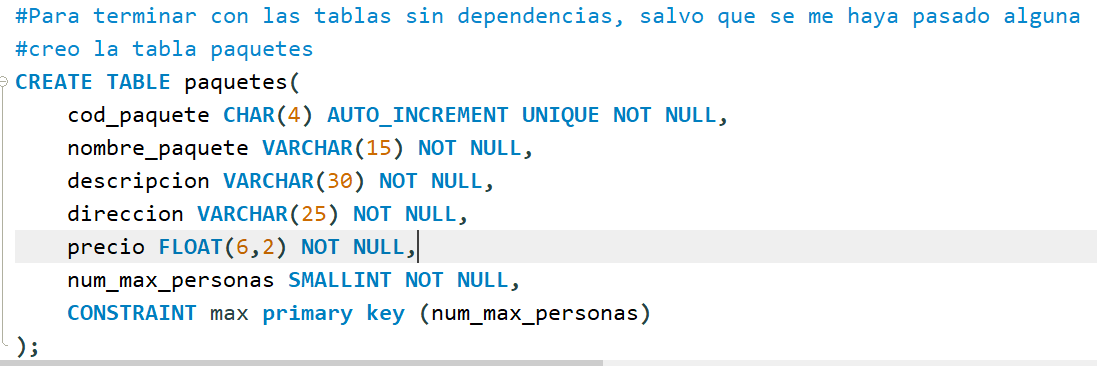


Pues esto sería la tabla paquetes. Intento siempre ajustar los valores de los atributos a lo que yo creo que sería realista, por ejemplo, cod\_paquete CHAR(4), ya que no creo que el código de paquete ocupe más de cuatro caracteres.

Intento que el atributo dirección sea de la misma longitud en todas las tablas que lo incluyen, por coherencia más que nada.

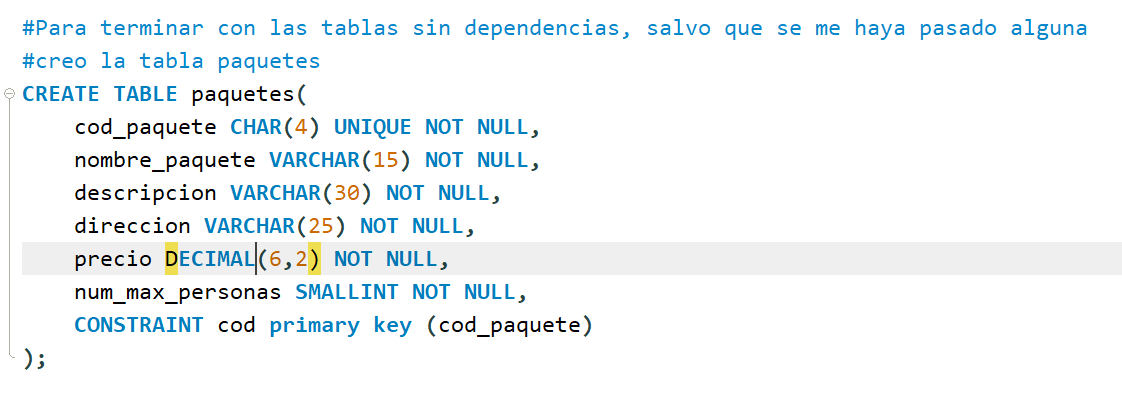
En el caso de num\_max\_personas he decidido que sea SMALLINT, simplemente por una cosa. Imaginemos que la empresa escala o necesita otra serie de cosas, como calcular media de clientes por actividad, etc, como incluir el precio de un paquete por persona, etc.

Ahora que lo he comentado, voy a añadir el precio de un paquete, completo, sin contar por grupo o con descuentos, que creo que así completo un poco al modelo.



Lo añado el campo precio como ya dije antes.

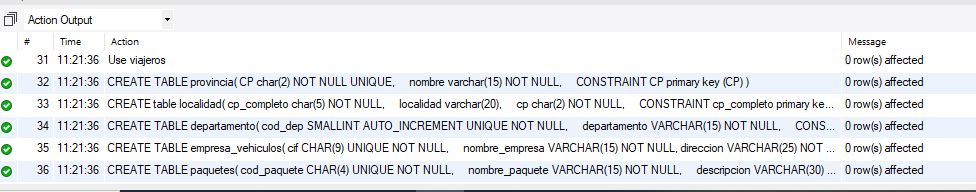
Llegado a este punto voy a ejecutar el script simplemente para ver que no da errores.



Gracias a hacer las pruebas de ejecutar el script como comprobación me di cuenta de un par de errores.

1. Puse un campo char como auto increment, cosas bastante improbable.
2. Me indicaba que marcar un campo como float era un practica deprecated y como indicó el profesor en clase lo cambie por decimal.

Una vez realizados los cambios el script quedó así y esto es lo que recoge la ejecución del scritp.



Vemos que ya no da ningún error o warning, así que ahora vamos a realizar un

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

Me acabo de fijar que la tabla alojamientos no depende de ninguna otra, según el modelo que yo generé, así que voy a crearla ahora.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Por último, creo la tabla monitores, que en el caso del ejemplo se llama instructors. 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.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

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

Diagrama

Descripción generada automáticamente

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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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 para 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.

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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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:

Diagrama

Descripción generada automáticamente

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

# DIAGRAMA FINAL

Diagrama

Descripción generada automáticamente

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:



Este sería el comando introducido en el script.

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



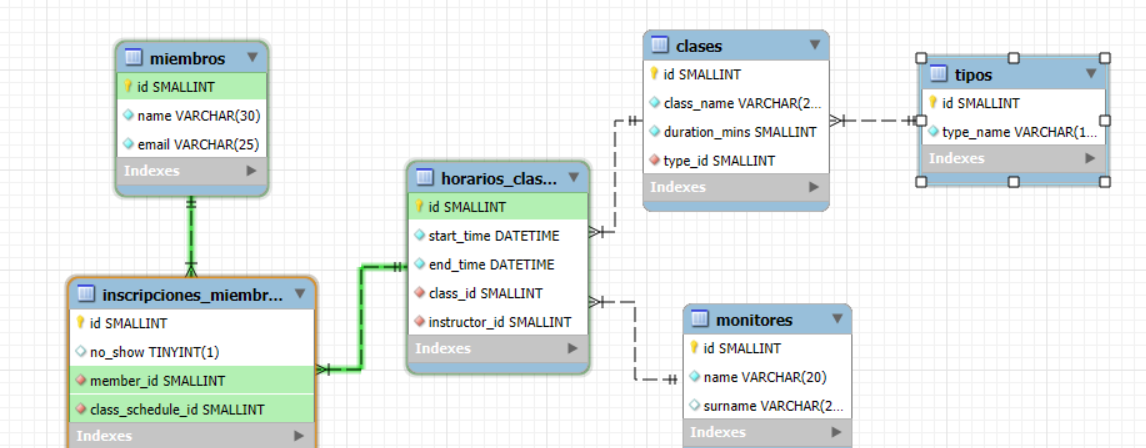
Y este es el comando que utilizo para realizarlo.

Vamos a ejecutar de nuevo el script:



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 RECIÉN 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)**

**);**

**#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);**