Universidad de Los Andes.  
Departamento de Ingeniería de Sistemas y Computación.  
Sistemas Transaccionales.  
Iteración IV “RotondAndes”

Christian Chavarro Espejo 201613724  
Juan Sanmiguel Mateus 201617603

1 (1 %) Análisis Ajuste el modelo del mundo (modelo conceptual: diagrama de clases UML) propuesto en la iteración anterior, si lo requiere. Indique cuáles clases del modelo del mundo fueron actualizadas o creadas en esta iteración.

No fueron realizados cambios en el modelo de mundo o esquema de las tablas, ellas continúan como en las iteraciones anteriores.

2. Diseño de la aplicación

A partir del diseño existente, analice el impacto que representa la introducción de los nuevos requerimientos y restricciones a nivel del modelo conceptual. Realice los cambios necesarios en su modelo relacional para respetar las reglas de negocio y asegurar la calidad del mismo. Tenga en cuenta los comentarios recibidos en la sustentación de los talleres anteriores. Documente el diseño y las decisiones tomadas para crear los elementos de la base de datos que da el respaldo de persistencia a la aplicación, a partir del modelo conceptual.

* Sea claro en mencionar explícitamente los cambios relevantes entre su diseño entregado en iteraciones anteriores y este

No fueron realizados cambios en el modelo de mundo o esquema de las tablas, ellas continúan como en las iteraciones anteriores.

El aumento en el tamaño de los datos, comparado con un mal manejo de los índices, podría afectar el desempeño de la aplicación.

Diseño físico. Analice la aplicación completa resultante de la iteración anterior y de los nuevos requerimientos para realizar el diseño físico correspondiente. En particular, diseñe los índices necesarios para el adecuado rendimiento global de la aplicación.

Documente su diseño físico

Justifique la selección de índices desde el punto de vista de cada uno de los requerimientos funcionales. Indique claramente cuál es el tipo de índice utilizado (B+, Hash, …, primario, secundario) y tenga en cuenta el costo de almacenamiento y mantenimiento asociado a los índices

Según su modelo de datos, para los índices creados de forma automática por Oracle

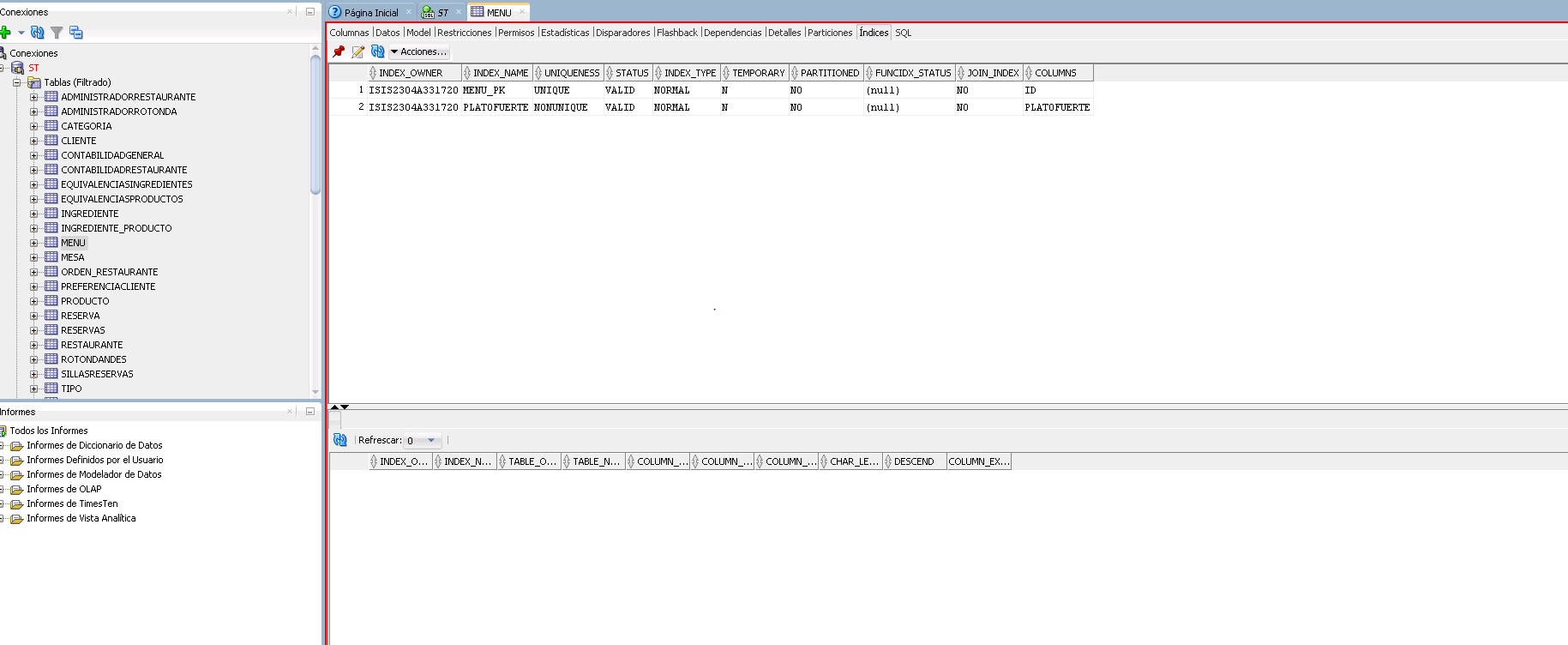
Incluya una foto de pantalla con la información generada por Oracle asociada a los índices existentes.

Analice los índices encontrados. Específicamente, analice por qué fueron creados por Oracle y si ayudan al rendimiento de los requerimientos funcionales.

Índices

Los índices, principalmente fueron creados sobre las tablas menú y orden\_restaurante. Ahora, serán descritos los índices por cada una de estas tablas, además se especificará cada decisión de selección de los índices basándonos en ejemplos de los RRC9-12.

MENU

  
Imagen 1: índices de la tabla menú.

Índices de la tabla.

MENUPK: Es el índice por defecto creado por Oracle, es la PK de la tabla de menú.

PLATOFUERTE: es un índice sobre la columna de plato fuerte de cada menú. El presenta datos repetidos, pero es un buen índice debido a que es seleccionado en los planes para realizar los joins de los requerimientos 11 y 12. Además, la cantidad de productos que pueden corresponder a esta columna es alta. La selectividad si bien, es intermedia, un índice de este tipo en estas consultas, ayudará a minimizar los costos en los planes.

Se espera que este índice corresponda a una implementación de hash. Además, él tiene datos repetidos, no puede ser el índice primario de esta tabla.

Índices Orden restaurante.

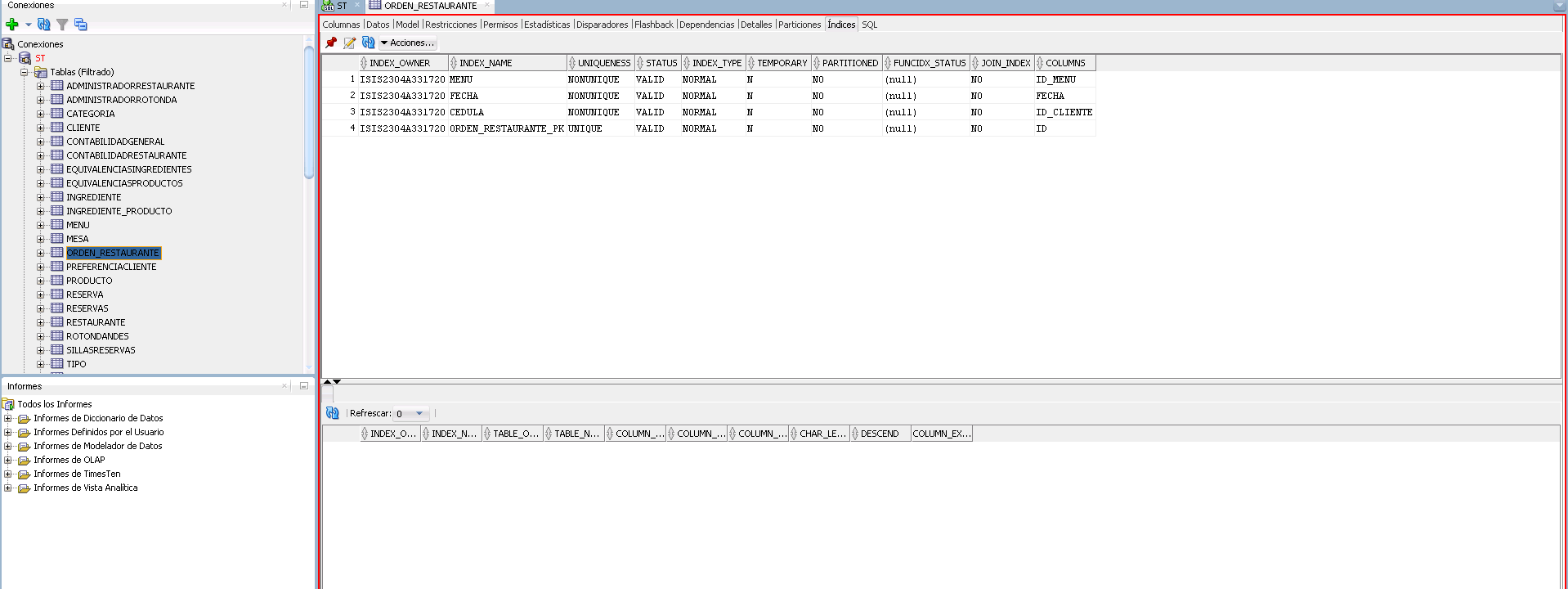


Imagen 2: índices de la tabla orden restaurante.

Orden Restaurante PK: Es el identificador primario, se espera que este bajo una estructura de B+, debido al volumen de órdenes, aproximadamente dos millones.

Orden Restaurante fecha: Es un índice sobre la columna de fecha en cada orden. Este índice no es único, pero es un buen índice debido a que es seleccionado en los planes para realizar los joins de los requerimientos. Además, la cantidad de fechas distintas es alta, mínimo hay 5 años en estas órdenes. Ahora, si bien es intermedia la selectividad, un índice de este tipo en estas consultas, ayudará a minimizar los costos en los planes.

Se espera una implementación bajo B+ o hash. También se tiene una implementación de combinación entre estas.

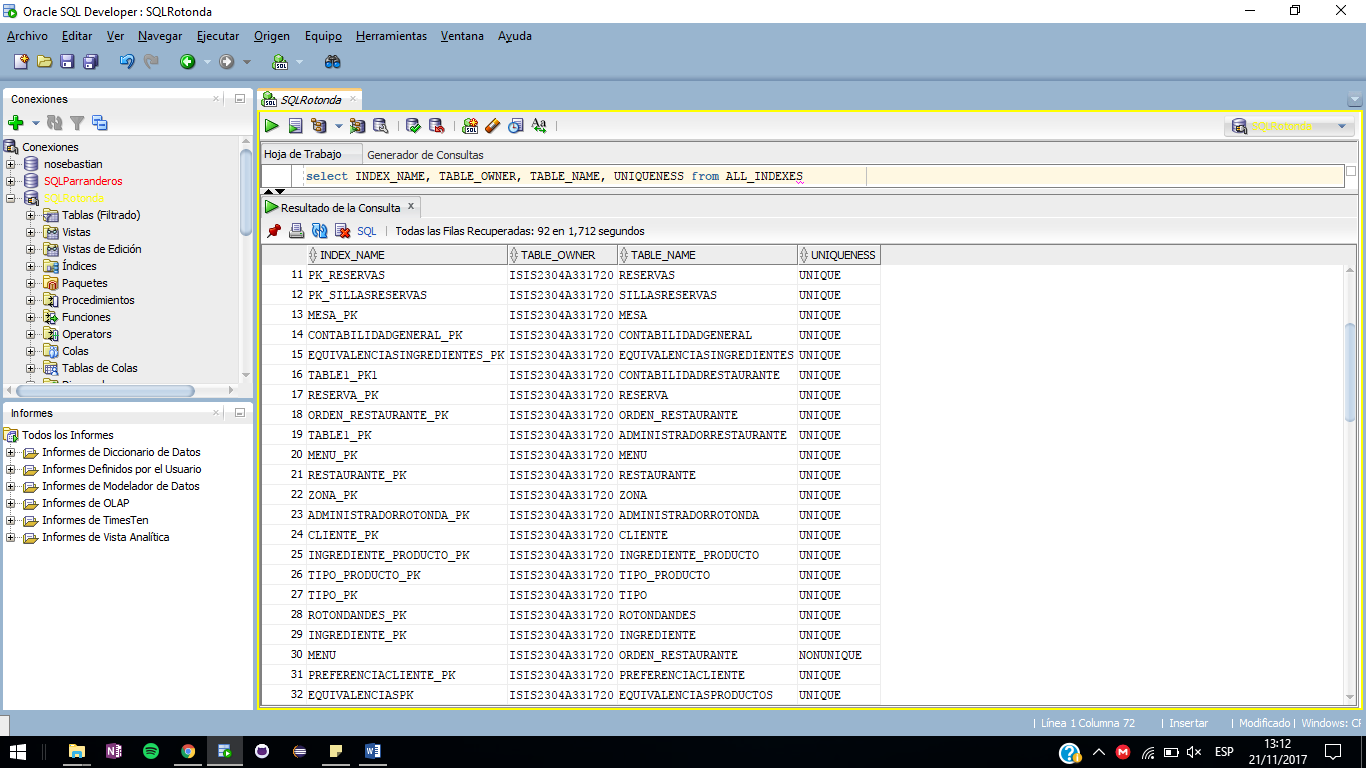
Orden Restaurante menú: Es un índice sobre la columna de menú en cada orden. Este índice no es único, pero es un buen índice debido a que es seleccionado en los planes para realizar los joins de los requerimientos. Además, la cantidad de menús, si bien no es alta, hay unos 100000 menús, la selectividad será muy baja, pero se hace bueno para un índice tipo hash. Este índice, es usado en los RFC12-11.

Se espera una implementación de hash.

Orden Restaurante cedula: Cada orden, tiene asociada una cedula de cliente, realizar un índice mediante este criterio, facilitará la obtención rápida de las ordenes de un cliente, esto facilita las búsquedas de los RFC9 y 10. Hay 1000000 de clientes y 2000000 de órdenes, la selectividad teórica será de un 50% realizando cálculos optimistas.

Se espera una implementación de hash.

Todos los índices de la DB.



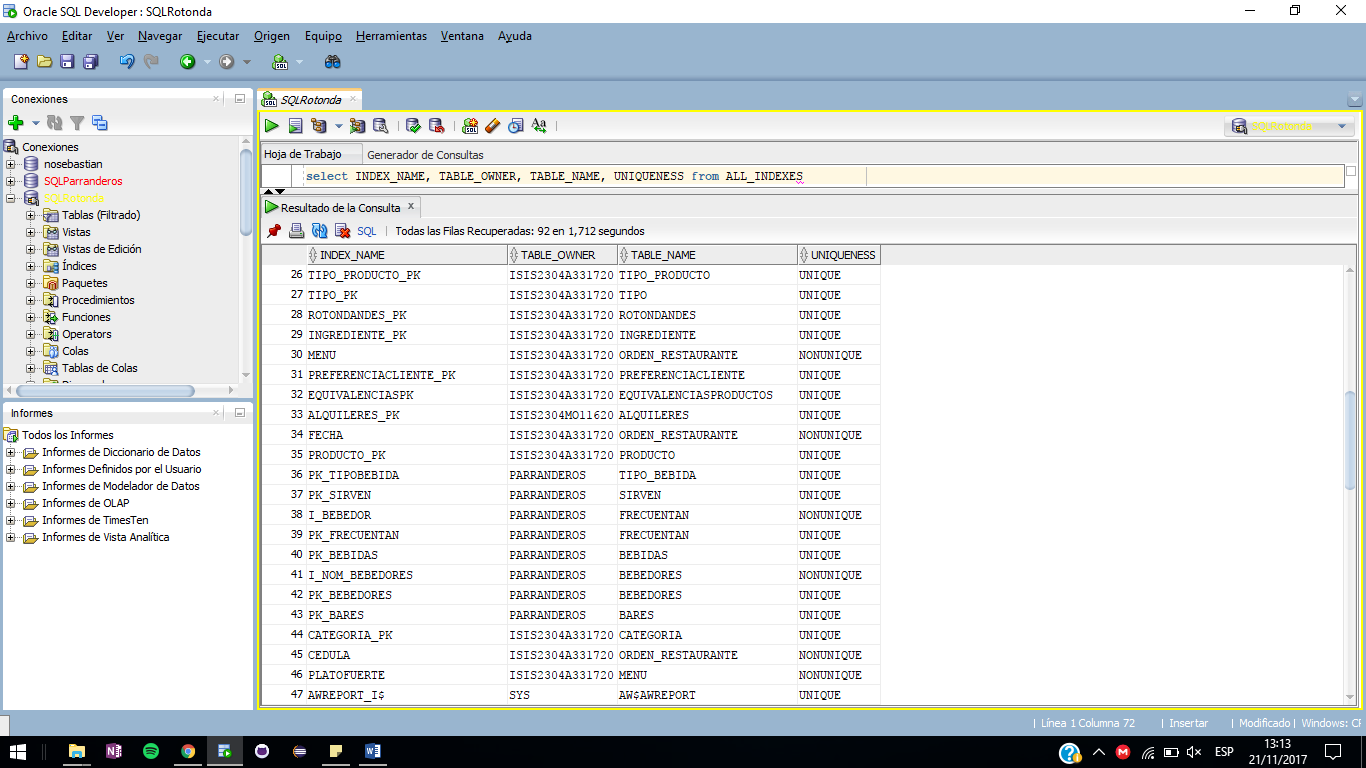


Imagen 3: todos los índices asociados al usuario de la DB.

Al analizar los índices ya existentes, solo se crearon los ya descritos anteriormente. Los otros criterios de las consultas o bien, son llaves primarias de las tablas o poseen bajas selectividades.

Costos de los índices.

Los costos de inserción en arboles b+ son logarítmicos en y en un hash son de acuerdo al número de colisiones.

Documente plenamente el análisis realizado, incluyendo los siguientes aspectos para cada requerimiento funcional de consulta solicitado

• Documentación del escenario de pruebas

* Sentencias SQL que responden el requerimiento y que fueron analizadas.
* Distribución de los datos con respecto a los parámetros de entrada utilizados en el requerimiento funcional. En particular se quiere un análisis de distribución que permita ver cómo puede cambiar el tamaño de la respuesta según el valor de los parámetros utilizados y la configuración de los datos de prueba.
* Valores de los parámetros utilizados en el análisis y que constituyen diferenciadores en los planes de ejecución obtenidos.
* Planes de consulta obtenidos en Oracle para la ejecución del requerimiento. Para ello, documente con una foto de pantalla los planes de consulta obtenidos en SQLDevelopper.
* Tiempos obtenidos con la ejecución de cada uno de los planes. Estos tiempos son medidos desde el núcleo de la aplicación, es decir, no incluyen la parte de interacción con el usuario, ingreso de datos ni despliegue de resultados.

NOTA: El porcentaje de evaluación correspondiente a cada uno de los requerimientos solicitados es proporcional al número de los requerimientos.

NOTA: La nota para cada uno de los requerimientos depende de los escenarios de ejecución definidos.

RFC9

**Sentencia SQL**

* Si la realiza un administrador, la sentencia es la siguiente:

SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO

FROM (CLIENTE RIGHT JOIN (ORDEN\_RESTAURANTE RIGHT JOIN (MENU

RIGHT JOIN PRODUCTO ON MENU.NOMBRE\_RESTAURANTE =

PRODUCTO.NOMBRE\_RESTAURANTE) ON MENU.ID =

ORDEN\_RESTAURANTE.ID\_MENU) ON CLIENTE.CEDULA =

ORDEN\_RESTAURANTE.ID\_CLIENTE)

WHERE ORDEN\_RESTAURANTE.ID\_CLIENTE IS NOT NULL AND FECHA >= ?

AND FECHA < ? AND MENU.NOMBRE\_RESTAURANTE = ?

ORDER BY ?;

* Si la realiza un cliente registrado, la sentencia es la siguiente:

SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO

FROM (CLIENTE RIGHT JOIN (ORDEN\_RESTAURANTE RIGHT JOIN (MENU

RIGHT JOIN PRODUCTO ON MENU.NOMBRE\_RESTAURANTE =

PRODUCTO.NOMBRE\_RESTAURANTE) ON MENU.ID =

ORDEN\_RESTAURANTE.ID\_MENU) ON CLIENTE.CEDULA =

ORDEN\_RESTAURANTE.ID\_CLIENTE)

WHERE ORDEN\_RESTAURANTE.ID\_CLIENTE IS NOT NULL AND FECHA >= ?

AND FECHA < ? AND MENU.NOMBRE\_RESTAURANTE = ? AND CEDULA = ?

ORDER BY ?;

**Path:**

GET/administradorrestaurante/consultarConsumo/{restaurante}/{fechaMin}/{fechaMax}/{orderBy}

GET /cliente/consultarConsumo/{ restaurante }/{cedula}/{fechaMin}/{fechaMax}/{orderBy}

restaurante corresponde al nombre de restaurante a considerar en la consulta.

fechaMin corresponde a la fecha mínima a considerar en la consulta.

fechaMax corresponde a la fecha máxima a considerar en la consulta.

orderBy corresponde al parámetro por el cual se va a hacer el order by de la consulta.

cedula corresponde a la cedula del cliente que hace la consulta.

**Ejemplos distribución parámetros**

Actualmente, nuestra base de datos0020cuenta con alrededor de 650,000 ordenes (tuplas en la tabla ORDEN\_RESTAURANTE), alrededor de 6,000 menús (tuplas en la tabla MENU), alrededor de 6,500 productos (tuplas en la tabla PRODUCTO) y cuenta con más de 1,000,000 de clientes (tuplas en la tabla CLIENTE). Resaltamos la información de estas tablas, debido a que son las que usamos para el desarrollo de este requerimiento. Las fechas de las órdenes se explican a continuación:

* **Distribución:** Tenemos ordenes con fechas que van desde el 01/01/01 (primero de enero del año 2001) hasta el 31/12/18 (treintaiuno de diciembre del año 2018). Como dijimos anteriormente, contamos con alrededor de 650,000 ordenes, distribuidas lo más uniformemente que nos fue posible entre este rango de fechas.
* **Selectividad máxima:** Las fechas de las ordenes van desde 01/01/01 (primero de enero del año 2001) hasta el 31/12/18 (diecinueve de noviembre del presente año), es decir, si pasamos como parámetros estas fechas, o antes que el 01/01/01 y posterior a 31/12/18, sabemos que esperamos considerar todas las ordenes y que obtendríamos los datos de todos los usuarios con órdenes en esta fecha, si un cliente no tiene una orden en esta fecha, no es considerado en la respuesta.
* **Selectividad mínima o nula:** Como se dijo en el punto anterior, las fechas van entre 01/01/01 y 31/12/18, por lo que si se escoge un rango de fechas tal que la intersección entre los dos sea vacía. Esperamos que la respuesta sea vacía, ya que no hay órdenes en la fecha por parámetro.
* **Selectividad media:** Debido a la distribución uniforme con la que creamos las tuplas de la base de datos, podemos esperar que la selectividad variara según el rango que decidamos, es decir, si escogemos un rango de fechas que represente alrededor de la mitad del rango total, la selectividad esperada de nuestra solicitud sea cerca del 50% del total de datos de la base que cumplen la sentencia.
* **Selectividad variable:** Como consecuencia de lo comentado anteriormente (distribución uniforme de las fechas, selectividad del ~50% al partir el rango a ~50%), podemos considerar también el hecho de que, si reducimos el rango de búsqueda a un valor X, tal que 0≤X≤100, entonces la selectividad será igual a X%, o variara bastante cerca a X%.

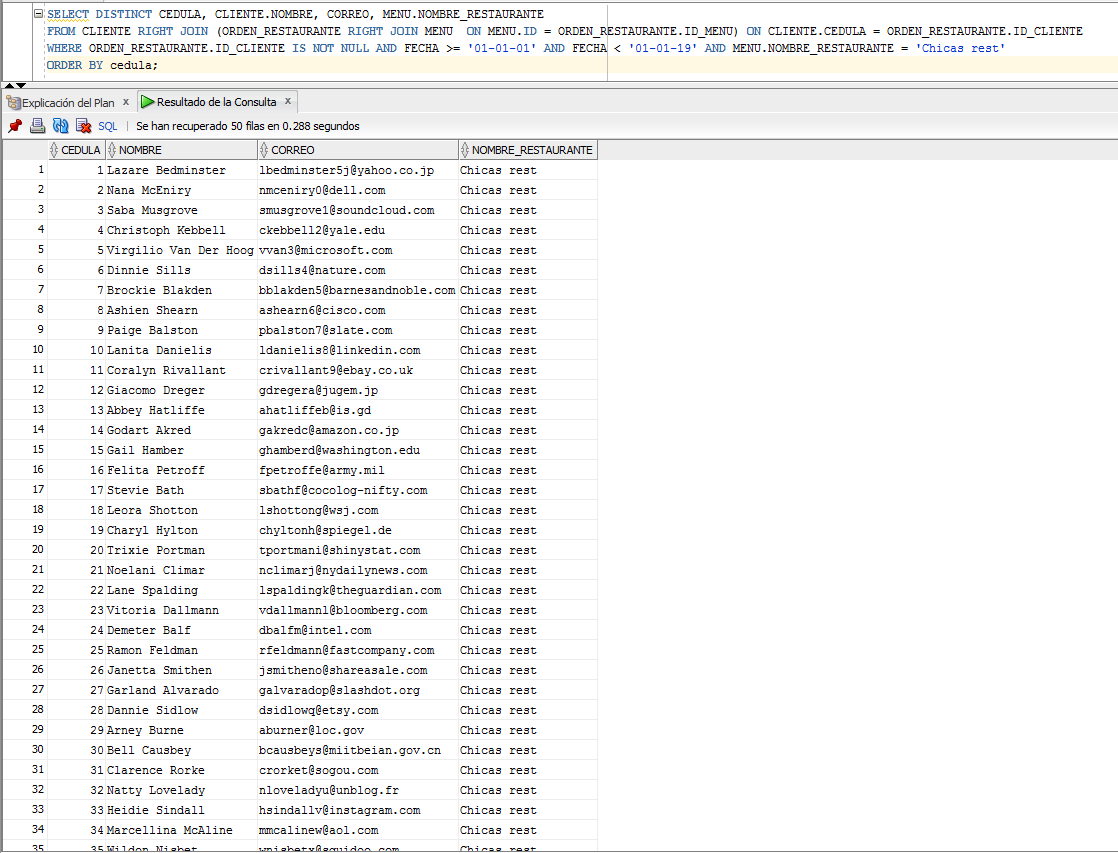
**Valores de parámetros para ejecución**

* + **Nombre restaurante:** Chicas rest

**Fecha mínima:** 01/01/01

**Fecha máxima:** 31/12/18

**Tiempo ejecución:** 0.288 segundos



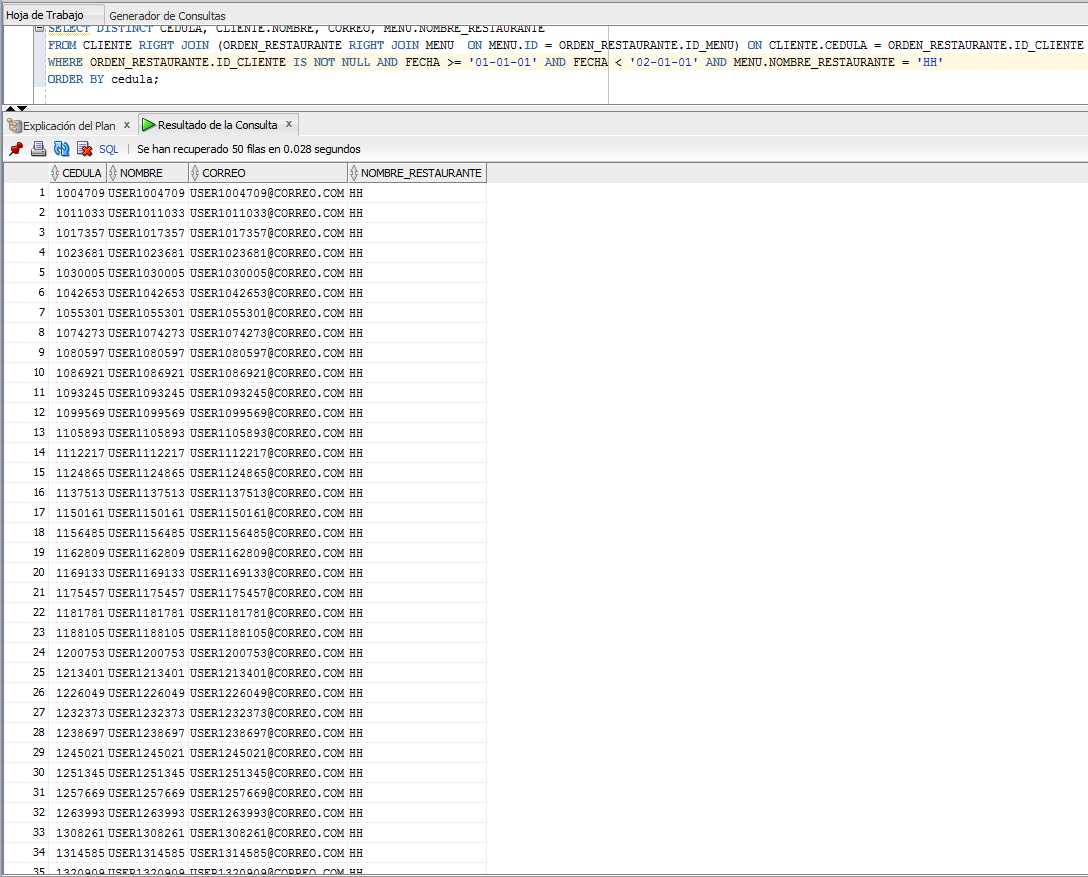
Este rango de fechas representa considerar todas las órdenes existentes en la rotonda.

* + **Nombre restaurante:** HH

**Fecha mínima:** 01/01/01

**Fecha máxima:** 02/01/01

**Tiempo ejecución:** 0.028 segundos



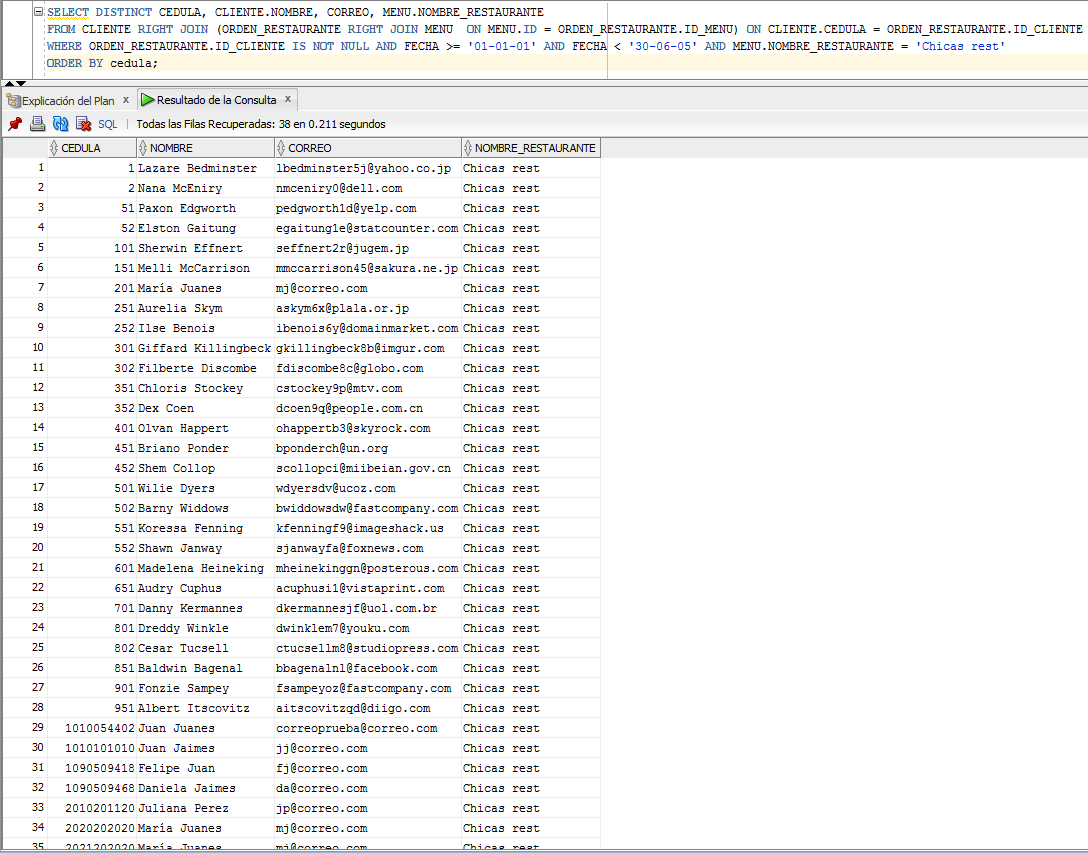
Este rango de fechas representa considerar las órdenes realizadas en un solo día, por lo que la selectividad es mucho más baja, comparada con la de todas las órdenes existentes.

* + **Nombre restaurante:** Chicas rest

**Fecha mínima:** 01/01/01

**Fecha máxima:** 30/06/05

**Tiempo ejecución:** 0.211 segundos



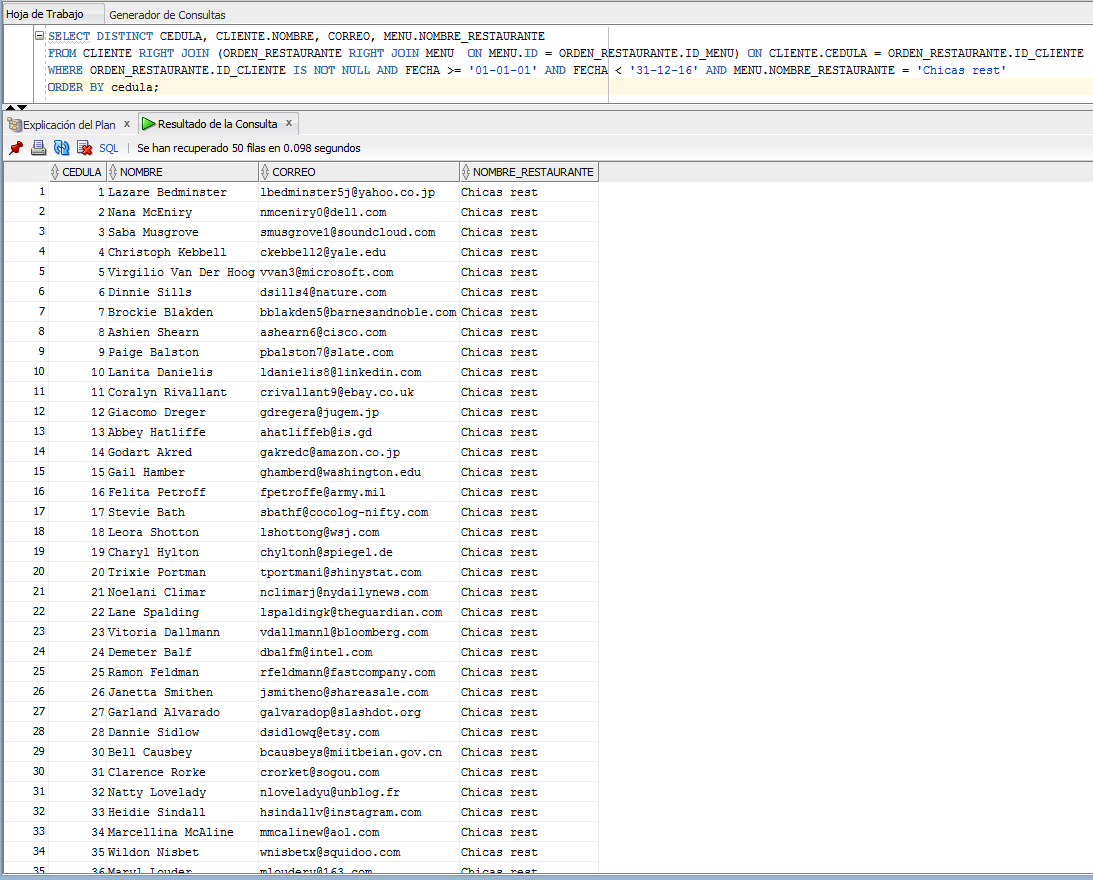
Este rango de fechas representa considerar aproximadamente el 25% de órdenes, del total de órdenes existentes.

* + **Nombre restaurante:** Chicas rest

**Fecha mínima:** 01/01/01

**Fecha máxima:** 31/12/16

**Tiempo ejecución:** 0.098 segundos



Este rango de fechas representa considerar aproximadamente el 75% de órdenes, del total de órdenes existentes.

RFC10

**Sentencia SQL**

* Si la realiza un administrador, la sentencia es la siguiente:

SELECT CEDULA, NOMBRE, CORREO

FROM CLIENTE

MINUS

SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO

FROM CLIENTE RIGHT JOIN (ORDEN\_RESTAURANTE RIGHT JOIN MENU ON

MENU.ID = ORDEN\_RESTAURANTE.ID\_MENU) ON CLIENTE.CEDULA =

ORDEN\_RESTAURANTE.ID\_CLIENTE

WHERE ORDEN\_RESTAURANTE.ID\_CLIENTE IS NOT NULL AND FECHA >= ? AND

FECHA < ? AND MENU.NOMBRE\_RESTAURANTE = ?

ORDER BY ?;

* Si la realiza un cliente registrado, la sentencia es la siguiente:

SELECT CEDULA, NOMBRE, CORREO

FROM CLIENTE

MINUS

SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO

FROM CLIENTE RIGHT JOIN (ORDEN\_RESTAURANTE RIGHT JOIN MENU ON

MENU.ID = ORDEN\_RESTAURANTE.ID\_MENU) ON CLIENTE.CEDULA = ORDEN\_RESTAURANTE.ID\_CLIENTE

WHERE ORDEN\_RESTAURANTE.ID\_CLIENTE IS NOT NULL AND FECHA >= ? AND

FECHA < ? AND MENU.NOMBRE\_RESTAURANTE = ? AND CEDULA = ?

ORDER BY ?;

**Path:**

GET/administradorrestaurante/consultarNoConsumo/{restaurante}/{fechaMin}/{fechaMax}/{orderBy}

GET /cliente/ consultarNoConsumo /{ restaurante }/{cedula}/{fechaMin}/{fechaMax}/{orderBy}

restaurante corresponde al nombre de restaurante a considerar en la consulta.

fechaMin corresponde a la fecha mínima a considerar en la consulta.

fechaMax corresponde a la fecha máxima a considerar en la consulta.

orderBy corresponde al parámetro por el cual se va a hacer el order by de la consulta.

cedula corresponde a la cedula del cliente que hace la consulta.

**Ejemplos distribución parámetros**

Actualmente, nuestra base de datos cuenta con alrededor de 650,000 ordenes (tuplas en la tabla ORDEN\_RESTAURANTE), alrededor de 6,000 menús (tuplas en la tabla MENU), alrededor de 6,500 productos (tuplas en la tabla PRODUCTO) y cuenta con más de 1,000,000 de clientes (tuplas en la tabla CLIENTE). Resaltamos la información de estas tablas, debido a que son las que usamos para el desarrollo de este requerimiento. Las fechas de las órdenes se explican a continuación:

* **Distribución:** Tenemos ordenes con fechas que van desde el 01/01/01 (primero de enero del año 2001) hasta el 31/12/18 (treintaiuno de diciembre del año 2018). Como dijimos anteriormente, contamos con alrededor de 650,000 ordenes, distribuidas lo más uniformemente que nos fue posible entre este rango de fechas.
* **Selectividad variable:** Las fechas de las ordenes van desde 01/01/01 (primero de enero del año 2001) hasta el 31/12/18 (diecinueve de noviembre del presente año), es decir, si pasamos como parámetros estas fechas, o antes que el 01/01/01 y posterior a 31/12/18, sabemos que esperamos considerar todas las ordenes y que obtendríamos los datos de todos los usuarios sin órdenes en esta fecha, si un cliente tiene una orden en esta fecha, no es considerado en la respuesta.
* **Selectividad máxima:** Como se dijo en el punto anterior, las fechas van entre 01/01/01 y 31/12/18, por lo que si se escoge un rango de fechas tal que la intersección entre los dos sea vacía. Esperamos que la respuesta sea máxima, ya que no hay órdenes en la fecha por parámetro.
* **Selectividad media:** Debido a la distribución uniforme con la que creamos las tuplas de la base de datos, podemos esperar que la selectividad variara según el rango que decidamos, es decir, si escogemos un rango de fechas que represente alrededor de la mitad del rango total, la selectividad esperada de nuestra solicitud sea cerca del 50% del total de datos.
* **Selectividad variable:** Como consecuencia de lo comentado anteriormente (distribución uniforme de las fechas, selectividad del ~50% al partir el rango a ~50%), podemos considerar también el hecho de que, si reducimos el rango de búsqueda a un valor X, tal que 0≤X≤100, entonces la selectividad será igual a X%, o variara bastante cerca a X%.

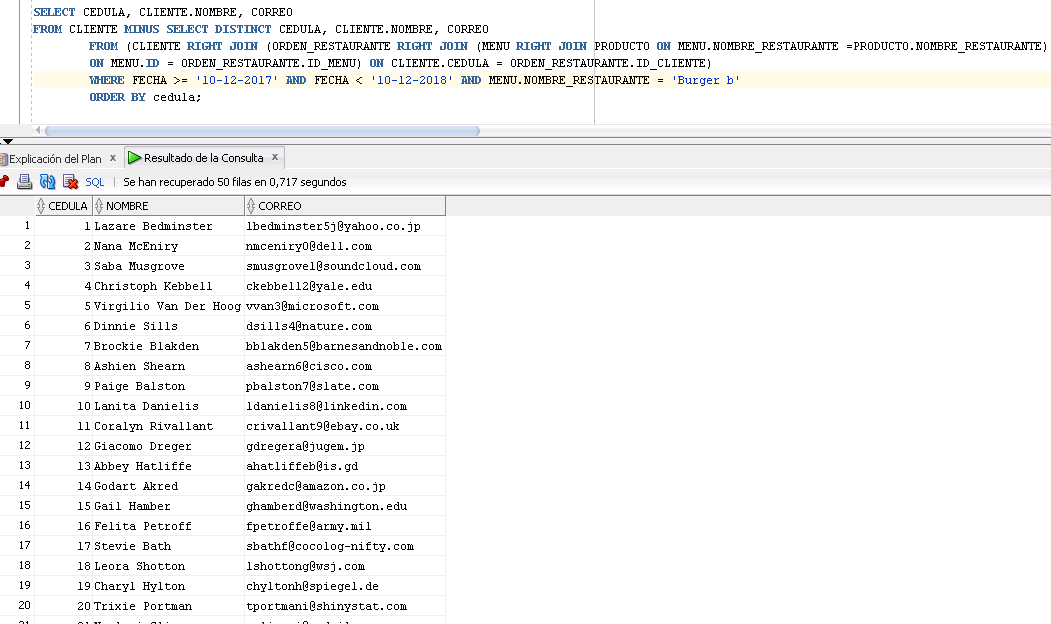
**Valores de parámetros para diferentes planes de ejecución**

* + **Nombre restaurante:** Burger b

**Fecha mínima:** 10/12/17

**Fecha máxima:** 10/12/18

**Tiempo ejecución:** 0.717 segundos

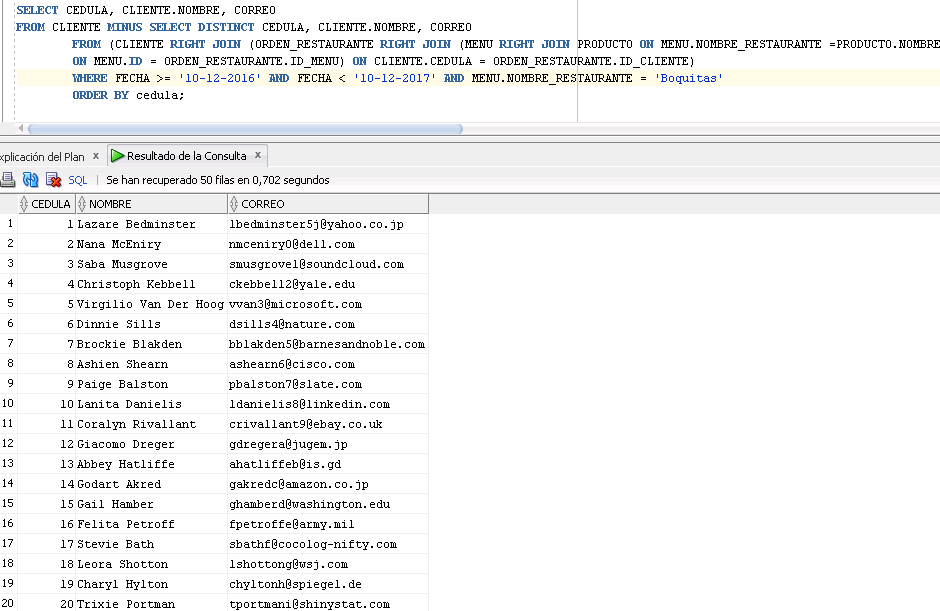


* + **Nombre restaurante:** Chicas rest

**Fecha mínima:** 10/12/16

**Fecha máxima:** 10/12/17

**Tiempo ejecución:** 0.702 segundos

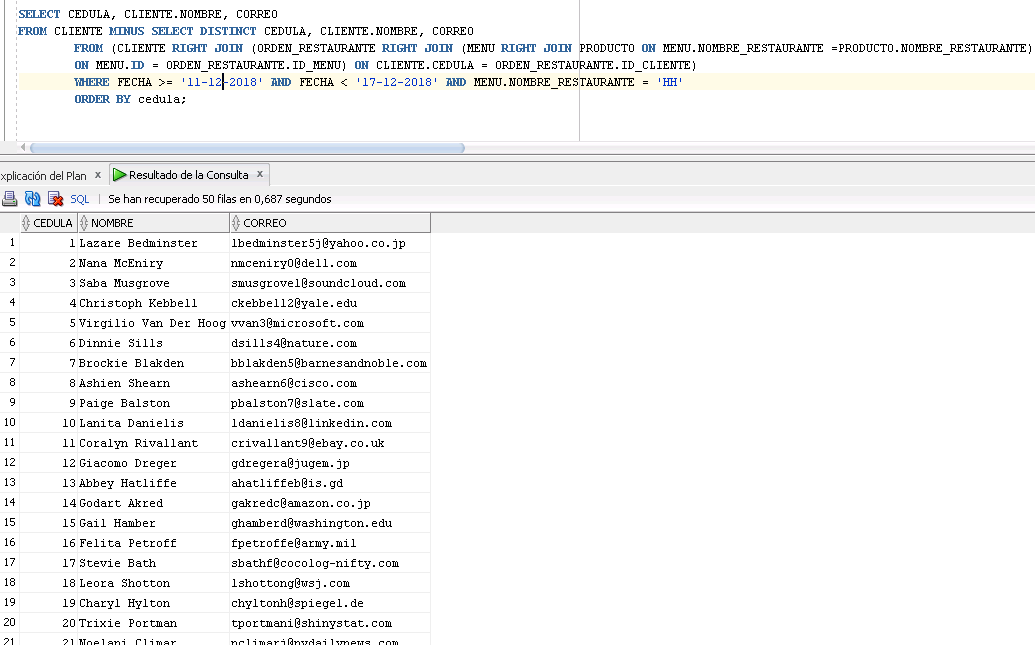


* + **Nombre restaurante:** Chicas rest

**Fecha mínima:** 11/12/18

**Fecha máxima:** 17/12/18

**Tiempo ejecución:** 0.687 segundos

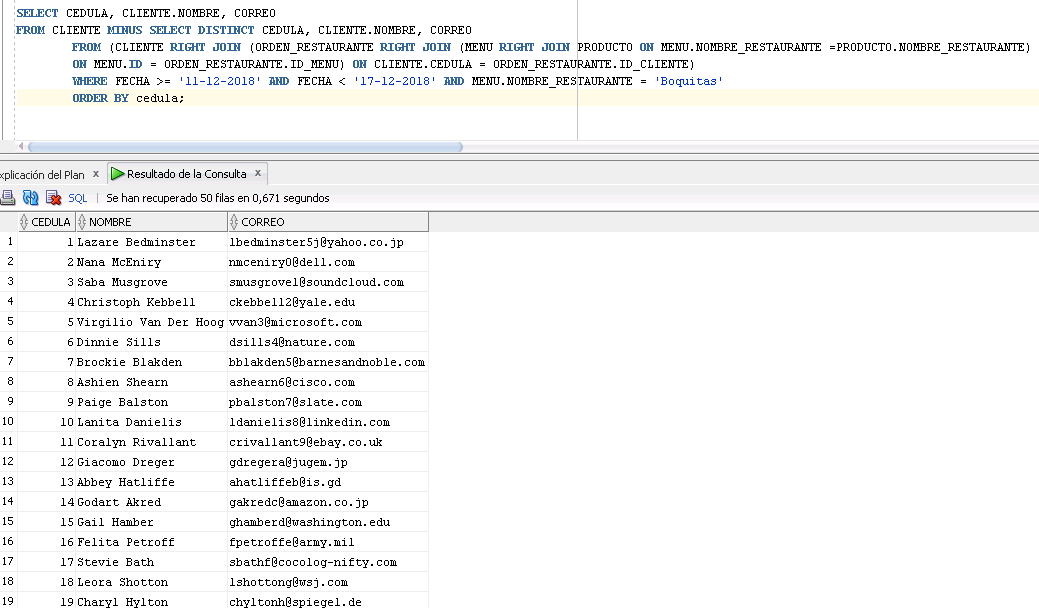


* + **Nombre restaurante:** Chicas rest

**Fecha mínima:** 11/12/18

**Fecha máxima:** 17/12/18

**Tiempo ejecución:** 0.671 segundos



RFC11

Sentencia SQL:

SELECT \* FROM (SELECT PRODUCTO.ID,PRODUCTO.NOMBRE\_RESTAURANTE, ORDEN\_RESTAURANTE.FECHA, COUNT(\*) AS CONTEO FROM PRODUCTO LEFT JOIN (MENU LEFT JOIN ORDEN\_RESTAURANTE ON MENU.ID = ORDEN\_RESTAURANTE.ID\_MENU) ON (MENU.POSTRE=PRODUCTO.ID OR MENU.PLATOFUERTE = PRODUCTO.ID OR MENU.ACOMPANIAMIENTO = PRODUCTO.ID OR MENU.BEBIDA=PRODUCTO.ID OR MENU.ENTRADA=PRODUCTO.ID) WHERE ORDEN\_RESTAURANTE.FECHA >='10-12-2018' AND ORDEN\_RESTAURANTE.FECHA <='17-12-2018' GROUP BY PRODUCTO.ID,PRODUCTO.NOMBRE\_RESTAURANTE, ORDEN\_RESTAURANTE.FECHA ORDER BY ORDEN\_RESTAURANTE.FECHA)  
WHERE CONTEO>=3 OR (CONTEO<2 AND CONTEO >0);

Distribución de los datos con respecto a los parámetros de entrada utilizados en el requerimiento funcional. En particular se quiere un análisis de distribución que permita ver cómo puede cambiar el tamaño de la respuesta según el valor de los parámetros utilizados y la configuración de los datos de prueba.

De acuerdo al rango de fechas dado por parámetro, el tamaño de la respuesta y el tiempo de ejecución de la sentencia puede variar. El valor más alto obtenido es de 0,8 segundos, los valores promedios, rondan los 0,4.

Valores de los parámetros utilizados en el análisis y que constituyen diferenciadores en los planes de ejecución obtenidos.

GET /administradorrotonda/consultarFuncionamiento/fechaMin/fechaMax/

GET [/administradorrotonda/consultarFuncionamiento/9-12-18/29-12-18/](http://localhost:8080/RotondAndes/rest/administradorrotonda/consultarFuncionamiento/9-12-18/29-12-18/)

Estos, son la url principal junto y el verbo para pruebas de Postman.

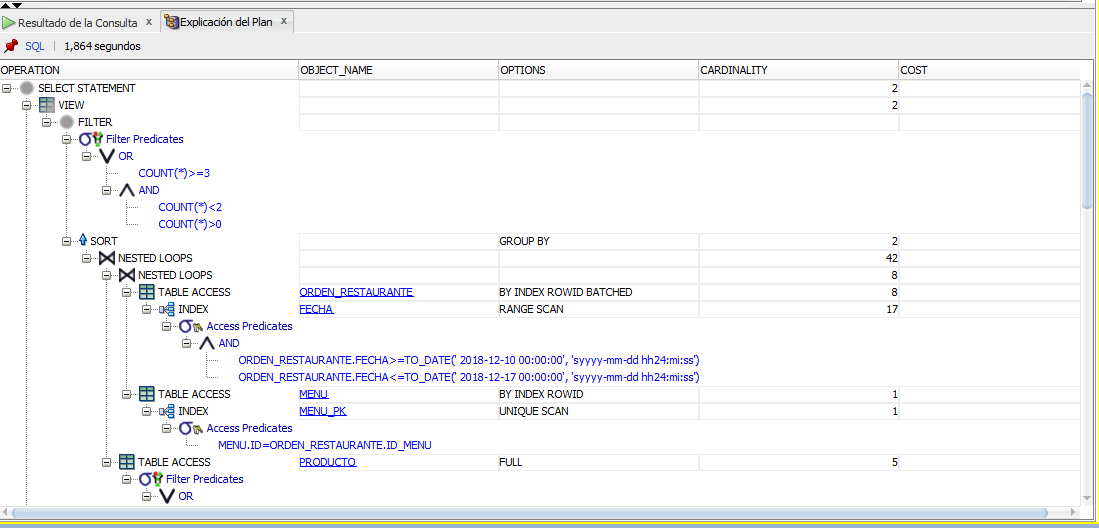
Algunos parámetros validos son:

/9-12-18/29-12-18/ - /9-2-18/29-12-18/- /9-12-10/29-12-18/- /9-12-13/29-12-16/

Donde básicamente, se modifica la fecha de inicio o final del rango de búsqueda.

Como se describió anteriormente, de acuerdo al rango de fechas, varían las respuestas, estas, vienen ordenadas por fecha descendentemente.

Planes de consulta obtenidos en Oracle para la ejecución del requerimiento. Para ello, documente con una foto de pantalla los planes de consulta obtenidos en SQLDevelopper.



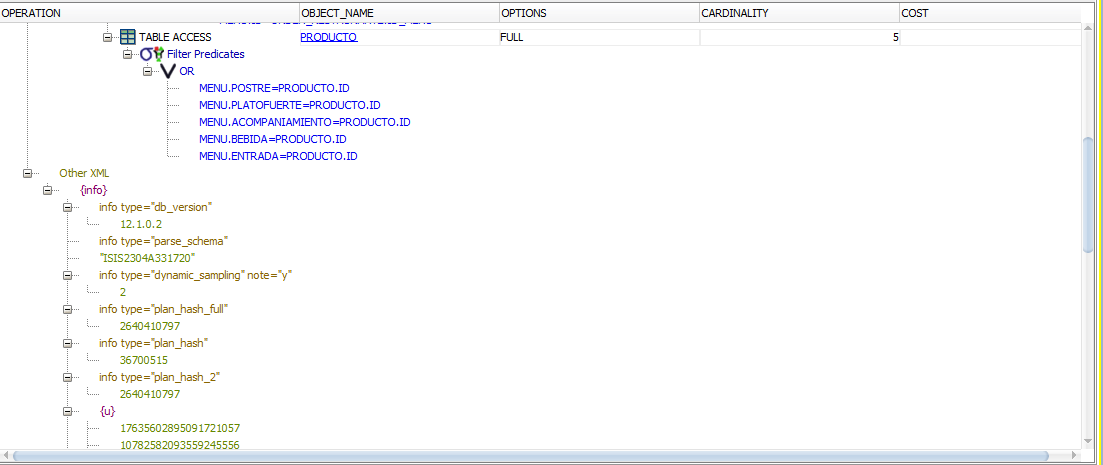
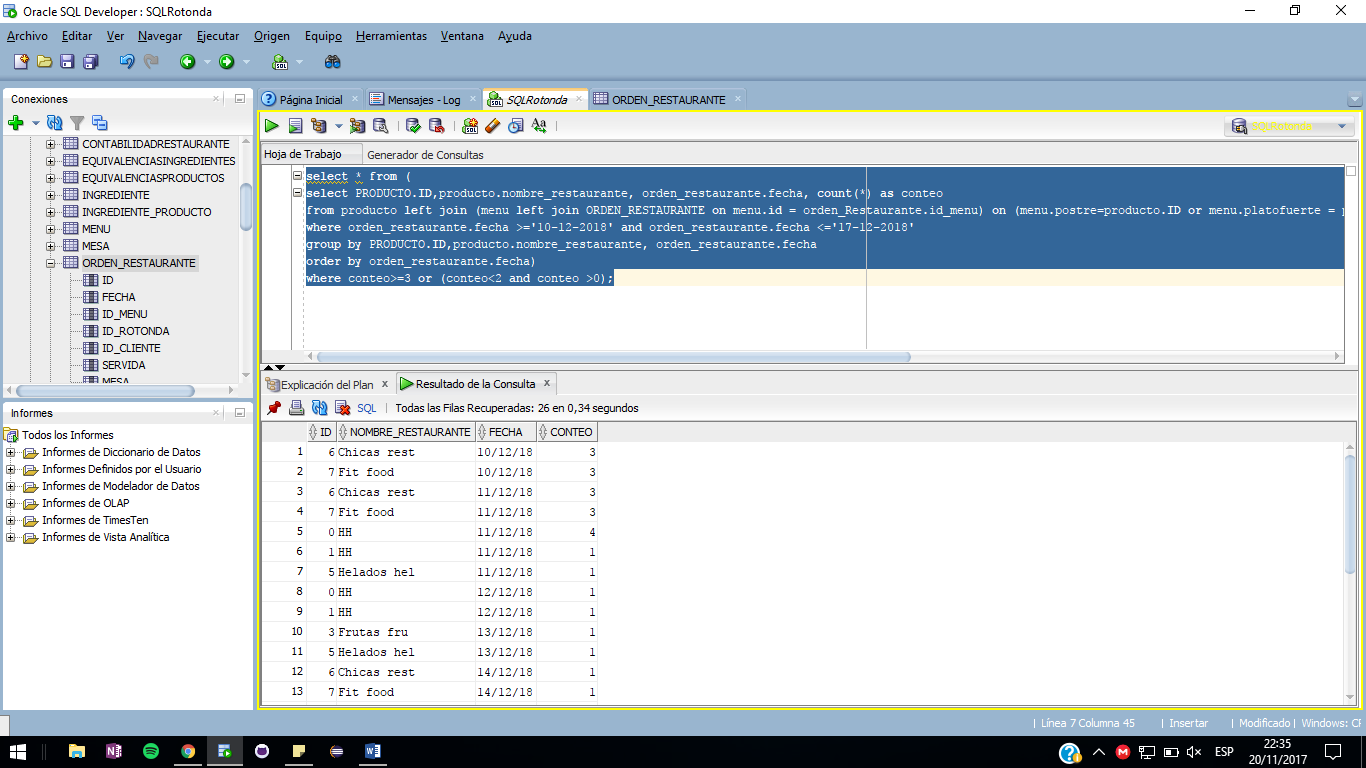
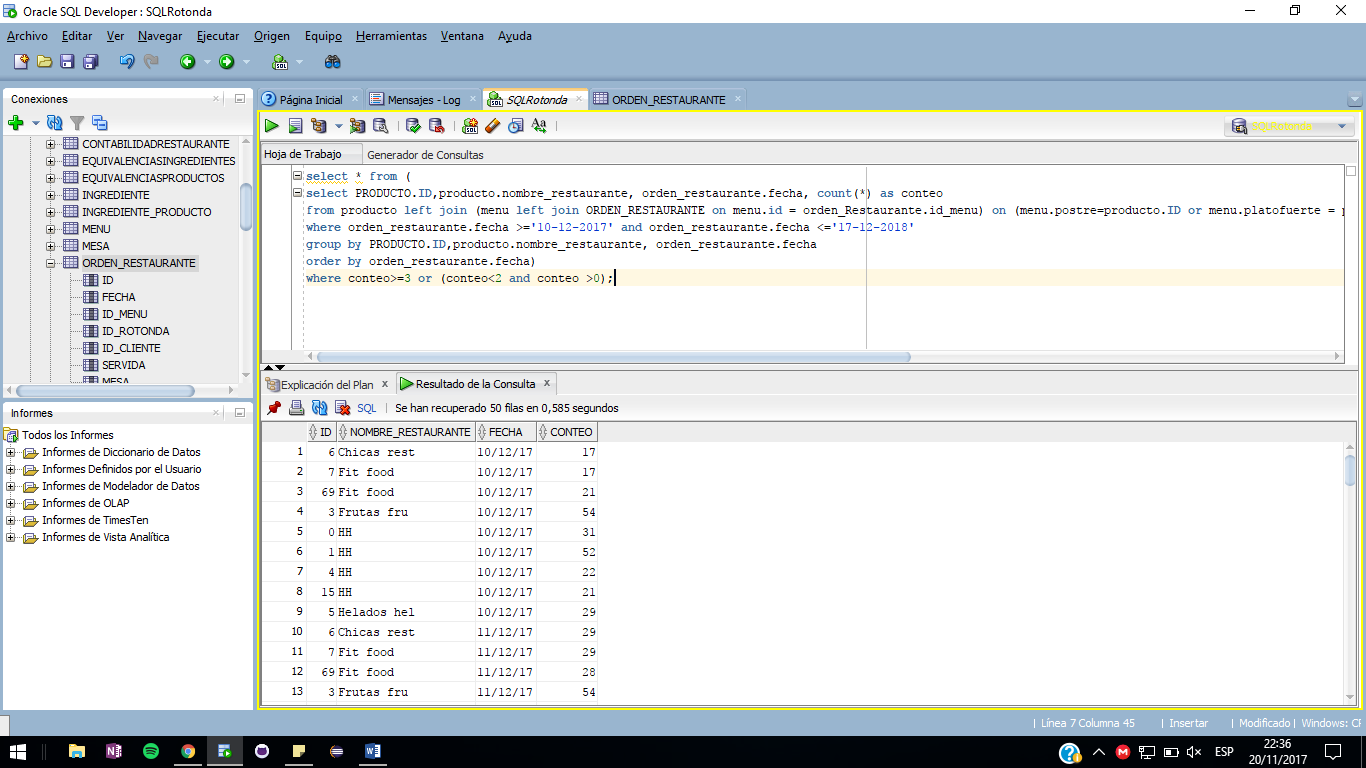


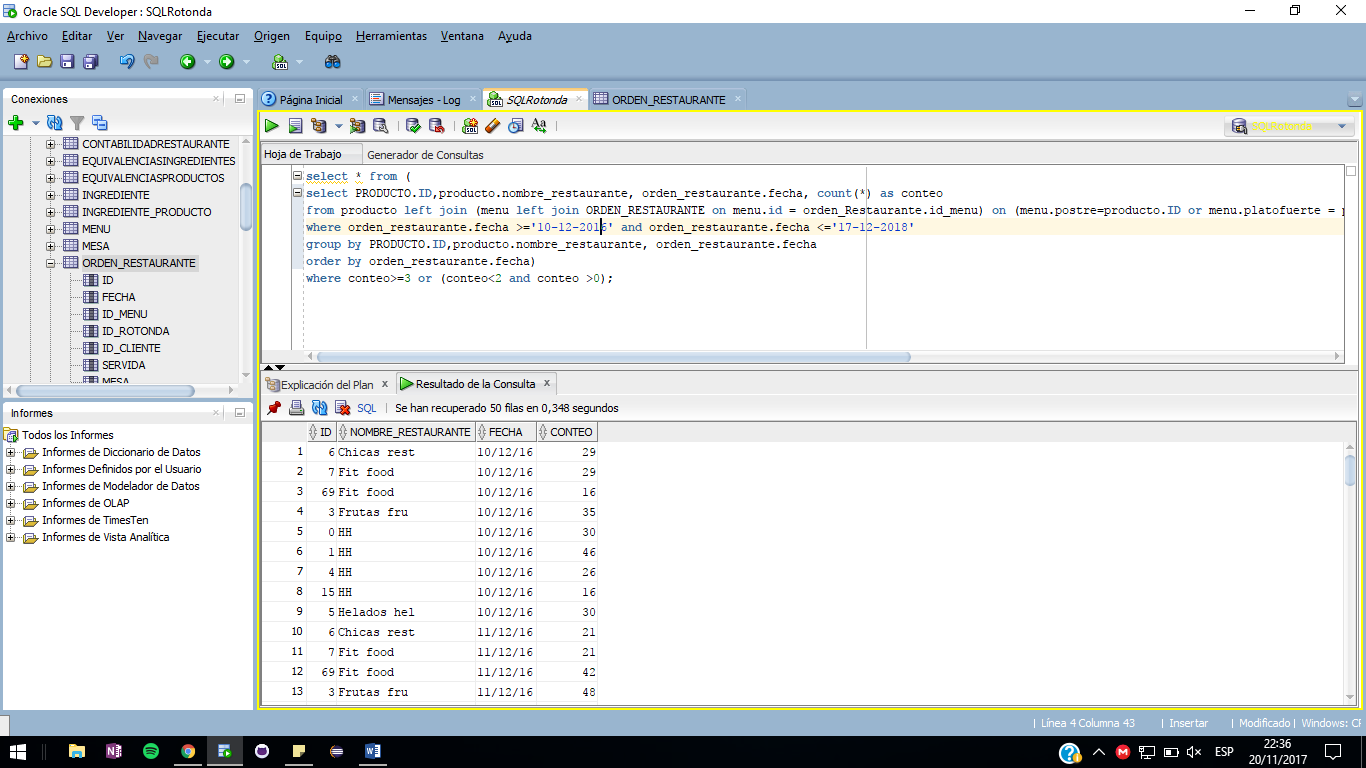
Imagen x: plan de trabajo para el RFC11.

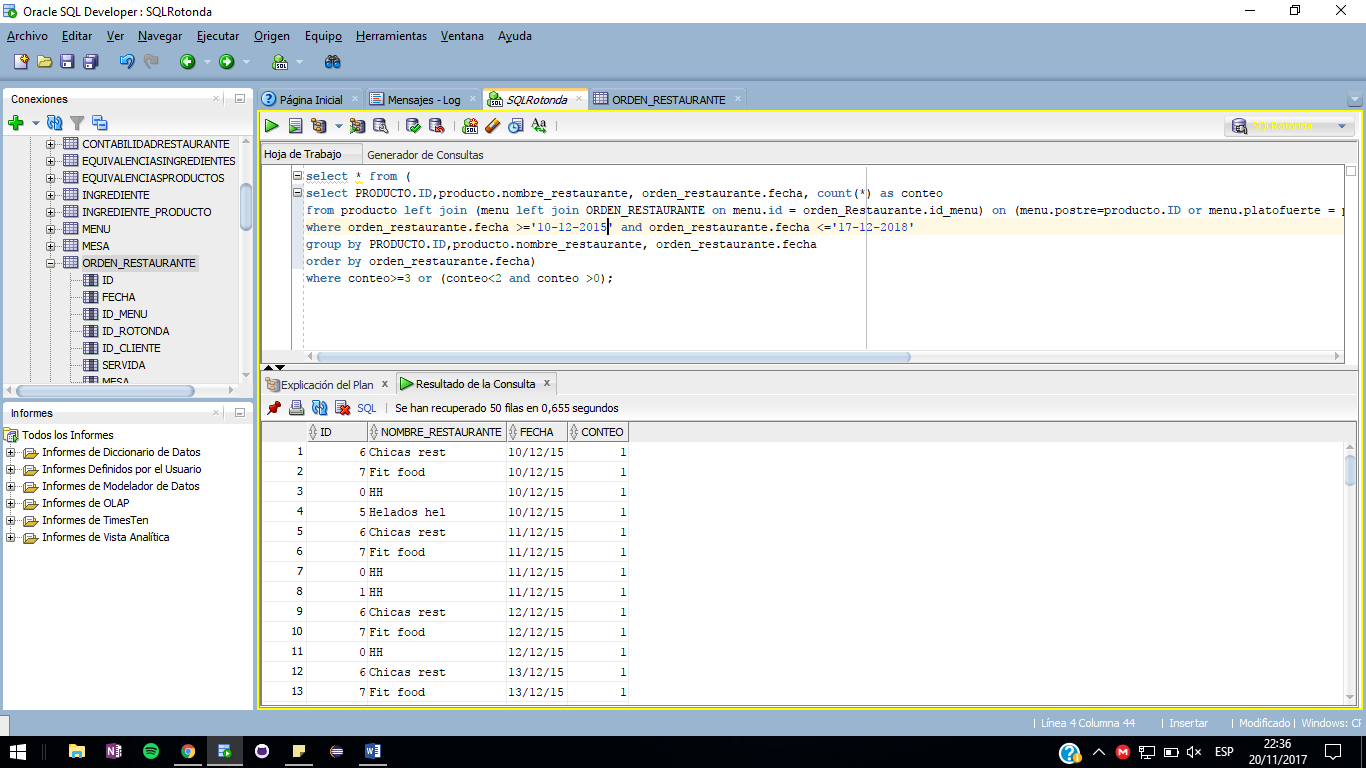
Utiliza los índices creados para él.

Tiempos RFC11.







  
Imágenes x-x: tiempos de ejecución del RFC11 para diferentes rangos de fechas. En todos, el tiempo está dentro de 0,8 segundos.

RFC12

Sentencia SQL:

SELECT CLIENTE.CEDULA,CLIENTE.NOMBRE,

CAST ((COUNT(ORDEN\_RESTAURANTE.FECHA) /52) AS INTEGER) AS NUMEROORDENES, MAX(PRODUCTO.PRECIO) AS PRECIOMINIMO

FROM CLIENTE LEFT JOIN

(ORDEN\_RESTAURANTE

LEFT JOIN (MENU

RIGHT JOIN PRODUCTO

ON MENU.PLATOFUERTE = PRODUCTO.ID)

ON ORDEN\_RESTAURANTE.ID\_MENU = MENU.ID)

ON CLIENTE.CEDULA = ORDEN\_RESTAURANTE.ID\_CLIENTE

WHERE ORDEN\_RESTAURANTE.FECHA BETWEEN ¿ AND ¿

GROUP BY CLIENTE.CEDULA, CLIENTE.NOMBRE

ORDER BY NUMEROORDENES DESC;

De acuerdo al rango de fechas dado por parámetro, el tamaño de la respuesta y el tiempo de ejecución de la sentencia puede variar. El valor más alto obtenido es de 0,8 segundos, los valores promedios, rondan los 0,4.

Valores de los parámetros utilizados en el análisis y que constituyen diferenciadores en los planes de ejecución obtenidos.

GET /administradorrotonda/clientesTipo/fechaMin/fechaMax/

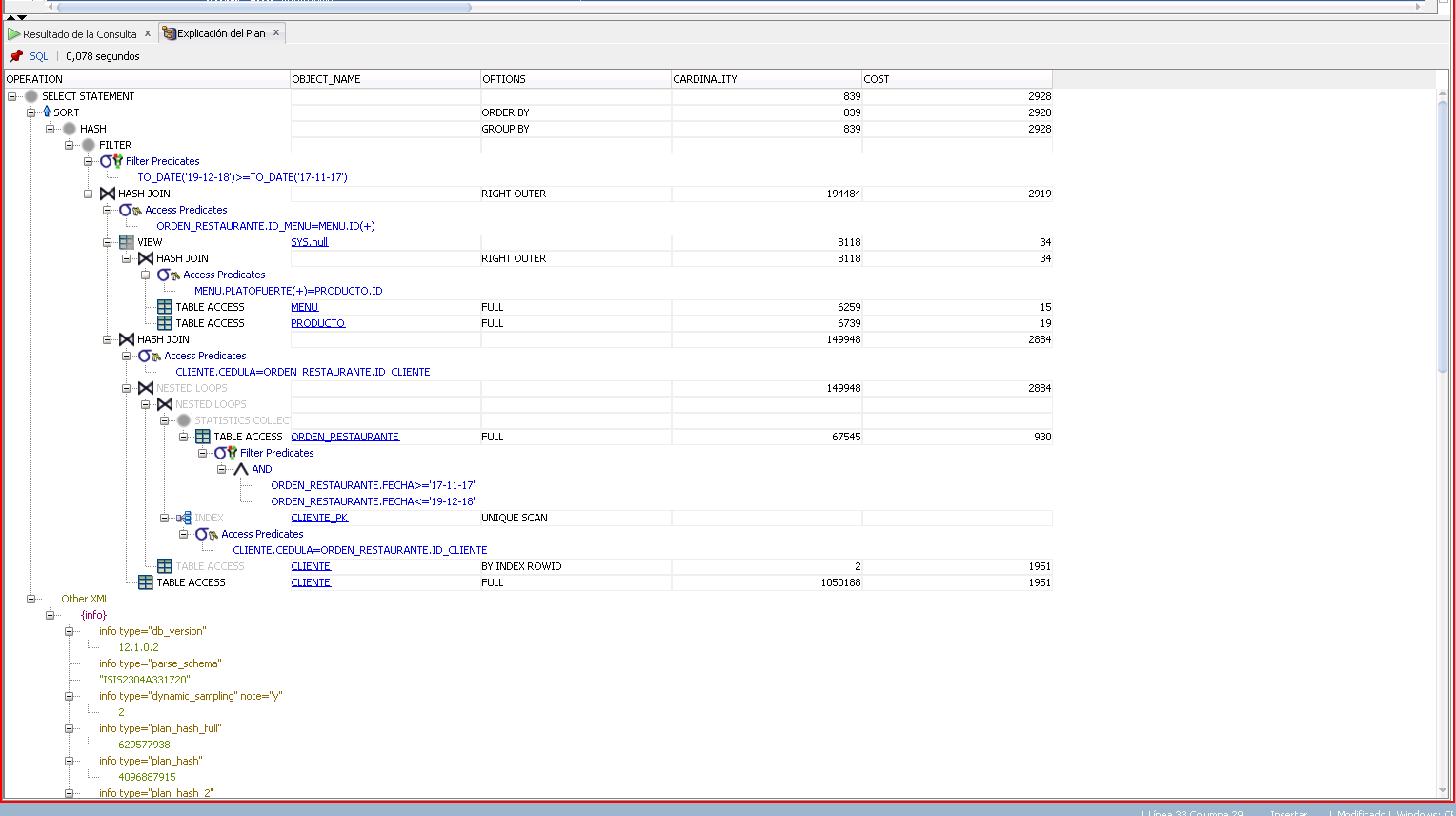
Estos, son la url principal junto y el verbo para pruebas de Postman.

Algunos parámetros validos son:

/9-12-18/29-12-18/ - /9-2-18/29-12-18/- /9-12-10/29-12-18/- /9-12-13/29-12-16/

Donde básicamente, se modifica la fecha de inicio o final del rango de búsqueda.

Planes de ejecución de Oracle.



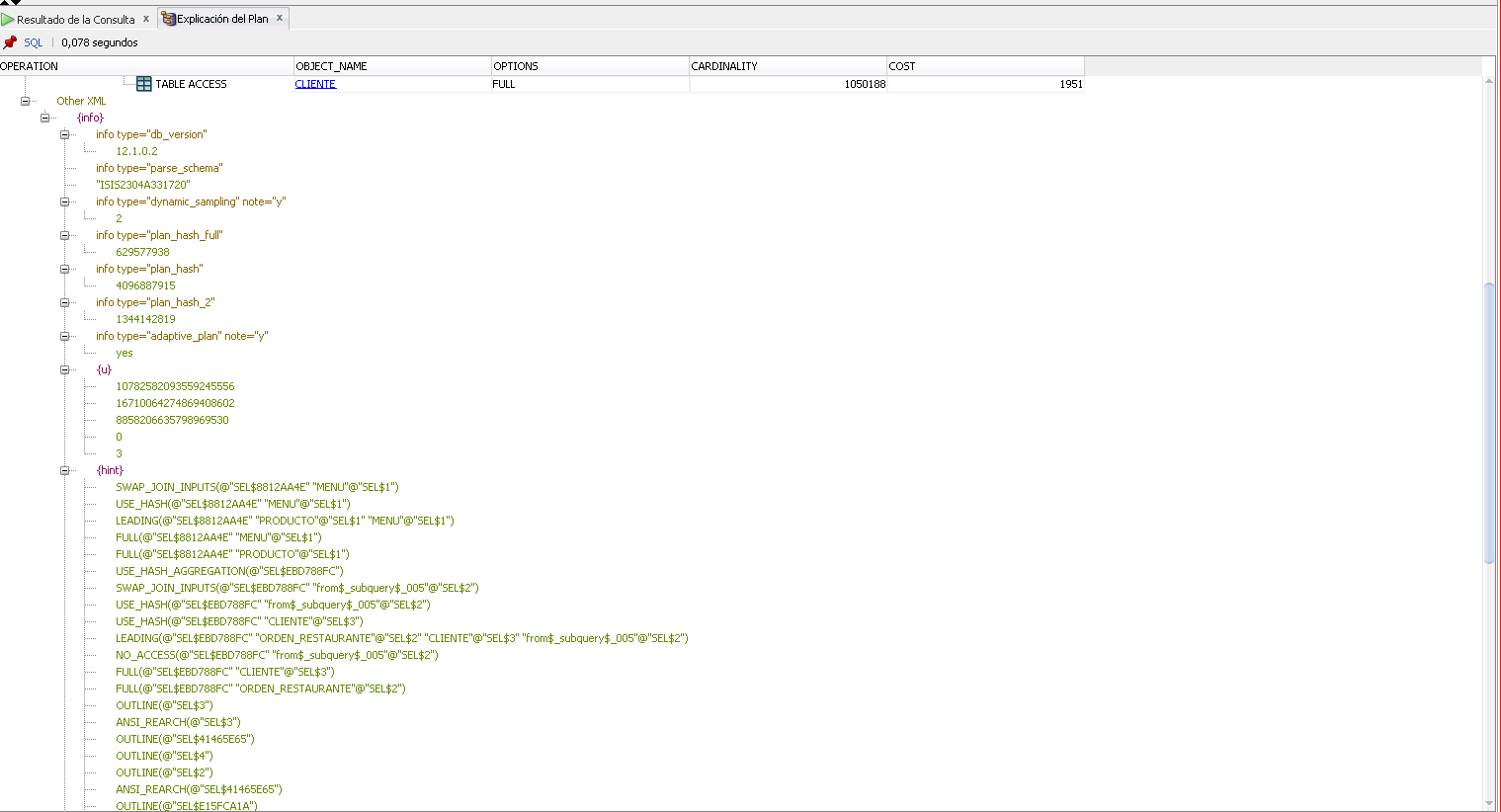
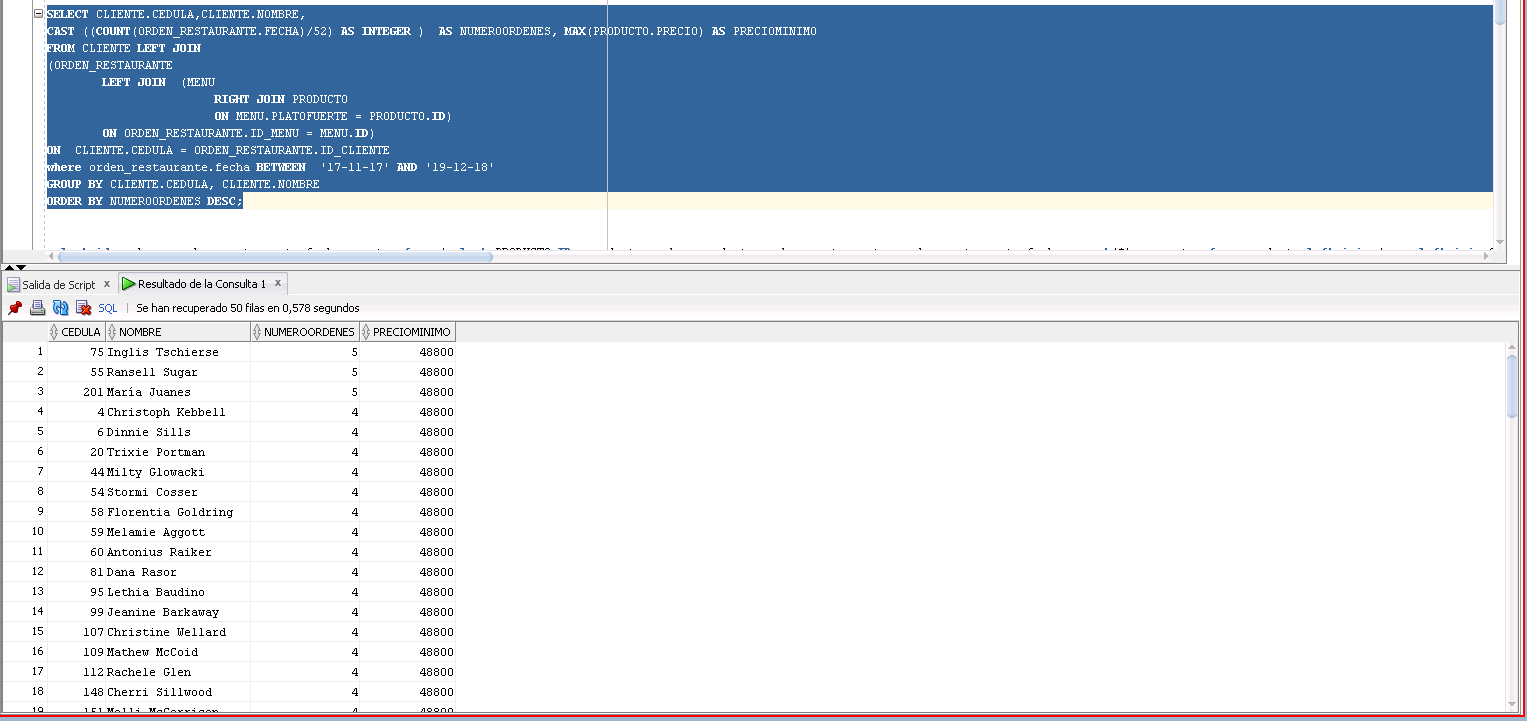
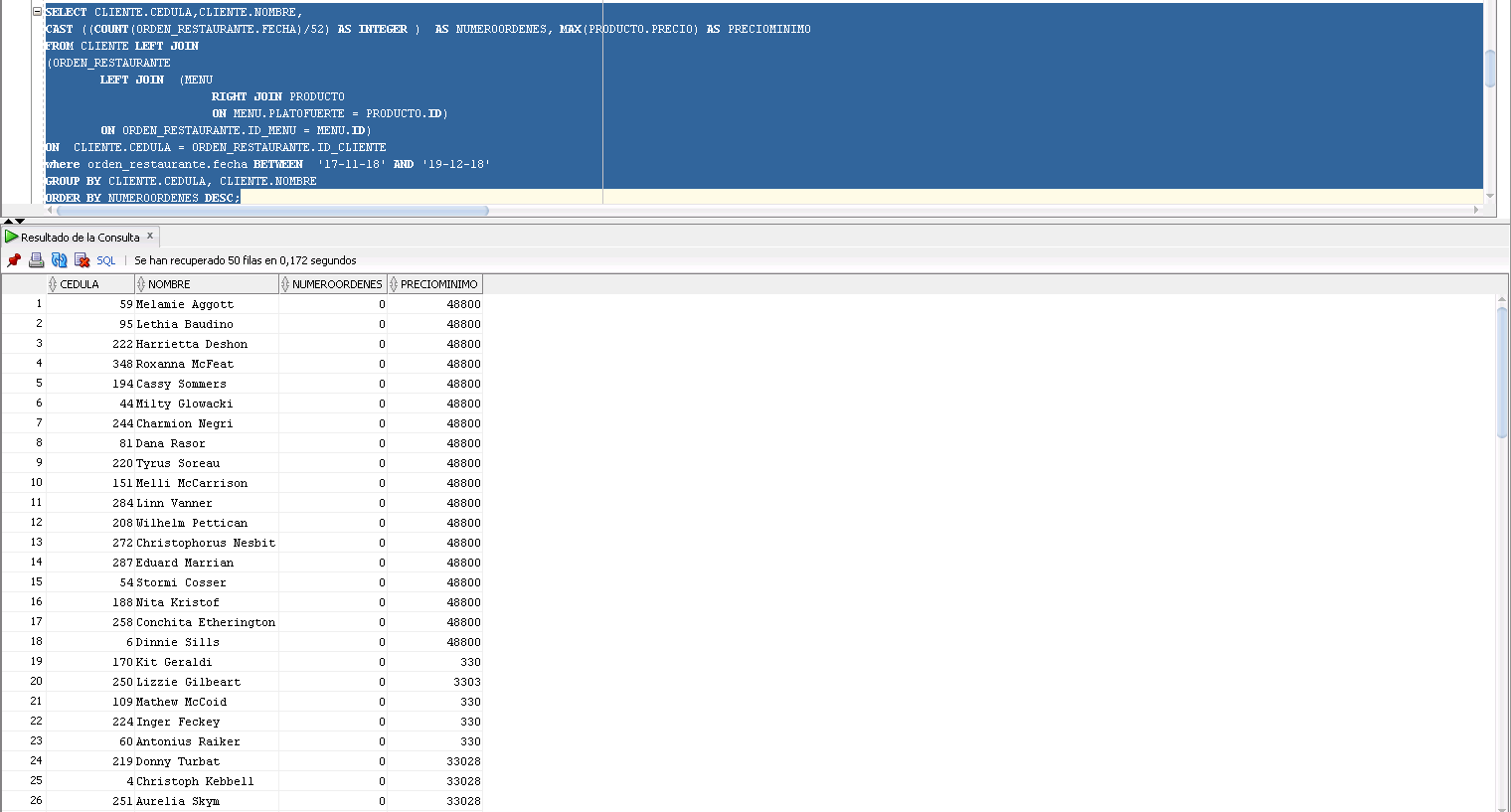
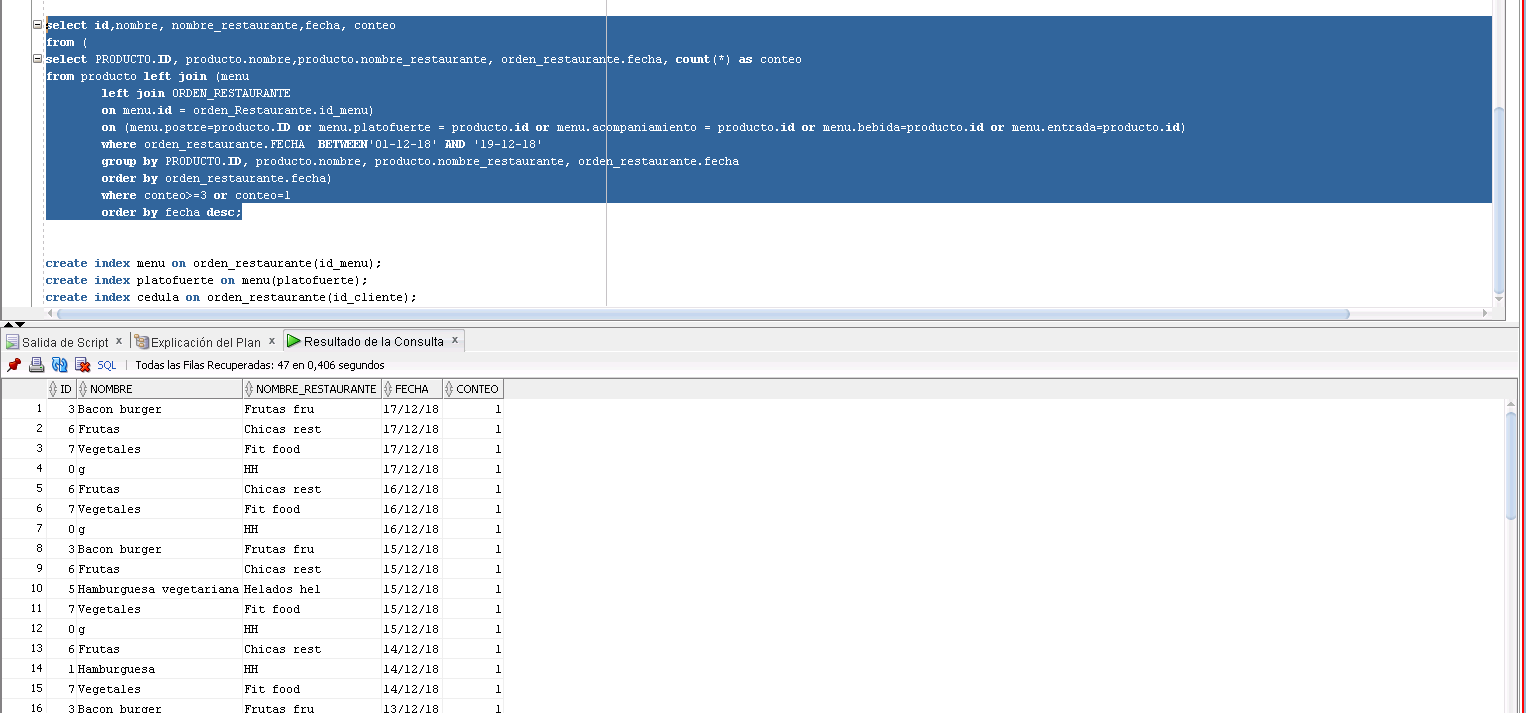


Imagen xx: plan de ejecución para e RFC12. Se refleja el uso de los índices.

Utiliza los índices creados para él.

Tiempos de ejecución





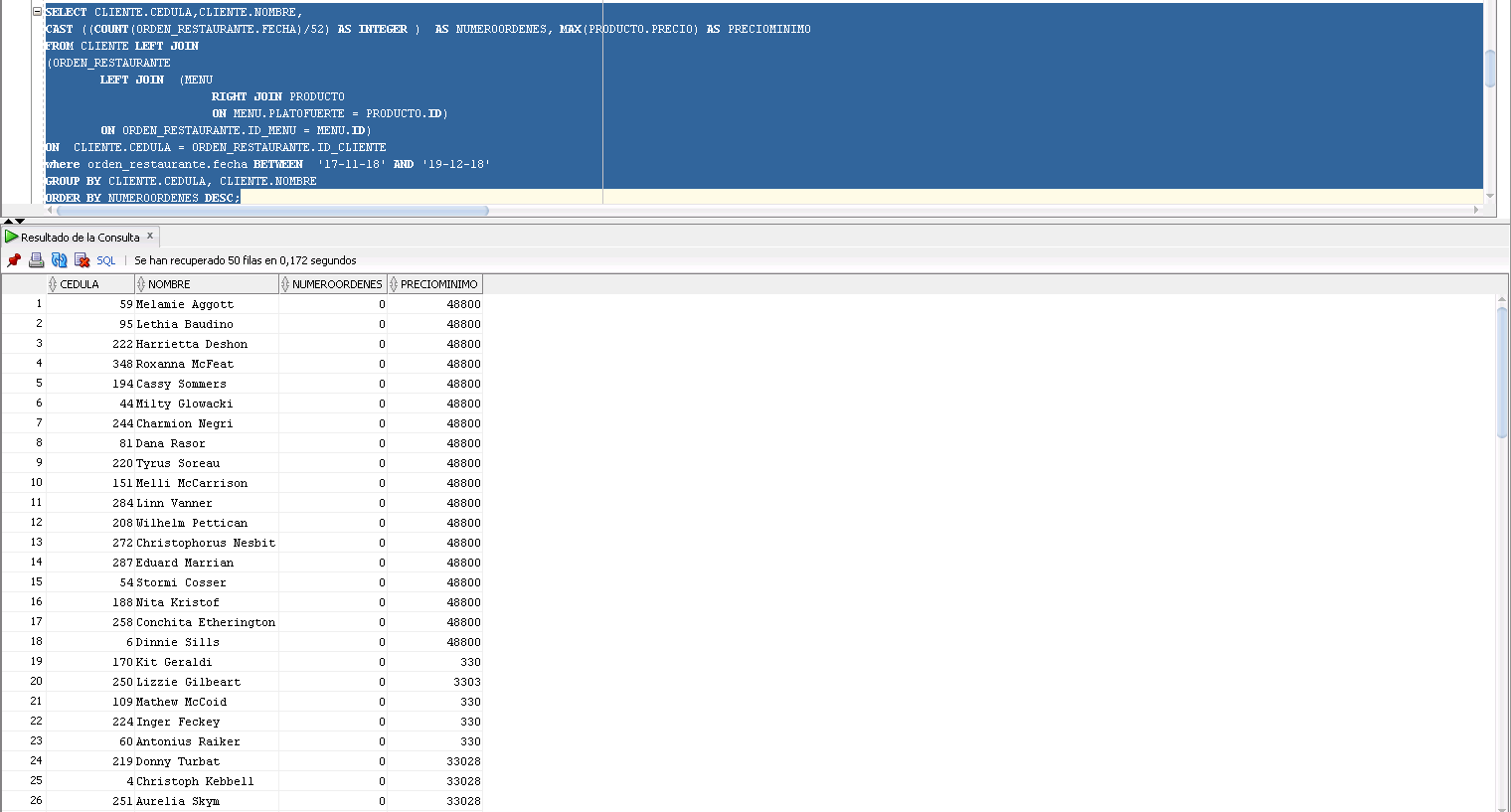


Imagen x-x: tiempos de ejecución para algunos parámetros en el RFC12.

**Análisis de eficiencia**

* Establezca escenarios de datos que le permitan validar diferentes selectividades.
* Para cada requerimiento funcional, seleccione un escenario de análisis y diseñe el plan de ejecución de consulta propuesto por el grupo, de acuerdo con su conocimiento del modelo y de la aplicación.
* Compare y analice el plan de ejecución propuesto por usted y el obtenido en Oracle.

A continuación, se comparan planes presentados por el desarrollador de la plataforma y planes de Oracle para cada requerimiento. Finalmente, para cada funcionalidad, se compararan estos planes y se argumentarán los cambios entre ambos planes y por qué el desarrollador pensaba que era así.

RFC9

RFC10

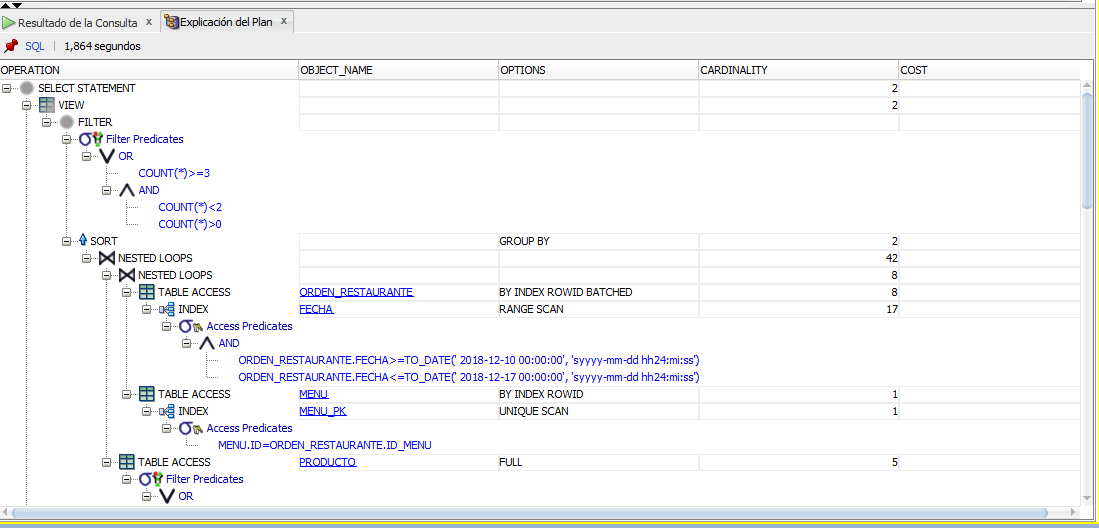
RFC11

Planes de consulta desarrollados por el desarrollador.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OPERACIÓN | | | | OBJETO | OPCIONES |
| SELECT |  |  |  |  |  |
|  | VIEW |  |  |  |  |
|  |  | FILTER |  |  |  |
|  |  |  | conteo>4 |  |  |
|  |  |  | conteo=1 |  |  |
|  |  | NESTED LOOPS |  |  | GROUP |
|  |  |  | TABLE ACCES | ORDEN\_RESTAURANTE |  |
|  |  |  | INDEX | FECHA | INDEX ROW ID |
|  |  |  | ACCESOS | FECHA RANGO | RANGE |
|  |  |  | TABLE ACCES | MENU | ROWID |
|  |  |  | INDEX | MENU\_PK | UNIQUE |
|  |  |  | ACCESOS | Menu = orden.menu |  |
|  |  |  | TABLE ACCES | PRODUCTO | FULL SCAN |
|  |  |  | INDEX | Menu.platofuerte | INDEX BY ROW ID |

Tabla x: descripción del plan para el RFC11. Solo se tienen en cuenta los índices y los accesos a las tablas. Las condiciones no son tomadas muy en cuenta para estos requerimientos.

Planes de consulta obtenidos en Oracle para la ejecución del requerimiento. Para ello, documente con una foto de pantalla los planes de consulta obtenidos en SQLDeveloper.



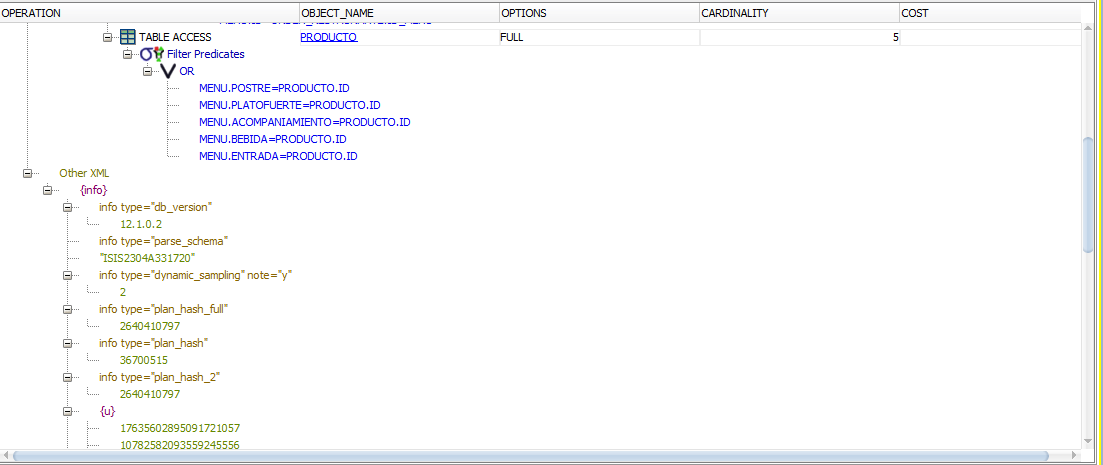


Imagen x: plan de trabajo para el RFC11.

Utiliza los índices creados para él.

Comparación planes.

En el plan desarrollado por el creador del requerimiento, se tenía pensado que a la hora de realizar el join entre los productos y menús, se utilizaría el índice correspondiente a plato fuerte para la tabla de menús. Esta, es la principal diferencia entre los planes de consulta y es debida a la condición del join. Una condición de join que involucra a variar columnas y solo una de ellas posee un índice será tomada como un recorrido total sin usar el índice existente.

Los otros accesos a las tablas, concuerdan entre los plantes, tal como el orden de los accesos a las mismas.

Selectividades.

Orden fecha: la selectividad de las fechas para las ordenes de los restaurantes es alta por la distribución de los datos.

Orden menú: los menús en las ordenes, están distribuidos de manera que la selectividad para este caso de estudio, en este rango de fechas, sea alta, por ende, el índice es utilizado por Oracle

Productos condición join: Si bien existe un join para el plato fuerte de cada menú, este no es utilizado debido a la distribución de los datos en las ordenes, además, el join es realizado sobre varias condiciones en el producto, se piden diferentes opciones para el join, esto por el V involucrado en él

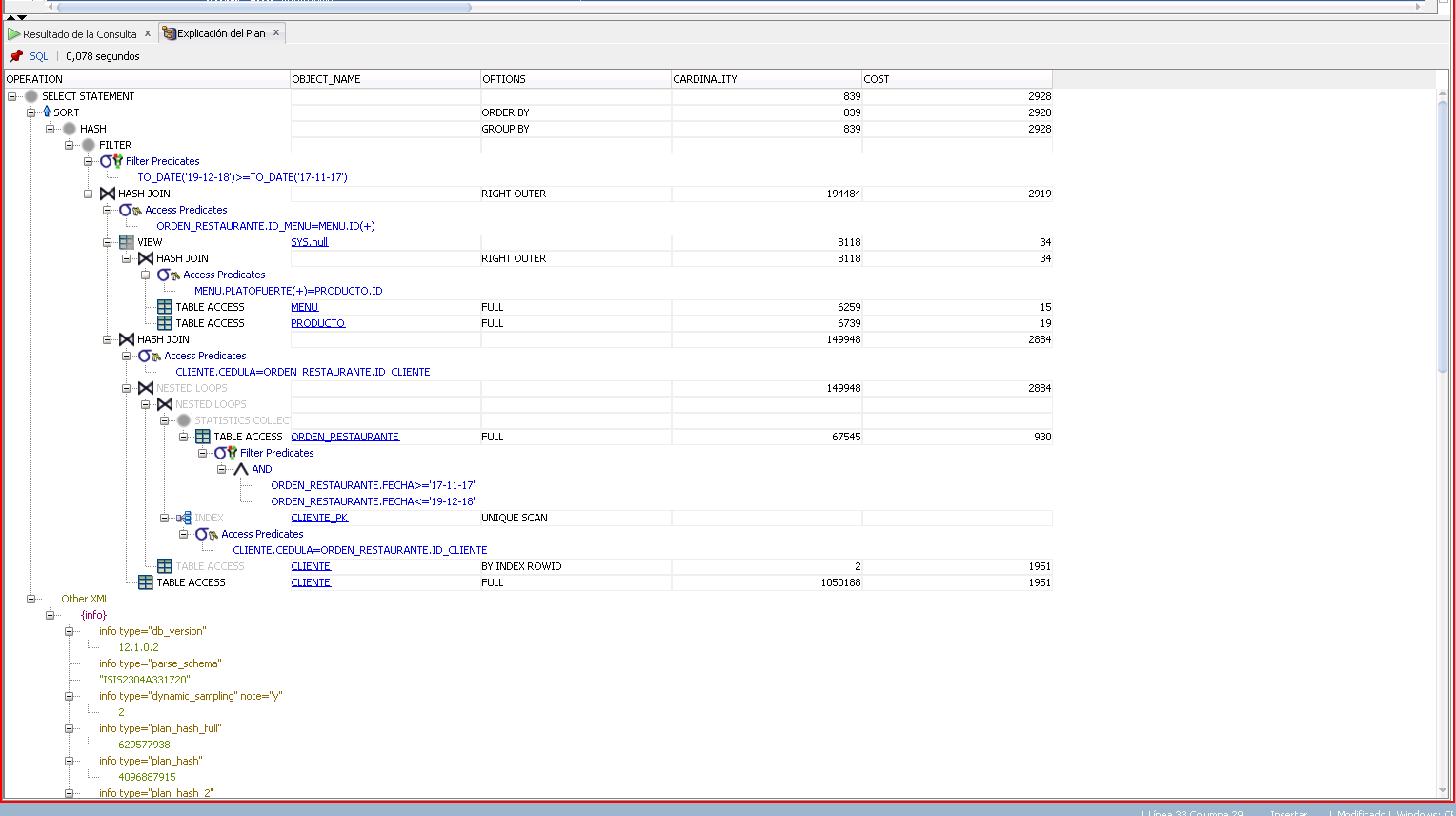
RFC12

Planes de consulta desarrollados por el desarrollador.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| OPERACIÓN | | | | OBJETO | OPCIONES | COMENTARIOS |
| SELECT |  |  |  |  |  |  |
|  | HASH |  |  |  |  |  |
|  |  | FILTER |  |  |  |  |
|  |  |  | DATE 1 |  |  |  |
|  |  |  | DATE 2 |  |  |  |
|  |  | HASH JOIN |  |  | GROUP |  |
|  |  |  | HASHJOIN |  |  | Ciclos anidados |
|  |  |  | TABLE ACCESS | MENU | FULL | indes menu.platofuerte |
|  |  |  | TABLE ACCESS | PRODUCTO | FULL |  |
|  |  |  | HASHJOIN |  |  |  |
|  |  |  | TABLE ACCESS | ORDEN\_RESTAURANTE | UNIQUE SCAN | INDEX orden.cliente |
|  |  |  | TABLE ACCESS | CLIENTE | FULL |  |
|  |  |  |  |  |  |  |

Tabla x: descripción del plan para el RFC12. Solo se tienen en cuenta los índices y los accesos a las tablas. Las condiciones no son tomadas muy en cuenta para estos requerimientos.

Planes de ejecución de Oracle.



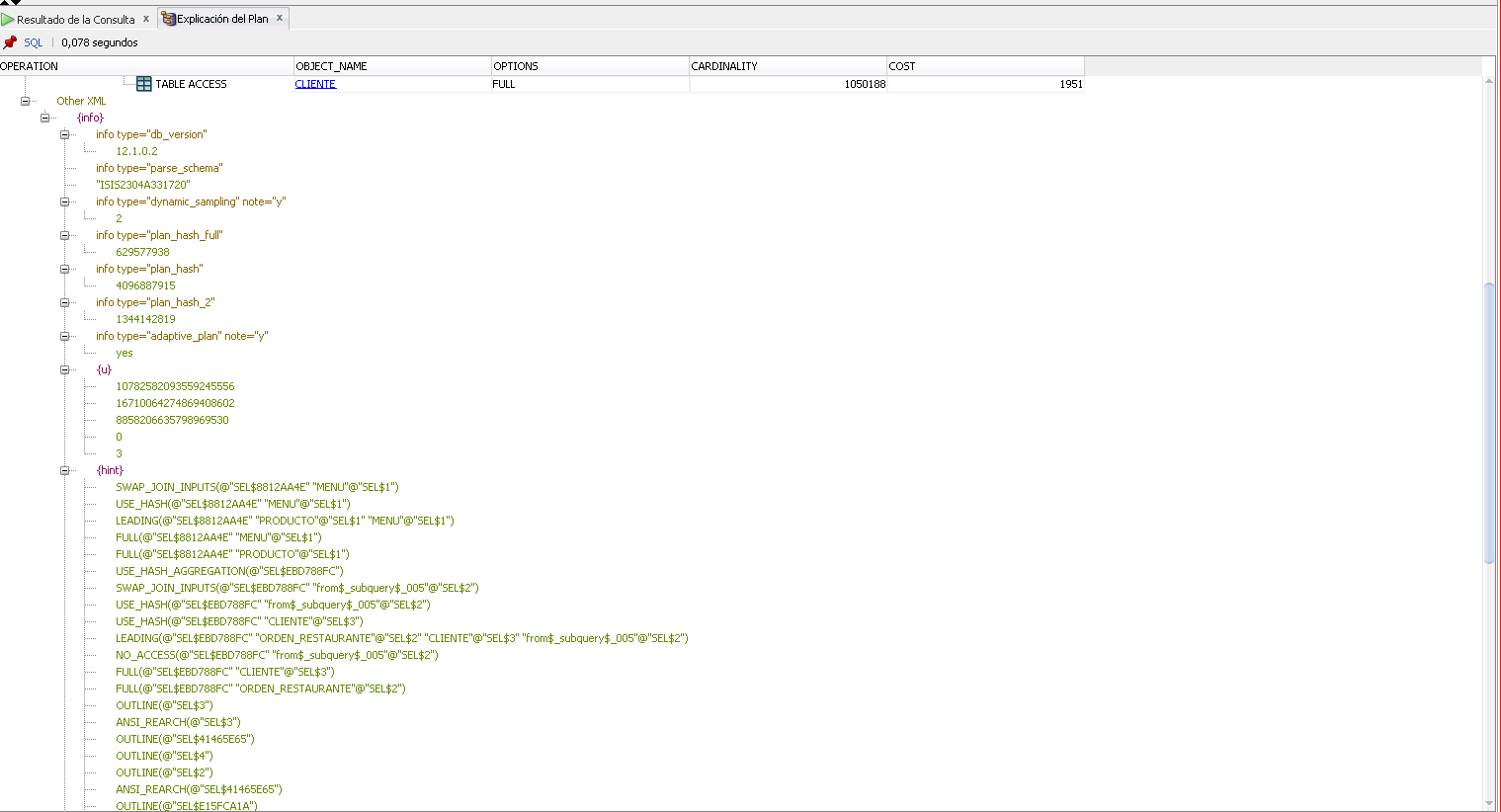


Imagen xx: plan de ejecución para e RFC12. Se refleja el uso de los índices.

Comparación planes de ejecución: Los dos planes de ejecución, estiman de la misma manera el orden para entrar a las tablas y el tipo de join a utilizar, un hash join. Esto, porque el tamaño de las tablas y el intervalo de fechas de búsqueda es grande para el volumen de datos existentes.

Los joins entre el plan del desarrollador y el plan de Oracle no coinciden esto puede ser debido a cuestiones de selectividad en el plato fuerte, la cantidad de ordenes de ese plato para el rango de fechas puede ser muy alta y la selectividad tenderá a disminuir. De esta manera, no es tomado en cuenta ese índice para este caso de comparación entre planes.

Selectividades.

Fechas: la selectividad de las fechas para las ordenes de los restaurantes es alta por la distribución de los datos.

Menu.plato fuerte: la selectividad es baja por la distribución de los datos y el intervalo de fechas.

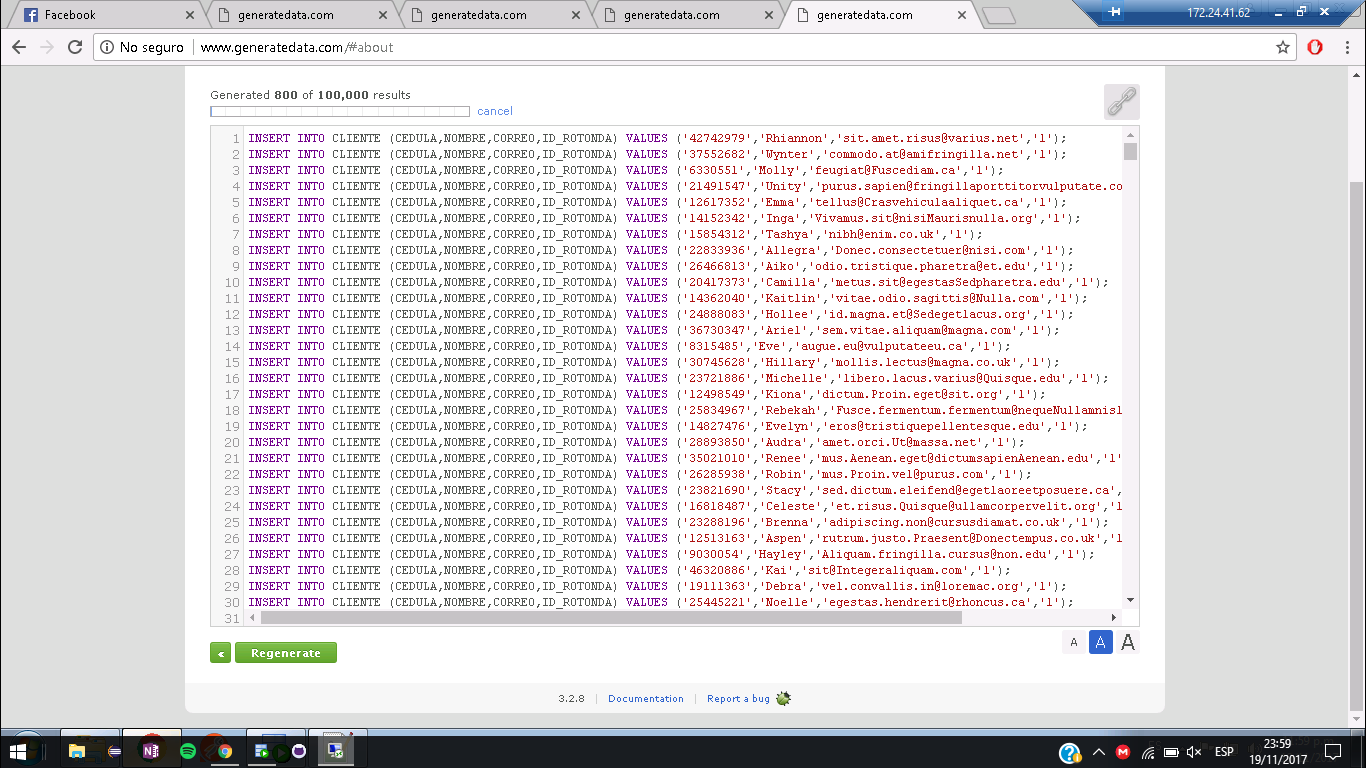
Puede variar dependiendo las fechas.

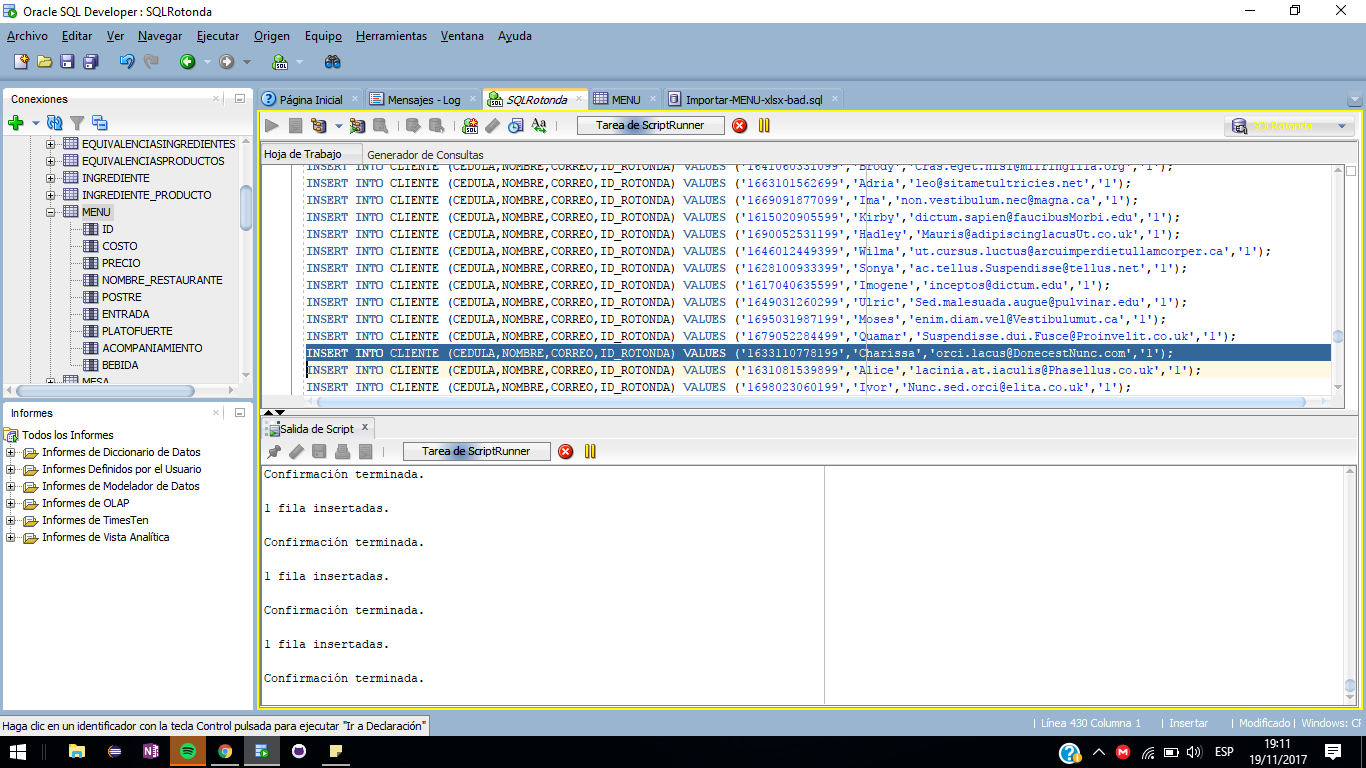
Orden restaurante cliente: Presenta selectividad alta, Oracle utiliza el índice para realizar la sentencia.

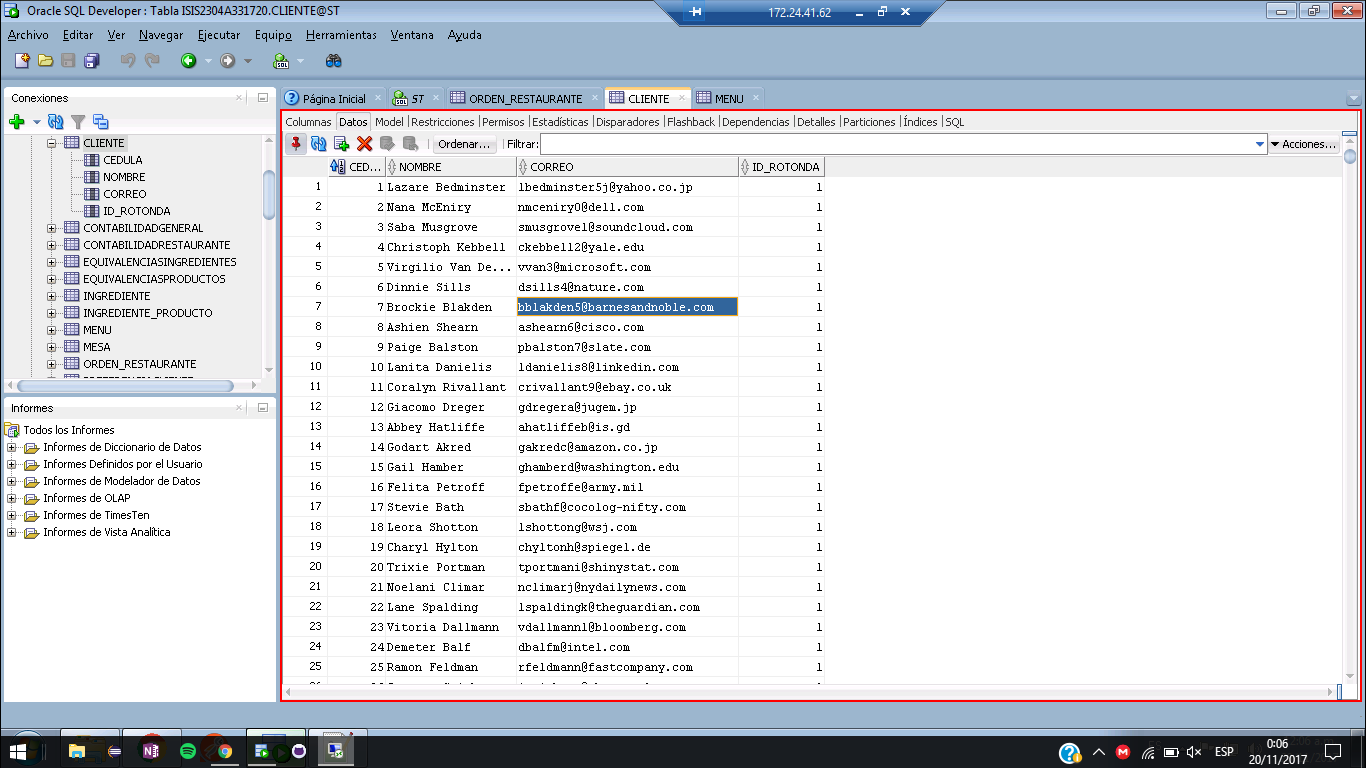
3 (35%) Construcción de la aplicación y análisis de resultados

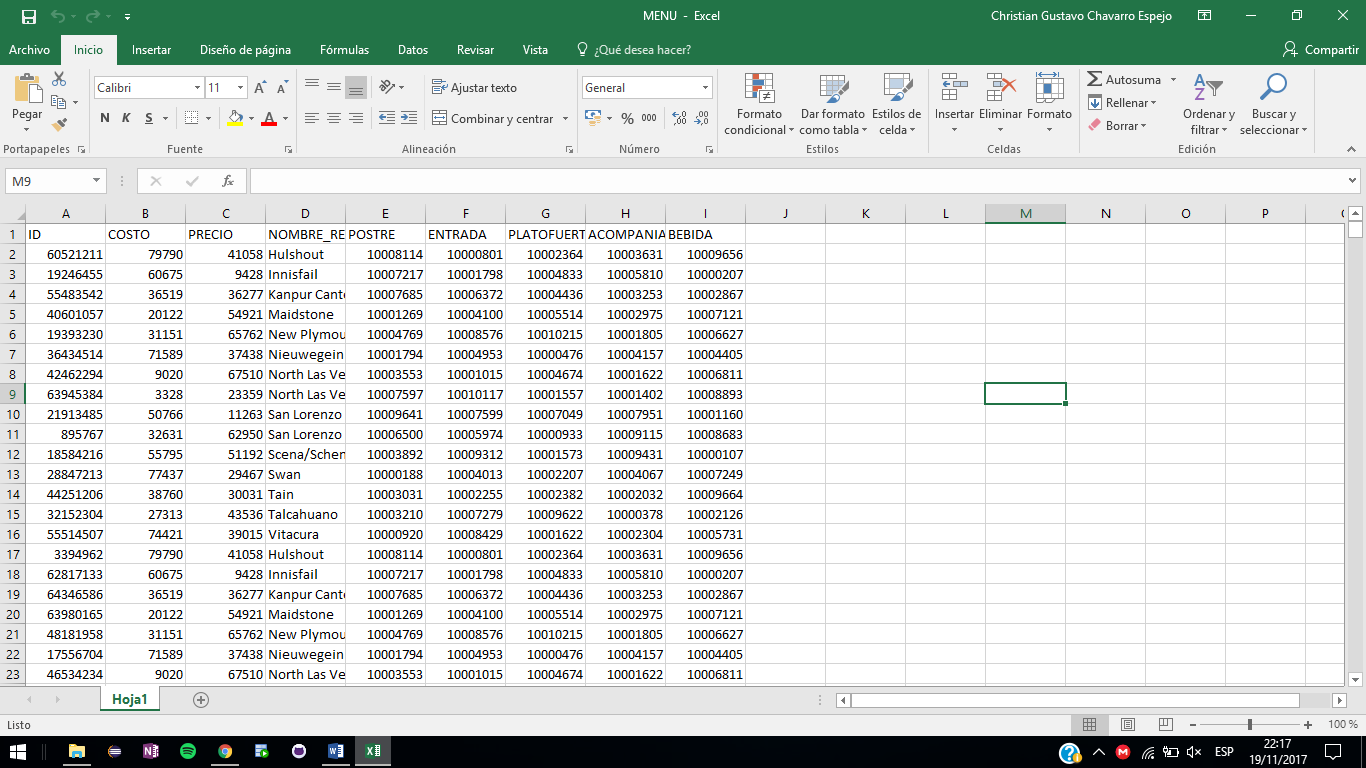
* Ajuste las tablas creadas en Oracle de acuerdo a las decisiones del punto anterior
* Diseño del escenario de pruebas de eficiencia. Cargue de datos necesarios para hacer el estudio de eficiencia de la aplicación.
* Diseñe los datos que le permitan verificar adecuadamente las reglas de negocio. Note que es importante generar adecuadamente los datos y para esta iteración lo es también el obtener un número muy grande de ellos. Se debe generar un volumen de datos tal que algunas tablas no quepan en la memoria principal de la máquina. El no cumplimiento de este requisito implica la invalidez de este componente de la evaluación
* Puede escribir un programa de generación automática de datos acorde al diseño establecido para los mismos.
* Para la población de las tablas utilice herramientas de carga masiva como SQLLoader o las disponibles en SQLDeveloper. Consulte el tutorial disponible en la wiki del curso sobre SQLLoader
* (5%) Documente claramente el proceso de carga de datos: Cómo fue realizado, cómo logró el volumen de datos solicitado

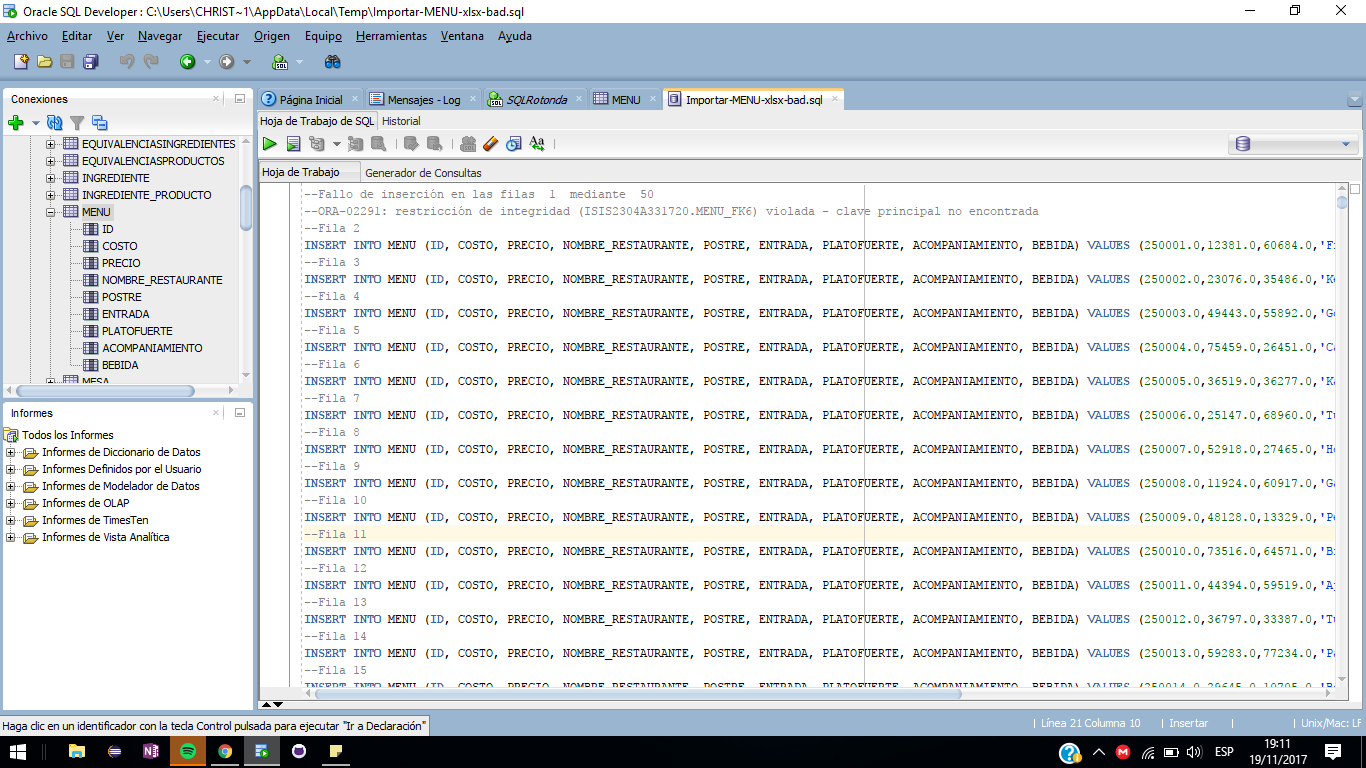
En una etapa de la carga de datos, se realizó mediante archivos de Excel y generadores de datos en línea.

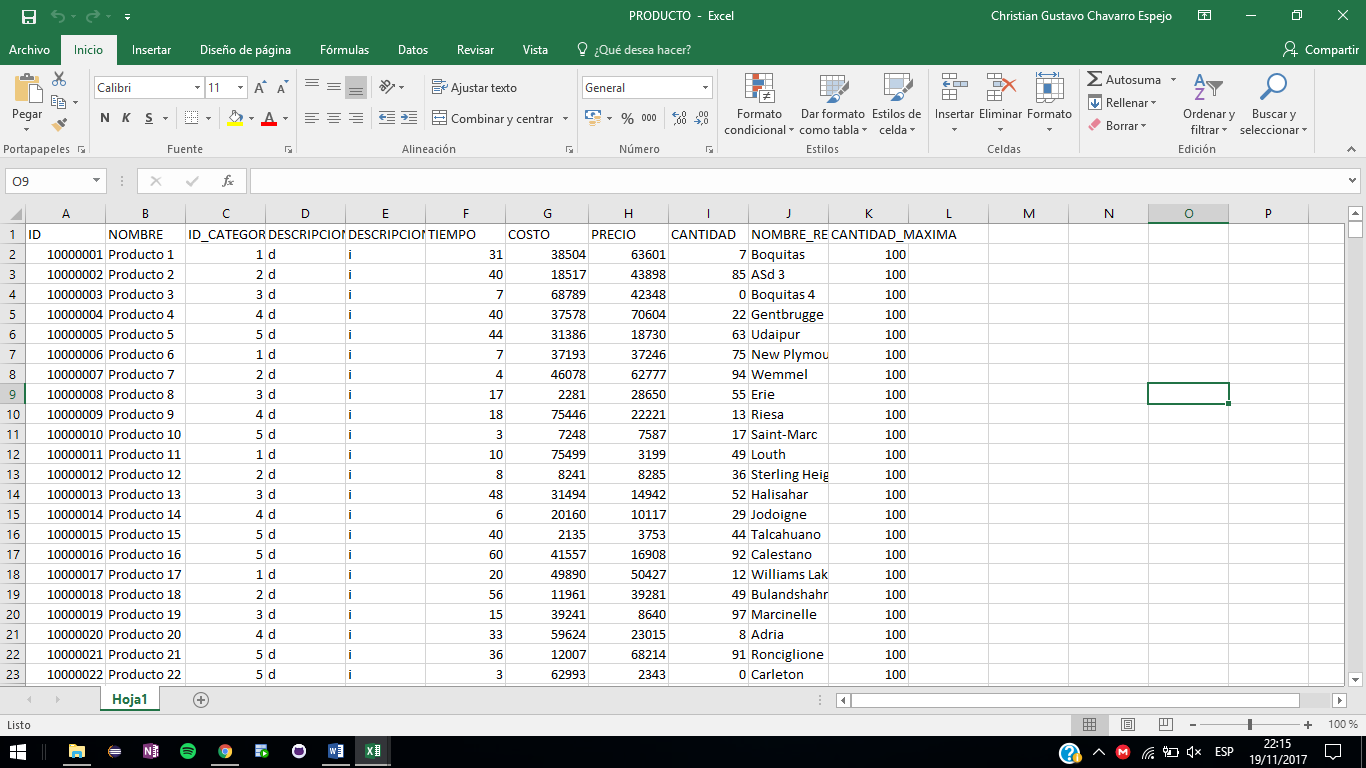
  
Imagen x: Generación de sentencias para la Inserción de clientes mediante sentencias por un generador de datos.

  
Imagen x: Inserción de clientes mediante sentencias ya realizadas en un generador de datos.

  
Imagen x: Clientes una vez insertados.

  
Imagen x: Ejemplos de menús que se insertan mediante Excel.

  
Imagen x: Inserción de los menús mediante el Excel.

  
Imagen x: productos en un archivo de Excel.

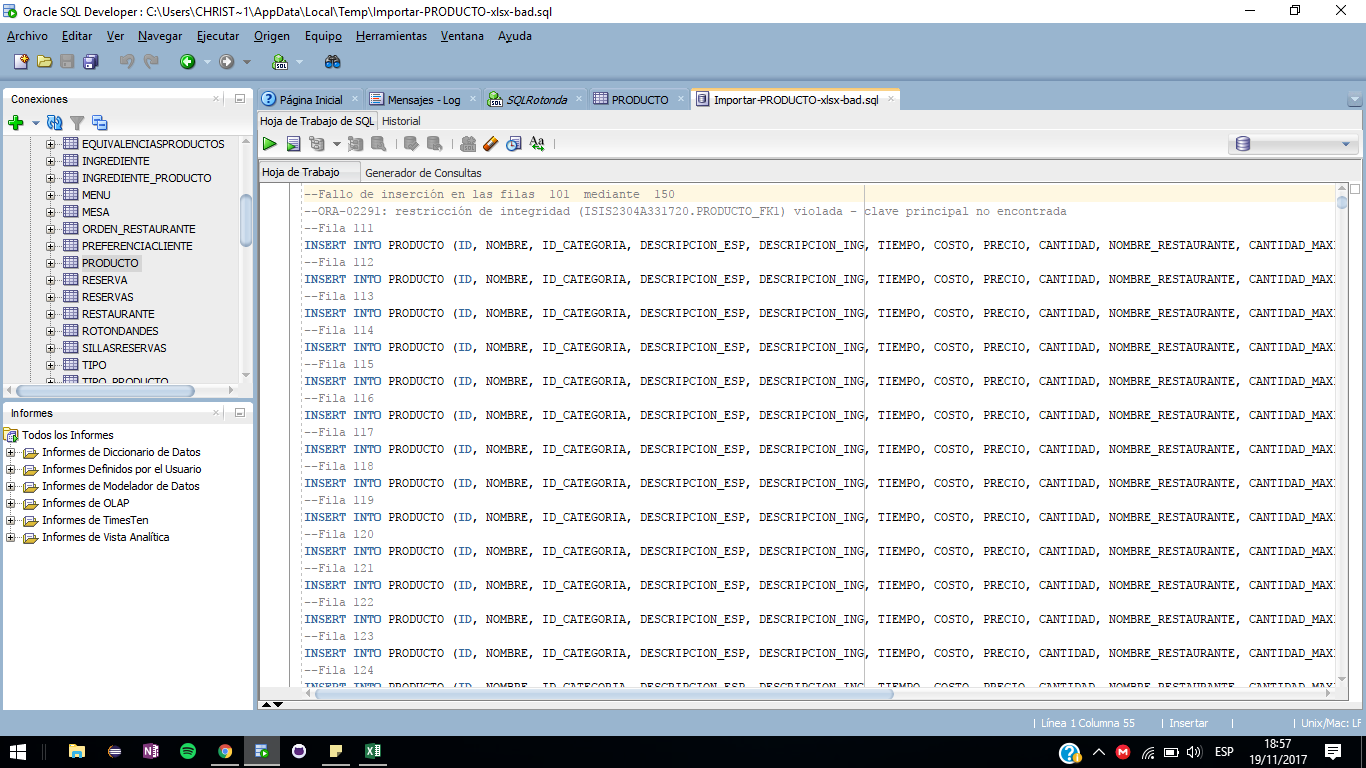


Imagen x: Inserción de productos mediante el archivo de Excel.

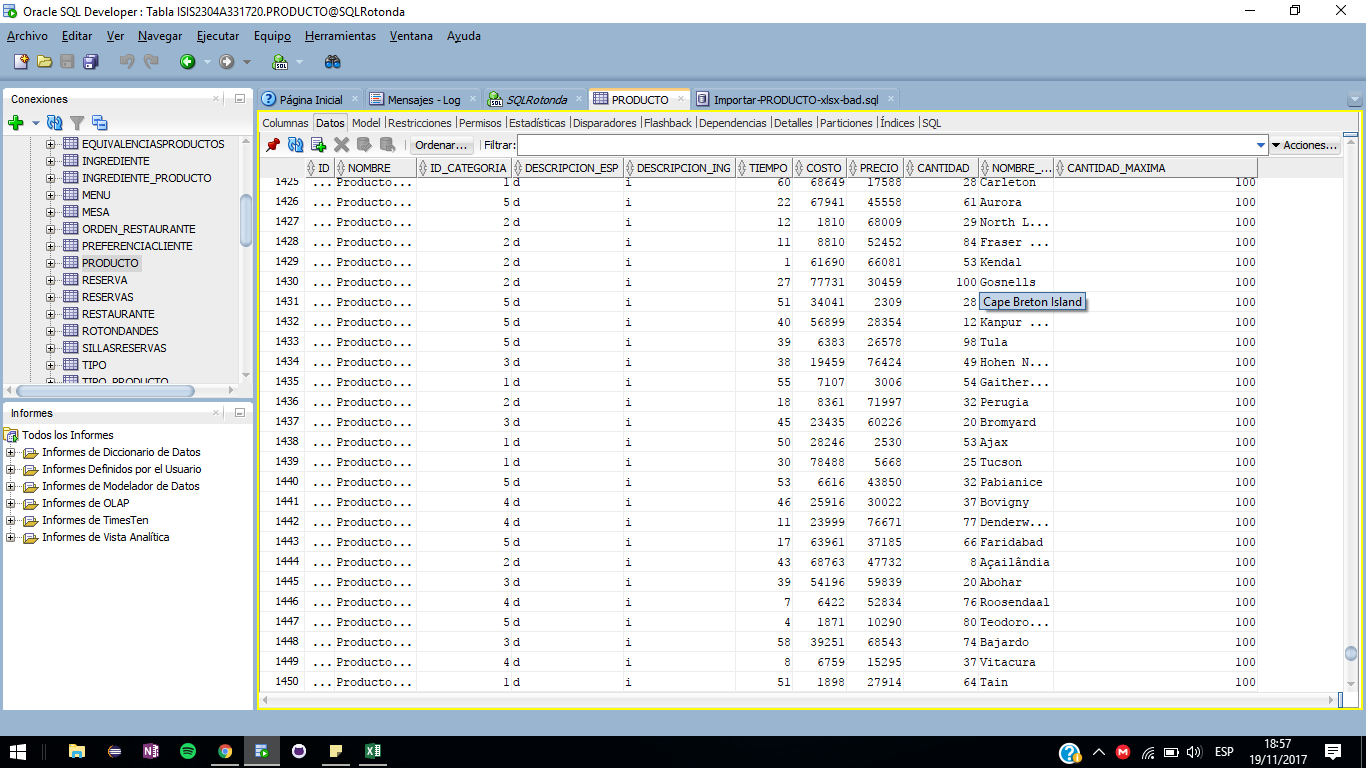
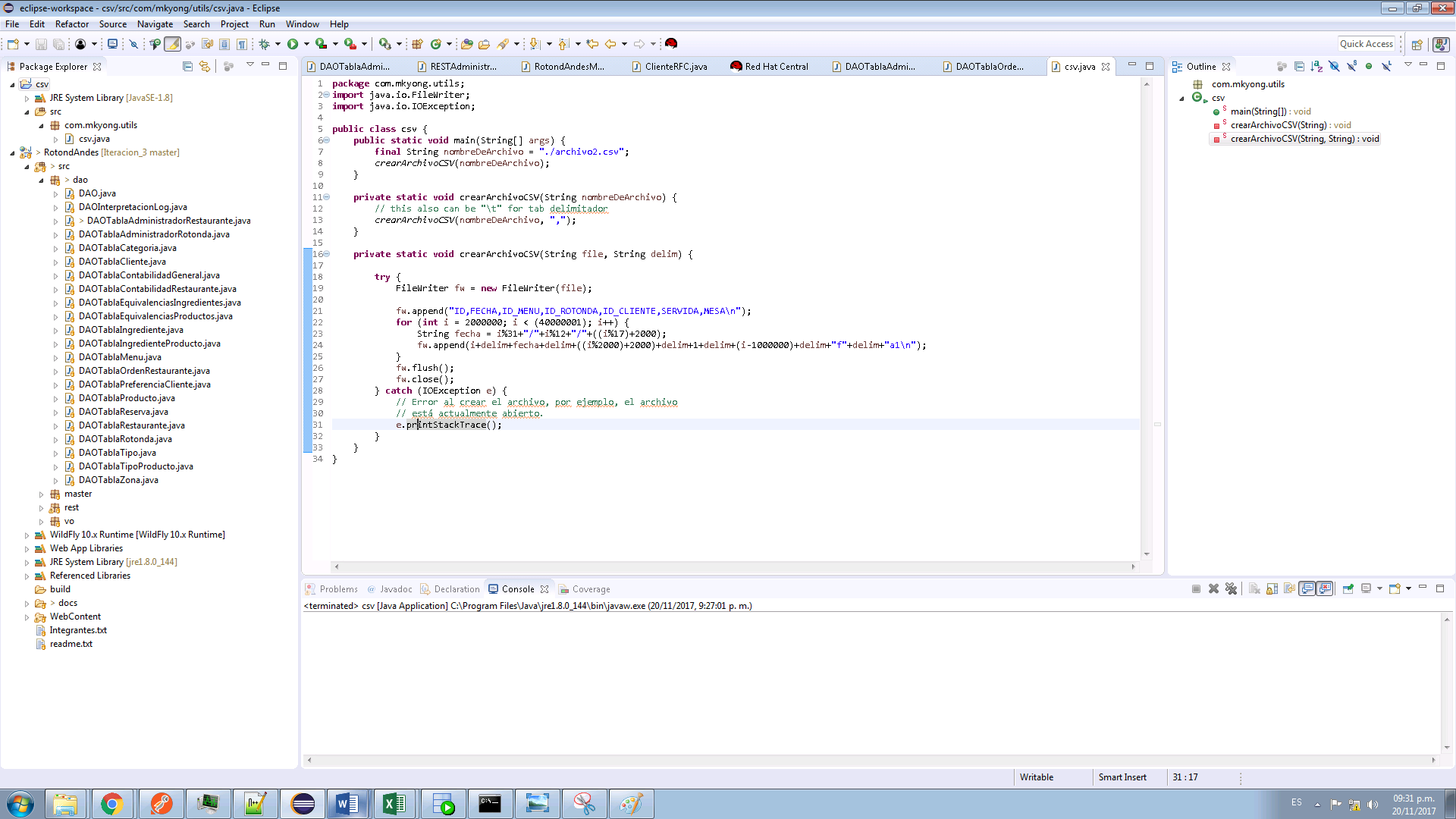
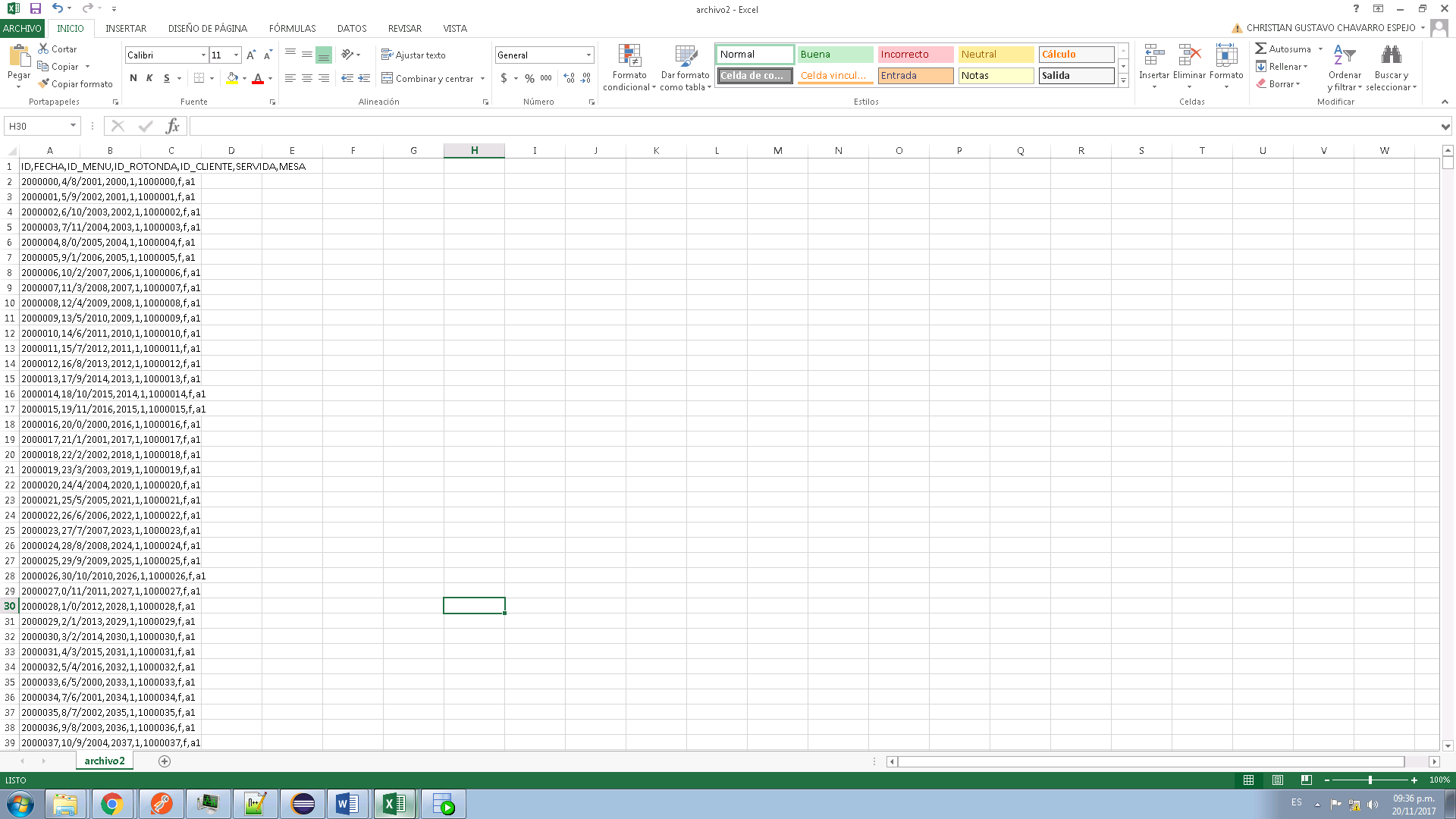
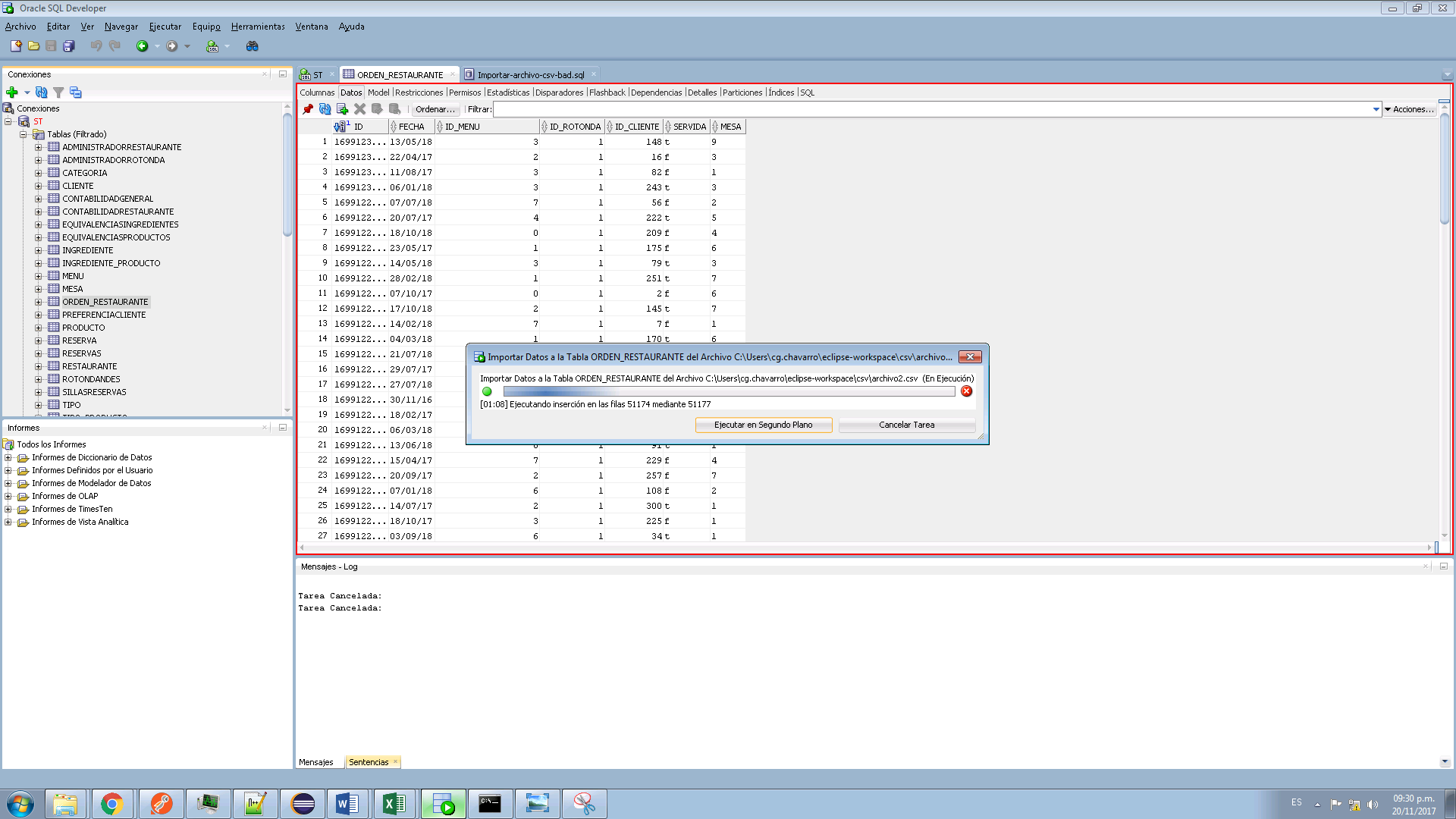


Imagen x: Productos después de ser insertados.

Imagen x: descripción del programa en java para generar los datos.  
Imagen x: respuesta del programa

  
Imagen x: inserción de los datos del csv.

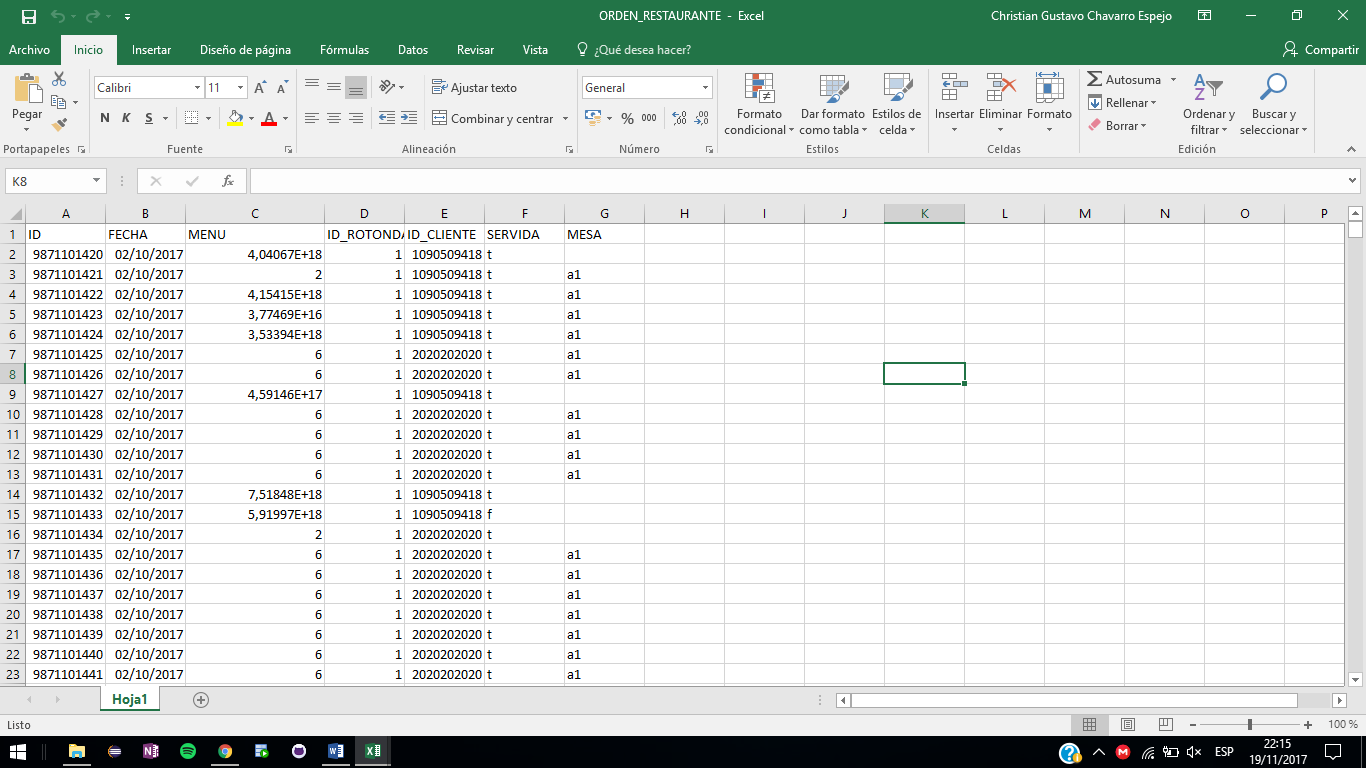
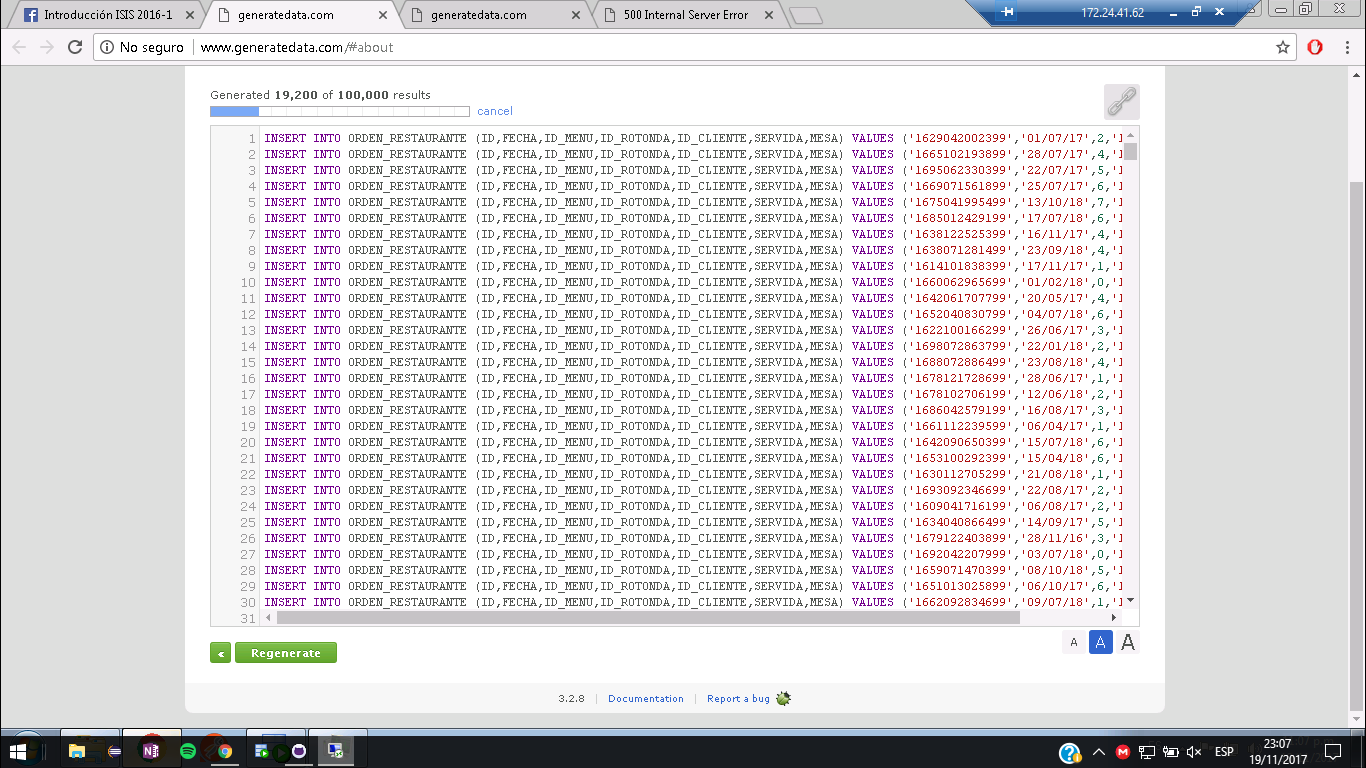


Imagen x: Ordenes a ser agregadas en la base de datos.

Imagen x: Ordenes a ser agregadas en la base de datos mediante sentencias generadas.

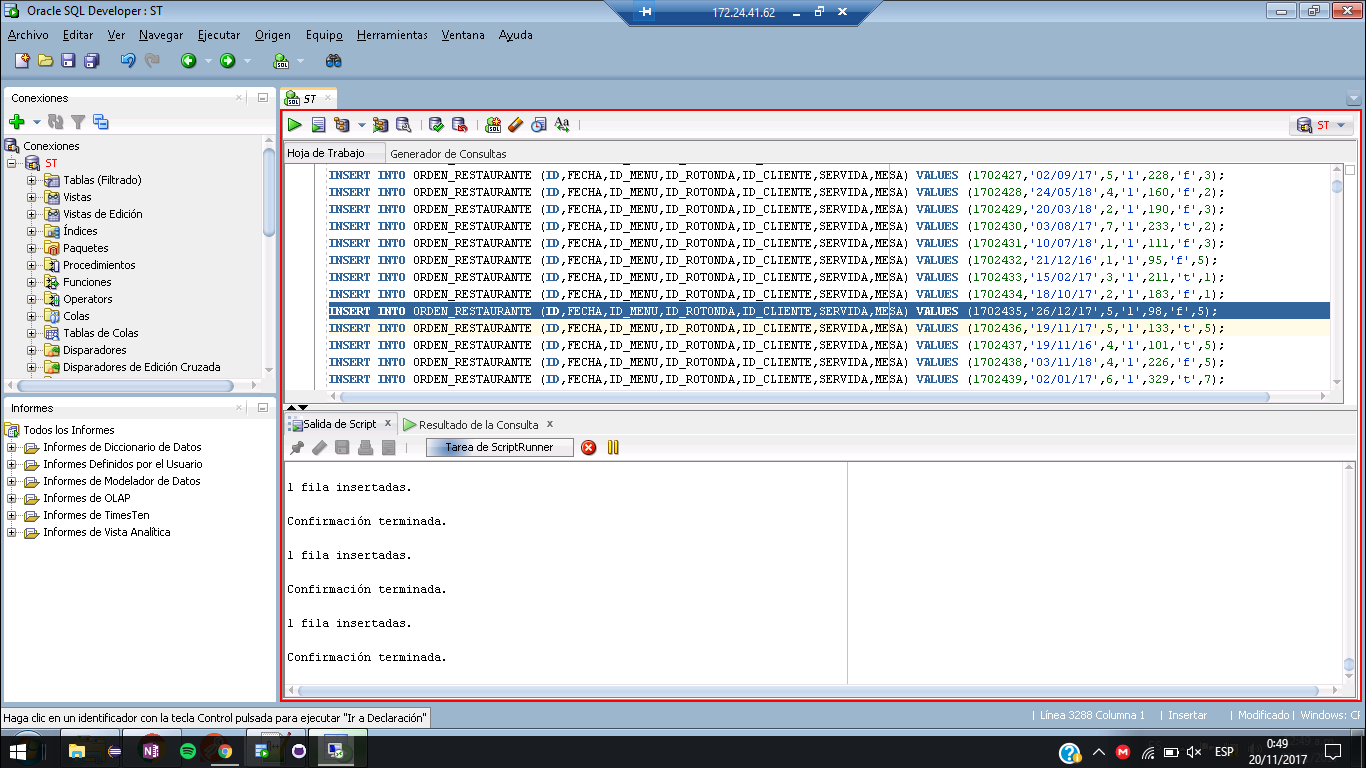


Imagen x: Ordenes siendo agregadas a la DB.

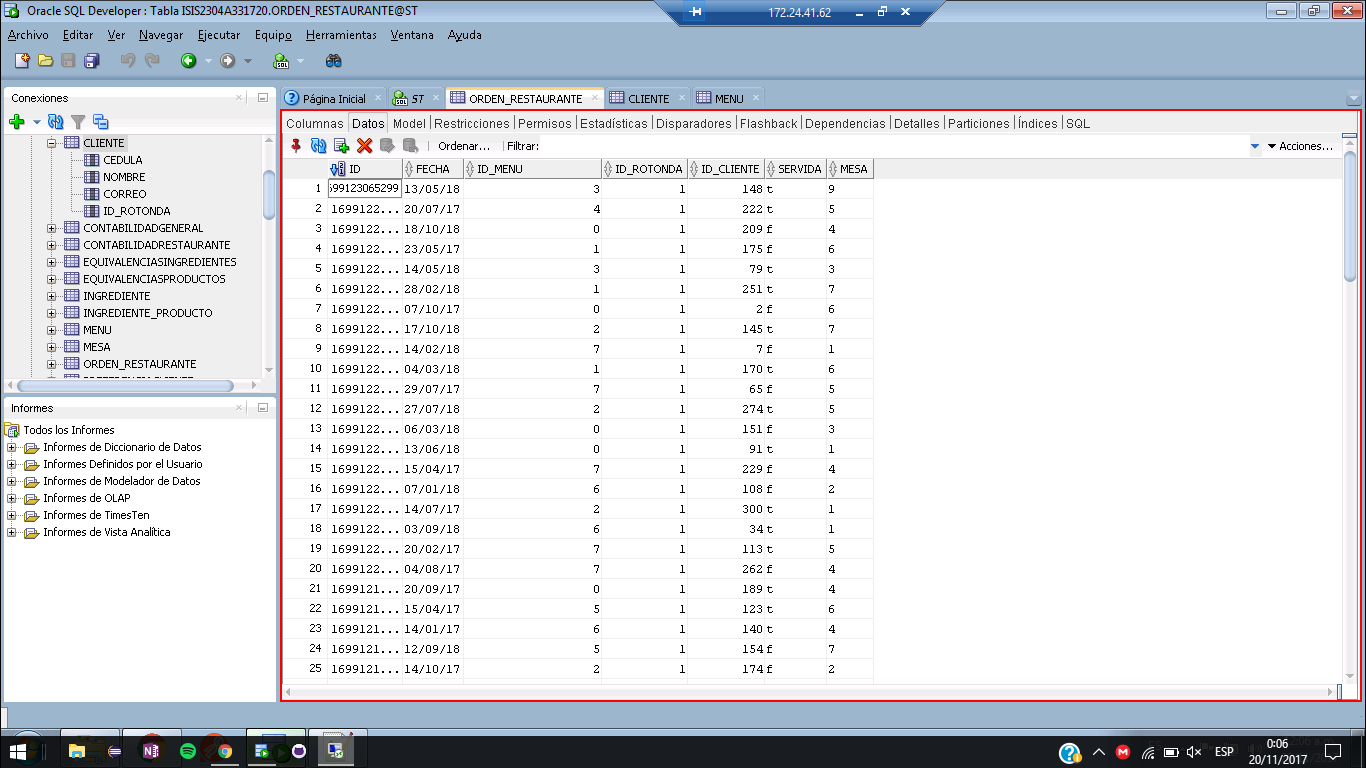


Imagen x: Ordenes una vez agregadas a la DB.

En las imágenes anteriores, se describen procesos de carga masiva de datos y solo corresponden a algunos ejemplos de la carga de estos mismos. En la aplicación, al momento de esta sustentación, el volumen de datos será más alto. Comparados con los pocos datos que se ve, son insertados en estos ejemplos.

También se muestra cómo se agrega y se crea el archivo de CSV para la creación de un millón de órdenes y de clientes, solo es mostrada la versión de la creación de órdenes.

Desarrolle o ajuste las clases involucradas en los nuevos requerimientos, de forma que complete o modifique los requerimientos funcionales y cumpla con las restricciones de negocio. Realice los cambios sobre las clases que corresponden a:

No se realizaron cambios en las tablas.

* Desarrollo y/o ajustes a los servicios REST para cumplir con 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 y 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.
* Documente el análisis realizado, de forma clara y concisa.

En este punto, primero se describirá cómo trabaja cada proceso de manera breve. Tras esto, se comparará la eficiencia y qué conlleva cada uno de estos trabajos, realizar las operaciones mediante la sentencia de SQL o trabajar con los datos en memoria principal.

SQL.

Imagine el proceso de búsqueda y catalogación de los clientes, para él, se involucran como mínimo las tablas de clientes, ordenes, menús y productos. El SMBD, en este caso ORACLE, gestiona los joins entre estas tablas mediante los índices, árboles que tiene creados en su memoria principal, la del servidor. Tras concatenar estas tablas, en la sentencia SQL se piden solo las filas que interesan para el requerimiento, de las cerca de 22 filas resultantes de los joins. Con estas filas, acompañadas de una operación de agregación, se puede retornar la respuesta a la consulta.

La complejidad asociada, variará de acuerdo al tipo de join que utilice la sentencia y si esta tiene algún índice asociado.

JAVA.

Antes de iniciar esta descripción, se presenta un problema de antemano, las tablas no logran ser procesadas en memoria principal, los objetos no caben en ella.

Aún, si todos los objetos entrasen en la memoria del dispositivo sobre el cual se realiza la acción, objetos de tipo: Cliente, Orden, Menú, Producto. Realizar las operaciones de obtener las órdenes para cada cliente, obtener el plato fuerte del menú de cada orden, verificar la cantidad de ordenes por cliente y responder solo la información del cliente, costaría como mínimo un recorrido a cada uno de esos arreglos. Asumiendo que cada uno es de tamaño n, sería o(n4)

Además, los objetos deberían estar en un inicio en algún sitio de memoria principal, si están en el ordenador o en un servidor de DB, en este caso, de Oracle. El costo de carga de estos objetos también se asociaría.

Después de realizar estas descripciones, esta tabla resume cómo se realizaría cada proceso en cuanto a ejecutar la sentencia y pedir los datos por la conexión o realizar todas las acciones en memoria principal.

|  |  |  |
| --- | --- | --- |
|  | SQL | JAVA + Memoria principal. |
| Ubicación datos. | Están ubicados en el servidor, el mismo que ejecuta las sentencias. | Se ubican en el servidor de la DB, puede ser local o remoto. Remoto en este caso. |
| Procesamiento datos. | Realizado tras convertir a algebra relacional, utiliza los índices y estructuras de datos optimizadas. | Manual, con funciones como while, if. Aumenta enormemente el costo del algoritmo. |
| Selección respuesta. | El select, toma algunas columnas de las involucradas en la sentencia. Se puede adaptar a lo ya existente en el programa, clientes para el RF12. | Se debe realizar sobre los objetos realizados en los recorridos, regularmente, creando otros. VO para esta implementación. |
| ¿Cuándo se pasan los objetos a memoria del programa? | Se pasan a memoria principal, al programa, una vez hayan sido procesados, los clientes por sus criterios en el RF12. | Todos son transferidos después de iniciar la transacción y procesados en la memoria del sistema. |

Tabla x: comparativa SQL vs trabajar todo en memoria principal con java.