

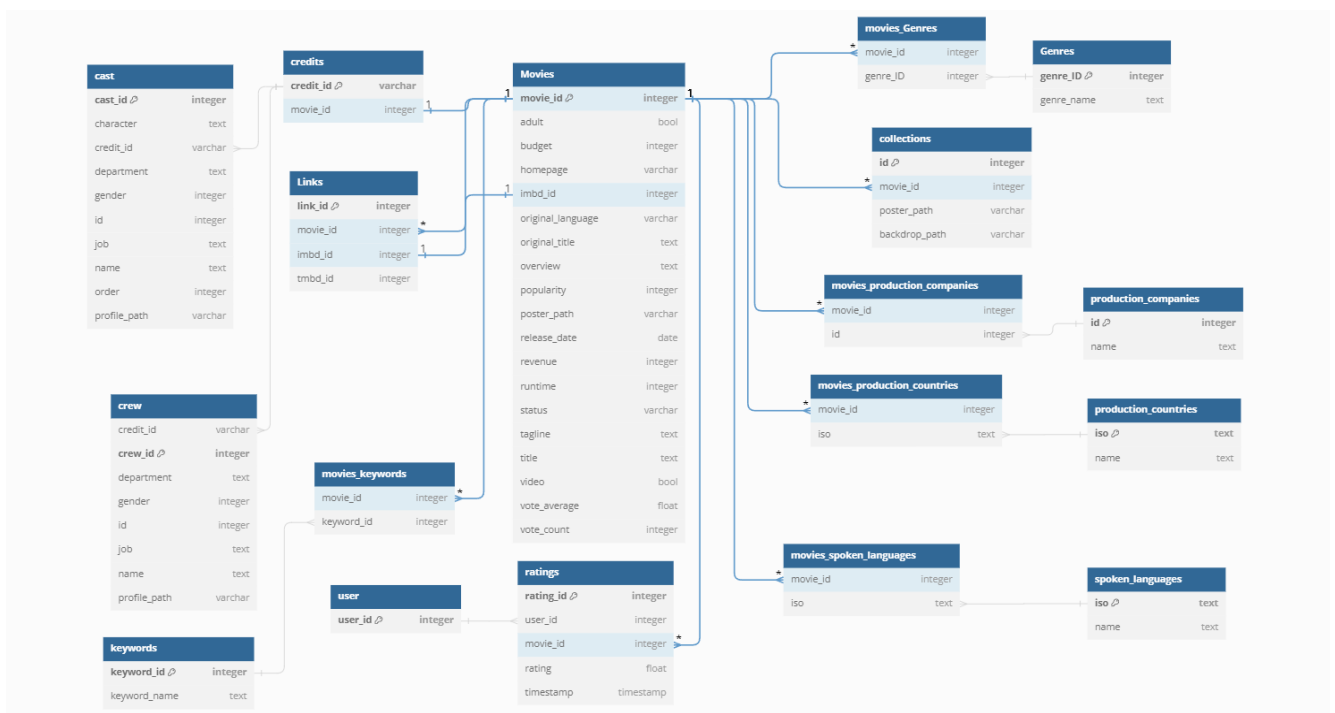
## PRÁCTICA 2 MAGD

Jaime Fuentes Martínez

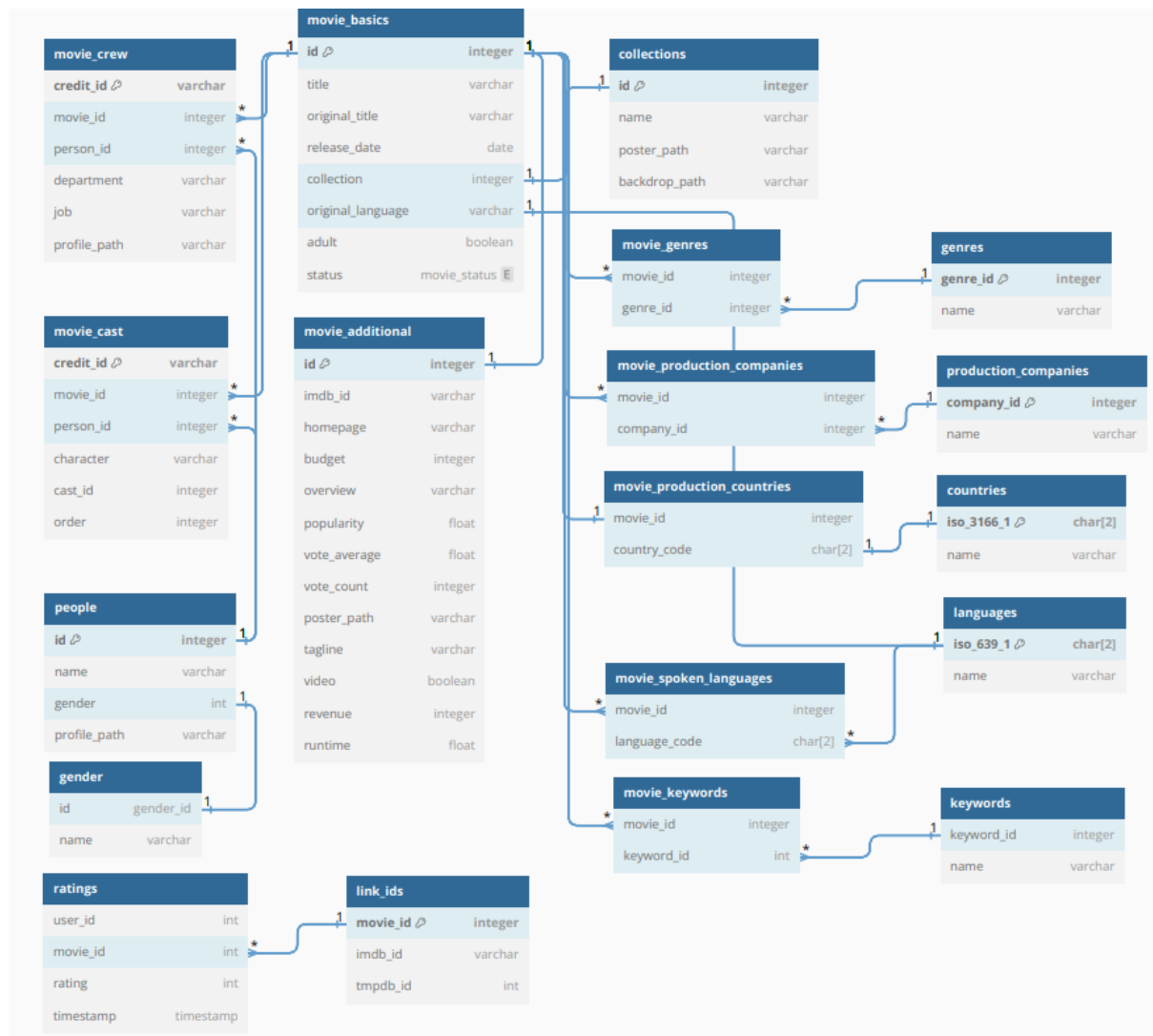
David López Valdivia

1. Diferencias entre nuestro diagrama relacional de la P1 y el diagrama que se nos proporciona para esta práctica. ¿Están justificadas estas diferencias? ¿Pueden considerarse errores o simplemente diferentes opciones o alternativas de diseño?

Este el diagrama relacional que hicimos anteriormente:



Y debajo se muestra el diagrama que se nos proporciona para la P2:



En primer lugar, nosotros decidimos no separar entre una parte de información básica de movies (*movie\_basics*) y otra entidad llamada *movie\_additional*, ya que nos parecía que algunos atributos encajaban muy bien en ambas tablas. Aun así, creemos que ambas formas son correctas porque los ambos modelos son óptimos haya o no una división o clasificación en *movies*..

Podemos notar sin embargo que nosotros no hemos llegado a elaborar dos entidades para gender y people, las cuales tienen mucho sentido ya que, por ejemplo, a la hora de analizar el casting se van

encontrando muchas personas que tienen nombre, una identificación, por lo que lo correcto sería crear un csv para contener toda la información de estas personas.

A su vez, cada persona tiene en puede tener en *gender* tres posibles valores 0,1,2 , por lo que vemos entendible la creación de una nueva entidad para saber que 0 es neutro, 1 es masculino y 2 femenino. No consideramos necesario el crear la entidad *gender* pero si reconocemos que lo de *people* es un error.

A su vez, tampoco le vemos sentido a una de nuestras entidades, que es *users*. La razón está en que no tiene lógica crear una entidad basada en un único atributo ; si los usuarios tuviesen nombres registrados o cualquier información adicional si que lo veríamos necesario, pero en este caso es otro error.

Por otro lado podemos ver que ambas coinciden en *collections, movie\_cast, movie\_crew* y en las entidades con formato json y sus respectivas relaciones con *movies* (salvo una excepción, la clave primaria en *movie\_cast* y *movie\_crew* es *credit\_id*).

## 2. Nuestro preprocesamiento de los datos.

¿Cómo habéis planteado el problema y qué enfoque habéis usado?

Hemos intentado abordar el problema de la manera más eficiente posible, diversificando tareas en distintas funciones que se llaman unas a otras para no repetir código innecesariamente y que con pocas llamadas a estas funciones se generen todos los csv que nos pedían. No hemos necesitado recodificar variables, pues nuestras funciones estaban escritas todas usando try y except para no tener problemas con distintos fallos que contuviese el csv y no perder información eliminando columnas (salvo duplicadas) aunque tuviesen datos faltantes, pues la ausencia de algunos datos no implica inutilidad de los mismos.

¿Cómo habéis estructurado el código?

La estructura del código consiste en un documento de *Jupyter Notebook* con varias celdas:

Primero, una celda donde se ha de colocar el path a todos los datasets originales. Después la celda con el preprocesado que se encarga de eliminar los valores sobrantes. Por último la sucesión de celdas con las distintas funciones que crean los datasets ya filtrados preparados para meter en las tablas.

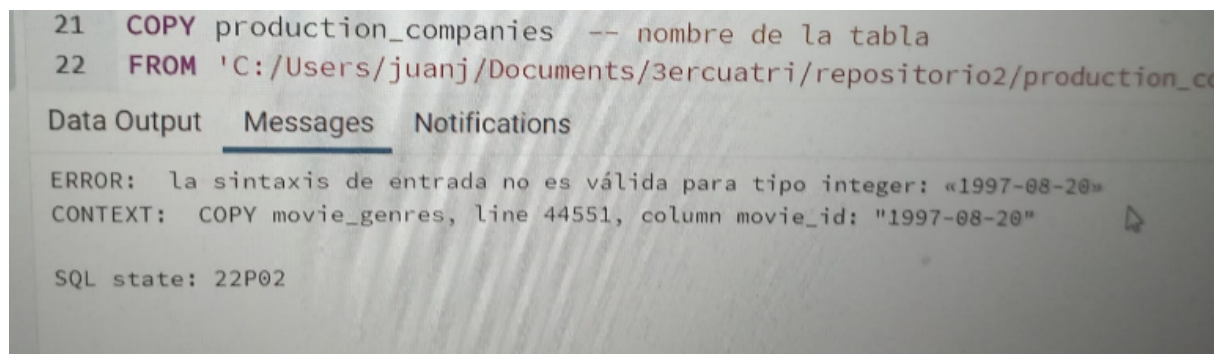
¿Qué problemas habéis encontrado?

El primer problema que encontramos es leer los jsons y poder trabajar con ellos. Este problema lo solucionamos con la librería *AST* que nos permite leerlos y “traducirlos” para poder extraer la información específica que necesitamos de ellos.

El segundo problema que nos hemos encontrado es la cantidad de errores que hay en el dataset, datos faltantes o distintos tipos de datos dentro de una misma columna que generaban errores dentro del código. Esto lo hemos arreglado entre el preprocesado y usando distintas excepciones, try y except y código abierto en nuestras funciones para no perder información y no generar errores y que se detenga la ejecución del código.

Otro problema que nos hemos encontrado es el tamaño de los datasets y un alto tiempo de apertura, lectura de los mismos y de ejecución y modificación con código sobre ellos. Para ello, hemos intentado optimizar nuestro código de la mejor manera posible.

Ya finalizando la práctica, al ejecutar en *PgAdmin* el fichero *createdb\_script.sql* nos saltó este error:



```
21 COPY production_companies -- nombre de la tabla
22 FROM 'C:/Users/juanj/Documents/3ercuatri/repositorio2/production_c

Data Output  Messages  Notifications
ERROR: la sintaxis de entrada no es válida para tipo integer: «1997-08-20»
CONTEXT: COPY movie_genres, line 44551, column movie_id: "1997-08-20"
SQL state: 22P02
```

Al parecer, existían tres columnas en *movies* con una fecha como *id*. Al sernos imposible cambiar este a un nuevo identificador, optamos por eliminar estas, y justo después tuvo éxito la ejecución del fichero.

¿Ha habido que modificar o eliminar datos?

Hemos modificado varios datos para no tener que eliminar ninguno que no fuese primordial quitar, pues no queríamos perder información dado que consideramos que cada columna, por poca información que tuviese, tenía información. Por ello, solo hemos eliminado los datos duplicados y modificado datos vacíos o erróneos, a excepción del caso concreto explicado antes.