Práctica 2

Sistemas

Informáticos

Giancarlo Lazo  
Pablo Izaguirre  
Grupo 1321

Contenido  
1. Introducción  
2. Diagrama de E-R  
3. Consultas, triggers y funciones postgresql

4. Mejoras

1. Introducción

Nos hemos centrado en la implementación de la base de datos (en PostgreSQL y con ayuda de phppgadmin) que dará soporte al backend diseñado en la práctica 1 y donde se guardarán los datos de productos, usuarios y compras.

2. Diagrama de E-R

3. Consultas, triggers y funciones postgresql

Ficheros:

- actualiza.sql: fichero que actualiza la base de datos proporcionada.

Explicado en el apartado mejoras

- setPrice.sql: fichero que completa la columna “price”.

Se hace un join de products, imdb\_movies, orders para luego sacar el price.

- setOrderAmount.sql: fichero que completa las columnas netamount y totalamount.

Se han actualizado netamount y totalamount. Netamount crea una tabla auxiliar S, para sacar la suma de los precios totales agrupados por el orderid. Después, con la tabla actualizada, igualas el totalamount (netamount más impuestos) con otra tabla auxiliar T.

- getTopSales.sql: función que recibe dos argumentos, para recibir las peliculas que tuvieron las mejores ventas por año.

Se han usado dos tablas una que se llama AUX, con todas las id de las peliculas, el año que le corresponde, venta por año, y su prod\_id. AUX2 es igual a AUX, pero sin prod\_id.

MaxSalesPerYearTable usa AUX2 teniendo como resultado el año y la venta máxima ese año.

Se le hace un inner join con AUX para recibir el movieid de las ventas máximas. Se intento devolver la salida con los out que decía la práctica, pero al usar select into y las variables out, solo retornaba la fila de más arriba y no todas las peliculas entre los dos años.

- getTopActors.sql: recibe un parametro “género” y devuelve los actores o actrices que más veces han actuado en dicho género.

- updOrders.sql: actualiza la tabla de orders.

- updRatings.sql: actualiza la información de la tabla 'imdb\_movies'  
cuando se añada, actualice o elimine una valoración, modificando los campos ratingmean y ratingcount.

- updInventoryAndCustomer.sql: actualiza la tabla inventory (products en nuestro caso) y descuenta el balance de un costumer id con el precio de su compra correspondiente.

Se ha declarado el rec como la union del old (orders), products y orderdetail para luego, con esa tabla actualizar las tablas products y customer. A productos se modifica su stock y sales siempre que prod\_id sea igual a rec.prod\_id, y a customer se le resta el balance de la compra siempre que su id sea igual al customer id de la tabla rec donde cada customer id tiene un producto que ha comprado dicho customer.

4. Mejoras

Se ha añadido la tabla ratings, como lo pedía la práctica, junto con los atributos customerid, movieid y rating; su clave principal es customerid junto a movieid.

Se han agregado las columnas en imdb\_movie, como ratingmean y ratingcount y también se cambió el atributo password en customers para que soporte 96 caracteres.

En customers, el campo email es único.

Se ha visto que en orderdetail se ve que no tenía clave primaria y como había dos pares que se repetían (orderid y prod\_id) se le asigno como clave primaria.

Había dos tablas antes del actualiza (inventory y products) que las terminamos uniendo debido a que ambas se referían a lo mismo, un conjunto de productos con su precio, stock y tal, por lo que es más sencillo tenerlo en una tabla.

Se han creado las tablas genres, languages y countries para garantizar la integridad de datos, como se pedía en la práctica; también se ha añadido el campo balance a la tabla customers que sea no null y por defecto tenga el valor devuelto por la función, que está en el actualiza.sql, dentro de un rango de 0 a N=100.

En el campo orders se le ha puesto por defecto tax = 15.

El orden de ejecución debe ser igual al especificado en el enunciado, en caso contrario puede que no salgan los datos correctamente, ocurran errores y la salida puede que no sea la esperada.