**Documento de Informe - Iteración 3**

Juan Esteban Berdugo González, Oscar Castañeda Universidad de los Andes, Bogotá, Colombia

{je.berdugo10, [oj.castaneda}@uniandes.edu.co](mailto:oj.castaneda%7D@uniandes.edu.co) Fecha de presentación: Mayo 3 de 2020

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Contenido  [1 Introducción 1](#_Toc41420383)  [2 Modelos 1](#_Toc41420384)  [2.1 Modelo Conceptual 1](#_Toc41420385)  [2.2 Modelo de datos relacional 2](#_Toc41420386)  [3 Diseño Físico 3](#_Toc41420387)  [3.1 Análisis requerimientos 3](#_Toc41420388)  [3.1.1 RFC10 3](#_Toc41420389)  [3.1.2 RFC11 5](#_Toc41420390)  [3.1.3 RFC12 6](#_Toc41420391)  [3.1.4 RFC13 9](#_Toc41420392) |  |  |
|  |  |  |
|  |  |  |

# 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.

# Modelos

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

## Modelo Conceptual

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

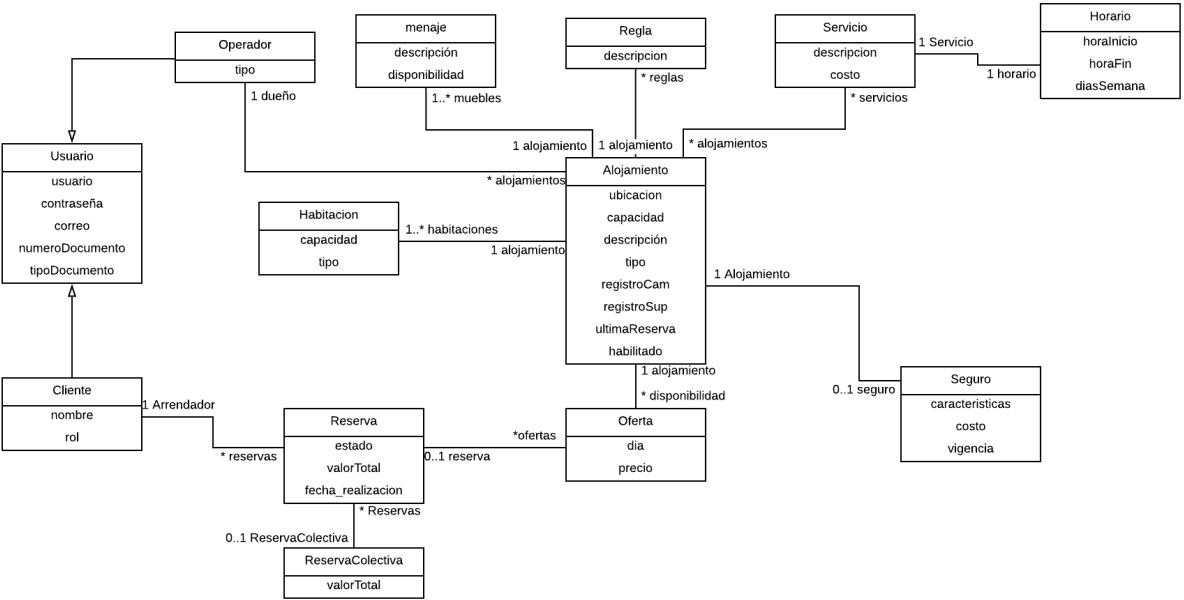


Figura 1. Diagrama UML – Modelo Conceptual AlohAndes

## Modelo de datos relacional

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

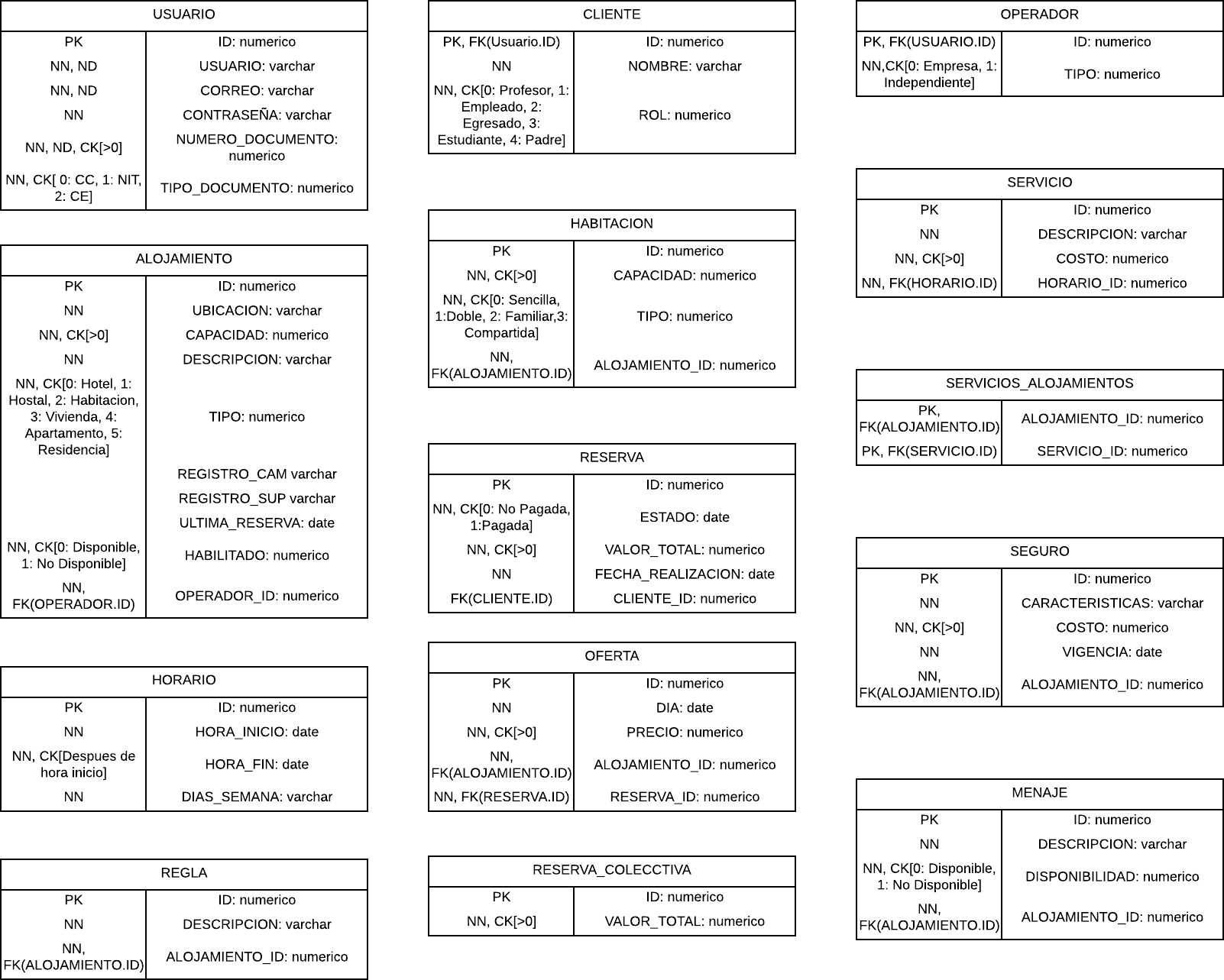


Figura 2. Modelo relacional AlohAndes

# Diseño Físico

Los índices generados por Oracle son los siguientes:

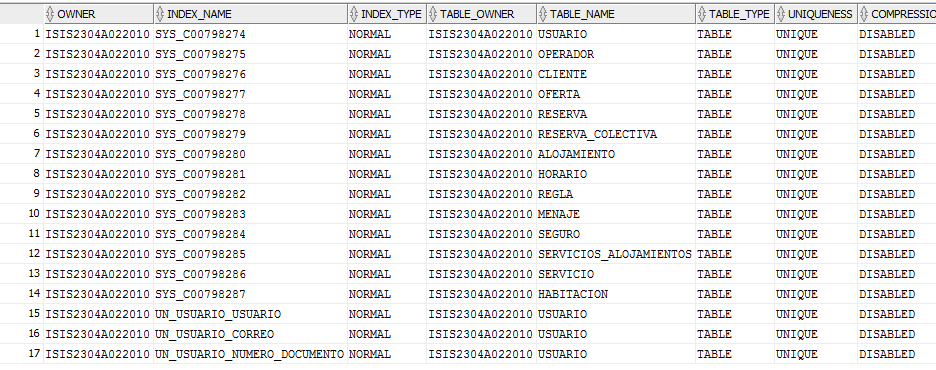


Figura 3. Indices generados por oracle

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.

# 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

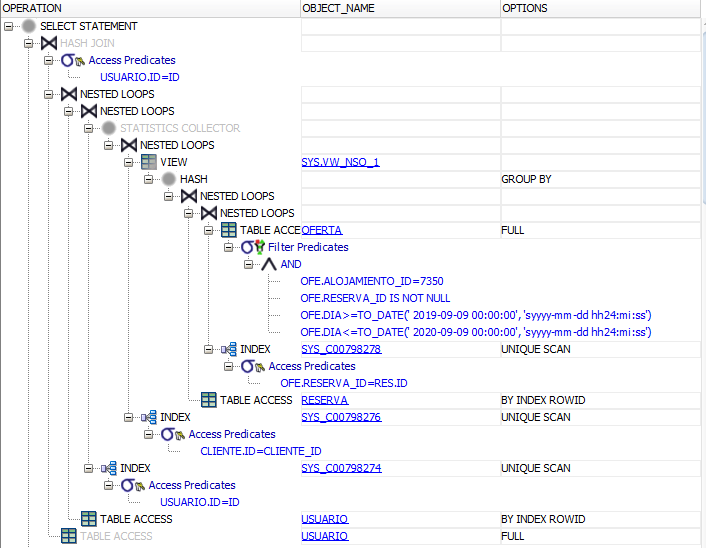
WHERE ofe.alojamiento\_id=7350 AND ofe.dia>='09/09/2019' AND ofe.dia<= '09/09/2020'

GROUP BY res.cliente\_id));

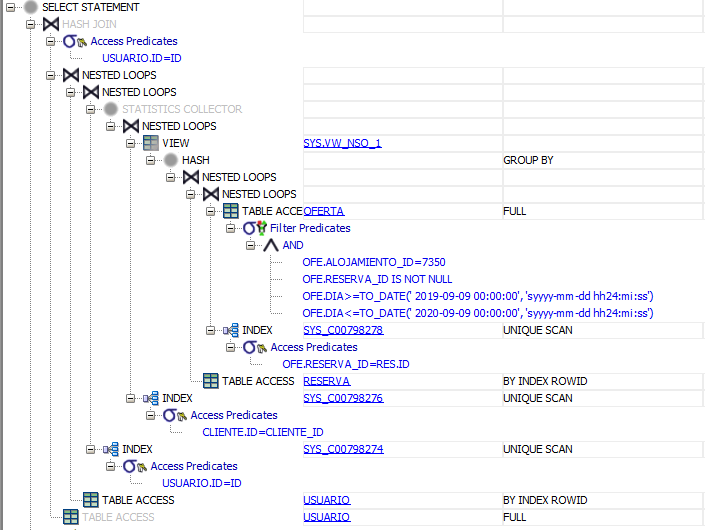
* La selectividad de este requerimiento tiende a ser baja ya que según la distribución de los datos generados, no hay muchos usuarios por cada alojamiento.
* Comparación tiempo:

|  |  |
| --- | --- |
| 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

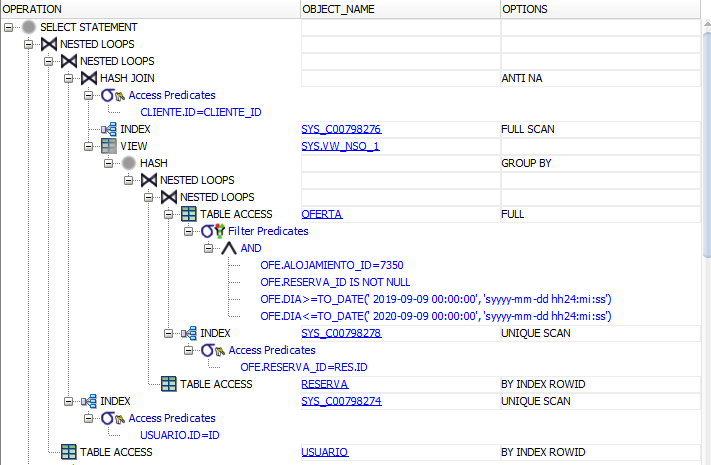
JOIN ALOJAMIENTO al ON ofe.alojamiento\_id=al.id

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'

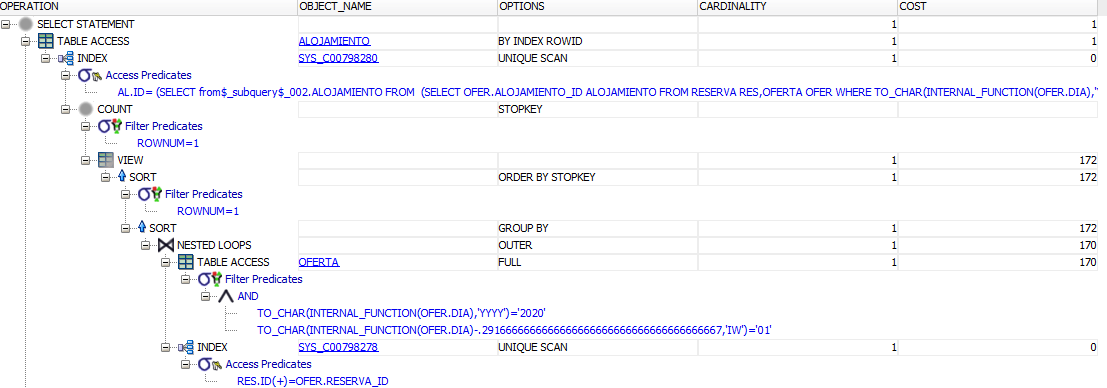
GROUP BY OFER.ALOJAMIENTO\_ID, TO\_CHAR(DIA - 7/24,'IW')

ORDER BY COUNT(RES.ID) ASC)

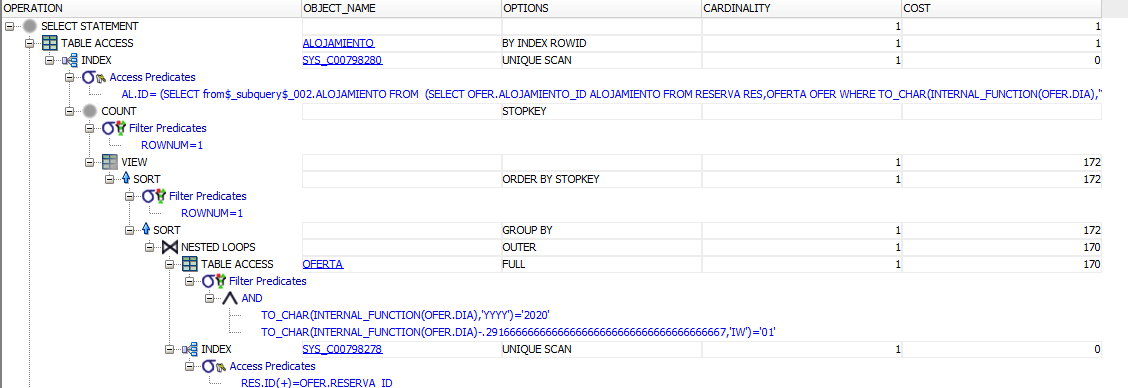
WHERE ROWNUM = 1);

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:



-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'

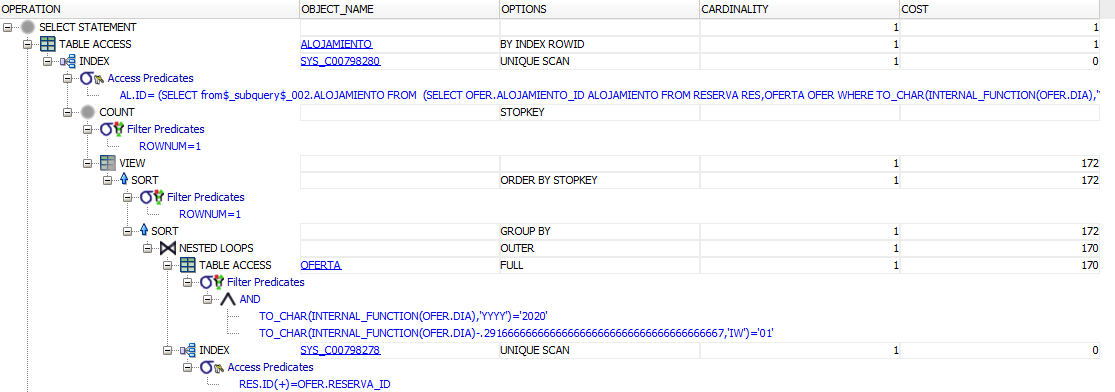
GROUP BY OFER.ALOJAMIENTO\_ID, TO\_CHAR(DIA - 7/24,'IW')

ORDER BY COUNT(RES.ID) DESC)

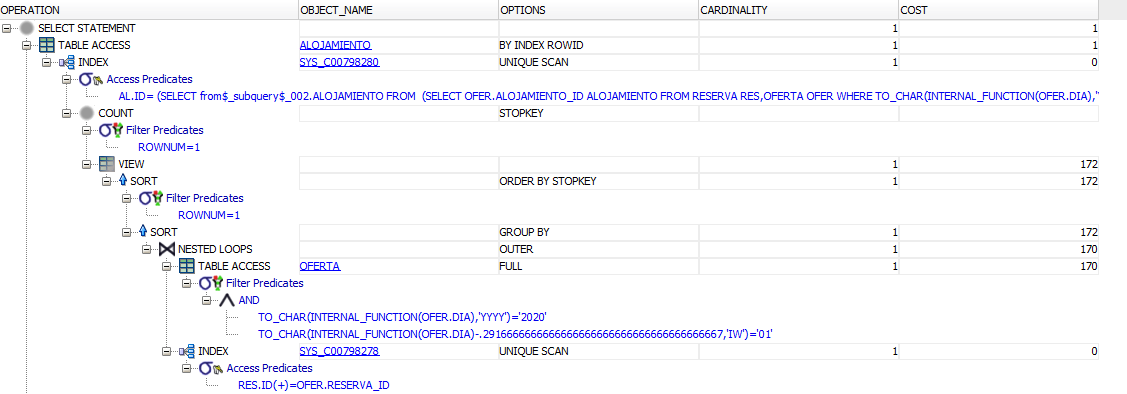
WHERE ROWNUM = 1);

-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:



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



-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'

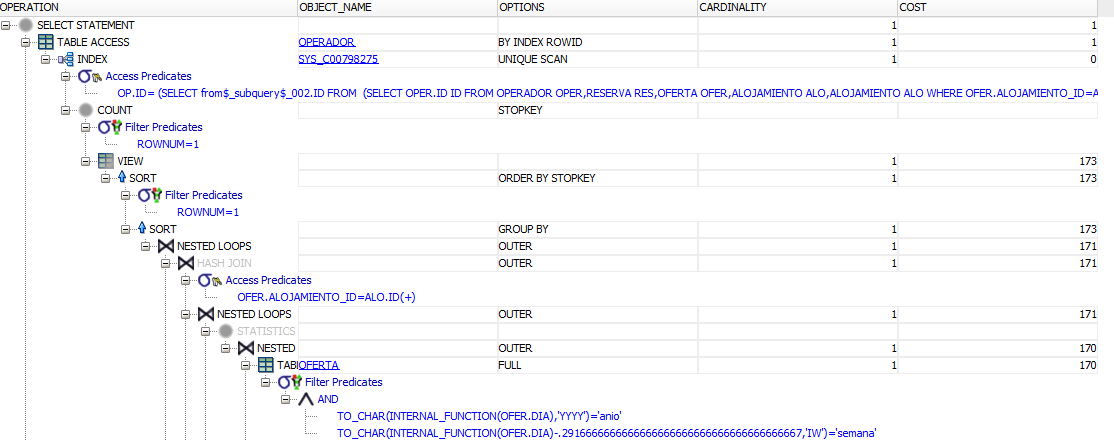
GROUP BY OPER.ID, TO\_CHAR(DIA - 7/24,'IW')

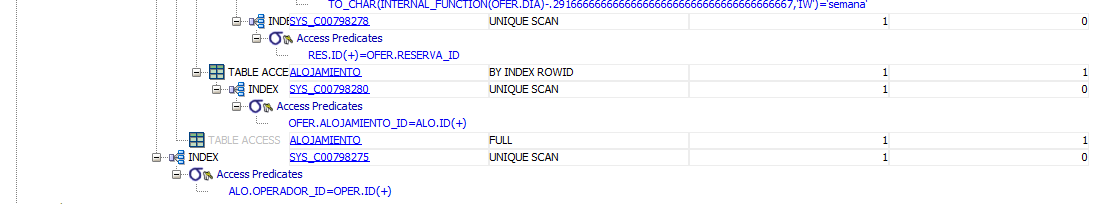
ORDER BY COUNT(RES.ID) ASC)

WHERE ROWNUM = 1);

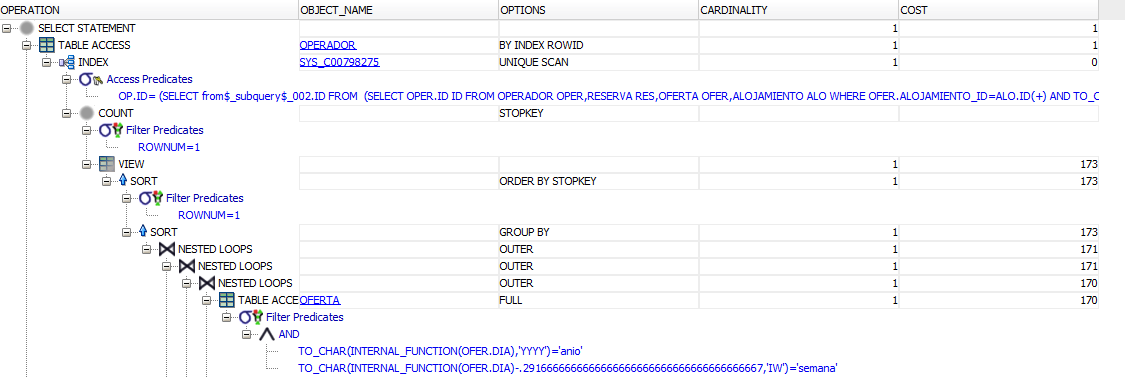
- La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos en un máximo de 365 días, los datos van a tender a ser muy parecidos.

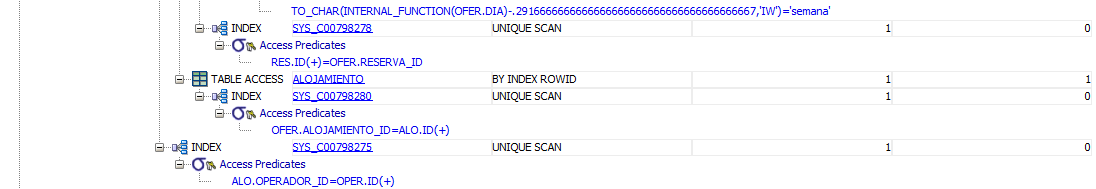
-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:





-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'

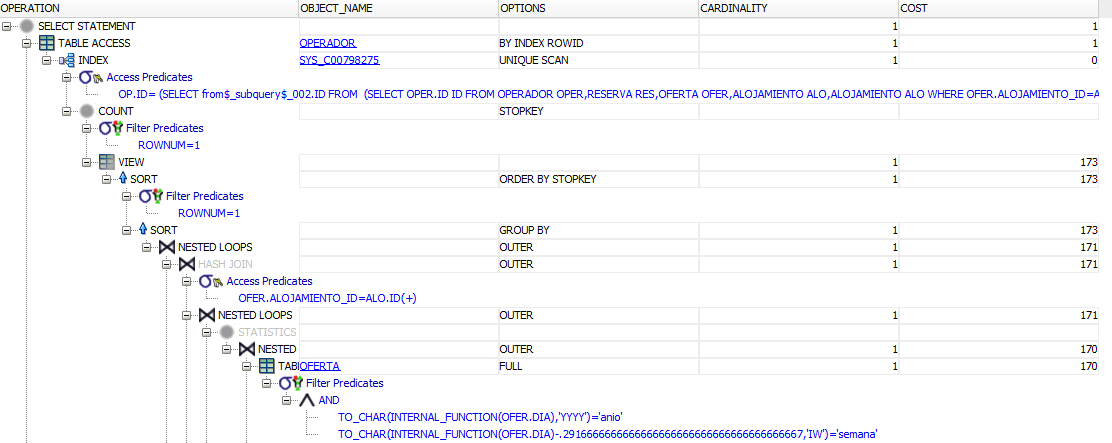
GROUP BY OPER.ID, TO\_CHAR(DIA - 7/24,'IW')

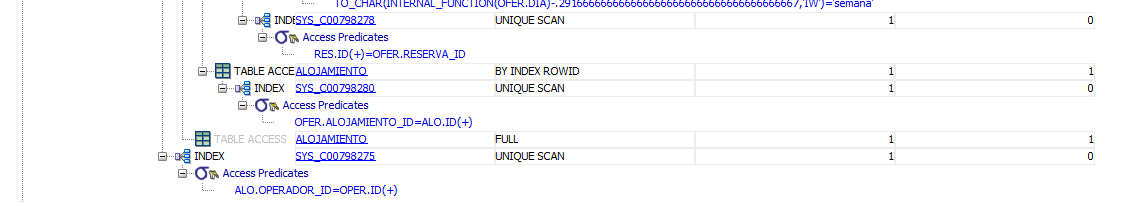
ORDER BY COUNT(RES.ID) DESC)

WHERE ROWNUM = 1);

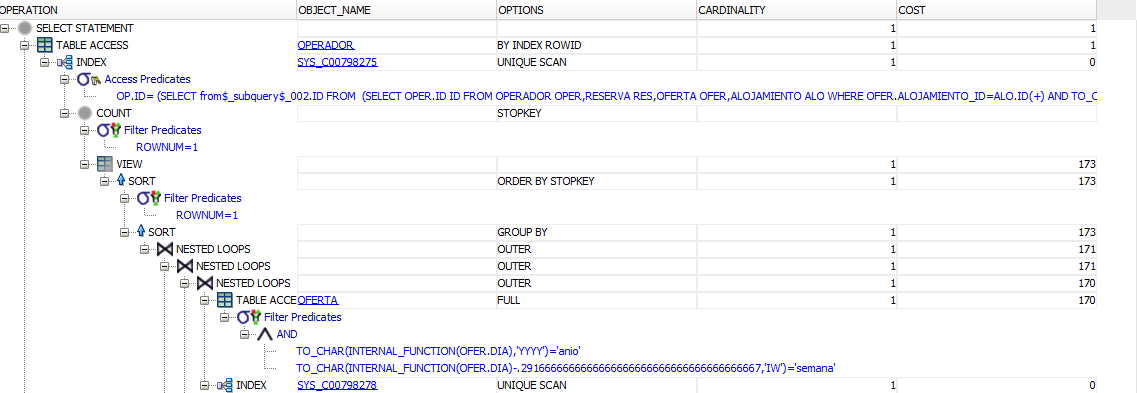
- La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos en un máximo de 365 días, los datos van a tender a ser muy parecidos.

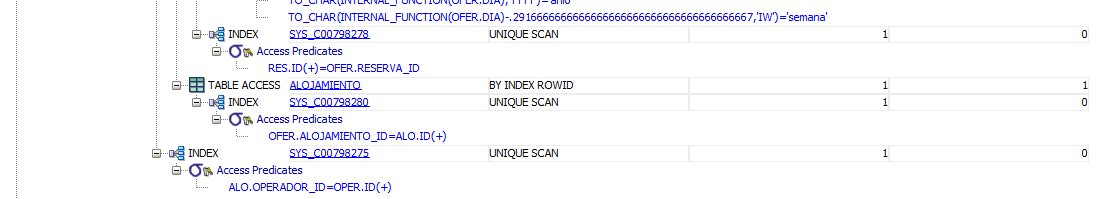
-Se obtuvo el siguiente plan de consulta con la arquitectura de la base de datos inicial:





-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

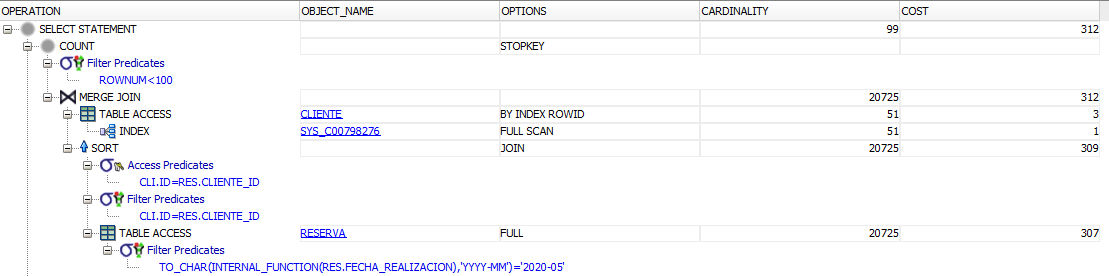
INNER JOIN RESERVA RES ON CLI.ID = RES.CLIENTE\_ID

WHERE TO\_CHAR(RES.FECHA\_REALIZACION,'YYYY-MM') LIKE '2020-05'

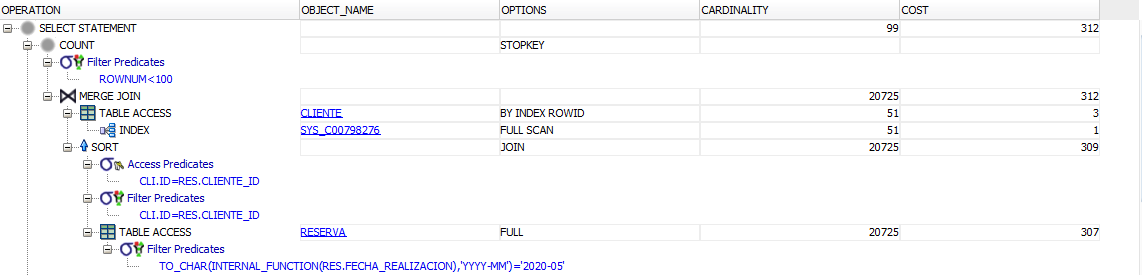
AND ROWNUM < 100;

-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:



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



-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

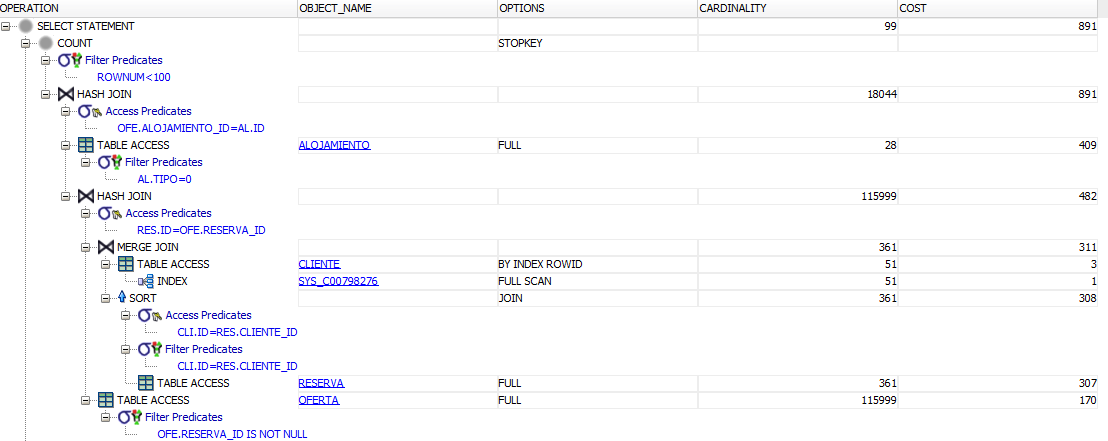
INNER JOIN ALOJAMIENTO AL ON OFE.ALOJAMIENTO\_ID = AL.ID

WHERE AL.TIPO = 0

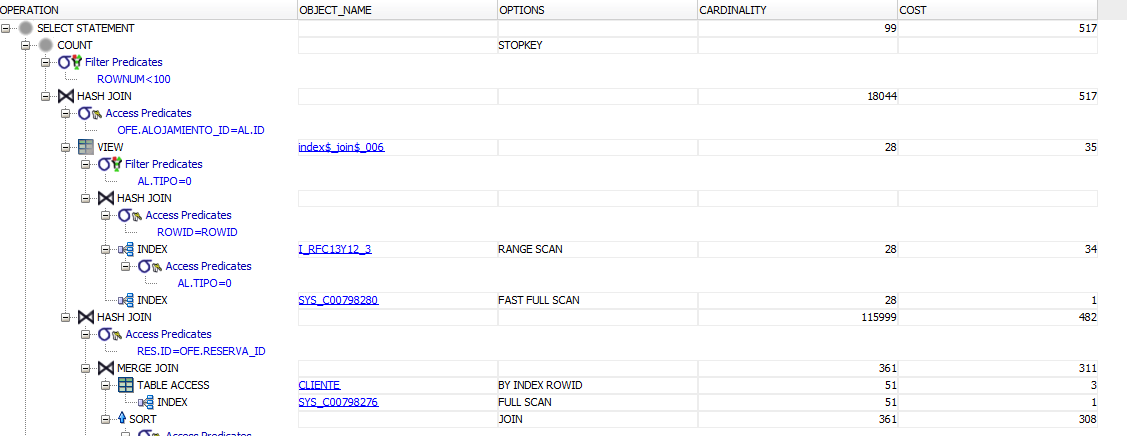
AND ROWNUM < 100;

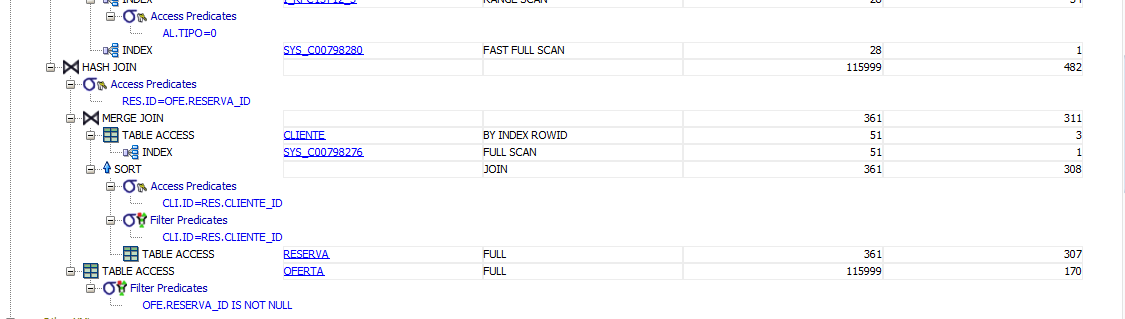
- La selectividad de esta consulta tiende a ser baja, ya que, debido a la distribución de un gran número de datos y solo tres distintas posibilidades, los datos van a tender a ser muy parecidos con respecto al tipo.

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



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





-La sentencia evaluada fue:

SELECT CLI.\* FROM CLIENTE CLI

INNER JOIN RESERVA RES ON CLI.ID = RES.CLIENTE\_ID

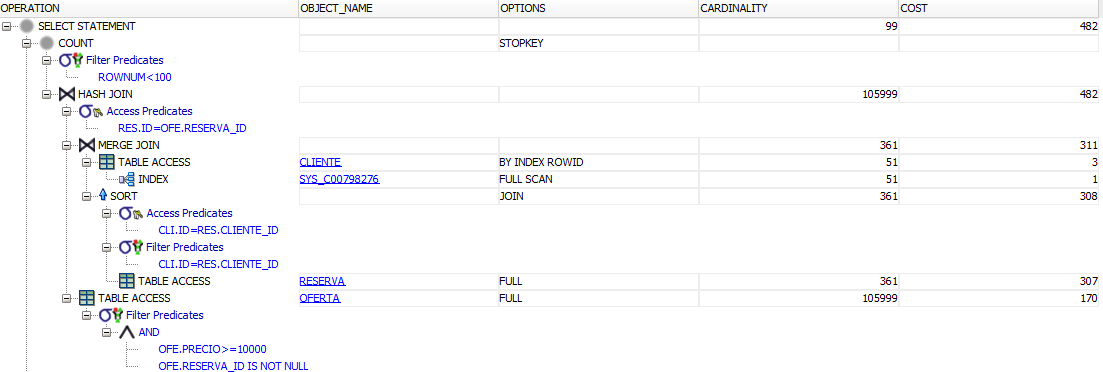
INNER JOIN OFERTA OFE ON RES.ID = OFE.RESERVA\_ID

WHERE OFE.PRECIO >= 10000

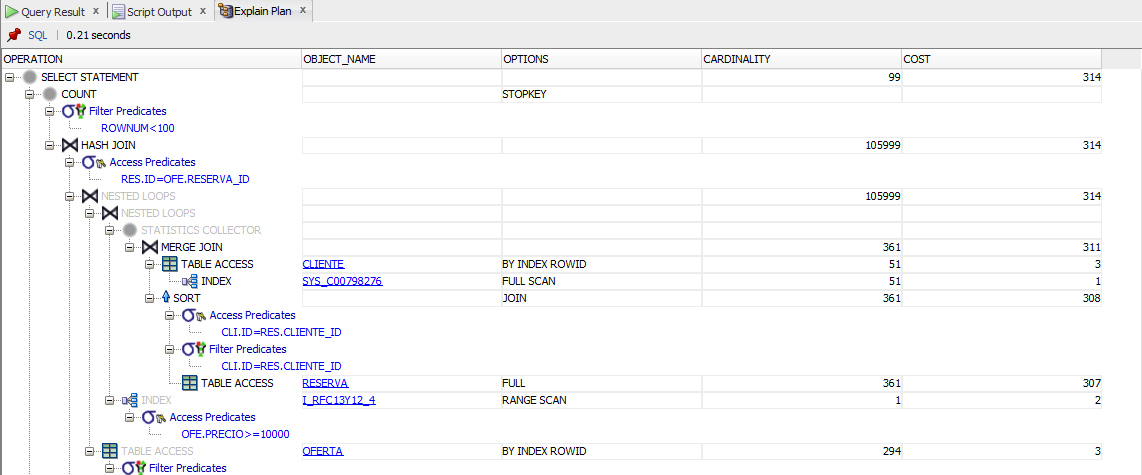
AND ROWNUM < 100;

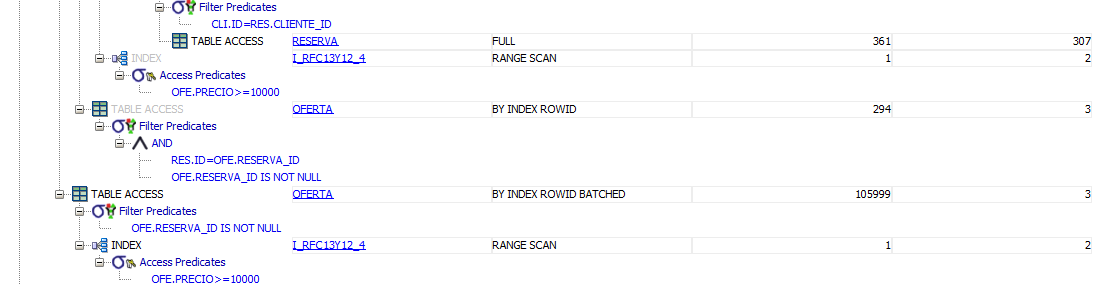
-La selectividad de esta consulta tiende a ser alta, ya que, debido a la distribución con la que se insertaron los datos (id = precio), los datos van a ser distintos siempre.

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



-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.

# 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.