“Iteración 2”

Juan Sebastián Díaz Serrano, Sergio Guzmán Mayorga
Taller SQL
Universidad de los Andes, Bogotá, Colombia
{js.diaz, s.guzmanm}@uniandes.edu.co

Fecha de presentación: Octubre 4 de 2017

Tabla de contenido

1 Introducción

En el presente documento se presentan los diagramas utilizados para resolver la parte prácitca de la iteración 2[1], y algunas aclaraciones importantes sobre la forma en qué trabajan los requerimientos funcionales y de consulta.

2 Modelos utilizados

Como se sabe que en este documento no se puede visualizar todos los diagramas planteados, se adjunta en la documentación cada uno en formato `png`.

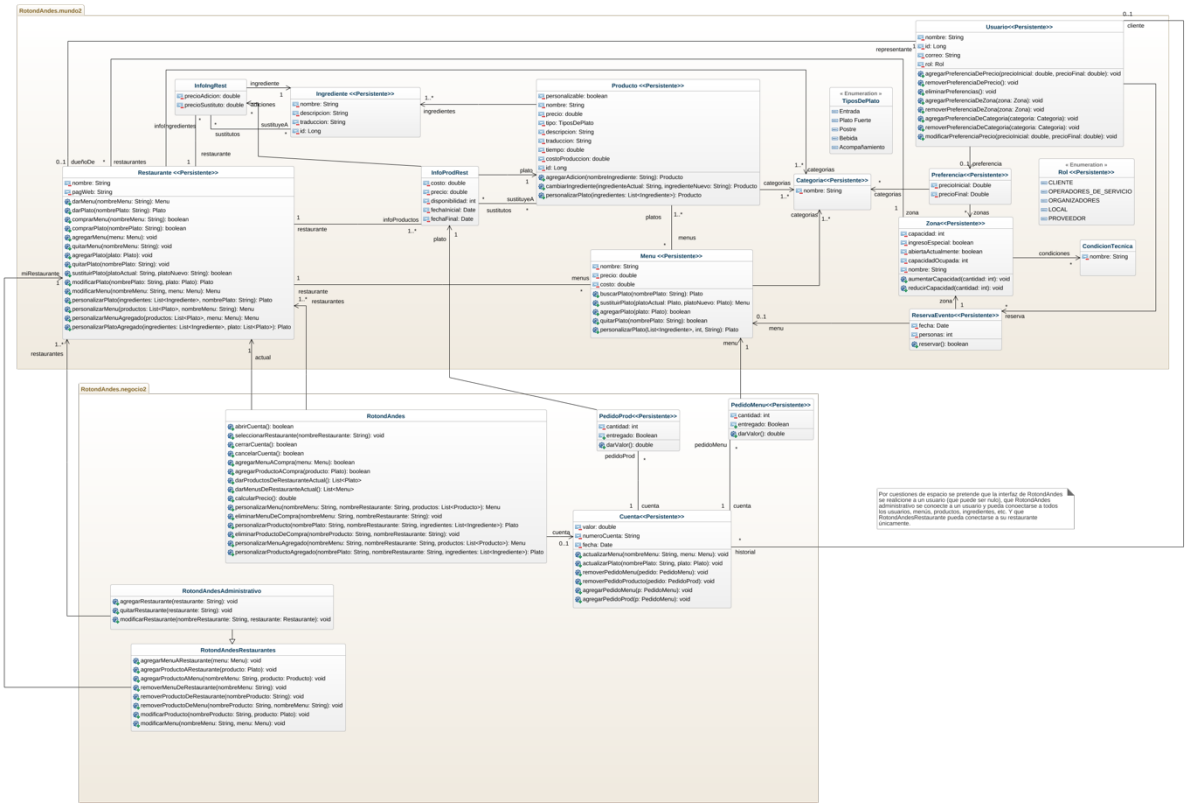

Figura 1. Modelo conceptual

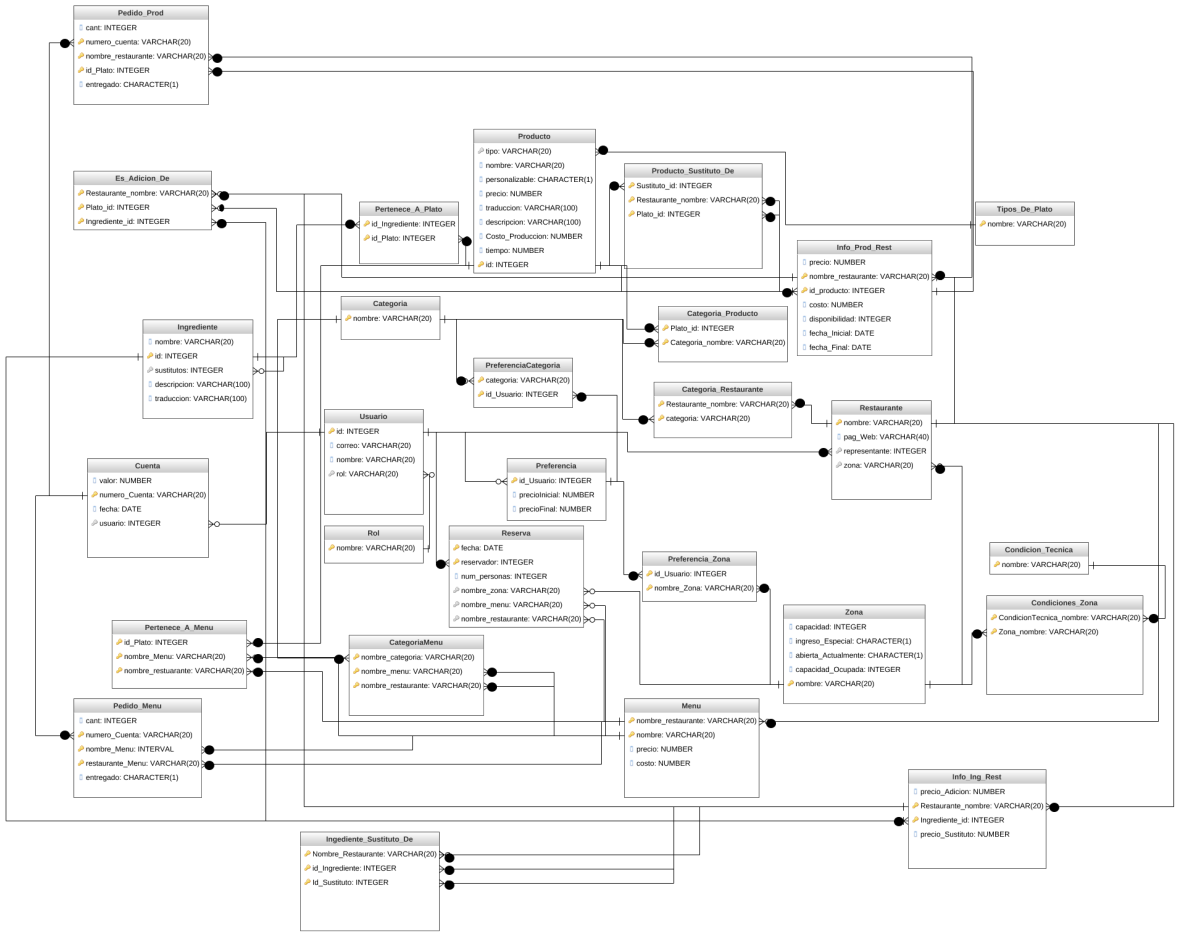

Figura 2. Modelo lógico

3 Distribución del trabajo

Como se sabe que existe la posibilidad de que no haya una sustentación propia del trabajo a presentarse en esta iteración, se adjunta las siguientes tablas con la forma en que distribuimos el trabajo entre los integrantes del grupo.

RECURSO	RESPONSABLE
Rol	Sergio Guzmán (s.guzmanm/jc161)
Usuario	Sergio Guzmán (s.guzmanm/jc161)
PreferenciaCategoria	Sergio Guzmán (s.guzmanm/jc161)
PreferenciaZona	Sergio Guzmán (s.guzmanm/jc161)
Preferencia	Sergio Guzmán (s.guzmanm/jc161)
Ingrediente	Sergio Guzmán (s.guzmanm/jc161)
Zona	Sergio Guzmán (s.guzmanm/jc161)
CondicionesZona	Sergio Guzmán (s.guzmanm/jc161)
CondicionTecnica	Sergio Guzmán (s.guzmanm/jc161)
Cuenta	Sergio Guzmán (s.guzmanm/jc161)
Producto	Sergio Guzmán (s.guzmanm/jc161)
Tipos_De_Plato	Sergio Guzmán (s.guzmanm/jc161)
Categoria	Sergio Guzmán (s.guzmanm/jc161)
Pertenece_A_Menu	Juan Sebastián Díaz (js.diaz)
Pertenece_A_Plato	Juan Sebastián Díaz (js.diaz)
Menu	Juan Sebastián Díaz (js.diaz)
CategoriaMenu	Juan Sebastián Díaz (js.diaz)
Restaurante	Juan Sebastián Díaz (js.diaz)
CategoriaRestaurante	Juan Sebastián Díaz (js.diaz)
CategoriaProducto	Juan Sebastián Díaz (js.diaz)
Reserva	Juan Sebastián Díaz (js.diaz)
InfoProdRest	Juan Sebastián Díaz (js.diaz)
InfoIngRest	Juan Sebastián Díaz (js.diaz)
PedidoMenu	Juan Sebastián Díaz (js.diaz)
PedidoProd	Juan Sebastián Díaz (js.diaz)

Figura 3 Tabla de distribución de recursos y escenarios de prueba

REQUERIMIENTO FUNCIONAL	RESPONSABLE
RF1	Sergio Guzmán (s.guzmanm/jc161)
RF2	Sergio Guzmán (s.guzmanm/jc161)

RF3	Juan Sebastián Díaz (js.diaz)
RF4	Sergio Guzmán (s.guzmanm/jc161)
RF5	Sergio Guzmán (s.guzmanm/jc161)
RF6	Juan Sebastián Díaz (js.diaz)
RF7	Sergio Guzmán (s.guzmanm/jc161)
RF8	Sergio Guzmán (s.guzmanm/jc161)
RF9	Juan Sebastián Díaz (js.diaz)
RF10	Sergio Guzmán (s.guzmanm/jc161)
RFC1	Juan Sebastián Díaz (js.diaz)
RFC2	Sergio Guzmán (s.guzmanm/jc161)
RFC3	Sergio Guzmán (s.guzmanm/jc161)
RFC4	Juan Sebastián Díaz (js.diaz)
RFC5	Sergio Guzmán (s.guzmanm/jc161)
RFC6	Juan Sebastián Díaz (js.diaz)

Figura 4. Distribución de requerimientos

4 Requerimientos funcionales

4.1 RF9

Para este requerimiento, cada vez que se agregaba un producto o un menú al pedido se verificaba la disponibilidad y se actualizaba el precio de la cuenta dependiendo del valor y cantidad dados.

4.2 RF10

Para este requerimiento se tuvo la decisión de coger cada producto y menú encontrado en la cuenta y verificar si era posible pagarlo o no con la disponibilidad actual del restaurante; si lo anterior fallaba de alguna manera se guardaba en una lista todos los artículos coprados que no se tuvieran en el momento y se le mostraba al usuario el listado respectivo. Con estelistado se puede decidir actualizar el pedido, dejar todo cómo está o , cuando se ofrezca la posibilidad de utilizar sustitutos, sustituir el producto dado por alguna de sus posibles opciones.

5 Requerimientos de consulta

Para satisfacer los requerimientos de consulta, y algunos requerimientos funcionales, se creó un paquete en el proyecto llamado `rfc`.

5.1 RFC1

Mismas del requerimiento siguiente.

5.2 RFC2

Para realizar este requerimiento se ofrece la posibilidad de analizarlo en dos variantes: Consultar por una zona específica, y consultar por todas las zonas de la rotonda. Las consultas se realizan enviando un objeto que contiene criterios (detallado más adelante). Como tal se utilizaron las siguientes clases del paquete mencionado al principio de esta sección: `Criterio` (maneja atributos de tablas), `CriterioAgregacion` (maneja operadores de agregación como

COUNT y SUM), CriterioOrden (maneja ordenamientos de atributos de forma ascendente y descendente), CriterioVerdad (maneja operaciones lógicas utilizadas en el equivalente en SQL a WHERE), CriterioVerdadHaving (maneja operaciones lógicas utilizadas en el equivalente en SQL a HAVING), Información (Guarda el nombre y el valor de una columna de una tabla), ContenedoraCriterios (Contiene todos los criterios que se le dan por parámetro) y ContenedoraInformacion (Contiene colecciones de la clase Información y se utiliza para retornar todos los resultados de la consulta esperada). A grandes rasgos el requerimiento consiste en intentar traducir lo que quiere el usuario en una consulta SQL para enviar a un manejador de criterios (DAOTablaCriterio) que se encargue de retornar la consulta en la contenedora variable de información mencionada; en esta consulta solo se pueden utilizar unos atributos dados obtenidos de los metadatos de las relaciones involucradas. Se soportan operaciones de ordenamiento usando ya sea todos los atributos de selección de la tabla, o los atributos de agrupación si existe tal aparte en la consulta deseado. A su vez se permite realizar agrupamientos de valores, tomando el conteo de elementos como operación de agregación predeterminada en caso de que no se proporcionen otras. Por último se permite la combinación lógica de operaciones de verdad utilizando CriterioVerdad y CriterioHaving que pueden ser del siguiente estilo:

- Criterio *op* Valor
- Criterio *op* Criterio
- Criterio *IN* (Valores)

Donde:

- Op: Operaciones de comparación (>.>=.=.<.<=, LIKE)
- Vaor: Valor fijo que no depende de un atributo
- Criterio: Atributo o criterio de agregación (dependiendo de si se pretende hacer un *WHERE* o un *HAVING* respectivamente)
- *IN*: Hace parte de.
- Valores: Conjunto de valores fijos separados por coma.

A su vez, cualquiera de estas operaciones se puede negar y se pueda concatenar entre sí con el uso de conjunciones (*AND*) y disyunciones (*OR*). Para tener más ejemplos de estas funcionalidades se invita a que el interesado revise el documento readme.txt con ejemplos al respecto.

5.3 RFC3

Para consultar toda la información de un cliente se tuvo que crear una nueva clase UsuarioCompleto donde se obtuviera toda la información del usuario, agregándole la frecuencia de compra por día de semana.

Para realizar esta consulta se tuvieron que crear tres nuevas clases: ProductoEnTotal (guarda el id del producto, su costo y valor total, y la cantidad vendida), CategoriaProducto (guarda la información total del producto, agrupada por categoría) y ContenedoraZonaCategoriaProducto (guarda las categorías de producto agrupadas por zona). De esta manera, se retorna la última clase que contiene toda la información para que el usuario interesado en la consulta visualize el resultado. Cabe resaltar que se envía en otra clase LimiteFechas la fecha de inicia y la de fin de la consulta.

6 Consideraciones adicionales

Dentro de la carpeta data se tienen las siguientes consideraciones:

- Se tienen los archivos de pruebas de *Postman* en la carpeta pruebas. Antes de ejecutarlas se debe correr el archivo PrePostman.sql.

- En el archivo `BorradoEInserciones.sql` se tienen los comandos para borrar todos los datos existentes en la base de datos y volver a poblarla en caso tal de que se corrompan de laguna manera los datos del sistema.
- En el archivo `DBSETUPFULL.sql` se tienen los esquemas de las tablas para reconstruir la base de datos dado el caso en que halla necesidad de hacerlo.
- En la carpeta `inicializacion` están todas las inserciones en tablas utilizadas en la población de la base de datos.
- La ruta se sigue llamando VideoAndes de acuerdo con el ejemplo de clase, dado que no se encontró la forma de cambiar el nombre a RotondAndes como se pretendía.

7 Análisis de resultados

7.1 Aprendizajes y logros

A lo largo del desarrollo de la iteración 2 se aprendió a crear un sistema transaccional básico con múltiples iteraciones entre clases que realiza algunas consultas interesantes.

7.2 Conclusiones

- Se debe saber determinar las operaciones transaccionales del sistema porque hay varias que requieren ser o todo o nada.
- SQL es una herramienta poderosa que permite realizar consultas complejas sobre modelos de datos gigantes.
- La interconexión entre cñases permite la funcionalidad completa de un sistema transaccional dado.
- Toda decisión de diseño se debe reflejar en la base de datos planteada.

8 Bibliografía

1. Universidad de los Andes. *Taller catálogo y SQL básico.* [En línea] Universidad de los Andes. [Citado el: 4 de Octubre de 2017.] https://sicuaplus.uniandes.edu.co/bbcswebdav/pid-1918111-dt-content-rid-20097375_1/courses/UN_201720_ISIS2304/isis2304-172-CasoEstudio-RotondAndes-v2.pdf