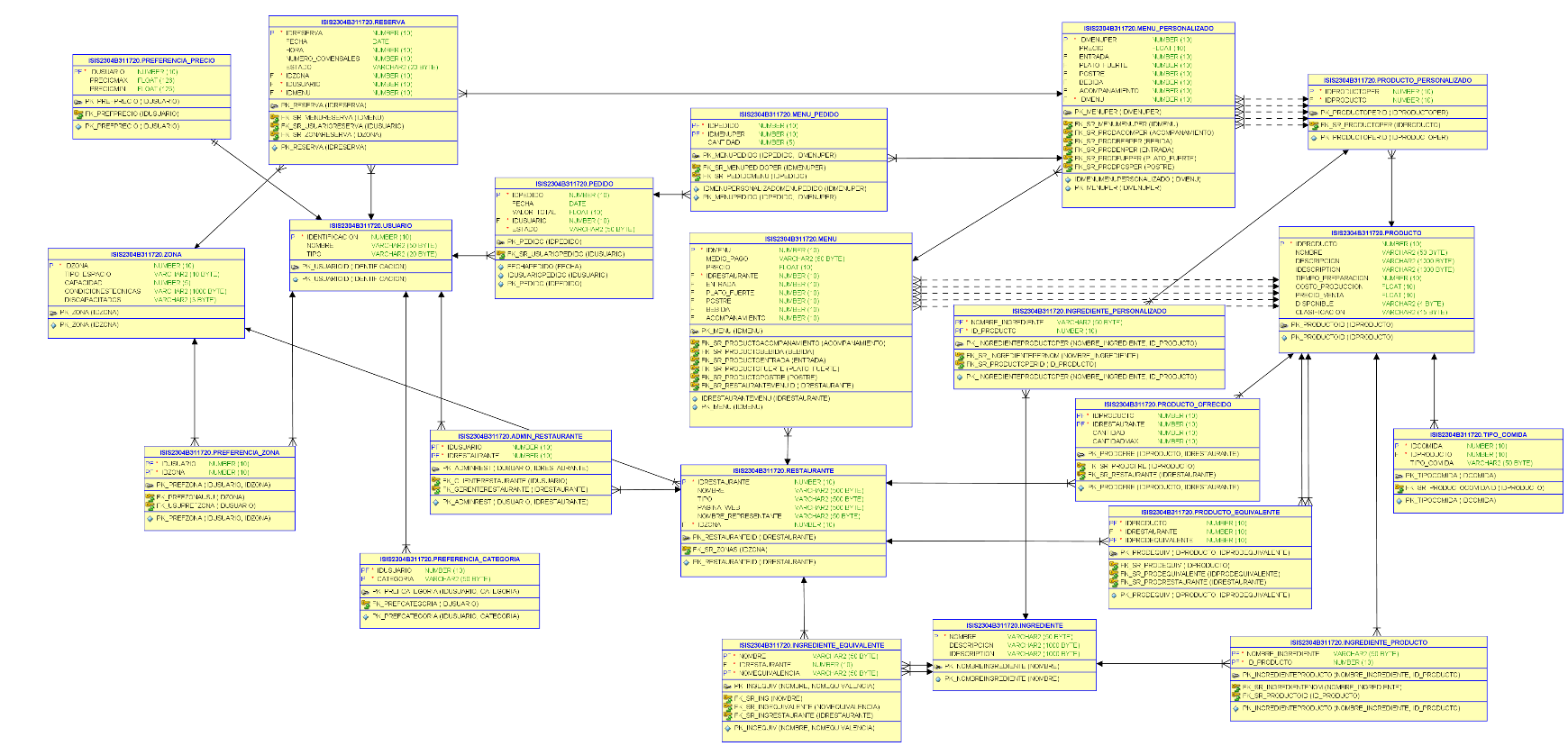
# Sistemas transaccionales – Caso de estudio Rotond Andes

Iteración 4

1-Algunos de los cambios realizados con respecto a las iteraciones pasadas para cumplir con los objetivos de la nueva fueron:

* Se agregó una nueva tabla que relaciona un administrador con un restaurante para cumplir con una de las condiciones del requerimiento nueve. Es decir, para asociar un administrador con los restaurantes que administra el mismo y que solamente pueda consultar el consumo del restaurante del cual es administrador.
* Se extendió la capacidad de las descripciones de productos e ingredientes.
* Se agregó un nuevo rol de usuario llamado AdminRestaurante, el cual representa un administrador de uno o más restaurantes.
* En las demás tablas no hicimos ningún cambio adicional.



2-**Diseño Físico**

-> Justificar la selección de índices para cada uno de los requerimientos funcionales, indicar cuál es el tipo de índice utilizado (B+, Hash, Primario o secundario) Agregar costos.

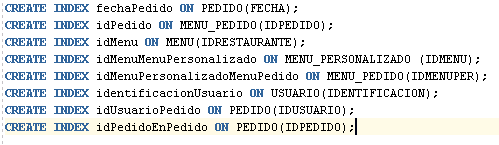
# Selección de índices

Requerimiento 9: Para el requerimiento nueve se utilizaron los siguientes índices:

* Índice en el id de un Pedido de la tabla menú pedido.\*
* Índice en el id de restaurante de la tabla menú.
* Índice en el id de menú de la tabla menú personalizado.
* Índice en el id de menú personalizado en la tabla menú pedido.
* Índice la identificación del usuario.\*
* Índice en el id de usuario de la tabla pedido.
* Índice en el id de pedido de la tabla pedido.\*

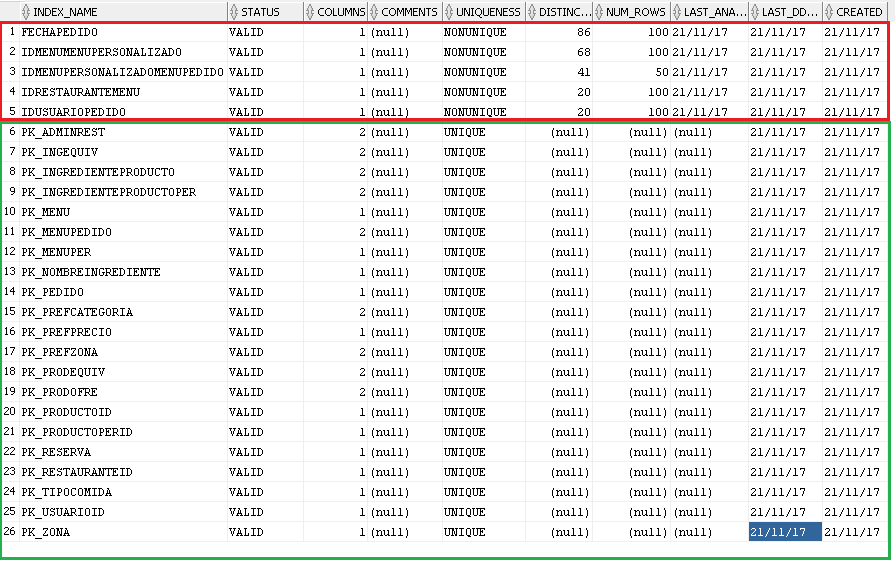
**Nota:** Índices marcados con \* son creados automáticamente por SQLdeveloper por ser o formar parte de la llave primaria.

Los índices fueron creados de la siguiente manera:



-> Para los índices creados de forma automática por Oracle: Incluya pantallazo con la información generada por Oracle sobre los índices existentes.

A continuación se muestran los índices que se planearon para el desarrollo del requerimiento número nueve (en rojo). Los que inician con PK\_NAME fueron creados automáticamente por SQL developer (en verde).



-> Analizar por qué fueron creados los índices por Oracle y si ayudan al rendimiento de los requerimientos funcionales.

Los índices fueron creados por SQL developer sobre las llaves primarias de la tabla, porque la selectividad de estos es igual a la mejor selectividad posible. Para probar esto, analicemos lo siguiente:

**Selectividad = 1 / Cantidad\_de\_filas\_con\_valores\_distintos =** 1/100000 (Las llaves primarias son únicas)

**Mejor\_Selectividad = 1 / 100000 (El número de llaves es el mismo que el de filas de la tabla)**

La selectividad calculada para este ejemplo es de 0.00001 y la mejor selectividad es de 0.00001. Entre más se acerque la selectividad calculada a la mejor selectividad es más eficiente escoger dicho índice, y mucho mejor si es la misma como en el caso anterior. Los índices mejoraron el rendimiento porque…………tatatatata

**Documentación análisis para cada requerimiento**

-> Documentación del escenario de pruebas

-> Sentencia SQL que responde al requerimiento

-> Distribución de los datos con respecto a los parámetros de entrada.

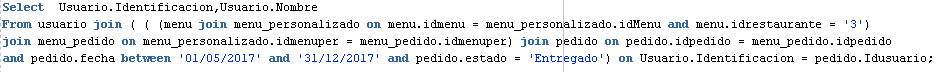
-> Planes de consulta obtenidos por Oracle para la ejecución del requerimiento. Agregar pantallazo del plan.

-> Tiempos obtenidos con la ejecución de cada uno de los planes. Desde el núcleo.

# Requerimiento 9:

## Escenario de prueba:

Se desea consultar la identificación y el nombre de los usuarios que al menos hicieron un pedido (que fue entregado) en el restaurante con id 3 en el rango de fechas '01/05/2017' y '31/12/2017’. Dicha consulta se realiza con la siguiente sentencia SQL:



Donde se tienen tres parámetros que son: el id del restaurante al cual se desea consultar el consumo, y las dos fechas que representan el rango por el que se desea hacer la consulta.

La anterior sentencia SQL retorna como respuesta:



## Para demostrar el funcionamiento de la anterior consulta veamos el siguiente ejemplo con algunos datos que se tenían en la tabla:

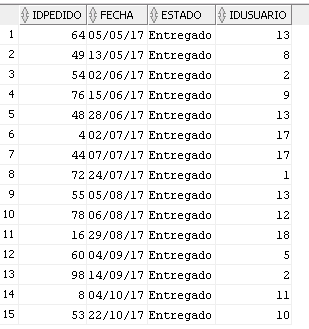
1. Se tienen 30 Usuarios registrados en la base de datos.





1. Vamos a consultar los pedidos que se han realizado en el rango de fechas establecido en la sentencia.





Se puede observar que se realizaron 15 pedidos (en estado entregado) dentro del rango de fechas '01/05/2017' y '31/12/2017’.

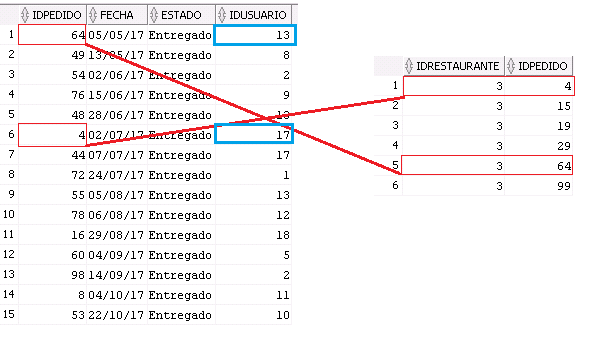
1. Ahora, debemos encontrar los pedidos que fueron realizados en el restaurante con id 3. Para esto se ejecuta la siguiente sentencia SQL:





Se pueden observar los ID de los 6 pedidos que se han realizado en el restaurante con id 3.

1. Entonces, vamos a comparar los resultados de la consulta (2) con los de la (3):

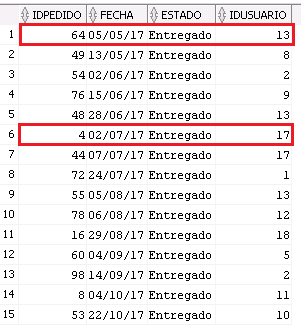


Se puede evidenciar que:

El pedido con id 64 fue realizado por el usuario con id 13 en el restaurante con id 3.

El pedido con id 4 fue realizado por el usuario con id 17 en el restaurante con id 3.

1. Finalmente, vamos a verificar que estos pedidos se hayan realizado en el rango de fechas establecido. Para ello se observa la tabla (2).

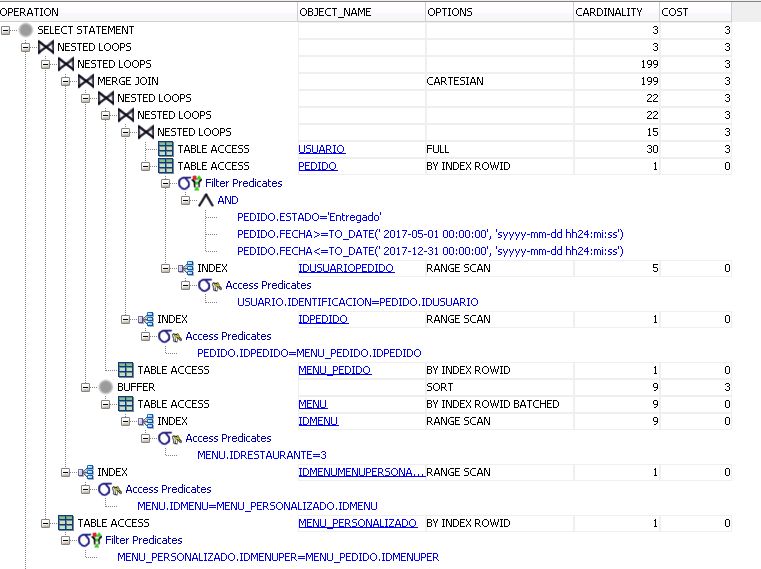


Se observa que estos pedidos fueron realizados entre las fechas '01/05/2017' y '31/12/2017’.

Entonces, se puede concluir que los usuarios con id 13 y 17, nombres Oberon y Beltran, hicieron al menos un pedido en el restaurante con id 3 en el rango de fechas '01/05/2017' y '31/12/2017’. Que coincide con la respuesta de la consulta planteada para la solución.



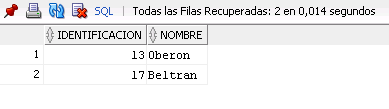
## Plan de consulta generado por SQL Developer



Se puede observar que en el plan que ejecutó SQL developer se utilizan varios nested loops, un merge joing y el acceso a los datos utilizando los índices, con un costo total de 3.

## Tiempo obtenido con la ejecución del plan

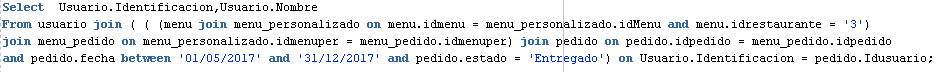
El tiempo que se tardó en ejecutar la sentencia y consultar las dos tuplas fue de 0.014 Segundos.



# Requerimiento 10:

## Escenario de prueba:

Se desea consultar la identificación y el nombre de los usuarios que no hicieron pedidos (que no fueron entregados) en el restaurante con id 3 en el rango de fechas '01/05/2017' y '31/12/2017’. Dicha consulta se realiza con la siguiente sentencia SQL:



Que retorna como respuesta:



# Análisis de eficiencia

-> Escenarios de datos que permitan validar diferentes selectividades.

-> Seleccionar un escenario de análisis y diseñe un plan de ejecución de consulta propuesto por el grupo.

-> Comparar el plan propuesto por el grupo y por Oracle.

3- Construcción de la aplicación y análisis de resultados.

* Diseño del escenario de pruebas de eficiencia. Cargar los datos suficientes para realizar un análisis de eficiencia. Estado de la base de datos.
* Documentar como fue el proceso de carga de datos, como se realizó como se logró el volumen de datos esperado.
* Ajustes servicios REST para los nuevos requerimientos.
* Cambios y desarrollo de las transacciones en RotondAndesMaster.
* Cambios en los Dao.
* Análisis del proceso de optimización y el modelo de ejecución de consultas. Analice la diferencia entre la ejecución de consultas delegada al manejador de bases de datos como Oracle compárelo con una ejecución donde la aplicación trae los datos a memoria principal y resuelve con instrucciones de control (if,while,etc.), los operadores involucrados en las consultas como joins, selecciones y proyecciones.