Iteración 2: SuperAndes

Andrés Felipe Hernández León, Jenifer Paola Rodríguez Villamizar
Reporte técnico Iteración 2
Universidad de los Andes, Bogotá, Colombia
{af.hernandezl, jp.rodriguezv}@uniandes.edu.co
Fecha de presentación: Noviembre 4 de 2018

Tabla de contenido

1 Introducción

En el siguiente documento se evidenciará el desarrollo de la segunda iteración del curso. Para cada parte de la actividad, se mostrará el trabajo logrado según lo requerido en el documento del proyecto. Adicionalmente, se podrán encontrar anexos a este archivo archivos *.sql* con las sentencias de los requerimientos en texto plano y el proyecto java.

2 Análisis

2.1 Modelo conceptual SuperAndes

En la figura 1 se muestra el modelo conceptual completo correspondiente al caso de estudio “SuperAndes”. Este modelo sufrió varios cambios para esta iteración que se mencionarán más adelante. Si no se puede observar bien la imagen, también esta anexada como se pedía en la carpeta data de nuestro proyecto en java.

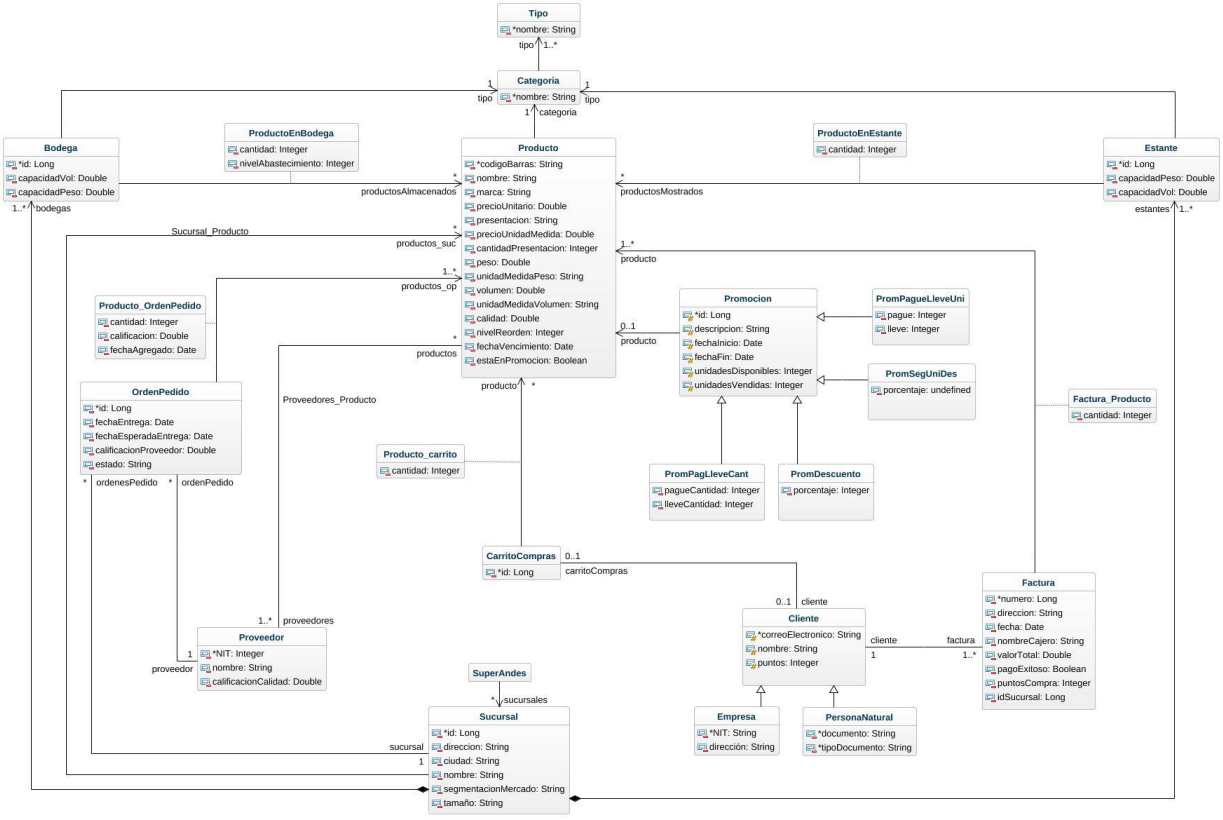

Figura 1. Modelo UML SuperAndes

En primer lugar, el modelo del mundo surge de la combinación de nuestras dos iteraciones, por tanto, hay cosas de ambas iteraciones. Se tomó como base el modelo del mundo de la iteración de Andrés y se complementó con el de Jenifer, así, el principal de los cambios fue en las promociones, donde se plantea una clase de la que hereda cada tipo de promoción para manejar correctamente los distintos tipos de promociones. Por otro lado, se modelo el administrador de la aplicación, quien conoce las distintas Sucursales del programa.

En cambios algo más pequeños, se asignó un id generado por el sistema para las sucursales como identificador único como se sugirió en la sustentación anterior (antes solía ser una llave compuesta entre dirección y ciudad de la sucursal). Siguiendo las otras sugerencias, se borró el vínculo entre el producto y la promoción, el producto ahora no conoce la promoción, pero si tiene un estado para saber si se encuentra en promoción o no y la cardinalidad se invirtió para que las promociones conozcan a sus productos.

3 Diseño de la aplicación

3.1 Modelo de datos relacional

Se presenta el modelo de datos relacional según los estándares del curso. También esta adjunto en un Excel en la documentación del proyecto por si se desea explorar un poco más fácilmente (el Excel se encuentra en “data/Iteracion2/ModeloRelacional”).

CATEGORIA
VARCHAR2(100BYTE)
nombre
PK, UA

TIPO
VARCHAR2(100BYTE)
nombre
PK, UA

TIPO_PRODUCTO	
VARCHAR2(100 BYTE)	VARCHAR2(100 BYTE)
codigoBarrasProducto	nombreTipo
PK, FK_PRODUCTO.codigoBarras, UA	PK, FK_TIPO.nombre, UA

CARRITOCOMPRAS		
NUMBER	VARCHAR2(255 BYTE)	NUMBER
id	cliente	sucursal
PK, SA	FK_CLIENTE.correo, UA	FK_SUCURSAL.id, UA, NN, NC

PRODUCTO_CARRITO		
NUMBER	NUMBER	VARCHAR2(13BYTE)
carrito	cantidad	codigoBarrasProducto
PK, FK.CARRITOCOMPRAS.id, SA	NN, UA	PK, FK.PRODUCTO.codigoBarras, SA

BODEGA				
NUMBER	NUMBER	NUMBER	VARCHAR2(100 BYTE)	NUMBER
id	capacidadVol	capacidadPeso	tipo	idSucursal
PK, SA	NN, UA	NN, UA	FK_CATEGORIA.NOMBRE, NN, UA	FK_SUCURSAL.id, NN, UA, NC

ESTANTE				
NUMBER	NUMBER	NUMBER	VARCHAR2(100 BYTE)	NUMBER
id	capacidadVol	capacidadPeso	tipo	idSucursal
PK, SA	NN, UA	NN, UA	FK_CATEGORIA.NOMBRE, NN, UA	FK_SUCURSAL.id, NN, UA, NC

SUCURSAL					
NUMBER	VARCHAR2(255 BYTE)	VARCHAR2(150 BYTE)	VARCHAR2(200 BYTE)	VARCHAR2(200 BYTE)	NUMBER
id	direccion	ciudad	nombre	segmentacionMercado	tamanio
PK, UA	NN, ND, UA	NN, UA	NN, ND, NC, UA	NN, UA	NN, UA, NC

PRODUCTOENBODEGA			
NUMBER	NUMBER(*)	NUMBER(*)	VARCHAR2(13BYTE)
idBodega	cantidad	nivelAbastecimiento	codigoBarrasProducto
PK, FK.BODEGA.id, SA	NN, UA	NN, NC, UA	PK, FK_PRODUCTO.codigoBarras, SA

PRODUCTOENESTANTE		
NUMBER	NUMBER(*)	VARCHAR2(13BYTE)
idEstante	cantidad	codigoBarrasProducto
PK, FK.ESTANTE.id, SA	NN, UA	PK, FK.PRODUCTO.codigoBarras, SA

SUCURSAL_PRODUCTO	
NUMBER	VARCHAR2(13 BYTE)
idSucursal	codigoBarrasProducto
FK_SUCURSAL.id, NN, UA, NC	PK, FK_PRODUCTO.codigoBarras, SA

PERSONANATURAL	
VARCHAR2(100BYTE)	VARCHAR2(100BYTE)
documento	tipoDocumento
PK, UA	NN, UA, CK

EMPRESA	
VARCHAR2(100BYTE)	VARCHAR2(255BYTE)
nit	direccion
PK, UA	NN, ND, UA

CLIENTE				
VARCHAR2(255 BYTE)	VARCHAR2(200 BYTE)	NUMBER(*)	VARCHAR2(100 BYTE)	VARCHAR2(100 BYTE)
correoElectronico	nombre	puntos	empresa	documentoPN
PK, UA	NN, NC, UA	NN, UA	FK_EMPRESA.nit, ND, NC, UA	FK_PERSONANATURAL.documento, ND, NC, UA

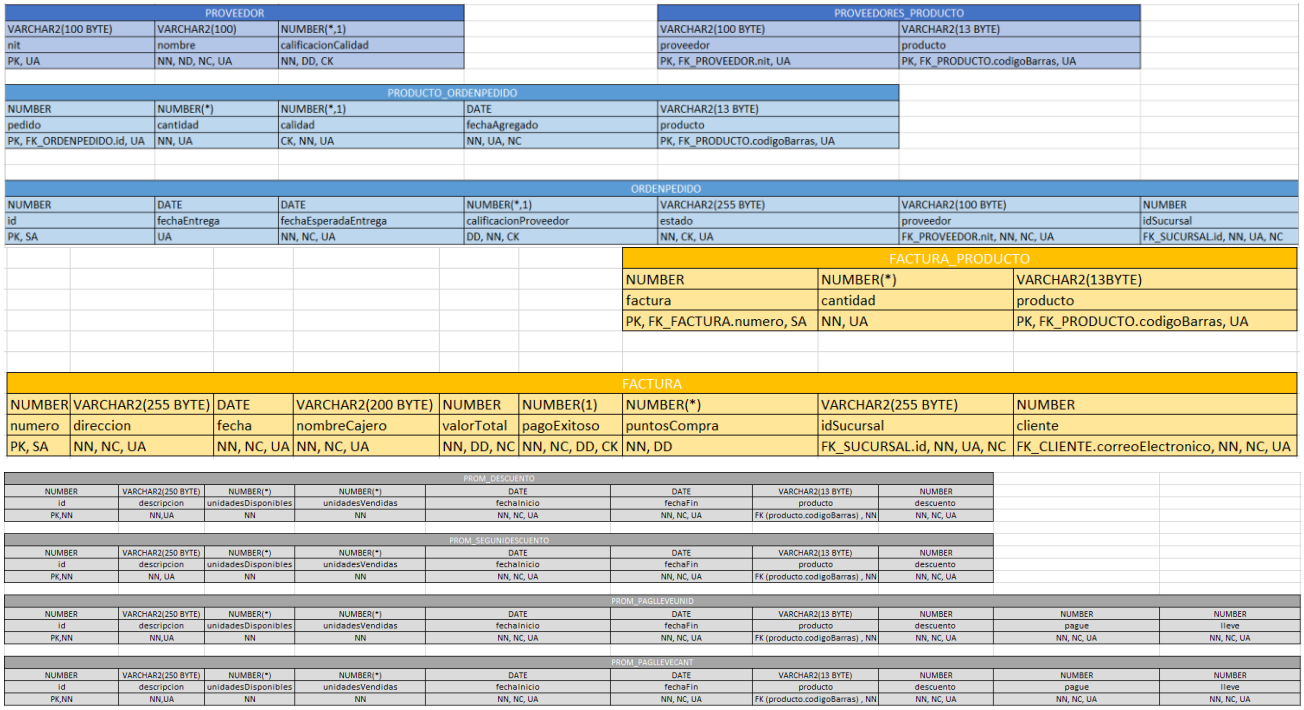

Figura 1. Modelo UML SuperAndes

3.2 Creación de las tablas del modelo relacional:

El archivo de scripts se encuentra adjunto como “esquemaSuperAndes.sql” en la carpeta data del proyecto.

4 Requerimientos funcionales modificación y consulta

Se adjuntan los scripts *sql* de los requerimientos de consulta en la carpeta “data/Iteracion2/RFC”.

Los requerimientos funcionales se pueden revisar en el proyecto de eclipse SuperAndes por medio de la interfaz interactiva con dos modalidades. Por un lado, si se ingresa como ADMINISTRADOR se puede manejar las sucursales, estantes, bodegas, ordenes de pedido, etc. Si se entra cómo cualquier otro usuario, se puede utilizar el carrito de compras, o en su defecto realizar compras directamente.

5 Construcción de la aplicación

5.1 Ajuste de tablas creadas en Oracle

Se modifica el archivo de la carpeta “data/Crear y poblar tablas/esquemaSuperAndes.sql” según los cambios que se generaron en el modelo conceptual y la unificación de los trabajos realizados anteriormente. Las cosas necesarias y cambios de algunos PK (sucursal) también se realizaron, se agregó un chequeo en el rango de la calificación/calidad de los proveedores, ordenes de pedido y productos. Adicionalmente, se agrega el carrito de compras junto con su clase de asociación.

5.2 Población de las tablas

Se adjunta en la carpeta “data/Crear y poblar tablas/poblarTablas.sql” un script con información de la vida real para llenar las tablas cuidadosamente seleccionada para el correcto funcionamiento de la aplicación. Esto incluye veintidós productos con variadas categorías

(once) y tipos, cuatro sucursales, y varios proveedores para las distintas categorías (y asociados con sus distintos productos).

Adicionalmente se vinculan los productos a las sucursales de acuerdo a la disponibilidad que tienen en estantes y bodegas (para cada categoría y por el peso que pueden almacenar). Así mismo, se crean 20 clientes, 10 personas naturales y 10 empresas.

Por último, se generan órdenes de pedido que cumplen el papel de surtir las diferentes sucursales, por lo que se generan como entregadas y con sus respectivos productos.

5.3 Ajustes correspondientes a los nuevos requerimientos:

Para la nueva iteración se implementó la clase carrito de compras, junto con su clase de asociación a producto ProductoCarritoCompras. El carrito de compras está asociado a un cliente, y guarda los productos que él quiere tomar del estante para después comprar, o dejarlos abandonados. El carrito solamente posee un identificador que permite la correcta asociación a la base de datos y la llave primaria de la sucursal a la que pertenece para poder realizar las transacciones que se describen más adelante. La clase de asociación tiene el id del carrito y el identificador del producto junto con la cantidad de unidades de este.

Para satisfacer las condiciones ACID, hacemos que el carrito persista cada vez que un cliente desea tomar un producto, por esta razón, la transacción inicia sacando los productos de un estante en la sucursal a la que pertenece el carrito, luego, se agregan a la clase productoCarrito y se realiza el commit. De esta manera, los otros clientes no podrán tomar los productos en el carrito de otros clientes, y no se generan unidades extra de productos que no deberían aparecer.

Por otro lado, en cuanto a la búsqueda de carritos abandonados se utilizó la herramienta ScheduledExecutorService que nos permite cada N tiempo realizar un procedimiento. Así, definimos que para que se viera en la aplicación, los carritos abandonados por un cliente (es decir, con un cliente nulo) se recogen cada 6 minutos, realizando las transacciones para devolver los productos a sus estantes y eliminando el carrito de la base de datos. Si en el periodo intermedio el cliente vuelve y no se ha realizado este proceso, por medio de un update puede recuperar el carrito con sus productos.

La venta de la iteración pasada no cambia, pero para la venta de los productos del carrito, cada producto del carrito se pasa a la factura, para así poder realizar el proceso de venta y en ese momento, si se realiza la verificación de inventarios trayendo los productos necesarios al estante de Bodega y si se viola el nivel de abastecimiento, se genera la orden de pedido a su respectivo proveedor.

Por último, en cuanto a las órdenes de pedido, desde la iteración pasada se tenía que una orden de pedido podría tener varios productos del mismo proveedor, por lo cual no se cambió nada en el modelaje para satisfacer este requerimiento. Sin embargo, ahora los pedidos se realizan de manera automática (y manual también como antes).

5.4 Pruebas

Se inicia el desarrollo del demo, se generan todas las transacciones necesarias para implementar CRUD, sin embargo, no se realiza la conexión entre la interfaz y estos métodos. Finalmente, se adjunta en la carpeta “data/ Iteracion2” el Excel con la documentación de las pruebas a realizar para cada requerimiento, casos de terminación exitosa, fallidos y las respuestas esperadas.