Iteración 4:

María Paula Franco Guzmán, Camilo Andrés Sánchez Salamanca
Iteración
Universidad de los Andes, Bogotá, Colombia
{mp.franco10, ca.sanchez38}@uniandes.edu.co

Fecha de presentación: Noviembre 21 de 2017

Tabla de contenido

1 Introducción

En el ámbito de los restaurantes en grandes centros comerciales, los sistemas transaccionales son componentes importantes para realizar las operaciones más importantes de la organización e incrementar la productividad de estas. Estos permiten controlar y administrar múltiples transacciones manteniendo seguridad, persistencia y consistencia de los datos manejados. En el caso de "RotondAndes", empresa que actúa como intermediaria entre restaurantes y usuarios finales, los sistemas transaccionales permiten a los usuarios consultar, comprar y pagar servicios de alimentación de acuerdo a sus intereses y a las restricciones de los proveedores.

Para realizar una implementación correcta de un sistema transaccional es necesario realizar procesos de diseño y análisis del sistema. El proceso inicial de diseño consiste en identificar los requerimientos funcionales del sistema, conocer las restricciones o condiciones de este y realizar modelos conceptuales que permitan representar el sistema e identificar entidades y relaciones entre estas. Posteriormente, se deben realizar modelos de datos y relaciones a partir de los modelos conceptuales. Se debe realizar la implementación del diseño realizado en un sistema transaccional y las características de este que le permiten tener las propiedades A.C.I.D. En el presente documento se proponen las modificaciones a los modelos conceptuales de la aplicación utilizada por la organización "RotondAndes" así como a los modelos de datos y relacionales de la aplicación.

Para finalizar es necesario asegurarse de haber realizado una implementación óptima, no únicamente óptima respecto a las operaciones en la sentencia, si no también con el aprovechamiento de índices en las tablas consultadas. Finalmente se presentan los resultados de la implementación de la aplicación transaccional y se documentan los requerimientos pedidos.

2 Etapa 1: Análisis

A nivel de modelo conceptual no se realzo ninguna modificación ya que los objetos que representan las diferentes cosas que pertenecen a la rotonda no tienen ningún cambio, así mismo no se agrega ni se elimina ninguna funcionalidad. Por tanto, el UML no cambia en ningún momento y se mantiene con respecto a la iteración pasada.

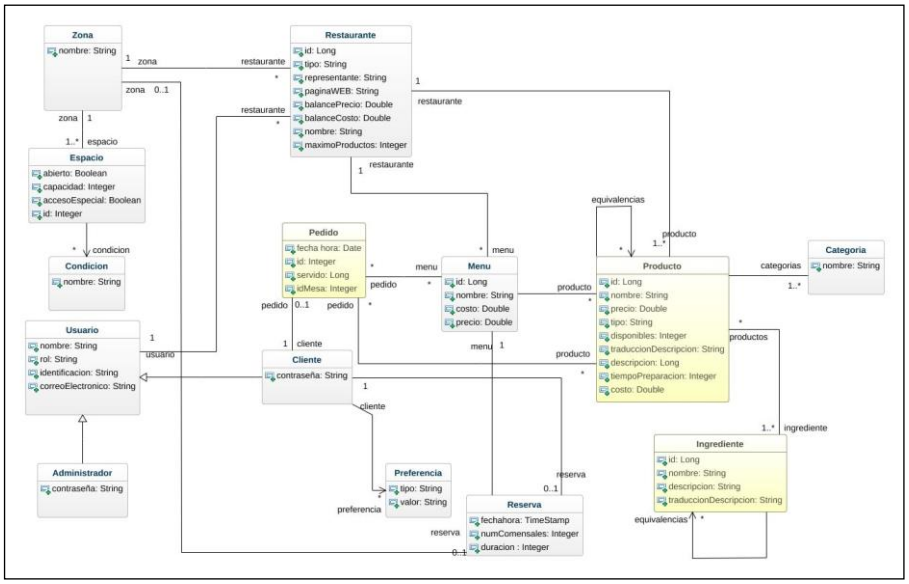


Figura 1 Modelo del mundo modificado

A nivel del modelo relacional, tampoco se realiza ningún cambio esto debido a que las relaciones se mantienen iguales, y al no haber cambio en el modelo conceptual tampoco se hace necesario agregar ninguna tabla nueva. Lo único que podría decirse que cambia respecto a este modelo es el uso de índices en las tablas para hacer que las consultas sean mucho más optimas, sin embargo, es un tema que se explicara más adelante y aunque se relaciona con el modelo relacional no hace parte de este.

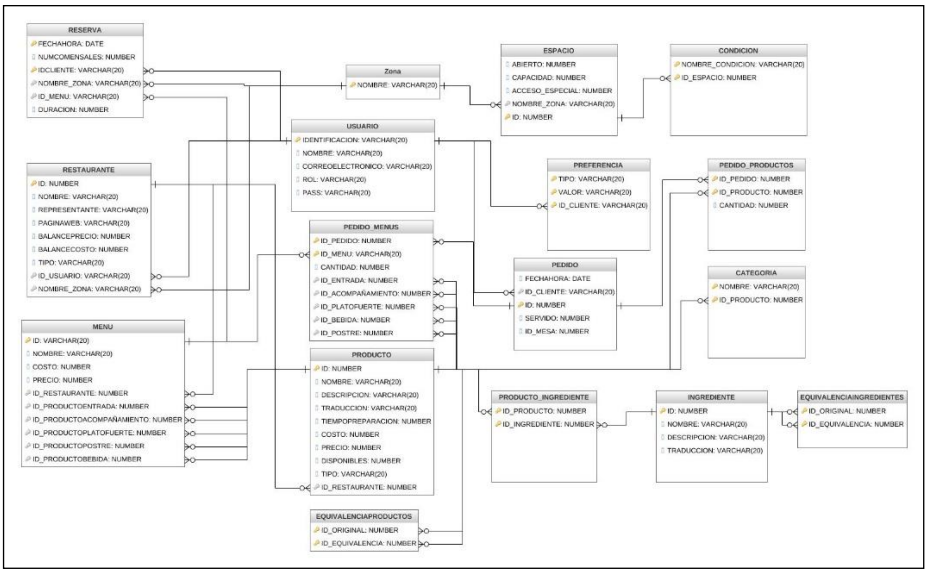


Figura 2 Modelo relacional

En la iteración anterior las tablas se habían diseñado en BCNF, por lo que no se realizaron cambios de calidad. La anormalidad que hay es la de contener los productos del menú en el pedido, pero esto es debido a que el restaurante necesita conocer qué es lo que el cliente quiere pedir. Mediante transacciones se manejan estas columnas.

3 Etapa 2: Diseño de la aplicación

Para esta iteración como ya se mencionó antes no se realizó ninguna modificación en las tablas de la base de datos, ya que la idea principal del trabajo era optimizar el trabajo sobre estas tablas, mas no agregar cosas nuevas como tal, sin embargo, hubiese sido útil crear algunas columnas nuevas con datos específicos, para ahorrar el uso de algunas funciones, como por ejemplo para fecha ya que se hace inútil crear un índex de fecha ya que se ejecuta una función sobre este valor para poder hacer el filtro.

Se crearon diferentes en las tablas para optimizar cada consulta.

Consulta 1:

Se colocaron índices sobre las tablas de relación de pedidos y productos, tanto sobre su id de producto y su id de restaurante, los cuales no funcionaron debido a que el plan de trabajo ya usaba las llaves primarias de estas tablas para realizar las búsquedas, por tanto, ya se está usando un índex en estas. A su vez se realizó un índice compuesto sobre la tabla pedido con su fecha y el id del cliente, sin embargo, al igual que los índices nombrados anteriormente no fueron de utilidad, ya que el análisis de la fecha se hace con una función por tanto el índex no puede ser utilizado ya que terminan revisándose todas las tuplas de la tabla.

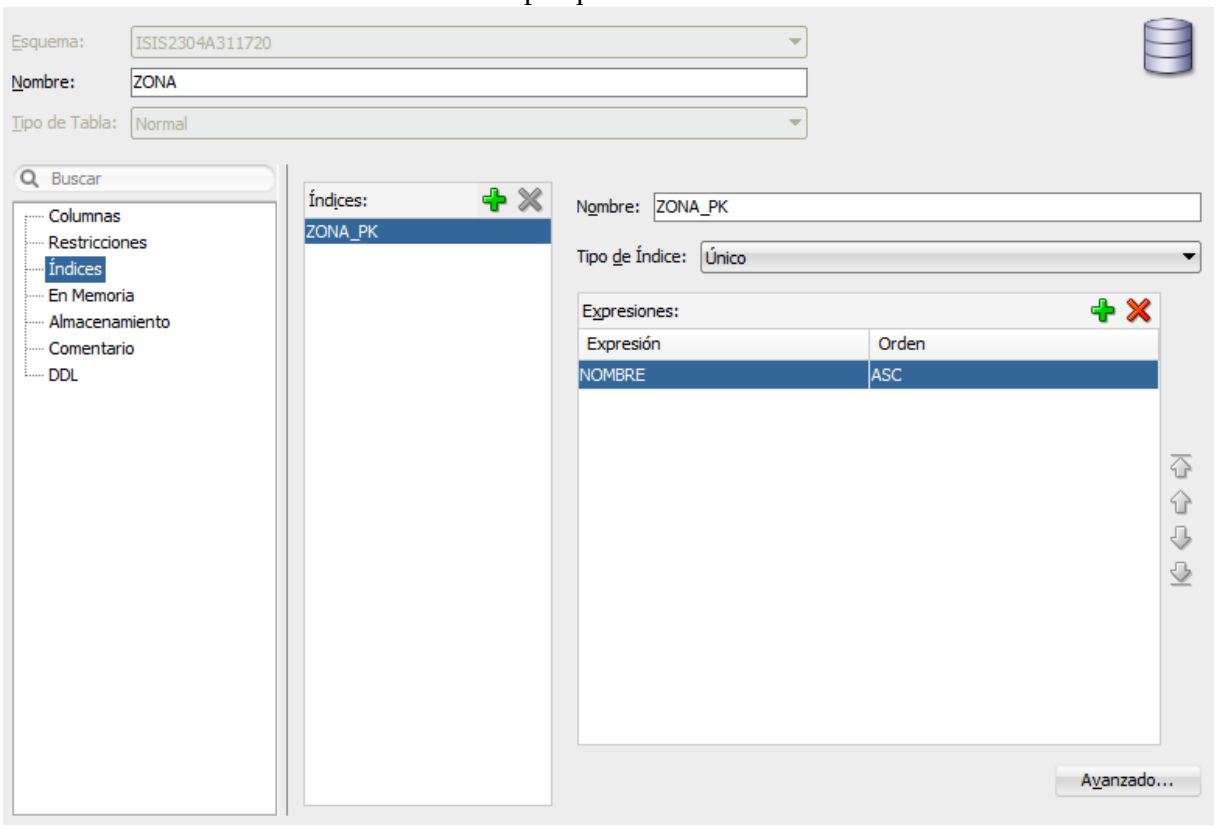
Consulta 2:

Ya que esta consulta es la negación de la anterior, esta se basa por completo en la anterior, solo que buscando aquellos usuarios que no pertenecen a esta tabla. Por tanto se hace imposible el uso de un índice para optimizarla, o al menos debido al plan de trabajo que se usa el cual ya esta usando llaves primarias en todas sus búsquedas, es completamente inútil crear mas índices, y el único índice compuesto es inútil también debido a la forma en la que se analiza el rango de fecha.

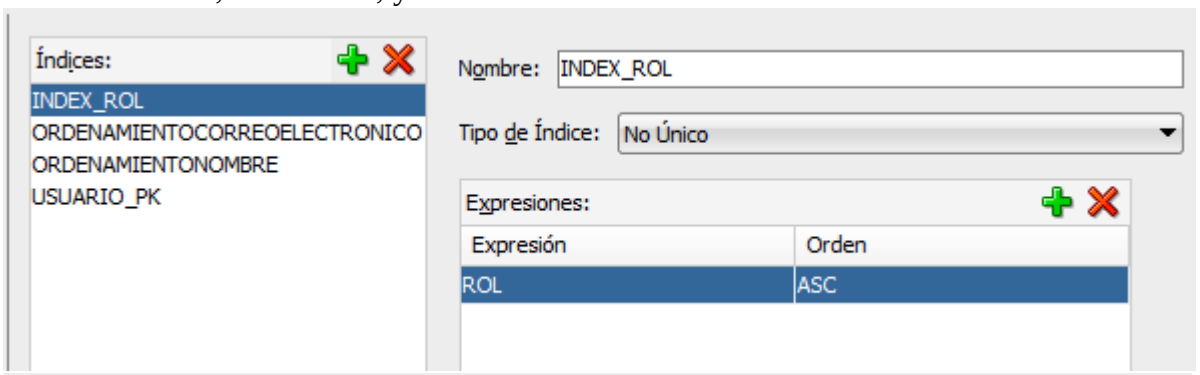
** En el archivo adjunto se muestra el análisis de eficiencia de las consultas realizadas, así como las sentencias SQL y los planes de ejecución de estas. Es decir, allí se presentan los escenarios de prueba.

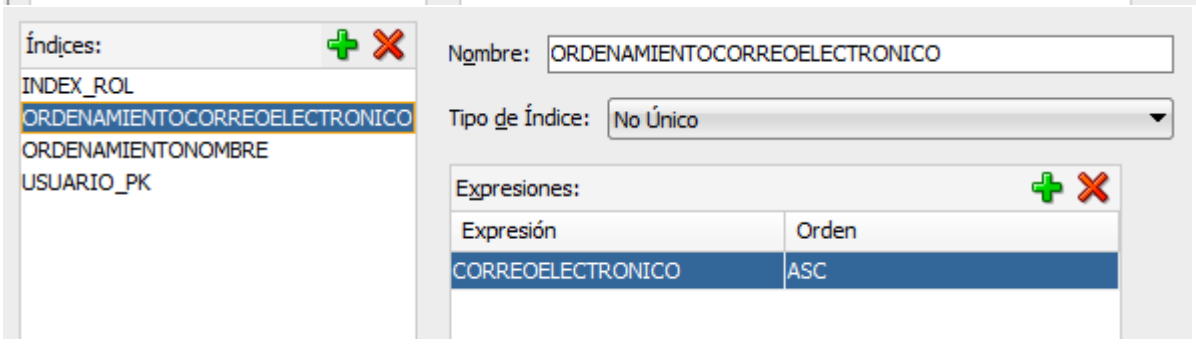
A continuación se muestran los índices implementados mostrados por Oracle:

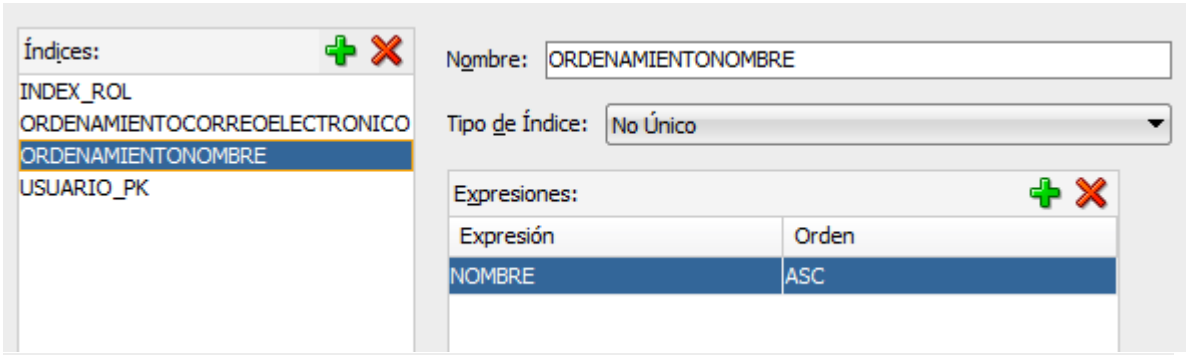
- Zona: Solo se encuentra el índice de PK porque solo tiene una columna

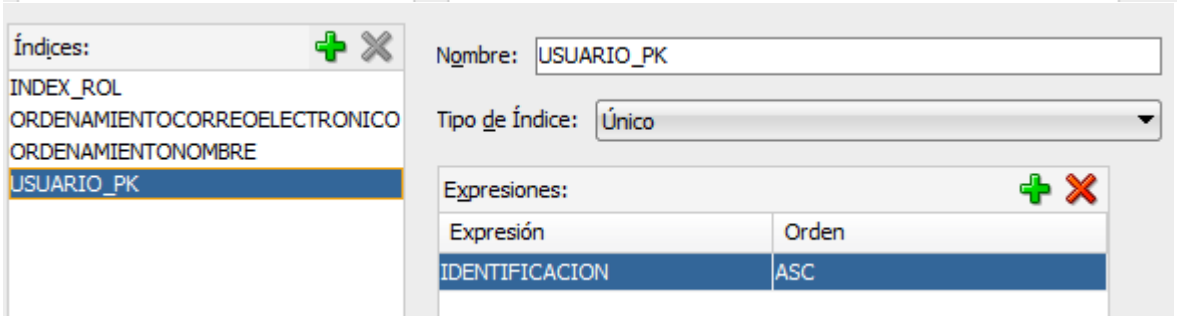


- Usuario: en esta tabla se encuentra un índice para el rol para obtener más rápidamente el rol de un usuario, aunque la selectividad del rol es muy baja. Se encuentra índice de la PK de usuario, del nombre, y del correo electrónico.

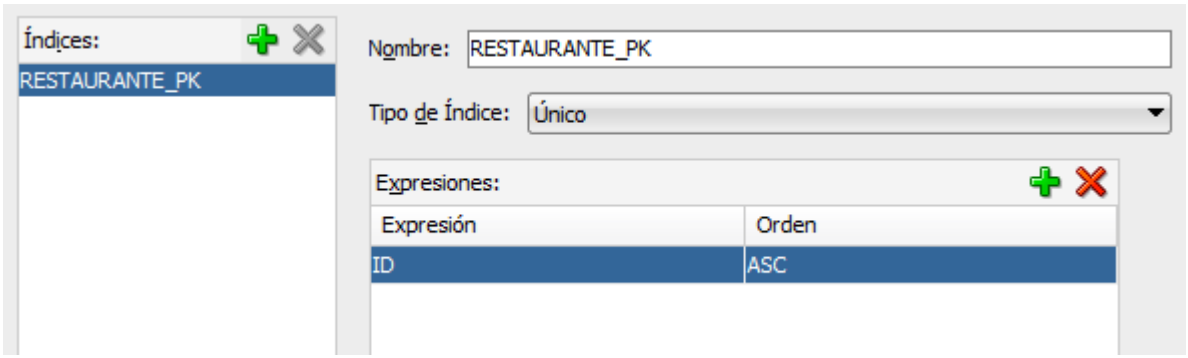




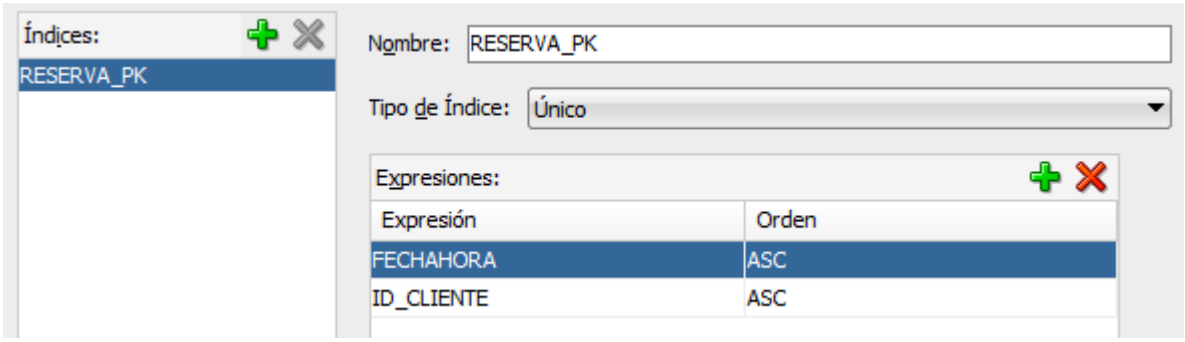




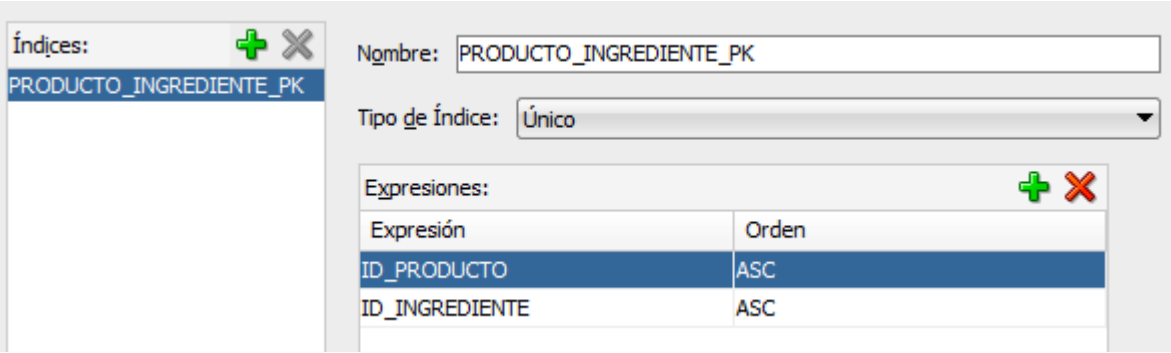
- Restaurante: en este caso solo se tiene un índice en la PK del restaurante.



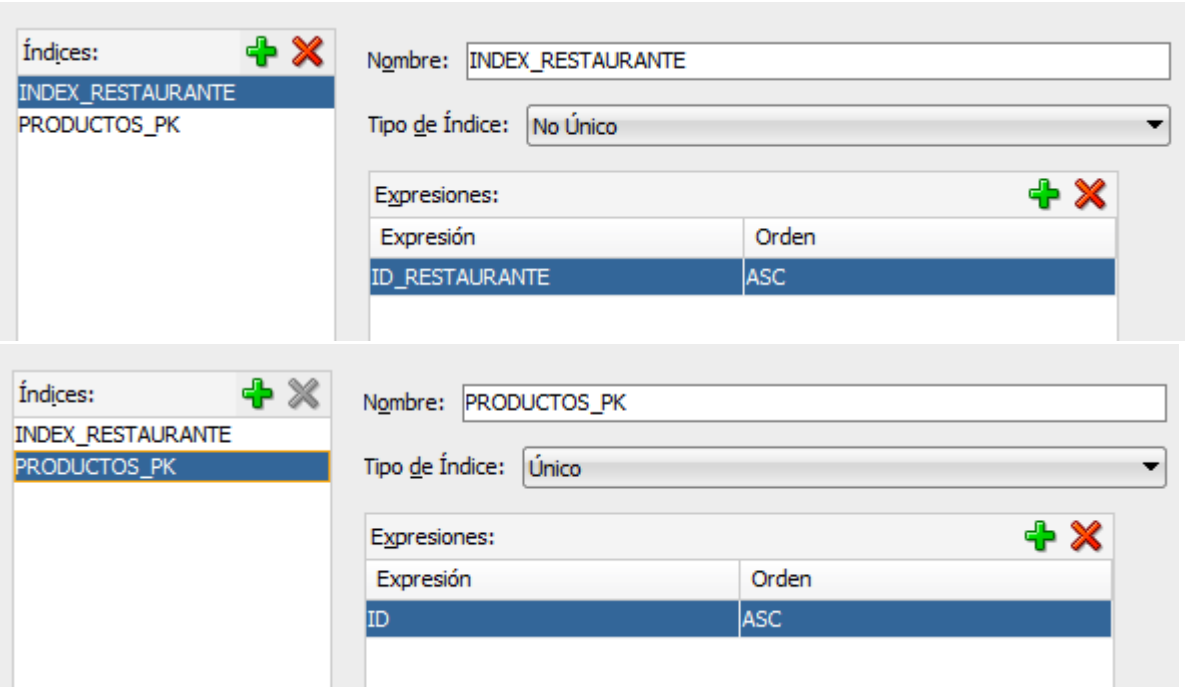
- Reserva: en este caso solo se tiene un índice sobre la PK ya que no se utiliza en las consultas



- Producto_ingrediente: en este caso solo se tiene un índice hacia la llave primaria de la tabla. Como la PK de la tabla es compuesta, se tienen dos atributos que pertenecen al índice.

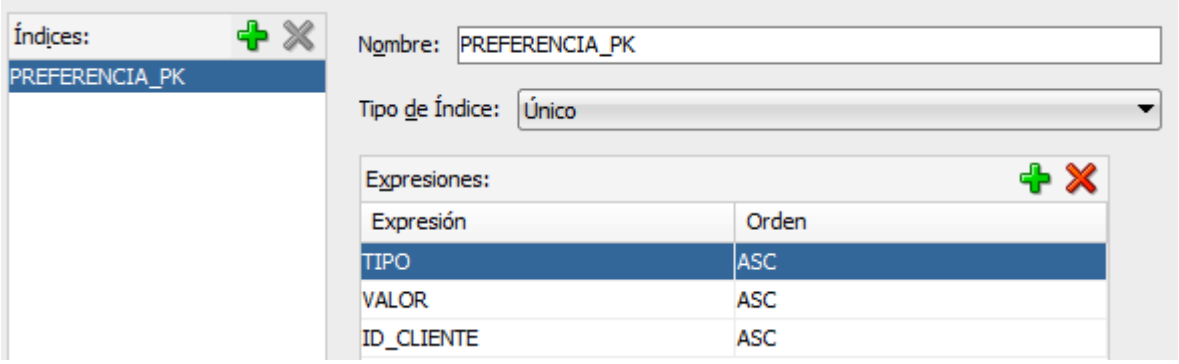


- Producto: se utiliza un índice para el restaurante del producto, pues están uniformemente distribuidos. También se utiliza el índice de PK de la tabla.

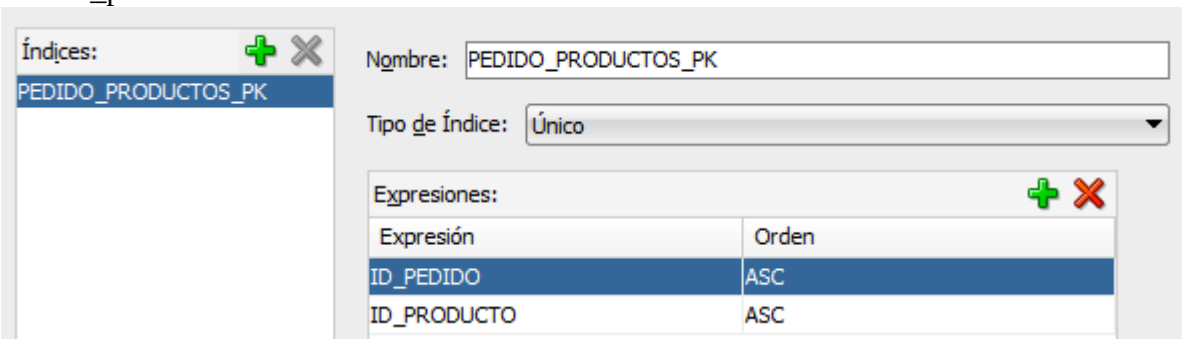


- Preferencia: En este caso solo se tiene el índice a la PK de la tabla. Este índice cuenta con 3 atributos. Tipo, valor e id_cliente. El índice los ordena así: primero el tipo, luego

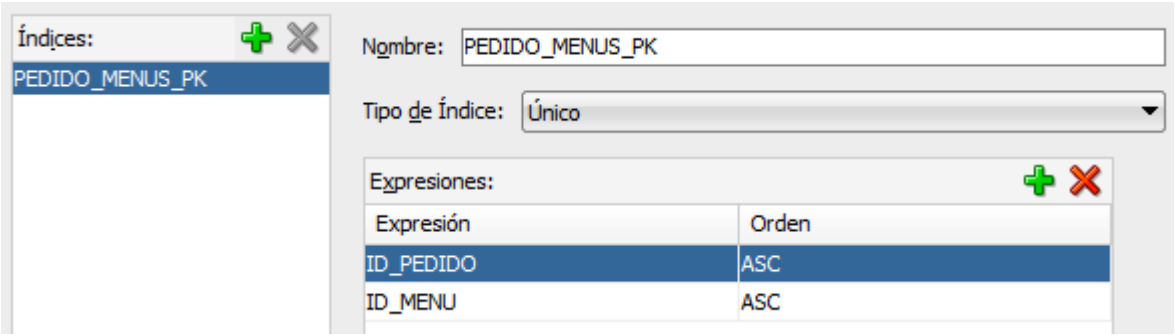
el valor y luego el id_cliente. Esto debido al orden en que se declaran las columnas al crear la tabla.



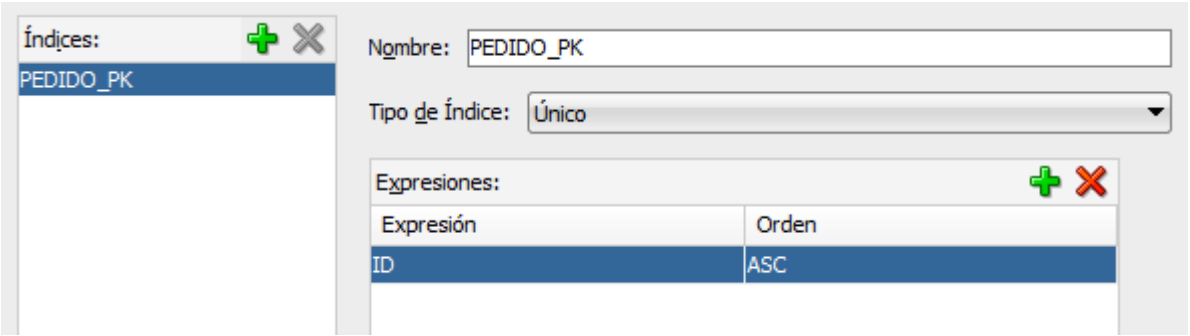
- Pedido_productos: en este caso solo se tiene el índice hacia la PK de la tabla.



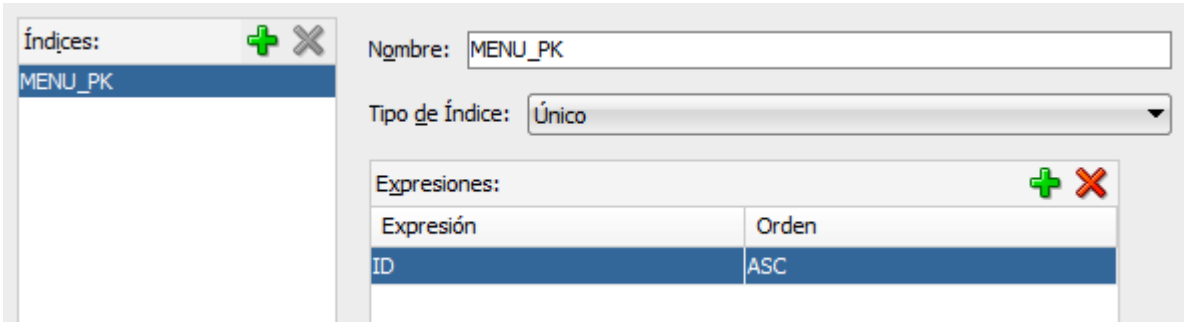
- Pedido_menus: en este caso solo se tiene el índice hacia la PK de la tabla.



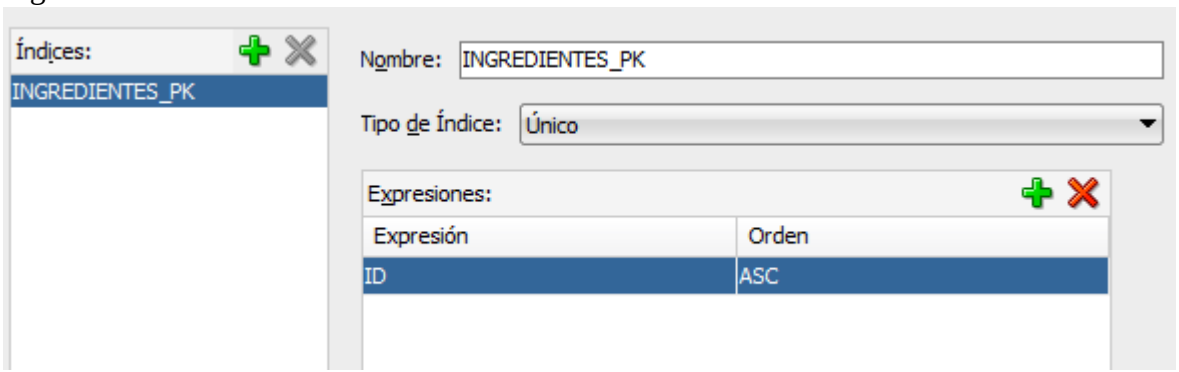
- Pedido: en este caso solo se tiene el índice hacia la PK de la tabla.



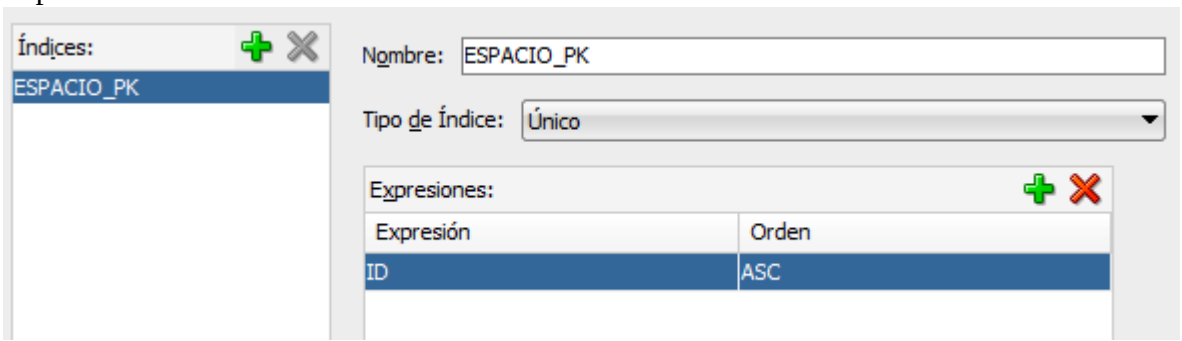
- Menu: en este caso solo se tiene un índice hacia la PK de la tabla.



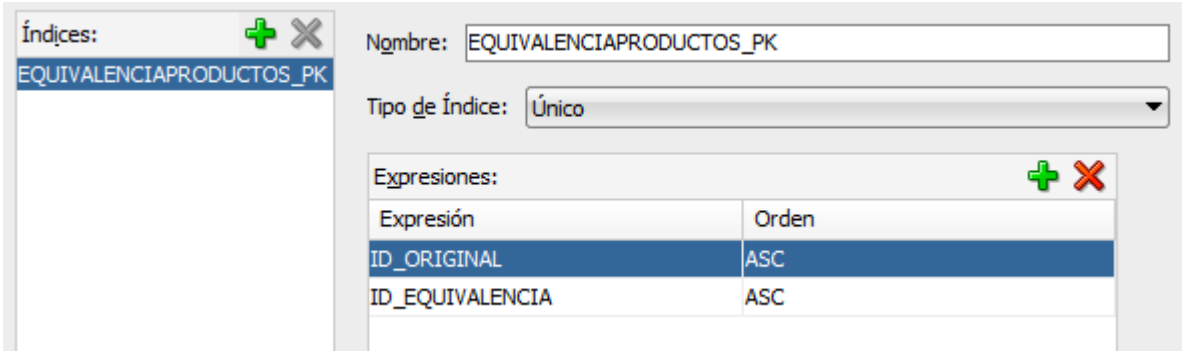
- Ingrediente: en este caso solo se tiene un índice hacia la PK de la tabla



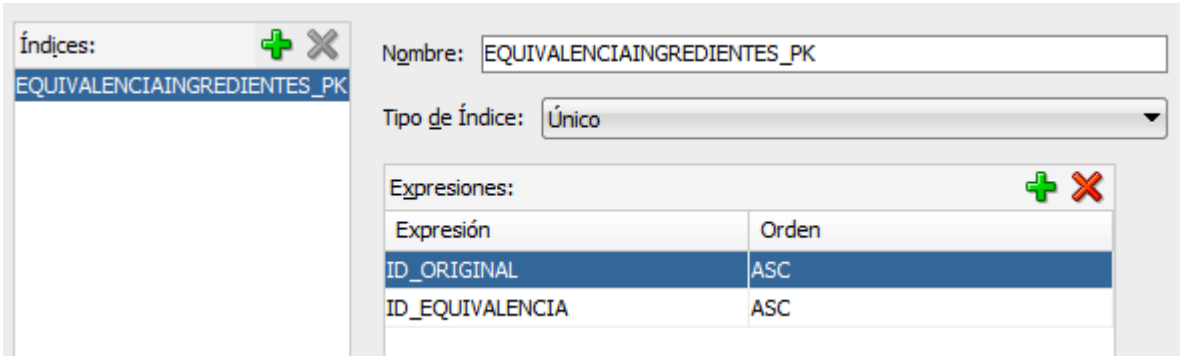
- Espacio: en este caso solo se tiene un índice hacia la PK de la tabla.



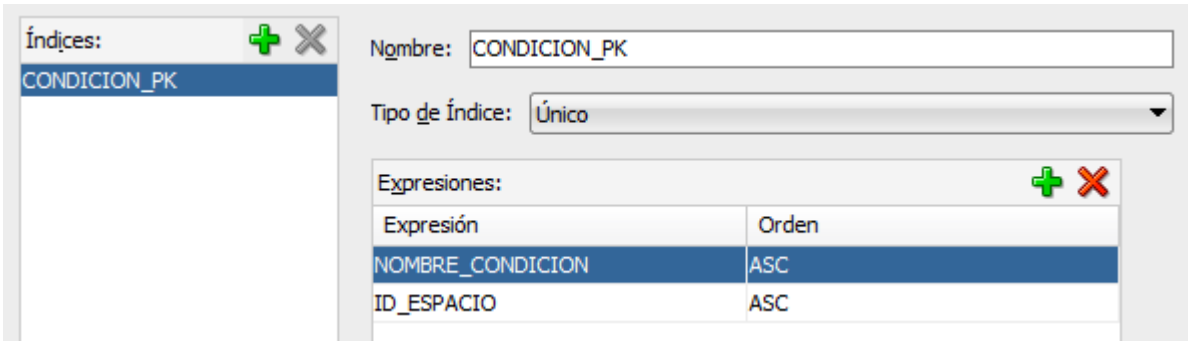
- Equivalenciaproductos: en este caso solo se utiliza el índice hacia la PK de la tabla.



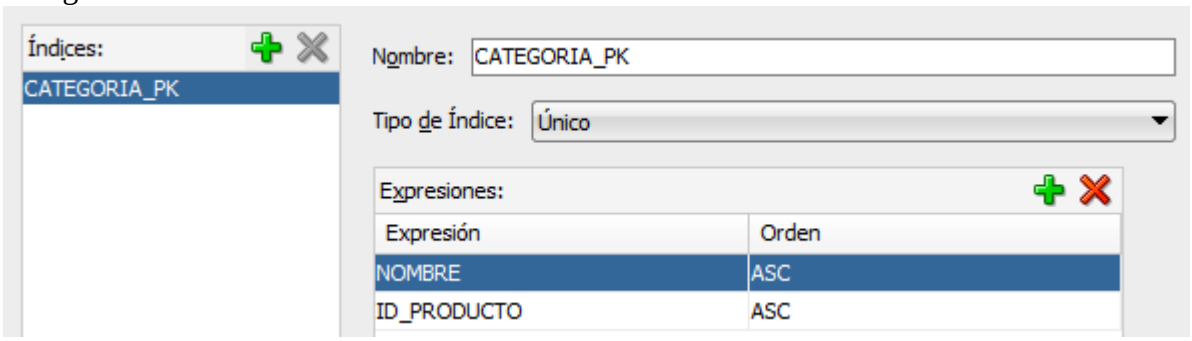
- Equivalenciaingredientes: en este caso solo se utiliza el índice hacia la PK de la tabla.



- Condicion: en este caso solo se utiliza el índice hacia la PK de la tabla.



- Categoria: en este caso solo se utiliza el índice hacia la PK de la tabla.



En cuanto al análisis de índices, los que generó Oracle automáticamente en todas las tablas fueron los de las PK. Esto le permite al sistema buscar y ordenar de manera eficiente debido a que la selectividad de las PK es del 100%, es decir, todos los valores de todas las filas son diferentes en la columna o columnas. Muchos de estos índices ayudan en consultas de iteraciones anteriores, pero los necesarios para agilizar las nuevas consultas fueron implementados por nosotros. Sin embargo, aquí se presentan en su mayoría pantallazos de índices generados automáticamente por Oracle.
Podemos ver que los índices se ordenan ascendentemente y los tipos de índice para las PK son siempre únicos.

4 Etapa 3: Construcción de la aplicación y análisis de resultados

Carga de datos:
Para cargar los datos en la base de datos se siguieron los siguientes procedimientos:

- ➔ Si la columna a llenar era una secuencia de números (1,2,…,n), cadenas de caracteres al azar, o un número al azar entre rangos se utilizó la herramienta de Kutools para Excel, que permite generar datos sencillos al azar.
- ➔ Si la columna dependía de llaves de otras tablas (foreign key) se seleccionaron datos al azar utilizando un programa de generación de datos realizado en Matlab. Esto se realiza poniendo las llaves primarias de una tabla en un arreglo y seleccionando un índice al azar para obtener un dato aleatorio.
- ➔ Finalmente se mezclaron los datos generados y se importaron a las tablas mediante un CSV, pues en este formato los datos se ingresan a la tabla muy rápido.

A continuación se muestran el número de datos ingresado a cada tabla:

- Zonas = 202
- Usuario = 1000001
- Restaurante = 200
- Reserva = 499405
- Producto_ingrediente = 900000
- Producto = 11371
- Preferencia = 360865
- Pedido_productos = 998668
- Pedido_menus = 600000
- Pedido = 799960
- Menu = 10000
- Ingredientes = 2000
- Espacio = 1000
- Equivalenciaproductos = 1000
- Equivalenciaingredientes = 1000
- Condicion = 1111
- Categoria = 11371

Total filas ingresadas: **5'198.154**

Las tablas en Oracle no se ajustaron para esta iteración, por lo que se llenaron los datos con el mismo formato de la iteración anterior. Se borraron los datos utilizados en la iteración anterior pues eran muy pocos, y se generaron unos nuevos. Por facilidad, cuando los identificadores de las tablas sencillos se piden, se ingresaron valores del tipo "Entidad1, Entidad2, Entidadn" si se trata de un id de tipo cadena de caracteres. Si se trata de un id numérico se ingresan secuencias de números "1,2,3,4,…,n".

Los datos se crearon de acuerdo al mundo real. Lo más lógico es que hayan más usuarios, pedidos, ingredientes en productos y preferencias. Las demás tablas tienen un número pequeño de datos (comparado con el número de las tablas más grandes) pero se estima que no se necesitan tantos datos en estas tablas para probar la eficiencia de la aplicación.

En cuanto a los servicios REST de la aplicación, se implementaron los Daos para las columnas que retorna cada consulta. Sin embargo, por falta de tiempo no se alcanzaron a realizar las pruebas y los cambios en las transacciones para los requerimientos de consulta.

En la iteración anterior se utilizó un archivo SQL, pero en este escenario no tiene lógica realizar un archivo SQL, pues si ejecuta la inserción de filas mediante un script en sqldeveloper correr este archivo duraría días. Esto se debe a que se trataron de insertar 50000 datos utilizando un script y tardó un tiempo aproximado de 30 minutos, por lo que

insertar los datos ya presentados tardaría aproximadamente 52 horas realizando una regla de 3. Por estas razones, las pruebas de Postman de iteraciones anteriores funcionan en la primera ejecución y después de esto no vuelven a funcionar. Para arreglar este error podría desarrollarse un archivo SQL que borre solo las tuplas utilizadas en las pruebas antes de ejecutarlas.

La ejecución de consultas puede cambiar de acuerdo a las operaciones que se realicen allí. Si las consultas se delegan al manejador de base de datos, este es capaz de realizar los algoritmos que dependen del tamaño de las tablas.

En nuestro caso no se utilizaron outer joins ya que estos cuestan más que los inner joins y natural joins. Por lo tanto, se utilizaron inner joins y natural joins, pues el manejador sabe realizar los algoritmos de acuerdo al tamaño de las tablas. No se realizaron joins manuales para evitar tiempos extras en las consultas.

Tampoco se utilizaron operaciones de LIKE, pues estas no utilizan índices y cuestan mucho en ejecución.

Se utilizaron operaciones de DISTINCT en algunos casos pues eran necesarias para las consultas, sin embargo, se conoce que estas operaciones cuestan en operaciones de I/O y hacen que la consulta sea más lenta.

Utilizamos UNION en un caso de consulta en el que era necesario, a nuestra vista, para cumplir con los requerimientos pedidos.

En resumen, tratamos de utilizar lo menos posible operaciones costosas y utilizamos las que provee el manejador de bases de datos (en este caso Oracle) para reducir tiempos y costos. Si nosotros hubiéramos implementado las consultas sin utilizar el manejador de bases de datos probablemente hubieran sido mucho más demoradas porque Oracle, como un gran sistema, conoce como optimizar las consultas y reducir los tiempos. Los índices implementados también nos sirvieron para agilizar las consultas realizadas.

5 Bibliografía

[1] Oracle, «Oracle Docs,» Oracle Help Center , 2017. [En línea]. Available: https://docs.oracle.com/en/.

[2] W3Schools, «SQL Tutorial,» W3Schools, 2017. [En línea]. Available: https://www.w3schools.com/sql/default.asp