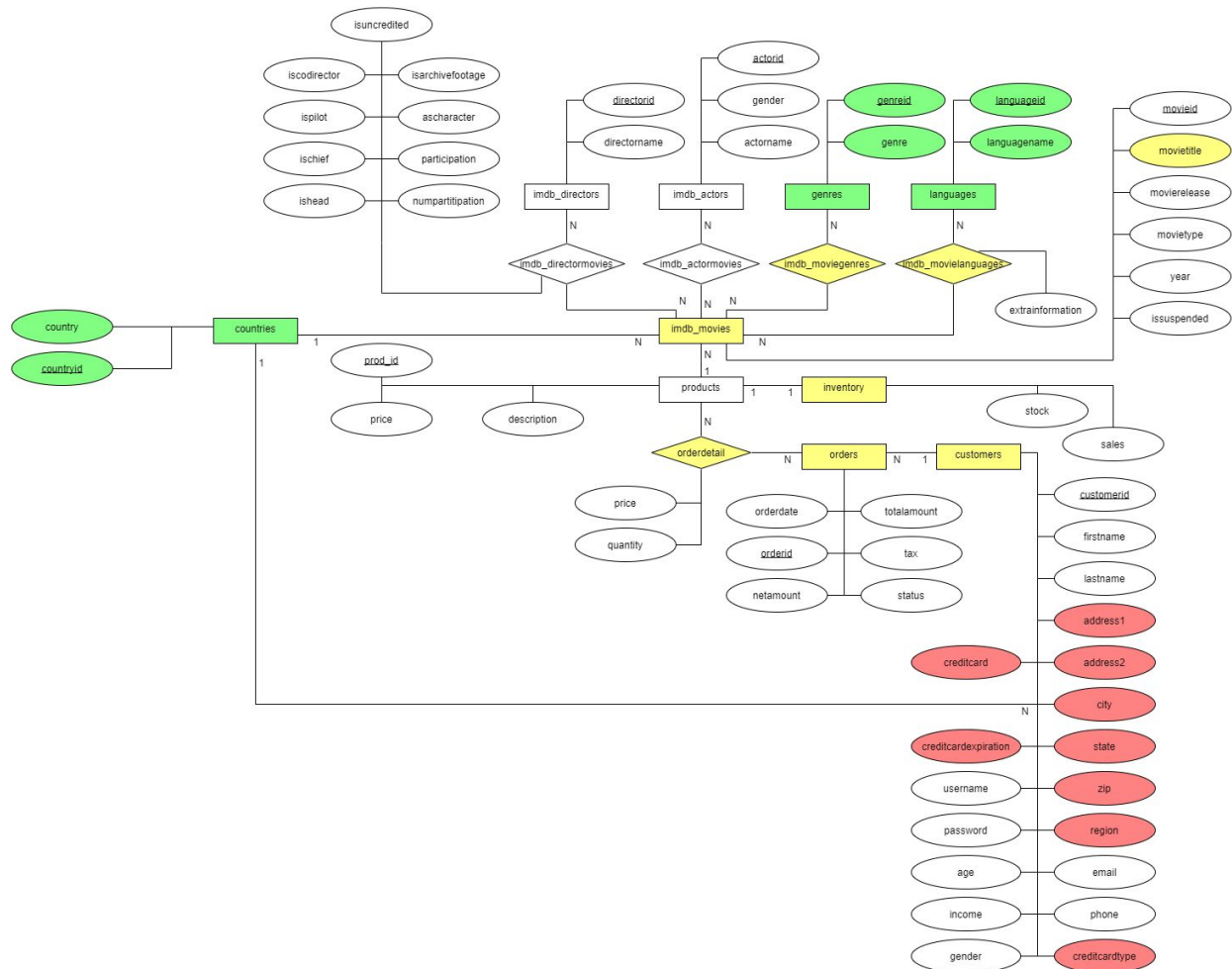


SI - Práctica 2

Group 1391

Parte 1

En esta parte hemos tenido que hacer el diagrama de la base de datos, rediseñarla y hacer un nuevo diagrama. El diagrama final de como nos ha quedado la base de datos es:



En verde están los elementos nuevos que hemos creado nosotros. En amarillo están los elementos que hemos modificado nosotros. Los elementos en rojo son

los que queríamos actualizar, pero que no hemos podido por cuestión de tiempo y que serán actualizados en la entrega final de ser posible.

Los cambios que hemos hecho en la base de datos son:

- Hemos creado una entidad para el idioma y modificado la relación para unirla con las películas.
- Hemos creado una entidad para el género y modificado la relación para unirla con las películas.
- Hemos creado una entidad para el país y quitado su relación. Ahora el país se usa tanto en el cliente como en las películas.
- Hemos modificado orderdetail para añadirle claves foráneas de products y orders.
- Hemos modificado orders para añadirle la clave foránea de customer.
- Hemos modificado inventory para añadirle la clave foránea de product.
- Hemos modificado imdb_actormovies para añadirle las claves foráneas de actor y de movie.
- Queremos añadir la tarjeta de crédito a su propia entidad, para que así una persona pueda tener varias tarjetas de crédito.
- Queremos añadir la dirección a su propia entidad, para que así una persona pueda tener varias direcciones guardadas.

Después de hacer el diagrama y el actualiza.sql pasamos a hacer las queries.

setPrice.sql: En esta consulta el objetivo era poner el precio a cada venta en la tabla orderdetails, reduciendo un 2% por cada año desde la venta de la película. Para resolverlo primero creamos una view para obtener una tabla con el precio y año de cada producto de todos los pedidos. Después creamos una view para obtener el precio de las películas reducidas un 2% anual. Y, por último, completamos la columna price de la tabla orderdetail.

setOrderAmount.sql: El objetivo de esta consulta era ponerle un precio final al pedido, y añadirle el impuesto para conseguir un precio para el cliente. Lo primero que hacemos es crear una view para obtener la suma de los precios de la película del pedido. Después creamos el procedimiento que completa las columnas netamount y totalamount de la tabla orders.

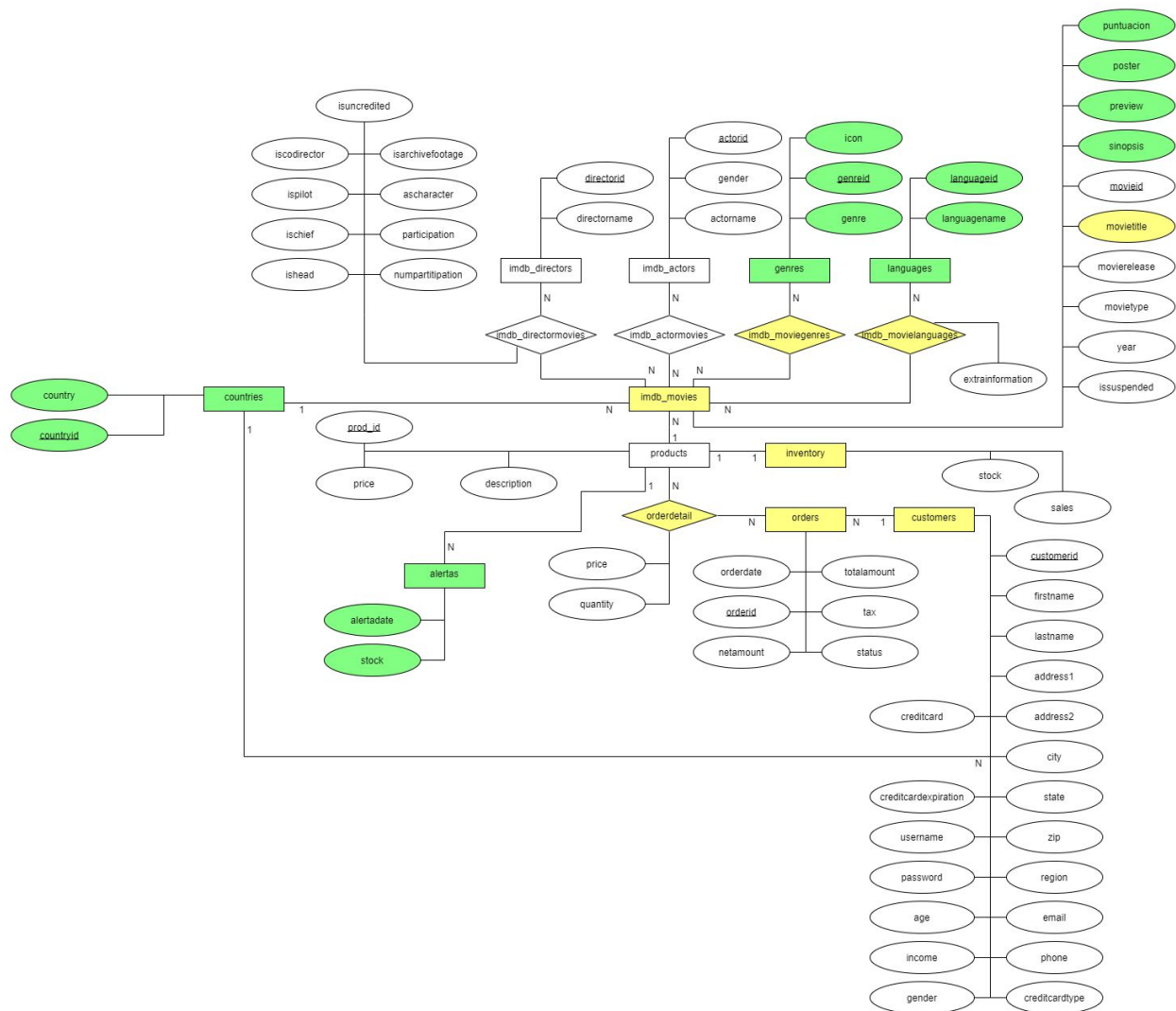
getTopVentas.sql: En este procedimiento almacenado el objetivo era obtener las películas más vendidas cada año junto con su número de ventas y el año el que es la más vendida entre dos años dados. Hemos hecho que en caso de empate se muestren dos ese año. Primero creamos una view para obtener todas las películas vendidas y el año en el que se vendieron. Segundo creamos una view que cuenta cuántas veces una película fue vendida cada año. Tercero creamos una view para obtener el número de veces que ha sido vendida la película más vendida de cada año. Por último terminamos creando el procedimiento que consigue las películas más vendidas de cada año y varias en caso de empate.

getTopMonths.sql: En este procedimiento almacenado el objetivo era obtener los meses que superasen el mínimo de ventas o de dinero que le dieramos como argumentos. Lo primero que hicimos fue crear una view con las ventas totales, ganancias, mes y año. Lo siguiente y último fue hacer el procedimiento que separa de la tabla anterior los meses que cumplen la condición. Devuelve el mes, el año, el dinero y las ventas.

La mayor complicación que hemos tenido con la práctica fue hacer los procedimientos, ya que no conseguíamos que funcionasen correctamente y a veces se quedaban haciendo la query eternamente.

Parte 2

En esta parte hemos tenido que rehacer el diagrama de la base de datos, añadiendo las modificaciones que hemos hecho en esta última parte, hacer los triggers y hacer que la página funcione con la base de datos, en vez de utilizar jsons y dats:



Los cambios en el diagrama esta vez han sido menores. Los tres cambios que hemos hecho son:

- Hemos creado una entidad para las alertas para que nos avise de cuando nos quedamos sin stock de una película.
- Hemos añadido cuatro atributos a imbd_movies que son la puntuación, la sinopsis, el poster y la preview, las cuales hacen que las películas se puedan mostrar en la página como cuando usábamos el catalogue.json, aunque hemos quitado crítica y reparto.
- Hemos añadido un campo icon a genres para que tenga la ubicación del icono de esa categoría o una por defecto en caso de no tener ningún icono.

Después de las actualizaciones de la base de datos hemos hecho los triggers.

updOrders.sql: Trigger que actualiza la información de la tabla orders cuando se modifica, añade o elimina un artículo en el carrito (tabla orderdetail). Esto hace que el precio en orders esté siempre actualizado y listo para ser pagado en cualquier momento. Cabe destacar que este trigger necesita de la función setOrderAmount.

updInventory.sql: Trigger que actualiza las tablas inventory y orders cuando se finalice la compra (status = Paid). También crea una alerta en la tabla alertas cuando el stock de alguna película a comprar llega a cero. Esto hace que nuestro stock de películas se mantenga actualizado, pudiendo avisarnos de cuando habría que reponerlas.

En esta parte hemos integrado la base de datos con la página web de la práctica anterior, con lo que borramos el catálogo y las categorías que teníamos antes (ya que ahora las obtendremos desde la base de datos). Para esta integración, muchas de las cosas que antes hacíamos con jsons, dats y sesiones las hacemos ahora con funciones que nos devuelven el resultado de queries de la base de datos.

getTopVentas: Nos devuelve las películas de la query homónima entre los años 2018 y 2020. Se usa para mostrar las películas en home.

db_register: Crea un nuevo usuario en la base de datos e inicializa su username, password, card, email e income. Es el único caso en el que se usa.

db_login: Inicia sesión con un usuario ya existente. Es el único caso en el que se usa.

db_saldo: Devuelve el saldo de un usuario dado. La usamos para ver el saldo de los clientes desde el carrito, su página de perfil o para cualquier operación que requiera comprobar cuánto dinero tiene.

db_update_saldo: Cambia el saldo de un usuario dado al saldo que le pasamos como argumento. La utilizamos para añadir o quitar saldo.

update_catalogue: Actualiza el catálogo de películas para mostrar las de la base de datos. La usamos para poblar el json de las películas.

update_categories: Devuelve una lista con todas las categorías de las películas. La usamos para poblar el menú lateral con categorías.

db_delete_order: Borra la orden dada de la base de datos. En caso de hacer una compra directa, tenemos que crear un pedido para calcular el precio total con impuestos. Si el cliente no tuviera dinero para pagarla, usaríamos esta función para borrar el pedido.

db_add_order: Crea una orden nueva para el usuario dado y con el status Processed. La usamos en db_get_order cuando no hay una cesta anterior.

db_get_order: devuelve la orden que usamos como carrito de un usuario dado, y en caso de no haber ninguna, la crea. La usamos al hacer login, para cargar la cesta anterior desde la base de datos.

db_update_order: Añade o quita una película de la orden dada. La usamos para añadir o quitar películas de la cesta

db_buy_order: Cambia el estado de una orden de Processed (en carrito) a Paid (comprada). La usamos al comprar el carrito.

db_get_cesta: Devuelve la lista de películas de la cesta. La usamos para cargar la cesta en la página.


db_order_price: Devuelve el precio final de la orden dada. La usamos para calcular el precio de la cesta.

db_get_history: Devuelve la lista de películas compradas por el cliente. La usamos para el historial en la página del perfil


Con estas funciones hemos conseguido integrar la base de datos en la página, haciendo que funcione como antes con los jsons.

Otra cosa que hemos hecho en esta práctica es arreglar algún problema que nos quedaba con el reescalado y algún

Galería


 **Filmbrary**

Buscar...

eladmin2 

Interestellar (2014)


EXPERIENCE IT IN IMAX




INTERSTELLAR

COWING SOON

Your Name (2016)




BEST SELLER of 2018:
Wizard of Oz, The (1939)




NO POSTER AVAILABLE

BEST SELLER of 2019:
Life Less Ordinary, A (1997)




NO POSTER AVAILABLE


BEST SELLER of 2020:
Life with Mikey (1993)




© Copyright Filmbrary.com | J.M Freire - Miguel Herrera

 **Filmbrary**

Buscar...

eladmin2 


Wizard of Oz, The (1939)



NO POSTER AVAILABLE

Eliminar

Wizard of Oz, The (1939)




NO POSTER AVAILABLE

Eliminar

Interestellar (2014)

EXPERIENCE IT IN IMAX




INTERSTELLAR

Eliminar

Interestellar (2014)


EXPERIENCE IT IN IMAX



INTERSTELLAR

Eliminar

Your Name (2016)



localhost5001/movie_id_442893 © Copyright Filmbrary.com | J.M Freire - Miguel Herrera



Life with Mikey (1993)



NO POSTER
AVAILABLE



Your Name (2016)

8.99 €

👑 (1)

Comprar Ahora

Sinopsis:

El joven Taki vive en Tokio; la joven Mitsuha, en un pequeño pueblo en las montañas. Durante el sueño, los cuerpos de ambos se intercambian. Recluidos en un cuerpo que les resulta extraño, comienzan a comunicarse.

Puntuación:

6.8 / 10

...

Filmbrary

Buscar...

🔍

eladmin2

👤

eladmin2

Cerrar Sesión

Saldo: 24€

Añadir saldo

Escribir dinero que añadir a sald

Confirmar

Película	Precio	Fecha
Your Name	8.99	2020-11-26
Your Name	8.99	2020-11-26
Your Name	8.99	2020-11-26
Your Name	8.99	2020-11-26
Life Less Ordinary, A	19.0	2020-11-26
Life with Mikey	16.0	2020-11-26

© Copyright Filmbrary.com | J.M Freire - Miguel Herrera

...

Filmbrary

Buscar...

🔍

eladmin2

👤

📁 Film-Noir

📁 Adventure

👤 Family

📺 History

😄 Comedy

★ Animation

👁 Horror

🎭 Drama

👁 Sci-Fi

🚒 War

⚙ Action

MAX

Wizard of Oz, The (1939)

NO POSTER AVAILABLE

© Copyright Filmbrary.com | J.M Freire - Miguel Herrera



Cesta



Película del día



Categorías

Life Less Ordinary, A (1997)

19.0 €



Comprar Ahora

Sinopsis:

None

Puntuación:

- / 10