Iteración 4

Angelo Marcetty Valero Sanchez, Sebastian Beltran.
Universidad de los Andes, Bogota, Colombia
Am.valero10@uniandes.edu.co
Js.beltran14@uniandes.edu.co

Tabla Contenido

1. **Análisis**: Luego de analizar sobre los nuevos requerimientos, se realizó una revisión al modelo UML perteneciente a la iteración anterior, el cual después de analizar, se decidió no modificar, ya que es posible cumplir con los requerimientos pedidos con este modelo, para las consultas más específicas, se realizaron clases auxiliares para la representación de la consulta y sus datos específicos. Para finalizar los datos sobre los pedidos contenían un atributo de TimeStamp el cual fue modificado a date para realizar consultas más efectivas.

2. **Diseño de la Aplicación**

   A) Cambios: Los nuevos requerimientos realizan consultas en tablas con tamaños enormes de datos, lo cual impacta enormemente en la obtención de datos en la consulta. También al estar tan poblados los datos la eficiencia de las consultas se ve enormemente afectada, aunque la corrección de iteraciones anteriores, ayudaron para validar reglas de negocio, las consultas están estrechamente afectadas. Explícitamente los cambios realizados fueron agregar nuevos VOS para las consultas de los nuevos requerimientos, DAO y Services para su respectivo requerimiento y la creación de índices para la optimización de consultas.

   B) Diseño Físico: La implementación de los índices se hicieron de acuerdo a las necesidades de la consulta, para la creación de los índices de los alimentos como acompañamiento, entrada, bebida, postre se pensaba implementar el índice de B+, pero al realizar la creación SQL-Developer arrojaba un error informando que ya existía una creación de índices por defecto.

```
Error que empieza en la línea: 4 del comando :
CREATE INDEX INDICE_ACOMP
ON ACOMPANIAMIENTO(ID)
Informe de error -
ORA-01408: esta lista de columnas ya está indexada
01408. 00000 -  "such column list already indexed"
*Cause:
*Action:
Confirmación terminada.
```

Los índices determinados por el SMDB son los siguientes en nuestras tablas más importantes y determinantes para las consultas, los cuales para la extracción de su información es facilitada por el ID y están compuestos por índices B+ realizado por defecto por el SMDB.

ACOMPANIAMIENTO

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A371720	ACOMPANIAMIENTO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

BEBIDA

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A371720	BEBIDA_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

CLIENTEUS

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A371720	CLIENTEUS_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

ENTRADA

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A371720	ENTRADA_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

POSTRE

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A371720	POSTRE_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

RESTAURANTES

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A371720	RESTAURANTE_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

Para la tabla de Plato_Fuerte la cual era estrechamente ligada a los nuevos requerimientos, se pensaba realizar un índice doble (Clustered), conformada por su ID y el PRECIO_VENTA, necesario para las consultas de la iteración, pero al momento de su creación informo de un problema de cuota de espacio.

```
CREATE INDEX INDICE_PLATO_FUERTE
ON PLATO_FUERTE(ID,PRECIO_VENTA);
COMMIT
```

```
Salida de Script | Explicación del Plan | Resultado de la Consulta
Tarea terminada en 0,188 segundos
Error que empieza en la línea: 4 del comando :
CREATE INDEX INDICE_PLATO_FUERTE
ON PLATO_FUERTE(ID,PRECIO_VENTA)
Informe de error -
ORA-01536: cuota de espacio excedida para tablespace 'TBSPROD'
01536. 00000 -  "space quota exceeded for tablespace '%s'"
*Cause:    The space quota for the segment owner in the tablespace has
           been exhausted and the operation attempted the creation of a
           new segment extent in the tablespace.
*Action:   Either drop unnecessary objects in the tablespace to reclaim
           space or have a privileged user increase the quota on this
           tablespace for the segment owner.
Confirmación terminada.
```

Para la tabla de PEDIDO era necesario la realización de un índice BITMAP en la columna de ESTADO, ya que era determinante para comprobar los pedidos que fueron completado(aquello que se realizó la compra de sus productos), para la realización de la consulta en fechas recurrentes dadas en las tablas de PEDIDO, al realizar el índice SQL_Developer nos informo de un problema por cuota de espacio.

```
CREATE BITMAP INDEX INDICE_ESTADO
ON PEDIDO(ESTADO)
COMMIT
```

```
Salida de Script | Explicación del Plan | Resultado de la Consulta
Tarea terminada en 0,103 segundos

rror que empieza en la línea: 8 del comando :
REATE BITMAP INDEX INDICE_ESTADO
N PEDIDO(ESTADO)
nforme de error -
RA-01536: cuota de espacio excedida para tablespace 'TBSPROD'
1536. 00000 -  "space quota exceeded for tablespace '%s'"
Cause:    The space quota for the segment owner in the tablespace has
          been exhausted and the operation attempted the creation of a
          new segment extent in the tablespace.
Action:   Either drop unnecessary objects in the tablespace to reclaim
          space or have a privileged user increase the quota on this
          tablespace for the segment owner.
```

Por último la necesidad de índices para las otras tablas es cubierta por SMDB gracias a su creación por defecto de B+ sobre los ID necesarios para los joins y consultas determinadas, estos índices son acertados ya que la mayoría de consultas recaen sobre llaves foráneas y la extracción de la información asociada a estas es más optimizada al buscarlas por su id.

C) Documentación Requerimientos Funcionales de Consulta

RFC9.

- **Escenario de pruebas**: Se realizó pruebas en Postman de dos post con la dirección: http://localhost:8080/RotondAndes/rest/administradores/1/clientes/ y http://localhost:8080/RotondAndes/rest/administradores/1/clientespedido/ el primero es cuando se desea ordenar por datos del cliente como nombre y el segundo por datos del pedido como costo. En caso de que no se quiera ningún orden se usa la primera ruta y se coloca nada en ordenar del cuerpo del json. los ejemplos del cuerpo del json son:

```
{
        "ordenar":"costo", // es el parámetro por el que se quiere ordenar
        "dini":11,          // es dia de la fecha inicial
        "mini":11,          // es mes de la fecha inicial
        "yini":2015,        // es año de la fecha inicial
        "dfin":11,          // es dia de la fecha final
        "mfin":11,          // es mes de la fecha final
        "yfin":2017,        // es año de la fecha final
        "idr":1             // es el id del restaurante
}
```

- **Sentencia SQL:**
  select c.ID,c.TIPOID,c.NOMBRE,c.CORREO,c.ROL, // selecciona los datos del cliente

  from CLIENTEUS c inner JOIN PEDIDO p on c.ID = p.IDUSUARIO // se realiza un inner join entre la tabla de clientes y pedido en el id que le pertenece al cliente

  where p.FECHA > '01-01-2016' AND p.FECHA < '31-12-2017' AND p.ID_RESTAURANTE = 1 //se especifica el rango de fechas y el restaurante deseado

  GROUP BY c.ID, c.TIPOID, c.NOMBRE, c.CORREO, c.ROL //agrupa por los datos del cliente

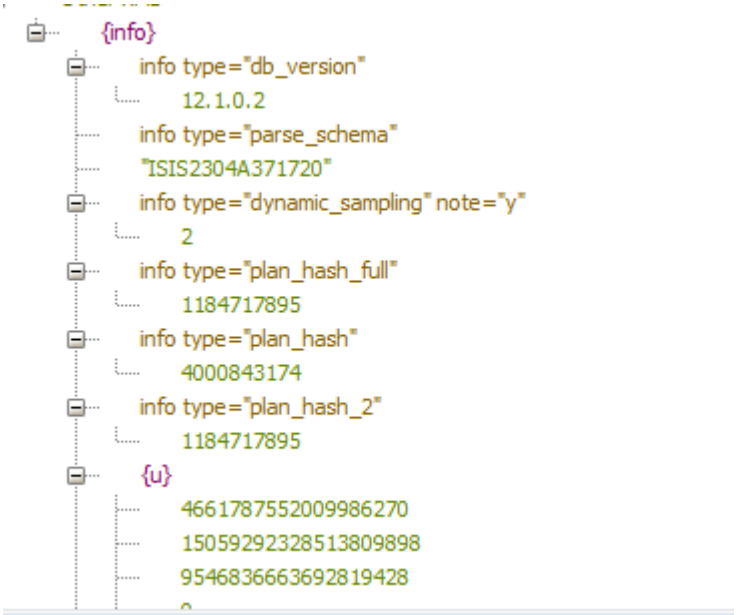
  ORDER BY p.fecha; // ordena por el parámetro deseado

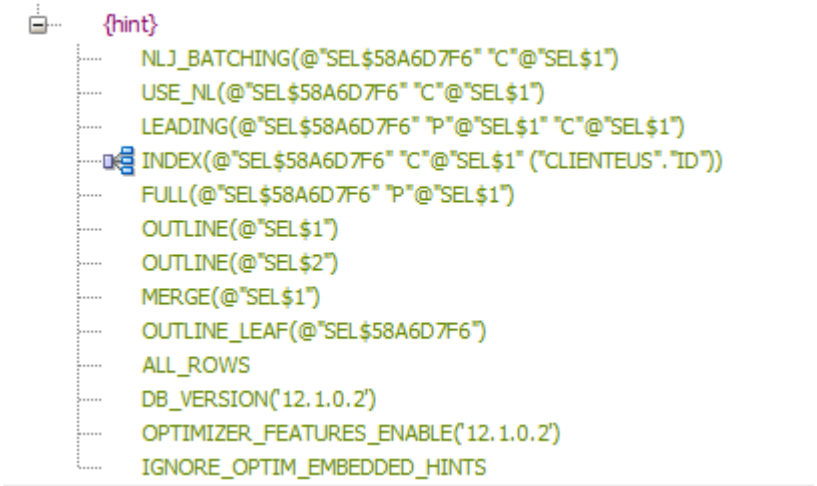
- **Distribución de los Datos:** el parámetro que más se debe tener en cuenta es el rango de fechas un rango arto genera más resultados que el corto.

- **Plan de Ejecucion y Tiempos Obtenidos:**

SQL | 1.167 segundos

ATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	14
SORT		GROUP BY	5	14
NESTED LOOPS			7	13
NESTED LOOPS			7	13
TABLE ACCESS	PEDIDO	FULL	5	3
Filter Predicates				
AND				
P.ID_RESTAURANTE=1				
P.FECHA>TO_DATE(' 2017-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
P.FECHA<TO_DATE(' 2017-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	CLIENTEUS_PK	UNIQUE SCAN	1	1
Access Predicates				
C.ID=P.IDUSUARIO				
TABLE ACCESS	CLIENTEUS	BY INDEX ROWID	1	2
Other XML				
{info}				





RFC10.

no se logro este punto, se tubo la idea de usar un except o un minus, pero final mente no que do funcional.

```
 select c.ID,c.TIPOID,c.NOMBRE,c.CORREO,c.ROL

 from CLIENTEUS c inner JOIN PEDIDO p on c.ID = p.IDUSUARIO

  where p.FECHA > '30-12-2015' AND p.FECHA < '31-12-2017' AND
p.ID_RESTAURANTE = 1

 GROUP BY c.ID, c.TIPOID, c.NOMBRE, c.CORREO, c.ROL

 minus

  select c.ID, c.TIPOID, c.NOMBRE,c.CORREO,c.ROL

  FROM CLIENTEUS c inner JOIN PEDIDO p on c.ID = p.IDUSUARIO

  where p.FECHA < '30-12-2015' AND p.FECHA > '31-12-2017' AND
p.ID_RESTAURANTE = 1

  GROUP BY c.ID, c.TIPOID, c.NOMBRE, c.CORREO, c.ROL;
```

se usaron varias iteraciones de este sql

RFC11. CONSULTAR FUNCIONAMIENTO

- **Escenario de pruebas**: Se realizo una prueba en Postman de un get con la dirección: http://localhost:8080/RotondAndes/rest/administradores/1/consultaFuncionamiento en la cual es solicitada y realizada por un administrador seguido por el id perteneciente a el y con el Service de la consulta.
- **Sentencia SQL:**

  Extracción de las fechas disponibles en la tabla de PEDIDO

```
SELECT DISTINCT FECHA
FROM PEDIDO
WHERE ESTADO = 'Servido'  ORDER BY FECHA"
```

Selección del restaurante más visitado dada una fecha especifica.

```
SELECT *
FROM
(
SELECT DISTINCT(P.ID_RESTAURANTE), COUNT(P.ID_RESTAURANTE)
AS RECURRENCIA ,
to_char(to_date('"+fecha+"','YYYY/MM/DD'),'Day') AS DIA
FROM PEDIDO P
WHERE FECHA = TO_DATE('"+ fecha +"','YYYY/MM/DD') AND
ESTADO = 'Servido'
GROUP BY P.ID_RESTAURANTE
) A INNER JOIN RESTAURANTES R ON A.ID_RESTAURANTE = R.ID
WHERE ROWNUM = 1
ORDER BY RECURRENCIA DESC
```

Selección del restaurante menos visitado dada una fecha especifica.

```
SELECT *
FROM
(
SELECT DISTINCT(P.ID_RESTAURANTE), COUNT(P.ID_RESTAURANTE)
AS RECURRENCIA
FROM PEDIDO P
WHERE FECHA = TO_DATE('"+ fecha +"','YYYY/MM/DD') AND
ESTADO = 'Servido'
GROUP BY P.ID_RESTAURANTE
ORDER BY RECURRENCIA ASC
) A INNER JOIN RESTAURANTES R ON A.ID_RESTAURANTE = R.ID
WHERE ROWNUM = 1
```

Selección del producto más consumido dada una fecha específica, sentencia con ejemplo de la ENTRADA.

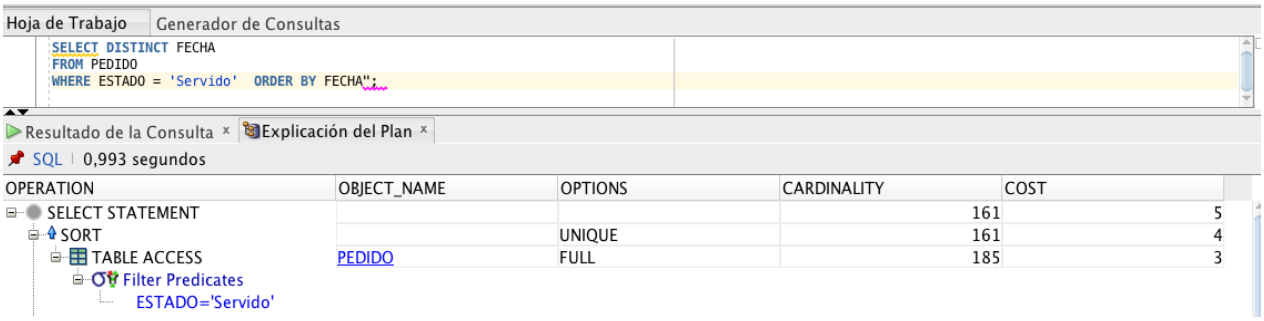
```
SELECT *
FROM
(
SELECT DISTINCT(P.ID_ENTRADA) AS ID, COUNT(P.ID_ENTRADA)
AS VECES
FROM PEDIDO P
WHERE FECHA = TO_DATE('"+fecha+"','YYYY/MM/DD') AND ESTADO
= 'Servido'
GROUP BY P.ID_ENTRADA
ORDER BY VECES DESC
)
 WHERE ROWNUM = 1
```

Selección del producto menos consumido dada una fecha específica, sentencia con ejemplo de la ENTRADA.

```
"SELECT *
FROM
(
SELECT DISTINCT(P.ID_ENTRADA) AS ID, COUNT(P.ID_ENTRADA)
AS VECES
FROM PEDIDO P
WHERE FECHA = TO_DATE('"+fecha+"','YYYY/MM/DD') AND ESTADO
= 'Servido'
GROUP BY P.ID_ENTRADA
ORDER BY VECES ASC
)
WHERE ROWNUM = 1
```

- **Distribucion de los Datos:** Debido a que la extracción de los datos resulta en una cantidad enorme de resultados, es importante especificar un cantidad razonable para su correcta consulta, los parámetros impuestos por defecto son menores a 20 debido a la cantidad de datos, para una considerable visualización.
- **Plan de Ejecucion y Tiempos Obtenidos:**

Consulta 1



Consulta 2

```
SELECT *
FROM
(
SELECT DISTINCT(P.ID_RESTAURANTE), COUNT(P.ID_RESTAURANTE) AS RECURRENCIA , to_char(to_date('"+fecha+"','YYYY/MM/DD'),'Day') AS DIA
FROM PEDIDO P
WHERE FECHA = TO_DATE('2017-02-17','YYYY/MM/DD') AND ESTADO = 'Servido'
GROUP BY P.ID_RESTAURANTE
) A INNER JOIN RESTAURANTES R ON A.ID_RESTAURANTE = R.ID
WHERE ROWNUM = 1
ORDER BY RECURRENCIA DESC;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,21 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
SORT		ORDER BY	1	7
COUNT		STOPKEY		
Filter Predicates				
ROWNUM=1				
HASH JOIN			1	6
Access Predicates				
A.ID_RESTAURANTE=R.ID				
NESTED LOOPS			1	6
NESTED LOOPS			1	6
STATISTICS COLLECTOR				
VIEW			1	4
HASH		GROUP BY	1	4
TABLE ACCESS	PEDIDO	FULL	1	3

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
TABLE ACCESS	PEDIDO	FULL	1	3
Filter Predicates				
AND				
FECHA=TO_DATE(' 2017-02-17 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
ESTADO='Servido'				
INDEX	RESTAURAN...	UNIQUE SCAN	1	1
Access Predicates				
A.ID_RESTAURANTE=R.ID				
TABLE ACCESS	RESTAURAN...	BY INDEX ROWID	1	2
TABLE ACCESS	RESTAURAN...	FULL	1	2

Consulta 3

Hoja de Trabajo | Generador de Consultas

```
SELECT *
FROM
(
SELECT DISTINCT(P.ID_RESTAURANTE), COUNT(P.ID_RESTAURANTE) AS RECURRENCIA
FROM PEDIDO P
WHERE FECHA = TO_DATE('2017-02-17','YYYY/MM/DD') AND ESTADO = 'Servido'
GROUP BY P.ID_RESTAURANTE
ORDER BY RECURRENCIA ASC
) A INNER JOIN RESTAURANTES R ON A.ID_RESTAURANTE = R.ID
WHERE ROWNUM = 1;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,813 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
COUNT		STOPKEY		
Filter Predicates				
ROWNUM=1				
HASH JOIN			1	7
Access Predicates				
A.ID_RESTAURANTE=R.ID				
NESTED LOOPS			1	7
NESTED LOOPS			1	7
STATISTICS COLLECTOR				
VIEW			1	5
SORT		ORDER BY	1	5
HASH		GROUP BY	1	5
TABLE ACCESS	PEDIDO	FULL	1	3

SQL | 0,813 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
HASH		GROUP BY	1	5
TABLE ACCESS	PEDIDO	FULL	1	3
Filter Predicates				
AND				
FECHA=TO_DATE(' 2017-02-17 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
ESTADO='Servido'				
INDEX	RESTAURAN...	UNIQUE SCAN	1	1
Access Predicates				
A.ID_RESTAURANTE=R.ID				
TABLE ACCESS	RESTAURAN...	BY INDEX ROWID	1	2
TABLE ACCESS	RESTAURAN...	FULL	1	2

Consulta 4

```
SELECT *
FROM
(
SELECT DISTINCT(P.ID_ENTRADA) AS ID, COUNT(P.ID_ENTRADA) AS VECES
FROM PEDIDO P
WHERE FECHA = TO_DATE('2017-02-17','YYYY/MM/DD') AND ESTADO = 'Servido'
GROUP BY P.ID_ENTRADA
ORDER BY VECES DESC
)
WHERE ROWNUM = 1;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,226 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	5
COUNT		STOPKEY		
Filter Predicates				
ROWNUM=1				
VIEW			1	5
SORT		ORDER BY STOPKEY	1	5
Filter Predicates				
ROWNUM=1				
HASH		GROUP BY	1	5
TABLE ACCESS	PEDIDO	FULL	1	3
Filter Predicates				
AND				
FECHA=TO_DATE(' 2017-02-17 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
ESTADO='Servido'				

Consulta 5

Hoja de Trabajo | Generador de Consultas

```
SELECT *
FROM
(
SELECT DISTINCT(P.ID_ENTRADA) AS ID, COUNT(P.ID_ENTRADA) AS VECES
FROM PEDIDO P
WHERE FECHA = TO_DATE('2017-02-17','YYYY/MM/DD') AND ESTADO = 'Servido'
GROUP BY P.ID_ENTRADA
ORDER BY VECES ASC
)
WHERE ROWNUM = 1;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,41 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	5
COUNT		STOPKEY		
Filter Predicates				
ROWNUM=1				
VIEW			1	5
SORT		ORDER BY STOPKEY	1	5
Filter Predicates				
ROWNUM=1				
HASH		GROUP BY	1	5
TABLE ACCESS	PEDIDO	FULL	1	3
Filter Predicates				
AND				
FECHA=TO_DATE(' 2017-02-17 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
ESTADO='Servido'				

RFC12. CONSULTAR LOS BUENOS CLIENTES: Para este requerimiento

- **Escenario de pruebas**: Se realizo una prueba en Postman de un get con la dirección: http://localhost:8080/RotondAndes/rest/administradores/1/consultaBuenosClientes en la cual es solicitada y realizada por un administrador seguido por el id perteneciente a el y con el Service de la consulta.
- **Sentencia SQL:**
  Extraer los clientes que han consumido por lo menos una vez a la semana.

```
SELECT DISTINCT(P.IDUSUARIO), P.FECHA,
to_char(to_date(P.FECHA,'DD/MM/YY'),'WW') AS SEMANA,
C.TIPOID, C.NOMBRE, C.CORREO, C.ROL,
to_char(to_date(P.FECHA,'DD/MM/YY'),'Day') AS DIA
FROM PEDIDO P INNER JOIN CLIENTEUS C ON P.IDUSUARIO =
C.ID
ORDER BY FECHA
```

  Extraer los clientes que no han realizado ningún consumo

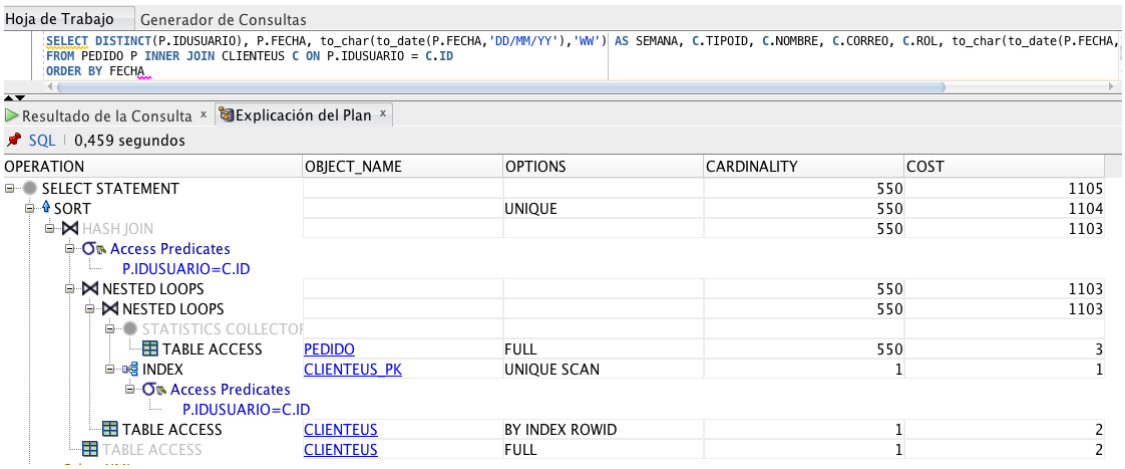
```
SELECT C.ID, C.TIPOID, C.NOMBRE, C.CORREO, C.ROL
FROM PEDIDO P RIGHT JOIN CLIENTEUS C ON P.IDUSUARIO = C.ID
WHERE P.IDUSUARIO IS NULL
ORDER BY C.ID ASC
```

  Extraer los clientes que tiene un consumo mayor a 1.5 SMDLV

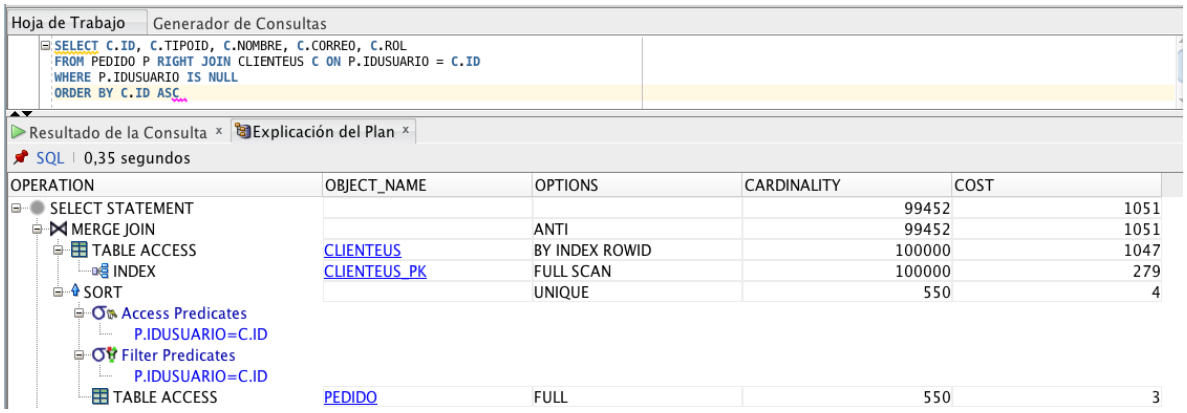
```
SELECT P.ID, C.TIPOID, C.NOMBRE, C.CORREO, C.ROL, P.COSTO
FROM PEDIDO P INNER JOIN CLIENTEUS C ON P.IDUSUARIO = C.ID
AND P.COSTO > 1107000 AND P.ESTADO = 'Servido'
```

- **Distribucion de los Datos:** Debido a que la extracción de los datos resulta en una cantidad enorme de resultados, es importante especificar un cantidad razonable para su correcta consulta, los parámetros impuestos por defecto son menores a 20 debido a la cantidad de datos, para una considerable visualización.
- **Plan de Ejecucion y Tiempos Obtenidos:**

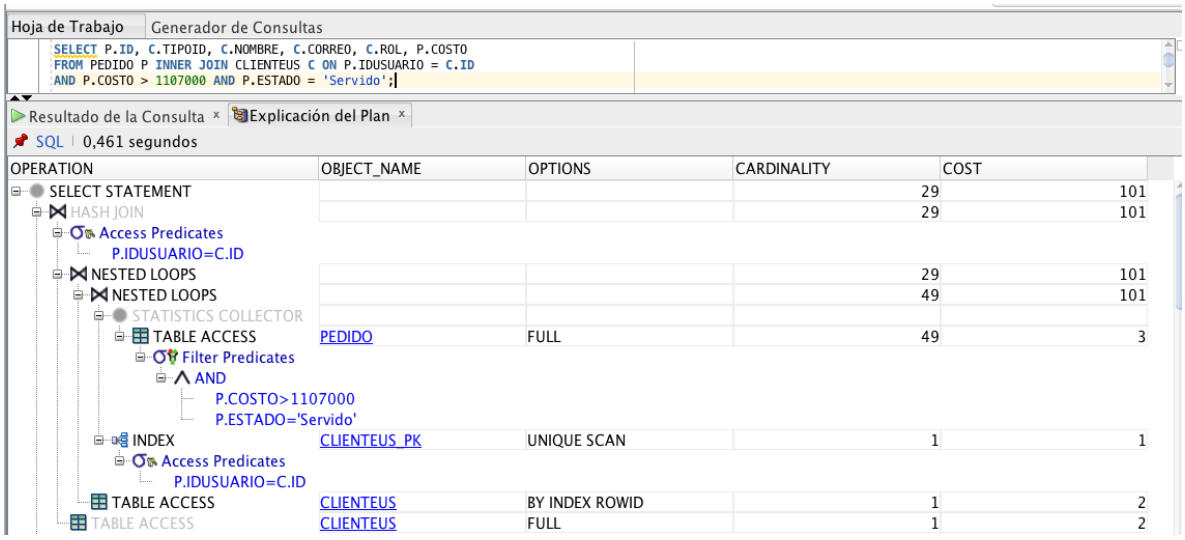
Consulta 1



Consulta 2

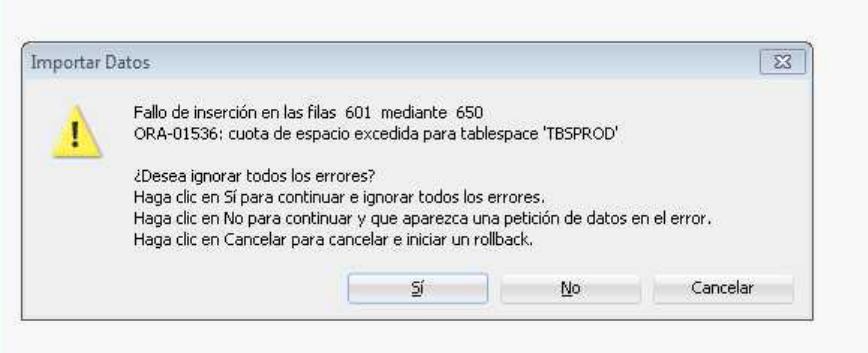


Consulta 3



3. Construcción de la Aplicación y Análisis de Resultados

- **Diseño del escenario de pruebas de eficiencia:** La carga de los datos fue realizada a partir de la creación de los mismos con una aplicación que generaba grandes volúmenes de datos, dado un rango o ciertas restricciones de estos. Se insertaron datos pertinentes y acordes con lo especificado en las columnas, pues para respetar las reglas de negocio, es necesario generar datos lo más coherentes posibles. Luego se exportaban en un archivo CSV para posteriormente por medio de SQL-Developer realizar la importación de datos en cada tabla, seleccionando la tabla, especificando el formato UTF-8, y emparejando las columnas asociadas a la información generada. Por último la importación de los datos se insertaron en las tablas con un costo de tiempo considerable, se insertaron un millón de tuplas por tabla aproximadamente, sin embargo para las tablas más importantes como pedido que se encuentra estrechamente relacionado con los requerimientos de la iteración, ocurrio un problema arrojado por el SMDB el cual hace referencia a una cuota de espacio, lo cual no nos permitió poblar las tablas adecuadamente insertando solo 500 filas de las de 1millon que estaban disponibles para PEDIDO.



- **Análisis del proceso de optimización y el modelo de ejecución de consultas**

Al momento de realizar consultas delegadas en su totalidad al SMDB este se encuentra totalmente optimizado siendo más eficiente en las consultas por las siguientes razones:

- Si los datos de una tabla no caben en memoria, al tener la necesidad de cargarlos por partes el SMDB tiene algoritmos de ordenamientos óptimos como el sort-merge para la resolución de este problema .
- Suponiendo que los datos caben en memoria, buscar y transferir los datos a memoria principal tiene un costo elevado, la carga y el tiempo de extracción de estos datos son altos, pero con el SMDB se puede hacer consultas mucho más precisas y exactas para la extracción de los mismos con tiempos y cargas más eficientes.
- Si es el caso y las tablas del sistema administrador se encuentra con índices correctos, las consultas y la extracción de los datos mejoran enormemente, ya que con los índices ordenados específicamente para ciertas consultas como los Clustered y Non-Clusteres, mejoran los tiempos de respuesta.

las consultas y la información siempre será mejor tratarlas y traer desde las sentencias ejecutadas en el SMDB, el uso de los datos traídos a memoria y ejecutados con instrucciones de control como if,while, etc tendrán un costo enorme en carga y ejecución lo cual no es recomendable para datos extremadamente grandes, reflejando tiempos de respuesta largos y a veces sin determinada estimación, a diferencia de los operadores como joins o select son en gran medida mejores y que además proceden de un sistema que se encuentra en un estado óptimo para la carga y consulta datos.