

# Iteración 3: SuperAndes

Andrés Felipe Hernández León, Jenifer Paola Rodríguez Villamizar

Reporte técnico Iteración 3

Universidad de los Andes, Bogotá, Colombia

{af.hernandezl, jp.rodriguezv}@uniandes.edu.co

Fecha de presentación: Noviembre 4 de 2018

## Tabla de contenido

1	Introducción
2	
2	Diseño de la aplicación
3	
2.1	Modelo conceptual SuperAndes
3	
3	Diseño físico
4	
3.1	Índices en todos los requerimientos funcionales de consulta:
4	
3.1.1	Requerimiento funcional de consulta 1 4
3.1.2	Requerimiento funcional de consulta 2 4
3.1.3	Requerimiento funcional de consulta 3 4
3.1.4	Requerimiento funcional de consulta 4 5
3.1.5	Requerimiento funcional de consulta 5 6
3.1.6	Requerimiento funcional de consulta 6 6
3.1.7	Requerimiento funcional de consulta 7 7
3.1.8	Requerimiento funcional de consulta 8 8
3.1.9	Requerimiento funcional de consulta 9 8
3.2	Análisis realizado para los cuatro nuevos requerimientos funcionales de consulta:
8	
3.2.1	Requerimiento funcional de consulta 10 8
3.2.2	Requerimiento funcional de consulta 11 9
3.2.3	Requerimiento funcional de consulta 12 10
3.2.4	Requerimiento funcional de consulta 13 11
4	Construcción de la aplicación, pruebas y análisis 13
4.1	Diseño de la aplicación, diseño y carga de datos:
13	
4.2	Ajustes al programa para los nuevos requerimientos:
13	
4.3	Optimización y ejecución de las consultas:
13	

## **1 Introducción**

En el siguiente documento se evidenciará el desarrollo de la tercera iteración del curso. Para cada parte de la actividad, se mostrará el trabajo logrado según lo requerido en el documento del proyecto. Adicionalmente, se podrán encontrar anexos a este archivo archivos *.sql* con las sentencias de los requerimientos en texto plano y el proyecto java.

## 2 Diseño de la aplicación

### 2.1 Modelo conceptual SuperAndes

En la figura 1 se muestra el modelo conceptual completo correspondiente al caso de estudio “SuperAndes”. Para esta iteración el modelo no sufrió ningún cambio.

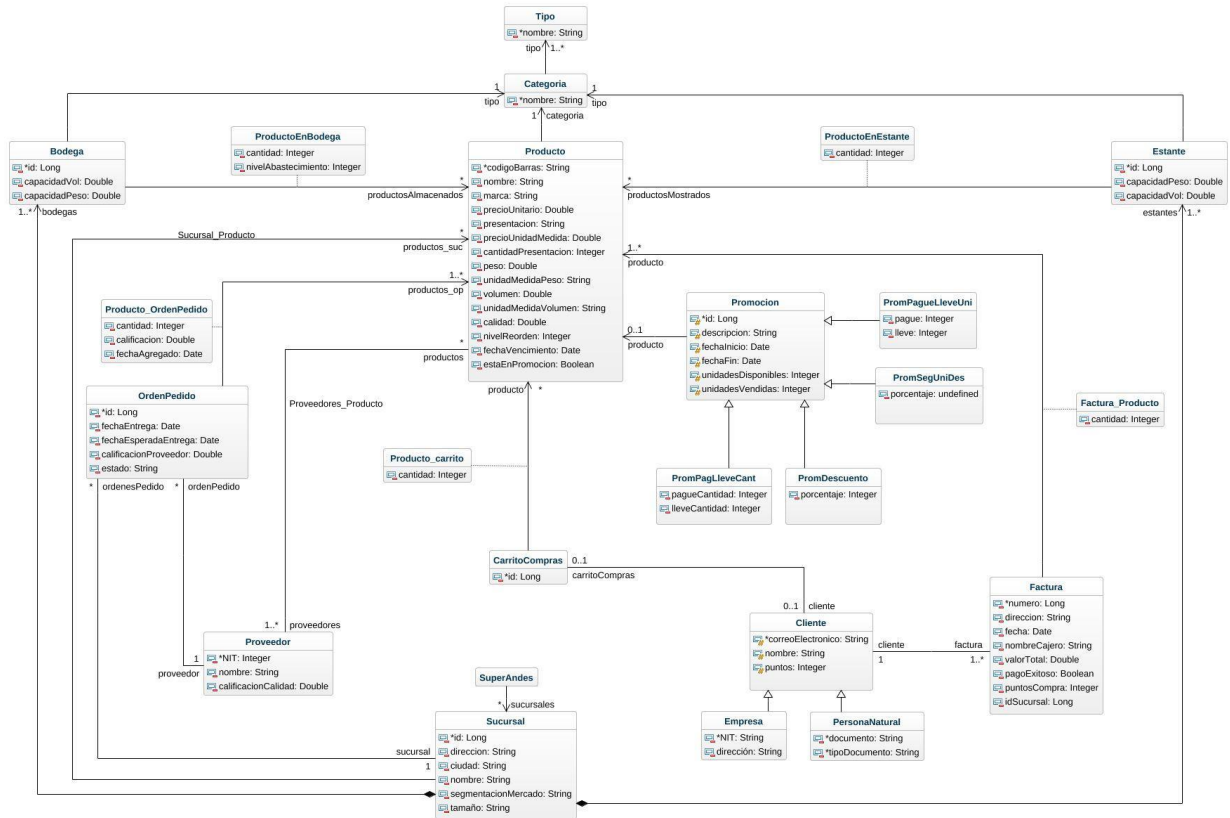


Figura 1. Modelo UML SuperAndes

### 3 Diseño físico

#### 3.1 Índices en todos los requerimientos funcionales de consulta:

##### 3.1.1 Requerimiento funcional de consulta 1

SELECT SUM (VALORTOTAL) AS VALORTOTAL FROM ( SELECT VALORTOTAL, FECHA, IDSUCURSAL FROM FACTURA WHERE EXTRACT (YEAR FROM FECHA) = EXTRACT (YEAR FROM (SELECT SYSDATE FROM DUAL)) AND FECHA BETWEEN 'DD/MM/YY' AND 'DD/MM/YY' AND IDSUCURSAL =? );

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				4
SORT				
FILTER				
Filter Predicates				
TO_DATE('DD/MM/YY')>=TO_DATE('DD/MM/YY')				
TABLE ACCESS	FACTURA	FULL	1	2
Filter Predicates				
AND				
IDSUCURSAL=TO_NUMBER(:1)				
FECHA='DD/MM/YY'				
EXTRACT(YEAR FROM INTERNAL_FUNCTION('FECHA'))=EXTRACT(YEAR FROM (SELECT SYSDATE@1 FROM SYS.DUAL))				
FAST DUAL			1	2

##### 3.1.2 Requerimiento funcional de consulta 2

SELECT \* FROM (SELECT RELACION, ID FROM (SELECT TRUNC ((SELECT SYSDATE FROM DUAL) - FECHAINICIAL) AS RELACION, ID FROM prom\_descuento) GROUP BY RELACION, ID ORDER BY RELACION DESC) WHERE rownum between 1 and 20;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				6
FAST DUAL			1	2
COUNT		STOPKEY		
Filter Predicates				
ROWNUM<=20				
FILTER				
Filter Predicates				
ROWNUM>=1				
VIEW				
SORT				
VIEW				
TABLE ACCESS	PROM_DESCUENTO	FULL	1	3
GROUP BY			1	6
SORT			1	6
VIEW			1	5

##### 3.1.3 Requerimiento funcional de consulta 3

SELECT IDESTANTE,SUM(INDICEPESO1) AS INDICEPESO,SUM(INDICEVOL1) INDICEVOL,IDSUCURSAL FROM(SELECT IDESTANTE, PESOTOTAL/CAPACIDADPESO AS INDICEPESO1, VOLUMENTOTAL/CAPACIDADVOL AS INDICEVOL1, IDSUCURSAL FROM(SELECT A.IDESTANTE AS ESTANTE, A.CANTIDAD, A.CODIGOBARRASPRODUCTO, (a.cantidad\*B.PESO) AS PESOTOTAL, (A.CANTIDAD\*B.VOLUMEN ) AS VOLUMENTOTAL FROM PRODUCTOSENESTANTE A JOIN (SELECT CODIGOBARRAS, PESO, VOLUMEN FROM PRODUCTO) B ON A.CODIGOBARRASPRODUCTO = b.codigobarras) K JOIN (SELECT CAPACIDADVOL, CAPACIDADPESO, IDSUCURSAL, ID AS IDESTANTE FROM ESTANTE) L ON k.ESTANTE= l.idestante) GROUP BY IDESTANTE, IDSUCURSAL;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				4
HASH				
NESTED LOOPS				
NESTED LOOPS				
NESTED LOOPS				
TABLE ACCESS	PRODUCTOSENESTANTE	FULL	1	3
TABLE ACCESS	ESTANTE	BY INDEX ROWID	1	3
INDEX	ESTANTE_PK	UNIQUE SCAN	1	0
Access Predicates				
A.IDESTANTE=ID				
INDEX	PRODUCTO_PK	UNIQUE SCAN	1	0
Access Predicates				
A.CODIGOBARRASPRODUCTO=CODIGOBARRAS				
TABLE ACCESS	PRODUCTO	BY INDEX ROWID	1	0

```

SELECT IDBODEGA, SUM(INDICEPESO1) AS INDICEPESO, SUM(INDICEVOL1)
INDICEVOL, IDSUCURSAL FROM( SELECT IDBODEGA,
PESOTOTAL/CAPACIDADPESO AS INDICEPESO1,
VOLUMENTOTAL/CAPACIDADVOL AS INDICEVOL1, IDSUCURSAL
FROM( SELECT A.IDBODEGA AS BODEGA, A.CANTIDAD,
A.CODIGOBARRASPRODUCTO, (a.cantidad*B.PESO) AS PESOTOTAL,
(A.CANTIDAD*B.VOLUMEN ) AS VOLUMENTOTAL FROM
PRODUCTOSENBODEGA A JOIN(SELECT CODIGOBARRAS, PESO, VOLUMEN
FROM PRODUCTO)B ON A.CODIGOBARRASPRODUCTO = b.codigobarras ) K JOIN
(SELECT CAPACIDADVOL, CAPACIDADPESO, IDSUCURSAL, ID AS IDBODEGA
FROM BODEGA) L ON k.BODEGA= l.idBODEGA) GROUP BY IDBODEGA,
IDSUCURSAL;

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1 4
HASH		GROUP BY		1 4
NESTED LOOPS				1 3
NESTED LOOPS				1 3
TABLE ACCESS	PRODUCTOSENBODEGA	FULL		1 3
TABLE ACCESS	BODEGA	BY INDEX ROWID		1 0
INDEX	BODEGA_PK	UNIQUE SCAN		1 0
Access Predicates	A.IDBODEGA=ID			
INDEX	PRODUCTO_PK	UNIQUE SCAN		1 0
Access Predicates	A.CODIGOBARRASPRODUCTO=CODIGOBARRAS			
TABLE ACCESS	PRODUCTO	BY INDEX ROWID		1 0

### 3.1.4 Requerimiento funcional de consulta 4

--con un precio en un rango dado.

```
SELECT * FROM PRODUCTOS WHERE PRECIO BETWEEN ? AND ?;
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				205 308
FILTER				
Filter Predicates				
TO_NUMBER(:2)>=TO_NUMBER(:1)				
TABLE ACCESS	PRODUCTO	FULL		205 308
Filter Predicates				
AND				
PRECIOUNITARIO>=TO_NUMBER(:1)				
PRECIOUNITARIO<=TO_NUMBER(:2)				

--productos ofrecidos por un proveedor.

```
SELECT * FROM PROVEEDORES_PRODUCTO WHERE PROVEEDOR = '?';
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1 0
INDEX	PROVEEDORES_PRODUCTO_PK	RANGE SCAN		1 0
Access Predicates	PROVEEDOR='44684864'			

--productos con fecha de vencimiento después de cierta fecha.

```
SELECT * FROM PRODUCTO WHERE FECHAVENCIMIENTO > ?;
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				205 308
FILTER				
Filter Predicates				
TO_NUMBER(:2)>=TO_NUMBER(:1)				
TABLE ACCESS	PRODUCTO	FULL		205 308
Filter Predicates				
AND				
PRECIOUNITARIO>=TO_NUMBER(:1)				
PRECIOUNITARIO<=TO_NUMBER(:2)				

--productos con un peso en un rango dado.

```
SELECT * FROM PRODUCTO WHERE PRECIOUNITARIO BETWEEN ? AND ?;
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				308
FILTER				205
Filter Predicates				
TO_NUMBER(:2) >= TO_NUMBER(:1)				
TABLE ACCESS	PRODUCTO	FULL		308
Filter Predicates				
AND				
PRECIOUNITARIO >= TO_NUMBER(:1)				
PRECIOUNITARIO <= TO_NUMBER(:2)				

--productos con un volumen en un rango dado.

SELECT \* FROM PRODUCTO WHERE VOLUMEN BETWEEN ? AND ?;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				308
FILTER				821
Filter Predicates				
TO_NUMBER(:2) >= TO_NUMBER(:1)				
TABLE ACCESS	PRODUCTO	FULL		308
Filter Predicates				
AND				
VOLUMEN >= TO_NUMBER(:1)				
VOLUMEN <= TO_NUMBER(:2)				

--productos con una categoria dado.

SELECT \* FROM PRODUCTO WHERE CATEGORIA= ?;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				308
TABLE ACCESS	PRODUCTO	FULL		3565
Filter Predicates				3565
CATEGORIA =:1				

--productos ofrecidos por cierta sucursal.

SELECT \* FROM SUCURSAL\_PRODUCTO WHERE PRODUCTO= ?;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				0
INDEX	SUCURSAL_PRODUCTO_PK	RANGE SCAN		1
Access Predicates				1
CODIGOPRODUCTO=:1				0

--productos ofrecidos en cierta ciudad.

SELECT ID, CIUDAD, CODIGOPRODUCTO FROM (SELECT ID, CIUDAD  
FROM SUCURSAL WHERE CIUDAD=? )A JOIN (SELECT \* FROM  
SUCURSAL\_PRODUCTO) B ON A.ID= B.IDSUCURSAL;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				0
NESTED LOOPS				1
NESTED LOOPS				1
INDEX	SUCURSAL_PRODUCTO_PK	FULL SCAN		1
INDEX	SUCURSAL_PK	UNIQUE SCAN		1
Access Predicates				1
ID=SUCURSAL_PRODUCTO.IDSUCURSAL				0
TABLE ACCESS	SUCURSAL	BY INDEX ROWID		1
Filter Predicates				0
CIUDAD=:1				

### 3.1.5 Requerimiento funcional de consulta 5

SELECT \* FROM ORDENPEDIDO WHERE ESTADO = 'Entregado' ORDER BY  
PROVEEDOR;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				4
SORT		ORDER BY		1
TABLE ACCESS	ORDENPEDIDO	FULL		1
Filter Predicates				3
ESTADO='Entregado'				

### 3.1.6 Requerimiento funcional de consulta 6

MOSTRAR LAS VENTAS DE SUPERANDES A UN USUARIO DADO, EN UN  
RANGO DE FECHAS INDICADO

select \* from factura

where cliente =? and fecha between ? and ?;

SQL | 0,158 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1 0
FILTER				
Filter Predicates				
AND				
NULL IS NOT NULL				
TO_DATE(")>=TO_DATE(")				
TABLE ACCESS	FACTURA	FULL	1	2
Filter Predicates				
FECHA="				

### 3.1.7 Requerimiento funcional de consulta 7

producto más solicitado

SELECT MAX(CANTIDAD), SEMANA FROM (SELECT COUNT(A.PRODUCTO) AS CANTIDAD, TO\_CHAR(fecha, '??') AS SEMANA FROM ( SELECT CODIGOBARRASPRODUCTO AS PRODUCTO FROM TIPO\_PRODUCTO WHERE NOMBRETIPO =") A JOIN (SELECT \* FROM(SELECT FACTURA , PRODUCTO FROM FACTURA\_PRODUCTO) P JOIN (SELECT NUMERO, FECHA FROM FACTURA )K ON P.FACTURA= K.NUMERO)B ON A.PRODUCTO = B.PRODUCTO GROUP BY TO\_CHAR(fecha, '??') ) GROUP BY SEMANA;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1 0
HASH		GROUP BY		1
FILTER				
Filter Predicates				
NULL IS NOT NULL				
NESTED LOOPS				19
MERGE JOIN		CARTESIAN		19
TABLE ACCESS	FACTURA	FULL	1	2
BUFFER				
TABLE ACCESS	TIPO_PRODUCTO	FULL	15998	17
INDEX	FACTURA_PRODUCTO_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
FACTURA=NUMERO				
CODIGOBARRASPRODUCTO=PRODUCTO				

producto menos solicitado

SELECT Min(CANTIDAD), SEMANA FROM (SELECT COUNT(A.PRODUCTO) AS CANTIDAD, TO\_CHAR(fecha, '??') AS SEMANA FROM ( SELECT CODIGOBARRASPRODUCTO AS PRODUCTO FROM TIPO\_PRODUCTO WHERE NOMBRETIPO =") A JOIN ( SELECT \* FROM ( SELECT FACTURA , PRODUCTO FROM FACTURA\_PRODUCTO)P JOIN (SELECT NUMERO, FECHA FROM FACTURA )K ON P.FACTURA= K.NUMERO)B ON A.PRODUCTO = B.PRODUCTO GROUP BY TO\_CHAR(fecha, '??')) GROUP BY SEMANA;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1 0
HASH		GROUP BY		1
FILTER				
Filter Predicates				
NULL IS NOT NULL				
NESTED LOOPS				19
MERGE JOIN		CARTESIAN		19
TABLE ACCESS	FACTURA	FULL	1	2
BUFFER				
TABLE ACCESS	TIPO_PRODUCTO	FULL	15998	17
INDEX	FACTURA_PRODUCTO_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
FACTURA=NUMERO				
CODIGOBARRASPRODUCTO=PRODUCTO				

mayores ingresos

SELECT MAX(CANTIDAD) as valorTotal, SEMANA FROM (SELECT COUNT(valortotal) AS cantidad, TO\_CHAR(fecha, '??') AS SEMAN FROM ( SELECT CODIGOBARRASPRODUCTO AS PRODUCTO FROM TIPO\_PRODUCTO WHERE NOMBRETIPO =") A JOIN (SELECT \* FROM(SELECT FACTURA , PRODUCTO FROM FACTURA\_PRODUCTO)P JOIN (SELECT NUMERO, FECHA, valortotal FROM FACTURA )K ON P.FACTURA= K.NUMERO)B ON A.PRODUCTO = B.PRODUCTO GROUP BY TO\_CHAR(fecha, '??')) GROUP BY SEMANA;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				0
HASH			1	1
FILTER		GROUP BY		1
Filter Predicates				
NULL IS NOT NULL				
NESTED LOOPS			1	19
MERGE JOIN		CARTESIAN	1	19
TABLE ACCESS	FACTURA	FULL	1	2
BUFFER				
TABLE ACCESS	TIPO_PRODUCTO	SORT	15998	17
INDEX	FACTURA_PRODUCTO_PK	FULL	15998	17
Access Predicates		UNIQUE SCAN	1	0
AND				
FACTURA=NUMERO				
CODIGOBARRASPRODUCTO=PRODUCTO				
Other XML				

### 3.1.8 Requerimiento funcional de consulta 8

SELECT MESES , COUNT(ID\_FACTURA), CLIENTE FROM(SELECT FECHA, EXTRACT(MONTH FROM FECHA) AS MESES, NUMERO AS ID\_FACTURA, CLIENTE FROM FACTURA) GROUP BY MESES, CLIENTE, ID\_FACTURA HAVING COUNT (ID\_FACTURA) >=2;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3
FILTER				1
Filter Predicates				
COUNT(*) >=2				
HASH		GROUP BY	1	3
TABLE ACCESS	FACTURA	FULL	1	2
Other XML				

### 3.1.9 Requerimiento funcional de consulta 9

## 3.2 Análisis realizado para los cuatro nuevos requerimientos funcionales de consulta:

### 3.2.1 Requerimiento funcional de consulta 10

select \*

from( select factura from factura\_producto

where producto = ? ) a join (select \*

from (( select numero , cliente from factura

where fecha BETWEEN ? AND ?) c join(select \* from cliente)d on c.cliente=d.correoelectronico )) b on a.factura=b.numero;

Sin indices



SQL | 0,113 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3
FILTER				
Filter Predicates				
TO_DATE(:3)>=TO_DATE(:2)				
HASH JOIN				3
Access Predicates				
CLIENTE=CLIENTE.CORREOELECTRONICO				
NESTED LOOPS				3
NESTED LOOPS				3
STATISTICS COLLECTOR				
NESTED LOOPS				2
TABLE ACCESS	FACTURA	FULL		2
Filter Predicates				
AND				
FECHA>=:2				
FECHA<=:3				
INDEX	FACTURA_PRODUCTO_PK	UNIQUE SCAN		0
Access Predicates				
AND				
FACTURA=NUMERO				
PRODUCTO=:1				
INDEX	CLIENTE_PK	UNIQUE SCAN		0
Access Predicates				
CLIENTE=CLIENTE.CORREOELECTRONICO				
TABLE ACCESS	CLIENTE	BY INDEX ROWID		1
TABLE ACCESS	CLIENTE	FULL		1

Con índices sobre fecha en factura y factura en factura producto

CREATE INDEX index\_fecha\_factura ON factura(fecha) ;

CREATE INDEX index\_FACTURA\_FACTURA\_PRD ON factura\_producto(factura) ;

SQL | 0,11 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
FILTER				
Filter Predicates				
TO_DATE(:3)>=TO_DATE(:2)				
HASH JOIN				2
Access Predicates				
CLIENTE=CLIENTE.CORREOELECTRONICO				
NESTED LOOPS				2
NESTED LOOPS				2
STATISTICS COLLECTOR				
NESTED LOOPS				1
TABLE ACCESS	FACTURA	BY INDEX ROWID BATCHED		1
INDEX	INDEX_FECHA_FACTURA	RANGE SCAN		1
Access Predicates				
AND				
FECHA>=:2				
FECHA<=:3				
TABLE ACCESS	FACTURA_PRODUCTO	BY INDEX ROWID BATCHED		0
Filter Predicates				
PRODUCTO=:1				
INDEX	INDEX_FACTURA_FACTURA_PRD	RANGE SCAN		0
Access Predicates				
FACTURA=NUMERO				
INDEX	CLIENTE_PK	UNIQUE SCAN		0
Access Predicates				

### 3.2.2 Requerimiento funcional de consulta 11

SELECT \* FROM (select \* from cliente) minus ( select correoelectronico, nombre, puntos,empresa, documentopn from( select factura from factura\_producto where producto = " ) a join (select \* from (( select numero , cliente from factura where fecha BETWEEN " AND " ) c join(select \* from cliente)d on c.cliente= d.correoelectronico )) b on a.factura=b.numero);

SQL | 0,114 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				145640 2157
MINUS				
SORT				
TABLE ACCESS	CLIENTE	UNIQUE		145640 2156
SORT				
TABLE ACCESS	CLIENTE	FULL		145640 308
TABLE ACCESS	CLIENTE	UNIQUE		1 1
FILTER				
Filter Predicates				
AND				
NULL IS NOT NULL				
TO_DATE(*)>=TO_DATE(*)				
NESTED LOOPS				1 4
NESTED LOOPS				1 4
NESTED LOOPS				1 3
TABLE ACCESS	FACTURA_PRODUCTO	FULL		1 2
TABLE ACCESS	FACTURA	BY INDEX ROWID		1 1
Filter Predicates				
FECHA=				
INDEX	FACTURA_PK	UNIQUE SCAN		1 0
Access Predicates				
FACTURA=NUMERO				
INDEX	CLIENTE_PK	UNIQUE SCAN		1 0
Access Predicates				
CLIENTE=CLIENTE.CORREOELECTRONICO				
TABLE ACCESS	CLIENTE	BY INDEX ROWID		1 1

Con indices sobre fecha en factura y factura en factura producto

CREATE INDEX index\_fecha\_factura ON factura(fecha) ;

CREATE INDEX index\_FACTURA\_FACTURA\_PRD ON factura\_producto(factura) ;

SQL | 0,115 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				145640 2157
MINUS				
SORT				
TABLE ACCESS	CLIENTE	UNIQUE		145640 2156
SORT				
TABLE ACCESS	CLIENTE	FULL		145640 308
TABLE ACCESS	CLIENTE	UNIQUE		1 1
FILTER				
Filter Predicates				
AND				
NULL IS NOT NULL				
TO_DATE(*)>=TO_DATE(*)				
NESTED LOOPS				1 2
NESTED LOOPS				1 2
NESTED LOOPS				1 1
TABLE ACCESS	FACTURA	SEMI		1 1
TABLE ACCESS	INDEX_FECHA_FACTURA	BY INDEX ROWID BATCHED		1 1
Access Predicates				
FECHA=				
INDEX	INDEX_FACTURA_FACTURA_PRD	RANGE SCAN		1 1
Access Predicates				
FACTURA=NUMERO				
INDEX	CLIENTE_PK	UNIQUE SCAN		1 0
Access Predicates				
CLIENTE=CLIENTE.CORREOELECTRONICO				
TABLE ACCESS	CLIENTE	BY INDEX ROWID		1 1

### 3.2.3 Requerimiento funcional de consulta 12

SELECT MAX(NUMERO\_PEDIDOS) , MIN(NUMERO\_PEDIDOS), SEMANA

FROM ((SELECT COUNT(PROVEEDOR) AS NUMERO\_PEDIDOS, proveedor, TO\_CHAR(fechaesperadaentrega, 'WW')AS SEMANA FROM ORDENPEDIDO

GROUP BY PROVEEDOR, TO\_CHAR(fechaesperadaentrega, 'WW')) A join (SELECT \* FROM PROVEEDOR) B ON A.PROVEEDOR= B.NIT)

GROUP BY SEMANA ;

SQL | 0,118 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1 5
HASH		GROUP BY		1 5
NESTED LOOPS		SEMI		1 4
VIEW				1 4
HASH		GROUP BY		1 4
TABLE ACCESS	ORDENPEDIDO	FULL		1 3
INDEX	PROVEEDOR_PK	UNIQUE SCAN		1 0
Access Predicates				
A.PROVEEDOR=PROVEEDOR.NIT				

Con indice sobre proveedor en orden pedido

CREATE INDEX index\_PROVEEDOR ON ORDENPEDIDO(PROVEEDOR) ;

SQL | 0,119 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
HASH		GROUP BY		2
VIEW	SYS.VM_NWWW_1			1
HASH		GROUP BY		1
NESTED LOOPS				1
NESTED LOOPS				0
INDEX	PROVEEDOR_PK	FULL SCAN		0
INDEX	INDEX_PROVEEDOR	RANGE SCAN		0
Access Predicates	PROVEEDOR=PROVEEDOR.NIT			1
TABLE ACCESS	ORDENPEDIDO	BY INDEX ROWID		0

Other XML

SELECT MAX (CANTIDAD) as p\_cantidad\_maxima , MIN (CANTIDAD) as p\_cantidad\_minima , semana

FROM(

SELECT sum(a.CANTIDAD) AS CANTIDAD, TO\_CHAR(b.fecha, 'WW') AS SEMANA  
FROM

(s(SELECT CANTIDAD, FACTURA, PRODUCTO FROM FACTURA\_PRODUCTO) A

JOIN (SELECT NUMERO, fecha FROM FACTURA) B ON a.FACTURA= B.NUMERO

GROUP BY TO\_CHAR(b.FECHA, 'WW'), a.producto

)

GROUP BY SEMANA;

SQL | 0,312 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				5
HASH		GROUP BY		5
VIEW				4
HASH		GROUP BY		4
NESTED LOOPS				3
NESTED LOOPS				3
TABLE ACCESS	FACTURA_PRODUCTO	FULL		2
INDEX	FACTURA_PK	UNIQUE SCAN		0
Access Predicates	FACTURA=NUMERO			1
TABLE ACCESS	FACTURA	BY INDEX ROWID		1

Other XML

con indices sobre factura en factura producto

CREATE INDEX index\_FACTURA\_FACTURA\_PRD ON factura\_producto(factura) ;

SQL | 0,113 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				4
HASH		GROUP BY		4
VIEW				3
HASH		GROUP BY		3
NESTED LOOPS				2
NESTED LOOPS				2
TABLE ACCESS	FACTURA	FULL		2
INDEX	INDEX_FACTURA_FACTURA_PRD	RANGE SCAN		0
Access Predicates	FACTURA=NUMERO			1
TABLE ACCESS	FACTURA_PRODUCTO	BY INDEX ROWID		0

Other XML

### 3.2.4 Requerimiento funcional de consulta 13

--Los clientes que realizan una compra por lo menos una vez al mes.

Select z.cliente, NumMesesCompra, productoCaro, productoCategorias

from(Select cliente, NumMesesCompra

from( select cliente, count(distinct(month)) as NumMesesCompra

from( select cliente, EXTRACT(MONTH FROM fecha) as month from factura where

EXTRACT(YEAR FROM FECHA) = 2018

) group by cliente

```

) where NumMesesCompra = 12
)z,
(SELECT a.cliente, productoCaro, productoCategorias
from ( SELECT cliente, count(numero) as numFacturas from factura group by cliente
)a,
(SELECT cliente, count(distinct(factura))as numFacturasCaras, nombre as
productoCaro
FROM FACTURA, FACTURA_PRODUCTO, PRODUCTO
WHERE PRODUCTO.PRECIOUNITARIO > 100000 group by cliente,nombre
),
(SELECT cliente, count(distinct(factura))as numFacturasCat, nombre as
productoCategorias
FROM FACTURA, FACTURA_PRODUCTO , PRODUCTO
WHERE PRODUCTO.CATEGORIA ='Technology' OR PRODUCTO.CATEGORIA
= 'Tools' group by cliente, nombre
)
where numFacturas = numFacturasCaras OR numFacturas = numFacturasCat)
order by cliente;

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				633
SORT		ORDER BY		633
NESTED LOOPS				632
MERGE JOIN		CARTESIAN		319
MERGE JOIN		CARTESIAN		6
VIEW				3
FILTER				
Filter Predicates				
COUNT(\$vm_col_1)=12				
HASH		GROUP BY		3
VIEW	SYS.VM_NWWW_3			3
HASH		GROUP BY		3
TABLE ACCESS	FACTURA	FULL		2
Filter Predicates				
EXTRACT(YEAR FROM INTERNAL_FUNCTION(FECHA))=2018				
BUFFER				6
VIEW				3
HASH		GROUP BY		3
TABLE ACCESS	FACTURA	FULL		2
BUFFER				316
VIEW				313
HASH		GROUP BY		313
VIEW	SYS.VM_NWWW_2			312
HASH		GROUP BY		312
MERGE JOIN		CARTESIAN		311
SQL				0,213 segundos
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
MERGE JOIN		CARTESIAN		3
INDEX	INDEX_FACTURA_FACTURA_PRD	FULL SCAN		1
BUFFER				2
TABLE ACCESS	FACTURA	FULL		2
TABLE ACCESS	PRODUCTO	FULL		310
Filter Predicates				308
PRODUCTO.PRECIOUNITARIO > 100000				
VIEW				313
Filter Predicates				
NUMFACTURAS=NUMFACTURASCARAS				
NUMFACTURAS=NUMFACTURASCAT				
SORT		GROUP BY		313
VIEW	SYS.VM_NWWW_1			312
SORT		GROUP BY		312
MERGE JOIN		CARTESIAN		311
MERGE JOIN		CARTESIAN		3
INDEX	INDEX_FACTURA_FACTURA_PRD	FULL SCAN		1
BUFFER				2
TABLE ACCESS	FACTURA	FULL		2
BUFFER				7147
TABLE ACCESS	PRODUCTO	FULL		310
TABLE ACCESS	PRODUCTO	FULL		7147

## **4 Construcción de la aplicación, pruebas y análisis**

### **4.1 Diseño de la aplicación, diseño y carga de datos:**

Se hicieron los cambios necesarios en la creación de las tablas que se encuentra en el proyecto en la ruta “./superAndes/data/Crear y poblar tablas/esquemaSuperAndes.sql”. En cuanto a la carga de datos, en principio la hicimos usando insert generados por Mockaroo y concatenados por nosotros. Luego se cargaron otras por csv en sqlLoader. Por tanto, se encuentran tanto los archivos .sql para llenar cada una de las tablas, como los .csv para llenar otras de las tablas con el orden respectivo en que deberían cargarse. Los datos fueron generados de forma uniforme (gracias a expresiones regulares generadas por nosotros) en el generador de datos Mockaroo. Finalmente, cabe resaltar que los datos ya fueron cargados a la base de datos que utiliza la aplicación.

### **4.2 Ajustes al programa para los nuevos requerimientos:**

Para esta iteración, no fue necesario realizar cambios mayores en la aplicación puesto que ya habíamos modelado para la iteración dos los roles de administrador y usuarios (persona natural - empresa) y no fue necesario cambiar nada del modelo relacional ni conceptual, por lo que solamente se implementaron las cosas necesarias para las consultas (por ejemplo, en las clases SQL de los debidos resultados agregar los nuevos métodos y conectarlos con interfaz, etc).

### **4.3 Optimización y ejecución de las consultas:**

Para esta sección se realizó la implementación de los índices que se puede observar en la sección 3.2 donde se explica para cada tabla como fue tomada la decisión de utilizar un índice o no. Para la ejecución de las consultas hay dos modalidades, una como administrador desde nuestra aplicación (primera opción para ingresar) o directamente corriendo el script en los archivos .sql encontrados en nuestro proyecto en la dirección “./data/Iteracion3/RFC” desde sqlDeveloper.

En lo que se refiere a los requerimientos funcionales de consulta 10 y 11, desde la aplicación se puede elegir los posibles group by u order by deseados. En sqlDeveloper, se adjuntan después del requerimiento básico las consultas para los posibles group by u order by para facilidad de la ejecución de estos.