

Memoria

P3 SI



David Pascual Hernández
Alain Messina Valverde

Grupo 1361

Descripción, funcionamiento y estructura

- Ficheros que componen la entrega:

En esta práctica, usamos el Framework Flask para desarrollar nuestra aplicación web en Python (a parte de HTML, CSS y Javascript), junto con apache mas wsgi para poder visualizarla. Además, hemos implementado una base de datos en PostgreSQL que nos permite almacenar toda la información necesaria para el correcto funcionamiento del programa. Esta base de datos puede ser editada usando alguno de los PGScripts implementados (carpeta SQL)

Todo el código se encuentra en la propia carpeta public_html. En esta, podemos encontrar todas las plantillas, routes.py, así como un archivo que hemos llamado database.py en el que hemos guardado todas las funciones que tienen que ver con la base de datos.

- Discusión de la base de datos:

Esta base de datos es el ejemplo perfecto de de BD Relacional ya que separa en tablas según los tipos de datos y para que se utilizan, y a la vez mantiene una integridad y coherencia de estos. Usando la herramienta SQL y su potencia se pueden hacer consultas en esta BD, ayudándonos de JOINS, claves externas, funciones, etc, de lo que deseamos saber, como hacemos en esta práctica.

La ventaja más importante la obtenemos al separar 'roles' o 'áreas' por darle un nombre. Por ejemplo si queremos saber los actores que han participado en películas con un director 'x', no necesitaremos consultar datos que nos son irrelevantes como productos, orders o customers, ganando así eficiencia y versatilidad.

Sin embargo la base de datos proporcionada no está del todo preparada para operar con nuestra web, y necesitamos hacerle unos cambios y añadir funcionalidades nuevos que comentamos en el siguiente apartado.

- Consultas solicitadas:

actualiza.sql

En este fichero hemos incluido todas las funciones necesarias para adaptar la base de datos inicialmente dada a una base de datos relacional que podamos manejar en la aplicación. Entre otras funcionalidades que necesitamos modificar o añadir para adaptar la base de datos a nuestras necesidades, se encuentran:

- **Añadir columna de saldo a costumers:** además de añadir una columna de tipo "numeric", le asignamos un valor DEFAULT de random()*100, lo que quiere decir, un valor aleatorio entre 0 y 100. Se usará en la web para comprar los productos.

- **Agrupar duplicados orderdetail:** en esta consulta agrupamos todos los pedidos en orderdetail que tengan mismo producto y orderid asociados, usando la columna quantity.
- **Añadir primary y foreign key a actormovies:** en esta consulta añadimos las claves primarias, constraints y claves foráneas necesarias a la tabla actormovies
- **Añadir foreign key a inventory:** como en el caso anterior, añadimos una clave foránea a la tabla inventory para relacionar el producto con inventory
- **Añadir primary y foreign key a orderdetail:** en esta consulta creamos una primary key formada por las ids de order y products combinadas. Además, añadimos las claves foráneas para las dos.
- **Añadir primary y foreign key a orders:** con esta consulta creamos la primary key de orders formada por orderid, así como la foreign key que relaciona el pedido con el customer.
- **Usuarios duplicados:** en esta consulta asignamos una concatenación de nombre+id a los usuarios cuyos nombres están duplicados, para que así este valor pueda ser UNIQUE.
- **Añadir unique key a username:** una vez eliminados los duplicados en la consulta anterior, añadimos el atributo UNIQUE a la columna username, para que no haya users repetidos
- **Añadir tabla alertas:** se usa en el trigger updInventory, para cuando el stock se quede a cero poder crear una alerta.

setPrice.sql

En esta consulta, simplemente actualizamos el precio con una operación matemática (solicitada en el enunciado) recorriendo todos los años y aumentando un 2% el precio.

setOrderAmount.sql

Esta consulta ejecuta una función (setOrderAmount.sql) que calcula el valor total de un producto en base a su precio multiplicado por su tasa.

getTopVentas.sql

Esta query recorre cada año para buscar el producto más vendido, y lo va devolviendo formando una sola tabla. Además, recibe como argumento el año de partida (por ejemplo, si recibe 2015, devolverá el producto más vendido de cada año desde 2015).

getTopMonths.sql

En este caso, getTopMonths es una función que devuelve los meses en los que se ha superado alguna de las cotas indicadas (por beneficio o por cantidad de ventas) separadas por meses y por años.

updOrders.sql

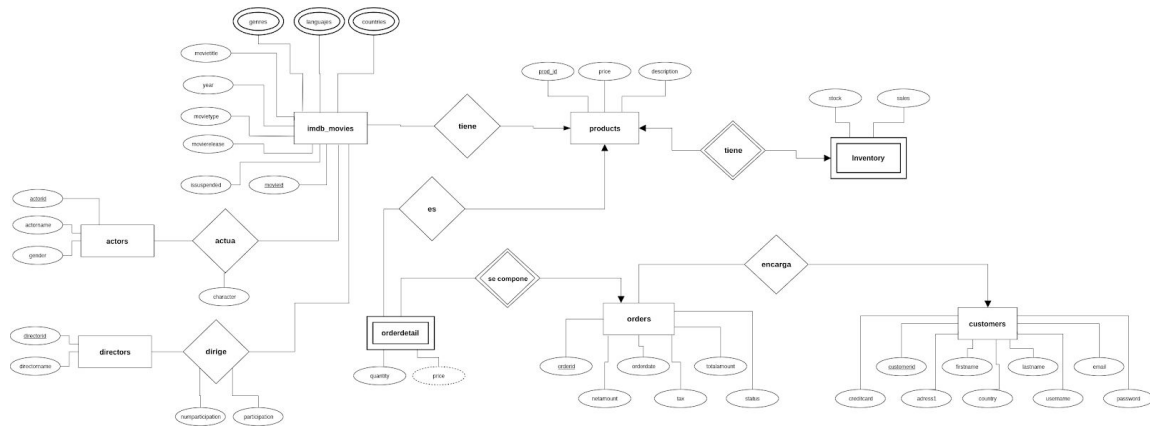
Este trigger realiza la funcionalidad solicitada, actualizando las orders.

updInventory.sql

En este trigger, se actualiza el inventario de la manera solicitada en el enunciado

- Diagrama:

(Imagen en completa resolución en la raíz del proyecto como ER.png)




Para realizar este diagrama ER, se ha hecho un proceso de ingeniería reversa, ya que normalmente que hace al revés. Caben destacar los siguientes factores:

- Genres, lenguajes y countries son atributos multivaluados, y desde el punto de vista ER es lo que serán y no una tabla.
- Las tablas Inventory y Alertas tiene una relación débil con productos, aunque algo fuera de lo común ya que esta relación es de uno a uno, no de 1 a N que es lo habitual. Podríamos decir que tienen la relación 1 a N característica con movies a través de productos.
- Movies tiene una relación muchos a muchos con actores y directores, ya que, por ejemplo muchos actores pueden participar en una película y a la vez uno de esos actores participa también en muchas otras películas. Estas relaciones también tienen sus atributos como podemos ver el diagrama, por ejemplo 'character'.
- Orderdetail tiene una relación débil con orders, esta vez si, de 1 a N, puesto que 1 order se puede componer de varios orderdetail, pero 1 orderdetail solo es de un order.

- Evidencias de los resultados obtenidos:

Como podemos ver, la base de datos y sus consultas funcionan correctamente. En el caso del inicio, por ejemplo, devuelve correctamente una lista ordenada del top de películas de los últimos 3 años.



Inicio

Usuario: 12345
Saldo: 4€
Cerrar sesión
Carrito

Filtrar:
Filtrar por
Buscar


Buscar por:
Escribe aquí
Buscar

Usuarios conectados:
175

Top películas últimos 3 años

Ranking	Título	Edición	Precio	Ventas	
1	'Breaker' Morant (1980)	Standard	13€	188	Añadir al carrito
2	Shuang long hui (1992)	Standard	19€	118	Añadir al carrito
3	Men in Black (1997)	Standard	17€	117	Añadir al carrito
4	Lonely Are the Brave (1962)	Standard	17€	117	Añadir al carrito
5	Jefferson in Paris (1995)	Standard	11€	115	Añadir al carrito
6	Mark of Zorro, The (1940)	Standard	16€	114	Añadir al carrito
7	Planète sauvage, La (1973)	Ultra	27€	112	Añadir al carrito
8	Tom & Viv (1994)	Standard	10€	111	Añadir al carrito
9	Second Jungle Book: Mowgli & Baloo, The (1997)	Standard	12€	110	Añadir al carrito
10	Rushmore (1998)	Gold	21.6€	109	Añadir al carrito
11	Drive Me Crazy (1999)	Standard	15€	108	Añadir al carrito
12	Carmen Miranda: Bananas Is My Business (1995)	Standard	16€	108	Añadir al carrito
13	One Magic Christmas (1985)	Standard	14€	107	Añadir al carrito
14	Three to Tango (1999)	Standard	10€	107	Añadir al carrito
15	Dracula (1931)	Gold	16.8€	106	Añadir al carrito
16	Last Resort (1994)	Standard	13€	106	Añadir al carrito
17	Bug's Life, A (1998)	Standard	14€	106	Añadir al carrito

También podemos ver cómo es correcta la funcionalidad del carrito, así como la del login.



Carrito

Sesión
Carrito

Filtrar:
Filtrar por
Buscar

Buscar por:
Escribe aquí
Buscar

Usuarios conectados:
177

Título	Cantidad	Precio	Borrar
Tom & Viv (1994)	1	10€	Borrar este elemento

Total:
10€

Comprar carrito
Limpiar la cesta

David y Alain ®

- Flujo ejecución:

createdb -U alumnodb si1

gunzip -c dump_v1.3.sql.gz | psql -U alumnodb si1

psql -h localhost -d si1 -U alumnodb -f actualiza.sql

psql -h localhost -d si1 -U alumnodb -f setPrice.sql

psql -h localhost -d si1 -U alumnodb -f setOrderAmount.sql

psql -h localhost -d si1 -U alumnodb -f getTopMonths.sql

psql -h localhost -d si1 -U alumnodb -f getTopVentas.sql

psql -h localhost -d si1 -U alumnodb -f updOrders.sql

psql -h localhost -d si1 -U alumnodb -f updInventory.sql

Referencias

- Toda la bibliografía incluida en el enunciado.
- Tutorial Core SQLAlchemy. Enlace:
<https://www.tutorialspoint.com/sqlalchemy/index.htm>