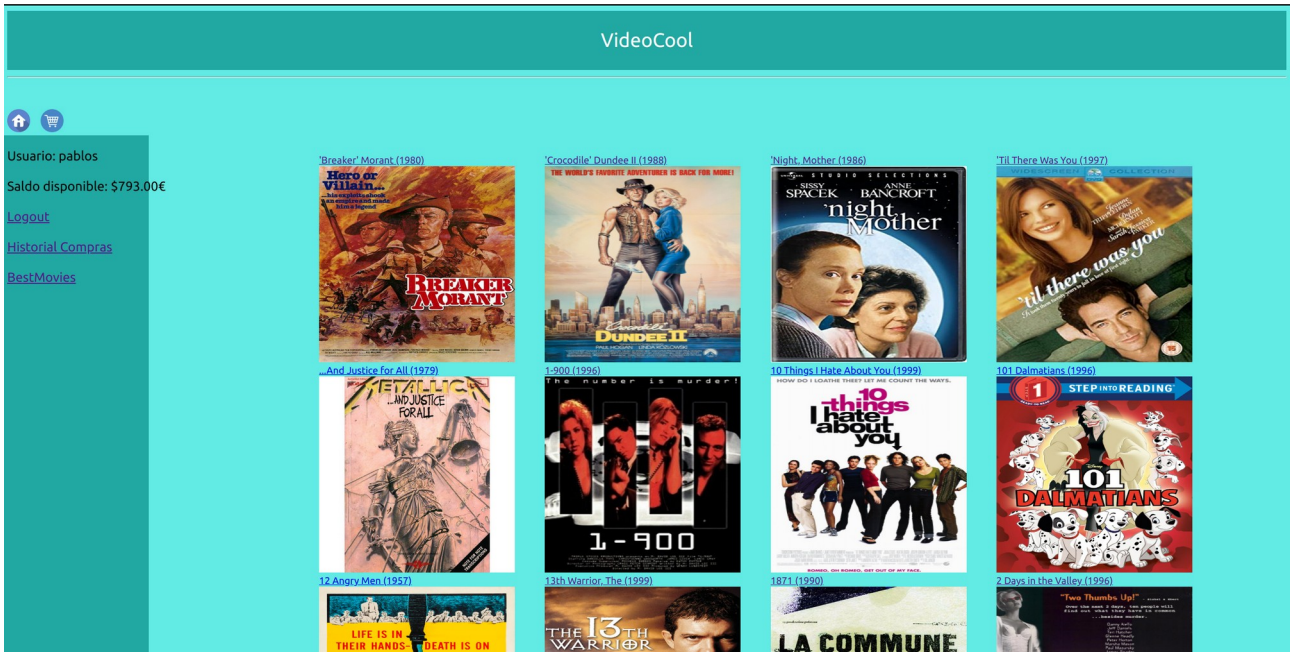
Memoria Práctica 2:

Práctica realizada por Pablo Sanchez Fernandez y Álvaro Rodríguez Rodrigo



Lo primero que hicimos para enforcar la práctica fue desarrollar el diagrama E/R ypara así tener una idea del funcionamiento de la base de datos.

Diseño de la base de datos:

E/R: Lo hicimos manualmente ya que es más sencillo y luego lo pasamos a digital para simplificar su corrección. Este diseño fue modificado durante la pratica si se iba considerando necesario

Diseño: Principalmente nos encontramos con un montón de atributos que debían de ser primarias o foraneas y no lo eran:

PK ======================>	FK
imdb_actor.actorid	imdb_actormovies.actorid
imdb_movies.movieid	imdb_actormivies.movieid
imdb_movies.movieid	products.movieid
products.prod_id	inventory.prod_id
products.prod_id	orderdetail.prod_id
customers.customerid	orders.customerid
orders.orderid	orderdetail.orderid

NUEVAS CLAVES PRIMARIAS
imdb_actormovies => (actorid, movieid)
orderdetail => (orderid, prod_id)

Estas son algunas de las relaciones que incluimos. Implementamos dichos cambios en actualiza.sql. Todos estos cambios y más están en actualiza.sql de la línea 43 a la 66.

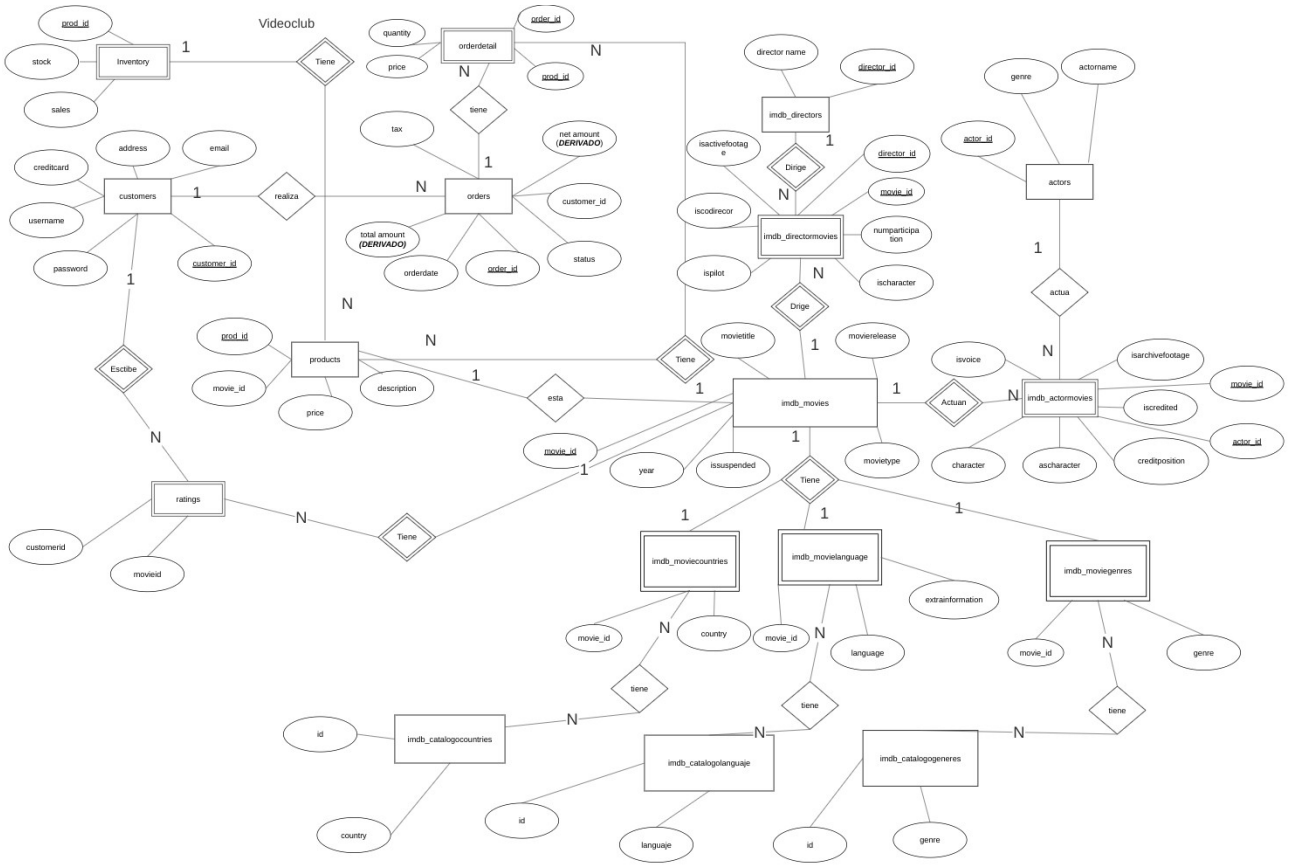
Algunas relaciones de claves fueron necesarias de rehacer ya que requería el modo DELETE ON CASCADE cuando la clave foránea esta en una entidad débil, para poder eliminar tuplas que consideramos poco prácticas o con errores y poder manejar la base de datos mas cómodamente.

Por último añadimos a actualiza.sql el resto de datos que se nos pidieron. Triggers necesarios para el correcto funcionamiento de la base de datos, el campo balance que tendrá el saldo de cada customer, la nueva entidad ratings y dos campos en imdb_movies que se encargarán de manipular las valoraciones de las peliculas.

Cambiamos el tamaño de de password basicamente porque a la hora de implementar la plataforma, la contraseña va cifrada y si no se cambia su tamaño la contraseña cifrada no entra en el campo.

También había problemas con las tablas de género, country y language de las películas, ya que las opciones que se contemplaban no estaban acotadas en un catálogo, por ello, codificamos en un catálogo cada categoría para que las opciones estuvieran acotadas.
Estos son imdbcatalogocountries, imdbcatalogogenres e imdbcatalogolanguaje.

Además creamos un procedimiento que inicializaría balance con un numero aleatorio entre 0 y N e hicimos una llamada para N=100.



Aqui se muestra el diagrama final de las entidades.

Setprice:

En esta consulta nos piden actualizar el precio “price” de orderdetail sabiendo que ha ido incrementando un 2% cada año, estando el precio inflado en la tabla products.
La consulta no es demasiado dificil, lo mas complicado es saber aplicar la formula de los porcentajes inversos en sql haciendo una llamada a year from current_date para obtener el año actual y así poder realizar el insert fácilmente en la tabla.

SetOrderAmount:

Esta consulta requiere el buen funcionamiento de la anterior, ya que sin ella, resulta imposible calcular el precio final sin tener precios asignados a cada orderdetail.

Esta consulta consiste calcular el precio sin y con impuestos de cada pedido.
El procedimiento consiste en hacer una query con tuplas agrupadas por orderid de cada pedido para posteriormente hacer el calculo de sumar todos los precios de todos los orderid, con este dato, rellenamos la columna de netamount (valor neto), y después aplicarle los impuestos e introducirlos en totalAmount.

GetTopSales:

Para getTopSales nos pedían que entre dos años dados, encontraramos las peliculas que más se han vendido en esos años. Aquí nos surgió la duda de si se refería a entre esos N años buscar todas las peliculas con más ventas y luego mostrar solo una por año o si de N años buscar por cada año la pelicula que más se vendió y mostrar esa pelicula. Finalmente implementamos la segunda acción y tuvimos también problemas ya que para las cantidades estabamos usando un count(*) y no un sum(*) por lo que a la hora de obtener el recuento total, las compras en las que se habían realizado dos compras de la misma pelicula, solo contaba como 1.
(Lo arreglamos y todo perfecto).

GetTopActors:

Fue una consulta bastante directa. Realizamos varias subconsultas, ya sea para encontrar el total de peliculas actuadas por cada actor. Luego otra para quitar los registros repetidos y para realizar la cuenta de estas peliculas. Por último mostramos todo.

Integridad de los datos:

f) Aquí al principio no entendíamos muy bien a que se refería pero tras preguntarle al profesor llegamos a la conclusión de que era crear catalogos para las tablas genres, languages y countries.

Lo que hicimos fue crear nuevas tablas de catalogos: catalogogenre, catalogolanguage y catalogomovie. A estas tablas les dimos un identificador y un id. El id sería secuencial y se añadiría de uno en uno. El catalogo no puede tener valores repetidos (unique). Por último hicimos que las tablas iniciales: imdb_movielanguage cambiaran su campo language de un char a un integer, este integer será el identificador de la tabla catalogo.

Se queda algo así:

imdb_movielanguage:		imdb_catalogolanguage	
moviei d	languages----------->id		languages
103	1	1	Drama
147	1	2	Comedy
147	2	3	Action
147	4	4	Adventure

Creacion de triggers:

UpdOrders:

Este trigger cambiarála información de orders cada vez que se añada un nuevo objeto al carrito. Es decir, como lo hemos supuesto es que cada vez que se añada, elimine o actualice un articulo de orderdetails con el orderid = orders.orderid entonces actualizaremos el valor del precio total y el precio con tax. Esto nos servirá para la página web. Ya que cada vez que se añada una pelicula se añadirá a orderdetails y automaticamente se buscará el carrito del cliente (NULL) y se modificará su precio.

UpdRatings:

Muy parecido a updorders, cada vez que se añada una nueva valoración, se buscará el movieid y se modificará el ratingmean y ratingcount para así tener una media correcta y poderla mostrar en la aplicación.

UpdInventoryAndCustomer:

Este trigger se ocupa de que cuando un pedido cambiaba de estado a "paid", se encargue de descontar los productos del pedido en su respectiva cantidad del inventario, y se proceda al cobro del cliente.

Integración en el portal:

j) La consulta que nos piden es la de getTopSales, como la consulta ya la teníamos era tan simple como obtener la consulta con un select * from gettopsales() y además buscar para cada pelicula su movieid (ya que así es como funcionan nuestros links al las peliculas) almacen<movieid> por lo cual con dos simples consultas y un pequeño array unimos ambas informaciones y se las mandamos al python. Lo que hicimos fue crear una nueva página para simplificación del css(ya que index.html estaba muy lleno de cosas).

k)Simplemente tubimos que obtener la información mediante el registro(ya validado previamente con js) e insertar en la tabla customers un valor nuevo con la nueva información del registro y la contraseña cifrada. A la hora del login, como no sabíamos como diferenciar una contraseña cifrada de las que no son, se incluira la comprobación con la contraseña cifrada y sin cifrar.

l)pagina principal:

Aquí decidimos que solo ibamos a mostrar 16 peliculas ya que no ibamos a incluir fotos para todas las peliculas. Mostramos las 16 primeras peliculas ordenadas por movieid y no dimos la posibilidad de mostrar más ya que se fastidiaría todo el css si una pelicula no tiene foto y el resto si. Para ello fue tan simple como una imagen que contenia el movieid y hacer una consulta que nos diera el movieid y otras cosas.

Pagina peliculas:

Al igual que con las 16 peliculas aqui solo se muestran fotos para las 16 peliculas aunque no hay errores si buscas el id de la pelicula a mano. (simplemente no aparece la foto).
Obtenemos toda la información que queremos mostrar(actores, directores, generos etc) mediante diferentes consultas que hacen diferentes cosas y mostramos todo mediante html.

Creamos los forms mediante html y python para controlarlos y estos forms lo que hará será hacer inserts a:

orderdetail si se añade una pelicula

ratings si se valora una pelicula.

En la parte de carrito, supusimos que como había prod_ids por ejemplo para 1 pelicula 3 prod_ids diferentes decidimos que en vez de que se pudieran añadir cantidades lo que se comprarían son prod_ids ya que es muy raro que alguien quiera comprar 20 peliculas del mismo tipo normalmente se compra 1 o 2.

Si se quieren comprar 20 se tendrían que hacer 20 clicks.

Para los ratings pones una valoracion del 1 al 5 y ya según si es insert o update el trigger saltará y hará algo en concreto.

m) El carrito fue un poco más raro por la parte de mantener parte en sesion hasta que se hace login que se mantiene en base de datos actualizando el carrito pero creo que como para está parte ya teníamos mas tacto con python y sql se nos dió bastante bien. Simplemente ibamos comprobando si session[‘usuario’] o session de carrito estaban activas y buscando o en session o en base de datos.

n)Valoraciones ya explicadas en l)