

“Iteracion 4”

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

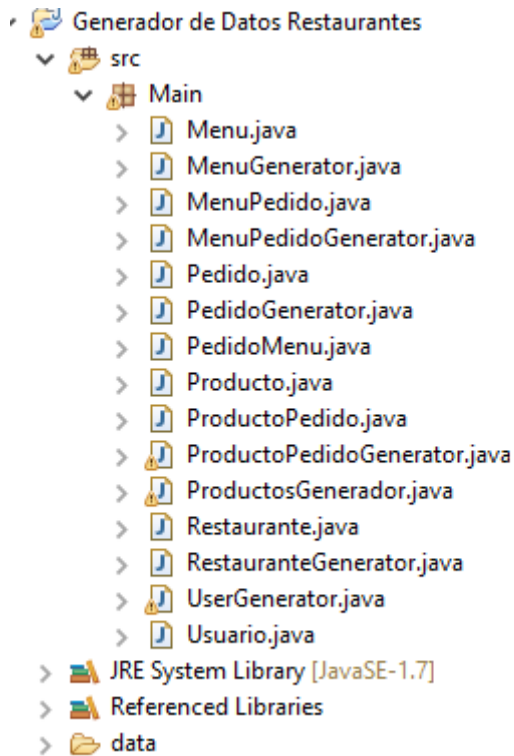
Fecha de presentación: noviembre 21 de 2017

Tabla de contenido

1	Introducción.....	¡Error! Marcador no definido.
2	Manejo de estilos de presentación	¡Error! Marcador no definido.
3	Otros aspectos de manejo de estilos	¡Error! Marcador no definido.
3.1	Manejo de referencias	¡Error! Marcador no definido.
3.2	Estilo de código fuente.....	¡Error! Marcador no definido.
3.3	Numeración de capítulos.....	¡Error! Marcador no definido.
3.4	Manejo de referencias	5
3.5	Enumeraciones y listas.....	¡Error! Marcador no definido.
3.6	Conclusiones.....	¡Error! Marcador no definido.
4	Bibliografía.....	¡Error! Marcador no definido.

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 así pudimos tener una distribución 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;

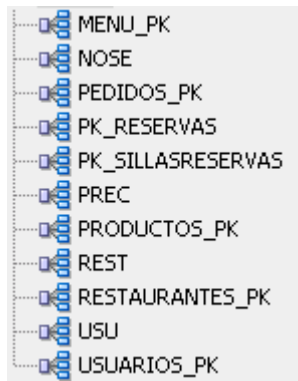
select dia, min(consumido) as min , min(idprod) keep (dense_rank first order by consumido desc) as producto from
 (select count(pedidoproducto.idproducto) as
 consumido,pedidoproducto.IDPRODUCTO as
 idprod,to_char(pedidos.fecha, 'dy') as dia
 from pedidos join pedidoproducto
 on pedidos.NUMPEDIDO = pedidoproducto.NUMPEDIDO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

SQL | 1,844 segundos

Hoja de Trabajo | Generador de Consultas

```

select * from usuarios
where usuarios.id 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);
  
```

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

SQL | 1,844 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	131
HASH JOIN			1	131
Access Predicates				
USUARIOS.ID=IDUSUARIO				
NESTED LOOPS			1	131
NESTED LOOPS			1	131
STATISTICS COLLECTOR				
VIEW	SYS.VW_INSO_1		5	129
HASH		UNIQUE	1	
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

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

3.2 RFC10:

Hoja de Trabajo: Generador de Consultas

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

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

SQL | 1,766 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				300
HASH JOIN		RIGHT ANTI		300
Access Predicates				
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\$JONES_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	INDEX	RANGE SCAN	2	2
Access Predicates				
PRODUCTOS.ID = PEDIDOPRODUCTO.IDPRODUCTO				

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

3.3 RFC11:

Hoja de Trabajo: Generador de Consultas

```
select dia, max(consumido) as maxi, max(idprod) keep (dense_rank first order by consumido desc) as producto from  
(select count(pedido_producto.idproducto) as consumido, pedido_producto.IDPRODUCTO as idprod, to_char(pedidos.fecha, 'dy') as dia  
from pedidos join pedido_producto  
on pedidos.NUMPEDIDO = pedido_producto.NUMPEDIDO  
group by pedido_producto.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				1968
SORT				1968
VIEW		GROUP BY		1961
HASH JOIN		GROUP BY		1961
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"
- TSS12304C341720
- 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\$64AE176")
- USE_HASH(@"SEL\$64AE176" "PEDIDOS"@"SEL\$2")
- LEADING(@"SEL\$64AE176" "PEDIDOPRODUCTO"@"SEL\$2" "PEDIDOS"@"SEL\$2")
- FULL(@"SEL\$64AE176" "PEDIDOS"@"SEL\$2")
- FULL(@"SEL\$64AE176" "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.

Hoja de Trabajo Generador de Consultas

```

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

```

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

SQL 0,266 segundos

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

Other XML (Info)

```

info type="db_version"
  12.1.0.2
info type="parse_schema"
  "TSS2304C341720"
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$64AE176")
  USE_HASH(@"SEL$64AE176" "PEDIDOS"@"SEL$2")
  LEADING(@"SEL$64AE176" "PEDIDOPRODUCTO"@"SEL$2" "PEDIDOS"@"SEL$2")
  FULL(@"SEL$64AE176" "PEDIDOS"@"SEL$2")
  FULL(@"SEL$64AE176" "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
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 x Resultado de la Consulta x Explicación del Plan x

SQL 0,375 segundos

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

Other XML (Info)

```

info type="db_version"
  12.1.0.2
info type="parse_schema"
  "TSS2304C341720"
info type="dynamic_sampling" note="y"
  2

```

```

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

```

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

SQL 0,14 segundos

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

Other XML (Info)

```

info type="db_version"
  12.1.0.2
info type="parse_schema"
  "TSS2304C341720"
info type="dynamic_sampling" note="y"
  2
info type="plan_hash_full"
  2068583066
info type="plan_hash"
  4270068973
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

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL query for RFC12. The bottom pane shows the execution plan for the query, which is a complex nested loop join involving multiple tables and views.

Query:

```

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);
  
```

Execution Plan:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				7553
UNION-ALL				
HASH JOIN		RIGHT ANTI		861
Access Predicates				
VIEW	SYS.VW_NSQ_1			690
HASH JOIN				690
Access Predicates				
NESTED LOOPS				690
NESTED LOOPS				
STATISTICS COLLECTOR				
TABLE ACCESS	PEDIDOMENU	FULL		5
INDEX	PEDIDOS_PK	UNIQUE SCAN		
Access Predicates				
TABLE ACCESS	PEDIDOS	BY INDEX ROWID		684
Filter Predicates				
TABLE ACCESS	PEDIDOS	FULL		684
TABLE ACCESS	USUARIOS	FULL		171
HASH JOIN				6692
Access Predicates				
NESTED LOOPS				6692
NESTED LOOPS				6692
STATISTICS COLLECTOR				

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

4 PRUEBAS POSTMAN:

RFC9:ADMIN

GET ▼ http://localhost:8080/RotAndes/rest/usuarios/1/fechas/14/query?fi=12/07/03&ff=17/07/17 Params

Authorization Headers Body Pre-request Script Tests

Type No Auth ▼

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON ▼ ≡

```
1- [
2-   {
3-     "id": 69332,
4-     "correo": "Janina_88.0@outlook.com",
5-     "name": "Janina",
6-     "rol": 1
7-   },
8-   {
9-     "id": 18966,
10-    "correo": "Nicky_56.0@hotmail.com",
11-    "name": "Nicky",
12-    "rol": 1
13-  },
14-  {
15-    "id": 19058,
16-    "correo": "Kelley_34.0@gmail.com",
17-    "name": "Kelley",
18-    "rol": 1
19-  },
20-  {
21-    "id": 41952,
22-    "correo": "Ethel_39.0@hotmail.com",
23-    "name": "Ethel",
24-    "rol": 1
25-  },
26-  {
27-    "id": 80009,
28-    "correo": "Pricilla_4.0@gmail.com",
29-    "name": "Pricilla",
30-    "rol": 1
31-  }
32- ]
```

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

RESTAURANTE:

GET ▼ http://localhost:8080/RotAndes/rest/restaurantes/1/fechas/query?fi=12/07/03&ff=17/07/19 Pa

Authorization Headers Body Pre-request Script Tests

Type No Auth ▼

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON ▼ ≡

```
1- [
2-   {
3-     "id": 38625,
4-     "correo": "Garnet_72.0@outlook.com",
5-     "name": "Garnet",
6-     "rol": 1
7-   },
8-   {
9-     "id": 509,
10-    "correo": "Sid_56.0@hotmail.com",
11-    "name": "Sid",
12-    "rol": 1
13-  }
14- ]
```

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

RFC10:

ADMIN:

GET http://localhost:8080/RotAndes/rest/usuarios/1/nofechas/14/query?fi=12/07/03&ff=17/07/17 Params

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON

```
1- [
2-   {
3-     "id": 1,
4-     "correo": "js.torres1@gmail.com",
5-     "name": "Sebastian",
6-     "rol": 0
7-   },
8-   {
9-     "id": 2,
10-    "correo": "mf.valliejo@gmail.com",
11-    "name": "Manuel",
12-    "rol": 0
13-  },
14-  {
15-    "id": 3,
16-    "correo": "Cuc_92.0@outlook.com",
17-    "name": "Cuc",
18-    "rol": 1
19-  },
20-  {
21-    "id": 4,
22-    "correo": "Eileen_11.0@gmail.com",
23-    "name": "Eileen",
24-    "rol": 1
25-  },
26-  {
27-    "id": 5,
28-    "correo": "Towanda_63.0@yahoo.com",
29-    "name": "Towanda",
30-    "rol": 1
31-  },
32-  {
33-    "id": 6,
34-    "correo": "Ina_31.0@yahoo.com",
35-    "name": "Ina",
36-    "rol": 1
37-  },
38-  {
39-    "id": 7,
40-    "correo": "Keturah_44.0@gmail.com",
41-    "name": "Keturah",
42-    "rol": 1
43-  },
44- ]
```

No cambiamos los Parametros.

RESTAURANTE:

GET http://localhost:8080/RotAndes/rest/restaurantes/1/nofechas/query?fi=12/07/03&ff=17/07/19 Params Send

Authorization Headers Body Pre-request Script Tests

Type No Auth

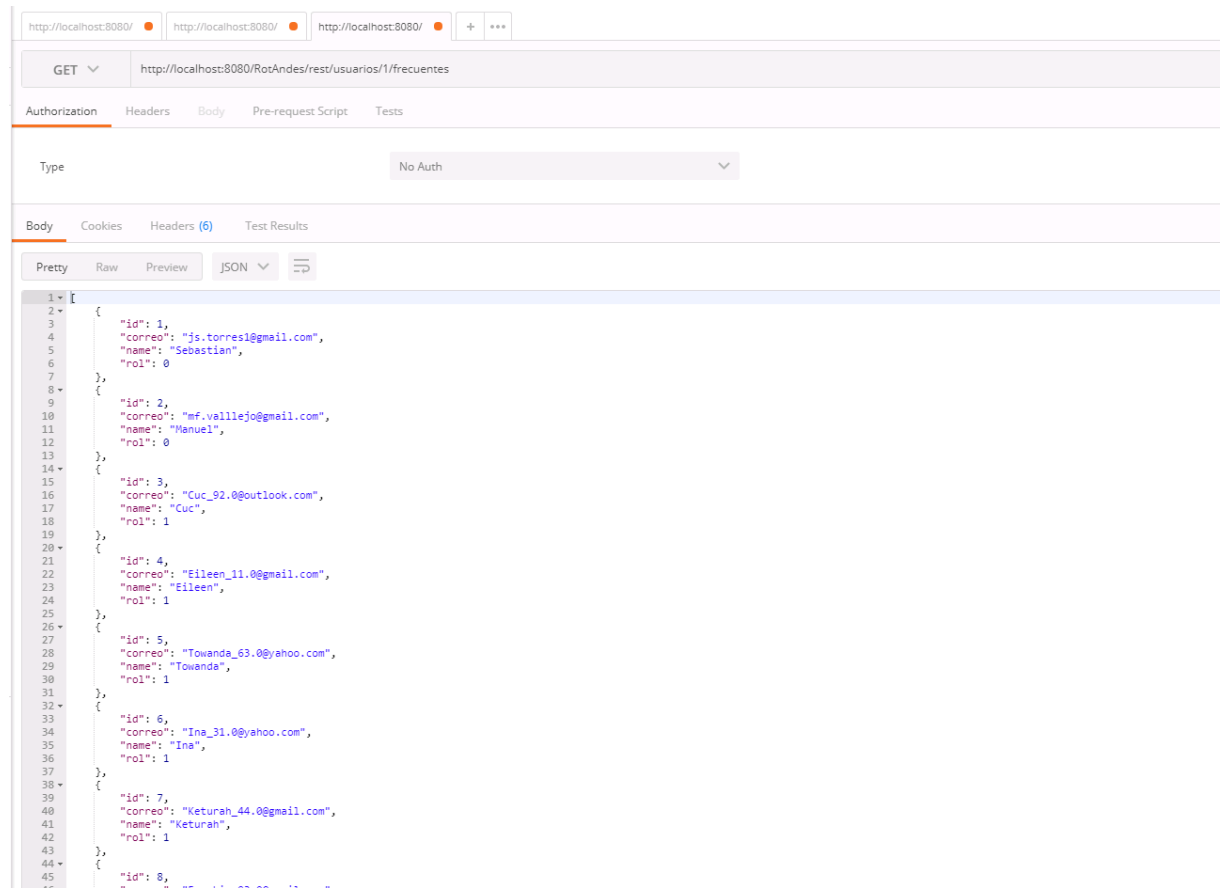
Body Cookies Headers (6) Test Results Status: 200 OK

Pretty Raw Preview JSON

```
1- [
2-   {
3-     "id": 1,
4-     "correo": "js.torres1@gmail.com",
5-     "name": "Sebastian",
6-     "rol": 0
7-   },
8-   {
9-     "id": 2,
10-    "correo": "mf.valliejo@gmail.com",
11-    "name": "Manuel",
12-    "rol": 0
13-  },
14-  {
15-    "id": 3,
16-    "correo": "Cuc_92.0@outlook.com",
17-    "name": "Cuc",
18-    "rol": 1
19-  },
20-  {
21-    "id": 4,
22-    "correo": "Eileen_11.0@gmail.com",
23-    "name": "Eileen",
24-    "rol": 1
25-  },
26-  {
27-    "id": 5,
28-    "correo": "Towanda_63.0@yahoo.com",
29-    "name": "Towanda",
30-    "rol": 1
31-  },
32-  {
33-    "id": 6,
34-    "correo": "Ina_31.0@yahoo.com",
35-    "name": "Ina",
36-    "rol": 1
37-  },
38-  {
39-    "id": 7,
40-    "correo": "Keturah_44.0@gmail.com",
41-    "name": "Keturah",
42-    "rol": 1
43-  },
44- ]
```

No cambiamos parametros.

RFC12:



5 ANALISIS

Debido a que pudimos tener pruebas tanto en postman como en sql podimos observar 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.