Universidad Autónoma

Prácticas de Sistemas Informáticos I

Práctica 3

Memoria

Autores:
Adrián Fernández
Santiago González-Carvajal

Pareja 11
Grupo 1401

25 de noviembre de 2018

Índice

1. Modelo E-R resultante

1.1. Diagrama

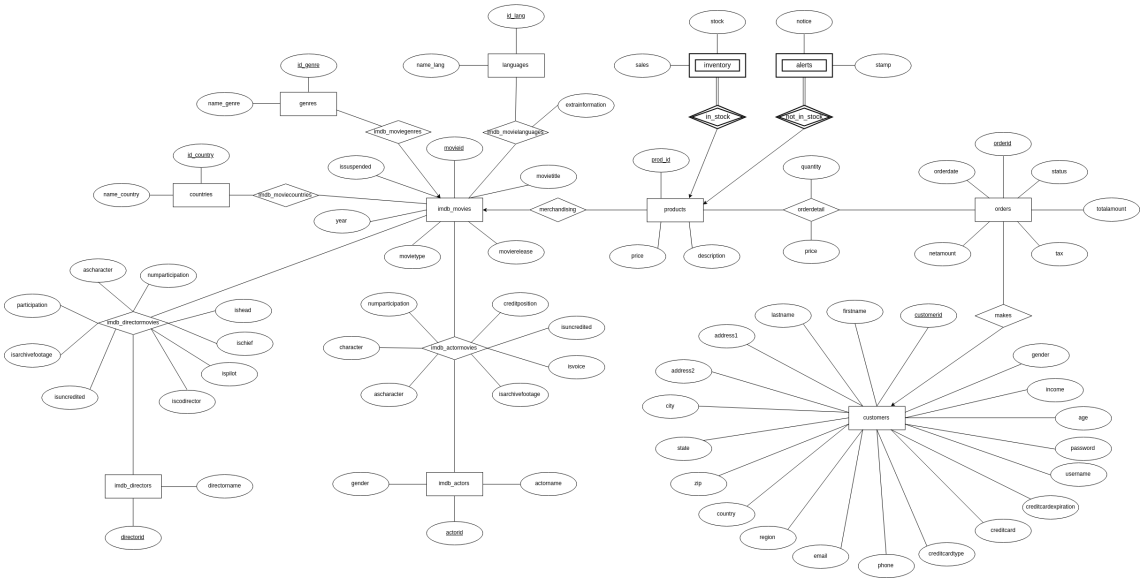

Figura 1: Diagrama E-R (Adjunto en la entrega para mejor visualización).

1.2. Cambios introducidos

Alertas

Se han implementado como una entidad débil de productos. Así, cuando salte una excepción por falta de stock, se generará una alerta en la tabla de alertas para ese producto en concreto.

Lenguajes

Se han implementado como una entidad diferente y se ha establecido una relación "many to many" entre películas y lenguajes.

Países

Se han implementado como una entidad diferente y se ha establecido una relación "many to many" entre películas y países.

Categorías

Se han implementado como una entidad diferente y se ha establecido una relación "many to many" entre películas y categorías.

Restricciones

Se han añadido restricciones entre las entidades relacionasdas para garantizar la consistencia de datos.

Actualizacion de precios

Se realiza ejecutando el codigo del script "setOrderAmount.sql", el cual calcula el precio de los pedidos a partir del precio de los productos.

2. Soluciones de las consultas solicitadas

2.1. setPrice.sql

Actualizamos la columna price de orderdetail con el valor obtenido de dividir el precio del producto entre 1.02 elevado al número de años transcurridos.

2.2. setOrderAmount.sql

Actualizamos el valor de netamount y totalamount de orders cuando netamount es NULL. Para obtener estos valores realizamos una consulta auxiliar, donde obtenemos la suma de los precios de cada producto de los pedidos. Una vez calculada la suma, actualizamos el valor de netamount con la suma y el valor de totalamount con la suma más impuestos.

2.3. getTopVentas.sql

Primero calculamos el número de ventas de cada película cada año. De esa consulta, seleccionamos el máximo en ventas por año mediante una "PARTITION" y mostramos las tuplas que también cumplan que el año es igual o superior al pedido.

2.4. getTopMonths.sql

Primero seleccionamos el año, mes, totalamount y suma de articulos vendidos por cada order. Tras esto, sumamos el totalamount y los articulos agrupando por año y mes. Finalmente mostramos los que superen al menos uno de los umbrales introducidos.

2.5. updOrders.sql

Cuando se inserte o borre un registro en orderdetail, se ejecutará una función sql. Esta función distingue si se ejecuta por insertar o por borrar. Si se ha insertado un registro, suma el precio del nuevo producto (por su cantidad) a netamount y recalcula el totalamount con el nuevo netamount. Si se ha borrado un registro, resta el precio del producto eliminado (por su cantidad) a netamount y recalcula el totalamount con el nuevo netamount.

2.6. updInventory.sql

Primero realizamos un trigger auxiliar que lanza una excepción y genera una alerta cuando se intenta modificar el stock de un producto a una cantidad negativa. Después implementamos otro trigger que detecta cuando se actualiza la tabla orders y, si se intenta mofificar el status a "paid", actualice el valor stock y sales de cada

artículo de ese order. Este trigger captura la excepción del auxiliar y, si no queda stock de uno de los artículos del order, cancela la modificación del status.

3. Evidencias obtenidas

3.1. setPrice.sql

	orderid integer	prod_id integer	price numeric	quantity integer
1	1043	4003		1
2	1050	466		1
3	1050	2480		1
4	1050	3140		1
5	1050	4571		1
6	1050	188		1
7	1050	6346		1
8	1050	3672		1
9	1050	3730		1
10	1050	1009		1
11	1050	899		1
12	1057	5916		1
13	1057	5312		1
14	1057	4381		1
15	1057	4519		1
16	1057	5530		1
17	1057	187		1
18	1057	4692		1
19	1057	449		1
20	1065	4890		1

Figura 2: Orderdetail antes de la llamada a setPrice.

	orderid integer	prod_id integer	price numeric	quantity integer
1	1043	4003	11	1
2	1050	466	9	1
3	1050	2480	10	1
4	1050	3140	12	1
5	1050	4571	8	1
6	1050	188	9	1
7	1050	6346	10	1
8	1050	3672	5	1
9	1050	3730	6	1
10	1050	1009	7	1
11	1050	899	5	1
12	1057	5916	6	1
13	1057	5312	11	1
14	1057	4381	7	1
15	1057	4519	12	1
16	1057	5530	7	1
17	1057	187	7	1
18	1057	4692	12	1
19	1057	449	9	1
20	1065	4890	12	1

Figura 3: Orderdetail después de la llamada a setPrice.

3.2. setOrderAmount.sql

	orderid [PK] serial	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)
1	1	2016-07-21	693		15		Shipped
2	2	2017-01-27	693		15		Shipped
3	3	2017-09-11	693		18		Paid
4	4	2016-08-18	693		15		Shipped
5	5	2015-09-01	693		15		Shipped
6	6	2015-03-05	693		15		Shipped
7	7	2018-01-26	851		18		Shipped
8	8	2016-11-04	851		15		Processed
9	9	2014-04-14	851		15		Shipped
10	10	2013-07-25	851		15		Shipped
11	11	2013-08-30	851		15		Shipped
12	12	2013-06-18	851		15		Shipped
13	13	2015-05-11	851		15		Shipped
14	14	2017-07-21	851		15		Shipped
15	15	2018-03-10	851		18		Shipped
16	16	2017-08-10	851		15		Shipped
17	17	2015-02-12	851		15		Shipped
18	18	2015-09-05	851		15		Shipped
19	19	2013-04-11	851		15		Shipped
20	20	2016-10-17	851		15		Processed

Figura 4: Orders antes de la llamada a setOrderAmount.

	orderid [PK] serial	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)
1	1	2016-07-21	693	33	15	38	Shipped
2	2	2017-01-27	693	30	15	35	Shipped
3	3	2017-09-11	693	161.6	18	191	Paid
4	4	2016-08-18	693	137.9	15	159	Shipped
5	5	2015-09-01	693	92	15	106	Shipped
6	6	2015-03-05	693	65.1	15	75	Shipped
7	7	2018-01-26	851	12	18	14	Shipped
8	8	2016-11-04	851	142.0	15	163	Processed
9	9	2014-04-14	851	143.1	15	165	Shipped
10	10	2013-07-25	851	56.8	15	65	Shipped
11	11	2013-08-30	851	79.2	15	91	Shipped
12	12	2013-06-18	851	73	15	84	Shipped
13	13	2015-05-11	851	148.7	15	171	Shipped
14	14	2017-07-21	851	142	15	163	Shipped
15	15	2018-03-10	851	11	18	13	Shipped
16	16	2017-08-10	851	109.2	15	126	Shipped
17	17	2015-02-12	851	125.3	15	144	Shipped
18	18	2015-09-05	851	159.0	15	183	Shipped
19	19	2013-04-11	851	108.8	15	125	Shipped
20	20	2016-10-17	851	39	15	45	Processed

Figura 5: Orders después de la llamada a setOrderAmount.

3.3. getTopVentas.sql

	year double precision	film character varying	sales bigint
1	2016	Wizard of Oz, The (1939)	134
2	2017	Life Less Ordinary, A (1997)	134
3	2018	Life with Mikey (1993)	57

Figura 6: Resultado de getTopVentas.

3.4. getTopMonths.sql

	year double precision	month double precision	price numeric	products numeric
1	2014	1	322148	18867
2	2014	7	322640	18760
3	2014	9	325201	19133
4	2014	10	326010	19086
5	2014	12	322869	18789
6	2015	3	320311	18605
7	2015	5	325774	19001
8	2015	10	320822	18698
9	2015	12	320316	18511
10	2016	3	322644	18683
11	2016	7	330705	19280
12	2017	9	321271	18387
13	2017	10	326181	18576

Figura 7: Resultado de getTopMonths.

3.5. updOrders.sql

Código de prueba

```
-- Insertamos un articulo en un pedido
INSERT INTO orderdetail (orderid, prod_id, price, quantity)
VALUES (1, 1, 2, 2);

-- Eliminamos el articulo del pedido
DELETE FROM orderdetail
WHERE orderid=1 AND prod_id=1 AND price=2 AND quantity=2;
```

Resultados

	orderid [PK] serial	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)
1	1	2016-07-21	693	33	15	38	Shipped
2	2	2017-01-27	693	30	15	35	Shipped
3	3	2017-09-11	693	161.6	18	191	Paid
4	4	2016-08-18	693	137.9	15	159	Shipped
5	5	2015-09-01	693	92	15	106	Shipped
6	6	2015-03-05	693	65.1	15	75	Shipped
7	7	2018-01-26	851	12	18	14	Shipped
8	8	2016-11-04	851	142.0	15	163	Processed
9	9	2014-04-14	851	143.1	15	165	Shipped
10	10	2013-07-25	851	56.8	15	65	Shipped
11	11	2013-08-30	851	79.2	15	91	Shipped
12	12	2013-06-18	851	73	15	84	Shipped
13	13	2015-05-11	851	148.7	15	171	Shipped
14	14	2017-07-21	851	142	15	163	Shipped
15	15	2018-03-10	851	11	18	13	Shipped
16	16	2017-08-10	851	109.2	15	126	Shipped
17	17	2015-02-12	851	125.3	15	144	Shipped
18	18	2015-09-05	851	159.0	15	183	Shipped
19	19	2013-04-11	851	108.8	15	125	Shipped
20	20	2016-10-17	851	39	15	45	Processed

Figura 8: Orderdetail antes de insertar un artículo en el pedido con id = 1.

	orderid [PK] serial	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)
1	1	2016-07-21	693	37	15	43	Shipped
2	2	2017-01-27	693	30	15	35	Shipped
3	3	2017-09-11	693	161.6	18	191	Paid
4	4	2016-08-18	693	137.9	15	159	Shipped
5	5	2015-09-01	693	92	15	106	Shipped
6	6	2015-03-05	693	65.1	15	75	Shipped
7	7	2018-01-26	851	12	18	14	Shipped
8	8	2016-11-04	851	142.0	15	163	Processed
9	9	2014-04-14	851	143.1	15	165	Shipped
10	10	2013-07-25	851	56.8	15	65	Shipped
11	11	2013-08-30	851	79.2	15	91	Shipped
12	12	2013-06-18	851	73	15	84	Shipped
13	13	2015-05-11	851	148.7	15	171	Shipped
14	14	2017-07-21	851	142	15	163	Shipped
15	15	2018-03-10	851	11	18	13	Shipped
16	16	2017-08-10	851	109.2	15	126	Shipped
17	17	2015-02-12	851	125.3	15	144	Shipped
18	18	2015-09-05	851	159.0	15	183	Shipped
19	19	2013-04-11	851	108.8	15	125	Shipped
20	20	2016-10-17	851	39	15	45	Processed

Figura 9: Orderdetail después de insertar el artículo.

	orderid [PK] serial	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)
1	1	2016-07-21	693	33	15	38	Shipped
2	2	2017-01-27	693	30	15	35	Shipped
3	3	2017-09-11	693	161.6	18	191	Paid
4	4	2016-08-18	693	137.9	15	159	Shipped
5	5	2015-09-01	693	92	15	106	Shipped
6	6	2015-03-05	693	65.1	15	75	Shipped
7	7	2018-01-26	851	12	18	14	Shipped
8	8	2016-11-04	851	142.0	15	163	Processed
9	9	2014-04-14	851	143.1	15	165	Shipped
10	10	2013-07-25	851	56.8	15	65	Shipped
11	11	2013-08-30	851	79.2	15	91	Shipped
12	12	2013-06-18	851	73	15	84	Shipped
13	13	2015-05-11	851	148.7	15	171	Shipped
14	14	2017-07-21	851	142	15	163	Shipped
15	15	2018-03-10	851	11	18	13	Shipped
16	16	2017-08-10	851	109.2	15	126	Shipped
17	17	2015-02-12	851	125.3	15	144	Shipped
18	18	2015-09-05	851	159.0	15	183	Shipped
19	19	2013-04-11	851	108.8	15	125	Shipped
20	20	2016-10-17	851	39	15	45	Processed

Figura 10: Orderdetail después de borrar el artículo.

3.6. updInventory.sql

Código de prueba

```
-- Realizamos una compra correctamente
UPDATE orders
SET status='Paid'
WHERE orders.orderid=88699;

-- Intentamos comprar una articulo con stock insuficiente.
INSERT INTO orderdetail (orderid, prod_id, price, quantity)
VALUES(147769, 220, 10, 2);

UPDATE orders
SET status='Paid'
WHERE orders.orderid=147769;
```

Resultados

	orderid integer	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)	orderid integer	prod_id integer	price numeric	quantity integer	prod_id integer	stock integer	sales integer
1	88699	2018-10-29	6836	144.3	21	175		88699	4293	12	1	4293	453	163
2	88699	2018-10-29	6836	144.3	21	175		88699	3908	6	1	3908	202	172
3	88699	2018-10-29	6836	144.3	21	175		88699	6304	10	1	6304	206	164
4	88699	2018-10-29	6836	144.3	21	175		88699	1739	16	1	1739	422	168
5	88699	2018-10-29	6836	144.3	21	175		88699	4702	7	1	4702	997	170
6	88699	2018-10-29	6836	144.3	21	175		88699	6608	6	1	6608	249	143
7	88699	2018-10-29	6836	144.3	21	175		88699	3123	12	1	3123	412	163
8	88699	2018-10-29	6836	144.3	21	175		88699	3745	6	4	3745	983	199
9	88699	2018-10-29	6836	144.3	21	175		88699	6282	9	1	6282	656	171

Figura 11: Datos del pedido con id = 88699 antes de comprar.

	orderid integer	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)	orderid integer	prod_id integer	price numeric	quantity integer	prod_id integer	stock integer	sales integer
1	88699	2018-10-29	6836	144.3	21	175	Paid	88699	4293	12	1	4293	452	164
2	88699	2018-10-29	6836	144.3	21	175	Paid	88699	3908	6	1	3908	201	173
3	88699	2018-10-29	6836	144.3	21	175	Paid	88699	6304	10	1	6304	205	165
4	88699	2018-10-29	6836	144.3	21	175	Paid	88699	1739	16	1	1739	421	169
5	88699	2018-10-29	6836	144.3	21	175	Paid	88699	4702	7	1	4702	996	171
6	88699	2018-10-29	6836	144.3	21	175	Paid	88699	6608	6	1	6608	248	144
7	88699	2018-10-29	6836	144.3	21	175	Paid	88699	3123	12	1	3123	411	164
8	88699	2018-10-29	6836	144.3	21	175	Paid	88699	3745	6	4	3745	979	203
9	88699	2018-10-29	6836	144.3	21	175	Paid	88699	6282	9	1	6282	655	172

Figura 12: Datos del pedido con id = 88699 después de comprar.

	orderid integer	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)	orderid integer	prod_id integer	price numeric	quantity integer	prod_id integer	stock integer	sales integer
1	147769	2018-10-24	11460	112.7	21	136		147769	220	10	2	220	1	161
2	147769	2018-10-24	11460	112.7	21	136		147769	578	12	1	578	443	176
3	147769	2018-10-24	11460	112.7	21	136		147769	4524	7	1	4524	557	167
4	147769	2018-10-24	11460	112.7	21	136		147769	414	4	1	414	585	156
5	147769	2018-10-24	11460	112.7	21	136		147769	6413	9	1	6413	310	160

Figura 13: Datos del pedido con id = 147769 antes de comprar.

	orderid integer	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)	orderid integer	prod_id integer	price numeric	quantity integer	prod_id integer	stock integer	sales integer
1	147769	2018-10-24	11460	112.7	21	136		147769	220	10	2	220	1	161
2	147769	2018-10-24	11460	112.7	21	136		147769	578	12	1	578	443	176
3	147769	2018-10-24	11460	112.7	21	136		147769	4524	7	1	4524	557	167
4	147769	2018-10-24	11460	112.7	21	136		147769	414	4	1	414	585	156
5	147769	2018-10-24	11460	112.7	21	136		147769	6413	9	1	6413	310	160

Figura 14: Datos del pedido con id = 147769 después de comprar.

	prod_id integer	notice character varying(64)	stamp timestamp without time zone
1	6413	Stock insuficiente. Imposible realizar la compra.	2018-11-25 20:52:32.220192

Figura 15: Alerta generada.

3.7.

Figura 16: El index con todas las películas de la BBDD (para todas hemos usado la misma imagen), además de la barra lateral (resultado de **getTopVentas** para los últimos 3 años).

Figura 17: Carrito con una película más a comprar (previamente hemos comprado otra como veremos en el historial de compras).

Figura 18: Historial de compras del usuario (en el que aparecen además las dos películas nuevas que hemos comprado).

Figura 19: Detalles de la película donde hemos añadido el nombre del autor (no hemos añadido todo el reparto por motivos estéticos, además de porque no se pide).

El resto de capturas de la página web serían iguales a las de la práctica anterior, ya que estéticamente no hemos cambiado nada más que lo que aparece reflejado en estas capturas. Sin embargo, ha cambiado la forma de gestionar todos los datos ya que en esta práctica gestionamos todo por medio de una base de datos en **PostgreSQL**.

4. Conclusión y propuestas

En esta práctica nos hemos limitado a cumplir con la funcionalidad solicitada. Sin embargo, se puede mejorar considerablemente la funcionalidad y el aspecto de la aplicación.
Una de nuestras propuestas para mejorar la funcionalidad sería crear una nueva entidad llamada "creditcard" separada de "customers", así se podría implementar la posibilidad de pagar con diferentes tarjetas de crédito.
Otra sería añadir un sistema de puntuación, creando una relación "many to many" entre "imdb_movies" y "customers" con un atributo entero que represente una valoración.

Referencias

[1] "Documentación de postgresql." `https://www.postgresql.org/docs/9.3/static/`.

[2] "Documentación de comandos postgresql." `https://www.postgresql.org/docs/9.3/static/sql-commands.html`.

[3] "Acceso a bases de datos en python usando sqlalchemy." `https://www.sqlalchemy.org/`.

[4] "Tutorial de sql w3schools." `https://www.w3schools.com/sql/default.asp`.

Además de todas las referencias anteriores, también hemos usado como referencia las diapositivas de teoría y algunos otros sitios web para algunas consultas no destacables.