implementando nuestro propio proyecto

Trabajo de Alejandro Sainz Sainz

BD-ACTIVIDAD 2.3

PRIMEROS PASOS 2

TABLAS DEPENDIENTES 5

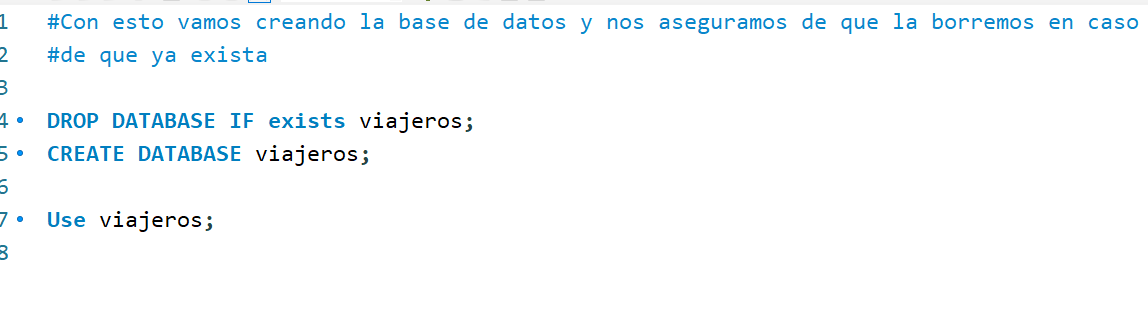
DIAGRAMA FINAL 10

MODIFICACIONES 11

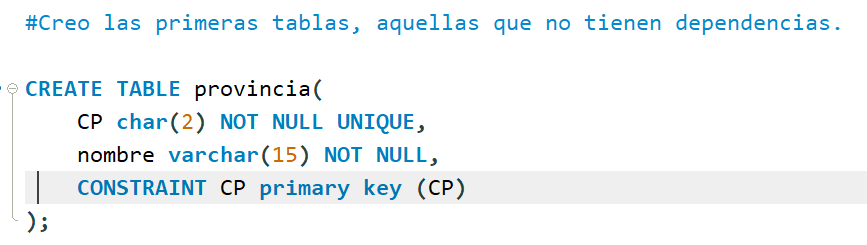
CODIGO SQL 13

# CREANDO EL SCRIP DE NUESTRO PROYECTO

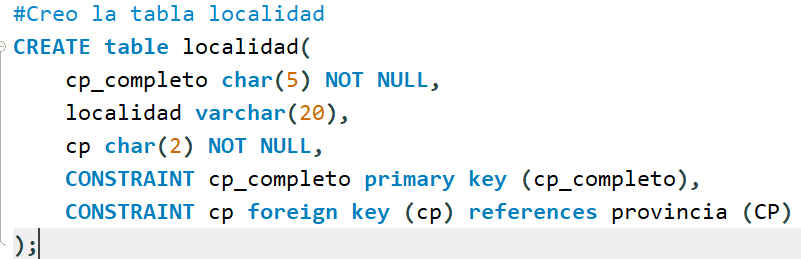
Lo primero es ir creando la BD en el script, e ir fijándonos en cuales son las tablas que no poseen dependencias para ir creándolas primero.



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

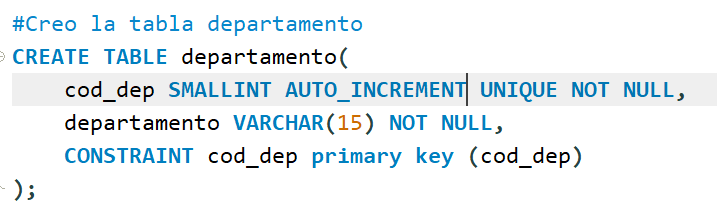


Ahora puedo crear la tabla localidad, ya que solo depende de la tabla provincia, así puedo dejar esa parte cerrada y seguir con otras tablas que no tengan dependencia.

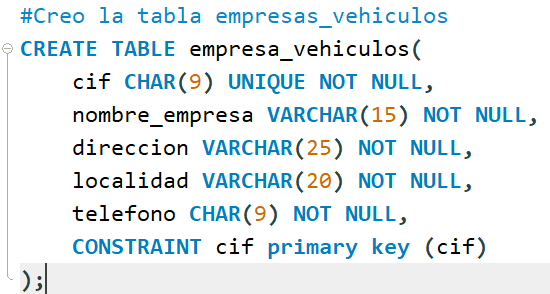


Al tener completadas estas dos, como luego necesito localidad para otras dos tablas como FK, ya puedo despreocuparme por ellas.

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

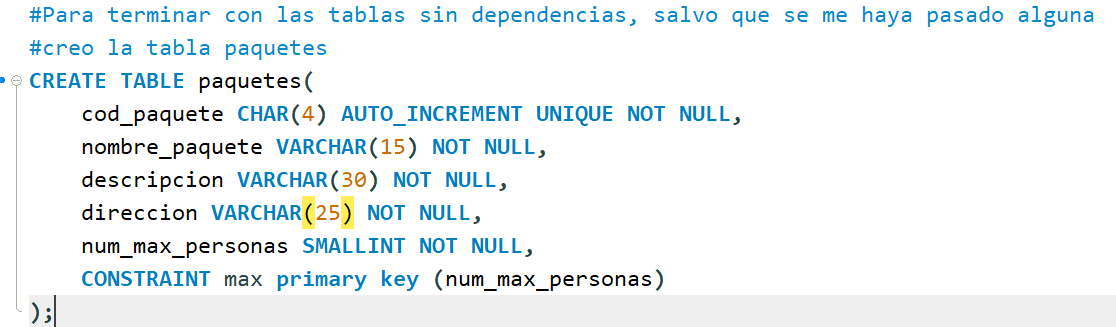


Esta es la parte del script correspondiente a la tabla departamento.



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.

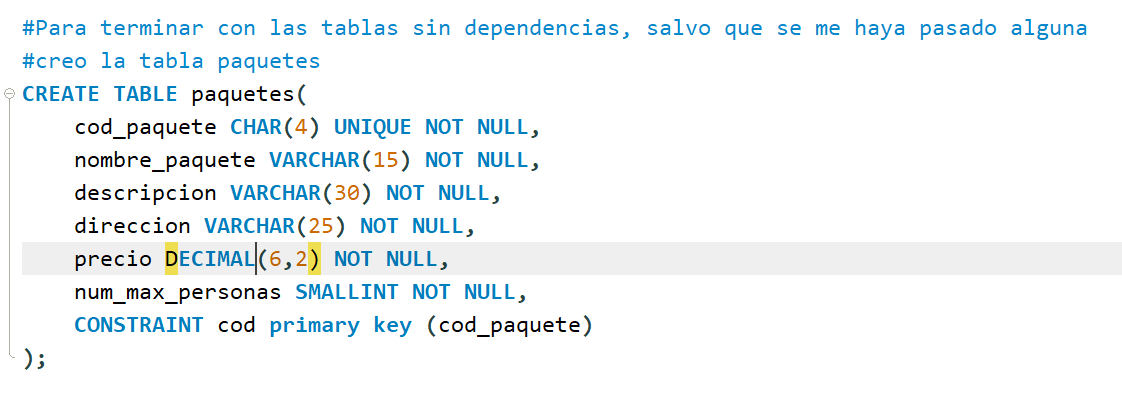
En el caso de num\_max\_personas he decidido que sea SMALLINT, simplemente por una cosa. Imaginemos que la empresa escala o necesita otra serie de cosas, como calcular media de clientes por actividad, etc, como incluir el precio de un paquete por persona, etc.

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.



Le añado el campo precio como ya dije antes.

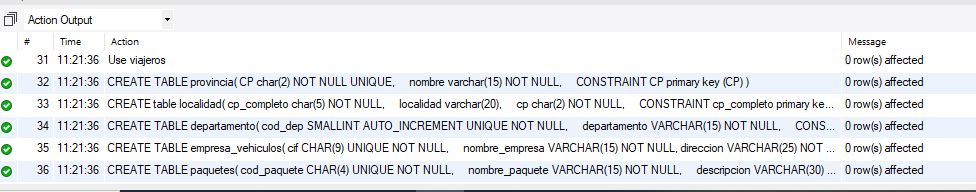
Llegado a este punto voy a ejecutar el script simplemente para ver que no da errores.



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, cosa 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 intento de ingeniería inversa, para ver como va quedando el diagrama, aunque me imagino que por ahora será bastante sencillo.

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

Interfaz de usuario gráfica, Texto, Aplicación

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

Con esto, ahora sí, creo que ya tengo completadas todas las tablas independientes.

Vuelvo a ejecutar el script para comprobar todo de nuevo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ningún warning o error, así que, ahora sí, puedo hacer una prueba para ver cómo va quedando el diagrama.

Diagrama

Descripción generada automáticamente

Por ahora va quedando como esperaba, aunque no era muy difícil, ya que todavía no existe ninguna relación en lo que he creado, salvo en el caso de localidad y provincia.

En este punto, tengo que empezar a crear las tablas que son dependientes de otras, e ir viendo como quedan en comparación con el diagrama que fui desarrollando yo en la realización del ejercicio.

Creo ahora la tabla grupos, que me acabo de dar cuenta en una parte posterior del ejercicio que está sin crear.

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

Descripción generada automáticamente

En este caso, he decidido darle a la clave principal, id\_grupo, un tipo int que se vaya incrementando según vayamos creando los grupos y a num\_personas un varchar(2) ya que no creo que el tamaño de un grupo esté en ningún caso por encima de 99.

Ahora que me doy cuenta, si quiero calcular medias de clientes y grupos, creo que es mejor cambiar el tipo de num\_personas a int, por si en algún caso deseo realizar operaciones con esos datos.

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

Descripción generada automáticamente

Lo dejo así.

# AGREGANDO 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.

Creo que, en este caso, la más importante va a ser la tabla sedes, que es la tabla central de nuestro ejercicio.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Este sería el fragmento de script que corresponde a la tabla sedes.

En un mismo orden de cosas, tengo la tabla clientes, que tiene la misma dependencia, de la tabla localidad, y como ya mostré en el informe anterior, se pude incluir en el mismo ejemplo.

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

Descripción generada automáticamente

Pues aquí tenemos creada la tabla clientes.

## COMPROBACIÓN DE TABLAS Y DIAGRAMA

Para comprobar que estas dos tablas y la relación con la tabla localidad y provincia queda como resolví en el ejercicio anterior voy a ejecutar el script y la ingeniería inversa, y compararé el resultado.

Diagrama

Descripción generada automáticamente

Este diagrama es como quedó resuelto mi ejercicio, vamos a comprobar como queda en el Workbench.

Lo primero, al ejecutar el script, me da un error, hay dos FK con el mismo nombre dentro del modelo, cambio la FK de la tabla clientes por cp\_comp, para que no se repita con la FK de la tabla sedes. Después de eso, ya puedo comprobar el diagrama.

Diagrama

Descripción generada automáticamente

Si lo comparamos con la resolución del ejercicio vemos que quedan iguales.

Ahora prosigamos con los siguientes ejemplos.

## MÁS RELACIONES RESULTAS

Ahora voy a ir resolviendo el caso de los empleados, que creo que lo hice mal en el ejercicio anterior. En el diagrama E/R indiqué una herencia excluyente.

Sin embargo, al crear las tablas en ese mismo ejercicio, creé una tabla departamentos, y luego pasé la clave del departamento como FK a la tabla empleados, así que no sé si está bien resuelto como herencia, aunque como dos tablas independientes que se relacionan creo que si está resuelto de una forma correcta, así que optaré por esta segunda opción.

Diagrama

Descripción generada automáticamente

Esta es la forma en que lo resolví en el ejercicio anterior, y esta es la que voy a seguir al crear la base de datos en Workbench.

La tabla departamentos ya la tengo creada en el apartado anterior, de la misma forma que clientes, así que ahora sólo me queda crear la tabla empleados y relacionarlo todo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteEsta es la creación de la tabla empleados, que es una de las más largas que tenía en el ejercicio. De nuevo voy a ejecutar el script y a ver el diagrama en Workbench.

De nuevo compruebo que al ejecutar el script no hay fallos (no incluyo captura para no alargarlo todo demasiado).

Diagrama

Descripción generada automáticamente

La colocación es diferente a la que creé yo en draw.io pero el resultado es el mismo.

Continuo resolviendo tablas que tienen que ver con sedes, en este caso voy a resolver la relación entre sedes y paquetes. Como la relación era N:N debo de crear la tabla intermedia.

Diagrama

Descripción generada automáticamente

Como se ve aquí, en vez de crear un código para la relación, use las dos FK para crear un PK compuesta. Vamos a resolverlo en el Script.

Diagrama

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamenteEste es la porción de Script. Veamos el resultado.

Las dos entidades quedan relacionadas así. Prosigamos.

Me acabo de dar cuenta de que tenía otra tabla independiente que tenía que crear (esto me pasa por dejarme llevar y hacer un ejercicio muy largo).

Lo que voy a hacer es volver al apartado anterior a crearla y ponerlo allí y después resolver la relación correspondiente.

La tabla en cuestión es la tabla grupos.

Una vez subsanado esto, procedo a crear la tabla intermedia tal y como resolví el ejercicio anterior.

Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Creando nombres distintos para la PK y las FK se genera esta tabla. El script se ejecuta de forma correcta. Vamos a ver el diagrama, que ya se empieza a liar un poco.

Diagrama

Descripción generada automáticamente

Ya tengo la relación resuelta, y como podemos observar, este fragmento del diagrama se parece bastante a la resolución del diagrama que yo creé.

Ahora que ya tenemos resulto el tema de los grupos, podemos pasar a resolver las reservas, que incluye todo esto que he hecho hasta ahora.

Diagrama

Descripción generada automáticamente

Como ya tengo creada la tabla grupos, puedo proceder a crear la tabla reservas. Vamos a ello.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

De esta forma he creado yo la tabla. Fecha\_fin\_reserva permito que sea null por si en la reserva, por el motivo que sea no se especifica fecha de fin. Incluyo el campo precio, un decimal, grupo completo (que sirve para indicar si un grupo completa el número de personas máximas que permite la actividad a realizar) indico que es del tipo boolean y por defecto su valor será false.

Después de esto indico la PK y la FK que forman parte de esta tabla. Al ejecutar el script todo correcto (que no quiere decir que esté bien, pero que se ejecuta sin ningún problema evidente).

Voy a crear ahora la tabla intermedia que usé para terminar de resolver todo este bloque de relaciones.

Esta tabla se llama actividad, ya que sería un listado de las actividades que se llevan a cabo como parte de lo contratado por un grupo en una reserva, indicando los paquetes que están contratados en una reserva en concreta. Vamos a ello.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteAsí es como resuelvo la creación de esta tabla intermedia. La ejecución en principio sin problema, veamos el diagrama para confirmar.

Diagrama

Descripción generada automáticamente

En principio todo correcto, el único problema, es que ya se empieza a hacer complicado que no haya líneas de relación por todas partes. Intento colocarlo todo de una forma en la que se vean de forma clara cuales son las líneas que corresponden a la relación resuelta en cada caso. Lo que me he dado cuenta al revisar, es que el campo boolean me le cambia el propio programa por un tinyint, que ya había visto en los apuntes que se considera lo mismo.

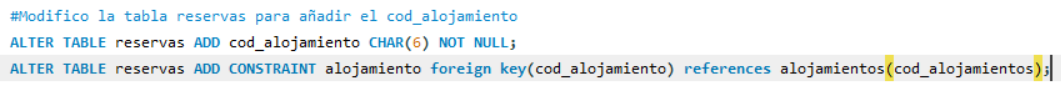
## AVANZANDO CON LA RESOLUCIÓN

Ahora que ya hemos ido resolviendo gran parte de lo que es el núcleo de nuestra BD, vamos a seguir con el resto de las relaciones.

Bien, este caso que viene ahora, lo dejé yo para este momento para probar comandos.

Como sabemos, en mi ejemplo, en la tabla reservas también se indica que alojamiento corresponde a cada reserva. Vamos a incluir eso ahora, y así tengo que usar comandos para modificar la tabla reservas.

Voy a usar el comando para añadir el cod\_alojamiento a la tabla reservas.



Con estos dos comandos, primero añado el nuevo campo a la tabla, después ya le añado a la tabla el CONSTRAINT indicando que ese nuevo campo es la foreign key y a que referencia. No se si se podría hacer en una sola línea, pero creo que así también es válido. Vamos a comprobar ahora el diagrama.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El atributo se ha añadido y se ha generado la relación, así que por ahora todo correcto.

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Vamos a continuar con la parte de los alojamientos. En el ejercicio dije que las sedes se encargan de las reservas de los alojamientos, así que vamos a implementar eso en nuestra BD.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Aquí creo el código para esta tabla intermedia.

Texto

Descripción generada automáticamente con confianza media

Aquí el resultado de la ingeniería.

## ULTIMAS RELACIONES A RESOLVER

Llegados a este punto, ya casi he resulto todas las relaciones y las he añadido al script, que no quiere decir que las que queden sean más fáciles o ligeras.

Así que vamos con ellas.

Había hablado del equipamiento que podía pertenecer a las sedes, y que luego este equipamiento era revisado por nuestros técnicos, así que vamos a resolver esa parte.

Diagrama

Descripción generada automáticamente

Lo primero es centrarse en la tabla equipamiento. La cual recibe como FK la PK de sedes.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Más adelante me di cuenta que cod\_sede tenía que ser INT y lo cambié.

En la foto anterior se puede ver la creación de la tabla equipamiento. Como siempre el código, salvo en algunos casos, es un CHAR(6) ya que creo que es mejor para poder tener más variedad de código a crear.

Un par de campos descriptivos que son VARCHAR(15) como tipo de equipamiento y modelo. Plazas es un SMALLINT pero que puede tener un valor nulo ya que quizá ese tipo de equipamiento no se refiera a un vehículo propio.

Añado también el cod\_sede como FK.

Diagrama

Descripción generada automáticamente

Antes de echar un vistazo a como va quedando el diagrama voy a resolver todo lo relacionado con el equipamiento.

En primera instancia debo de crear la tabla equipo, luego las dos tablas intermedias en cualquier orden.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Este es el fragmento de script de esas 3 tablas. Lo hacemos así y ahorramos algo de texto y de tiempo. El script se ejecuta bien, vamos a intentar ver el diagrama.

Diagrama

Descripción generada automáticamente

De nuevo, de la misma forma que en el ejercicio original, se empieza a complicar la visualización, pero espero poder mejorarlo en la parte final del ejercicio. Así es como quedan estas tablas relacionadas después de ejecutar el script.

Diagrama

Descripción generada automáticamente

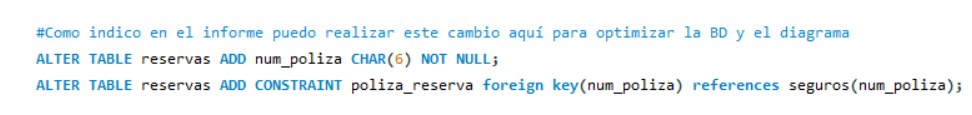
En el ejercicio también se hablaba de que se contrataban seguros para los grupos y los vehículos, así que vamos con esas partes, que ya son las últimas, y después veremos como se puede organizar el diagrama para que sea mínima mente legible y entendible.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En el script anterior se crean la tabla seguros y la tabla intermedia que la relaciona con grupos.

\*\*Incluyo esto aquí, ya que es una corrección. Después de revisar todo, me di cuenta de que podía ahorrarme esta tabla intermedia en la que los seguros se relacionan con los grupos, ya que es más eficiente si relaciono la tabla seguros con la tabla reservas, ya que esa tabla relaciona todo lo necesario con los grupos que contratan la reserva. Dicho esto procedo a eliminar la tabla intermedia del script y a modificar la tabla reservas.



Eliminando el código de la tabla intermedia y añadiendo estas dos líneas puedo optimizar todo mucho más, o eso creo yo.

De todo eso me di cuenta al ver la parte de vehículos y su tabla alquileres.

Ahora vamos con la parte de vehículos.

Diagrama

Descripción generada automáticamente

La tabla empresas\_vehiculos ya esta creada, así que solo tengo que añadir la tabla vehículos y la intermedia, alquiler vehículos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Aquí están creadas ya la tabla vehículos y la de alquiler\_vehiculos. En estas partes me estoy saltando las muestras intermedias del diagrama para aligerar las cosas.

Vamos, ahora ya sí, con la parte de las agencias para terminar con todo ello.

Diagrama

Descripción generada automáticamente

Bueno, esto ya fue la última parte de mi ejercicio anterior, que quizá no hacía falta para entender y ejecutar el ejercicio, pero ya que lo presenté así hay que terminarlo.

En este caso tengo que crear primero la tabla agencias, después de eso, puedo crear ya las tablas intermedias.

Vamos a ello.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

La creación de la tabla agencias es bastante estándar. Luego las otras dos tablas intermedias también, bastante común a lo que ya he hecho en el ejercicio con anterioridad, PK compuesta de dos FK.

Ejecuto el script por última vez (supongo):

Texto

Descripción generada automáticamente

Todo Ok, salvo que no lo ejecuté por última vez. Al tener tantas tablas he creado nombres de FK repetidos, así que ha tocado modificar un par de cosas y probar hasta que todo queda como tiene que quedar.

# DIAGRAMA FINAL

Después de generar todo el código necesario vamos a ejecutar la ingeniería inversa y ver como queda todo.

Diagrama

Descripción generada automáticamente

En un principio el programa lo genera así. Voy a intentar ordenarlo un poco para que quede más legible.

Viendo el diagrama y moviendo las tablas me doy cuenta de que no añadí matrícula como FK a la tabla alquileres\_vehiculos, cosa que arreglo en este momento mientras voy ordenando el diagrama.

Diagrama

Descripción generada automáticamente

Este sería el diagrama, colocándolo un poco más, aún así es un diagrama bastante grande, con muchas relaciones entre muchas tablas. A ratos se sigue haciendo difícil de leer.

# 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):



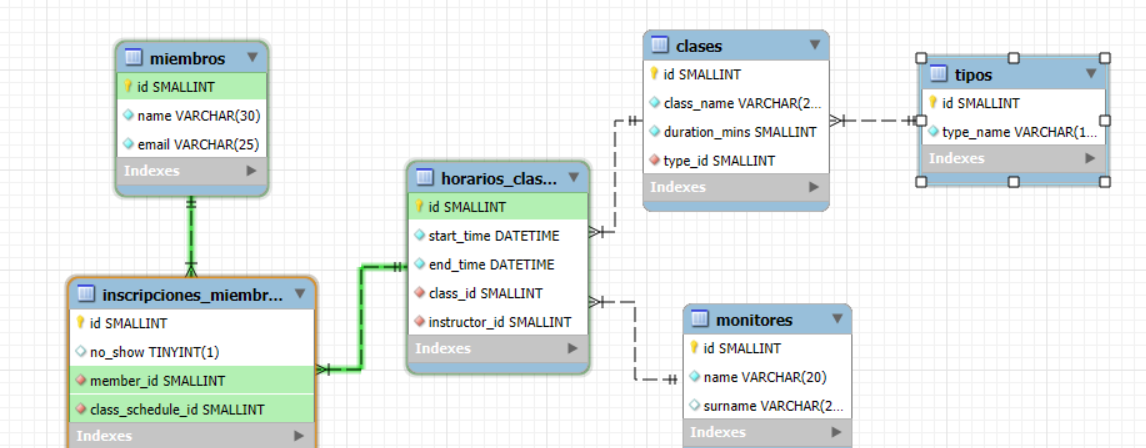
Y este es el comando que utilizo para realizarlo.

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);**