Documento de Informe - Iteración 3

Juan Esteban Berdugo González, Oscar Castañeda
Universidad de los Andes, Bogotá, Colombia
{je.berdugo10, oj.castaneda}@uniandes.edu.co
Fecha de presentación: Mayo 3 de 2020

Contenido

1 Introducción

Este documento corresponde al informe general asociado a la cuarta iteración del proyecto correspondiente al curso Sistemas Transaccionales. En este proyecto se busca modelar la dinámica asociada a un sistema de alojamiento gestionado por la universidad para su comunidad. Entre las funcionalidades se requiere que se puedan registrar alojamientos y reservas, entre otras características. Por medio de este documento se espera representar de forma completa el proceso de desarrollo de la iteración y el análisis correspondiente.

2 Modelos

A continuación, se presenta el apartado de modelos del proyecto AlohAndes

2.1 Modelo Conceptual

El modelo conceptual asociado al caso de AlohAndes se presenta a continuación:

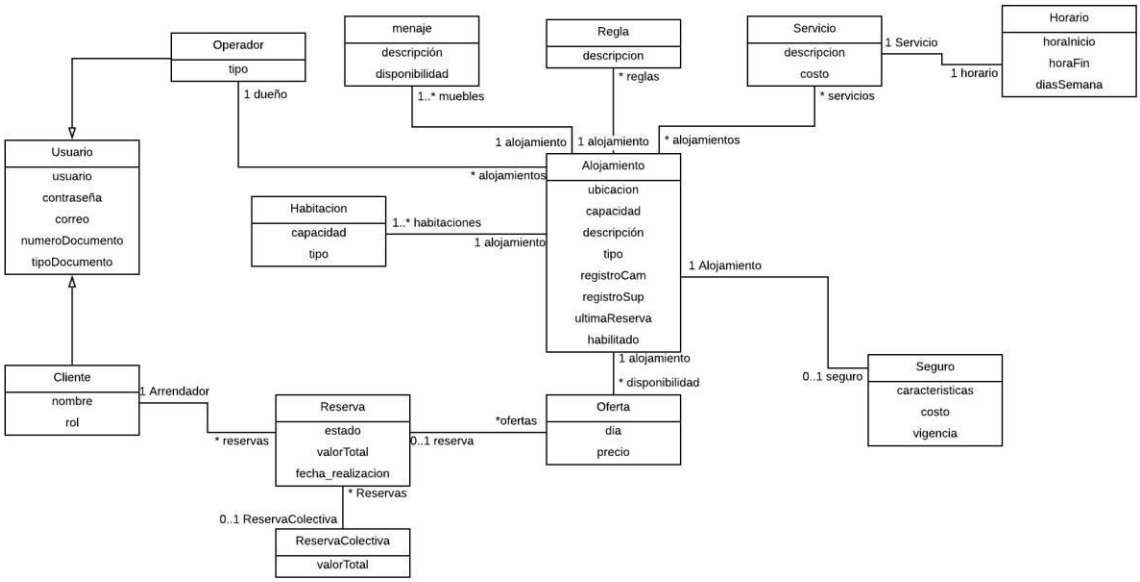


Figura 1. Diagrama UML – Modelo Conceptual AlohAndes

2.2 Modelo de datos relacional

A continuación, se presenta el modelo de datos relacional obtenido a partir del desarrollo del modelo conceptual anteriormente presentado:

USUARIO	
PK	ID: numerico
NN, ND	USUARIO: varchar
NN, ND	CORREO: varchar
NN	CONTRASEÑA: varchar
NN, ND, CK[>0]	NUMERO_DOCUMENTO: numerico
NN, CK[0: CC, 1: NIT, 2: CE]	TIPO_DOCUMENTO: numerico

ALOJAMIENTO	
PK	ID: numerico
NN	UBICACION: varchar
NN, CK[>0]	CAPACIDAD: numerico
NN	DESCRIPCION: varchar
NN, CK[0: Hotel, 1: Hostal, 2: Habitacion, 3: Vivienda, 4: Apartamento, 5: Residencia]	TIPO: numerico
	REGISTRO_CAM varchar
	REGISTRO_SUP varchar
	ULTIMA_RESERVA: date
NN, CK[0: Disponible, 1: No Disponible]	HABILITADO: numerico
NN, FK(OPERADOR.ID)	OPERADOR_ID: numerico

HORARIO	
PK	ID: numerico
NN	HORA_INICIO: date
NN, CK[Despues de hora inicio]	HORA_FIN: date
NN	DIAS_SEMANA: varchar

REGLA	
PK	ID: numerico
NN	DESCRIPCION: varchar
NN, FK(ALOJAMIENTO.ID)	ALOJAMIENTO_ID: numerico

CLIENTE	
PK, FK(Usuario.ID)	ID: numerico
NN	NOMBRE: varchar
NN, CK[0: Profesor, 1: Empleado, 2: Egresado, 3: Estudiante, 4: Padre]	ROL: numerico

HABITACION	
PK	ID: numerico
NN, CK[>0]	CAPACIDAD: numerico
NN, CK[0: Sencilla, 1:Doble, 2: Familiar,3: Compartida]	TIPO: numerico
NN, FK(ALOJAMIENTO.ID)	ALOJAMIENTO_ID: numerico

RESERVA	
PK	ID: numerico
NN, CK[0: No Pagada, 1:Pagada]	ESTADO: date
NN, CK[>0]	VALOR_TOTAL: numerico
NN	FECHA_REALIZACION: date
FK(CLIENTE.ID)	CLIENTE_ID: numerico

OFERTA	
PK	ID: numerico
NN	DIA: date
NN, CK[>0]	PRECIO: numerico
NN, FK(ALOJAMIENTO.ID)	ALOJAMIENTO_ID: numerico
NN, FK(RESERVA.ID)	RESERVA_ID: numerico

RESERVA_COLECCTIVA	
PK	ID: numerico
NN, CK[>0]	VALOR_TOTAL: numerico

OPERADOR	
PK, FK(USUARIO.ID)	ID: numerico
NN,CK[0: Empresa, 1: Independiente]	TIPO: numerico

SERVICIO	
PK	ID: numerico
NN	DESCRIPCION: varchar
NN, CK[>0]	COSTO: numerico
NN, FK(HORARIO.ID)	HORARIO_ID: numerico

SERVICIOS_ALOJAMIENTOS	
PK, FK(ALOJAMIENTO.ID)	ALOJAMIENTO_ID: numerico
PK, FK(SERVICIO.ID)	SERVICIO_ID: numerico

SEGURO	
PK	ID: numerico
NN	CARACTERISTICAS: varchar
NN, CK[>0]	COSTO: numerico
NN	VIGENCIA: date
NN, FK(ALOJAMIENTO.ID)	ALOJAMIENTO_ID: numerico

MENAJE	
PK	ID: numerico
NN	DESCRIPCION: varchar
NN, CK[0: Disponible, 1: No Disponible]	DISPONIBILIDAD: numerico
NN, FK(ALOJAMIENTO.ID)	ALOJAMIENTO_ID: numerico

Figura 2. Modelo relacional AlohAndes

3 Diseño Físico

Los índices generados por Oracle son los siguientes:

	OWNER	INDEX_NAME	INDEX_TYPE	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	UNIQUENESS	COMPRESSIC
1	ISIS2304A022010	SYS_C00798274	NORMAL	ISIS2304A022010	USUARIO	TABLE	UNIQUE	DISABLED
2	ISIS2304A022010	SYS_C00798275	NORMAL	ISIS2304A022010	OPERADOR	TABLE	UNIQUE	DISABLED
3	ISIS2304A022010	SYS_C00798276	NORMAL	ISIS2304A022010	CLIENTE	TABLE	UNIQUE	DISABLED
4	ISIS2304A022010	SYS_C00798277	NORMAL	ISIS2304A022010	OFERTA	TABLE	UNIQUE	DISABLED
5	ISIS2304A022010	SYS_C00798278	NORMAL	ISIS2304A022010	RESERVA	TABLE	UNIQUE	DISABLED
6	ISIS2304A022010	SYS_C00798279	NORMAL	ISIS2304A022010	RESERVA_COLECTIVA	TABLE	UNIQUE	DISABLED
7	ISIS2304A022010	SYS_C00798280	NORMAL	ISIS2304A022010	ALOJAMIENTO	TABLE	UNIQUE	DISABLED
8	ISIS2304A022010	SYS_C00798281	NORMAL	ISIS2304A022010	HORARIO	TABLE	UNIQUE	DISABLED
9	ISIS2304A022010	SYS_C00798282	NORMAL	ISIS2304A022010	REGLA	TABLE	UNIQUE	DISABLED
10	ISIS2304A022010	SYS_C00798283	NORMAL	ISIS2304A022010	MENAJE	TABLE	UNIQUE	DISABLED
11	ISIS2304A022010	SYS_C00798284	NORMAL	ISIS2304A022010	SEGURO	TABLE	UNIQUE	DISABLED
12	ISIS2304A022010	SYS_C00798285	NORMAL	ISIS2304A022010	SERVICIOS_ALOJAMIENTOS	TABLE	UNIQUE	DISABLED
13	ISIS2304A022010	SYS_C00798286	NORMAL	ISIS2304A022010	SERVICIO	TABLE	UNIQUE	DISABLED
14	ISIS2304A022010	SYS_C00798287	NORMAL	ISIS2304A022010	HABITACION	TABLE	UNIQUE	DISABLED
15	ISIS2304A022010	UN_USUARIO_USUARIO	NORMAL	ISIS2304A022010	USUARIO	TABLE	UNIQUE	DISABLED
16	ISIS2304A022010	UN_USUARIO_CORREO	NORMAL	ISIS2304A022010	USUARIO	TABLE	UNIQUE	DISABLED
17	ISIS2304A022010	UN_USUARIO_NUMERO_DOCUMENTO	NORMAL	ISIS2304A022010	USUARIO	TABLE	UNIQUE	DISABLED

Figura 3. Indices generados por oracle

Estos se crean a partir de las llaves primarias de las tablas y según se puede observar en la figura 3, el atributo uniqueness indica si son arboles b+. Adicional a estos que se crean, se puede ver como también se crean varios índices sobre la tabla usuarios, la cual recibe la mayoría de las consultas.

3.1 Análisis requerimientos

3.1.1 RFC10

- La sentencia evaluada fue:

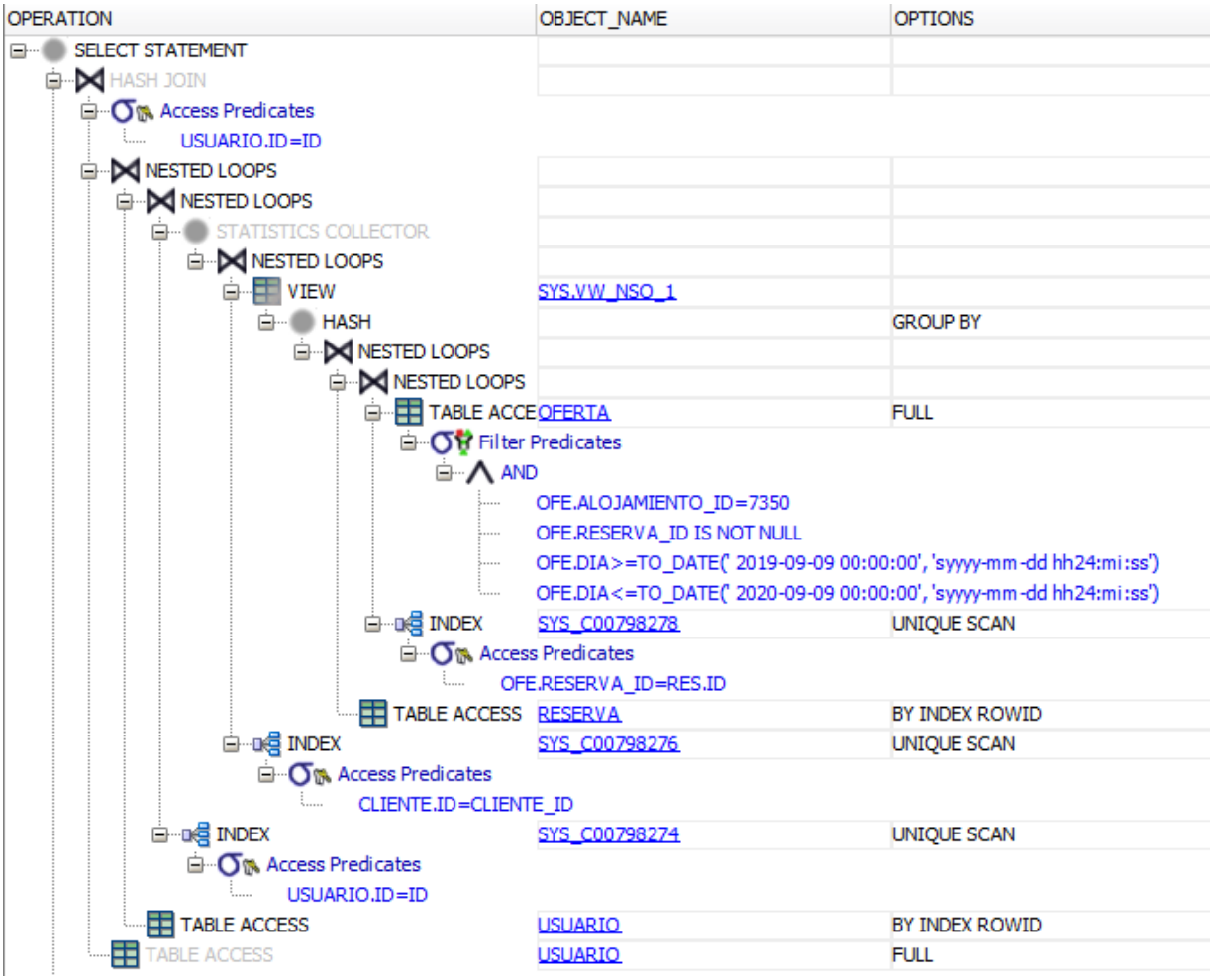
```
SELECT * FROM USUARIO NATURAL JOIN (
SELECT ID FROM CLIENTE WHERE CLIENTE.id IN(
SELECT res.cliente_id FROM OFERTA ofe
JOIN ALOJAMIENTO al ON ofe.alojamiento_id=al.id
JOIN RESERVA res ON ofe.reserva_id=res.id
WHERE ofe.alojamiento_id=7350 AND ofe.dia>='09/09/2019' AND ofe.dia<=
    '09/09/2020'
GROUP BY res.cliente_id));
```

- La selectividad de este requerimiento tiende a ser baja ya que según la distribución de los datos generados, no hay muchos usuarios por cada alojamiento.

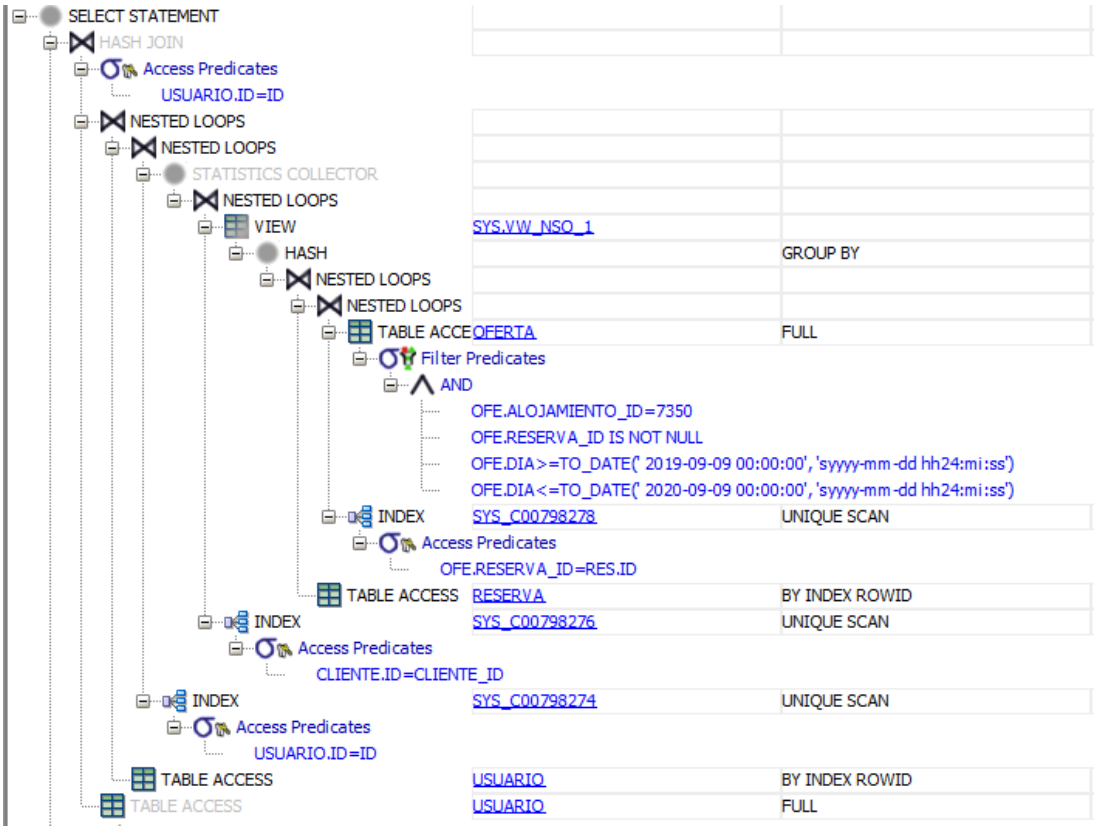
- Comparación tiempo:

SQLDeveloper	0,12 segundos
Indices generados por Oracle en app	5 segundos
Indices generados por nosotros en app	5 segundos

- Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:



- Se implementaron los indices I_alojamiento_operador_id y I_reserva_cliente_id, pero no tuvieron ningun efecto ya que según se puede observar, el plan de ejecucion de la consulta es el mismo y no se utilizan los indices generados.



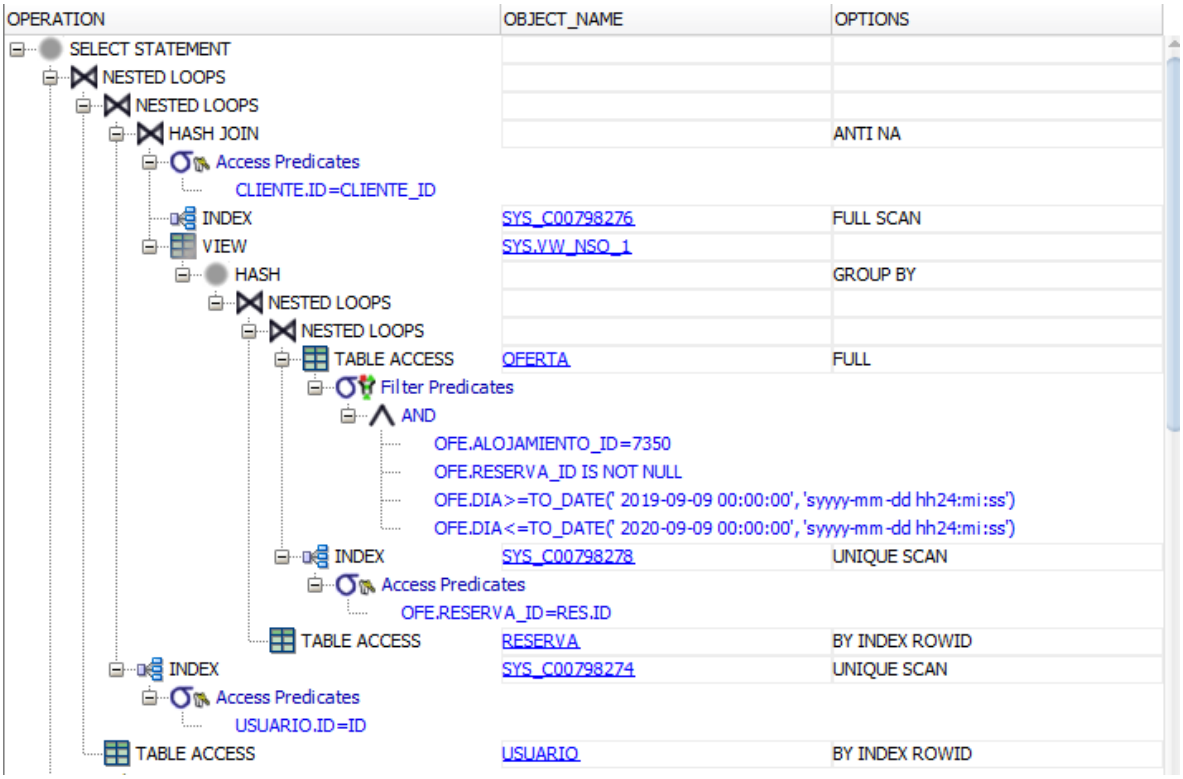
3.1.2 RFC11

- La sentencia evaluada fue:

```
SELECT * FROM USUARIO NATURAL JOIN (
SELECT ID FROM CLIENTE WHERE CLIENTE.id NOT IN(
SELECT res.cliente_id FROM OFERTA ofe
JOIN ALOJAMIENTO al ON ofe.alojamiento_id=al.id
JOIN RESERVA res ON ofe.reserva_id=res.id
WHERE ofe.alojamiento_id=7350 AND ofe.dia>='09/09/2019' AND ofe.dia<=
    '09/09/2020'
GROUP BY res.cliente_id));
```

- El tamaño de la respuesta es problemático en este requerimiento ya que la selectividad es muy alta. Todos los datos que no están en el RFC10 están en este requerimiento. Por lo tanto la respuesta indica la mayoría de los usuarios(99%).

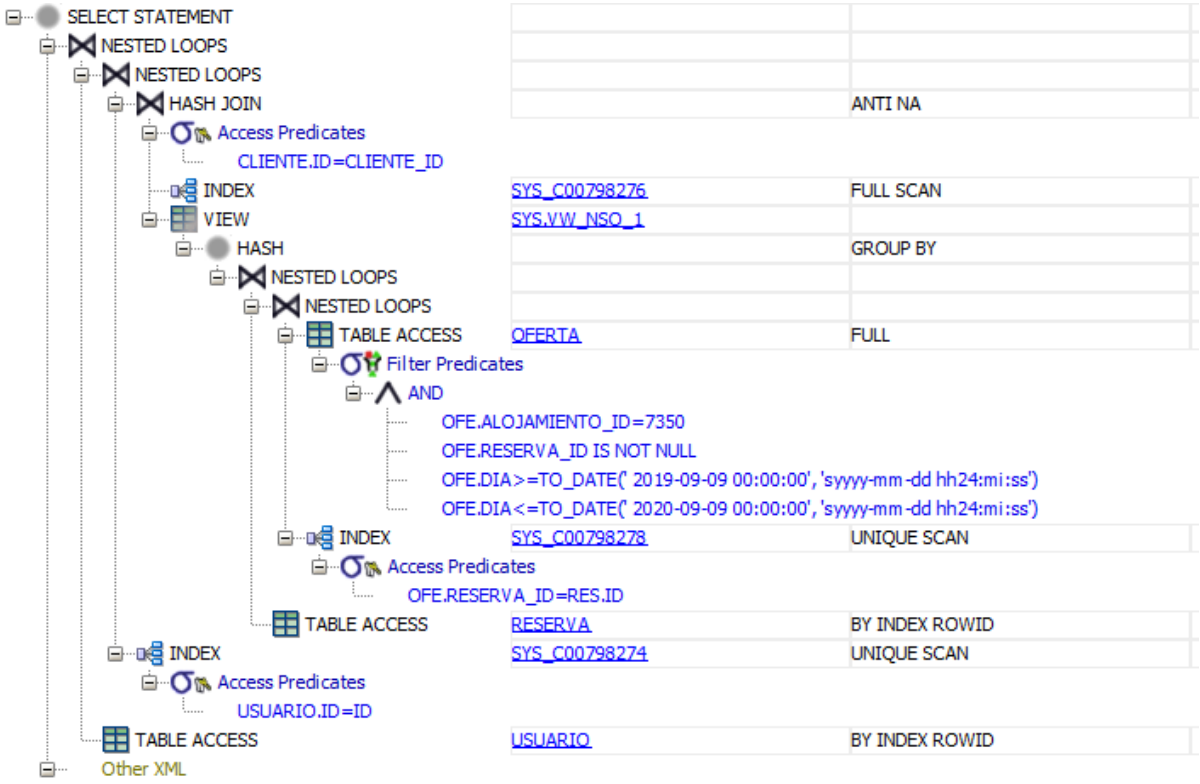
- Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:



- Comparación tiempo:

SQLDeveloper	4 minutos
Indices generados por Oracle en app	10 minutos
Indices generados por nosotros en app	10 minutos

- Se implementaron los indices I_alojamiento_operador_id y I_reserva_cliente_id, pero no tuvieron ningun efecto ya que según se puede observar, el plan de ejecucion de la consulta es el mismo



3.1.3 RFC12

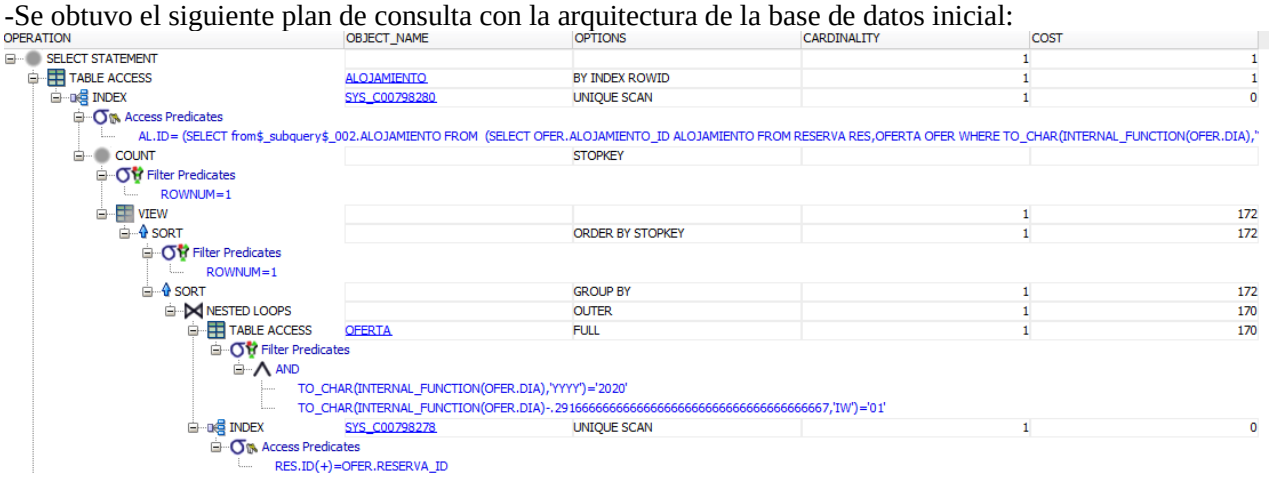
Para este requerimiento se usaron un total de 4 consultas con las siguientes sentencias SQL y pl**anes de consulta.**

-La sentencia evaluada fue:

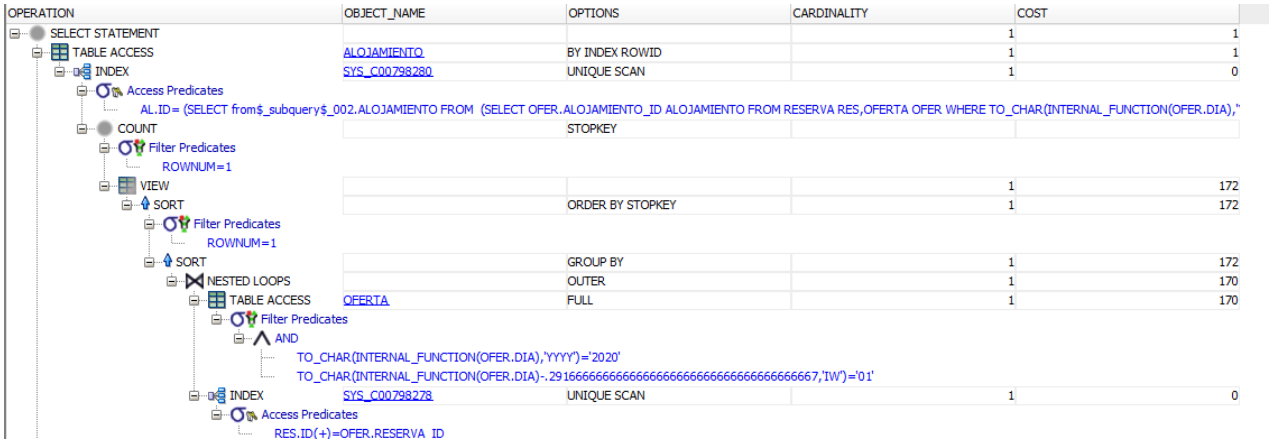
```
SELECT * FROM ALOJAMIENTO AL
WHERE AL.ID = (
SELECT * FROM(
SELECT OFER.ALOJAMIENTO_ID AS ALOJAMIENTO
FROM ALOJAMIENTO ALO
FULL OUTER JOIN OFERTA OFER ON OFER.ALOJAMIENTO_ID = ALO.ID
FULL OUTER JOIN RESERVA RES ON RES.ID = OFER.RESERVA_ID
WHERE TO_CHAR(OFER.DIA,'YYYY') LIKE '2020'
AND TO_CHAR(DIA - 7/24,'IW') LIKE '01'
GROUP BY OFER.ALOJAMIENTO_ID, TO_CHAR(DIA - 7/24,'IW')
ORDER BY COUNT(RES.ID) ASC)
WHERE ROWNUM = 1);
```

La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos en un máximo de 365 días, los datos van a tender a ser muy parecidos.

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:



-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos con índices:



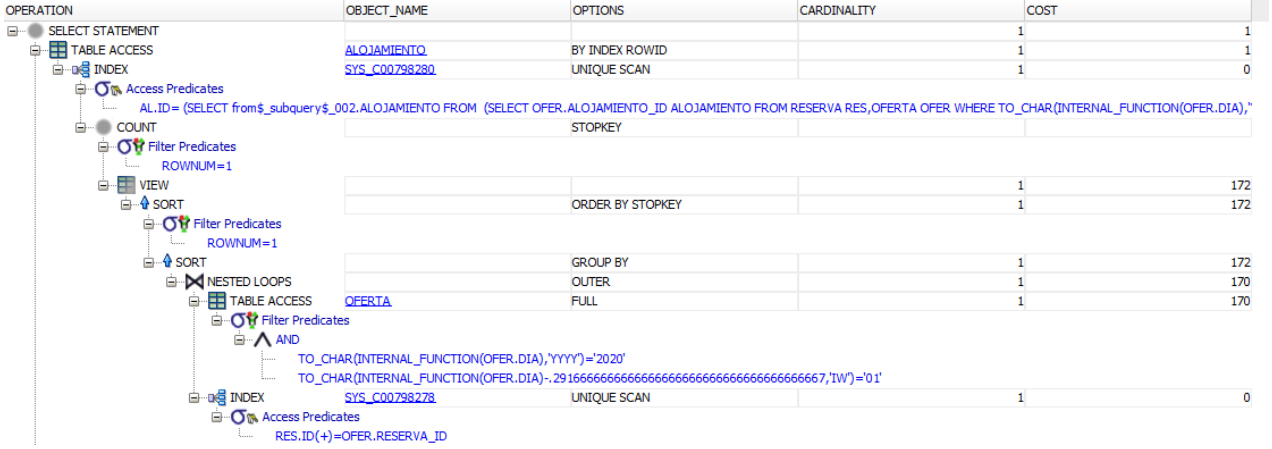
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
TABLE ACCESS	ALOJAMIENTO	BY INDEX ROWID	1	1
INDEX	SYS_C00798280	UNIQUE SCAN	1	0
Access Predicates				
AL.ID= (SELECT from$_subquery$_002.ALOJAMIENTO FROM (SELECT OFER.ALOJAMIENTO_ID ALOJAMIENTO FROM RESERVA RES,OFERTA OFER WHERE TO_CHAR(INTERNAL_FUNCTION(OFER.DIA),'				
COUNT		STOPKEY		
Filter Predicates				
ROWNUM=1				
VIEW			1	172
SORT		ORDER BY STOPKEY	1	172
Filter Predicates				
ROWNUM=1				
SORT		GROUP BY	1	172
NESTED LOOPS		OUTER	1	170
TABLE ACCESS	OFERTA	FULL	1	170
Filter Predicates				
AND				
TO_CHAR(INTERNAL_FUNCTION(OFER.DIA),'YYYY')='2020'				
TO_CHAR(INTERNAL_FUNCTION(OFER.DIA)-.291666666666666666666666666666666666667,'IW')='01'				
INDEX	SYS_C00798278	UNIQUE SCAN	1	0
Access Predicates				
RES.ID(+)=OFER.RESERVA_ID				

-La sentencia evaluada fue:

```
SELECT * FROM ALOJAMIENTO AL
WHERE AL.ID = (
SELECT * FROM(
SELECT OFER.ALOJAMIENTO_ID AS ALOJAMIENTO
FROM ALOJAMIENTO ALO
FULL OUTER JOIN OFERTA OFER ON OFER.ALOJAMIENTO_ID = ALO.ID
FULL OUTER JOIN RESERVA RES ON RES.ID = OFER.RESERVA_ID
WHERE TO_CHAR(OFER.DIA,'YYYY') LIKE '2020'
AND TO_CHAR(DIA - 7/24,'IW') LIKE '01'
GROUP BY OFER.ALOJAMIENTO_ID, TO_CHAR(DIA - 7/24,'IW')
ORDER BY COUNT(RES.ID) DESC)
WHERE ROWNUM = 1);
```

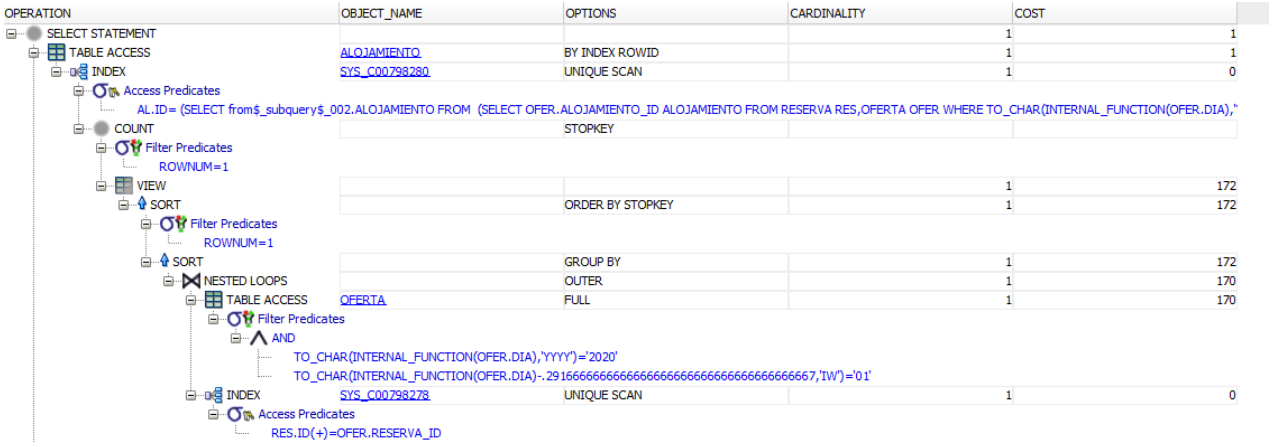
-La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos en un máximo de 365 días, los datos van a tender a ser muy parecidos.

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
TABLE ACCESS	ALOJAMIENTO	BY INDEX ROWID	1	1
INDEX	SYS_C00798280	UNIQUE SCAN	1	0
Access Predicates				
AL.ID= (SELECT from$_subquery$_002.ALOJAMIENTO FROM (SELECT OFER.ALOJAMIENTO_ID ALOJAMIENTO FROM RESERVA RES,OFERTA OFER WHERE TO_CHAR(INTERNAL_FUNCTION(OFER.DIA),'				
COUNT		STOPKEY		
Filter Predicates				
ROWNUM=1				
VIEW			1	172
SORT		ORDER BY STOPKEY	1	172
Filter Predicates				
ROWNUM=1				
SORT		GROUP BY	1	172
NESTED LOOPS		OUTER	1	170
TABLE ACCESS	OFERTA	FULL	1	170
Filter Predicates				
AND				
TO_CHAR(INTERNAL_FUNCTION(OFER.DIA),'YYYY')='2020'				
TO_CHAR(INTERNAL_FUNCTION(OFER.DIA)-.291666666666666666666666666666666666667,'IW')='01'				
INDEX	SYS_C00798278	UNIQUE SCAN	1	0
Access Predicates				
RES.ID(+)=OFER.RESERVA_ID				

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos con índices:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
TABLE ACCESS	ALOJAMIENTO	BY INDEX ROWID	1	1
INDEX	SYS_C00798280	UNIQUE SCAN	1	0
Access Predicates				
AL.ID= (SELECT from$_subquery$_002.ALOJAMIENTO FROM (SELECT OFER.ALOJAMIENTO_ID ALOJAMIENTO FROM RESERVA RES,OFERTA OFER WHERE TO_CHAR(INTERNAL_FUNCTION(OFER.DIA),'				
COUNT		STOPKEY		
Filter Predicates				
ROWNUM=1				
VIEW			1	172
SORT		ORDER BY STOPKEY	1	172
Filter Predicates				
ROWNUM=1				
SORT		GROUP BY	1	172
NESTED LOOPS		OUTER	1	170
TABLE ACCESS	OFERTA	FULL	1	170
Filter Predicates				
AND				
TO_CHAR(INTERNAL_FUNCTION(OFER.DIA),'YYYY')='2020'				
TO_CHAR(INTERNAL_FUNCTION(OFER.DIA)-.291666666666666666666666666666666666667,'IW')='01'				
INDEX	SYS_C00798278	UNIQUE SCAN	1	0
Access Predicates				
RES.ID(+)=OFER.RESERVA_ID				

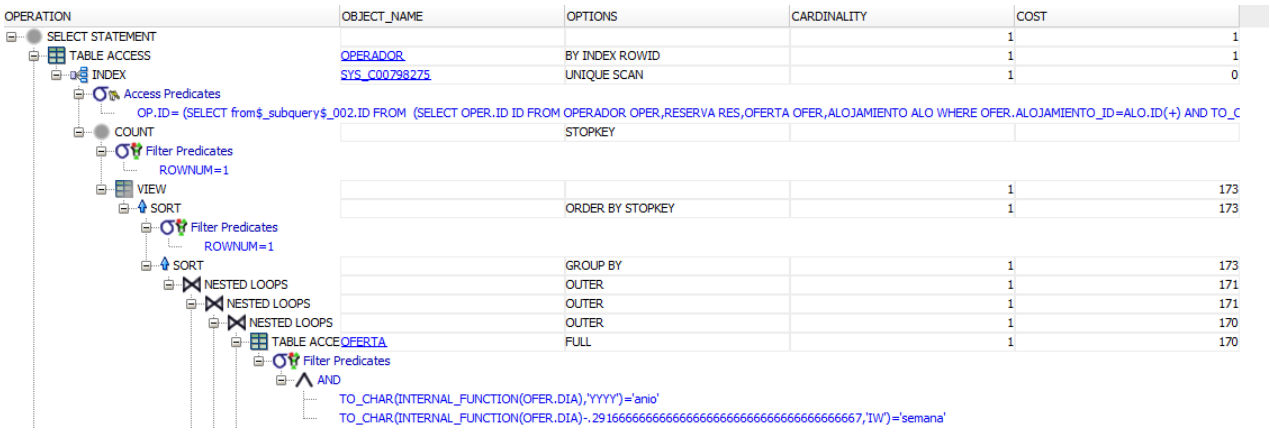
-La sentencia evalu**ada fue:**

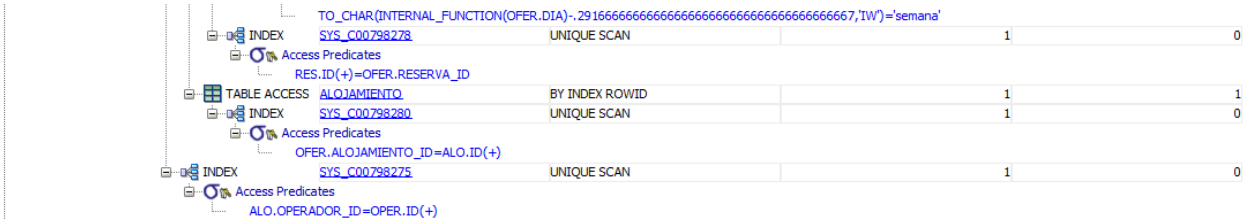
```
SELECT * FROM OPERADOR OP
WHERE OP.ID = (
SELECT * FROM(
SELECT OPER.ID FROM ALOJAMIENTO ALO
FULL OUTER JOIN OFERTA OFER ON OFER.ALOJAMIENTO_ID = ALO.ID
FULL OUTER JOIN RESERVA RES ON RES.ID = OFER.RESERVA_ID
FULL OUTER JOIN OPERADOR OPER ON ALO.OPERADOR_ID = OPER.ID
WHERE TO_CHAR(OFER.DIA,'YYYY') LIKE 'anio'
AND TO_CHAR(DIA - 7/24,'IW') LIKE 'semana'
GROUP BY OPER.ID, TO_CHAR(DIA - 7/24,'IW')
ORDER BY COUNT(RES.ID) ASC)
WHERE ROWNUM = 1);
```

- La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos en un máximo de 365 días, los datos van a tender a ser muy parecidos.

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos con índices:

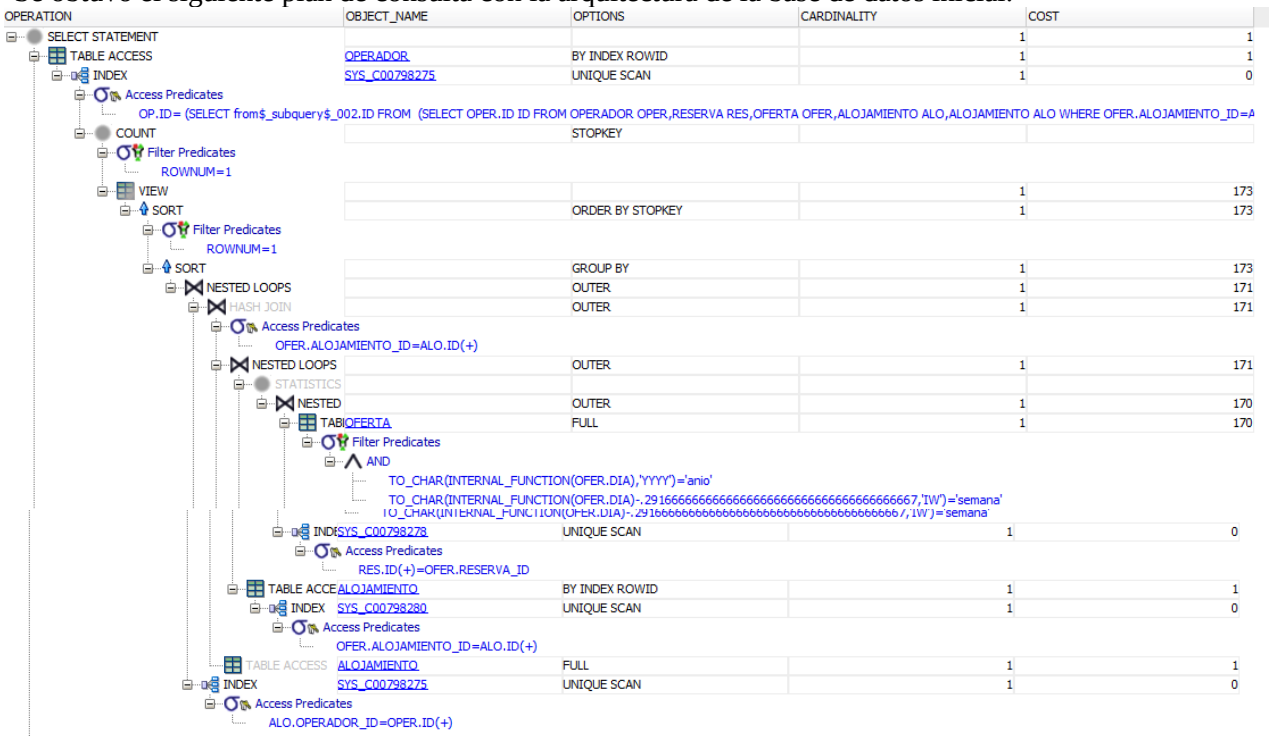




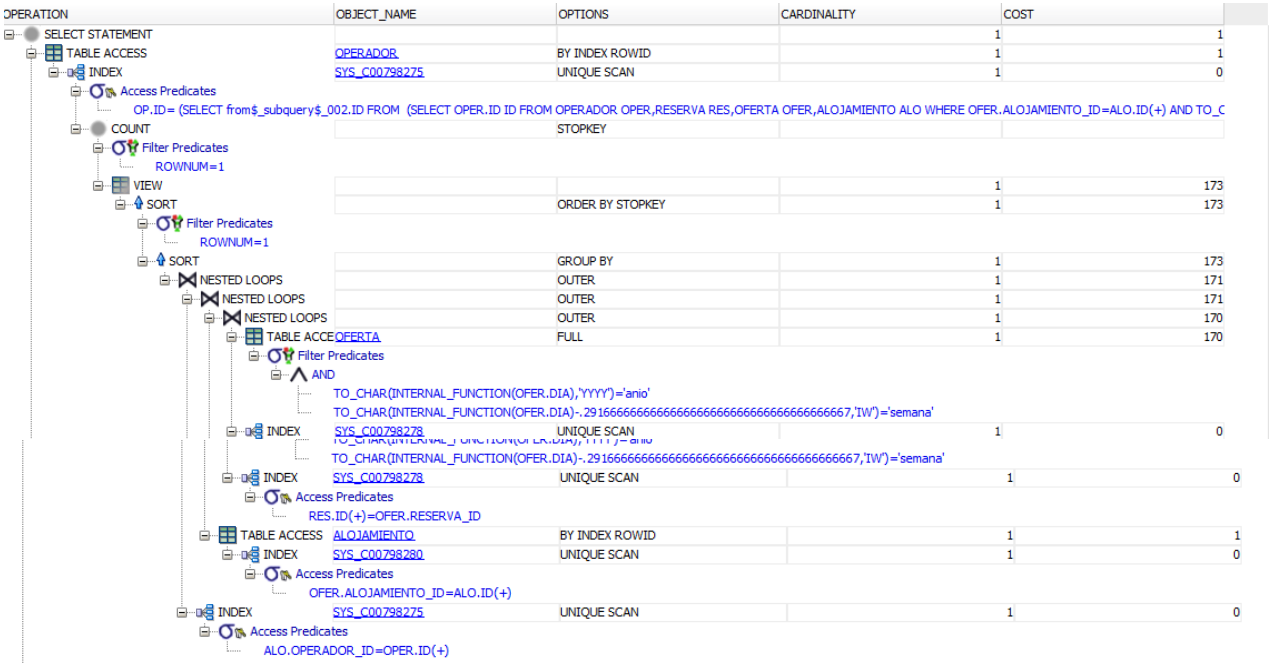

```
SELECT * FROM OPERADOR OP
WHERE OP.ID = (
SELECT * FROM(
SELECT OPER.ID FROM ALOJAMIENTO ALO
FULL OUTER JOIN OFERTA OFER ON OFER.ALOJAMIENTO_ID = ALO.ID
FULL OUTER JOIN RESERVA RES ON RES.ID = OFER.RESERVA_ID
FULL OUTER JOIN OPERADOR OPER ON ALO.OPERADOR_ID = OPER.ID
WHERE TO_CHAR(OFER.DIA,'YYYY') LIKE 'anio'
AND TO_CHAR(DIA - 7/24,'IW') LIKE 'semana'
GROUP BY OPER.ID, TO_CHAR(DIA - 7/24,'IW')
ORDER BY COUNT(RES.ID) DESC)
WHERE ROWNUM = 1);
```

- La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos en un máximo de 365 días, los datos van a tender a ser muy parecidos.

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:



-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos con índices:



Resultados obtenidos en aplicación:

# Prueba	Tiempo (ms)
1	92802
2	62540
3	69749
4	75970
5	66395

Resultados obtenidos en aplicación con índices:

# Prueba	Tiempo (ms)
1	60434
2	53860
3	53954
4	55157
5	52631

Se hizo la inserción de un único índice en el día de la reserva.

3.1.4 RFC13

Para este requerimiento se usaron un total de 3 consultas con las siguien**tes sentencias SQL y planes de consulta.**

-La sentencia evaluada fue:

```
SELECT CLI.* FROM CLIENTE CLI
INNER JOIN RESERVA RES ON CLI.ID = RES.CLIENTE_ID
WHERE TO_CHAR(RES.FECHA_REALIZACION,'YYYY-MM') LIKE '2020-05'
AND ROWNUM < 100;
```

-La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos en un máximo de 30 días, los datos van a tender a ser muy parecidos.

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			99	312
COUNT		STOPKEY		
Filter Predicates				
ROWNUM<100				
MERGE JOIN			20725	312
TABLE ACCESS	CLIENTE	BY INDEX ROWID	51	3
INDEX	SYS_C00798276	FULL SCAN	51	1
SORT		JOIN	20725	309
Access Predicates				
CLI.ID=RES.CLIENTE_ID				
Filter Predicates				
CLI.ID=RES.CLIENTE_ID				
TABLE ACCESS	RESERVA	FULL	20725	307
Filter Predicates				
TO_CHAR(INTERNAL_FUNCTION(RES.FECHA_REALIZACION),'YYYY-MM')='2020-05'				

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos con índices:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			99	312
COUNT		STOPKEY		
Filter Predicates				
ROWNUM<100				
MERGE JOIN			20725	312
TABLE ACCESS	CLIENTE	BY INDEX ROWID	51	3
INDEX	SYS_C00798276	FULL SCAN	51	1
SORT		JOIN	20725	309
Access Predicates				
CLI.ID=RES.CLIENTE_ID				
Filter Predicates				
CLI.ID=RES.CLIENTE_ID				
TABLE ACCESS	RESERVA	FULL	20725	307
Filter Predicates				
TO_CHAR(INTERNAL_FUNCTION(RES.FECHA_REALIZACION),'YYYY-MM')='2020-05'				

-La sentencia evaluada fue:

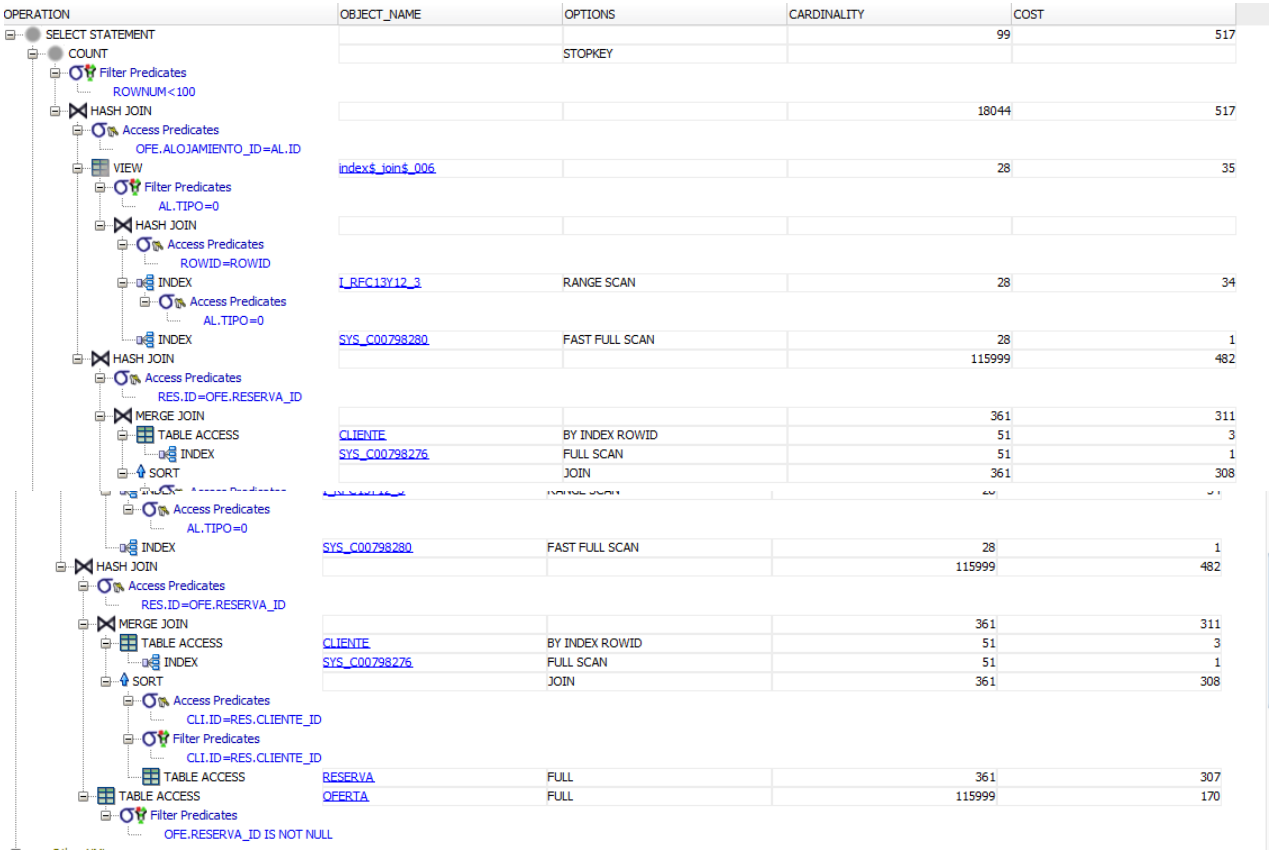
```
SELECT CLI.* FROM CLIENTE CLI
INNER JOIN RESERVA RES ON CLI.ID = RES.CLIENTE_ID
INNER JOIN OFERTA OFE ON RES.ID = OFE.RESERVA_ID
INNER JOIN ALOJAMIENTO AL ON OFE.ALOJAMIENTO_ID = AL.ID
WHERE AL.TIPO = 0
AND ROWNUM < 100;
```

- La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos y solo tres distintas posibilidades, los datos van a tender a ser muy parecidos con respecto al tipo.

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			99	891
COUNT		STOPKEY		
Filter Predicates				
ROWNUM<100				
HASH JOIN			18044	891
Access Predicates				
OFE.ALOJAMIENTO_ID=AL.ID				
TABLE ACCESS	ALOJAMIENTO	FULL	28	409
Filter Predicates				
AL.TIPO=0				
HASH JOIN			115999	482
Access Predicates				
RES.ID=OFE.RESERVA_ID				
MERGE JOIN			361	311
TABLE ACCESS	CLIENTE	BY INDEX ROWID	51	3
INDEX	SYS_C00798276	FULL SCAN	51	1
SORT		JOIN	361	308
Access Predicates				
CLI.ID=RES.CLIENTE_ID				
Filter Predicates				
CLI.ID=RES.CLIENTE_ID				
TABLE ACCESS	RESERVA	FULL	361	307
TABLE ACCESS	OFERTA	FULL	115999	170
Filter Predicates				
OFE.RESERVA_ID IS NOT NULL				

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos con índices:



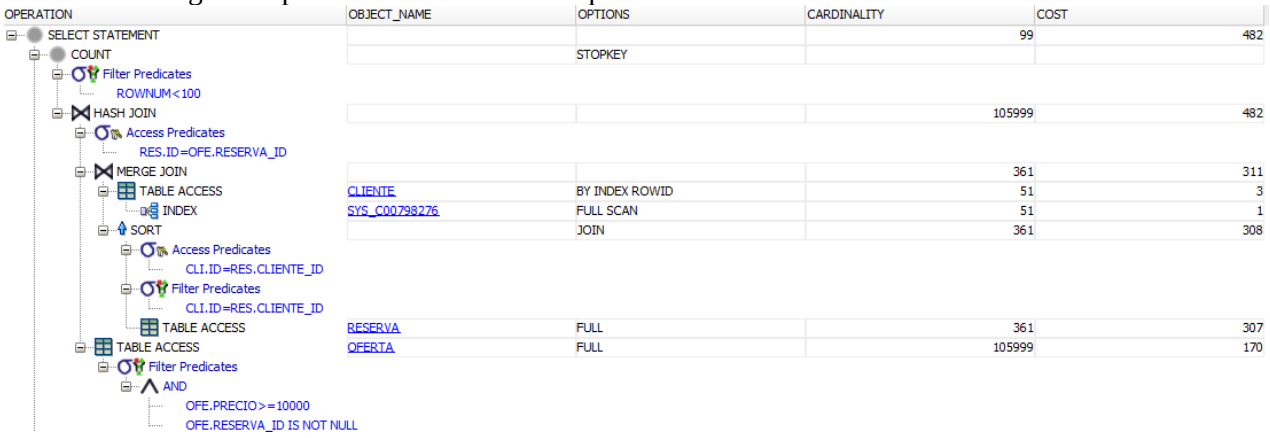
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			99	517
COUNT		STOPKEY		
Filter Predicates				
ROWNUM<100				
HASH JOIN			18044	517
Access Predicates				
OFE.ALOJAMIENTO_ID=AL.ID				
VIEW	index$_join$_006		28	35
Filter Predicates				
AL.TIPO=0				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	I_RFC13Y12_3	RANGE SCAN	28	34
Access Predicates				
AL.TIPO=0				
INDEX	SYS_C00798280	FAST FULL SCAN	28	1
HASH JOIN			115999	482
Access Predicates				
RES.ID=OFE.RESERVA_ID				
MERGE JOIN			361	311
TABLE ACCESS	CLIENTE	BY INDEX ROWID	51	3
INDEX	SYS_C00798276	FULL SCAN	51	1
SORT		JOIN	361	308
Access Predicates				
AL.TIPO=0				
INDEX	SYS_C00798280	FAST FULL SCAN	28	1
HASH JOIN			115999	482
Access Predicates				
RES.ID=OFE.RESERVA_ID				
MERGE JOIN			361	311
TABLE ACCESS	CLIENTE	BY INDEX ROWID	51	3
INDEX	SYS_C00798276	FULL SCAN	51	1
SORT		JOIN	361	308
Access Predicates				
CLI.ID=RES.CLIENTE_ID				
Filter Predicates				
CLI.ID=RES.CLIENTE_ID				
TABLE ACCESS	RESERVA	FULL	361	307
TABLE ACCESS	OFERTA	FULL	115999	170
Filter Predicates				
OFE.RESERVA_ID IS NOT NULL				

-La sentencia evaluada fue:

```
SELECT CLI.* FROM CLIENTE CLI
INNER JOIN RESERVA RES ON CLI.ID = RES.CLIENTE_ID
INNER JOIN OFERTA OFE ON RES.ID = OFE.RESERVA_ID
WHERE OFE.PRECIO >= 10000
AND ROWNUM < 100;
```

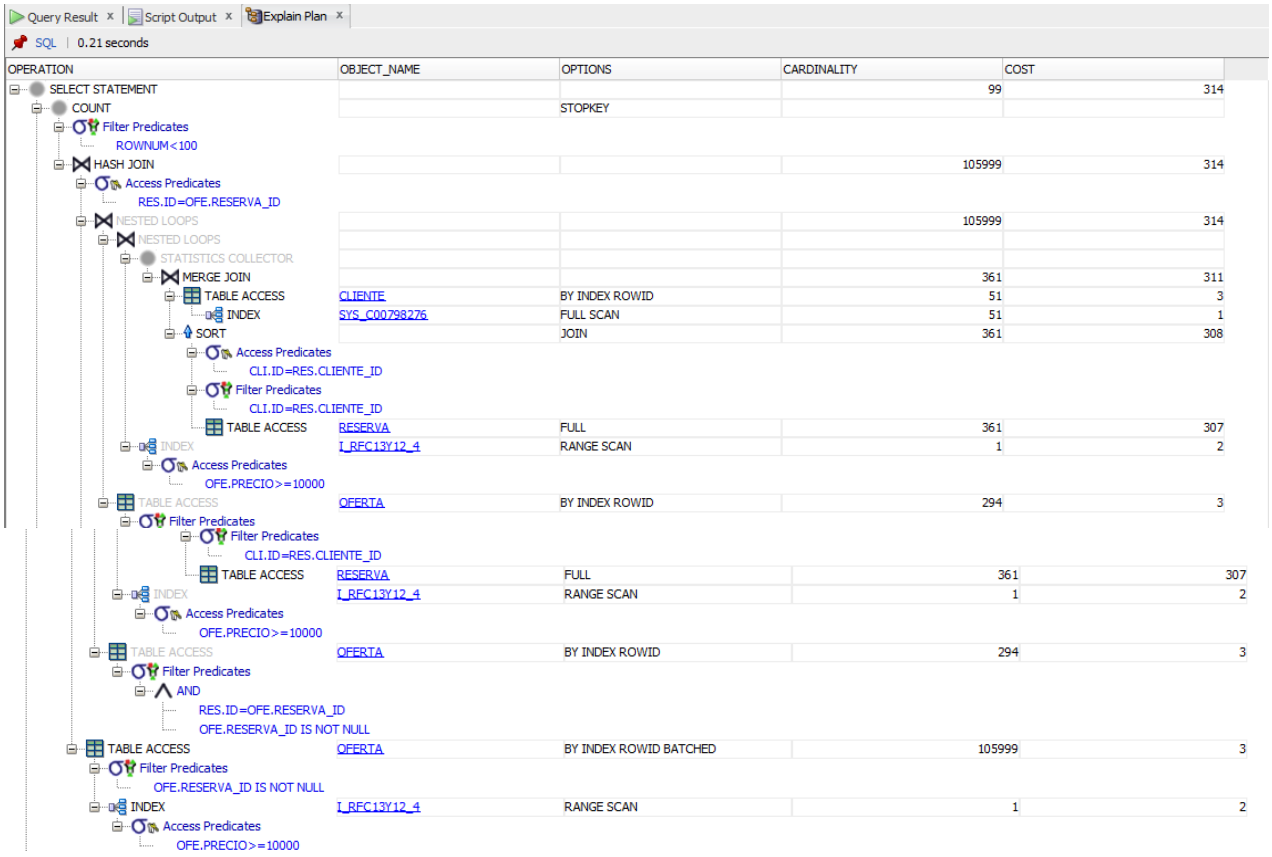
-La selectividad de esta consulta tiende a ser alta, ya que, debido a la distribución con la que se insertaron los datos (id = precio), los datos van a ser distintos siempre.

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			99	482
COUNT		STOPKEY		
Filter Predicates				
ROWNUM<100				
HASH JOIN			105999	482
Access Predicates				
RES.ID=OFE.RESERVA_ID				
MERGE JOIN			361	311
TABLE ACCESS	CLIENTE	BY INDEX ROWID	51	3
INDEX	SYS_C00798276	FULL SCAN	51	1
SORT		JOIN	361	308
Access Predicates				
CLI.ID=RES.CLIENTE_ID				
Filter Predicates				
CLI.ID=RES.CLIENTE_ID				
TABLE ACCESS	RESERVA	FULL	361	307
TABLE ACCESS	OFERTA	FULL	105999	170
Filter Predicates				
AND				
OFE.PRECIO>=10000				
OFE.RESERVA_ID IS NOT NULL				

-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos con índices:



Resultados obtenidos en aplicación:

# Prueba	Tiempo (ms)
1	3190
2	2887
3	4421
4	3033
5	2494

Resultados obtenidos en aplicación con índices:

# Prueba	Tiempo (ms)
1	2673
2	2100
3	2330
4	3443
5	2362

Se hizo la inserción de tres índices en el tipo del alojamiento, costo de la oferta y en la fecha de realización de la reserva.

4 Conclusiones

A partir de los resultados presentados, se puede concluir que el trabajo que hace el SMBD es óptimo y a pesar de las modificaciones que se hagan a los índices que ya existen, no se va a obtener un mejor resultado que el que se propone, por más que en algunas consultas se hagan uso de algunos de los indices insertados.

Por otro lado, se puede observar como las consultas en SQLDeveloper son mucho más rápidas que cuando se llevan a memoria en la aplicación. Es precisamente en ese proceso de lectura en memoria secundaria donde se pasa a memoria principal, en que se dan este tipo de demoras.