“Documentación Iteración 4”

María Alejandra Abril, Juan Daniel Brawn
Documentación
Universidad de los Andes, Bogotá, Colombia
{ma.abril, jd.carrillor}@uniandes.edu.co

Fecha de presentación: Noviembre 21 de 2017

Tabla de contenido

1 Introducción. 1
2 Manejo de estilos de presentación. 2
3 Otros aspectos de manejo de estilos. 3
3.1 Manejo de referencias. 3
3.2 Estilo de código fuente. 3
3.3 Numeración de capítulos. 3
3.4 Manejo de referencias. 3
3.5 Enumeraciones y listas. 3
3.6 Conclusiones. 3
4 Bibliografía. 3

1.Analisis:
En esta iteración se agregaron clases pertinentes como lo son la clase Funcionamiento, la cual es útil para representar la respuesta a la consulta del requerimiento 11.

2. Diseño de la aplicación

--Cambios y análisis
En esta iteración se implementaron cambios importantes que influyen en el desarrollo de la aplicación. Dentro de dichos cambios, se agregaron nuevas clases (VOS) en la aplicación, dentro de dichas clases se agregó la clase Funcionamiento, util para el requerimiento 11. De igual forma en la base de datos se usaron tablas temporales para el desarrollo de dicho requerimiento. La introducción de los nuevos requerimientos de consulta y los no funcionales, afecta de manera directa la aplicación, ya que con dichos requerimientos no funcionales, se debe garantizar la eficiencia de las consultas. Esto implica la introducción de técnicas como lo son los índices y modificación de iteraciones pasadas.

---Diseño físico

1 Selección de índices

Índices existentes:
Tabla PRODUCTO

Conexion | PRODUCTO

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304B011720	PRODUCTO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDPRODUCTO

Tabla PEDIDO:

Conexion | PEDIDO

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304B011720	PEDIDO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDPEDIDO

Tabla USUARIO:

Conexion | USUARIO

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304B011720	USUARIO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDUSUARIO

Tabla TIPO:

Conexion | TIPO

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304B011720	TIPO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDTIPO

Tabla TIPOPRODUCTO:

Conexion | TIPOPRODUCTO

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304B011720	TIPOPRODUCTO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_PRODUCTOTIPO, ID_TIPOPROD

Tabla RESTAURANTE:

Conexion | RESTAURANTE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304B011720	RESTAURANTE_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDRESTAURANTE

Oracle crea los índices anteriores automáticamente en los PK de cada tabla.

Para ambos requerimientos se creó un índice en la tabla PEDIDO de tipo bitmap, de modo que la consulta más lenta (la cual tardaba 0.816), tarda 0.033. Se puso en la columna ID_PRODUCTO debido a que la consulta hace un join de los usuarios con los pedidos y luego con los productos y los restaurantes, para saber un cierto producto que un usuario no haya consumido de algún restaurante en cierta fecha. Entonces, la FK de la tabla PEDIDO es ID_PRODUCTO, la cual logra el join de los productos pedidos con los tipos de producto, que luego hace join con la tabla RESTAURANTE.

2 Análisis

Requerimiento 9:
Consulta 1

```
SELECT IDUSUARIO, NOMBREUSUARIO, ROL, CORREO FROM (WITH AAA AS (SELECT * FROM USUARIO
NATURAL JOIN (SELECT ID_USUARIO AS IDUSUARIO, ID_PRODUCTO AS IDPRODUCTO, FECHA FROM PEDIDO))
SELECT DISTINCT * FROM AAA NATURAL JOIN (SELECT IDPRODUCTO,NOMBRE AS NOMBREPRODUCTO, ID_RESTAURANTE FROM PRODUCTO)
NATURAL JOIN (SELECT ID_PRODUCTOTIPO AS IDPRODUCTO,ID_TIPOPROD FROM TIPOPRODUCTO)
NATURAL JOIN (SELECT IDTIPO AS ID_TIPOPRODUCTO, NOMBRETIPO FROM TIPO)
NATURAL JOIN (SELECT IDRESTAURANTE AS ID_RESTAURANTE, NOMBRERESTAURANTE FROM RESTAURANTE)
WHERE CORREO = 'ap@gmail.com' AND NOMBREPRODUCTO = 'Anillos de cebolla'
AND NOMBRETIPO = 'vegetariano' AND NOMBRERESTAURANTE = 'El Cal' AND FECHA >= '01-JAN-00'
AND FECHA <= '01-JAN-03');
```

Script Output | Explain Plan | Query Result

SQL | All Rows Fetched: 1 in 0.063 seconds

	IDUSUARIO	NOMBREUSUARIO	ROL	CORREO
1	2	Andres	General	ap@gmail.com

Parámetros: correo de un usuario, nombre de un producto, tipo, restaurante y rango de fechas
Tamaño: devuelve una tupla
Plan de ejecución:

SQL | 0.572 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	11
VIEW			1	11
HASH		UNIQUE	1	11
FILTER				
Filter Predicates				
TO_DATE('01-JAN-03')>=TO_DATE('01-JAN-00')				
MERGE JOIN		CARTESIAN	1	10
NESTED LOOPS			1	7
NESTED LOOPS			1	7
HASH JOIN			1	6
Access Predicates				
ID_PRODUCTO=IDPRODUCTO				
NESTED LOOPS		SEMI	1	3
NESTED LOOPS			1	2
INDEX	TIPOPRODUCTO_PK	FULL SCAN	1	1
TABLE ACCE	PRODUCTO	BY INDEX ROWID	1	1
Filter Predicates				
NOMBRE='Anillos de cebolla'				
INDEX	PRODUCTO_PK	UNIQUE SCAN	1	0
Access Predicates				
IDPRODUCTO=ID_PRODUCTOTIPO				
TABLE ACCESS	RESTAURANTE	BY INDEX ROWID	1	1
Filter Predicates				
NOMBRERESTAURANTE='El Cal'				
INDEX	RESTAURANTE_PK	UNIQUE SCAN	1	0

SQL | 0.572 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
INDEX	PRODUCTO_PK	UNIQUE SCAN	1	0
Access Predicates				
IDPRODUCTO=ID_PRODUCTOTIPO				
TABLE ACCESS	RESTAURANTE	BY INDEX ROWID	1	1
Filter Predicates				
NOMBRERESTAURANTE='El Cal'				
INDEX	RESTAURANTE_PK	UNIQUE SCAN	1	0
Access Predicates				
ID_RESTAURANTE=IDRESTAURANTE				
TABLE ACCESS	PEDIDO	FULL	9	3
Filter Predicates				
AND				
FECHA<='01-JAN-03'				
FECHA>='01-JAN-00'				
INDEX	USUARIO_PK	UNIQUE SCAN	1	0
Access Predicates				
USUARIO.IDUSUARIO=ID_USUARIO				
TABLE ACCESS	USUARIO	BY INDEX ROWID	1	1
Filter Predicates				
USUARIO.CORREO='ap@gmail.com'				
BUFFER		SORT	1	9
TABLE ACCESS	TIPO	FULL	1	3
Filter Predicates				
NOMBRETIPO='vegetariano'				

Debido a todos los parametros de busqueda, se debe acceder a cada tabla involucrada (USUARIO, PEDIDO, RESTAURANTE, TIPOPRODUCTO, TIPO, y PRODUCTO). Gracias al índice creado este acceso a las tablas y las búsquedas de los productos se simplifica. Luego, se hacen hash join entre los resultados de estas tablas y al final un merge join que devuelve la tupla encontrada.

Requerimiento 10:
Tipo consulta 1, sin índices nuevos

Worksheet | Query Builder

```
SELECT * FROM USUARIO MINUS (SELECT IDUSUARIO, NOMBREUSUARIO, ROL, CORREO FROM (WITH AAA AS (SELECT * FROM USUARIO
NATURAL JOIN (SELECT ID_USUARIO AS IDUSUARIO, ID_PRODUCTO AS IDPRODUCTO, FECHA FROM PEDIDO))
SELECT DISTINCT * FROM AAA NATURAL JOIN (SELECT IDPRODUCTO,NOMBRE AS NOMBREPRODUCTO, ID_RESTAURANTE FROM PRODUCTO)
NATURAL JOIN (SELECT ID_PRODUCTOTIPO AS IDPRODUCTO,ID_TIPOPROD FROM TIPOPRODUCTO)
NATURAL JOIN (SELECT IDTIPO AS ID_TIPOPRODUCTO, NOMBRETIPO FROM TIPO)
NATURAL JOIN (SELECT IDRESTAURANTE AS ID_RESTAURANTE, NOMBRERESTAURANTE FROM RESTAURANTE)
WHERE CORREO = 'ap@gmail.com' AND NOMBREPRODUCTO = 'Anillos de cebolla'
AND NOMBRETIPO = 'vegetariano' AND NOMBRERESTAURANTE = 'El Cal' AND FECHA >= '01-JAN-00'
AND FECHA <= '01-JAN-03'));
```

Script Output | Explain Plan | Query...

SQL | Fetched 50 rows in 0.71 seconds

	IDUSUARIO	NOMBREUSUARIO	ROL	CORREO
1	1	Maria	Cliente	ma@hotmail.com
2	3	Diego	Administrador	dr@yahoo.com
3	4	Laura	Cliente	llm@hotmail.com
4	5	John	Administrador	john@gmail.com
5	6	Camilo	Cliente	john@gmail.com

Tipo consulta 1, con índice bitmap en ID_PRODUCTO en la tabla PEDIDOS
Parámetros: correo del usuario, nombre de un producto, tipo de producto, restaurante, rango de fechas

```sql
SELECT * FROM USUARIO MINUS (SELECT IDUSUARIO, NOMBREUSUARIO, ROL, CORREO FROM (WITH AAA AS (SELECT * FROM USUARIO
NATURAL JOIN (SELECT ID_USUARIO AS IDUSUARIO, ID_PRODUCTO AS IDPRODUCTO, FECHA FROM PEDIDO))
SELECT DISTINCT * FROM AAA NATURAL JOIN (SELECT IDPRODUCTO,NOMBRE AS NOMBREPRODUCTO, ID_RESTAURANTE FROM PRODUCTO)
NATURAL JOIN (SELECT ID_PRODUCTOTIPO AS IDPRODUCTO,ID_TIPOPROD FROM TIPOPRODUCTO)
NATURAL JOIN (SELECT IDTIPO AS ID_TIPOPRODUCTO, NOMBRETIPO FROM TIPO)
NATURAL JOIN (SELECT IDRESTAURANTE AS ID_RESTAURANTE, NOMBRERESTAURANTE FROM RESTAURANTE)
WHERE CORREO = 'ap@gmail.com' AND NOMBREPRODUCTO = 'Anillos de cebolla'
AND NOMBRETIPO = 'vegetariano' AND NOMBRERESTAURANTE = 'El Cal' AND FECHA >= '01-JAN-00'
AND FECHA <= '01-JAN-03'));
```

Script Output | Explain Plan | Query Result

SQL | Fetched 50 rows in 0.05 seconds

	IDUSUARIO	NOMBREUSUARIO	ROL	CORREO
1	1	Maria	Cliente	ma@hotmail.com
2	3	Diego	Administrador	dr@yahoo.com
3	4	Laura	Cliente	llm@hotmail.com
4	5	John	Administrador	john@gmail.com
5	6	Camilo	Cliente	john@gmail.com

Plan de ejecución:

SQL | 0.221 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			6	25
MINUS				
SORT		UNIQUE	6	14
TABLE ACCESS	USUARIO	FULL	6	13
SORT		UNIQUE	1	11
FILTER				
Filter Predicates				
TO_DATE('01-JAN-03')>=TO_DATE('01-JAN-00')				
HASH JOIN			1	10
Access Predicates				
USUARIO.IDUSUARIO=ID_USUARIO				
NESTED LOOPS			1	10
NESTED LOOPS			1	10
STATISTICS COLLECTOR				
HASH JOIN			1	9
Access Predicates				
ID_PRODUCTO=IDPRODUCTO				
NESTED LOOPS		SEMI	1	6
HASH JOIN			1	5
Access Predicates				
IDPRODUCTO=ID_PRODUCTOTIPO				
NESTED			1	5
STA				
		CARTESIAN	1	4
	TIPO	FULL	1	3

SQL | 0.221 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
	TIPO	FULL	1	3
Filter Predicates				
NOMBRETIPO='vegetariano'				
		SORT	1	1
	TIPOPRODUCTO_PK	FULL SCAN	1	1
TABI	PRODUCTO	BY INDEX ROWID	1	1
Filter Predicates				
NOMBRE='Anillos de cebolla'				
	PRODUCTO_PK	UNIQUE SCAN	1	0
Access Predicates				
IDPRODUCTO=ID_PRODUCTOTIPO				
TABLE A	PRODUCTO	FULL	1	1
Filter Predicates				
NOMBRE='Anillos de cebolla'				
TABLE ACCE	RESTAURANTE	BY INDEX ROWID	1	1
Filter Predicates				
NOMBRERESTAURANTE='El Cal'				
INDEX	RESTAURANTE_PK	UNIQUE SCAN	1	0
Access Predicates				
ID_RESTAURANTE=IDRESTAURANTE				
TABLE ACCESS	PEDIDO	FULL	9	3
Filter Predicates				
AND				
FECHA<='01-JAN-03'				
FECHA>='01-JAN-00'				

SQL | 0.221 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
NOMBRERESTAURANTE='El Cal'				
INDEX	RESTAURANTE_PK	UNIQUE SCAN	1	0
Access Predicates				
ID_RESTAURANTE=IDRESTAURANTE				
TABLE ACCESS	PEDIDO	FULL	9	3
Filter Predicates				
AND				
FECHA<='01-JAN-03'				
FECHA>='01-JAN-00'				
INDEX	USUARIO_PK	UNIQUE SCAN	1	0
Access Predicates				
USUARIO.IDUSUARIO=ID_USUARIO				
TABLE ACCESS	USUARIO	BY INDEX ROWID	1	1
Filter Predicates				
USUARIO.CORREO='ap@gmail.com'				
TABLE ACCESS	USUARIO	FULL	1	1
Filter Predicates				
USUARIO.CORREO='ap@gmail.com'				
Other XML				

Tipo consulta 2, con índice bitmap en ID_PRODUCTO en la tabla PEDIDOS

```
SELECT * FROM USUARIO MINUS (SELECT IDUSUARIO, NOMBREUSUARIO, ROL, CORREO FROM (WITH AAA AS (SELECT * FROM USUARIO
NATURAL JOIN (SELECT ID_USUARIO AS IDUSUARIO, ID_PRODUCTO AS IDPRODUCTO, FECHA FROM PEDIDO))
SELECT DISTINCT * FROM AAA NATURAL JOIN (SELECT IDPRODUCTO,NOMBRE AS NOMBREPRODUCTO, ID_RESTAURANTE FROM PRODUCTO)
NATURAL JOIN (SELECT ID_PRODUCTOTIPO AS IDPRODUCTO,ID_TIPOPROD FROM TIPOPRODUCTO)
NATURAL JOIN (SELECT IDTIPO AS ID_TIPOPRODUCTO, NOMBRETIPO FROM TIPO)
NATURAL JOIN (SELECT IDRESTAURANTE AS ID_RESTAURANTE, NOMBRERESTAURANTE FROM RESTAURANTE)
WHERE NOMBREUSUARIO = 'Camilo' AND FECHA >= '01-JAN-00'));
```

Script Output | Explain Plan | Query Result

SQL | Fetched 50 rows in 0.048 seconds

	IDUSUARIO	NOMBREUSUARIO	ROL	CORREO
1	1	Maria	Cliente	ma@hotmail.com
2	2	Andres	General	ap@gmail.com
3	3	Diego	Administrador	dr@yahoo.com
4	4	Laura	Cliente	llm@hotmail.com
5	5	John	Administrador	john@gmail.com

Parámetros: nombre del usuario y rango de fechas
Tamaño: gran cantidad de datos
Tiempo: 0.129s
Plan de ejecución:

SQL | 0.129 seconds

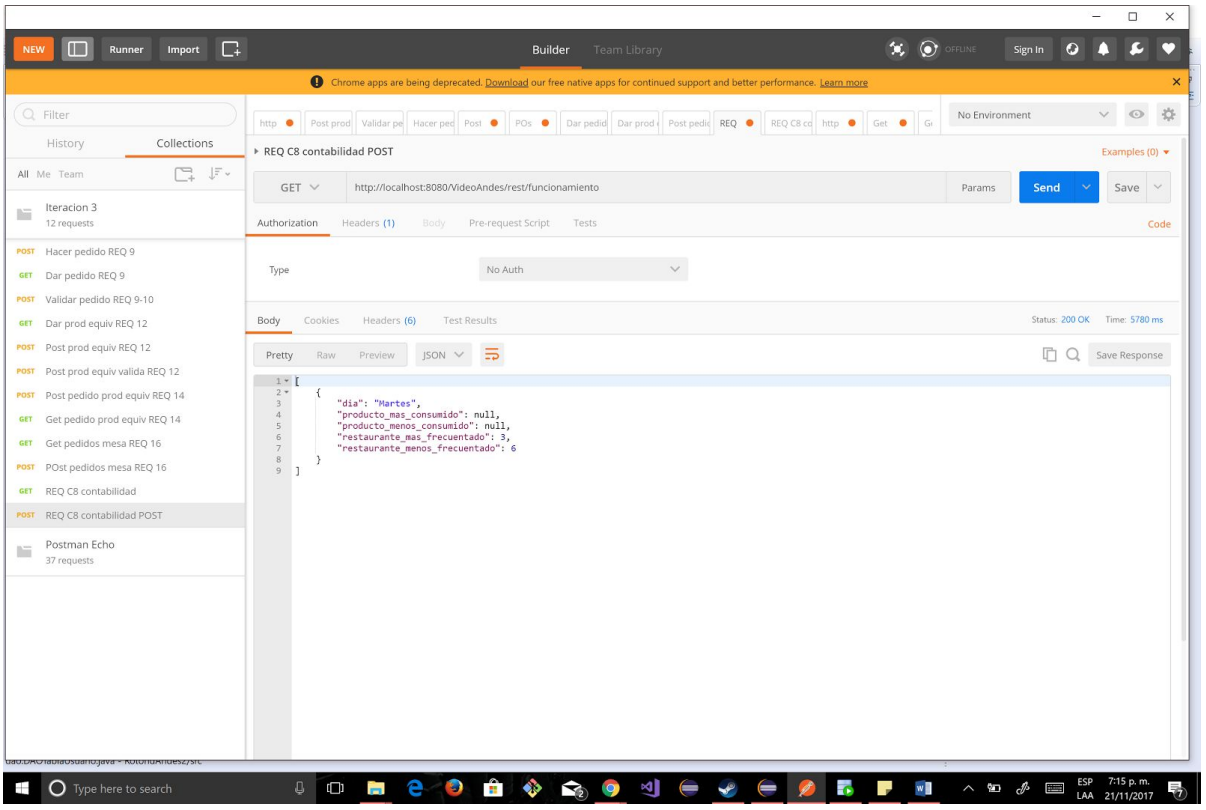
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			6	24
MINUS				
SORT		UNIQUE	6	14
TABLE ACCESS	USUARIO	FULL	6	13
SORT		UNIQUE	2	10
MERGE JOIN		CARTESIAN	2	9
NESTED LOOPS		SEMI	1	8
HASH JOIN			1	8
Access Predicates				
USUARIO.IDUSUARIO=ID_USUARIO				
NESTED LOOPS			1	8
STATISTICS COLLECTOR				
HASH JOIN			3	5
Access Predicates				
ID_PRODUCTO=IDPRODUCTO				
HASH JOIN			1	2
Access Predicates				
IDPRODUCTO=ID_PRODUCTOTIPO				
NESTED LOC			1	2
NESTED			1	2
STA				
	TIPOPRODUCTO_PK	FULL SCAN	1	1
INDE	PRODUCTO_PK	UNIQUE SCAN	1	0
Access Predicates				
IDPRODUCTO=ID_PRODUCTOTIPO				

SQL | 0.129 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
Access Predicates				
IDPRODUCTO=ID_PRODUCTOTIPO				
TABLE A	PRODUCTO	BY INDEX ROWID	1	1
TABLE ACC	PRODUCTO	FULL	1	1
TABLE ACCESS	PEDIDO	FULL	18	3
Filter Predicates				
FECHA>='01-JAN-00'				
TABLE ACCESS	USUARIO	BY INDEX ROWID	1	1
Filter Predicates				
USUARIO.NOMBREUSUARIO='Camilo'				
INDEX	USUARIO_PK	UNIQUE SCAN	1	0
Access Predicates				
USUARIO.IDUSUARIO=ID_USUARIO				
TABLE ACCESS	USUARIO	FULL	1	1
Filter Predicates				
USUARIO.NOMBREUSUARIO='Camilo'				
INDEX	RESTAURANTE_PK	UNIQUE SCAN	6	0
Access Predicates				
ID_RESTAURANTE=IDRESTAURANTE				
BUFFER		SORT	3	9
INDEX	TIPO_PK	FULL SCAN	3	1
Other XML				
{info}				

En este plan de ejecución se ve que se acceden a todas las tablas que se necesitan para hacer la consulta, llegando con facilidad al resultado gracias a los índices en las PK y el índice en el id del producto creado en la tabla PEDIDO. Cada uno genera un costo, pero mucho menor que si no hubieran los índices, y luego se hace un hash join entre las tablas.

Requerimiento 11
Servicio REST:



Sentencias SQL

Este requerimiento implmenta el uso de sentencias sql y el de metodos en java para lograr el correcto funcionamiento.

Indices de Oracle

SQL1
SELECT * FROM (PEDIDOPRODUCTO INNER JOIN PEDIDO ON PEDIDOPRODUCTO.ID_PEDIDO=PEDIDO.IDPEDIDO);

```
SELECT * FROM (PEDIDOPRODUCTO INNER JOIN PEDIDO ON PEDIDOPRODUCTO.ID_PEDIDO=PEDIDO.IDPEDIDO);
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 19 en 0,162 segundos

	ID_PRODUCTO	ID_PEDIDO	IDPEDIDO	FECHA	ID_USUARIO	ID_PRODUCTO_1	ID_MENU
1	7	3	3	09/09/00	3	2	3
2	6	3	3	09/09/00	3	2	3
3	8	4	4	09/09/17	6	4	(null)
4	7	4	4	09/09/17	6	4	(null)
5	2	4	4	09/09/17	6	4	(null)
6	2	5	5	09/09/17	6	3	(null)
7	5	7	7	05/08/00	2	6	8
8	4	16	16	09/09/00	4	4	4
9	4	18	18	09/09/00	4	4	4
10	8	21	21	09/01/05	4	7	6
11	15	2	2	06/11/17	4	7	5
12	6	8	8	09/09/00	6	3	8
13	2	10	10	09/09/00	3	2	4
14	2	11	11	27/10/17	2	3	5

Plan

```
SELECT * FROM (PEDIDOPRODUCTO INNER JOIN PEDIDO ON PEDIDOPRODUCTO.ID_PEDIDO=PEDIDO.IDPEDIDO);
```

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

SQL | 0,061 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			19	4
HASH JOIN			19	4
Access Predicates				
PEDIDOPRODUCTO.ID_PEDIDO=PEDIDO.IDPEDIDO				
NESTED LOOPS			19	4
NESTED LOOPS				
STATISTICS COLLECTOR				
INDEX	PEDIDOPRODUCTO_PK	FULL SCAN	19	1
INDEX	PEDIDO_PK	UNIQUE SCAN		
Access Predicates				
PEDIDOPRODUCTO.ID_PEDIDO=PEDIDO.IDPEDIDO				
TABLE ACCESS	PEDIDO	BY INDEX ROWID	1	3
TABLE ACCESS	PEDIDO	FULL	20	3
Other XML				
{info}				
info type="db_version"				
12.1.0.2				

Retorna los productos que fueron pedidos junto con la información de dicho pedido. En esta consulta se puede evidenciar que los costos se ven afectados por el indice por defecto de

oracle PedidoProductoPK esto se debe a que con el uso de este indice, se reduce el costo a 4. No obstante el indice genera un costo.

SQL2
SELECT *FROM((SELECT * FROM (PRODUCTO INNER JOIN PEDIDOPRODUCTO ON PRODUCTO.IDPRODUCTO=PEDIDOPRODUCTO.ID_PRODUCTO))s INNER JOIN PEDIDO ON s.ID_PEDIDO=PEDIDO.IDPEDIDO);

```
SELECT *FROM((SELECT * FROM (PRODUCTO INNER JOIN PEDIDOPRODUCTO ON PRODUCTO.IDPRODUCTO=PEDIDOPRODUCTO.ID_PRODUCTO))s INNER JOIN PEDIDO ON s.ID_PEDIDO=PEDIDO.IDI
```

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

SQL | Todas las Filas Recuperadas: 18 en 0,207 segundos

	IDPRODUCTO	NOMBRE	COSTOPRODUCCION	DESCRIPCION	TRADUCCION	TIEMPO	PRECIO	CATEGORI
1	7	Limonada	4	limon con agua y hielo	lemon with ice and water	2	9	Bebida
2	6	Anillos de cebolla	6	Cebolla en forma de anillos	Onion rings	5	10	Acompañam
3	8	Batido fresa/naranja	40	batido de fresa y naranja	orange/strawberry smoothie	30	90	Bebida
4	7	Limonada	4	limon con agua y hielo	lemon with ice and water	2	9	Bebida
5	2	Hamburguesa	10	Tomate, lechuga, queso, salsas	Tomatoe, lettuce, cheese, sauces	10	20	Plato fue
6	2	Hamburguesa	10	Tomate, lechuga, queso, salsas	Tomatoe, lettuce, cheese, sauces	10	20	Plato fue
7	5	Papas fritas	6	Papas	French fries	5	10	Acompañam
8	4	Brownie con helado	5	Brownie con una bola de helado de vainilla	Brownie with a scoop of vanilla ice cream	5	15	Postre
9	4	Brownie con helado	5	Brownie con una bola de helado de vainilla	Brownie with a scoop of vanilla ice cream	5	15	Postre
10	8	Batido fresa/naranja	40	batido de fresa y naranja	orange/strawberry smoothie	30	90	Bebida
11	6	Anillos de cebolla	6	Cebolla en forma de anillos	Onion rings	5	10	Acompañam
12	2	Hamburguesa	10	Tomate, lechuga, queso, salsas	Tomatoe, lettuce, cheese, sauces	10	20	Plato fue
13	2	Hamburguesa	10	Tomate, lechuga, queso, salsas	Tomatoe, lettuce, cheese, sauces	10	20	Plato fue
14	6	Anillos de cebolla	6	Cebolla en forma de anillos	Onion rings	5	10	Acompañam

Plan

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

SQL | 0,06 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			19	8
HASH JOIN			19	8
Access Predicates				
PEDIDOPRODUCTO.ID_PEDIDO=PEDIDO.IDPEDIDO				
NESTED LOOPS			19	8
NESTED LOOPS				
STATISTICS COLLECTOR				
MERGE JOIN			19	5
TABLE ACCESS	PRODUCTO	BY INDEX ROWID	8	3
INDEX	PRODUCTO_PK	FULL SCAN	8	1
SORT		JOIN	19	2
Access Predicates				
PRODUCTO.IDPRODUCTO=PEDIDOPRODUCTO.ID_PRODUCTO				
Filter Predicates				
PRODUCTO.IDPRODUCTO=PEDIDOPRODUCTO.ID_PRODUCTO				
INDEX	PEDIDOPRODUCTO_PK	FULL SCAN	19	1
INDEX	PEDIDO_PK	UNIQUE SCAN		

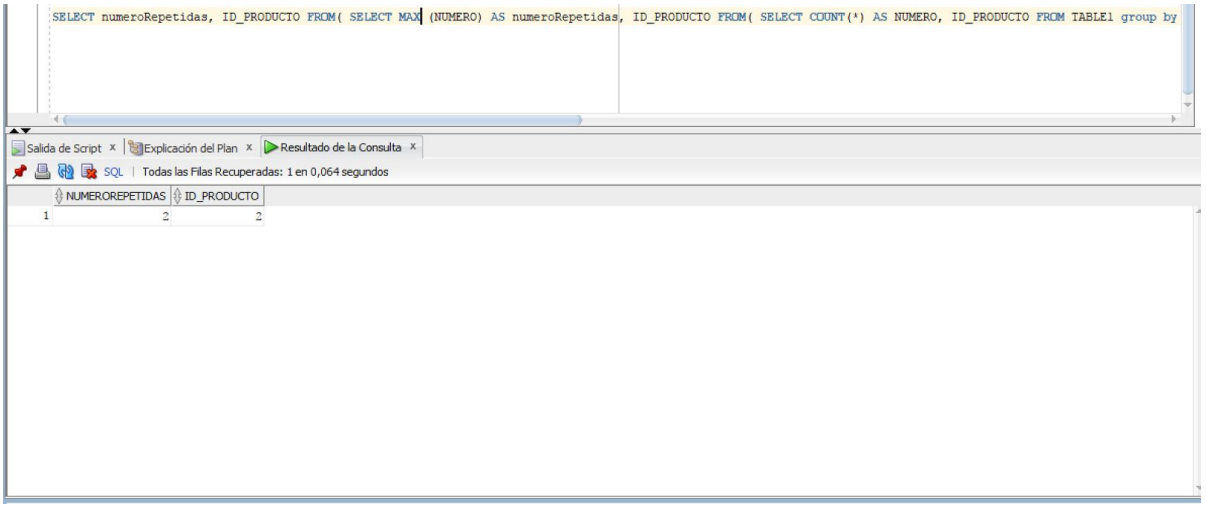
En este requerimiento se obtienen los restaurantes en los cuales se pidio algo en una fecha especifica. Oracle proporciona el indice ProductoPK en la columna IDPRODUCTO lo cual considero que es una movida correcta ya que con dicho indice se reduce el costo de consulta de dicho requerimiento ya que el join se hace con dicho atributo.

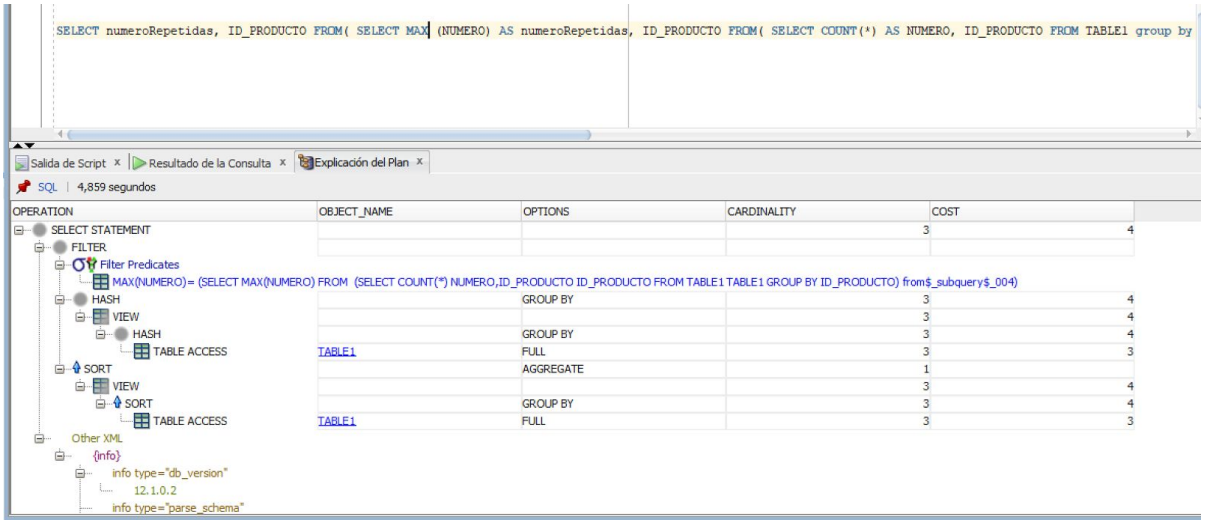
SQL3
CREATE TABLE TABLE1(IDPEDIDO NUMBER(*, 0) NOT NULL , ID_PRODUCTO NUMBER);

INSERT INTO TABLE1 VALUES(1,2);
INSERT INTO TABLE1 VALUES(1,2);
INSERT INTO TABLE1 VALUES(1,1);

SELECT numeroRepetidas, ID_PRODUCTO FROM(SELECT MAX (NUMERO) AS numeroRepetidas, ID_PRODUCTO FROM(SELECT COUNT(*) AS NUMERO, ID_PRODUCTO FROM TABLE1 group by ID_PRODUCTO) group by ID_PRODUCTO) WHERE numeroRepetidas = (SELECT MAX (NUMERO) AS numeroRepetidas FROM(SELECT COUNT(*) AS NUMERO, ID_PRODUCTO FROM TABLE1 group by ID_PRODUCTO));

(Estas sentencias se hacen cuando se ejecutan el servicio con sus datos)

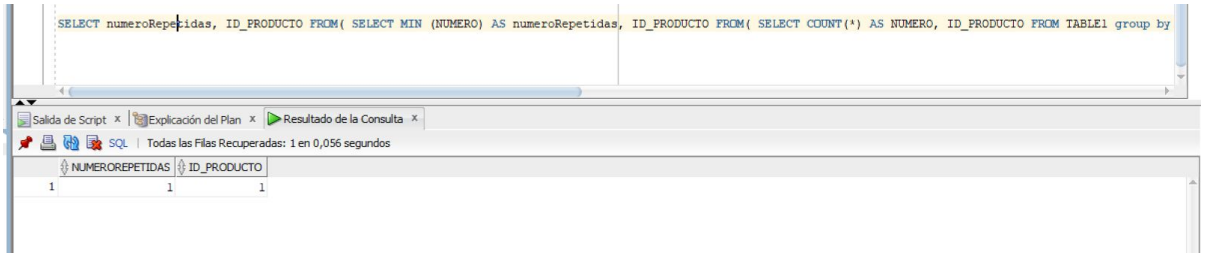


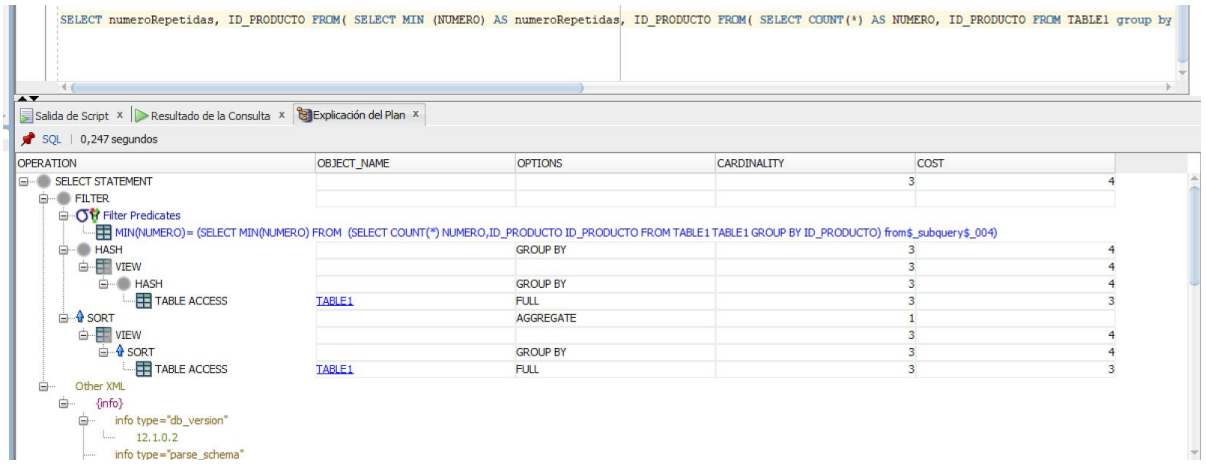


En esta consulta retorna el producto mas pedido en un dia especifico. El costo de esta consulta se ve reducido ya que es complementado por una función en java que agrupa los productos pedidos por dia, es por esto que no hay necesidad de hacerlo en una sentencia. Es por esto que gracias a dicha distribución de tareas se reduce el costo a 4. No obstante se hacen group by en esta sentencia agregando costo a la consulta.

SELECT numeroRepetidas, ID_PRODUCTO FROM(SELECT MIN (NUMERO) AS numeroRepetidas, ID_PRODUCTO FROM(SELECT COUNT(*) AS NUMERO, ID_PRODUCTO FROM TABLE1 group by ID_PRODUCTO) group by ID_PRODUCTO) WHERE numeroRepetidas = (SELECT MIN (NUMERO) AS numeroRepetidas FROM(

SELECT COUNT(*) AS NUMERO, ID_PRODUCTO FROM TABLE1 group by ID_PRODUCTO));





Esta función retorna el producto menos pedido en un dia especifico. El costo de esta consulta se ve reducido ya que es complementado por una función en java que agrupa los productos pedidos por dia, es por esto que no hay necesidad de hacerlo en una sentencia. Es por esto que gracias a dicha distribución de tareas se reduce el costo a 4. No obstante se hacen group by en esta sentencia agregando costo a la consulta.

Estas dos consultas se realizan por medio de una tabla temporal que se crea y borra una vez se termina de usar.

SQL 4
CREATE TABLE TABLE2(FECHA DATE, ID_RESTAURANTE NUMBER);
INSERT INTO TABLE2 VALUES('01/01/17',2);
INSERT INTO TABLE2 VALUES('01/02/17',2);
INSERT INTO TABLE2 VALUES('01/03/17',1);

SELECT numeroRepetidas, ID_RESTAURANTE FROM(SELECT MAX (NUMERO) AS numeroRepetidas, ID_RESTAURANTE FROM(SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TABLE2 group by ID_RESTAURANTE) group by ID_RESTAURANTE) WHERE numeroRepetidas = (SELECT MAX (NUMERO) AS numeroRepetidas FROM(SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TABLE2 group by ID_RESTAURANTE));

```
SELECT numeroRepetidas, ID_RESTAURANTE FROM( SELECT MAX (NUMERO) AS numeroRepetidas, ID_RESTAURANTE FROM( SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TABLE2
```

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

SQL | Todas las Filas Recuperadas: 1 en 0,031 segundos

	NUMEROREPETIDAS	ID_RESTAURANTE
1	2	2

```
SELECT numeroRepetidas, ID_RESTAURANTE FROM( SELECT MAX (NUMERO) AS numeroRepetidas, ID_RESTAURANTE FROM( SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TABL
```

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

SQL | 3,636 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	4
FILTER				
Filter Predicates				
MAX(NUMERO)= (SELECT MAX(NUMERO) FROM (SELECT COUNT(*) NUMERO,ID_RESTAURANTE ID_RESTAURANTE FROM TABLE2 TABLE2 GROUP BY ID_RESTAURANTE) from$_subquery$_004)				
HASH		GROUP BY	3	4
VIEW			3	4
HASH		GROUP BY	3	4
TABLE ACCESS	TABLE2	FULL	3	3
SORT		AGGREGATE	1	
VIEW			3	4
SORT		GROUP BY	3	4
TABLE ACCESS	TABLE2	FULL	3	3
Other XML				
{info}				
info type="db_version"				
12.1.0.2				
info type="parse_schema"				

En este requerimiento se retorna el restaurante más frecuentando en un dia. al igual que en la anterior se implementa una tabla temporal para poner los datos de los restaurantes frecuentados por dia. Esto es realizado por un programa en java lo que reduce su costo. En esta consulta se tiene un costo de 4 y no se implementan índices.El group by genera costo de 4 lo cual es significativo

SELECT numeroRepetidas, ID_RESTAURANTE FROM(SELECT MIN (NUMERO) AS numeroRepetidas, ID_RESTAURANTE FROM(SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TABLE2 group by ID_RESTAURANTE) group by ID_RESTAURANTE) WHERE numeroRepetidas = (SELECT MIN (NUMERO) AS numeroRepetidas FROM(SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TABLE2 group by ID_RESTAURANTE));

```
SELECT numeroRepetidas, ID_RESTAURANTE FROM( SELECT MIN (NUMERO) AS numeroRepetidas, ID_RESTAURANTE FROM( SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TAB
```

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

SQL | Todas las Filas Recuperadas: 1 en 0,028 segundos

	NUMEROREPETIDAS	ID_RESTAURANTE
1	1	1

```
SELECT numeroRepetidas, ID_RESTAURANTE FROM( SELECT MIN (NUMERO) AS numeroRepetidas, ID_RESTAURANTE FROM( SELECT COUNT(*) AS NUMERO, ID_RESTAURANTE FROM TABLE2
```

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

SQL | 0,29 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	4
FILTER				
Filter Predicates				
MIN(NUMERO)= (SELECT MIN(NUMERO) FROM (SELECT COUNT(*) NUMERO,ID_RESTAURANTE ID_RESTAURANTE FROM TABLE2 TABLE2 GROUP BY ID_RESTAURANTE) from$_subquery$_004)				
HASH		GROUP BY	3	4
VIEW			3	4
HASH		GROUP BY	3	4
TABLE ACCESS	TABLE2	FULL	3	3
SORT		AGGREGATE	1	
VIEW			3	4
SORT		GROUP BY	3	4
TABLE ACCESS	TABLE2	FULL	3	3
Other XML				
{info}				
info type="db_version"				
12.1.0.2				
info type="parse_schema"				

En esta consulta se retorna el restaurante menos frecuentando en un dia. al igual que en la anterior se implementa una tabla temporal para poner los datos de los restaurantes frecuentados por dia. Esto es realizado por un programa en java lo que reduce su costo. En esta consulta se tiene un costo de 4 y no se implementan índices.El group by genera costo de 4 lo cual es significativo

Requerimiento 12:

SQL

SELECT *FROM((SELECT * FROM((SELECT IDPEDIDO, FECHA, ID_USUARIO,PEDIDOPRODUCTO.ID_PRODUCTO,ID_MENU,ID_PEDIDO FROM PEDIDO INNER JOIN PEDIDOPRODUCTO ON PEDIDO.IDPEDIDO= PEDIDOPRODUCTO.ID_PEDIDO)k INNER JOIN USUARIO ON k.ID_USUARIO=USUARIO.IDUSUARIO)WHERE ROL = 'Cliente')j INNER JOIN PRODUCTO ON PRODUCTO.IDPRODUCTO=j.ID_PRODUCTO) WHERE CATEGORIA ='Plato fuerte' ;

```
SELECT *FROM((SELECT * FROM((SELECT IDPEDIDO, FECHA, ID_USUARIO,PEDIDOPRODUCTO.ID_PRODUCTO,ID_MENU,ID_PEDIDO FROM PEDIDO INNER JOIN PEDIDOPRODUCTO ON PEDIDO.IDI
```

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

SQL | Todas las Filas Recuperadas: 2 en 0,722 segundos

	IDPEDIDO	FECHA	ID_USUARIO	ID_PRODUCTO	ID_MENU	ID_PEDIDO	IDUSUARIO	NOMBREUSUARIO	ROL	CORREO	IDPRODUCTO	NOMBRE	COSTOPRODUCCION	DESCRIP
1	4	09/09/17	6	2	(null)	4	6	Camilo	Cliente	john@gmail.com	2	Hamburguesa	10	Tomate,
2	5	09/09/17	6	2	(null)	5	6	Camilo	Cliente	john@gmail.com	2	Hamburguesa	10	Tomate,

```
SELECT *FROM((SELECT * FROM((SELECT IDPEDIDO, FECHA, ID_USUARIO,PEDIDOPRODUCTO.ID_PRODUCTO,ID_MENU,ID_PEDIDO FROM PEDIDO INNER JOIN PEDIDOPRODUCTO ON PEDIDO.IDI
```

Salida de Script × | Rastreo Automático × | Resultado de la Consulta × | Explicación del Plan ×

SQL | 0,181 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	17
HASH JOIN			3	17
Access Predicates				
PRODUCTO.IDPRODUCTO=PEDIDOPRODUCTO.ID_PRODUCTO				
NESTED LOOPS			3	17
NESTED LOOPS			9	17
STATISTICS COLLECTOR				
HASH JOIN			9	8
Access Predicates				
PEDIDO.IDPEDIDO=PEDIDOPRODUCTO.ID_PEDIDO				
MERGE JOIN			10	7
TABLE ACCESS	USUARIO	BY INDEX ROWID	2	3
Filter Predicates				
USUARIO.ROL='Cliente'				
INDEX	USUARIO_PK	FULL SCAN	6	1
SORT		JOIN	20	4
Access Predicates				

Esta consulta retorna el cliente que cumple con ciertas condiciones. En el plan de ejecución oracle implementa un índice en el id del usuario por defecto este tiene un costo de 1. Esta consulta se puede ver beneficiada por el uso de un índice en el atributo de CATEGORIA en la tabla producto ya que con dicho índice se reduciría la ejecución

Con indice

Indice en CATEGORIA

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304B011720	INDEX2	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	CATEGORIA
2	ISIS2304B011720	PRODUCTO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDPRODUCTO

```
SELECT *FROM((SELECT * FROM((SELECT IDPEDIDO, FECHA, ID_USUARIO,PEDIDOPRODUCTO.ID_PRODUCTO,ID_MENU,ID_PEDIDO FROM PEDIDO INNER JOIN PEDIDOPRODUCTO ON PEDIDO.IDI
```

Salida de Script × | Rastreo Automático × | Explicación del Plan × | Resultado de la Consulta ×

SQL | Todas las Filas Recuperadas: 2 en 0,024 segundos

	IDPEDIDO	FECHA	ID_USUARIO	ID_PRODUCTO	ID_MENU	ID_PEDIDO	IDUSUARIO	NOMBREUSUARIO	ROL	CORREO	IDPRODUCTO	NOMBRE	COSTOPRODUCCION	DESCRIP
1	4	09/09/17	6	2	(null)	4	6	Camilo	Cliente	john@gmail.com	2	Hamburguesa	10	Tomate,
2	5	09/09/17	6	2	(null)	5	6	Camilo	Cliente	john@gmail.com	2	Hamburguesa	10	Tomate,

```
SELECT *FROM((SELECT * FROM((SELECT IDPEDIDO, FECHA, ID_USUARIO,PEDIDOPRODUCTO.ID_PRODUCTO,ID_MENU,ID_PEDIDO FROM PEDIDO INNER JOIN PEDIDOPRODUCTO ON PEDIDO.
```

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

SQL | 0,127 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	17
HASH JOIN			3	17
Access Predicates				
PRODUCTO.IDPRODUCTO=PEDIDOPRODUCTO.ID_PRODUCTO				
NESTED LOOPS			3	17
NESTED LOOPS			9	17
STATISTICS COLLECTOR				
HASH JOIN			9	8
Access Predicates				
PEDIDO.IDPEDIDO=PEDIDOPRODUCTO.ID_PEDIDO				
MERGE JOIN			10	7
TABLE ACCESS	USUARIO	BY INDEX ROWID	2	3
Filter Predicates				
USUARIO.ROL='Cliente'				
INDEX	USUARIO_PK	FULL SCAN	6	1
SORT		JOIN	20	4
Access Predicates				

Con la aplicacion de un indice el atributo en teoria se deberia reducir el costo ya que este atributo se encuentra en el AND de una condicion del where, la aplicacion de u indice reduciria el tiempo de ejecucion y costo.

Seleccion de indices

En el requerimiento 12 se implemento un indice en el atributo en el cual se usa para evaluar una condicion del where. Esto ayuda ya que se realiza una busqueda de igualdad. Por lo anterior, al ser una busqueda de igualdad, un indice de hash seria mas optimo puesto que al hacer hashing se obtendria el dato exacto a buscar.
Por lo demas no se agregaron mas indices en el requerimiento 11 y 12. Solo los propuestos por oracle.

Analisis de eficiencia

En los resultados de consulta obtenidos para los requerimientos 11 y 12, la implementacion de indices afecta la ejecucion de los mismos. Por su parte en el requerimiento 11 los indices propuestos por oracle son los adecuados ya que ayudan a optimizar los tiempos como se vio en las imagenes. Por otro lado la implementacion de un indice en el requerimiento 12 ayuda a su ejecucion, dicho indice se emplementa en el atributo que se compara en el where ayuda a reducir el costo. Al ser una consulta de igualdad

Analisis de porcesos de optimizacion y modelo de ejecucion

En las propuestas de indice de oracle en el caso de los req 11 son acertadas ya que el indice en el cual se implementa el indice es el adecuado, por otro lado en el requerimiento 12 no siempre el atributo propuesto es el mejor ya que en este no se implemento uno que este en la condicion del where. lo cual ayudaria mucho con la ejecucion.