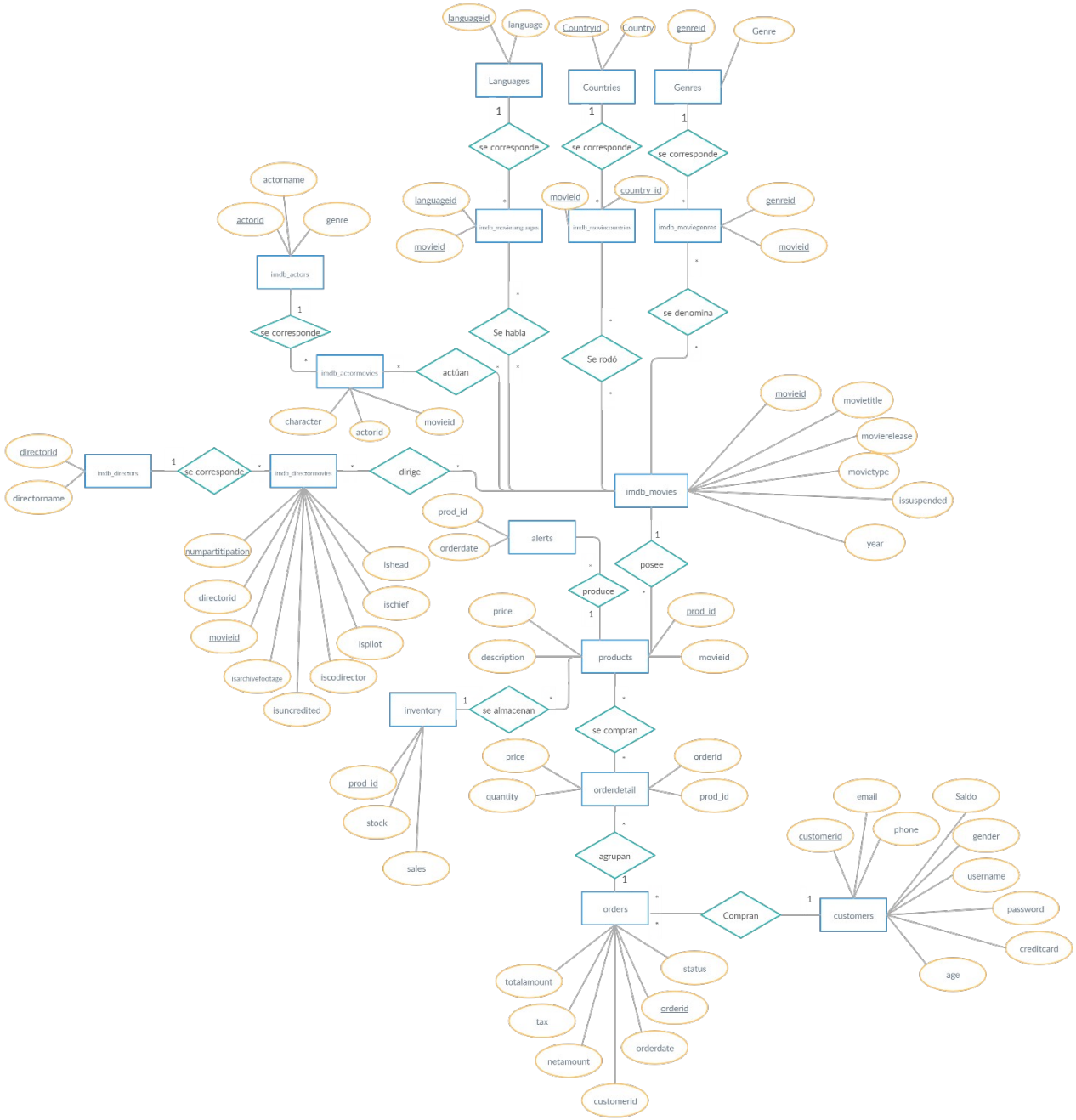
MEMORIA PRÁCTICA 3 DE SISTEMAS INFORMÁTICOS

Victoria Pelayo y Sofía Sánchez

Diagrama Entidad-Relación



Apreciamos que hemos añadido en el modelo E-R todos los cambios implementados en nuestra aplicación. Todas las entidades forman una tabla en nuestra base de datos y en todas las entidades encontramos una clave primaria y sus atributos.
Además, también hemos indicado las cardinalidades de cada relación entre entidades.

Observamos que en *imdb_movies* tenemos las películas, como tal, y se relacionan con otras entidades según el idioma que se habla, dónde se han grabado, que actores participan, quiénes la dirigen…

Las tablas *genres, languages* y *countries* las hemos creado para trabajar con id's y eliminar redundancia en *imdb_countries, imdb_languages* y *imdb_genres*

Película también se relaciona con *products*. Los productos corresponden a lo que es el producto en si, lo que comprarán los compradores, contiene una descripción del producto en si y su precio, además de los id's que le caracterizan.

Los productos al ser los que se compran serán los que se relacionen con *orderdetail*, dónde almacenamos que producto se va a comprar y cuántos.

Finalmente, las *orders* están compuestas de *orderdetails*, es decir, de productos que se quieren comprar, cada uno con un precio determinado.

En *orderdetail* guardamos el precio del producto. Y, en *orders* guardamos el precio que tendrá esa order, ya habiendo aplicado impuestos, descuentos…

Los compradores, cuándo finalicen la compra, lo que realmente están comprando es la *order*, es decir, todos los productos que la componen.

Cabe destacar la tabla *alerts*, cuya finalidad es controlar las veces que una película se ha quedado sin stock, almacenamos también la fecha de cuando esto ocurrió ya que puede proporcionar información importante, como por ejemplo que películas en enero han estado más de moda.

EXPLICACIÓN DE ACTUALIZA.SQL

- Actualizar los valores de customerid y orderid

Los id's "customerid" y "orderid" de nuestras tablas los tenemos para que se metan en orden secuencial, inicialmente hemos hecho que la secuencia, a partir de la cuál van aumentando los id's cuando se introduzcan en la tabla, sea la del máximo id presente.

-Cambios en columnas de tablas

Hemos borrado las columnas de las tablas que no hemos considerado necesarias para nuestra aplicación. De la tabla customers hemos eliminado las columnas que no podian estar a null y que no utilizamos en nuestra aplicación, estas son: address1, address3, zip, country, city, state, income, region, firstname, lastname, creditcardtype y creditcardexpiration. De la tabla imdb_actormovies hemos eliminado las columnas ascharacter, isvoice, isuncredited, isarchivefootpage y creditsposition puesto que la mayoria de ellas estaban vacias y no ibamos a necesitarlas en nuestra aplicación. De la tabla imdb_directormovies hemos eliminado las columnas participation y ascharacter y finalmente de la tabla imdb_movielanguages hemos eliminado la columna extrainformation por las mismas razones que hemos expuesto anteriormente.

- Nuevas tablas

Hemos creado tres nuevas tablas: languages, genres y countries donde se guardarán los distintos tipos de idioma, género o país con sus id's correspondientes. Por ejemplo la nueva tabla languages estara formada por *languageid* y *language*.

Para completar estas tablas hemos realizado un *INSERT* junto con un *SELECT DISTINC* del idioma, genero o pais de las tablas originales (*imdb_movielangugages, imdb_moviecountries* o

imdb_moviegenres). De esta manera tendremos 3 tablas nuevas donde los idomas, los paises y los generos no estarán repetidos.

Finalmente para completar las nuevas tablas tendremos que meter en las antiguales estos nuevos id's para posteriormente poder actualizar las nuevas relacionando los id's de ambas tablas.

Como por ejemplo lo que hacemos en esta query:

```
UPDATE imdb_movielanguages
SET languageid = l.languageid
FROM languages AS l
WHERE l.language = imdb_movielanguages.language;
```

Por ultimo eliminamos de las tablas antiguas (imdb_movielanguages, imdb_moviecountries, imdb_moviegenres) las columnas que se corresponden con el idioma, el genero y el pais puesto que ya tenemos en estas los id's. Además, como en estas tenemos los id's del idioma y de la pelicula, lo que hacemos es crear nuevas primary keys que se componen de ambos id's. Por ejemplo, en imdb_moviecountries la primary key es (movieid, countryid).

- Cambios en orderdetail

Inicialmente teníamos en esta tabla teníamos varias entradas correspondientes al mismo producto y de la misma orden. Para evitar esto hemos reagrupado todas estas entradas en una misma.
Primero hemos creado una nueva tabla (que será la definitiva) en esta hemos puesto como primary key orderid y prod_id.

Hemos reagrupado, correctamente las entradas con redundancia, sumando sus cantidades y finalmente nos quedamos con esta tabla como orderdetail.

-Alerts

Creamos una tabla que se llama alerts que tiene solo prod_id y orderdate. No tiene primary key porque, hemos supuesto que en nuestra tienda se podría reponer y volver a acabar el stock de un mismo producto todo el mismo día, y queremos que haya varias alertas para llevar cuenta de que pelis son las que más se han agotado.

- Saldo

Hemos añadido el atributo saldo en la tabla de customers para guardar el saldo actual de cada customer. Por defecto lo hemos puesto a 100 siguiendo las pautas de la anterior práctica.

EXPLICACIÓN DE LAS CONSULTAS

getTopVentas:

En esta consulta vamos a mostrar el top de Ventas por año desde el año que se introduce hasta el actual. Mostraremos una película por año, la que más éxito haya tenido.
Por defecto, en nuestra función si hay mas de una pelicula en el mismo año con el mismo numero de ventas solo se mostrara una.

getTopMonths:

Una vez implementamos getTopVentas, implementar esta nos resulto más sencillo. Lo que haremos sera comprobar que se superan los umbrales que introducimos como argumentos de entrada a la funcion que seran el numero de productos y el importe.

SetPrice:

Esta consulta la ejecutamos y comprobamos que se introducen bien en el campo price, los precios del producto. Se introduce el precio del producto, independientemente de la cantidad.

Para poder realizar la funcion matematica necesitaremos restar el año actual con el año de la pelicula y además necesitaremos el precio del producto, todo esto lo cogeremos de las tablas orders, products y orderdetail y las relacionaremos de manera que el orderid de orders y orderdetail sea el mismo y el prod_id de products y orders tambien lo sea.

SetOrderAmount:

Tras ejecutar SetPrice, comprobamos que netamount y totalamount, en la tabla orders, se han actualizado correctamente.

Para el contenido de los campos que se actualizan/inicializan con SetPrice como SetOrderAmount hemos seguido las pautas del enunciado.

updInventory y updOrders

Estos triggers son necesarios para el correcto funcionamiento del carrito. Los hemos programado, y hemos comprobado que los campos de las tablas correspondientes se actualizan.

En updInventory hacemos un bucle en el que vamos restando al stock actual de cada producto la cantidad que se ha comprado, y a las ventas de ese producto le sumamos la cantidad. En caso de que nos quedásemos sin stock añadimos una alerta en la tabla de alerts.

En updOrders distinguimos entre dos casos: INSERT y UPDATE. No barajamos la posibilidad de delete, pues nosotros los productos en el carrito los vamos eliminando de “uno en uno” por lo que siempre utilizaremos update y dentro del trigger comprobaremos si la cantidad es 0 o no.

Si es insert, totalamount y netamount los inicializamos y actualizamos la tabla orders.
Si es update:
Si la cantidad nueva es 0, eliminamos esa entrada de la tabla orderdetail
Si no, actualizamos las cantidades de netamount y totalamount de orderdetail, teniendo en cuenta los valores previos de éstas.

En general, todas las funciones/consultas/triggers realizadas en la práctica se puede comprobar su funcionamiento a través de la aplicación y viendo como varía la tabla de datos.

Mejoras:

En esta práctica no se pedían las siguientes funcionalidades, “buscar” e “historial” , pero, creemos que son necesarias para el funcionamiento completo de la aplicación. Ya que proporcionan funcionalidad necesaria en una aplicación como la que estamos desarrollando.

También, sería interesante a parte de comprar películas, que a través de la aplicación pudieses seleccionar si quieres que te las envíen a casa, recogerlas en un punto de recogida… Así como su seguimiento. Esta funcionalidad para esta práctica nos era inviable llevarla a cabo, pero, creíamos que es interesante barajar la idea.