“Iteracion 4”

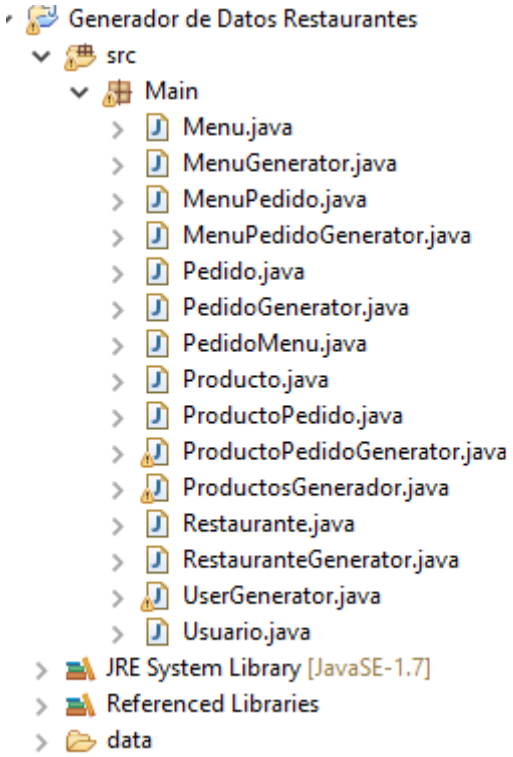
Juan Sebastián Torres, Manuel Vallejo
Contexto de Presentación del documento
Universidad de los Andes, Bogotá, Colombia
{js.torres1,mf.valllejo} @uniandes.edu.co

Fecha de presentación: noviembre 21 de 2017

Tabla de contenido

1 Creación Base de Datos

Para poblar las bases de datos creamos un programa que generara csv con las características que necesitaban nuestras tablas de la base de datos, estos programas escribían los datos con coherencia, pero manteniendo el requerimiento de aleatoriedad



Con este programa generaba csv a partir de otros csv.

Las Tablas en las que se insertaron los datos fueron

1. Restaurantes
2. Usuarios
3. Pedidos
4. Menus
5. PedidoMenu
6. PedidoProducto

Entonces asi pudimos tener una distrubucion uniforme usando 500000 Restaurantes , 100000 usuarios, 300000 pedidos, 300000PedidosProducto, 100000 Menus, 10000 MenuProducto.

2 RFS en Base de Datos

1. **select * from usuarios where usuarios.id in (Select b.id as idUsuario from productos join (select pedidoproducto.IDPRODUCTO, a.id,a.name,a.fecha from pedidoproducto Join (select usuarios.id, usuarios.name , pedidos.fecha,pedidos.numpedido from usuarios join pedidos on idusuario = usuarios.ID where fecha between '12/07/10' and '13/07/30') a on pedidoproducto.NUMPEDIDO = a.NUMPEDIDO) b on productos.ID= b.IDPRODUCTO where productos.IDRESTAUANTE = '1');**

2. **select * from usuarios where usuarios.id not in (Select b.id as idUsuario from productos join (select pedidoproducto.IDPRODUCTO, a.id,a.name,a.fecha from pedidoproducto Join (select usuarios.id,**

```
usuarios.name , pedidos.fecha,pedidos.numpedido from usuarios join
pedidos on idusuario = usuarios.ID where fecha between '12/07/10' and
'13/07/10') a on pedidoproducto.NUMPEDIDO = a.NUMPEDIDO) b on
productos.ID= b.IDPRODUCTO where productos.IDRESTAUANTE = '1');
```

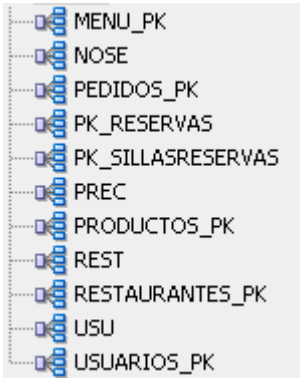
3. ```
   select dia,max(consumido) as maxi , max(idprod) keep (dense_rank first
   order by consumido desc) as producto from
       (select count(pedidoproducto.idproducto) as
   consumido,pedidoproducto.IDPRODUCTO as
   idprod,to_char(pedidos.fecha, 'dy') as dia
       from pedidos join pedidoproducto
       on pedidos.NUMPEDIDO = pedidoproducto.NUMPEDIDO
       group by pedidoproducto.IDPRODUCTO, to_char(pedidos.fecha, 'dy'))
   group by dia;

       select dia, min(consumido) as min , min(idprod) keep (dense_rank first
   order by consumido desc) as producto from
       (select count(pedidoproducto.idproducto) as
   consumido,pedidoproducto.IDPRODUCTO as
   idprod,to_char(pedidos.fecha, 'dy') as dia
       from pedidos join pedidoproducto
       on pedidos.NUMPEDIDO = pedidoproducto.NUMPEDIDO
       group by pedidoproducto.IDPRODUCTO, to_char(pedidos.fecha, 'dy'))
   group by dia;
   ```

4. ```
   select *
       from usuarios where usuarios.id not in  (select a.id from pedidomenu
   Join (select usuarios.id, usuarios.name, pedidos.numpedido from
   usuarios join pedidos
       on idusuario = usuarios.ID ) a
       on pedidomenu.NUMPEDIDO = a.NUMPEDIDO)
       union all
       select *
       from usuarios where usuarios.id in(Select b.id
       from productos join (select a.id,pedidoproducto.IDPRODUCTO from
   pedidoproducto Join (select usuarios.id, usuarios.name,
   pedidos.numpedido from usuarios join pedidos
       on idusuario = usuarios.ID ) a
       on pedidoproducto.NUMPEDIDO = a.NUMPEDIDO) b
       on productos.id = b.IDPRODUCTO where productos.precio > 36885);
   ```

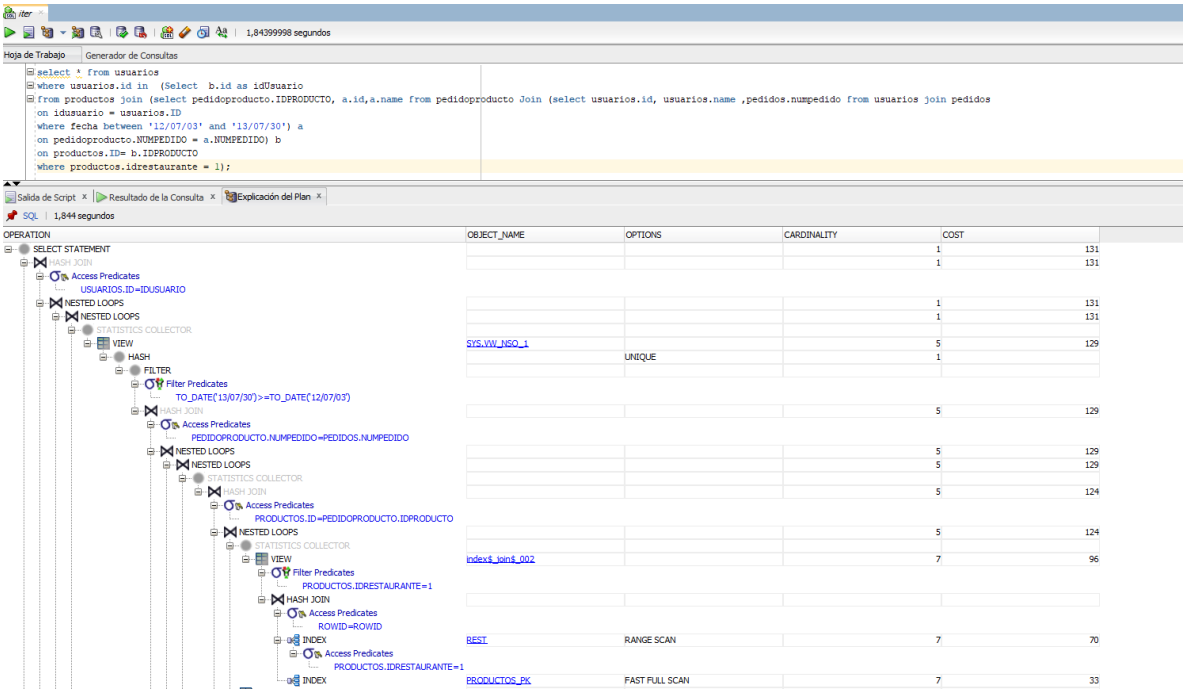
3 PLANES:

A continuación, se muestra todos los planes de Cada Requerimiento con la información que utilizamos para reducir costes y velocidad.

INDEX:

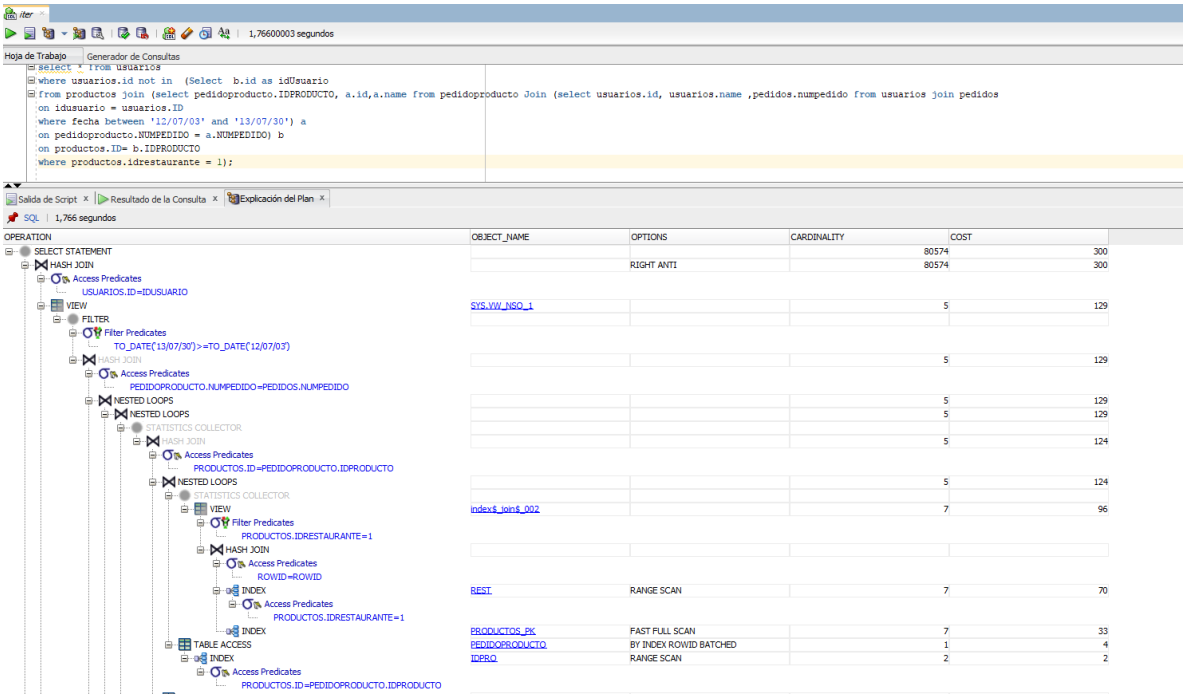


3.1 RFC9:



El indice de id en restaurantes y utilizamos otro indice en Pedidos con usuario nos bajo la velociadad de la consulta y del costo bajo bastantes puntos.

3.2 RFC10:

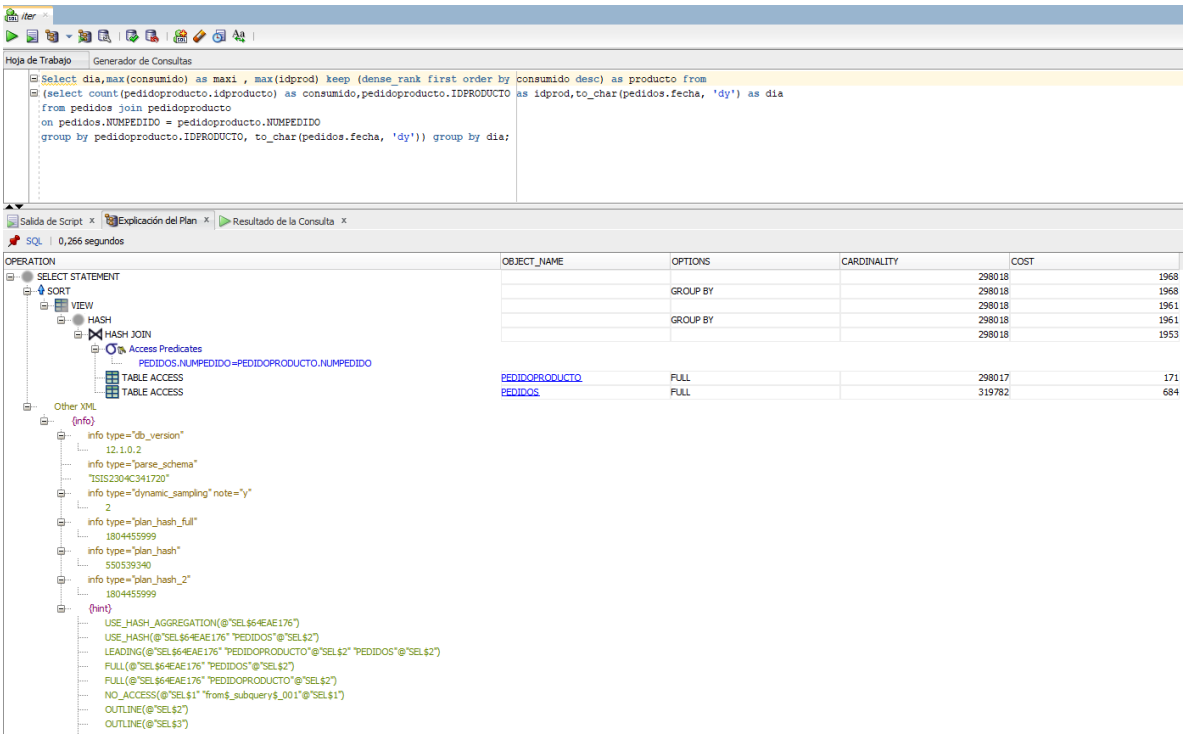


```
select * from usuarios
where usuarios.id not in  (Select  b.id as idUsuario
from productos join (select pedidoproducto.IDPRODUCTO, a.id,a.name from pedidoproducto Join (select usuarios.id, usuarios.name ,pedidos.numpedido from usuarios join pedidos
on idusuario = usuarios.ID
where fecha between '12/07/03' and '13/07/30') a
on pedidoproducto.NUMPEDIDO = a.NUMPEDIDO) b
on productos.ID= b.IDPRODUCTO
where productos.idrestaurante = 1);
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			80574	300
HASH JOIN		RIGHT ANTI	80574	300
Access Predicates				
USUARIOS.ID=IDUSUARIO				
VIEW	SYS.VW_NSO_1		5	129
FILTER				
Filter Predicates				
TO_DATE('13/07/30')>=TO_DATE('12/07/03')				
HASH JOIN			5	129
Access Predicates				
PEDIDOPRODUCTO.NUMPEDIDO=PEDIDOS.NUMPEDIDO				
NESTED LOOPS			5	129
NESTED LOOPS			5	129
STATISTICS COLLECTOR				
HASH JOIN			5	124
Access Predicates				
PRODUCTOS.ID=PEDIDOPRODUCTO.IDPRODUCTO				
NESTED LOOPS			5	124
STATISTICS COLLECTOR				
VIEW	index$_join$_002		7	96
Filter Predicates				
PRODUCTOS.IDRESTAURANTE=1				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	REST	RANGE SCAN	7	70
Access Predicates				
PRODUCTOS.IDRESTAURANTE=1				
INDEX	PRODUCTOS_PK	FAST FULL SCAN	7	33
TABLE ACCESS	PEDIDOPRODUCTO	BY INDEX ROWID BATCHED	1	4
INDEX	IDPRO	RANGE SCAN	2	2
Access Predicates				
PRODUCTOS.ID=PEDIDOPRODUCTO.IDPRODUCTO				

Utilizamos los mismos indices que el rquerimiento anterior sin embargo el coste sube por el numero de tuplas que cumplen ademas de la funcion not in.

3.3 RFC11:



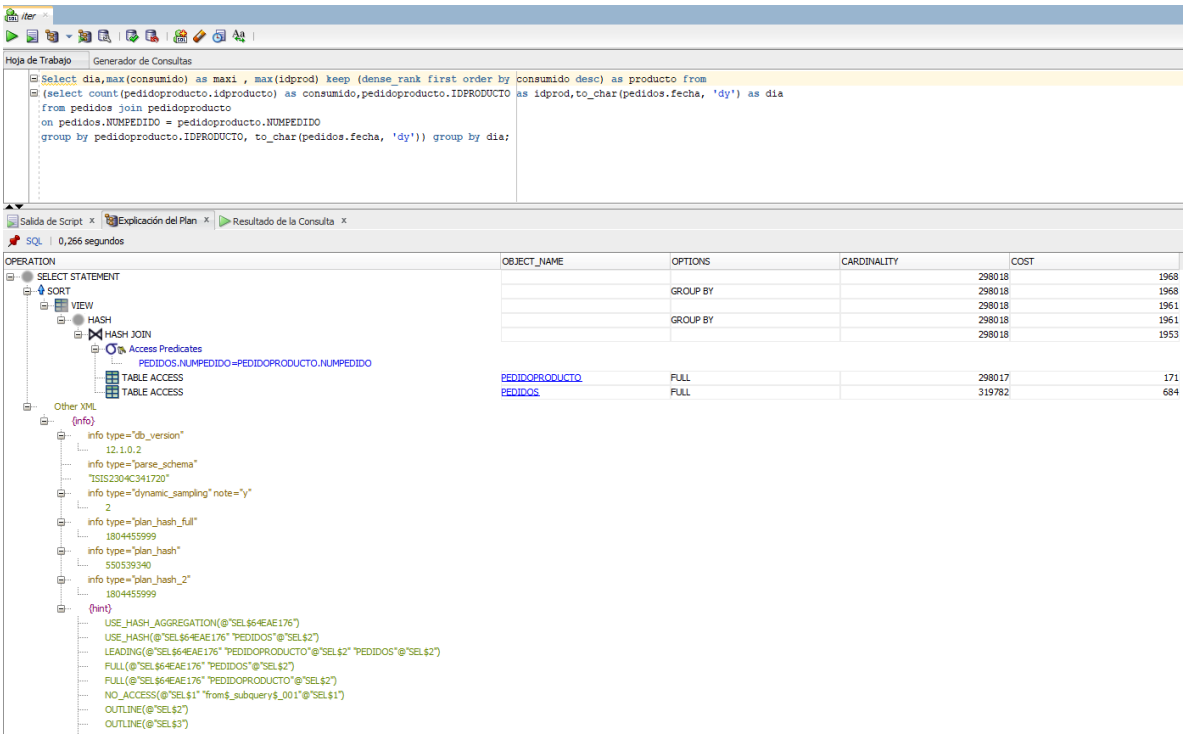
```
Select dia,max(consumido) as maxi , max(idprod) keep (dense_rank first order by consumido desc) as producto from
(select count(pedidoproducto.idproducto) as consumido,pedidoproducto.IDPRODUCTO as idprod,to_char(pedidos.fecha, 'dy') as dia
from pedidos join pedidoproducto
on pedidos.NUMPEDIDO = pedidoproducto.NUMPEDIDO
group by pedidoproducto.IDPRODUCTO, to_char(pedidos.fecha, 'dy')) group by dia;
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			298018	1968
SORT		GROUP BY	298018	1968
VIEW			298018	1961
HASH		GROUP BY	298018	1961
HASH JOIN			298018	1953
Access Predicates				
PEDIDOS.NUMPEDIDO=PEDIDOPRODUCTO.NUMPEDIDO				
TABLE ACCESS	PEDIDOPRODUCTO	FULL	298017	171
TABLE ACCESS	PEDIDOS	FULL	319782	684

Other XML

```
{info}
  info type="db_version"
    12.1.0.2
  info type="parse_schema"
  "ISIS2304C341720"
  info type="dynamic_sampling" note="y"
    2
  info type="plan_hash_full"
    1804455999
  info type="plan_hash"
    550539340
  info type="plan_hash_2"
    1804455999
  {hint}
    USE_HASH_AGGREGATION(@"SEL$64EAE176")
    USE_HASH(@"SEL$64EAE176" "PEDIDOS"@"SEL$2")
    LEADING(@"SEL$64EAE176" "PEDIDOPRODUCTO"@"SEL$2" "PEDIDOS"@"SEL$2")
    FULL(@"SEL$64EAE176" "PEDIDOS"@"SEL$2")
    FULL(@"SEL$64EAE176" "PEDIDOPRODUCTO"@"SEL$2")
    NO_ACCESS(@"SEL$1" "from$_subquery$_001"@"SEL$1")
    OUTLINE(@"SEL$2")
    OUTLINE(@"SEL$3")
```

No supimos poner correctamente el index para reducir costos y eficiencia, intentamos ponerlo en el dia pero no lo utiliza en el plan de busqueda.



```
Select dia,max(consumido) as maxi , max(idprod) keep (dense_rank first order by consumido desc) as producto from
(select count(pedidoproducto.idproducto) as consumido,pedidoproducto.IDPRODUCTO as idprod,to_char(pedidos.fecha, 'dy') as dia
from pedidos join pedidoproducto
on pedidos.NUMPEDIDO = pedidoproducto.NUMPEDIDO
group by pedidoproducto.IDPRODUCTO, to_char(pedidos.fecha, 'dy')) group by dia;
```

Salida de Script | Explicación del Plan | Resultado de la Consulta

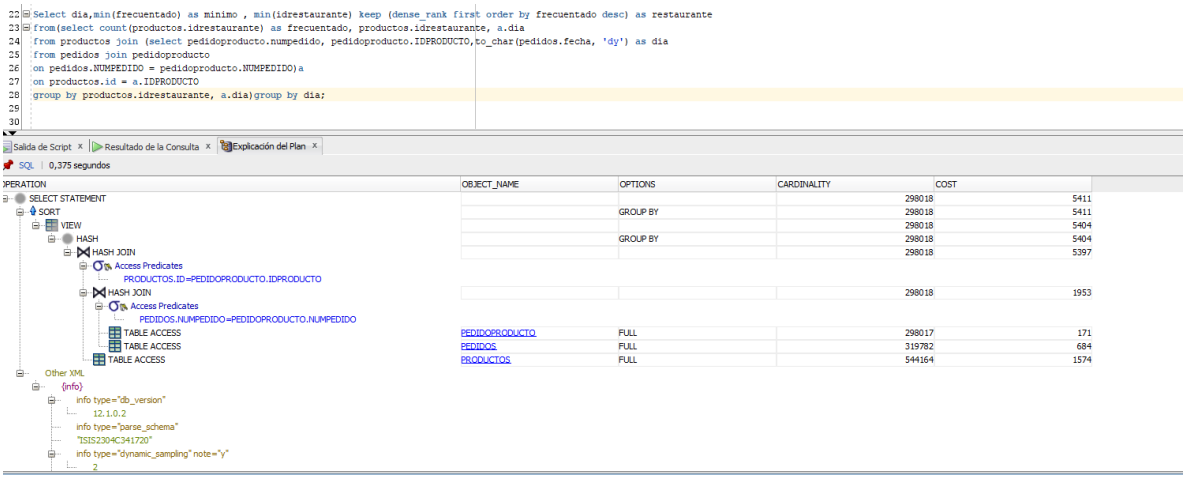
SQL | 0,266 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			298018	1968
SORT		GROUP BY	298018	1968
VIEW			298018	1961
HASH		GROUP BY	298018	1961
HASH JOIN			298018	1953
Access Predicates				
PEDIDOS.NUMPEDIDO=PEDIDOPRODUCTO.NUMPEDIDO				
TABLE ACCESS	PEDIDOPRODUCTO	FULL	298017	171
TABLE ACCESS	PEDIDOS	FULL	319782	684

Other XML

```
{info}
  info type="db_version"
    12.1.0.2
  info type="parse_schema"
  "ISIS2304C341720"
  info type="dynamic_sampling" note="y"
    2
  info type="plan_hash_full"
    1804455999
  info type="plan_hash"
    550539340
  info type="plan_hash_2"
    1804455999
  {hint}
    USE_HASH_AGGREGATION(@"SEL$64EAE176")
    USE_HASH(@"SEL$64EAE176" "PEDIDOS"@"SEL$2")
    LEADING(@"SEL$64EAE176" "PEDIDOPRODUCTO"@"SEL$2" "PEDIDOS"@"SEL$2")
    FULL(@"SEL$64EAE176" "PEDIDOS"@"SEL$2")
    FULL(@"SEL$64EAE176" "PEDIDOPRODUCTO"@"SEL$2")
    NO_ACCESS(@"SEL$1" "from$_subquery$_001"@"SEL$1")
    OUTLINE(@"SEL$2")
    OUTLINE(@"SEL$3")
```

Al igual que en el anterior no cogio el indice de dia.



```
22 Select dia,min(frecuentado) as minimo , min(idrestaurante) keep (dense_rank first order by frecuentado desc) as restaurante
23 from(select count(productos.idrestaurante) as frecuentado, productos.idrestaurante, a.dia
24 from productos join (select pedidoproducto.numpedido, pedidoproducto.IDPRODUCTO,to_char(pedidos.fecha, 'dy') as dia
25 from pedidos join pedidoproducto
26 on pedidos.NUMPEDIDO = pedidoproducto.NUMPEDIDO)a
27 on productos.id = a.IDPRODUCTO
28 group by productos.idrestaurante, a.dia)group by dia;
29
30
```

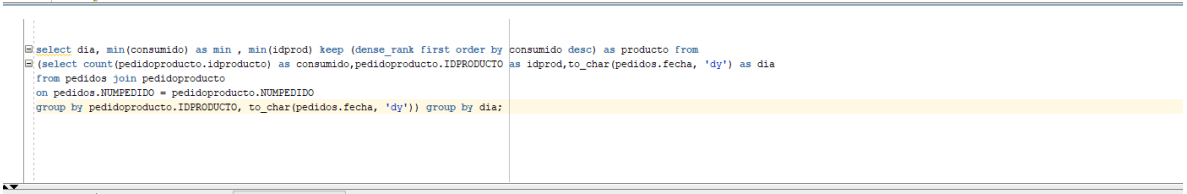
Salida de Script | Resultado de la Consulta | Explicación del Plan

SQL | 0,375 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			298018	5411
SORT		GROUP BY	298018	5411
VIEW			298018	5404
HASH		GROUP BY	298018	5404
HASH JOIN			298018	5397
Access Predicates				
PRODUCTOS.ID=PEDIDOPRODUCTO.IDPRODUCTO				
HASH JOIN			298018	1953
Access Predicates				
PEDIDOS.NUMPEDIDO=PEDIDOPRODUCTO.NUMPEDIDO				
TABLE ACCESS	PEDIDOPRODUCTO	FULL	298017	171
TABLE ACCESS	PEDIDOS	FULL	319782	684
TABLE ACCESS	PRODUCTOS	FULL	544164	1574

Other XML

```
{info}
  info type="db_version"
    12.1.0.2
  info type="parse_schema"
  "ISIS2304C341720"
  info type="dynamic_sampling" note="y"
    2
```



```
select dia, min(consumido) as min , min(idprod) keep (dense_rank first order by consumido desc) as producto from
(select count(pedidoproducto.idproducto) as consumido,pedidoproducto.IDPRODUCTO as idprod,to_char(pedidos.fecha, 'dy') as dia
from pedidos join pedidoproducto
on pedidos.NUMPEDIDO = pedidoproducto.NUMPEDIDO
group by pedidoproducto.IDPRODUCTO, to_char(pedidos.fecha, 'dy')) group by dia;
```

Salida de Script | Resultado de la Consulta | Explicación del Plan

SQL | 0,14 segundos

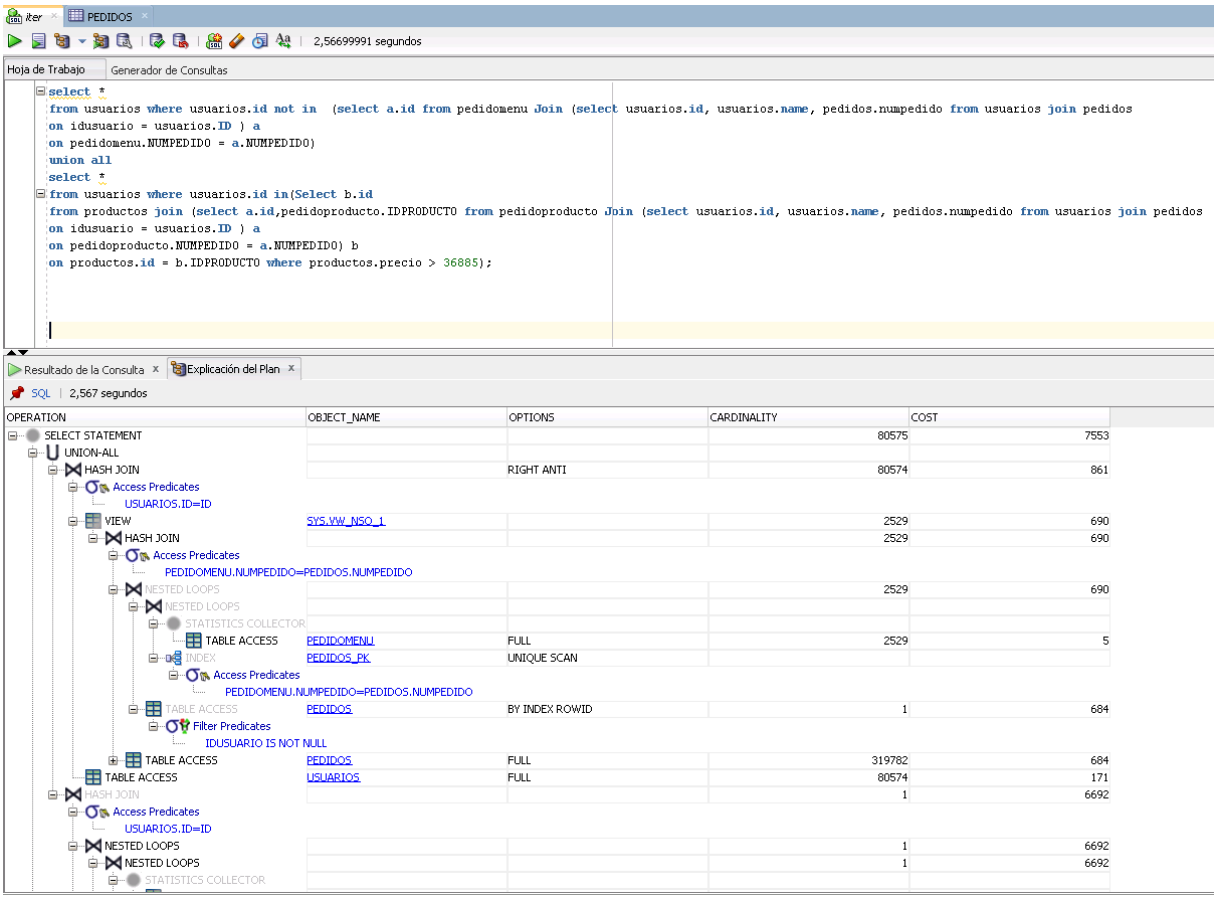
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			298018	1923
SORT		GROUP BY	298018	1923
VIEW			298018	1916
HASH		GROUP BY	298018	1916
HASH JOIN			298018	1908
Access Predicates				
PEDIDOS.NUMPEDIDO=PEDIDOPRODUCTO.NUMPEDIDO				
TABLE ACCESS	PEDIDOS	FULL	319782	684
TABLE ACCESS	PEDIDOPRODUCTO	FULL	298017	171

Other XML

```
{info}
  info type="db_version"
    12.1.0.2
  info type="parse_schema"
  "ISIS2304C341720"
  info type="dynamic_sampling" note="y"
    2
  info type="plan_hash_full"
    2068583066
  info type="plan_hash"
    4270069873
  info type="plan_hash_2"
    2068583066
```

Ambos requerimientos no pudimos modificarlos con ningun index, intentamos varios pero ninguno fue utilizado en el plan sto probablemente a la cantidad de comparaciones que va haciendo.

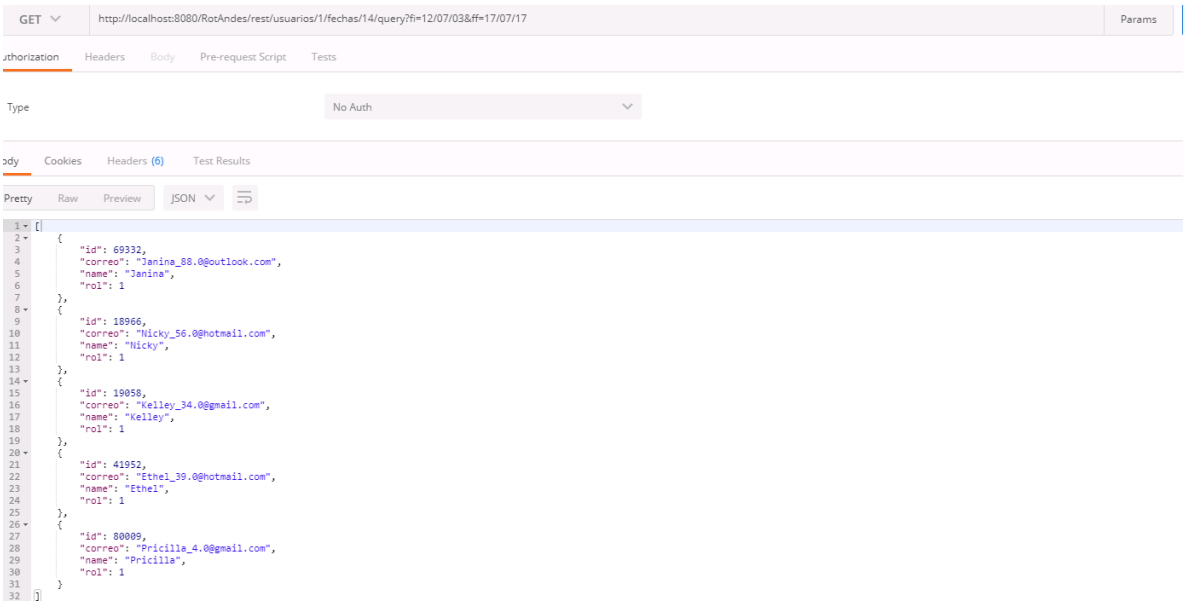
3.4 RFC12



Utilizando un índice de Precio redujo el costo sin embargo es bastante alto.

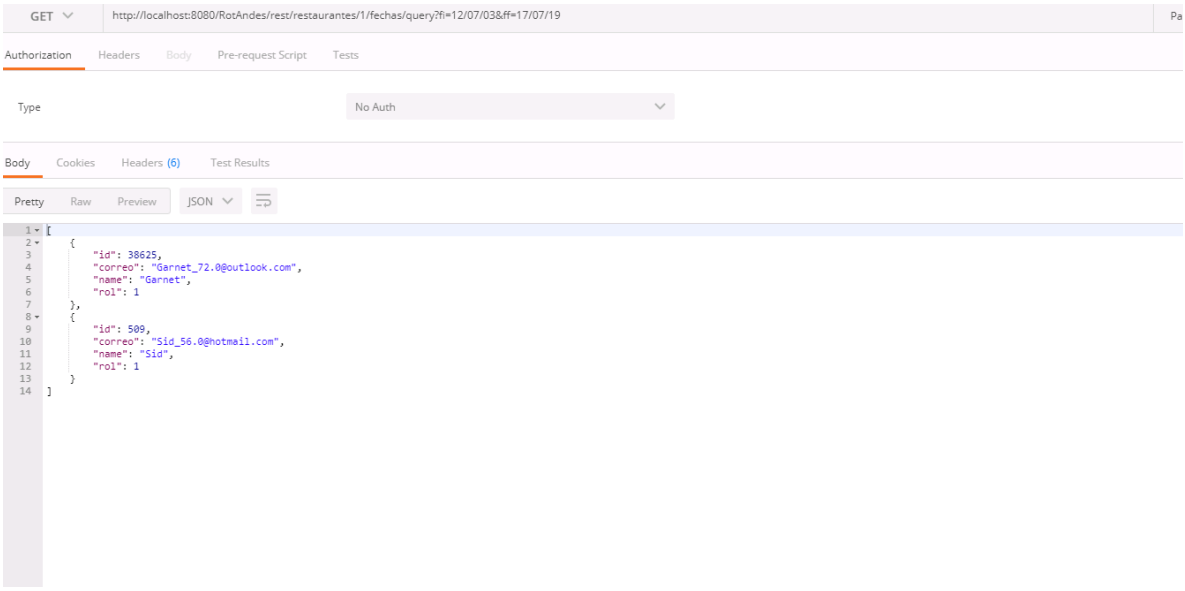
4 PRUEBAS POSTMAN:

RFC9:ADMIN



Aquí utilizamos un id restaurante 14. Y entran en una querry y esto esta dado en las fechas 12/07/03 y 15/07/17

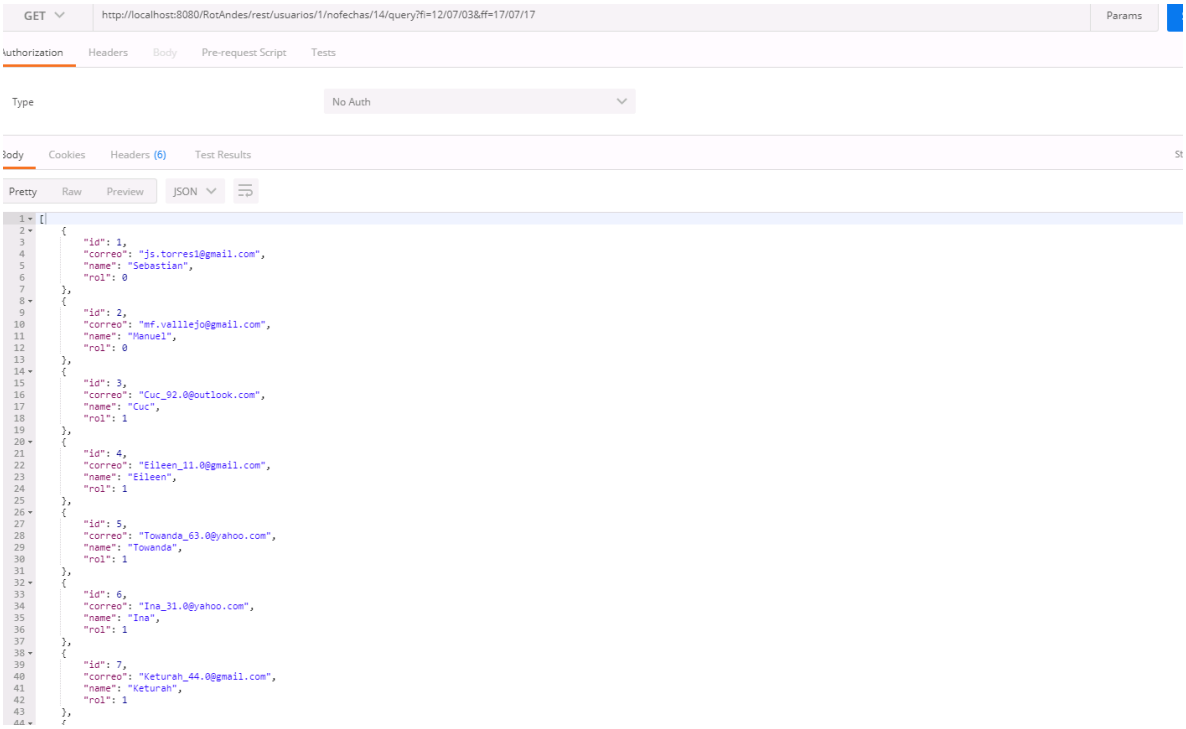
RESTAURANTE:



Aquí utilizamos un id restaurante 1. Y entran en una querry y esto esta dado en las fechas 12/07/03 y 15/07/17

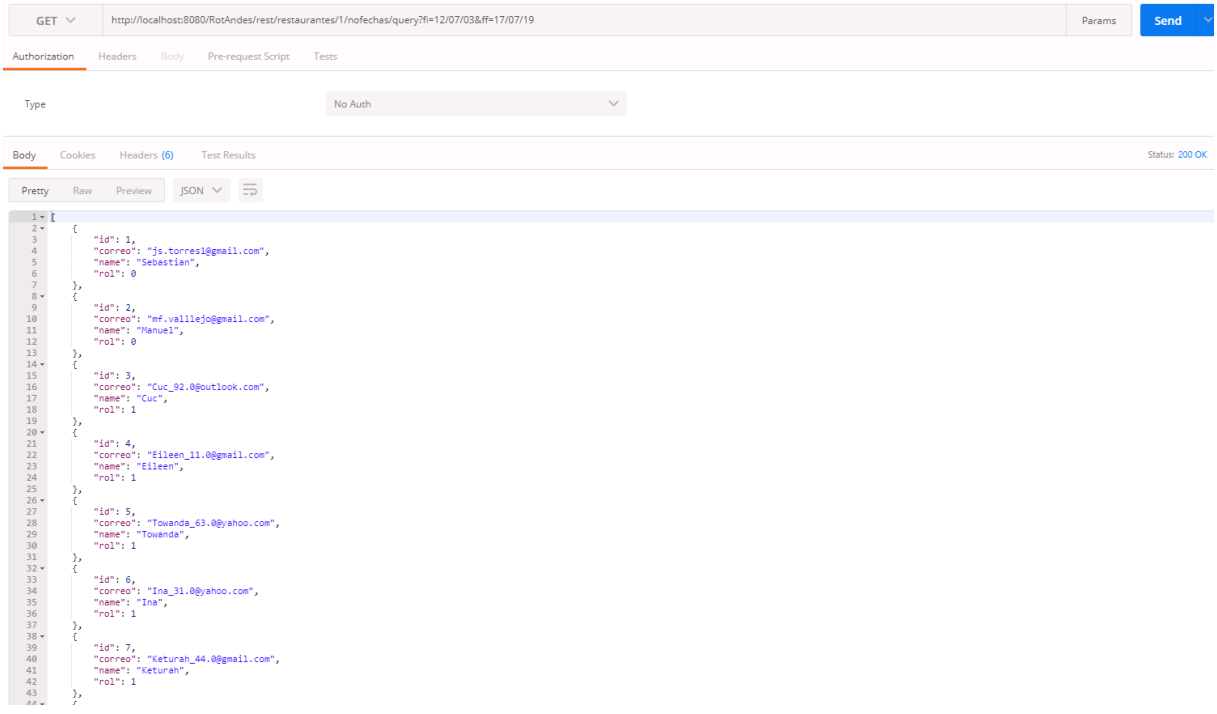
RFC10:

ADMIN:



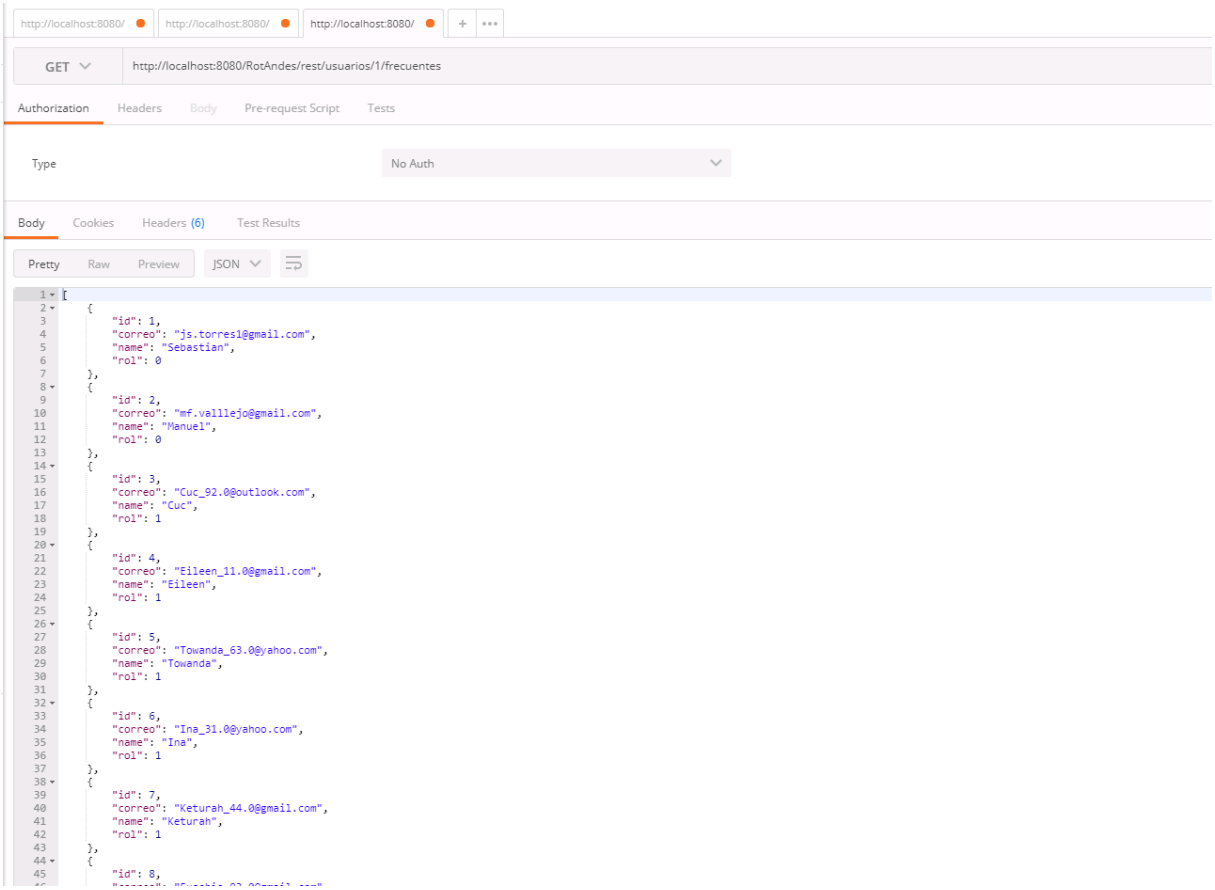
No cambiamos los Parametros.

RESTAURANTE:



No cambiamos parametros.

RFC12:



5 ANALISIS

Debido a que pudimos tener pruebas tanto en postman como en sql podimos observer que las operaciones eran mas lentas sobre el acceso directo a las tablas que sobre el acceso directo a memoria. Cada operacion como tiene que pasar dos entradas a consultar tanto tablas como memoria tarda un poco mas.