

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

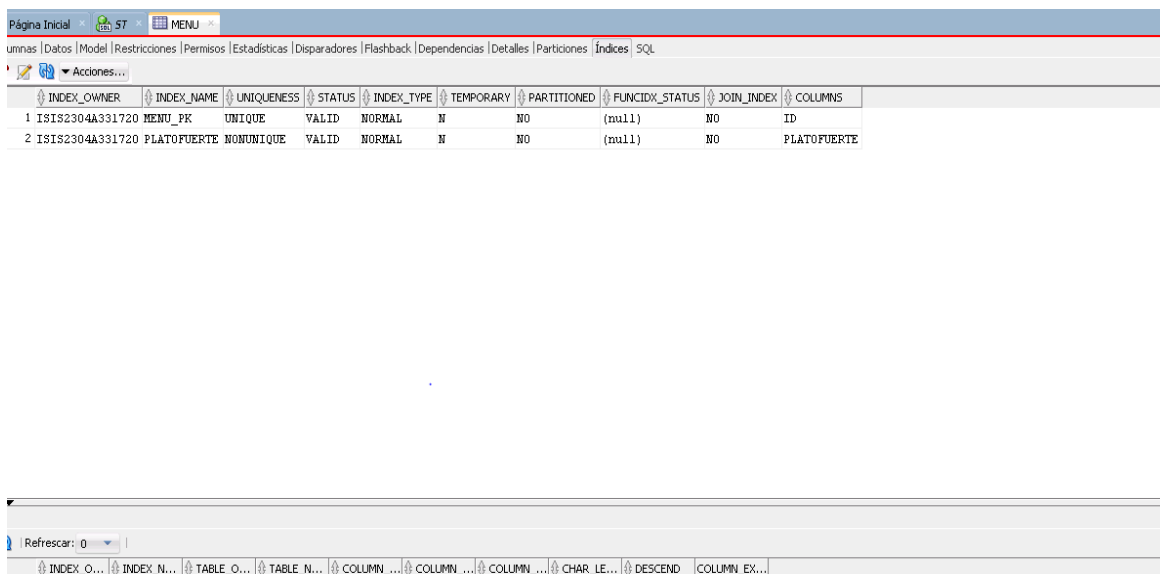
Incluya una foto de pantalla con la información generada por Oracle asociada a los índices existentes.

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



INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCTION_STATUS	JOIN_INDEX	COLUMNS
ISI82304A331720	MENU_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID
ISI82304A331720	PLATOFUERTE	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	PLATOFUERTE

Imagen 1: índices de la tabla menú.

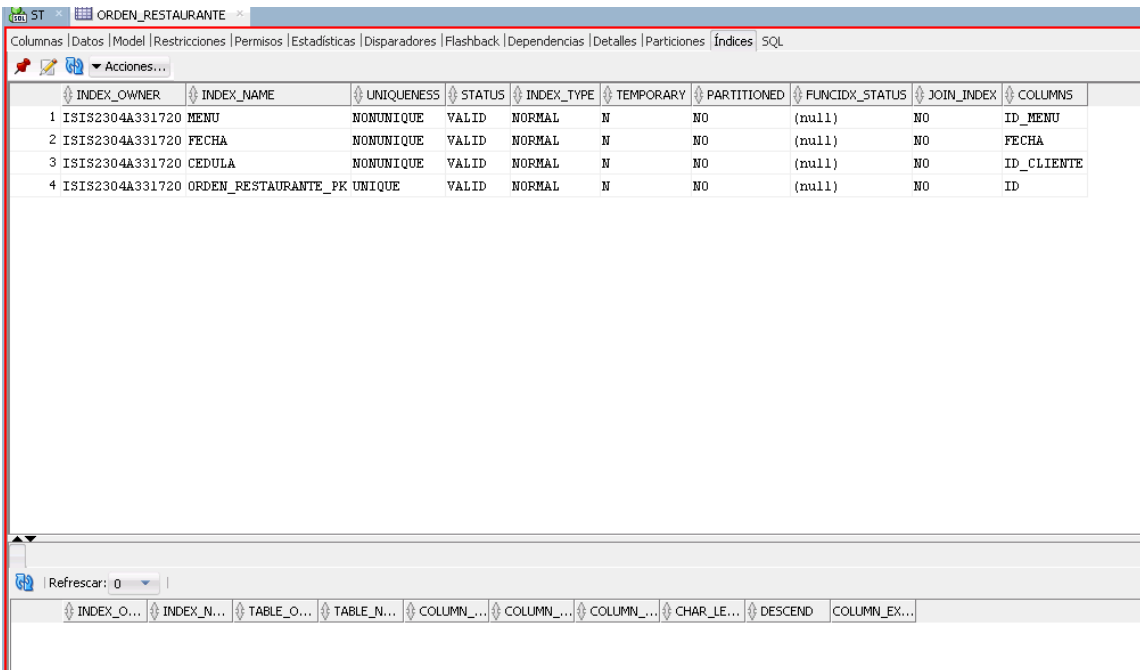
Índices de la tabla.

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.



	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A331720	MENU	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_MENU
2	ISIS2304A331720	FECHA	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	FECHA
3	ISIS2304A331720	CEDULA	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_CLIENTE
4	ISIS2304A331720	ORDEN_RESTAURANTE_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

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.

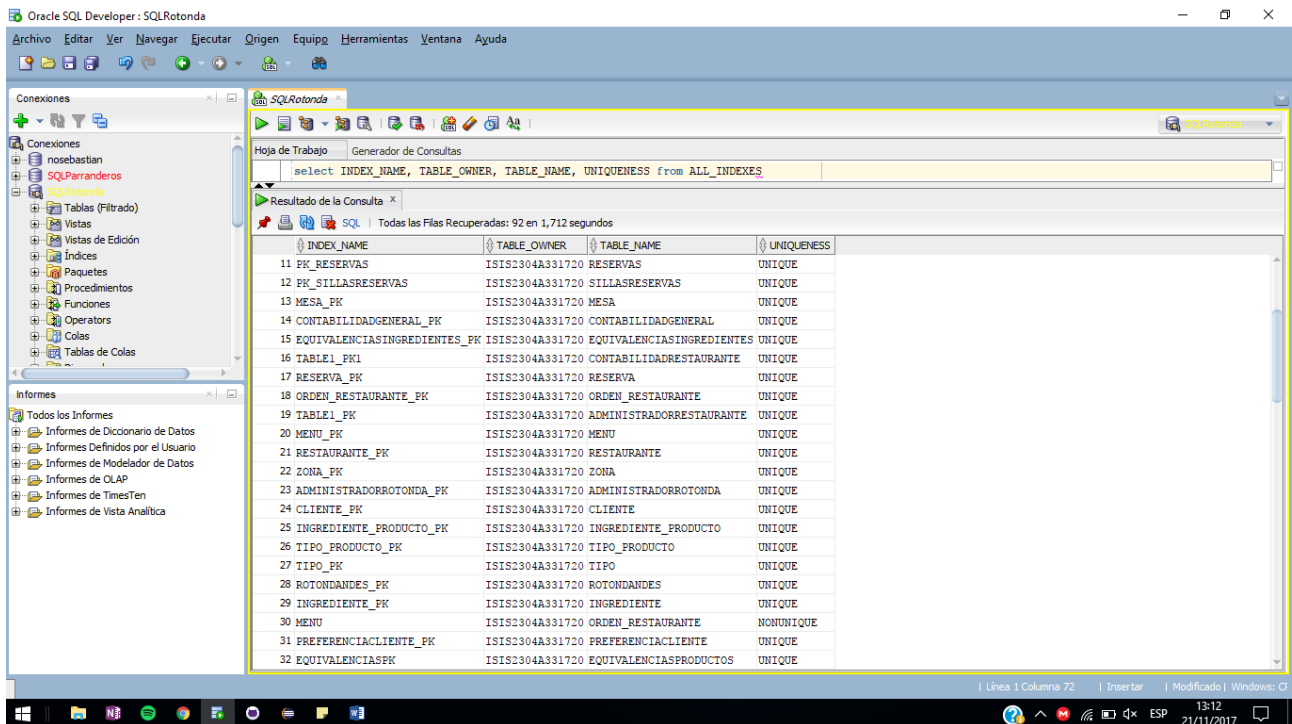
Orden Restaurante menú: Es un índice sobre la columna de menú en cada orden. Este índice no es único, pero es un buen índice debido a que es seleccionado en los planes para realizar los joins de los requerimientos. Además, la cantidad de menús, si bien no es alta, hay unos 100000 menús, la selectividad 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.



Oracle SQL Developer: SQLRotonda

Archivo Editar Ver Navegar Ejecutar Origen Equipo Herramientas Ventana Ayuda

Conexiones

SQLRotonda

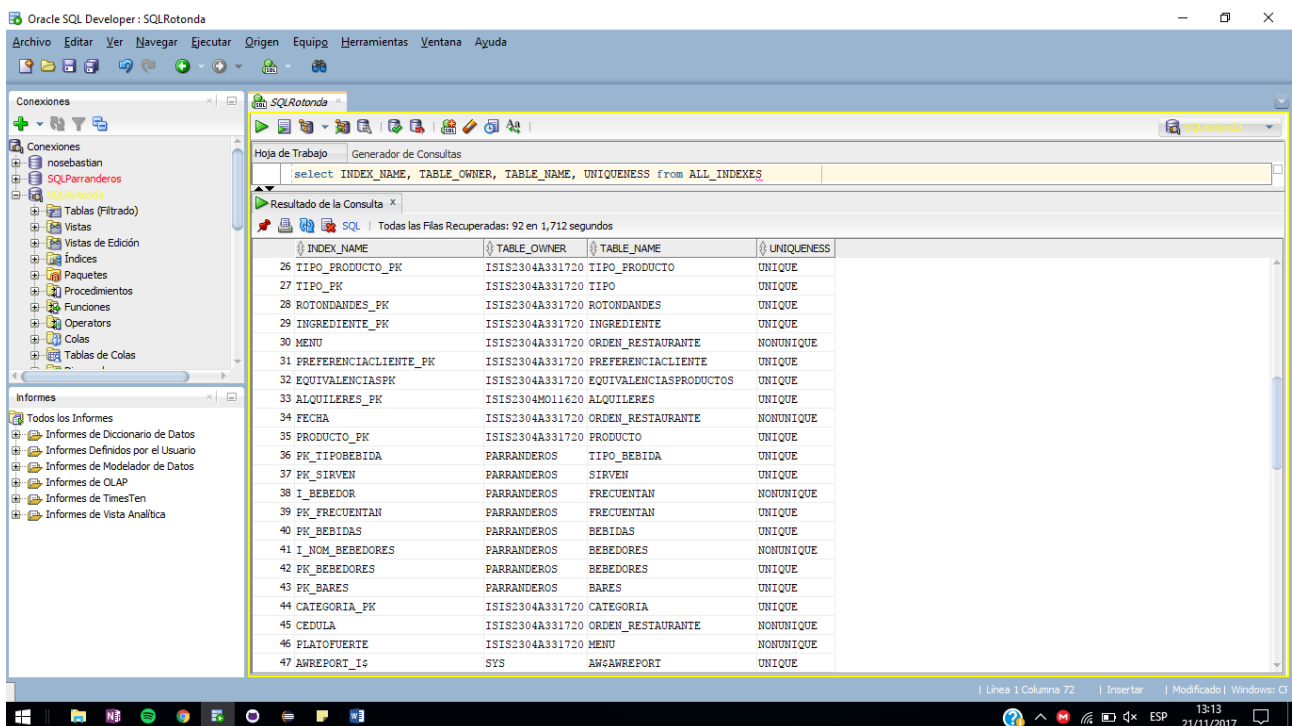
Hoja de Trabajo Generador de Consultas

`select INDEX_NAME, TABLE_OWNER, TABLE_NAME, UNIQUENESS from ALL_INDEXES`

Resultado de la Consulta

Todas las Filas Recuperadas: 92 en 1,712 segundos

INDEX_NAME	TABLE_OWNER	TABLE_NAME	UNIQUENESS
11 FK_RESERVAS	ISIS2304A331720	RESERVAS	UNIQUE
12 PK_SILLASRESERVAS	ISIS2304A331720	SILLASRESERVAS	UNIQUE
13 MESA_FK	ISIS2304A331720	MESA	UNIQUE
14 CONTABILIDADGENERAL_FK	ISIS2304A331720	CONTABILIDADGENERAL	UNIQUE
15 EQUIVALENCIASINGREDIENTES_FK	ISIS2304A331720	EQUIVALENCIASINGREDIENTES	UNIQUE
16 TABLE_FK1	ISIS2304A331720	CONTABILIDADRESTAURANTE	UNIQUE
17 RESERVA_FK	ISIS2304A331720	RESERVA	UNIQUE
18 ORDEN_RESTAURANTE_FK	ISIS2304A331720	ORDEN_RESTAURANTE	UNIQUE
19 TABLE_FK	ISIS2304A331720	ADMINISTRADORRESTAURANTE	UNIQUE
20 MENU_FK	ISIS2304A331720	MENU	UNIQUE
21 RESTAURANTE_FK	ISIS2304A331720	RESTAURANTE	UNIQUE
22 ZONA_FK	ISIS2304A331720	ZONA	UNIQUE
23 ADMINISTRADORROTONDA_FK	ISIS2304A331720	ADMINISTRADORROTONDA	UNIQUE
24 CLIENTE_FK	ISIS2304A331720	CLIENTE	UNIQUE
25 INGREDIENTE_PRODUCTO_FK	ISIS2304A331720	INGREDIENTE_PRODUCTO	UNIQUE
26 TIPO_PRODUCTO_FK	ISIS2304A331720	TIPO_PRODUCTO	UNIQUE
27 TIPO_FK	ISIS2304A331720	TIPO	UNIQUE
28 ROTONDANDES_FK	ISIS2304A331720	ROTONDANDES	UNIQUE
29 INGREDIENTE_FK	ISIS2304A331720	INGREDIENTE	UNIQUE
30 MENU	ISIS2304A331720	ORDEN_RESTAURANTE	NONUNIQUE
31 PREFERENCIACLIENTE_FK	ISIS2304A331720	PREFERENCIACLIENTE	UNIQUE
32 EQUIVALENCIASPK	ISIS2304A331720	EQUIVALENCIASPRODUCTOS	UNIQUE



Oracle SQL Developer: SQLRotonda

Archivo Editar Ver Navegar Ejecutar Origen Equipo Herramientas Ventana Ayuda

Conexiones

SQLRotonda

Hoja de Trabajo Generador de Consultas

`select INDEX_NAME, TABLE_OWNER, TABLE_NAME, UNIQUENESS from ALL_INDEXES`

Resultado de la Consulta

Todas las Filas Recuperadas: 92 en 1,712 segundos

26 TIPO_PRODUCTO_FK	ISIS2304A331720	TIPO_PRODUCTO	UNIQUE
27 TIPO_FK	ISIS2304A331720	TIPO	UNIQUE
28 ROTONDANDES_FK	ISIS2304A331720	ROTONDANDES	UNIQUE
29 INGREDIENTE_FK	ISIS2304A331720	INGREDIENTE	UNIQUE
30 MENU	ISIS2304A331720	ORDEN_RESTAURANTE	NONUNIQUE
31 PREFERENCIACLIENTE_FK	ISIS2304A331720	PREFERENCIACLIENTE	UNIQUE
32 EQUIVALENCIASPK	ISIS2304A331720	EQUIVALENCIASPRODUCTOS	UNIQUE
33 ALQUILERES_FK	ISIS2304M011620	ALQUILERES	UNIQUE
34 FECHA	ISIS2304A331720	ORDEN_RESTAURANTE	NONUNIQUE
35 PRODUCTO_FK	ISIS2304A331720	PRODUCTO	UNIQUE
36 PK_TIPOBEBIDA	PARRANDEROS	TIPO_BEBIDA	UNIQUE
37 PK_SIRVEN	PARRANDEROS	SIRVEN	UNIQUE
38 I_BEBEDOR	PARRANDEROS	FRECUELTAN	NONUNIQUE
39 FK_FRECUELTAN	PARRANDEROS	FRECUELTAN	UNIQUE
40 FK_BEBIDAS	PARRANDEROS	BEBIDAS	UNIQUE
41 I_NOM_BEBEDORES	PARRANDEROS	BEBEDORES	NONUNIQUE
42 FK_BEBEDORES	PARRANDEROS	BEBEDORES	UNIQUE
43 FK_BARES	PARRANDEROS	BARES	UNIQUE
44 CATEGORIA_FK	ISIS2304A331720	CATEGORIA	UNIQUE
45 CEDULA	ISIS2304A331720	ORDEN_RESTAURANTE	NONUNIQUE
46 PLATOFUERTE	ISIS2304A331720	MENU	NONUNIQUE
47 AWREPORT_I4	SYS	AW4AWREPORT	UNIQUE

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 SQLDeveloper.
- 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 ?;
```

- **Plan Oracle**

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	2004
SELECT STATEMENT				379	2404
SORT		ORDER BY		379	2404
HASH		UNIQUE		379	2403
HASH JOIN		OUTER		1017	2402
Access Predicates					
CLIENTE.CEDULA(+) = ORDEN_RESTAURANTE.ID_CLIENTE					
HASH JOIN				1017	448
Access Predicates					
MENU.ID = ORDEN_RESTAURANTE.ID_MENU					
NESTED LOOPS				1017	448
NESTED LOOPS				2052	448
STATISTICS COLLECTOR					
HASH JOIN		SEMI		4	34
Access Predicates					
MENU.NOMBRE_RESTAURANTE = PRODUCTO.NOMBRE_RESTAURANTE					
TABLE ACCESS <u>MENU</u>		FULL		4	15
Filter Predicates					
MENU.NOMBRE_RESTAURANTE = 'Chicas rest'					
TABLE ACCESS <u>PRODUCTO</u>		FULL		16	19
Filter Predicates					
PRODUCTO.NOMBRE_RESTAURANTE = 'Chicas rest'					
INDEX <u>MENU</u>		RANGE SCAN		513	2
Access Predicates					
MENU.ID = ORDEN_RESTAURANTE.ID_MENU					
TABLE ACCESS <u>ORDEN_RESTAURANTE</u>		BY INDEX ROWID		254	408
Filter Predicates					
AND					
ORDEN_RESTAURANTE.FECHA >= TO_DATE(' 2010-12-11 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.FECHA < TO_DATE(' 2018-12-17 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.ID_CLIENTE IS NOT NULL					
TABLE ACCESS <u>ORDEN_RESTAURANTE</u>		FULL		254	408
Filter Predicates					
AND					
ORDEN_RESTAURANTE.FECHA >= TO_DATE(' 2010-12-11 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.FECHA < TO_DATE(' 2018-12-17 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.ID_CLIENTE IS NOT NULL					
TABLE ACCESS <u>CLIENTE</u>		FULL		1050188	195

```

    {info}
      info type="db_version"
        12.1.0.2
      info type="parse_schema"
        "TIS22304A331720"
      info type="dynamic_sampling" note="y"
        2
      info type="plan_hash_full"
        1105516967
      info type="plan_hash"
        1792663728
      info type="plan_hash_2"
        3343500937
      info type="adaptive_plan" note="y"
        yes
      {v}
        11772725536356023955
        2394384430148205229
        10250827045396034032
        16710064274869408602
        10782582093559245556
        4333359815089530554
        8858206635798969530
        4
        3
      {hint}
        PARTIAL_JOIN(@"SEL$0FCD1ED0" "PRODUCTO"@"SEL$1")
        USE_HASH_AGGREGATION(@"SEL$0FCD1ED0")
        USE_HASH(@"SEL$0FCD1ED0" "CLIENTE"@"SEL$3")
        NLJ_BATCHING(@"SEL$0FCD1ED0" "ORDEN_RESTAURANTE"@"SEL$2")
        USE_NL(@"SEL$0FCD1ED0" "ORDEN_RESTAURANTE"@"SEL$2")
        USE_HASH(@"SEL$0FCD1ED0" "PRODUCTO"@"SEL$1")
        LEADING(@"SEL$0FCD1ED0" "MENU"@"SEL$1" "PRODUCTO"@"SEL$1" "ORDEN_RESTAURANTE"@"SEL$2" "CLIENTE"@"SEL$3")
        FULL(@"SEL$0FCD1ED0" "CLIENTE"@"SEL$3")
        INDEX(@"SEL$0FCD1ED0" "ORDEN_RESTAURANTE"@"SEL$2" ("ORDEN_RESTAURANTE","ID_MENU"))
        FULL(@"SEL$0FCD1ED0" "PRODUCTO"@"SEL$1")
        FULL(@"SEL$0FCD1ED0" "MENU"@"SEL$1")
        OUTLINE(@"SEL$2")
        OUTLINE(@"SEL$1")
        ANSI_REARCH(@"SEL$2")
        OUTLINE(@"SEL$948754D7")
  
```

Imagen 4: Plan de ejecución RFC9.

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 (treintauno 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 $0 \leq X \leq 100$, 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

<pre> SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO, MENU.NOMBRE_RESTAURANTE 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 >= '01-01-01' AND FECHA < '01-01-19' AND MENU.NOMBRE_RESTAURANTE = 'Chicas rest' ORDER BY cedula; </pre>			
<p>Explicación del Plan x Resultado de la Consulta x</p> <p>Se han recuperado 50 filas en 0.288 segundos</p>			
CEDULA	NOMBRE	CORREO	NOMBRE_RESTAURANTE
1	1 Lazare Bedminster	lbedminster5j@yahoo.co.jp	Chicas rest
2	2 Nana McEniry	nmceniry0@dell.com	Chicas rest
3	3 Saba Musgrove	smusgrove1@soundcloud.com	Chicas rest
4	4 Christoph Kebbell	ckebbell2@yale.edu	Chicas rest
5	5 Virgilio Van Der Hoog	vvan3@microsoft.com	Chicas rest
6	6 Dinnie Sills	dsills4@nature.com	Chicas rest
7	7 Brockie Blakden	bblakden5@barnesandnoble.com	Chicas rest
8	8 Ashien Shearn	ashearn6@cisco.com	Chicas rest
9	9 Paige Balston	pbalston7@slate.com	Chicas rest
10	10 Lanita Danielis	ldanielis8@linkedin.com	Chicas rest
11	11 Coralyn Rivallant	crivallant9@ebay.co.uk	Chicas rest
12	12 Giacomo Dreger	gdregera@jugem.jp	Chicas rest
13	13 Abbey Hatliffe	ahatliffeb@is.gd	Chicas rest
14	14 Godart Akred	gakredc@amazon.co.jp	Chicas rest
15	15 Gail Hamber	ghamberd@washington.edu	Chicas rest
16	16 Felita Petroff	fpetroffe@army.mil	Chicas rest
17	17 Stevie Bath	sbathf@cocolog-nifty.com	Chicas rest
18	18 Leora Shotton	lshottong@wsj.com	Chicas rest
19	19 Charyl Hylton	chyltonh@spiegel.de	Chicas rest
20	20 Trixie Portman	tportmani@shinystat.com	Chicas rest
21	21 Noelani Climar	nclimarj@nydailynews.com	Chicas rest
22	22 Lane Spalding	lspaldingk@theguardian.com	Chicas rest
23	23 Vitoria Dallmann	vdallmannl@bloomberg.com	Chicas rest
24	24 Demeter Balf	dbalfm@intel.com	Chicas rest
25	25 Ramon Feldman	rfeldmann@fastcompany.com	Chicas rest
26	26 Janetta Smithen	jsmitheno@shareasale.com	Chicas rest
27	27 Garland Alvarado	galvaradop@slashdot.org	Chicas rest
28	28 Dannie Sidlow	dsidlowq@etsy.com	Chicas rest
29	29 Arney Burne	aburner@loc.gov	Chicas rest
30	30 Bell Causbey	bcausbeys@mitbeian.gov.cn	Chicas rest
31	31 Clarence Rorke	crorket@sogou.com	Chicas rest
32	32 Natty Lovelady	nloveladyu@unblog.fr	Chicas rest
33	33 Heidie Sindall	hsindallv@instagram.com	Chicas rest
34	34 Marcellina McAline	mmcalinew@aol.com	Chicas rest
35	35 Wildon Niebat	wniebatv@amdoo.com	Chicas rest

Imagen 5: Ejecución RFC9 con los parámetros especificados anteriormente.

- **Nombre restaurante:** HH
Fecha mínima: 01/01/01
Fecha máxima: 02/01/01
Tiempo ejecución: 0.028 segundos

Hoja de Trabajo

Generador de Consultas

```

SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO, MENU.NOMBRE_RESTAURANTE
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 >= '01-01-01' AND FECHA < '02-01-01' AND MENU.NOMBRE_RESTAURANTE = 'HH'
ORDER BY cedula;

```

Explicación del Plan

Resultado de la Consulta

SQL

Se han recuperado 50 filas en 0.028 segundos

	CEDULA	NOMBRE	CORREO	NOMBRE_RESTAURANTE
1	1004709	USER1004709	USER1004709@CORREO.COM	HH
2	1011033	USER1011033	USER1011033@CORREO.COM	HH
3	1017357	USER1017357	USER1017357@CORREO.COM	HH
4	1023681	USER1023681	USER1023681@CORREO.COM	HH
5	1030005	USER1030005	USER1030005@CORREO.COM	HH
6	1042653	USER1042653	USER1042653@CORREO.COM	HH
7	1055301	USER1055301	USER1055301@CORREO.COM	HH
8	1074273	USER1074273	USER1074273@CORREO.COM	HH
9	1080597	USER1080597	USER1080597@CORREO.COM	HH
10	1086921	USER1086921	USER1086921@CORREO.COM	HH
11	1093245	USER1093245	USER1093245@CORREO.COM	HH
12	1099569	USER1099569	USER1099569@CORREO.COM	HH
13	1105893	USER1105893	USER1105893@CORREO.COM	HH
14	1112217	USER1112217	USER1112217@CORREO.COM	HH
15	1124865	USER1124865	USER1124865@CORREO.COM	HH
16	1137513	USER1137513	USER1137513@CORREO.COM	HH
17	1150161	USER1150161	USER1150161@CORREO.COM	HH
18	1156485	USER1156485	USER1156485@CORREO.COM	HH
19	1162809	USER1162809	USER1162809@CORREO.COM	HH
20	1169133	USER1169133	USER1169133@CORREO.COM	HH
21	1175457	USER1175457	USER1175457@CORREO.COM	HH
22	1181781	USER1181781	USER1181781@CORREO.COM	HH
23	1188105	USER1188105	USER1188105@CORREO.COM	HH
24	1200753	USER1200753	USER1200753@CORREO.COM	HH
25	1213401	USER1213401	USER1213401@CORREO.COM	HH
26	1226049	USER1226049	USER1226049@CORREO.COM	HH
27	1232373	USER1232373	USER1232373@CORREO.COM	HH
28	1238697	USER1238697	USER1238697@CORREO.COM	HH
29	1245021	USER1245021	USER1245021@CORREO.COM	HH
30	1251345	USER1251345	USER1251345@CORREO.COM	HH
31	1257669	USER1257669	USER1257669@CORREO.COM	HH
32	1263993	USER1263993	USER1263993@CORREO.COM	HH
33	1308261	USER1308261	USER1308261@CORREO.COM	HH
34	1314585	USER1314585	USER1314585@CORREO.COM	HH
35	1320909	USER1320909	USER1320909@CORREO.COM	HH

Imagen 6: Ejecución RFC9 con los parámetros especificados anteriormente.

- **Nombre restaurante:** Chicas rest
- Fecha mínima:** 01/01/01
- Fecha máxima:** 30/06/05
- Tiempo ejecución:** 0.211 segundos

<pre> SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO, MENU.NOMBRE_RESTAURANTE 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 >= '01-01-01' AND FECHA < '30-06-05' AND MENU.NOMBRE_RESTAURANTE = 'Chicas rest' ORDER BY cedula; </pre>				
<div> <div>Explicación del Plan</div> <div>Resultado de la Consulta</div> </div> <div> <div>Todas las Filas Recuperadas: 38 en 0.211 segundos</div> </div>				
CEDULA	NOMBRE	CORREO	NOMBRE_RESTAURANTE	
1	1 Lazare Bedminster	lbedminster5j@yahoo.co.jp	Chicas rest	
2	2 Nana McEniry	nmceniry0@dell.com	Chicas rest	
3	51 Paxon Edgworth	pedgworth1d@yelp.com	Chicas rest	
4	52 Elston Gaitung	egaitungle@statcounter.com	Chicas rest	
5	101 Sherwin Effnert	seffnert2r@jugem.jp	Chicas rest	
6	151 Melli McCarrison	mmccarrison45@sakura.ne.jp	Chicas rest	
7	201 Maria Juanes	mj@correo.com	Chicas rest	
8	251 Aurelia Skym	askym6x@plala.or.jp	Chicas rest	
9	252 Ilse Benois	ibenois6y@domainmarket.com	Chicas rest	
10	301 Giffard Killingbeck	gkillingbeck8b@imgur.com	Chicas rest	
11	302 Filberte Discombe	fdiscombe8c@globo.com	Chicas rest	
12	351 Chloris Stockey	cstockey9p@mtv.com	Chicas rest	
13	352 Dex Coen	dcoen9q@people.com.cn	Chicas rest	
14	401 Olvan Happert	ohappertb3@skyrock.com	Chicas rest	
15	451 Briano Ponder	bponderch@un.org	Chicas rest	
16	452 Shem Collop	scollopai@miibeian.gov.cn	Chicas rest	
17	501 Wilie Dyers	wdyersdv@ucoz.com	Chicas rest	
18	502 Barny Widdows	bwiddowsdw@fastcompany.com	Chicas rest	
19	551 Koressa Fenning	kfenningf9@imageshack.us	Chicas rest	
20	552 Shawn Janway	sjanwayfa@foxnews.com	Chicas rest	
21	601 Madelena Heineking	mheinekingn@posterous.com	Chicas rest	
22	651 Audry Cuphus	acuphusi1@vistaprint.com	Chicas rest	
23	701 Danny Kermannes	dkermannesjf@uol.com.br	Chicas rest	
24	801 Dreddy Winkle	dwinklem7@youku.com	Chicas rest	
25	802 Cesar Tucsell	ctucsellm8@studiopress.com	Chicas rest	
26	851 Baldwin Bagenal	bbagenaln1@facebook.com	Chicas rest	
27	901 Fonzie Sampey	fsampeyoz@fastcompany.com	Chicas rest	
28	951 Albert Itscovitz	aitscovitzqd@diigo.com	Chicas rest	
29	1010054402 Juan Juanes	correoprueba@correo.com	Chicas rest	
30	1010101010 Juan Jaimes	jj@correo.com	Chicas rest	
31	1090509418 Felipe Juan	fj@correo.com	Chicas rest	
32	1090509468 Daniela Jaimes	da@correo.com	Chicas rest	
33	2010201120 Juliana Perez	jp@correo.com	Chicas rest	
34	2020202020 Maria Juanes	mj@correo.com	Chicas rest	
35	2021202020 Maria Juanes	mj@correo.com	Chicas rest	

Imagen 7: Ejecución RFC9 con los parámetros especificados anteriormente.

- **Nombre restaurante:** Chicas rest
- Fecha mínima:** 01/01/01
- Fecha máxima:** 31/12/16
- Tiempo ejecución:** 0.098 segundos

Hoja de Trabajo

Generador de Consultas

```
SELECT DISTINCT CEDULA, CLIENTE.NOMBRE, CORREO, MENU.NOMBRE_RESTAURANTE
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 >= '01-01-01' AND FECHA < '31-12-16' AND MENU.NOMBRE_RESTAURANTE = 'Chicas rest'
ORDER BY cedula;
```

Explicación del Plan

Resultado de la Consulta

Se han recuperado 50 filas en 0.098 segundos

	CEDULA	NOMBRE	CORREO	NOMBRE_RESTAURANTE
1	1	Lazare Bedminster	lbedminster5j@yahoo.co.jp	Chicas rest
2	2	Nana McEniry	nmceniry0@del1.com	Chicas rest
3	3	Saba Musgrove	smusgrove1@soundcloud.com	Chicas rest
4	4	Christoph Kebbell	ckebbell2@yale.edu	Chicas rest
5	5	Virgilio Van Der Hoog	vvvan3@microsoft.com	Chicas rest
6	6	Dinnie Sills	dsills4@nature.com	Chicas rest
7	7	Brookie Blakden	bblakden5@barnesandnoble.com	Chicas rest
8	8	Ashien Shearn	ashearn6@cisco.com	Chicas rest
9	9	Paige Balston	pbalston7@slate.com	Chicas rest
10	10	Lanita Danielis	ldanielis8@linkedin.com	Chicas rest
11	11	Coralyn Rivallant	crivallant9@ebay.co.uk	Chicas rest
12	12	Giacomo Dreger	gdregera@jugem.jp	Chicas rest
13	13	Abbey Hatliffe	ahatliffeb@is.gd	Chicas rest
14	14	Godart Akred	gakredc@amazon.co.jp	Chicas rest
15	15	Gail Hamber	ghamberd@washington.edu	Chicas rest
16	16	Felita Petroff	fpetroffe@army.mil	Chicas rest
17	17	Stevie Bath	sbathf@cocolog-nifty.com	Chicas rest
18	18	Leora Shotton	lshottong@wsj.com	Chicas rest
19	19	Charyl Hylton	chyltonh@spiegel.de	Chicas rest
20	20	Trixie Portman	tportmani@shinystat.com	Chicas rest
21	21	Noelani Climar	nclimarj@nydailynews.com	Chicas rest
22	22	Lane Spalding	lspaldingk@theguardian.com	Chicas rest
23	23	Vitoria Dallmann	vdallmannl@bloomberg.com	Chicas rest
24	24	Demeter Balf	dbalfm@intel.com	Chicas rest
25	25	Ramon Feldman	rfeldmann@fastcompany.com	Chicas rest
26	26	Janetta Smithen	jsmitheno@shareasale.com	Chicas rest
27	27	Garland Alvarado	galvaradop@slashdot.org	Chicas rest
28	28	Dannie Sidlow	dsidlowg@etsy.com	Chicas rest
29	30	Bell Causbey	bcausbeys@mitbeian.gov.cn	Chicas rest
30	31	Clarence Rorke	crorket@sogou.com	Chicas rest
31	32	Natty Lovelady	nloveladyu@unblog.fr	Chicas rest
32	33	Heidie Sindall	hsindallv@instagram.com	Chicas rest
33	34	Marcellina McAline	mmcalinev@aol.com	Chicas rest
34	35	Wildon Nisbet	wnisbetx@squidoo.com	Chicas rest
35	36	Masul Loundar	mloundaru@163.com	Chicas rest

Imagen 8: Ejecución RFC9 con los parámetros especificados anteriormente.

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 ?;
```

• Plan Oracle

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1050188 33577
MINUS				
SORT				
TABLE ACCESS	CLIENTE	UNIQUE	1050188	13481
SORT				
TABLE ACCESS	CLIENTE	FULL	1050188	1951
NESTED LOOPS				
HASH JOIN		UNIQUE	1	41
HASH JOIN		OUTER	1	40
HASH JOIN		SEMI	1	38
Access Predicates	MENU.NOMBRE_RESTAURANTE=PRODUCTO.NOMBRE_RESTAURANTE			
HASH JOIN			1	19
Access Predicates	MENU.ID=ORDEN_RESTAURANTE.ID_MENU			
NESTED LOOPS			1	19
NESTED LOOPS			4	19
STATISTICS COLLECTOR				
TABLE ACCESS	ORDEN_RESTAURANTE	BY INDEX ROWID BATCHED	4	15
INDEX	FECHA	RANGE SCAN	12	3
Access Predicates	ORDEN_RESTAURANTE.FECHA>=TO_DATE('2018-12-11 00:00:00','yyyy-mm-dd hh24:mi:ss') ORDEN_RESTAURANTE.FECHA<TO_DATE('2018-12-17 00:00:00','yyyy-mm-dd hh24:mi:ss')			
AND				
INDEX	MENU_PK	UNIQUE SCAN	1	0
Access Predicates	MENU.ID=ORDEN_RESTAURANTE.ID_MENU			
TABLE ACCESS	MENU	BY INDEX ROWID	1	1
Filter Predicates	MENU.NOMBRE_RESTAURANTE='Boquitas'			
TABLE ACCESS	MENU	FULL	1	1
Filter Predicates	MENU.NOMBRE_RESTAURANTE='Boquitas'			
TABLE ACCESS	PRODUCTO	FULL	21	19
Filter Predicates	PRODUCTO.NOMBRE_RESTAURANTE='Boquitas'			
TABLE ACCESS	CLIENTE	BY INDEX ROWID	1	2
INDEX	CLIENTE_PK	UNIQUE SCAN	1	1
Access Predicates	CLIENTE.CEDULA(+) = ORDEN_RESTAURANTE.ID_CLIENTE			
Other XML	<pre> {info} info type="db_version" 12.1.0.2 info type="parse_schema" "15152304A331720" info type="plan_hash_full" 1391212209 info type="plan_hash" 861512523 info type="plan_hash_2" 2707857907 info type="adaptive_plan" note="y" yes {v} 11772725536356023955 2394384430148205229 10250827045396034032 4333359815089530554 8858206635798969530 10782582093559245556 16710064274869408602 5 2 {hint} FULL(@SEL\$4 "CLIENTE"@SEL\$4) PARTIAL_JOIN(@SEL\$FB96BC69 "PRODUCTO"@SEL\$1) USE_NL(@SEL\$FB96BC69 "CLIENTE"@SEL\$3) USE_HASH(@SEL\$FB96BC69 "PRODUCTO"@SEL\$1) NL_BATCHING(@SEL\$FB96BC69 "MENU"@SEL\$1) USE_NL(@SEL\$FB96BC69 "MENU"@SEL\$1) LEADING(@SEL\$FB96BC69 "ORDEN_RESTAURANTE"@SEL\$2 "MENU"@SEL\$1 "PRODUCTO"@SEL\$1 "CLIENTE"@SEL\$3) INDEX_RS_ASC(@SEL\$FB96BC69 "CLIENTE"@SEL\$3 ("CLIENTE"."CEDULA")) FULL(@SEL\$FB96BC69 "PRODUCTO"@SEL\$1) INDEX(@SEL\$FB96BC69 "MENU"@SEL\$1 ("MENU"."ID")) BATCH_TABLE_ACCESS_BY_ROWID(@SEL\$FB96BC69 "ORDEN_RESTAURANTE"@SEL\$2) INDEX_RS_ASC(@SEL\$FB96BC69 "ORDEN_RESTAURANTE"@SEL\$2 ("ORDEN_RESTAURANTE"."FECHA")) OUTLINE(@SEL\$2) OUTLINE(@SEL\$1) ANSI_REARCH(@SEL\$2) OPTIMIZER_FEATURES_ENABLE('19.0.0') </pre>			

Imagen 9: Plan de ejecución del RFC10.

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 (treintauno 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 $0 \leq X \leq 100$, 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

```
SELECT CEDULA, CLIENTE.NOMBRE, CORREO
FROM CLIENTE MINUS 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 FECHA >= '10-12-2017' AND FECHA < '10-12-2018' AND MENU.NOMBRE_RESTAURANTE = 'Burger b'
ORDER BY cedula;
```

Explicación del Plan x Resultado de la Consulta x

SQL | Se han recuperado 50 filas en 0,717 segundos

	CEDULA	NOMBRE	CORREO
1	1	Lazare Bedminster	lbedminster5j@yahoo.co.jp
2	2	Nana McEniry	nmceniry0@deli.com
3	3	Saba Musgrove	smusgrove1@soundcloud.com
4	4	Christoph Kebbell	ckebbell2@yale.edu
5	5	Virgilio Van Der Hoog	vvan3@microsoft.com
6	6	Dinnie Sills	dsills4@nature.com
7	7	Brookie Blakden	bblakden5@barnesandnoble.com
8	8	Ashien Shearn	ashearn6@cisco.com
9	9	Paige Balston	pbalston7@slate.com
10	10	Lanita Danielis	ldanielis8@linkedin.com
11	11	Coralyn Rivallant	crivallant9@ebay.co.uk
12	12	Giacomo Dreger	gdregera@jugem.jp
13	13	Abbey Hatliffe	ahatliffeb@is.gd
14	14	Godart Akred	gakredc@amazon.co.jp
15	15	Gail Hamber	ghamberd@washington.edu
16	16	Felita Petroff	fpetroffe@army.mil
17	17	Stevie Bath	sbathf@cocolog-nifty.com
18	18	Leora Shotton	lshottong@wsj.com
19	19	Charyl Hylton	chyltonh@spiegel.de
20	20	Trixie Portman	tportmani@shinystat.com

Imagen 10: Ejecución RFC10 con los parámetros especificados anteriormente.

- **Nombre restaurante:** Chicas rest
Fecha mínima: 10/12/16
Fecha máxima: 10/12/17
Tiempo ejecución: 0.702 segundos

Explicación del Plan		Resultado de la Consulta
Se han recuperado 50 filas en 0,702 segundos		
CEDULA	NOMBRE	CORREO
1	1 Lazare Bedminster	lbedminster5j@yahoo.co.jp
2	2 Nana McEniry	nmceniry0@deli.com
3	3 Saba Musgrove	smusgrovel@soundcloud.com
4	4 Christoph Kebbell	ckebbell12@yale.edu
5	5 Virgilio Van Der Hoog	vvan3@microsoft.com
6	6 Dinnie Sills	dsills4@nature.com
7	7 Brookie Blakden	bblakden5@barnesandnoble.com
8	8 Ashien Shearn	ashearn6@cisco.com
9	9 Paige Balston	pbalston7@slate.com
10	10 Lanita Danielis	ldanielis8@linkedin.com
11	11 Coralyn Rivallant	crivallant9@ebay.co.uk
12	12 Giacomo Dreger	gdregera@jugem.jp
13	13 Abbey Hatliffe	ahatliffeb@is.gd
14	14 Godart Akred	gakredc@amazon.co.jp
15	15 Gail Hamber	ghamberd@washington.edu
16	16 Felita Petroff	fpetroff@army.mil
17	17 Stevie Bath	sbathf@cocolog-nifty.com
18	18 Leora Shotton	lshottong@wsj.com
19	19 Charyl Hylton	chyltonh@spiegel.de
20	20 Trixie Portman	tportman1@shinystat.com

- **Nombre restaurante:** Chicas rest
- Fecha mínima:** 11/12/18
- Fecha máxima:** 17/12/18
- Tiempo ejecución:** 0.687 segundos

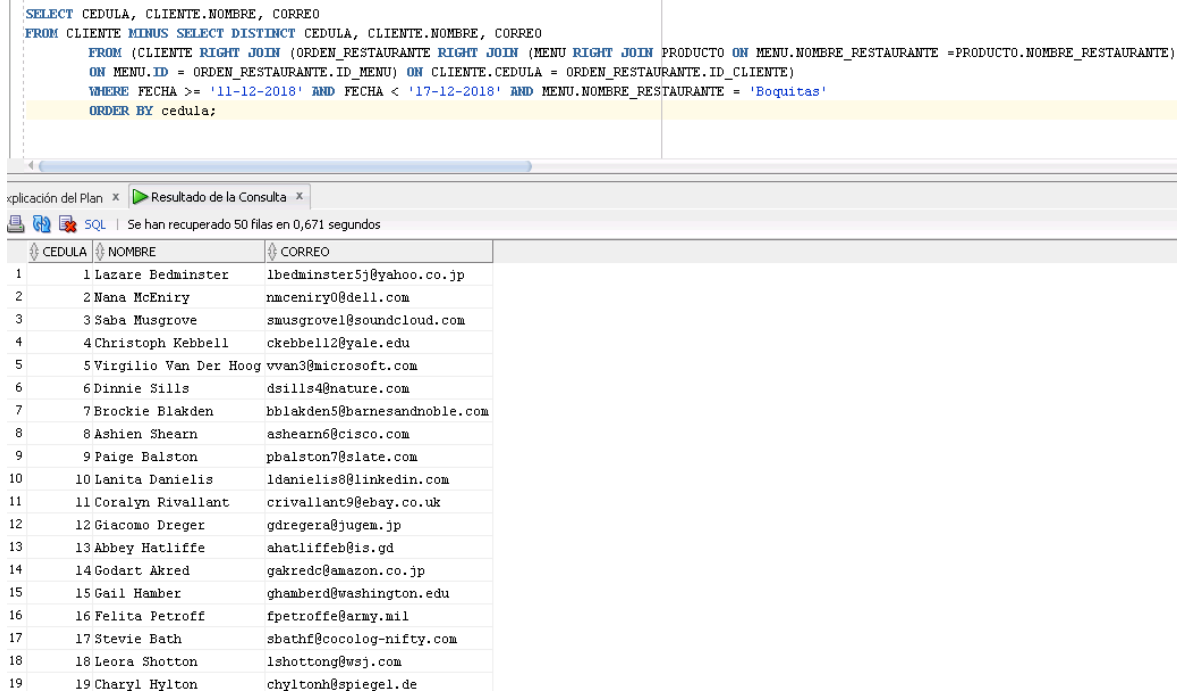
Explicación del Plan x Resultado de la Consulta x

SQL | Se han recuperado 50 filas en 0,687 segundos

CEDULA	NOMBRE	CORREO
1	1 Lazare Bedminster	lbedminster5j@yahoo.co.jp
2	2 Nana McEniry	nmceniry0@dell.com
3	3 Saba Musgrove	smusgrove1@soundcloud.com
4	4 Christoph Kebbell	ckebbell12@yale.edu
5	5 Virgilio Van Der Hoog	vvan3@microsoft.com
6	6 Dinnie Sills	dsills4@nature.com
7	7 Brockie Blakden	bblakden5@barnesandnoble.com
8	8 Ashien Shearn	ashearn6@cisco.com
9	9 Paige Balston	pbalston7@slate.com
10	10 Lanita Danielis	ldanielis8@linkedin.com
11	11 Coralyn Rivallant	crivallant9@ebay.co.uk
12	12 Giacomo Dreger	gdregera@jugem.jp
13	13 Abbey Hatliffe	ahatliffe@is.gd
14	14 Godart Akred	gakredc@amazon.co.jp
15	15 Gail Hamber	ghamberd@washington.edu
16	16 Felita Petroff	fpetroffe@army.mil
17	17 Stevie Bath	sbathf@cocolog-nifty.com
18	18 Leora Shotton	lshottong@wsj.com
19	19 Charyl Hylton	chyltonh@spiegel.de
20	20 Trixie Portman	tportmani@shinystat.com
21	21 Moelani Clinear	mclinear@twinklnews.com

- **Nombre restaurante:** Chicas rest

Tiempo ejecución: 0.671 segundos



```
SELECT CEDULA, CLIENTE.NOMBRE, CORREO
FROM CLIENTE MINUS 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 FECHA >= '11-12-2018' AND FECHA < '17-12-2018' AND MENU.NOMBRE_RESTAURANTE = 'Boquitas'
ORDER BY cedula;
```

	CEDULA	NOMBRE	CORREO
1	1	Lazare Bedminster	lbedminster5j@yahoo.co.jp
2	2	Nana McEniry	nmceniry08dell.com
3	3	Saba Musgrove	smausgrove1@soundcloud.com
4	4	Christoph Kebbell	ckebbell12@yale.edu
5	5	Virgilio Van Der Hoog	vvan3@microsoft.com
6	6	Dinnie Sills	dsills4@nature.com
7	7	Brockie Blakden	bblakden5@barnesandnoble.com
8	8	Ashien Shearn	ashearn6@cisco.com
9	9	Paige Balston	pbalston7@slate.com
10	10	Lanita Danielis	ldanielis8@linkedin.com
11	11	Coralyn Rivallant	crivallant9@ebay.co.uk
12	12	Giacomo Dreger	gdregera@jugem.jp
13	13	Abbey Hatliffe	ahatliffeb@is.gd
14	14	Godart Akred	gakredc@amazon.co.jp
15	15	Gail Hamber	ghamberd@washington.edu
16	16	Felita Petroff	fpetroffe@army.mil
17	17	Stevie Bath	sbathf@cocolog-nifty.com
18	18	Leora Shotton	lshottong@wsj.com
19	19	Charyl Hylton	chyltonh@spiegel.de

Imagen 13: Ejecución RFC10 con los parámetros especificados anteriormente.

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.ACOMPAÑAMIENTO =
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/>

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 descendientemente.

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.

Resultado de la Consulta x Explicación del Plan x

SQL | 1,864 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
VIEW				2
FILTER				
OR				
COUNT(*) >= 3				
AND				
COUNT(*) < 2				
COUNT(*) > 0				
SORT		GROUP BY		2
NESTED LOOPS				42
NESTED LOOPS				8
TABLE ACCESS	ORDEN_RESTAURANTE	BY INDEX ROWID BATCHED		8
INDEX	FECHA	RANGE SCAN		17
Access Predicates				
AND				
ORDEN_RESTAURANTE.FECHA >= TO_DATE(' 2018-12-10 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
ORDEN_RESTAURANTE.FECHA <= TO_DATE(' 2018-12-17 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
TABLE ACCESS	MENU	BY INDEX ROWID		1
INDEX	MENU_PK	UNIQUE SCAN		1
Access Predicates				
MENU.ID = ORDEN_RESTAURANTE.ID_MENU				
TABLE ACCESS	PRODUCTO	FULL		5
Filter Predicates				
OR				

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
TABLE ACCESS	PRODUCTO	FULL		5
Filter Predicates				
OR				
MENU.POSTRE = PRODUCTO.ID				
MENU.PLATOFUERTE = PRODUCTO.ID				
MENU.ACOMPAÑAMIENTO = PRODUCTO.ID				
MENU.BEBIDA = PRODUCTO.ID				
MENU.ENTRADA = PRODUCTO.ID				
Other XML				
(info)				
info type="db_version"				
12.1.0.2				
info type="parse_schema"				
"TIS2304A331720"				
info type="dynamic_sampling" note="y"				
2				
info type="plan_hash_full"				
2640410797				
info type="plan_hash"				
36700515				
info type="plan_hash_2"				
2640410797				
(u)				
17635602895091721057				
10782582093559245556				

Imagen 14: plan de trabajo para el RFC11.

Utiliza los índices creados para él.

Tiempos RFC11.

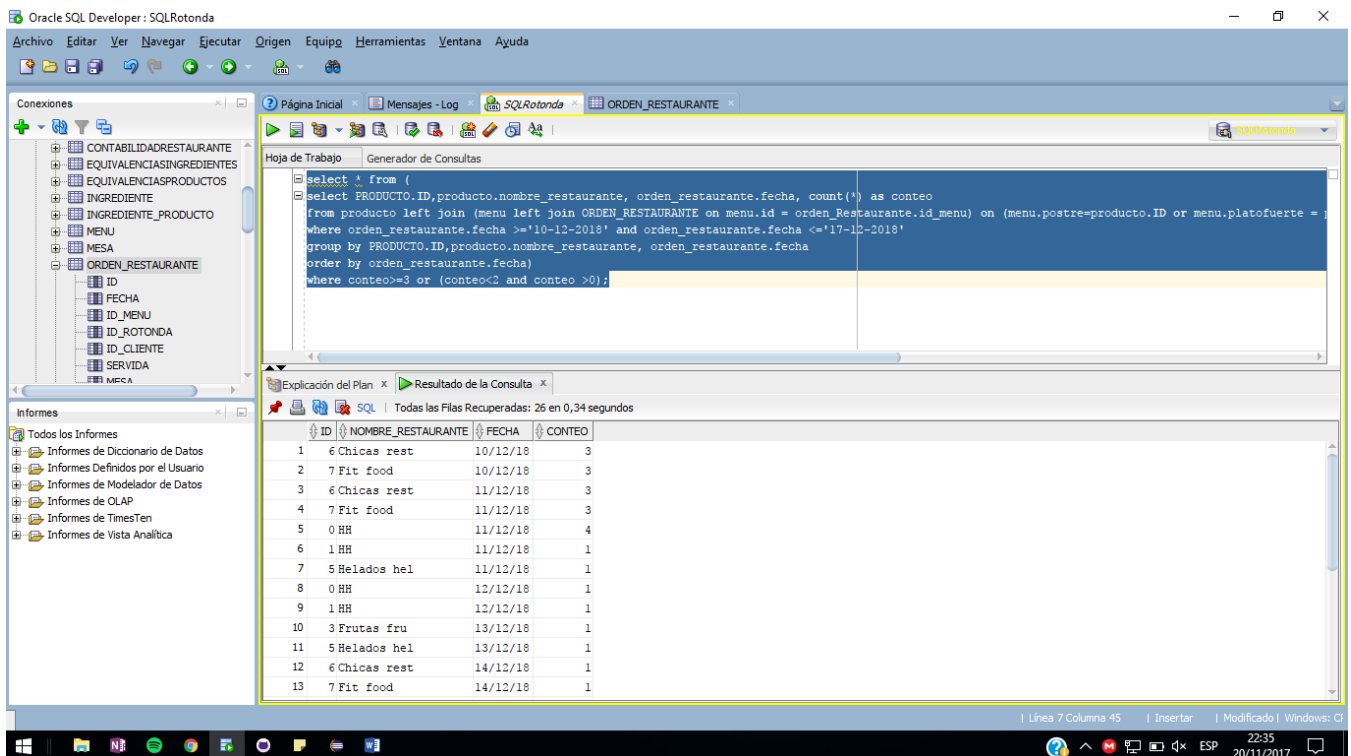


Figura 1: este rango de fechas evalúa las de una semana **Fecha mínima: 10/12/18 Fecha máx: 17/12/18**

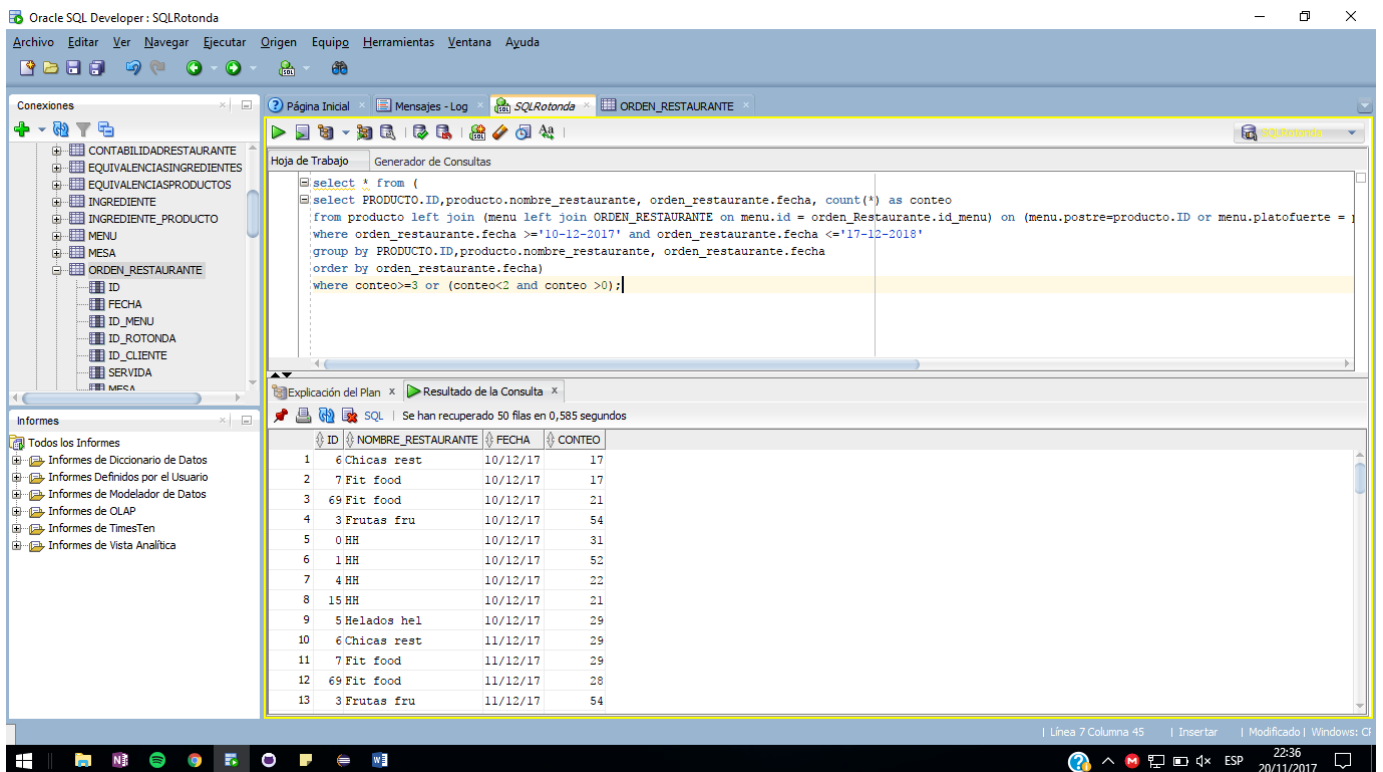


Figura 2: Este rango de fechas evalúa las de varias semanas, 52 **Fecha mínima: 10/12/17 Fecha máxima: 17/12/18**

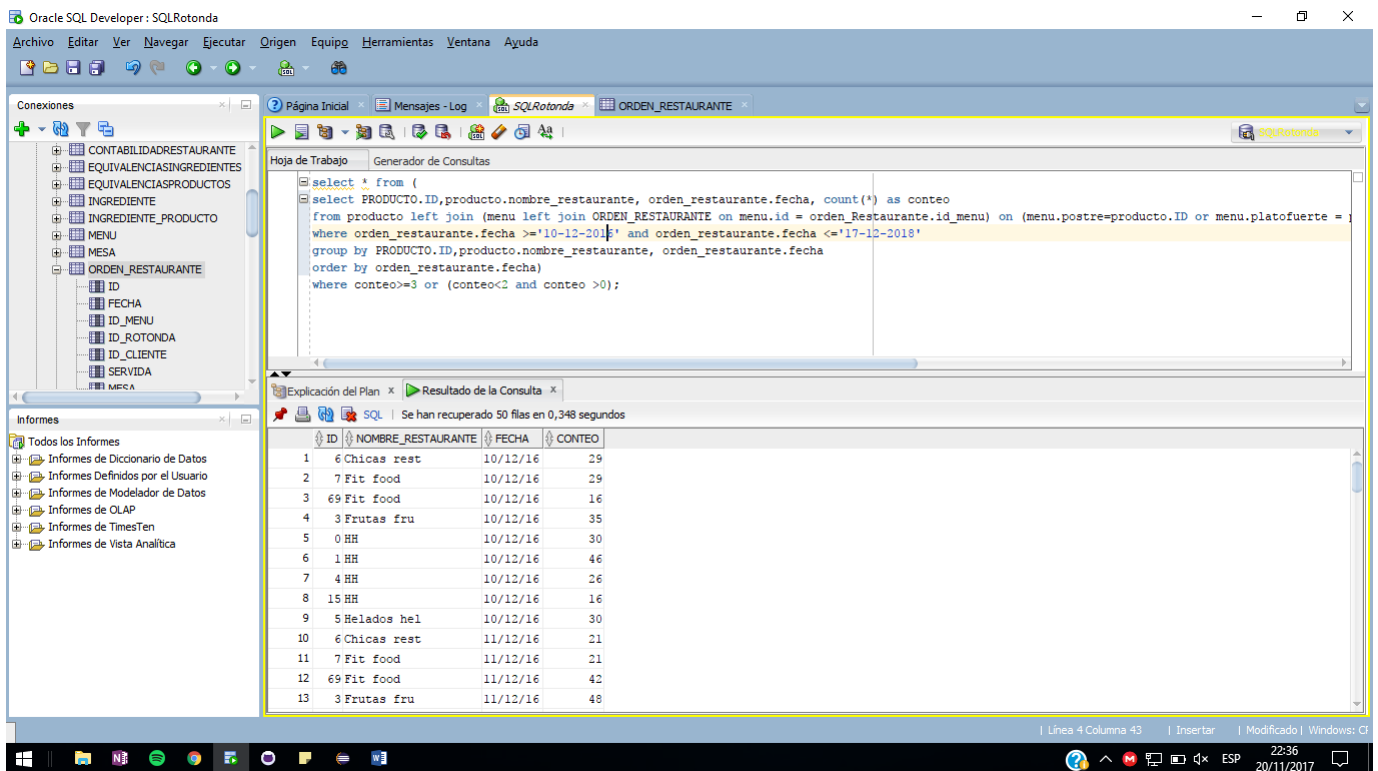


Figura 3: Este rango de fechas evalúa las de varias semanas, 104 Fecha mínima: 1/12/16 Fecha máxima: 17/12/18

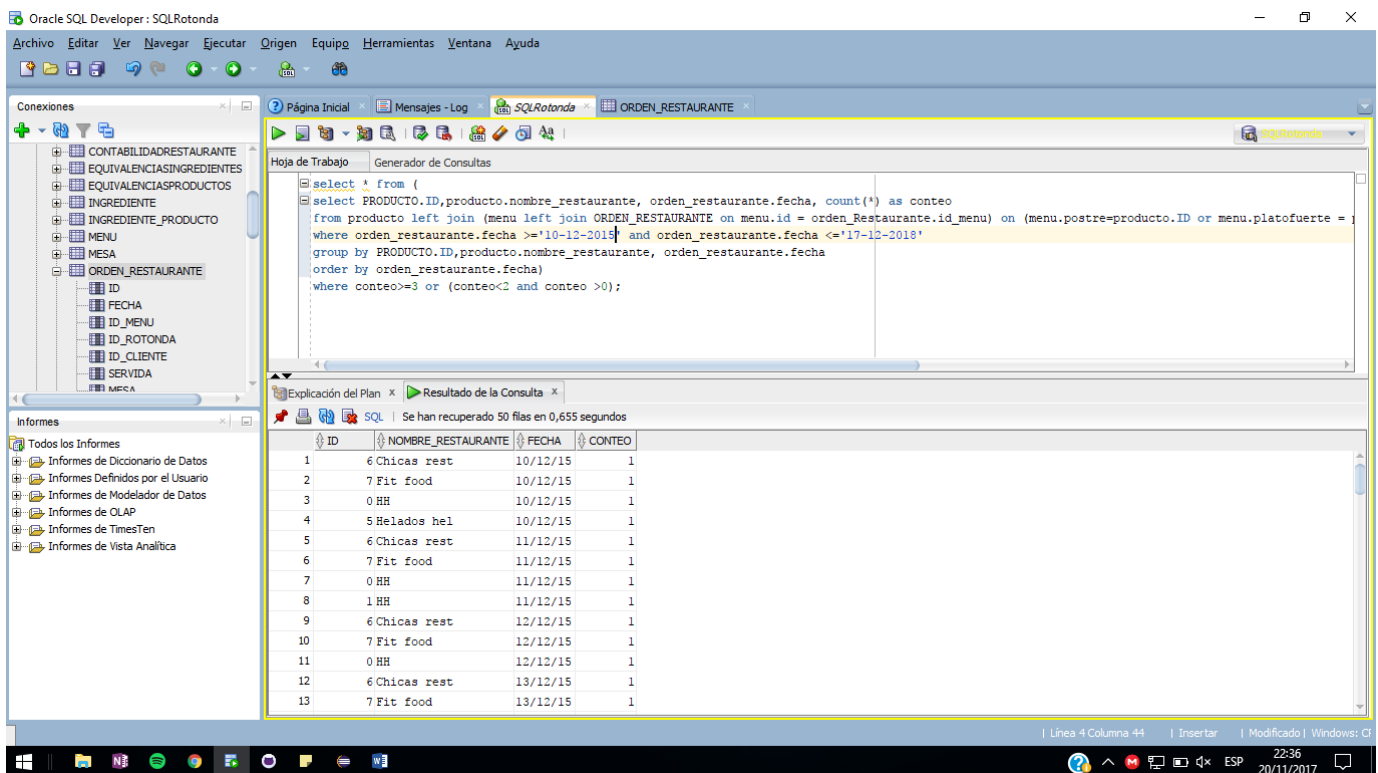


Figura 4: Este rango de fechas evalúa las de varias semanas, 156 Fecha mínima: 1/12/15 Fecha máxima: 17/12/18.

Figuras 1-4: 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.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2928
SORT		ORDER BY	839	2928
HASH		GROUP BY	839	2928
FILTER				
Filter Predicates				
TO_DATE('19-12-18')>=TO_DATE('17-11-17')				
HASH JOIN		RIGHT OUTER	194484	2919
Access Predicates				
ORDEN_RESTAURANTE.ID_MENU=MENU.ID(+)				
VIEW	SYS.NULL		8118	34
HASH JOIN		RIGHT OUTER	8118	34
Access Predicates				
MENU.PLATOFUERTE(+) = PRODUCTO.ID				
TABLE ACCESS	MENU	FULL	6259	15
TABLE ACCESS	PRODUCTO	FULL	6739	19
HASH JOIN			149948	2884
Access Predicates				
CLIENTE.CEDULA=ORDEN_RESTAURANTE.ID_CLIENTE				
NESTED LOOPS			149948	2884
STATISTICS COLLECT				
TABLE ACCESS	ORDEN_RESTAURANTE	FULL	67545	930
Filter Predicates				
AND				
ORDEN_RESTAURANTE.FECHA>='17-11-17'				
ORDEN_RESTAURANTE.FECHA<='19-12-18'				
INDEX	CLIENTE_PK	UNIQUE SCAN		
Access Predicates				
CLIENTE.CEDULA=ORDEN_RESTAURANTE.ID_CLIENTE				
TABLE ACCESS	CLIENTE	BY INDEX ROWID	2	1951
TABLE ACCESS	CLIENTE	FULL	1050188	1951

Other XML

{info}

- info type="db_version"
- 12.1.0.2
- info type="parse_schema"
- "TS1S2304A331720"
- info type="dynamic_sampling" note="y"
- 2
- info type="plan_hash_full"
- 629577938
- info type="plan_hash"
- 4096887915
- info type="plan_hash_2"
- 1344142819
- info type="adaptive_plan" note="y"
- yes

{u}

- 10782582093559245556
- 16710064274869408602
- 8858206635796969530
- 0
- 3

{hint}

- SWAP_JOIN_INPUTS(@"SEL\$8812AA4E" "MENU"@@"SEL\$1")
- USE_HASH(@"SEL\$8812AA4E" "MENU"@@"SEL\$1")
- LEADING(@"SEL\$8812AA4E" "PRODUCTO"@@"SEL\$1" "MENU"@@"SEL\$1")
- FULL(@"SEL\$8812AA4E" "MENU"@@"SEL\$1")
- FULL(@"SEL\$8812AA4E" "PRODUCTO"@@"SEL\$1")
- USE_HASH_AGGREGATION(@"SEL\$E8B0788FC")
- SWAP_JOIN_INPUTS(@"SEL\$E8B0788FC" "from\$_subquery\$005"@@"SEL\$2")
- USE_HASH(@"SEL\$E8B0788FC" "from\$_subquery\$005"@@"SEL\$2")
- USE_HASH(@"SEL\$E8B0788FC" "CLIENTE"@@"SEL\$3")
- LEADING(@"SEL\$E8B0788FC" "ORDEN_RESTAURANTE"@@"SEL\$2" "CLIENTE"@@"SEL\$3" "from\$_subquery\$005"@@"SEL\$2")
- NO_ACCESS(@"SEL\$E8B0788FC" "from\$_subquery\$005"@@"SEL\$2")
- FULL(@"SEL\$E8B0788FC" "CLIENTE"@@"SEL\$3")
- FULL(@"SEL\$E8B0788FC" "ORDEN_RESTAURANTE"@@"SEL\$2")
- OUTLINE(@"SEL\$3")
- ANSI_REARCH(@"SEL\$3")
- OUTLINE(@"SEL\$41465E65")
- OUTLINE(@"SEL\$4")
- OUTLINE(@"SEL\$2")
- ANSI_REARCH(@"SEL\$41465E65")
- OUTLINE(@"SEL\$E15FCA1A")

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
TABLE ACCESS	CLIENTE	FULL	1050188	1951

Other XML

{info}

- info type="db_version"
- 12.1.0.2
- info type="parse_schema"
- "TS1S2304A331720"
- info type="dynamic_sampling" note="y"
- 2
- info type="plan_hash_full"
- 629577938
- info type="plan_hash"
- 4096887915
- info type="plan_hash_2"
- 1344142819
- info type="adaptive_plan" note="y"
- yes

{u}

- 10782582093559245556
- 16710064274869408602
- 8858206635796969530
- 0
- 3

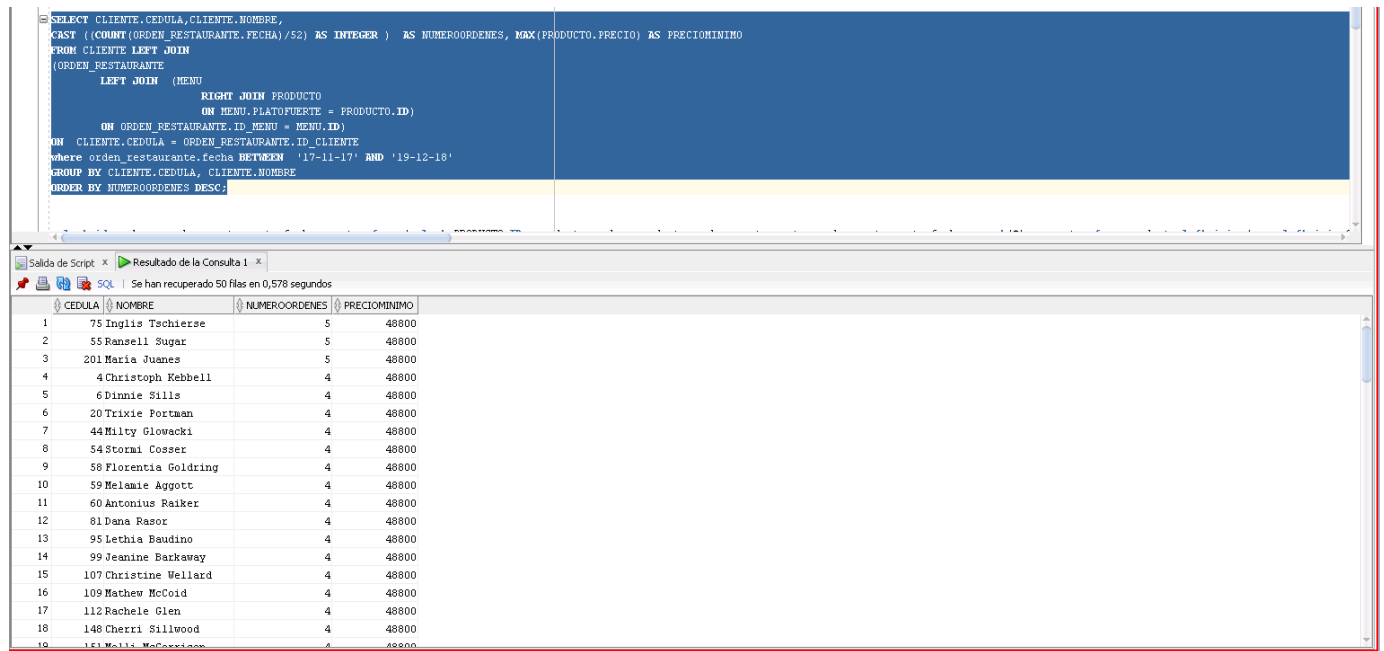
{hint}

- SWAP_JOIN_INPUTS(@"SEL\$8812AA4E" "MENU"@@"SEL\$1")
- USE_HASH(@"SEL\$8812AA4E" "MENU"@@"SEL\$1")
- LEADING(@"SEL\$8812AA4E" "PRODUCTO"@@"SEL\$1" "MENU"@@"SEL\$1")
- FULL(@"SEL\$8812AA4E" "MENU"@@"SEL\$1")
- FULL(@"SEL\$8812AA4E" "PRODUCTO"@@"SEL\$1")
- USE_HASH_AGGREGATION(@"SEL\$E8B0788FC")
- SWAP_JOIN_INPUTS(@"SEL\$E8B0788FC" "from\$_subquery\$005"@@"SEL\$2")
- USE_HASH(@"SEL\$E8B0788FC" "from\$_subquery\$005"@@"SEL\$2")
- USE_HASH(@"SEL\$E8B0788FC" "CLIENTE"@@"SEL\$3")
- LEADING(@"SEL\$E8B0788FC" "ORDEN_RESTAURANTE"@@"SEL\$2" "CLIENTE"@@"SEL\$3" "from\$_subquery\$005"@@"SEL\$2")
- NO_ACCESS(@"SEL\$E8B0788FC" "from\$_subquery\$005"@@"SEL\$2")
- FULL(@"SEL\$E8B0788FC" "CLIENTE"@@"SEL\$3")
- FULL(@"SEL\$E8B0788FC" "ORDEN_RESTAURANTE"@@"SEL\$2")
- OUTLINE(@"SEL\$3")
- ANSI_REARCH(@"SEL\$3")
- OUTLINE(@"SEL\$41465E65")
- OUTLINE(@"SEL\$4")
- OUTLINE(@"SEL\$2")
- ANSI_REARCH(@"SEL\$41465E65")
- OUTLINE(@"SEL\$E15FCA1A")

Imagen 19: plan de ejecución para e RFC12. Se refleja el uso de los índices.

Utiliza los índices creados para él.

Tiempos de ejecución

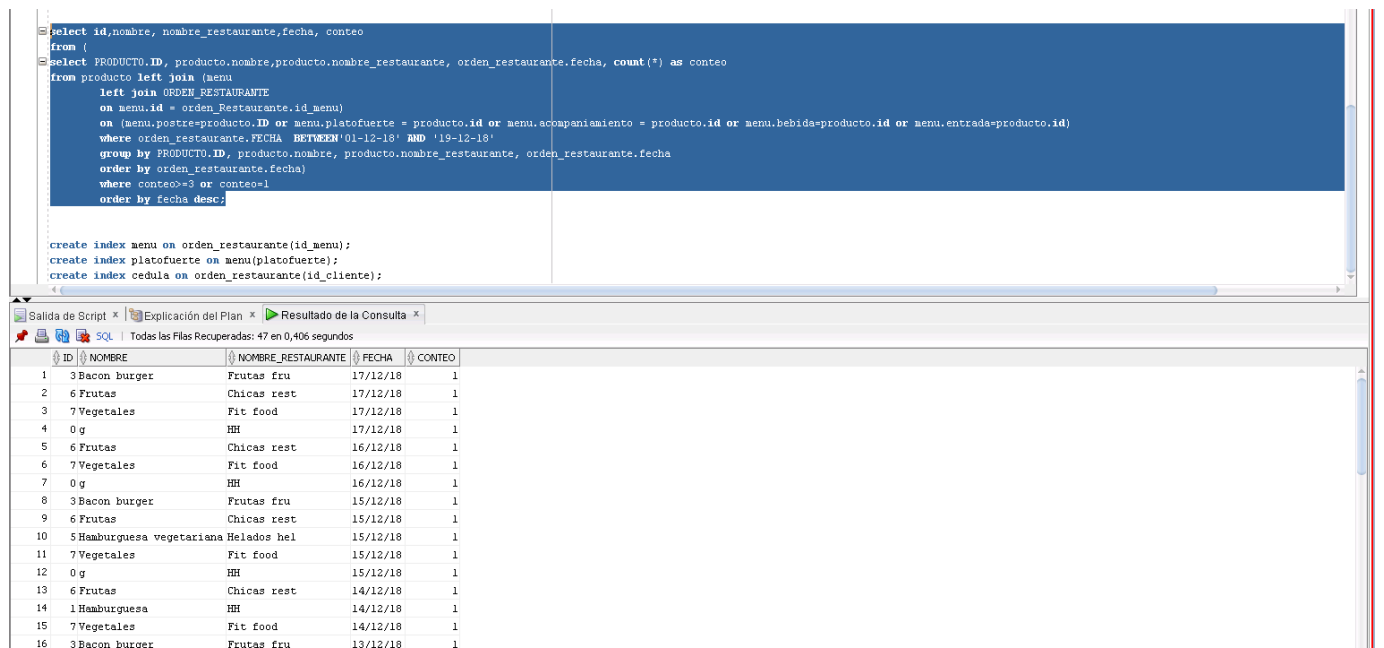


The screenshot shows a SQL query window with a query that calculates the average number of orders and the maximum price per item for each client, filtered by a date range from December 11, 2017, to December 19, 2017. The results are displayed in a table with 50 rows.

```
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 '17-11-17' AND '19-12-18'
GROUP BY CLIENTE.CEDULA, CLIENTE.NOMBRE
ORDER BY NUMEROORDENES DESC;
```

CEDULA	NOMBRE	NUMEROORDENES	PRECIOMINIMO
75	Inglis Tschierse	5	48800
55	Ransell Sugar	5	48800
201	Maria Juanes	5	48800
4	Christoph Kebbell	4	48800
6	Dimmie Sills	4	48800
20	Trixie Portman	4	48800
44	Milty Glowacki	4	48800
54	Stormi Coeser	4	48800
58	Florentia Goldring	4	48800
59	Melanie Aggott	4	48800
60	Antonius Raiker	4	48800
81	Dana Rasor	4	48800
95	Lethia Baudino	4	48800
99	Jeanine Barkawey	4	48800
107	Christine Wellard	4	48800
109	Mathew McCoid	4	48800
112	Rachele Glen	4	48800
148	Cherri Sillwood	4	48800
151	Malika McPeeck	4	48800

Figura 5: Este rango de fechas evalúa las de varias semanas, 3 Fecha mínima: 06/12/18 Fecha máxima: 19/12/18



The screenshot shows a SQL query window with a query that identifies products with a count of 3 or more orders within a date range from December 11, 2017, to December 19, 2017. The results are displayed in a table with 16 rows. Below the query, there are three index creation statements.

```
select id,nombre, nombre_restaurante,fecha, conteo
from (
select PRODUCTO.ID, producto.nombre, 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.acompanamiento = producto.id or menu.bebida=producto.id or menu.entrada=producto.id)
where orden_restaurante.FECHA BETWEEN '01-12-18' AND '19-12-18'
group by PRODUCTO.ID, producto.nombre, producto.nombre_restaurante, orden_restaurante.fecha
order by orden_restaurante.fecha)
where conteo>=3 or conteo=1
order by fecha desc;
```

```
create index menu on orden_restaurante(id_menu);
create index platofuerte on menu(platofuerte);
create index cedula on orden_restaurante(id_cliente);
```

ID	NOMBRE	NOMBRE_RESTAURANTE	FECHA	CONTEO
3	Bacon burger	Frutas fru	17/12/18	1
6	Frutas	Chicas rest	17/12/18	1
7	Vegetales	Fit food	17/12/18	1
0 g		HH	17/12/18	1
6	Frutas	Chicas rest	16/12/18	1
7	Vegetales	Fit food	16/12/18	1
0 g		HH	16/12/18	1
3	Bacon burger	Frutas fru	15/12/18	1
6	Frutas	Chicas rest	15/12/18	1
5	Hamburguesa vegetariana	Helados hel	15/12/18	1
7	Vegetales	Fit food	15/12/18	1
0 g		HH	15/12/18	1
6	Frutas	Chicas rest	14/12/18	1
1	Hamburguesa	HH	14/12/18	1
7	Vegetales	Fit food	14/12/18	1
3	Bacon burger	Frutas fru	13/12/18	1

Figura 6: Este rango de fechas evalúa las de varias semanas, 56 Fecha mínima: 12/11/17 Fecha máxima: 19/12/18

<pre> 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.PLATOFUENTE = PRODUCTO.ID) ON ORDEN.RESTAURANTE.ID MENU = MENU.ID) ON CLIENTE.CEDULA = ORDEN.RESTAURANTE.ID CLIENTE where orden.restaurante.fecha BETWEEN '17-11-18' AND '19-12-18' GROUP BY CLIENTE.CEDULA, CLIENTE.NOMBRE ORDER BY NUMEROORDENES DESC; </pre>				
Resultado de la Consulta				
Se han recuperado 50 filas en 0,172 segundos				
CEDULA	NOMBRE	NUMEROORDENES	PRECIOMINIMO	
1	59 Melamie Aggott	0	48800	
2	95 Iethia Baudino	0	48800	
3	222 Harrietta Deshon	0	48800	
4	348 Roxanna McFeat	0	48800	
5	194 Cassy Sommers	0	48800	
6	44 Milty Glowacki	0	48800	
7	244 Charmion Negri	0	48800	
8	81 Dana Rasor	0	48800	
9	220 Tyrus Soreau	0	48800	
10	151 Melli McCarrison	0	48800	
11	284 Linn Vanner	0	48800	
12	208 Wilhelm Pettican	0	48800	
13	272 Christophorus Nesbit	0	48800	
14	287 Eduard Marrian	0	48800	
15	54 Stormi Cosser	0	48800	
16	188 Nita Kristof	0	48800	
17	258 Conchita Etherington	0	48800	
18	6 Dinnie Sillis	0	48800	
19	170 Kit Gerald	0	330	
20	250 Lizzie Gilbeart	0	3303	
21	109 Mathew McCoid	0	330	
22	224 Inger Feckey	0	330	
23	60 Antonius Raiker	0	330	
24	219 Donny Turbat	0	33028	
25	4 Christoph Kebbell	0	33028	
26	251 Aurelia Skym	0	33028	

Figura 7: Este rango de fechas evalúa las de varias semanas, 54 Fecha **mínima:** 17/11/18 **Fecha máxima:** 19/12/19

Figuras 5-7: 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

Planes de consulta desarrollados por el desarrollador.

OPERACIÓN					OBJETO	OPCIONES	COMENTARIOS
SELECT							
	SORT						
		HASH					
			FILTER				
				RESTAURANTE			
				DATE 1			
				DATE 2		GROUP	
			HASH JOIN				Ciclos anidados
				HASHJOIN	MENU	FULL	indes menu.nombrerestaurante
				TABLE ACCESS	PRODUCTO	FULL	
				TABLE ACCESS			
				HASHJOIN	ORDEN_RESTAURANTE	UNIQUE SCAN	INDEX orden.cliente
				TABLE ACCESS	CLIENTE	FULL	
				TABLE ACCESS			

Tabla 1: descripción del plan para el RFC9. 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.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	
SELECT STATEMENT					2404
SORT		ORDER BY		379	2404
HASH		UNIQUE		379	2403
HASH JOIN		OUTER		1017	2402
Access Predicates					
CLIENTE.CEDULA(+) = ORDEN_RESTAURANTE.ID_CLIENTE					
HASH JOIN				1017	448
Access Predicates					
MENU.ID = ORDEN_RESTAURANTE.ID_MENU					
NESTED LOOPS				1017	448
NESTED LOOPS				2052	448
STATISTICS COLLECTOR					
HASH JOIN		SEMI		4	34
Access Predicates					
MENU.NOMBRE_RESTAURANTE = PRODUCTO.NOMBRE_RESTAURANTE					
TABLE ACCESS MENU		FULL		4	15
Filter Predicates					
MENU.NOMBRE_RESTAURANTE = 'Chicas rest'					
TABLE ACCESS PRODUCTO		FULL		16	19
Filter Predicates					
PRODUCTO.NOMBRE_RESTAURANTE = 'Chicas rest'					
INDEX MENU		RANGE SCAN		513	2
Access Predicates					
MENU.ID = ORDEN_RESTAURANTE.ID_MENU					
TABLE ACCESS ORDEN_RESTAURANTE		BY INDEX ROWID		254	408
Filter Predicates					
AND					
ORDEN_RESTAURANTE.FECHA >= TO_DATE('2010-12-11 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.FECHA < TO_DATE('2018-12-17 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.ID_CLIENTE IS NOT NULL					
TABLE ACCESS ORDEN_RESTAURANTE		FULL		254	408
Filter Predicates					
AND					
ORDEN_RESTAURANTE.FECHA >= TO_DATE('2010-12-11 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.FECHA < TO_DATE('2018-12-17 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
ORDEN_RESTAURANTE.ID_CLIENTE IS NOT NULL					
TABLE ACCESS CLIENTE		FULL		1050188	1951

Other XML

```

{info}
  info type="db_version"
    12.1.0.2
  info type="parse_schema"
    "ISI52304A331720"

```



Imagen 24: plan de trabajo para el RFC9.

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 nombre restaurante 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 planes, 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.

RFC10

Planes de consulta desarrollados por el desarrollador.

OPERACIÓN					OBJETO	OPCIONES	COMENTARIOS
SELECT							
MINUS	SORT						
		HASH					
			FILTER				
				RESTAURANTE			
				DATE 1			
				DATE 2		GROUP	
			HASH JOIN				Ciclos anidados
				HASHJOIN	MENU	FULL	indes menu.nombrerestaurante
				TABLE ACCESS	PRODUCTO	FULL	
				TABLE ACCESS			
				HASHJOIN	ORDEN_RESTAURANTE	UNIQUE SCAN	INDEX orden.cliente
				TABLE ACCESS	CLIENTE	FULL	
				TABLE ACCESS			

Tabla 2: descripción del plan para el RFC10. 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, documento con una foto de pantalla los planes de consulta obtenidos en SQLDeveloper.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1050188 33577
MINUS				
SORT				
TABLE ACCESS	CLIENTE	UNIQUE		1050188 13481
SORT				
TABLE ACCESS		FULL	1050188	1951
NESTED LOOPS		UNIQUE	1	41
HASH JOIN		OUTER	1	40
Access Predicates		SEMI	1	38
MENU.NOMBRE_RESTAURANTE=PRODUCTO.NOMBRE_RESTAURANTE				
HASH JOIN			1	19
Access Predicates				
MENU.ID=ORDEN_RESTAURANTE.ID_MENU				
NESTED LOOPS			1	19
NESTED LOOPS			4	19
STATISTICS COLLECT				
TABLE ACCESS	ORDEN_RESTAURANTE	BY INDEX ROWID BATCHED	4	15
INDEX	FECHA	RANGE SCAN	12	3
Access Predicates				
AND				
ORDEN_RESTAURANTE.FECHA>=TO_DATE(' 2018-12-11 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
ORDEN_RESTAURANTE.FECHA<TO_DATE(' 2018-12-17 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
INDEX	MENU_PK	UNIQUE SCAN	1	0
Access Predicates				
MENU.ID=ORDEN_RESTAURANTE.ID_MENU				
TABLE ACCESS	MENU	BY INDEX ROWID	1	1
Filter Predicates				
MENU.NOMBRE_RESTAURANTE='Boquitas'				
TABLE ACCESS	MENU	FULL	1	1
Filter Predicates				
MENU.NOMBRE_RESTAURANTE='Boquitas'				
TABLE ACCESS	PRODUCTO	FULL	21	19
Filter Predicates				
PRODUCTO.NOMBRE_RESTAURANTE='Boquitas'				
TABLE ACCESS	CLIENTE	BY INDEX ROWID	1	2
INDEX	CLIENTE_PK	UNIQUE SCAN	1	1
Access Predicates				
CLIENTE.CEDULA(+) = ORDEN_RESTAURANTE.ID_CLIENTE				
Other XML				
{info}				
info type="db_version"				
12.1.0.2				
info type="parse_schema"				
"ISIS2304A331720"				
info type="dynamic_sampling" note="y"				
2				
info type="plan_hash_full"				
1391212209				
info type="plan_hash"				
861512523				
info type="plan_hash_2"				
2707857907				
info type="adaptive_plan" note="y"				
yes				
{v}				
11772725536356023955				
2394384430148205229				
10250827045396034032				
4333359815089530554				
8858206635798969530				
10782582093559245556				
16710064274869408602				
5				
2				
{hint}				
FULL(@SEL\$1 "CLIENTE"@"SEL\$4")				
PARTIAL_JOIN(@SEL\$FB96BC69 "PRODUCTO"@"SEL\$1")				
USE_NL(@SEL\$FB96BC69 "CLIENTE"@"SEL\$3")				
USE_HASH(@SEL\$FB96BC69 "PRODUCTO"@"SEL\$1")				
NLJ_BATCHING(@SEL\$FB96BC69 "MENU"@"SEL\$1")				
USE_NL(@SEL\$FB96BC69 "MENU"@"SEL\$1")				
LEADING(@SEL\$FB96BC69 "ORDEN_RESTAURANTE"@"SEL\$2" "MENU"@"SEL\$1" "PRODUCTO"@"SEL\$1" "CLIENTE"@"SEL\$3")				
INDEX_RS_ASC(@SEL\$FB96BC69 "CLIENTE"@"SEL\$3" ("CLIENTE"."CEDULA"))				
FULL(@SEL\$FB96BC69 "PRODUCTO"@"SEL\$1")				
INDEX(@SEL\$FB96BC69 "MENU"@"SEL\$1" ("MENU"."ID"))				
BATCH_TABLE_ACCESS_BY_ROWID(@SEL\$FB96BC69 "ORDEN_RESTAURANTE"@"SEL\$2")				
INDEX_RS_ASC(@SEL\$FB96BC69 "ORDEN_RESTAURANTE"@"SEL\$2" ("ORDEN_RESTAURANTE"."FECHA"))				
OUTLINE(@SEL\$2)				
OUTLINE(@SEL\$1)				
ANSI_REARCH(@SEL\$2)				
CUT TRUE (@"SEL\$4"@"SEL\$1")				

Imagen 25: plan de trabajo para el RFC10.

Utiliza los índices creados para él.

Comparación planes.

Podemos observar que este plan de ejecución es muy similar al del RFC9, con la diferencia de que para este tenemos una operación MINUS, donde a el total de clientes de la base de datos, le restamos los resultantes del query del RFC9.

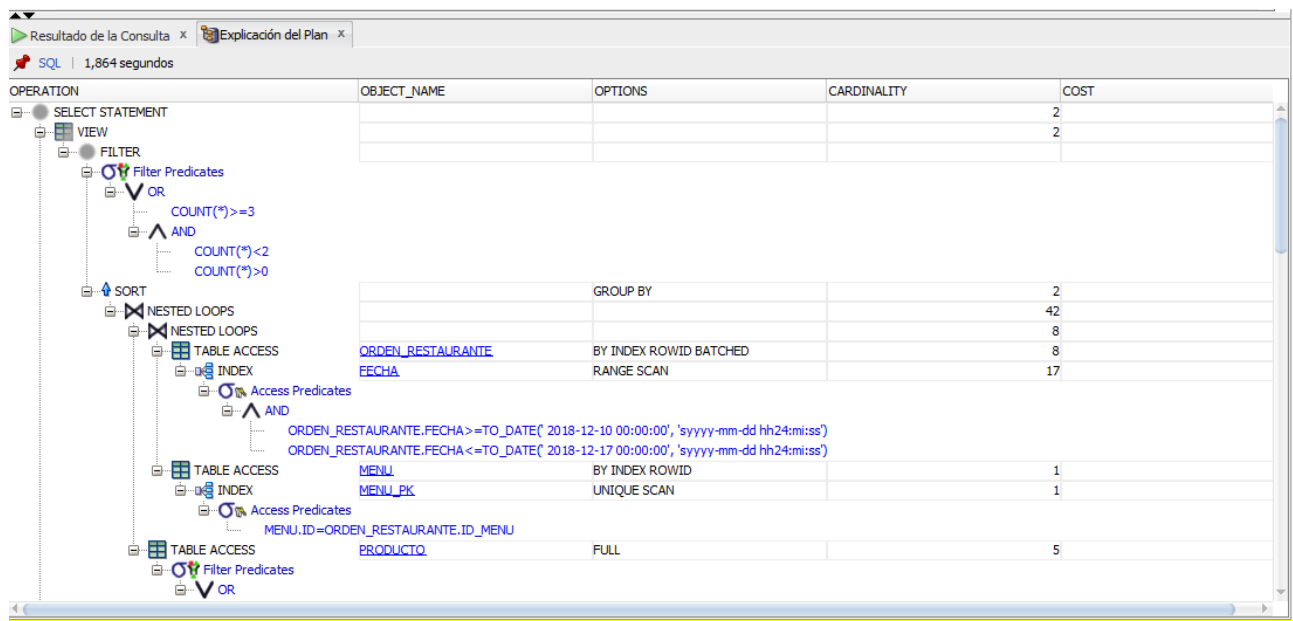
RFC11

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 3: 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.



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
VIEW				2
FILTER				
Filter Predicates				
OR				
COUNT(*)>=3				
AND				
COUNT(*)<2				
COUNT(*)>0				
SORT				
NESTED LOOPS		GROUP BY		2
NESTED LOOPS				42
TABLE ACCESS	ORDEN_RESTAURANTE	BY INDEX ROWID BATCHED		8
INDEX	FECHA	RANGE SCAN		8
Access Predicates				
AND				
ORDEN_RESTAURANTE.FECHA>=TO_DATE(' 2018-12-10 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
ORDEN_RESTAURANTE.FECHA<=TO_DATE(' 2018-12-17 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
TABLE ACCESS	MENU	BY INDEX ROWID		1
INDEX	MENU_PK	UNIQUE SCAN		1
Access Predicates				
MENU.ID=ORDEN_RESTAURANTE.ID_MENU				
TABLE ACCESS	PRODUCTO	FULL		5
Filter Predicates				
OR				

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
TABLE ACCESS	PRODUCTO	FULL		5
Filter Predicates				
OR				
MENU.POSTRE=PRODUCTO.ID				
MENU.PLATOFUERTE=PRODUCTO.ID				
MENU.ACOMPAÑAMIENTO=PRODUCTO.ID				
MENU.BEBIDA=PRODUCTO.ID				
MENU.ENTRADA=PRODUCTO.ID				
Other XML				
{info}				
info type="db_version"				
12.1.0.2				
info type="parse_schema"				
"ISIS2304A331720"				
info type="dynamic_sampling" note="y"				
2				
info type="plan_hash_full"				
2640410797				
info type="plan_hash"				
36700515				
info type="plan_hash_2"				
2640410797				
{u}				
17635602895091721057				
10782582093559245556				

Imagen 26: 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 planes, 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.

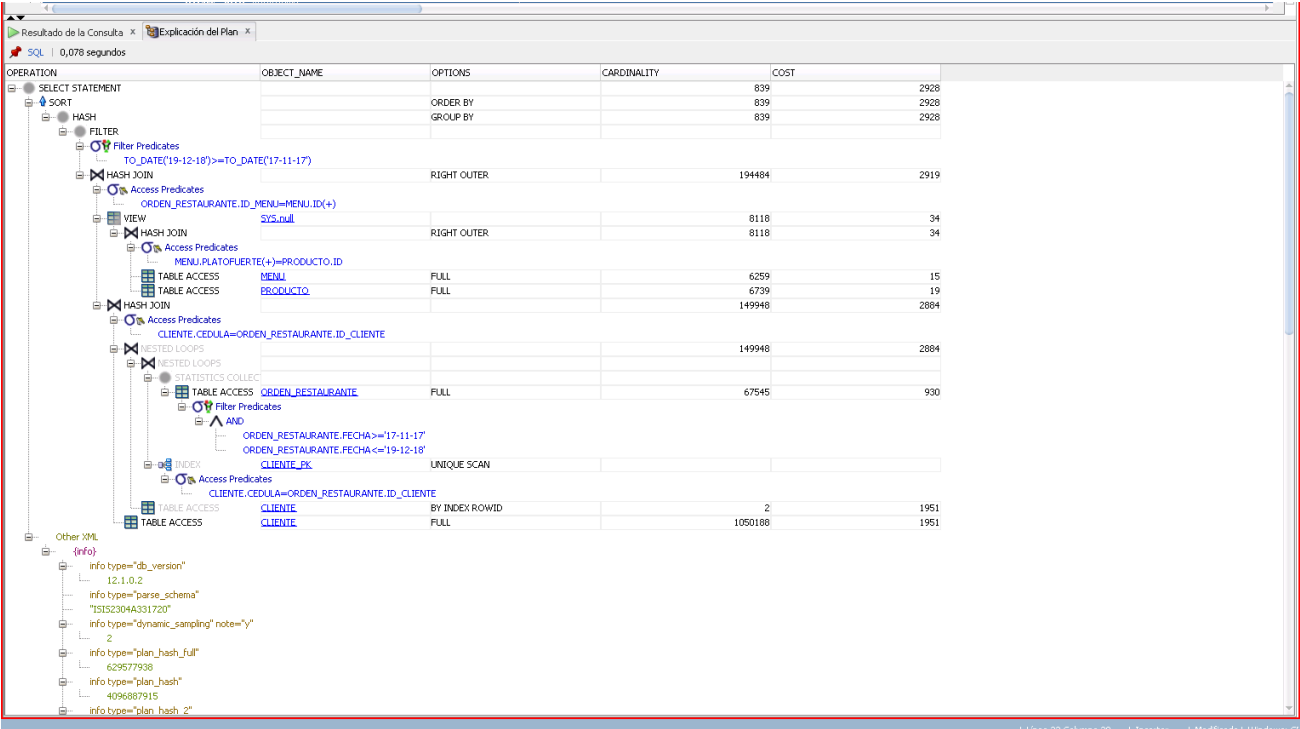
RFC12

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 4: 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.



OPERATION	TABLE ACCESS	OBJECT_NAME	OPTIONS	CARDINALITY	COST	
		CLIENTE	FULL		1050188	1951

Other XML	{info}	info type="db_version"	12.1.0.2
		info type="parse_schema"	"TS1S2304A331720"
		info type="dynamic_sampling" note="y"	2
		info type="plan_hash_full"	629577930
		info type="plan_hash"	4096887915
		info type="plan_hash_2"	1344142819
		info type="adaptive_plan" note="y"	yes
	{u}		10782582093559245556
			16710064274869408602
			885826635796969530
			0
			3
	{hint}		
		SWAP_JOIN_INPUTS(@"SEL\$8812AA4E" "MENU"@"SEL\$1")	
		USE_HASH(@"SEL\$8812AA4E" "MENU"@"SEL\$1")	
		LEADING(@"SEL\$8812AA4E" "PRODUCTO"@"SEL\$1" "MENU"@"SEL\$1")	
		FULL(@"SEL\$8812AA4E" "MENU"@"SEL\$1")	
		FULL(@"SEL\$8812AA4E" "PRODUCTO"@"SEL\$1")	
		USE_HASH_AGGREGATION(@"SEL\$E80788FC")	
		SWAP_JOIN_INPUTS(@"SEL\$E80788FC" "from\$_subquery\$_005"@"SEL\$2")	
		USE_HASH(@"SEL\$E80788FC" "from\$_subquery\$_005"@"SEL\$2")	
		USE_HASH(@"SEL\$E80788FC" "CLIENTE"@"SEL\$3")	
		LEADING(@"SEL\$E80788FC" "ORDEN_RESTAURANTE"@"SEL\$2" "CLIENTE"@"SEL\$3" "from\$_subquery\$_005"@"SEL\$2")	
		NO_ACCESS(@"SEL\$E80788FC" "from\$_subquery\$_005"@"SEL\$2")	
		FULL(@"SEL\$E80788FC" "CLIENTE"@"SEL\$3")	
		FULL(@"SEL\$E80788FC" "ORDEN_RESTAURANTE"@"SEL\$2")	
		OUTLINE(@"SEL\$3")	
		ANSI_REARCH(@"SEL\$3")	
		OUTLINE(@"SEL\$41465E65")	
		OUTLINE(@"SEL\$4")	
		OUTLINE(@"SEL\$2")	
		ANSI_REARCH(@"SEL\$41465E65")	
		OUTLINE(@"SEL\$E15FCA1A")	

Imagen 27: 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 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.

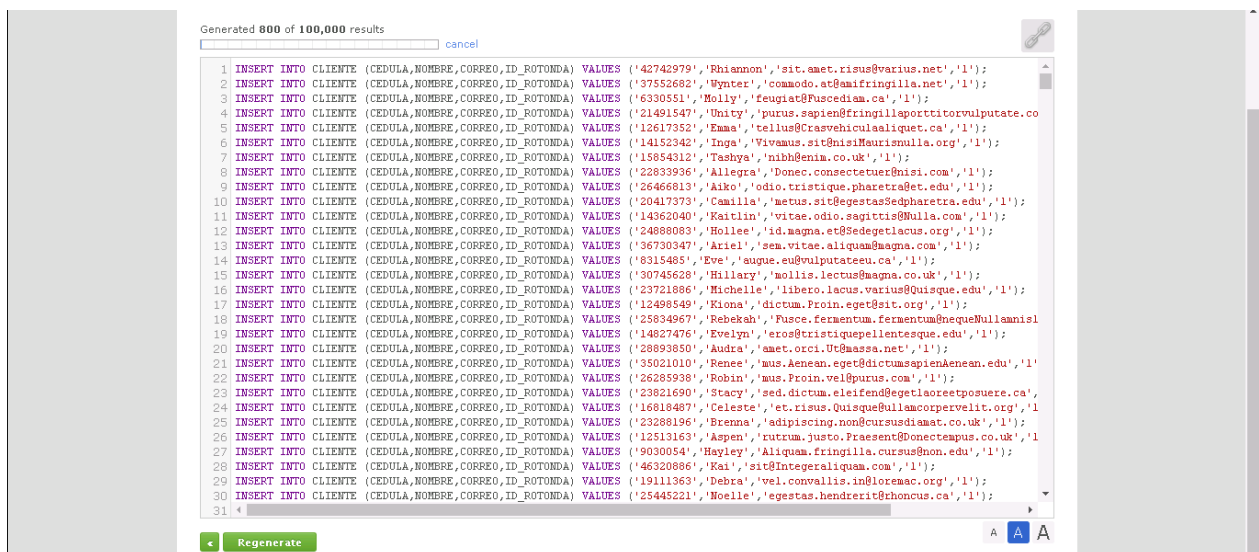


Imagen 28: Generación de sentencias para la Inserción de clientes mediante sentencias por un generador de datos.

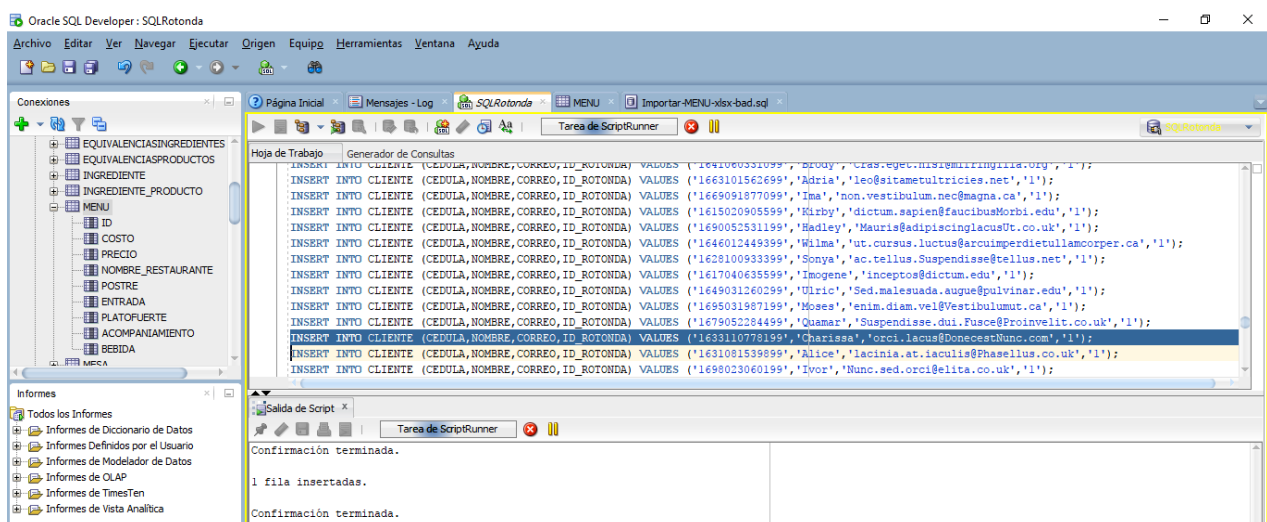


Imagen 29: Inserción de clientes mediante sentencias ya realizadas en un generador de datos.

Oracle SQL Developer : Tabla ISIS2304A331720.CLIENTE@ST

Archivo Editar Ver Navegar Ejecutar Equipo Herramientas Ventana Ayuda

172.24.41.62

Conexiones

Página Inicial ST ORDEN_RESTAURANTE CLIENTE MENU

Columnas: Datos Model Restricciones Permisos Estadísticas Disparadores Flashback Dependencias Detalles Particiones Índices SQL

Ordenar... Filtrar:

	CED...	NOMBRE	CORREO	ID_ROTONDA
1	1	Lazare Bedminster	lbedminster5@yahoo.co.jp	1
2	2	Nena McEniry	maceniry0@deill.com	1
3	3	Saba Musgrove	smusgrovel@soundcloud.com	1
4	4	Christoph Kebbell	ckebbell2@yale.edu	1
5	5	Virgilio Van De...	vvan3@microsoft.com	1
6	6	Dinnie Sills	dsills4@nature.com	1
7	7	Brockie Blakden	bblakden5@barnesandnoble.com	1
8	8	Ashien Shearn	ashearn6@ciaco.com	1
9	9	Paige Balston	pbalston7@slate.com	1
10	10	Ianita Danielis	ldanielis8@linkedin.com	1
11	11	Coralyn Rivallant	crivallant9@ebay.co.uk	1
12	12	Giacomo Dreger	gdregera@jugem.jp	1
13	13	Abbey Hatliffe	ahatliffeb@is.gd	1
14	14	Godart Akred	gakred@amazon.co.jp	1
15	15	Gail Hamber	ghamberd@washington.edu	1
16	16	Felita Petroff	fpetroffe@army.mil	1
17	17	Stevie Bath	sbathf@ecologic-nifty.com	1
18	18	Leora Shotton	lshottong@wsj.com	1
19	19	Charyl Hylton	chyltonh@epiegel.de	1
20	20	Trixie Portman	tportmani@shinystat.com	1
21	21	Noelani Climar	nclimarj@nydailynews.com	1
22	22	Lane Spalding	lspaldingk@theguardian.com	1
23	23	Vitoria Dellmann	vdallmanni@loomborg.com	1
24	24	Demeter Balf	dbalfm@intel.com	1
25	25	Ramon Feldman	rfeldmann@fastcompany.com	1

Informes

Todos los Informes

- Informes de Diccionario de Datos
- Informes Definidos por el Usuario
- Informes de Modelador de Datos
- Informes de OLAP
- Informes de TimesTen
- Informes de Vista Analítica

Imagen 30: Clientes una vez insertados.

MENU - Excel

Christian Gustavo Chavarro Espejo

Archivo Inicio Insertar Diseño de página Fórmulas Datos Revisar Vista ¿Qué desea hacer? Compartir

Calibri 11 Fuente Ajustar texto General

Formato condicional Dar formato Estilos de celda Insertar Eliminar Formato Celdas Edición

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	ID	COSTO	PRECIO	NOMBRE_RE	POSTRE	ENTRADA	PLATOFUERT	ACOMPANIA	BEBIDA							
2	60521211	79790	41058	Hulshout	10008114	10000801	10002364	10003631	10009656							
3	19246455	60675	9428	Innisfail	10007217	10001798	10004833	10005810	10000207							
4	55483542	36519	36277	Kanpur Cant	10007685	10006372	10004436	10003253	10002867							
5	40601057	20122	54921	Maldstone	10001269	10004100	10005514	10002975	10007121							
6	19393230	31151	65762	New Plymou	10004769	10008576	10010215	10001805	10006627							
7	36434514	71589	37438	Nieuwegein	10001794	10004953	10000476	10004157	10004405							
8	42462294	9020	67510	North Las Ve	10003553	10001015	10004674	10001622	10006811							
9	63945384	3328	23359	North Las Ve	10007597	10010117	10001557	10001402	10008893							
10	21913485	50766	11263	San Lorenzo	10009641	10007599	10007049	10007951	10001160							
11	895767	32631	62950	San Lorenzo	10006500	10005974	10000933	10009115	10008683							
12	18584216	55795	51192	Scena/Scher	10003892	10009312	10001573	10009431	10000107							
13	28847213	77437	29467	Swan	10000188	10004013	10002207	10004067	10007249							
14	44251206	38760	30031	Tain	10003031	10002255	10002382	10002032	10009664							
15	32152304	27313	43536	Talcahuano	10003210	10007279	10009622	10000378	10002126							
16	55514507	74421	39015	Vitacura	10000920	10008429	10001622	10002304	10005731							

Imagen 31: Ejemplos de menús que se insertan mediante Excel.

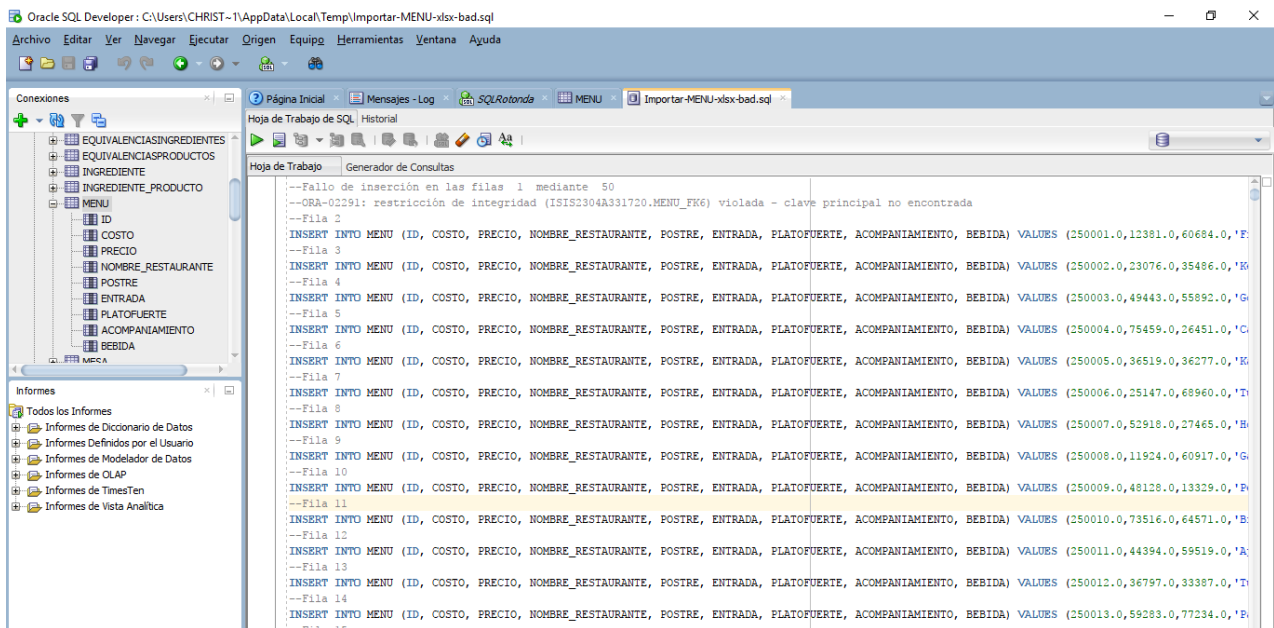


Imagen 32: Inserción de los menús mediante el Excel.

Imagen 33: productos en un archivo de Excel.

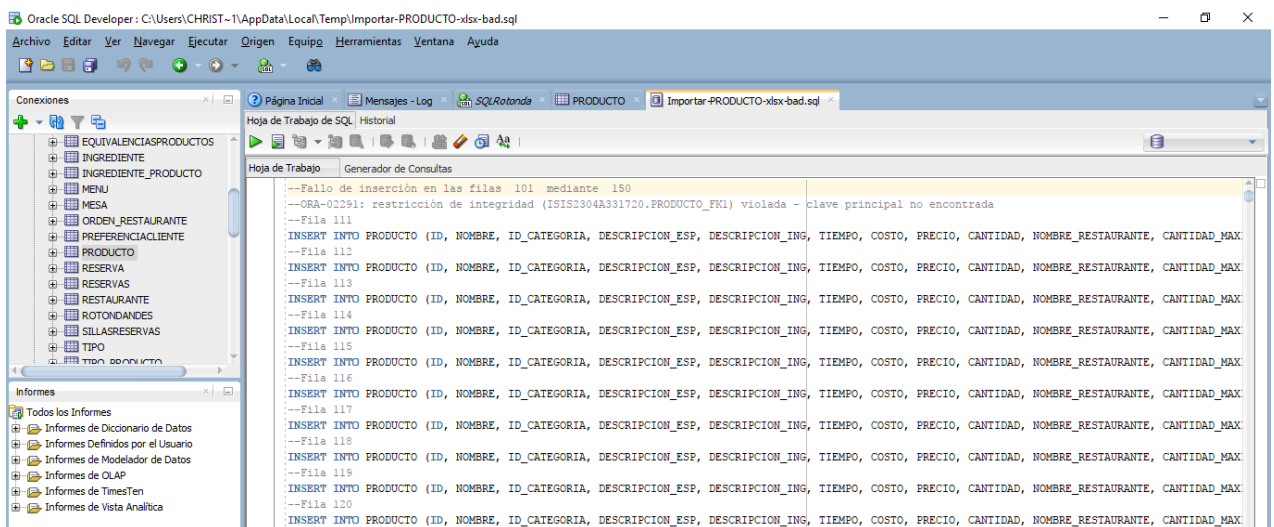


Imagen 34: Inserción de productos mediante el archivo de Excel.

Oracle SQL Developer: Tabla IS12304A331720.PRODUCTO@SQLRotonda

Archivo Editar Ver Navegar Ejecutar Equipo Herramientas Ventana Ayuda

Conexiones

Página Inicial Mensajes - Log SQLRotonda PRODUCTO Importar-PRODUCTO-xlsx-bad.sql

Columnas Datos Model Restricciones Permisos Estadísticas Disparadores Flashback Dependencias Detalles Particiones Índices SQL

Ordenar... Filtrar:

ID	NOMBRE	ID_CATEGORIA	DESCRIPCION_ESP	DESCRIPCION_ING	TIEMPO	COSTO	PRECIO	CANTIDAD	NOMBRE_...	CANTIDAD_MAXIMA
1425	... Producto...	1 d	i		60	68649	17588	28	Carleton	100
1426	... Producto...	5 d	i		22	67941	45558	61	Aurora	100
1427	... Producto...	2 d	i		12	1810	68009	29	North L...	100
1428	... Producto...	2 d	i		11	8810	52452	84	Fraser ...	100
1429	... Producto...	2 d	i		1	61690	66081	53	Kendal	100
1430	... Producto...	2 d	i		27	77731	30459	100	Gosnells	100
1431	... Producto...	5 d	i		51	34041	2309	28	Cape Breton Island	100
1432	... Producto...	5 d	i		40	56899	28354	12	Kanpur ...	100
1433	... Producto...	5 d	i		39	6383	26578	98	Tula	100
1434	... Producto...	3 d	i		38	19459	76424	49	Bohen N...	100
1435	... Producto...	1 d	i		55	7107	3006	54	Gaither...	100
1436	... Producto...	2 d	i		18	8361	71997	32	Perugia	100
1437	... Producto...	3 d	i		45	23435	60226	20	Bromyard	100
1438	... Producto...	1 d	i		50	28246	2530	53	Ajax	100

Informes

Todos los Informes

Imagen 35: Productos después de ser insertados.

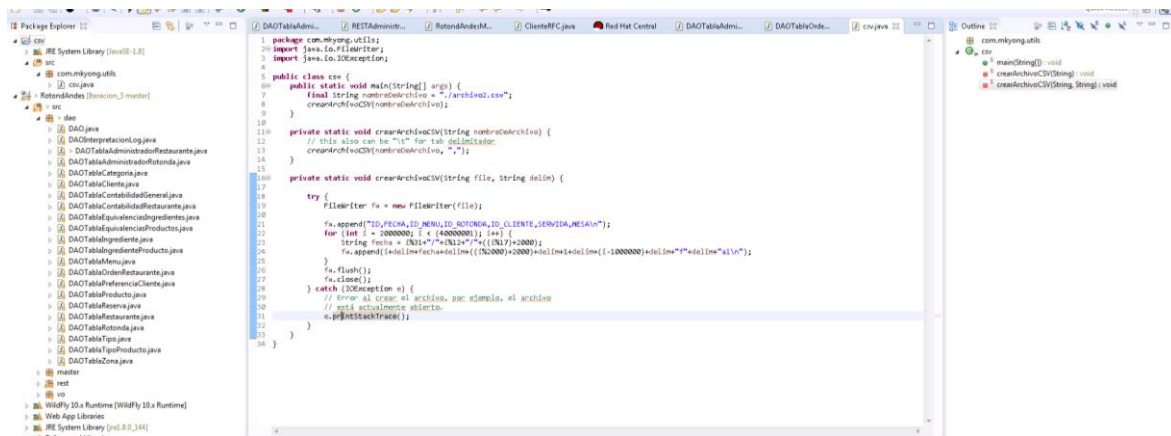


Imagen 36: descripción del programa en java para generar los datos.

[illegible]

Imagen 37: respuesta del programa

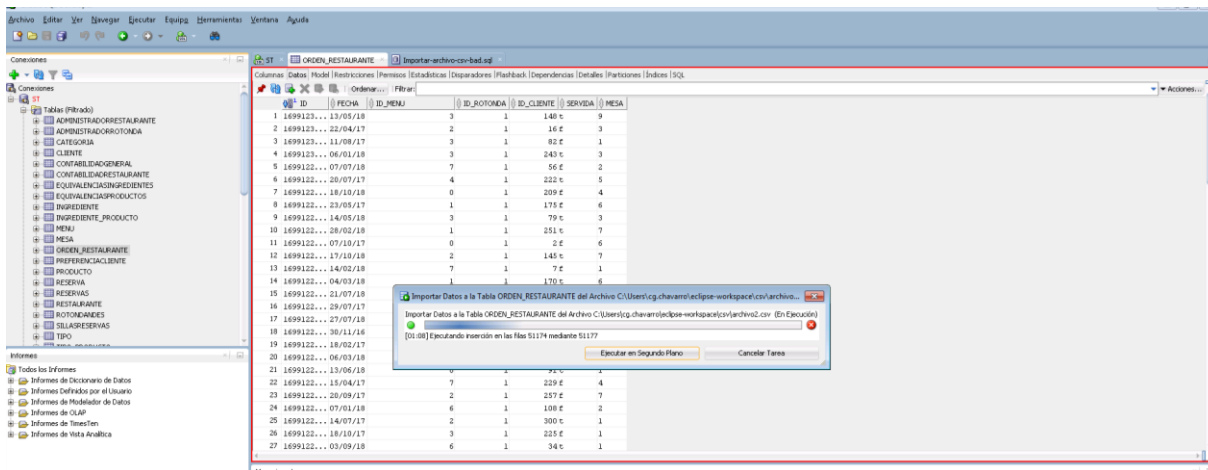


Imagen 38: inserción de los datos del csv.

The screenshot shows an Excel spreadsheet titled 'ORDEN_RESTAURANTE - Excel'. The spreadsheet contains a list of orders with columns A through P. The data is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	ID	FECHA	MENU	ID_ROTONDA	ID_CLIENTE	SERVIDA	MESA									
2	9871101420	02/10/2017	4,04067E+18	1	1090509418	t										
3	9871101421	02/10/2017		2	1	1090509418	t	a1								
4	9871101422	02/10/2017	4,15415E+18	1	1090509418	t		a1								
5	9871101423	02/10/2017	3,77469E+16	1	1090509418	t		a1								
6	9871101424	02/10/2017	3,53394E+18	1	1090509418	t		a1								
7	9871101425	02/10/2017		6	1	2020202020	t	a1								
8	9871101426	02/10/2017		6	1	2020202020	t	a1								
9	9871101427	02/10/2017	4,59146E+17	1	1090509418	t										
10	9871101428	02/10/2017		6	1	2020202020	t	a1								
11	9871101429	02/10/2017		6	1	2020202020	t	a1								
12	9871101430	02/10/2017		6	1	2020202020	t	a1								
13	9871101431	02/10/2017		6	1	2020202020	t	a1								
14	9871101432	02/10/2017	7,51848E+18	1	1090509418	t										
15	9871101433	02/10/2017	5,91997E+18	1	1090509418	f										
16	9871101434	02/10/2017		2	1	2020202020	t									

Imagen 39: Ordenes a ser agregadas en la base de datos.

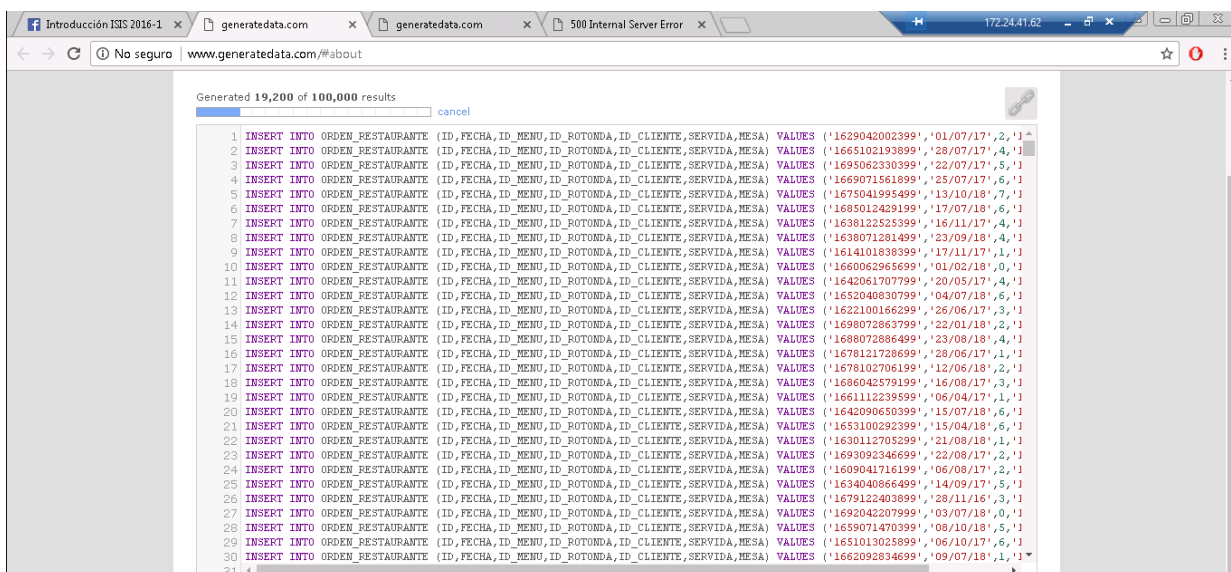


Imagen 40: Ordenes a ser agregadas en la base de datos mediante sentencias generadas.

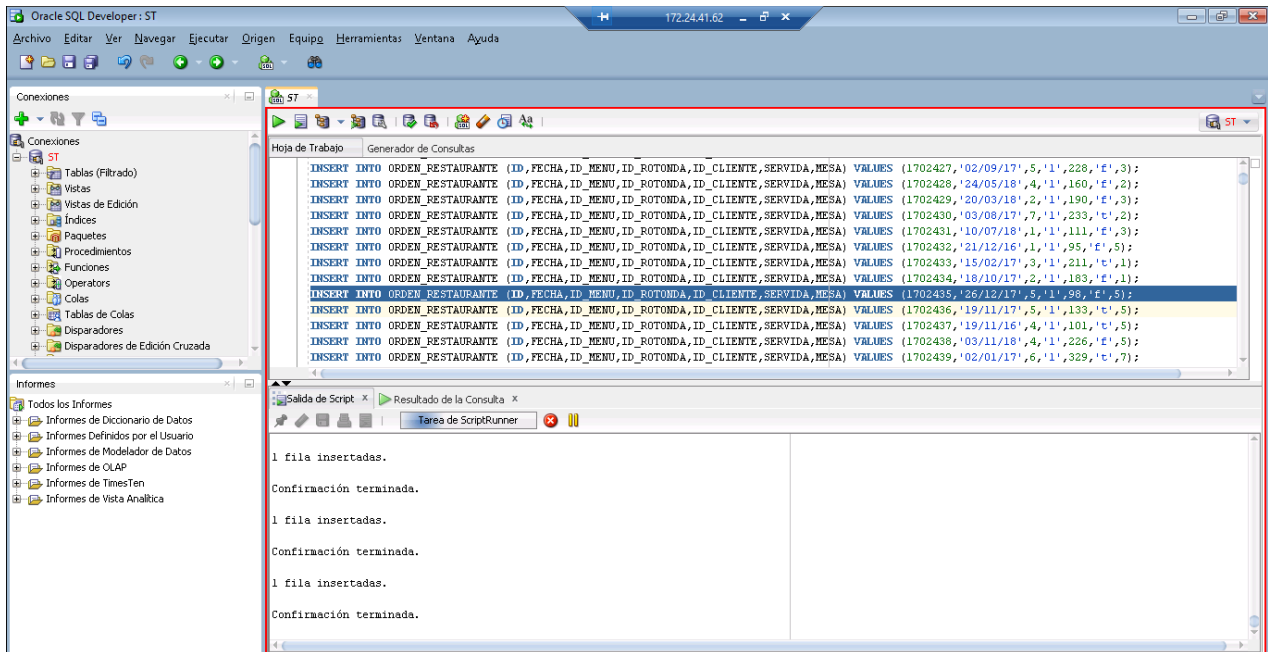


Imagen 41: Ordenes siendo agregadas a la DB.

The screenshot shows the Oracle SQL Developer interface with the ORDEN_RESTAURANTE table selected. The table view displays the following data:

ID	FECHA	ID_MENU	ID_ROTONDA	ID_CLIENTE	SERVIDA	MESA
1	199123065299	13/05/18	3	1	148 t	9
2	1699122...	20/07/17	4	1	222 t	5
3	1699122...	18/10/18	0	1	209 f	4
4	1699122...	23/05/17	1	1	175 f	6
5	1699122...	14/05/18	3	1	79 t	3
6	1699122...	28/02/18	1	1	251 t	7
7	1699122...	07/10/17	0	1	2 f	6
8	1699122...	17/10/18	2	1	145 t	7
9	1699122...	14/02/18	7	1	7 f	1
10	1699122...	04/03/18	1	1	170 t	6
11	1699122...	29/07/17	7	1	65 f	5
12	1699122...	27/07/18	2	1	274 t	5
13	1699122...	06/03/18	0	1	151 f	3
14	1699122...	13/06/18	0	1	91 t	1
15	1699122...	15/04/17	7	1	229 f	4
16	1699122...	07/01/18	6	1	108 f	2
17	1699122...	14/07/17	2	1	300 t	1
18	1699122...	03/09/18	6	1	34 t	1
19	1699122...	20/02/17	7	1	113 t	5
20	1699122...	04/08/17	7	1	262 f	4
21	1699121...	20/09/17	0	1	189 t	4
22	1699121...	15/04/17	5	1	123 t	6
23	1699121...	14/01/17	6	1	140 t	4
24	1699121...	12/09/18	5	1	154 f	7
25	1699121...	14/10/17	2	1	174 f	2

Imagen 42: 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 $O(n^4)$

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