EMPEZANDO A IMPLEMENTAR

DAW 1 – Bases de Datos

Práctica 2.1

Una práctica donde se empezará a ver la transición desde los modelos Entidad-Relación y relacional, saltándose directamente hacia programación en SQL.

Marco Valiente Rodríguez
Mvalienter2501@educantabria.es

Índice

Contenido

Índice 1
Ejercicio 1 – El Bufete de Abogados 2
a. Identificación de las entidades 2
b. Atributos de las entidades 2
c. Identificación de las claves primarias 3
d. Identificación, cardinalidad, y atributos de relaciones 3
e. Esquema E/R 4
f. Salto al Esquema Relacional 4
g. Esquema Relacional 5
h. Creación del código SQL 5
i. Código SQL 6
j. Revisión y Validación 7
Ejercicio 2 – Estilo Libre 8
a. Descripción de la idea 8
b. Esquema E/R 9
c. Esquema Relacional 10
d. Código en MySQL 11
e. Revisión 12
Conclusiones 13
Bibliografía 14

Ejercicio 1 – El Bufete de Abogados

Se nos proporciona el siguiente enunciado.

> En este caso tenemos un bufete de abogados en el que se manejan una serie de expedientes. Cada expediente tiene un número de archivo único para su identificación y corresponde a un solo cliente; aunque un cliente puede tener varios expedientes. El expediente debe almacenar el período de trabajo (fecha de inicio y fecha de finalización). Cada expediente tiene un estado (en proceso, archivado, completado) y se debe conocer la fecha en que comenzó en ese estado. Para cada expediente, es necesario conocer los datos personales del cliente al que pertenece (DNI, nombre, primer apellido, segundo apellido, número de teléfono, dirección, correo electrónico, código postal).
>
> No debemos olvidarnos del estado del archivo, la lógica nos dice que habrá una tercera tabla, estado, que estará relacionada con la tabla de archivos.
>
> Crea un diagrama E/R, un diagrama relacional y la implementación en código SQL. Muestra en tu diagrama relacional los tipos de datos más apropiados para cada atributo. Muestra tu código SQL.

De estas cuales, al ser introducido directamente, vamos a elaborar en los diferentes pasos. Vamos a mantener el sistema de observación de las entidades tanto para el modelo E/R, como el modelo relacional, y como el nuevo salto, el código SQL por un análisis previo de las entidades elaborado aquí previamente.

a. Identificación de las entidades

Tal como hemos sido introducidos, tenemos que, en este caso, las entidades son simplemente la de expediente, la de cliente, y a petición del propio enunciado, tendremos una tercera entidad que será el estado de dicho archivo.

b. Atributos de las entidades

Los atributos son determinados únicamente por el texto, al no haber ningún tipo de exigencia sobre la falta de atributos.

Vamos a definir para cliente los datos que van a ser necesitados por parte del expediente, como el texto expresa de forma directa. Esto son: DNI, nombre, primer apellido, segundo apellido, número de teléfono, dirección, correo electrónico, código postal.

Para los atributos de expediente, se tiene: Número de archivo único (NDAU abreviado), fecha de inicio, fecha de finalización.

Por último, los atributos de estado de archivo, los cuales son: Tipo de Estado (En proceso/archivado/completado), Fecha de comienzo de estado, ID de Estado.

c. Identificación de las claves primarias

Las claves primarias han de ser capaces de diferenciarnos dentro de una misma tabla, y por eso, han de ser únicas.

Tenemos que, para cliente, el dato proporcionado DNI nos sirve como primary key.

Para expediente, dado por el enunciado en sí es el número de archivo único para su identificación, el cual, al ser para su identificación, es por definición una primary key. Lo abreviamos a NDAU para su mejor utilización y referencia a lo largo del ejercicio.

Por último, para estado, como no podemos simplemente usar el tipo de estado en el que está, sino que se necesita de al menos otro dato (un ID de estado, por ejemplo). Este dato, al tener que ceñirnos únicamente a la tabla, puede ser formado por el NDAU seguido del tipo de estado en el que está.

d. Identificación, cardinalidad, y atributos de relaciones

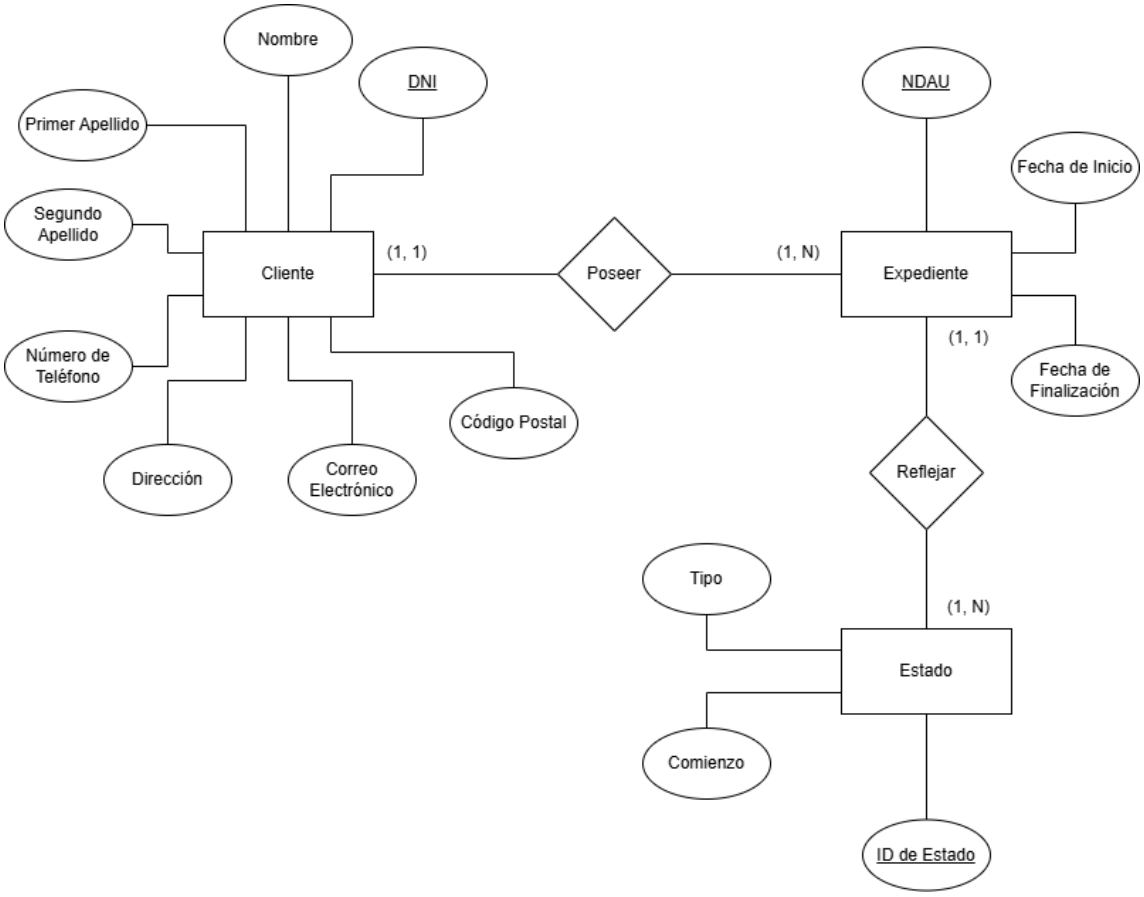
Describiendo las relaciones, tenemos que la relación entre cliente y expediente es una tal que un expediente pertenece a un cliente, es única por cliente con una relación 1 a 1, y un cliente puede poseer de 1 a N expedientes en contraposición.

A la hora de hacer la relación entre expediente y estado, tenemos que un estado refleja un expediente, y un expediente puede estar definido de 1 a 3 estados. Esto significa que la relación de estado a expediente es de 1 a 1, y de expediente a estado es de 1 a 3, que vamos a transformar a de 1 a N a la hora de la elaboración del esquema por dos motivos. El primero es para tomarlo como una formalización, y el segundo es el de habilitar en el futuro una mejor adición de nuevos tipos de estado si es deseado.

Debido a estas relaciones siendo no de N a N, tenemos que ninguna de estas relaciones posee un atributo.

e. Esquema E/R

Con estos datos, podemos elaborar sin problemas nuestro esquema E/R.



f. Salto al Esquema Relacional

Al pasar del Modelo E/R al nuevo modelo relacional, tenemos que no vamos a necesitar la creación de ninguna tabla intermedia, ya que no relación es de N a N.

El único dato para mencionar que cambia del modelo E/R es que el ID de estado puede ser descompuesto a la hora de nombrar la Foreign Key que recibe de expediente. Sin embargo, se puede considerar que es mejor dejarlo como un dato introducido aparte por un motivo sencillo; es más eficiente a la hora de buscar datos, aunque sea menos eficiente a la hora de guardarlos.

g. Esquema Relacional

Y con esto, elaboramos nuestro esquema relacional.

h. Creación del código SQL

Una vez que tenemos todo esto declarado, lo único que queda es elaborar y trabajar el código para ver cómo sería en cuando este esté introducido dentro de MySQL.

Para esta elaboración, voy a usar dos herramientas. Una de estas es el programa Sublime Text, el cual es una plantilla para la edición y corrección de código, aunque no ejecución, y la segunda será MySQL workbench para su ejecución.

Ha medida que se trabaja, se ha de asignar el tamaño a las variables.

- DNI tiene 9 valores, 8 números y 1 letra
- Nombre le damos hasta 20 letras para darle espacio, por precaución
- El Primer apellido también, así como el segundo apellido
- El número de teléfono son 12, un signo +, dos primeros números de localidad, y el resto son tu número
- Dirección tienen hasta 50 por una estimación.
- El código postal es un VARCHAR, ya que los códigos postales pueden tener 6 caracteres si son extranjeros como de Reino Unido.
- Con sus ajustes y requerimientos, hechos, como que los datos importantes no sean nulos.
- Primary Key es DNI.
- Saltando a la siguiente tabla, el Expediente, tenemos que la decisión del tamaño del NDAU va a ser importante. Un dato a tener en cuenta es que va a ser un valor numérico, pero no vamos a operar con este, por lo que no tiene sentido guardarlo como un INT o parecido, mejor como un VARCHAR. Le vamos a dar un rango de 5, es decir, hasta 99999, un numero razonable para este negocio si se llega a expandir lo suficiente.
- Describimos las fechas con su dato fecha, y por último asignamos a estos valores NOT NULL.
- Finalizamos con la última tabla, la cual establecemos so FK de forma apropiada y le damos dos caracteres adicionales aparte de los 5 del NDAU, para tener espacio de definir su estado. Con la finalización de esto, el código SQL está terminado.

i. Código SQL

El código SQL es el siguiente.

```sql
DROP DATABASE IF EXISTS BufeteAbogados;
CREATE DATABASE IF NOT EXISTS BufeteAbogados;
USE BufeteAbogados;
CREATE TABLE IF NOT EXISTS cliente(
    dni CHAR(9),
    nombre VARCHAR(20) NOT NULL,
    primer_apellido VARCHAR(20) NOT NULL,
    segundo_apellido VARCHAR(20),
    numero_de_telefono CHAR(12) NOT NULL,
    direccion VARCHAR(50) NOT NULL,
    correo_electronico VARCHAR(76) NOT NULL,
    codigo_postal VARCHAR(6) NOT NULL,
    CONSTRAINT dni PRIMARY KEY (dni)
);
CREATE TABLE IF NOT EXISTS expendiente(
    ndau CHAR(5),
    dni CHAR(9) NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE,
    CONSTRAINT ndau PRIMARY KEY (ndau),
    CONSTRAINT dni FOREIGN KEY (dni) REFERENCES cliente (dni)
);
CREATE TABLE IF NOT EXISTS estado(
    id_estado CHAR(7),
    ndau CHAR(5) NOT NULL,
    tipo VARCHAR(15) NOT NULL,
    fecha_comienzo DATE NOT NULL,
    CONSTRAINT estado_pk PRIMARY KEY (id_estado),
    CONSTRAINT ndau FOREIGN KEY (ndau) REFERENCES expendiente (ndau)
);
```

Action Output

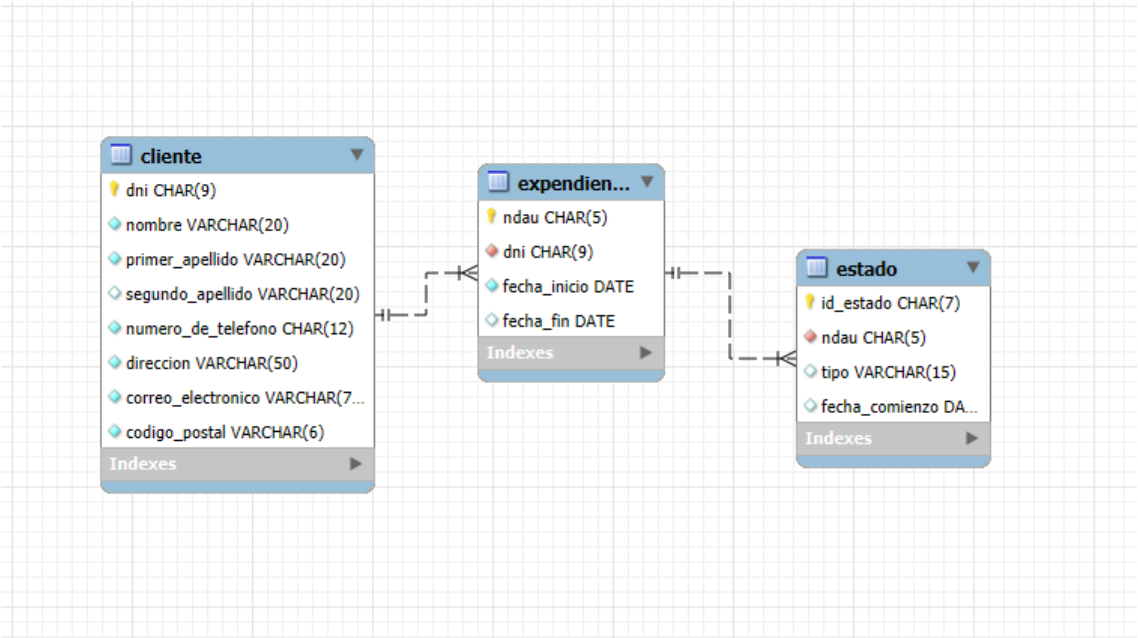
#	Time	Action	Message
1	11:06:24	DROP DATABASE IF EXISTS BufeteAbogados	3 row(s) affected
2	11:06:24	CREATE DATABASE IF NOT EXISTS BufeteAbogados	1 row(s) affected
3	11:06:24	USE BufeteAbogados	0 row(s) affected
4	11:06:24	CREATE TABLE IF NOT EXISTS cliente(dni CHAR(9), nombre VARCHAR(20) NOT NULL, primer_apellido VARCHAR(20) NOT NULL, segundo_apellido ...	0 row(s) affected
5	11:06:24	CREATE TABLE IF NOT EXISTS expendiente(ndau CHAR(5), dni CHAR(9) NOT NULL, fecha_inicio DATE NOT NULL, fecha_fin DATE, CONSTRAINT...	0 row(s) affected
6	11:06:24	CREATE TABLE IF NOT EXISTS estado(id_estado CHAR(7), ndau CHAR(5) NOT NULL, tipo VARCHAR(15) NOT NULL, fecha_comienzo DATE NOT N...	0 row(s) affected

j. Revisión y Validación

Hemos elaborado este esquema con los posibles estados de forma apropiada. Una de las posibilidades a la hora de la elaboración de esta base de datos, es la de la creación de la tabla de los posibles estados como una tabla genérica, y después cada estado puntual siendo su propio estado. Considero que, a forma práctica, no se necesita crear un nuevo estado para su clasificación de tal manera, sino que ser descrito en un atributo y ya es suficiente, centrándonos en las aplicaciones prácticas del ejercicio.

De esta forma, lo nuevo que queda para revisar es nuestro esquema SQL, ya que la revisión de los otros dos es correcta. Usando MySQL, podemos usar una función de reversión para comprobar si la tabla generada se parece a nuestro esquema relacional, un elemento básico para la comprobación de que esté elaborado no de forma errónea.

El resultado de esto, es la siguiente imagen:



Amblas tablas son prácticamente idénticas, quitando pequeños detalles como que esta marca las unidades, por lo que se puede confirmar que está bien realizado.

Ejercicio 2 – Estilo Libre

Con el ejercicio anterior finalizado, ahora vamos a dar pie a una idea propia y a intentar adaptarla en el sistema de MySQL después de su evaluación adecuada.

La elección del tema es sujeta a nuestro criterio, y debido a esto, voy a optar por la elaboración de un nuevo sistema de Streaming, ya que en un contexto actual ese modelo de negocios está entrando en crisis con clientes huyendo a la piratería.

a. Descripción de la idea

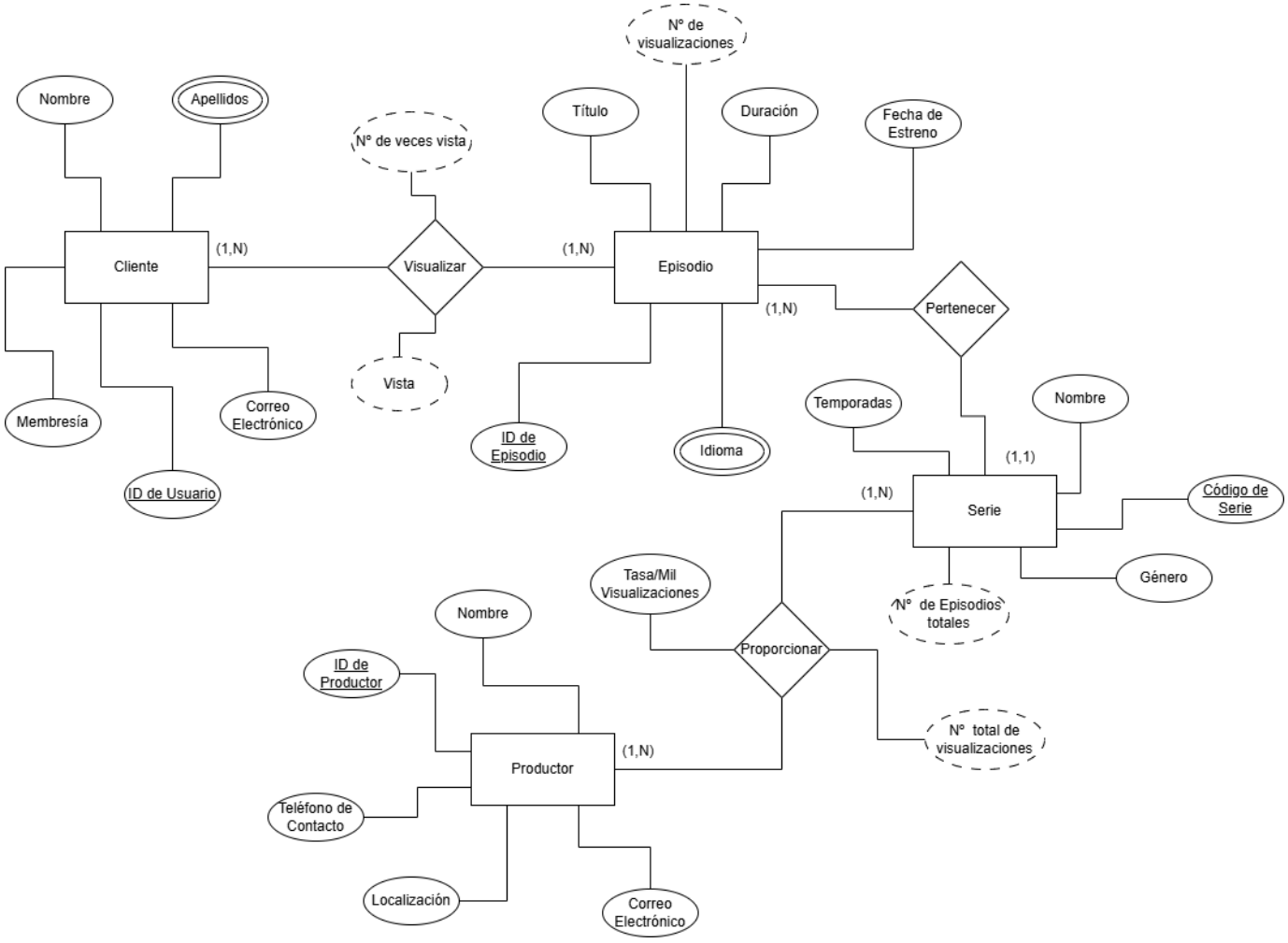
Queremos diseñar una plataforma de Streaming donde los usuarios puedan consumir capítulos o episodios de series. Cada serie tiene un código de serie, un nombre, y el género al que pertenece. Cada cliente puede tener una subscripción a nuestro modelo que le permite o no tener acceso a los últimos episodios que hayan salido esta semana, así como tiene un nombre de usuario y un correo electrónico.

Una serie está compuesta por uno o más episodios, los cuales los clientes pueden ver, y los episodios pertenecen a una serie. Los clientes pueden ver, además, si ya han visto al menos una vez a un particular episodio, para saber por qué episodio van.

Las series tienen asociadas a estas un único productor particular, las cuales reciben una comisión al suministrarnos las series que nosotros transmitimos, por cada mil visualizaciones, y estas financian directamente a los trabajadores del estudio que elaboran los episodios, sin los hábitos indirectos y cuestionables que estos modelos están acostumbrados a seguir. Este productor puede cambiar con cambios de estudio. Un productor tiene un nombre, teléfono de contacto, y localización.

b. Esquema E/R

Nuestro esquema E/R es el siguiente:



Describimos solo algunos atributos para las relaciones de N a N. Estos serán desarrollados más adelante con el paso al sistema relacional. Se elaboran los atributos que van a ser requeridos para este modelo libre.

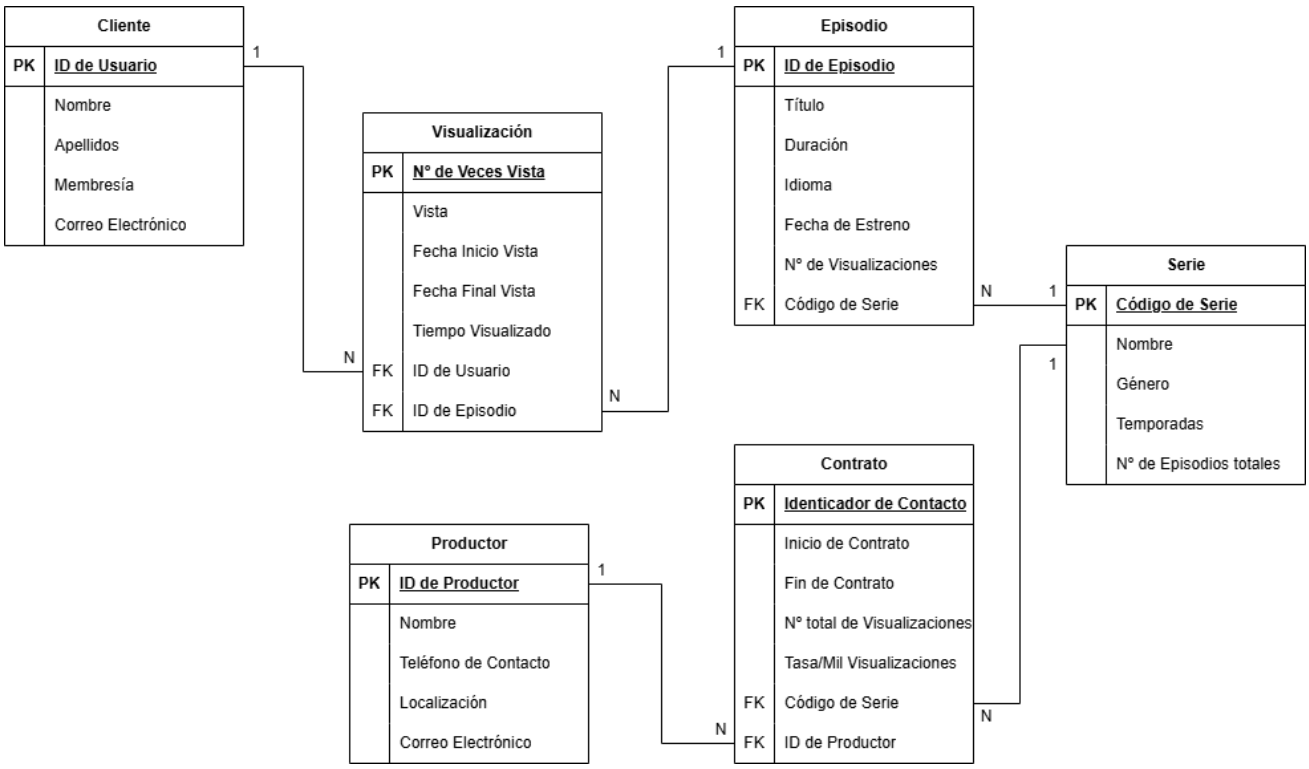
Por el enunciado, tenemos el desarrollo de la idea, pero esta no nombra todos los atributos, solo los más esenciales. Por tanto, además, al ser nosotros mismos los dueños de este proyecto, tenemos mayores libertades para decidir añadir más atributos.

Los más importante a comentar es que, aunque Vista y Nº de Veces Vista parecen similares, Vista es u booleano que sirve como indicación si el episodio ha sido visualizado anteriormente completamente, y el Nº de Veces Vista incluye la cantidad de veces que un usuario ha hecho click para ver este episodio.

Por último, el Nº total de visualizaciones para la relación de proporcionar se refieren a las visualizaciones dentro de la duración del contrato.

c. Esquema Relacional

Con el esquema E/R y el desarrollo hecho, nos queda el salto al esquema relacional.



Se ha logrado hacer la transición exitosamente, aunque con algunos componentes de más.

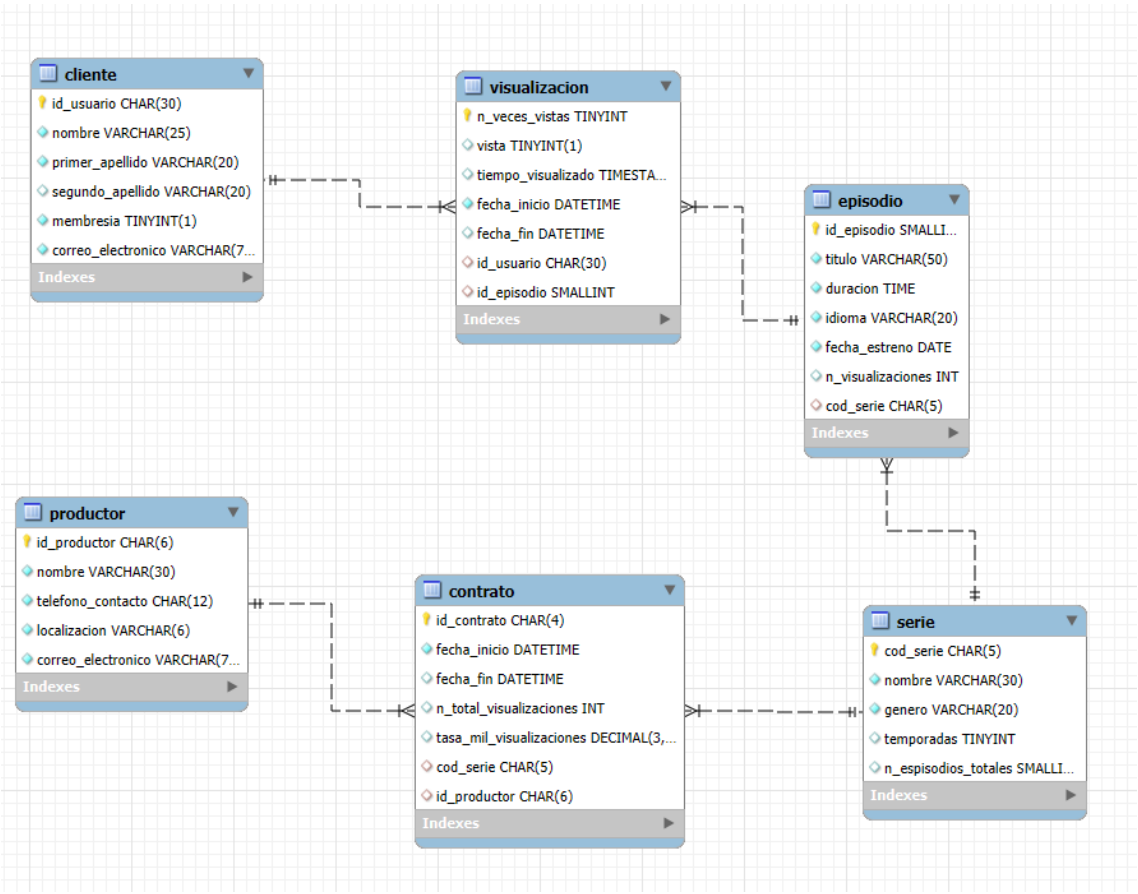
d. Código en MySQL

Este es el salto a la hora de la elaboración de las gráficas.

```sql
DROP DATABASE IF EXISTS StreamingService;
CREATE DATABASE IF NOT EXISTS StreamingService;
USE StreamingService;
CREATE TABLE IF NOT EXISTS cliente(
    id_usuario CHAR(30),
    nombre VARCHAR(25) NOT NULL,
    primer_apellido VARCHAR(20) NOT NULL,
    segundo_apellido VARCHAR(20),
    membresia BOOLEAN NOT NULL,
    correo_electronico VARCHAR(76) NOT NULL,
    CONSTRAINT id_usuario PRIMARY KEY (id_usuario)
);
CREATE TABLE IF NOT EXISTS serie(
    cod_serie CHAR(5),
    nombre VARCHAR(30) NOT NULL,
    genero VARCHAR(20) NOT NULL,
    temporadas TINYINT,
    n_espisodios_totales SMALLINT,
    CONSTRAINT cod_serie PRIMARY KEY (cod_serie)
);
CREATE TABLE IF NOT EXISTS productor(
    id_productor CHAR(6),
    nombre VARCHAR(30) NOT NULL,
    telefono_contacto CHAR(12) NOT NULL,
    localizacion VARCHAR(6) NOT NULL,
    correo_electronico VARCHAR(76) NOT NULL,
    CONSTRAINT id_productor PRIMARY KEY (id_productor)
);
CREATE TABLE IF NOT EXISTS episodio(
    id_episodio SMALLINT,
    titulo VARCHAR(50) NOT NULL,
    duracion TIME NOT NULL,
    idioma VARCHAR(20) NOT NULL,
    fecha_estreno DATE NOT NULL,
    n_visualizaciones INT,
    cod_serie CHAR(5),
    CONSTRAINT id_episodio PRIMARY KEY (id_episodio),
    CONSTRAINT cod_serie FOREIGN KEY (cod_serie) REFERENCES serie (cod_serie)
);
CREATE TABLE IF NOT EXISTS visualizacion(
    n_veces_vistas TINYINT,
    vista BOOLEAN,
    tiempo_visualizado TIMESTAMP,
    fecha_inicio DATETIME NOT NULL,
    fecha_fin DATETIME,
    id_usuario CHAR(30),
    id_episodio SMALLINT,
    CONSTRAINT PK_visualizacion PRIMARY KEY (n_veces_vistas),
    CONSTRAINT id_usuario FOREIGN KEY (id_usuario) REFERENCES cliente (id_usuario),
    CONSTRAINT id_episodio FOREIGN KEY (id_episodio) REFERENCES episodio (id_episodio)
);
CREATE TABLE IF NOT EXISTS contrato(
    id_contrato CHAR(4)
    fecha_inicio DATETIME NOT NULL,
    fecha_fin DATETIME,
    n_total_visualizaciones INT,
    tasa_mil_visualizaciones DECIMAL(3,2),
    cod_serie CHAR(5),
    id_productor CHAR(6),
    CONSTRAINT PK_contrato PRIMARY KEY (id_contrato),
    CONSTRAINT serie_codigo FOREIGN KEY (cod_serie) REFERENCES serie (cod_serie),
    CONSTRAINT id_productor FOREIGN KEY (id_productor) REFERENCES productor (id_productor)
);
```

e. Revisión

Tenemos la siguiente reversión:



Encaja perfectamente. Los datos elegidos son los escogidos apropiados para nuestro modelo.

Conclusiones

No hay mucho que comentar de este trabajo, aparte de los errores que se han encontrado a medida que este trabajo estuviese siendo desarrollado. Hemos encontrado que a la hora de declarar cod_serie como una FK para tanto episodio como para contrato, nos salió un error ya que este nombre de FK ya estaba declarado, por tanto, se puede ver cómo va a interesar en el futuro a ponerle nombres distintos para evitar que este tipo de fallos aparezcan.

Quitando esto, se podría decir que el salto realizado a el nuevo sistema de programación en SQL ha resultado exitoso, con los suficientes conocimientos y problemas superados.

Bibliografía

- Recurso del curso de Desarrollo de Aplicaciones Web, Documento de Empezando a Implementar y Creación de Base de Datos.