Diseño de consultas para Requerimientos funcionales

1. **Sentencias:** Archivo SQL anexado a esta entrega

2. **Análisis de distribución:**

   -RF1:

   Rango de Fechas: Uno de los parámetros clave en esta consulta es el rango de fechas utilizado en la cláusula WHERE. El rango de fechas es r.fechaentrada >= TRUNC(SYSDATE, 'YYYY') - INTERVAL '1' YEAR, lo que significa que se están considerando las reservas realizadas durante el último año calendario. El tamaño de la respuesta variará significativamente según el rango de fechas elegido.

   Número de Habitaciones y Servicios: El tamaño de la respuesta también dependerá del número de habitaciones y servicios disponibles en la base de datos.

   -RF2:

   Rango de Fechas: El rango de fechas definido en la cláusula WHERE es crucial. Los valores '2023-01-01' y '2023-11-02' representan el inicio y el final del período analizado. El tamaño de la respuesta variará según el rango de fechas elegido. Si se acorta o se amplía el rango, afectará directamente la cantidad de registros incluidos en el resultado. Un rango más amplio incluirá más consumos, mientras que un rango más corto limitará la cantidad de consumos en la respuesta.

   Numero de Servicios y ReservasServicios: La cantidad de registros en las tablas servicios y reservasservicios en los datos de prueba influirá en el tamaño de la respuesta. Si hay una gran cantidad de registros, la respuesta será más grande en comparación con una base de datos con menos registros.

   -RF3:

   Rango de Fechas: El rango de fechas definido en la cláusula WHERE es fundamental. El rango es r.fechasalida >= (SYSDATE - 365), lo que significa que se están considerando las reservas cuya fecha de salida está dentro del último año (365 días) a partir de la fecha actual (SYSDATE). El tamaño de la respuesta variará según el rango de fechas elegido

   Habitaciones y reservas: La cantidad de registros en las tablas habitaciones y reservas en los datos de prueba influirá en el tamaño de la respuesta. Si hay una gran cantidad de registros de reservas, la respuesta será más grande en comparación con una base de datos con menos registros de reservas.

-RF4:

Rango de Fechas: El rango de fechas definido en la cláusula WHERE es fundamental. El rango es rs.fecha BETWEEN TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-11-05', 'YYYY-MM-DD'). Esto significa que se están considerando los registros de reservasservicios cuyas fechas están dentro de este período. El tamaño de la respuesta variará según el rango de fechas elegido. Cambiar las fechas de inicio y final afectará el número de registros incluidos en el resultado.

Costo del Servicio: La condición s.costo BETWEEN 0 AND 16000000 filtra los servicios con un costo dentro de este rango. Modificar el rango de costos afectará la cantidad de servicios que cumplan con este criterio. Si se amplía el rango, se incluirán servicios más costosos en la respuesta.

Nombre del Servicio: La condición s.nombreservicio = 'Gimnasio' filtra los registros que coinciden con el nombre de servicio 'Gimnasio'. Cambiar el nombre del servicio afectará los resultados, ya que solo se incluirán registros relacionados con el servicio especificado.

-RF5:

Nombre del Usuario: El valor 'Esteban' en la condición u.nombreusuario = 'Esteban' filtra las reservas y consumos asociados al usuario con este nombre. Cambiar el nombre del usuario afectará los resultados, ya que se incluirán registros relacionados con el usuario especificado.

Rango de Fechas: El rango de fechas definido en las condiciones c.fecha >= TO_DATE('2023-01-01', 'YYYY-MM-DD') y c.fecha <= TO_DATE('2023-11-05', 'YYYY-MM-DD') afecta los consumos incluidos en la respuesta. Cambiar las fechas de inicio y final afectará el período de tiempo que se considera para los consumos. Si se amplía o reduce el rango de fechas, afectará la cantidad de consumos en la respuesta.

-RF6

Fecha entrada: La cantidad de valores de fecha entrada de la tabla reservas afecta los resultados. Entre más reservas haya más registros habrá

Consumos: La cantidad de consumos realizados afecta directamente en los resultados de respuesta, entre mas registros de consumo, más valores de fechas estarán en la tabla, afectando así los resultados.

-RF7

El criterio de "Ha estado en el hotel por al menos dos semanas" se basa en la diferencia entre las fechas de entrada y salida en las reservas. Los parámetros clave en este caso son las fechas utilizadas en la condición r.fechasalida >= TO_DATE('2022-11-05', 'YYYY-MM-DD') - INTERVAL '1' YEAR para calcular la estadía de al menos dos semanas. Cambiar la fecha de

referencia (por ejemplo, cambiar el '1 YEAR' a '6 MONTHS') afectará los resultados y el número de "buenos clientes" identificados

El criterio de "Su consumo total es mayor a 15.000.000" se basa en la suma de los costos de consumo de los servicios y el costo de las habitaciones. Los parámetros clave en este caso son el umbral de 15.000.000 y las tablas involucradas en el JOIN. Cambiar el umbral de consumo afectará el número de "buenos clientes" identificados. Además, la cantidad de datos en las tablas usuarios, reservas, habitaciones, tiposh, reservasservicios, servicios y consumos influirá en los resultados.

-RF8

El criterio es "Encontrar los servicios que hayan sido solicitados menos de 3 veces semanales, durante el último año de operación de HotelAndes." Los parámetros clave en este caso son el número "3" (umbral de demanda) y la fecha utilizada para el último año de operación (SYSDATE - 365). Cambiar el umbral de demanda o ajustar la fecha afectará los resultados y el número de servicios identificados con poca demanda.

-RF9

Criterio de Consumo de Servicio : El criterio es "clientes que consumieron al menos una vez un determinado servicio del hotel" en un rango de fechas. Los parámetros clave en este caso son:
*El nombre del servicio
*El rango de fechas utilizado para filtrar las reservas.

Cambiar el nombre del servicio o el rango de fechas afectará los resultados y el número de clientes identificados.

-RF10

El criterio es "clientes que no consumieron el servicio de Piscina" en un rango de fechas. Los parámetros clave en este caso son:
*El nombre del servicio
*El rango de fechas utilizado para filtrar las reservas.

Cambiar el nombre del servicio o el rango de fechas afectará los resultados y el número de clientes identificados.

-RF12

La consulta tiene dos secciones con diferentes criterios para identificar a un "Excelente Cliente". Los criterios clave en este caso son:
*Para la primera sección: alojarse al menos una vez por trimestre.

*Para la segunda sección: tener reservas de SPA, salones de reuniones con duración > 4 horas o reservas con costo total >= 300,000.
Cambiar estos criterios afectará los resultados y el número de clientes identificados como "Excelentes Clientes".
La cantidad de registros en las tablas usuarios, reservas, reservasservicios, y servicios influirá en los resultados, entre más registros haya, más afectará a los resultados y al tiempo de ejecución de la consulta.

3. Valores de parámetros utilizados en el análisis

```
INSERT INTO tiposh VALUES ('Familiar', 'Tiene 1 minifridge, 2 camas king, 2 banos, 1 televisor', 5, 50000);
INSERT INTO tiposh VALUES ('SuitePresidencial', 'Tiene 3 minifridge, 4 camas king, 4 banos, 3 televisor', 7, 80000);
```

```
INSERT INTO habitaciones VALUES (701, 'Familiar');
INSERT INTO habitaciones VALUES (702, 'SuitePresidencial');
INSERT INTO habitaciones VALUES (703, 'Familiar');
INSERT INTO habitaciones VALUES (704, 'SuitePresidencial');
INSERT INTO habitaciones VALUES (705, 'Familiar');
INSERT INTO habitaciones VALUES (706, 'SuitePresidencial');
INSERT INTO habitaciones VALUES (707, 'SuitePresidencial');
INSERT INTO habitaciones VALUES (708, 'SuitePresidencial');
INSERT INTO habitaciones VALUES (709, 'Familiar');
INSERT INTO habitaciones VALUES (710, 'Familiar');
INSERT INTO habitaciones VALUES (711, 'Familiar');
INSERT INTO habitaciones VALUES (712, 'SuitePresidencial');
```

```
INSERT INTO servicios VALUES (1, 'Piscina', 'Piscina', '6:00 AM - 11:00 PM', 100, 5000);
INSERT INTO servicios VALUES (2, 'Gimnasio', 'Gimnasio', '6:00 AM - 11:00 PM', 300, 3000);
INSERT INTO servicios VALUES (3, 'Prueba15Millones', 'Prueba15Millones', '6:00 AM - 11:00 PM', 300, 16000000);
INSERT INTO servicios VALUES (4, 'SPA', 'SPA', '6:00 AM - 11:00 PM', 10, 400000);
INSERT INTO servicios VALUES (5, 'Salones de Reuniones', 'Salones de Reuniones', '6:00 AM - 11:00 PM', 120, 300000);
```

```
INSERT INTO tiposusuarios VALUES ('Cliente');
INSERT INTO tiposusuarios VALUES ('Empleado');
```

```
INSERT INTO empleados VALUES (100, 'Empleado');
INSERT INTO empleados VALUES (101, 'Empleado');
INSERT INTO empleados VALUES (102, 'Empleado');
INSERT INTO empleados VALUES (104, 'Empleado');
```

```
INSERT INTO usuarios VALUES ('0000000005', 'CC', 'Esteban', 'esteban@uniandes.edu.co', 'Cliente');
INSERT INTO usuarios VALUES ('0000000006', 'CC', 'Juan', 'Juan@uniandes.edu.co', 'Cliente');
INSERT INTO usuarios VALUES ('0000000007', 'CC', 'Jose', 'Jose@uniandes.edu.co', 'Cliente');
INSERT INTO usuarios VALUES ('0000000008', 'CC', 'Alejandro', 'Alejandro@uniandes.edu.co', 'Cliente');
INSERT INTO usuarios VALUES ('0000000009', 'CC', 'Andrea', 'Andrea@uniandes.edu.co', 'Cliente');
INSERT INTO usuarios VALUES ('0000000010', 'CC', 'Daniela', 'Daniela@uniandes.edu.co', 'Cliente');
INSERT INTO usuarios VALUES ('0000000011', 'CC', 'Cristian', 'Cristian@uniandes.edu.co', 'Cliente');
```

```
INSERT INTO planes VALUES ('Basico', 0.1, TO_DATE('2020-01-01', 'YYYY-MM-DD'), TO_DATE('2025-10-10', 'YYYY-MM-DD'));
INSERT INTO planes VALUES ('VIP', 0, TO_DATE('2020-01-01', 'YYYY-MM-DD'), TO_DATE('2025-10-10', 'YYYY-MM-DD'));
```

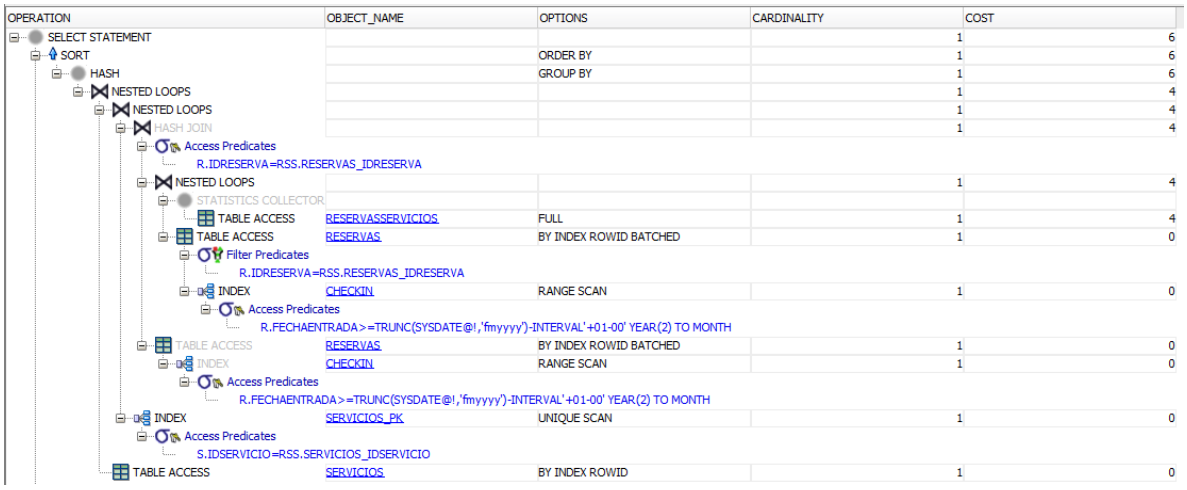
```
INSERT INTO reservas VALUES (1, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-01-10', 'YYYY-MM-DD'), 2, 'Basico', 0000000005, 701);
INSERT INTO reservas VALUES (2, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-01-10', 'YYYY-MM-DD'), 3, 'VIP', 0000000006, 702);
INSERT INTO reservas VALUES (3, TO_DATE('2023-01-11', 'YYYY-MM-DD'), TO_DATE('2023-01-15', 'YYYY-MM-DD'), 4, 'VIP', 0000000007, 703);
INSERT INTO reservas VALUES (4, TO_DATE('2023-01-12', 'YYYY-MM-DD'), TO_DATE('2023-02-12', 'YYYY-MM-DD'), 5, 'VIP', 0000000009, 704);
INSERT INTO reservas VALUES (5, TO_DATE('2023-02-11', 'YYYY-MM-DD'), TO_DATE('2023-03-11', 'YYYY-MM-DD'), 4, 'VIP', 0000000008, 705);
INSERT INTO reservas VALUES (6, TO_DATE('2023-02-15', 'YYYY-MM-DD'), TO_DATE('2023-02-16', 'YYYY-MM-DD'), 4, 'VIP', 0000000010, 706);
INSERT INTO reservas VALUES (7, TO_DATE('2023-05-15', 'YYYY-MM-DD'), TO_DATE('2023-05-16', 'YYYY-MM-DD'), 4, 'VIP', 0000000010, 707);
INSERT INTO reservas VALUES (8, TO_DATE('2023-03-11', 'YYYY-MM-DD'), TO_DATE('2023-03-15', 'YYYY-MM-DD'), 4, 'VIP', 0000000011, 708);
INSERT INTO reservas VALUES (9, TO_DATE('2023-04-01', 'YYYY-MM-DD'), TO_DATE('2023-04-10', 'YYYY-MM-DD'), 2, 'Basico', 0000000005, 709);
INSERT INTO reservas VALUES (10, TO_DATE('2023-07-01', 'YYYY-MM-DD'), TO_DATE('2023-07-10', 'YYYY-MM-DD'), 2, 'Basico', 0000000005, 710);
INSERT INTO reservas VALUES (11, TO_DATE('2023-10-01', 'YYYY-MM-DD'), TO_DATE('2023-10-10', 'YYYY-MM-DD'), 2, 'Basico', 0000000005, 711);
INSERT INTO reservas VALUES (12, TO_DATE('2023-10-01', 'YYYY-MM-DD'), TO_DATE('2023-10-10', 'YYYY-MM-DD'), 2, 'Basico', 0000000011, 712);
INSERT INTO reservas VALUES (13, TO_DATE('2023-01-20', 'YYYY-MM-DD'), TO_DATE('2023-01-22', 'YYYY-MM-DD'), 2, 'Basico', 0000000005, 701);
```

```
INSERT INTO consumos VALUES (1, 'Almuerzo piscina', TO_DATE('2023-10-30', 'YYYY-MM-DD'), 40, 1);
INSERT INTO consumos VALUES (2, 'Entrenamiento personalizado GYM', TO_DATE('2023-10-30', 'YYYY-MM-DD'), 20, 2);
```

```
INSERT INTO reservasservicios VALUES (1,TO_DATE('2023-01-04', 'YYYY-MM-DD'),5 , 1, 1, 101);
INSERT INTO reservasservicios VALUES (2, TO_DATE('2023-01-05', 'YYYY-MM-DD'),5, 2, 2, 101);
INSERT INTO reservasservicios VALUES (3, TO_DATE('2023-01-06', 'YYYY-MM-DD'),5, 2, 3, 102);
INSERT INTO reservasservicios VALUES (4, TO_DATE('2023-01-07', 'YYYY-MM-DD'),5, 3, 4, 104);
INSERT INTO reservasservicios VALUES (5, TO_DATE('2023-01-08', 'YYYY-MM-DD'),5, 2, 5, 104);
INSERT INTO reservasservicios VALUES (6, TO_DATE('2023-01-08', 'YYYY-MM-DD'),6, 4, 12, 104);
```

4. Planes de consulta obtenidos por Oracle:

RF1:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	6
SORT		ORDER BY	1	6
HASH		GROUP BY	1	6
NESTED LOOPS			1	4
NESTED LOOPS			1	4
HASH JOIN			1	4
Access Predicates				
R.IDRESERVA=RSS.RESERVAS_IDRESERVA				
NESTED LOOPS			1	4
STATISTICS COLLECTOR				
TABLE ACCESS	RESERVASSERVICIOS	FULL	1	4
TABLE ACCESS	RESERVAS	BY INDEX ROWID BATCHED	1	0
Filter Predicates				
R.IDRESERVA=RSS.RESERVAS_IDRESERVA				
INDEX	CHECKIN	RANGE SCAN	1	0
Access Predicates				
R.FECHAENTRADA>=TRUNC(SYSDATE@!,'fmyyyy')-INTERVAL'+01-00' YEAR(2) TO MONTH				
TABLE ACCESS	RESERVAS	BY INDEX ROWID BATCHED	1	0
INDEX	CHECKIN	RANGE SCAN	1	0
Access Predicates				
R.FECHAENTRADA>=TRUNC(SYSDATE@!,'fmyyyy')-INTERVAL'+01-00' YEAR(2) TO MONTH				
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
S.IDSERVICIO=RSS.SERVICIOS_IDSERVICIO				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0

Se utiliza un ORDER BY con el fin de ordenar los resultados por número de habitación y nombre del servicio

Se utiliza un GROUP BY con el fin de ordenar los resultados por número de habitación y nombre del servicio

Se utiliza un FULL para acceder a la tabla reservaServicios, es decir, un recorrido completo de la tabla.

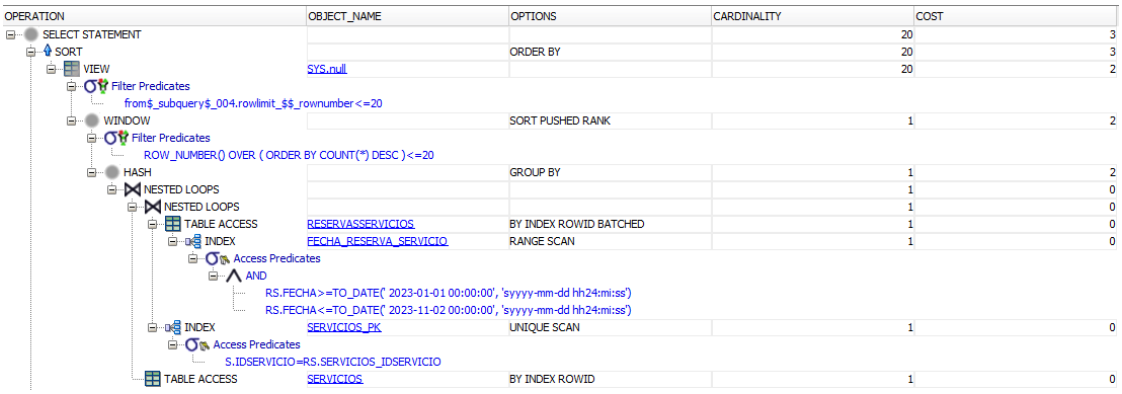
Se utiliza un By INDEX ROWID BATCHED para acceder a la tabla reservas 2 veces, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

Se utiliza un RANGE SCAN 2 veces, que es una operación de acceso a datos que utiliza un índice para recuperar dentro un rango especifico de valores en la tabla, en este caso el valor CHECKIN

Se utiliza UNIQUE SCAN para recuperar una fila especifica en la tabla Servicios, en este caso, el ID

Se utiliza un BY INDEX ROWID para acceder a la tabla servicios con el fin de encontrar la coincidencia en s.idServicio y rss.servicios_idservicio.

RF2:



Se utiliza un ORDER BY con el fin de ordenar los resultados por el total de consumos

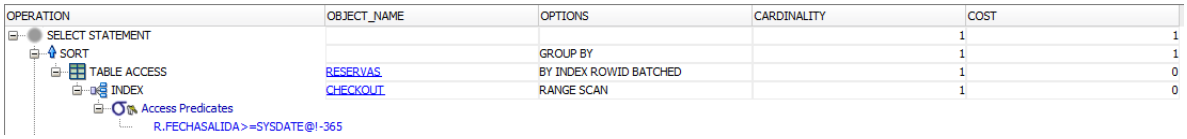
Se utiliza un GROUP BY con el fin de agrupar los resultados por s.nombreservicio

Se utiliza 2 BY INDEX ROWID BATCHED para acceder a la tabla reservaservicios y servicios, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

Se utiliza un RANGE SCAN que es una operación de acceso a datos que utiliza un índice para recuperar dentro un rango especifico de valores en la tabla, en este caso fecha_reserva_servicio

Se utiliza un UNIQUE SCAN para recuperar una fila especifica en la tabla servicio

RF3:

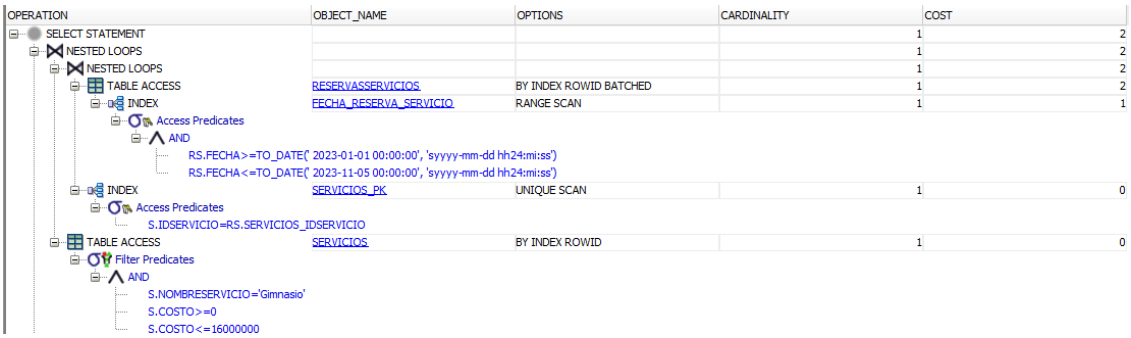


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
SORT		GROUP BY	1	1
TABLE ACCESS	RESERVAS	BY INDEX ROWID BATCHED	1	0
INDEX	CHECKOUT	RANGE SCAN	1	0
Access Predicates				
R.FECHASALIDA>=SYSDATE@!-365				

Se utiliza un GROUP BY con el fin de agrupar los resultados de la consulta por el numero de habitación o h.numero

Se utiliza un BY INDEX ROWID BATCHED para acceder a la tabla reservas, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

RF4:



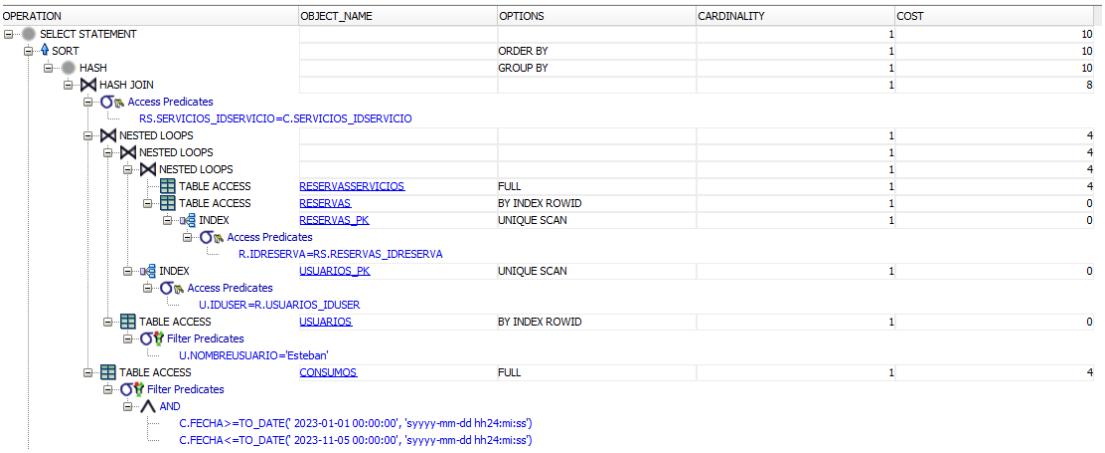
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
NESTED LOOPS			1	2
NESTED LOOPS			1	2
TABLE ACCESS	RESERVASSERVICIOS	BY INDEX ROWID BATCHED	1	2
INDEX	FECHA_RESERVA_SERVICIO	RANGE SCAN	1	1
Access Predicates				
AND				
RS.FECHA>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RS.FECHA<=TO_DATE(' 2023-11-05 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
S.IDSERVICIO=RS.SERVICIOS_IDSERVICIO				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
S.NOMBRESERVICIO='Gimnasio'				
S.COSTO>=0				
S.COSTO<=16000000				

Se utiliza 2 BY INDEX ROWID BATCHED para acceder a la tabla reservasservicios y servicios, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

Se utiliza un RANGE SCAN que es una operación de acceso a datos que utiliza un índice para recuperar dentro un rango especifico de valores en la tabla, en este caso fecha_reserva_servicio

Se utiliza un UNIQUE SCAN para recuperar una fila especifica en la tabla servicio

RF5:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	10
SORT		ORDER BY	1	10
HASH		GROUP BY	1	10
HASH JOIN			1	8
Access Predicates				
RS.SERVICIOS_IDSERVICIO=C.SERVICIOS_IDSERVICIO				
NESTED LOOPS			1	4
NESTED LOOPS			1	4
NESTED LOOPS			1	4
TABLE ACCESS	RESERVASSERVICIOS	FULL	1	4
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
R.IDRESERVA=RS.RESERVAS_IDRESERVA				
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
U.IDUSER=R.USUARIOS_IDUSER				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0
Filter Predicates				
U.NOMBREUSUARIO='Esteban'				
TABLE ACCESS	CONSUMOS	FULL	1	4
Filter Predicates				
AND				
C.FECHA>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
C.FECHA<=TO_DATE(' 2023-11-05 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				

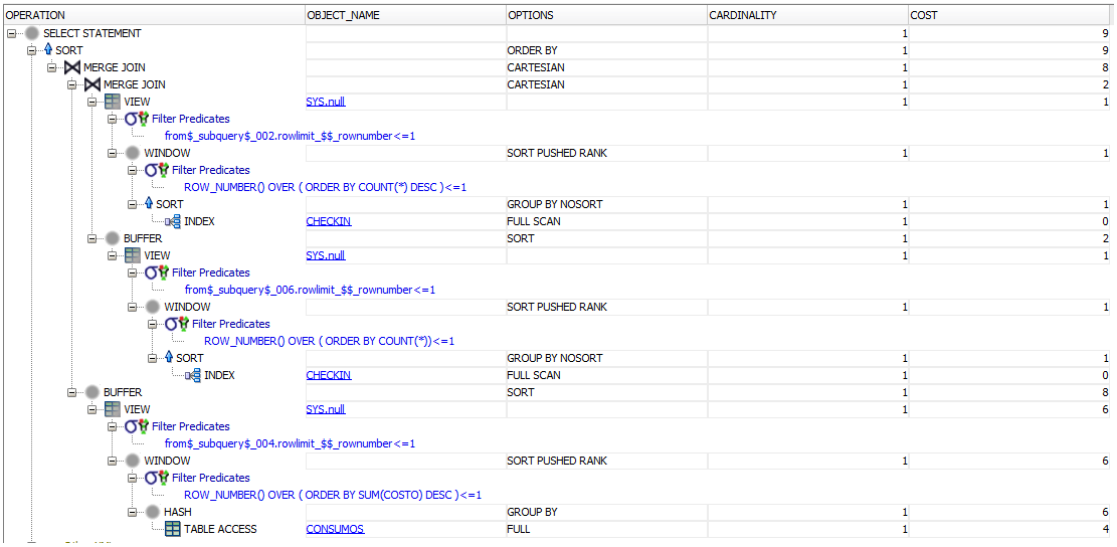
Se utiliza un ORDER y GROUP by con el fin de ordenar y agrupar los resultados en base a id del usuario, el nombre y en TOTAL GASTADO EN COSUMOS

Se utiliza 2 FULL para acceder a la tabla reservaServicios y consumos, es decir, un recorrido completo de la tabla.

Se utiliza un BY INDEX ROWID para acceder a la tabla reservas, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

Se utiliza 2 UNIQUE SCAN para recuperar una fila especifica en la tabla usuarios

RF6:

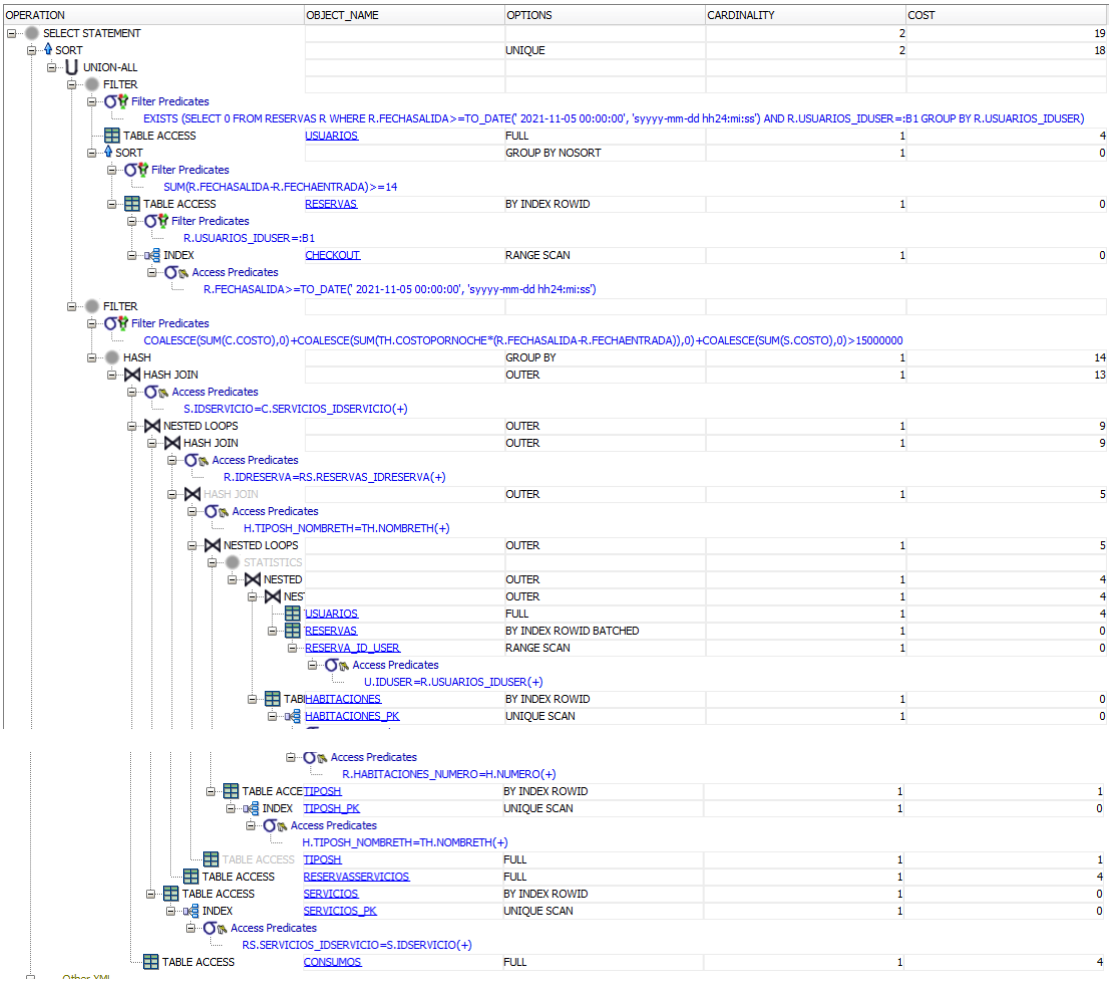


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	9
SORT		ORDER BY	1	9
MERGE JOIN		CARTESIAN	1	8
MERGE JOIN		CARTESIAN	1	2
VIEW	SYS.null		1	1
Filter Predicates				
from$_subquery$_002.rowlimit_$$_rownumber<=1				
WINDOW		SORT PUSHED RANK	1	1
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC)<=1				
SORT		GROUP BY NOSORT	1	1
INDEX	CHECKIN	FULL SCAN	1	0
BUFFER		SORT	1	2
VIEW	SYS.null		1	1
Filter Predicates				
from$_subquery$_006.rowlimit_$$_rownumber<=1				
WINDOW		SORT PUSHED RANK	1	1
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*))<=1				
SORT		GROUP BY NOSORT	1	1
INDEX	CHECKIN	FULL SCAN	1	0
BUFFER		SORT	1	8
VIEW	SYS.null		1	6
Filter Predicates				
from$_subquery$_004.rowlimit_$$_rownumber<=1				
WINDOW		SORT PUSHED RANK	1	6
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY SUM(COSTO) DESC)<=1				
HASH		GROUP BY	1	6
TABLE ACCESS	CONSUMOS	FULL	1	4

Se utiliza un ORDER BY y ORDER BY con el fin de ordenar los resultados, esto depende de la subconsulta, en este caso para “Ocupación” se utiliza fechaentrada de reserva y cantidad_ocupaciones. Para “Ingresos” se utiliza la fecha de consumo e ingresos totales. Finalmente, para “Demanda” se utiliza fechaentrada de reserva y cantidad_ocupaciones

Se utiliza 2 FULL para acceder a la tabla consumos

RF7:



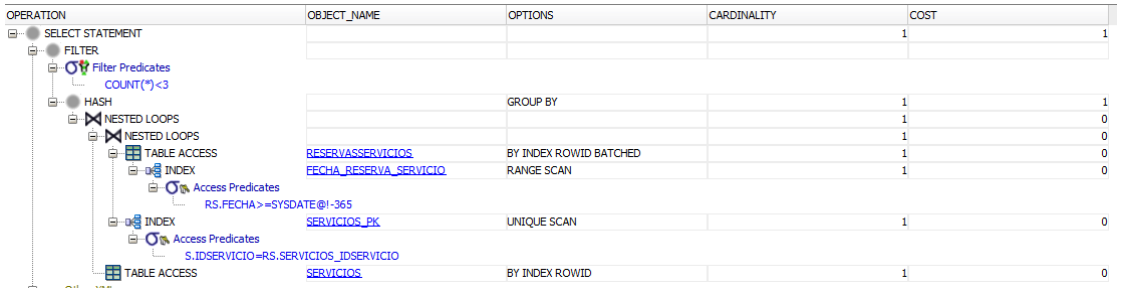
Se utiliza FULL para hacer un recorrido completo de USUARIOS en 2 ocasiones

Se utiliza GROUP BY para ordenar los resultados de la primera subconsulta por el id del usuario

Se utiliza 3 BY INDEX ROWID para acceder a la tabla reservas, habitaciones y servicios, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

Se utiliza 2 UNIQUE SCAN para acceder a TiposH y Servicios que es un acceso a datos que utiliza un índice único para buscar y recuperar la fila

RF8:



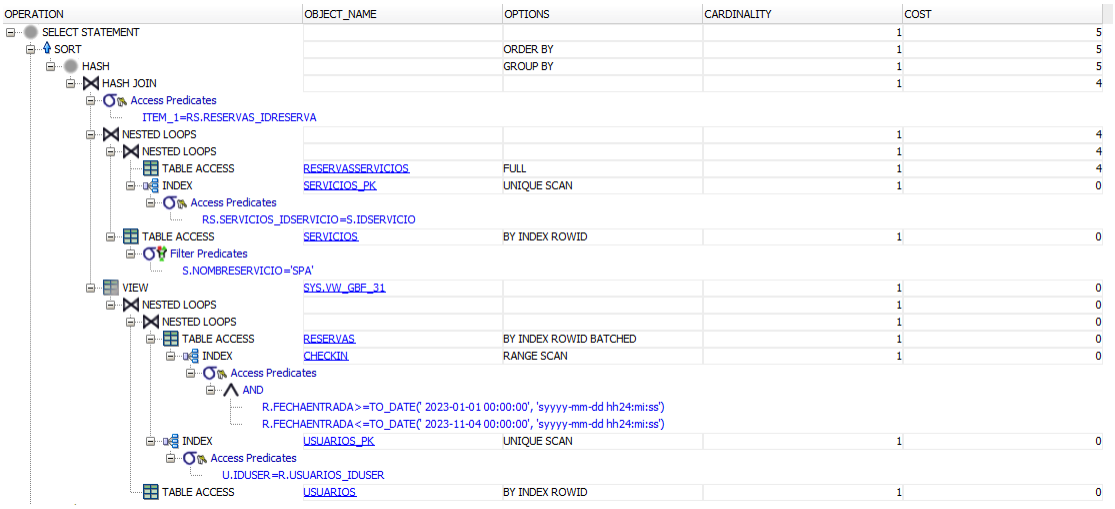
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
FILTER				
Filter Predicates				
COUNT(*)<3				
HASH		GROUP BY	1	1
NESTED LOOPS			1	0
NESTED LOOPS			1	0
TABLE ACCESS	RESERVASSERVICIOS	BY INDEX ROWID BATCHED	1	0
INDEX	FECHA_RESERVA_SERVICIO	RANGE SCAN	1	0
Access Predicates				
RS.FECHA>=SYSDATE@!-365				
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
S.IDSERVICIO=RS.SERVICIOS_IDSERVICIO				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0

Se utiliza un GROUP BY con el fin de agrupar los resultados en base al ID del servicio y el nombre del servicio

Se utiliza 2 BY INDEX ROWID para acceder a la tabla reservasservicios y servicios, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

Se utiliza un UNIQUE SCAN para recuperar una fila especifica en la tabla servicio

RF9:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	5
SORT		ORDER BY	1	5
HASH		GROUP BY	1	5
HASH JOIN			1	4
Access Predicates				
ITEM_1=RS.RESERVAS_IDRESERVA				
NESTED LOOPS			1	4
NESTED LOOPS			1	4
TABLE ACCESS	RESERVASSERVICIOS	FULL	1	4
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
RS.SERVICIOS_IDSERVICIO=S.IDSERVICIO				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0
Filter Predicates				
S.NOMBRESERVICIO='SPA'				
VIEW	SYS.VW_GBF_31		1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
TABLE ACCESS	RESERVAS	BY INDEX ROWID BATCHED	1	0
INDEX	CHECKIN	RANGE SCAN	1	0
Access Predicates				
AND				
R.FECHAENTRADA>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
R.FECHAENTRADA<=TO_DATE(' 2023-11-04 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
U.IDUSER=R.USUARIOS_IDUSER				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0

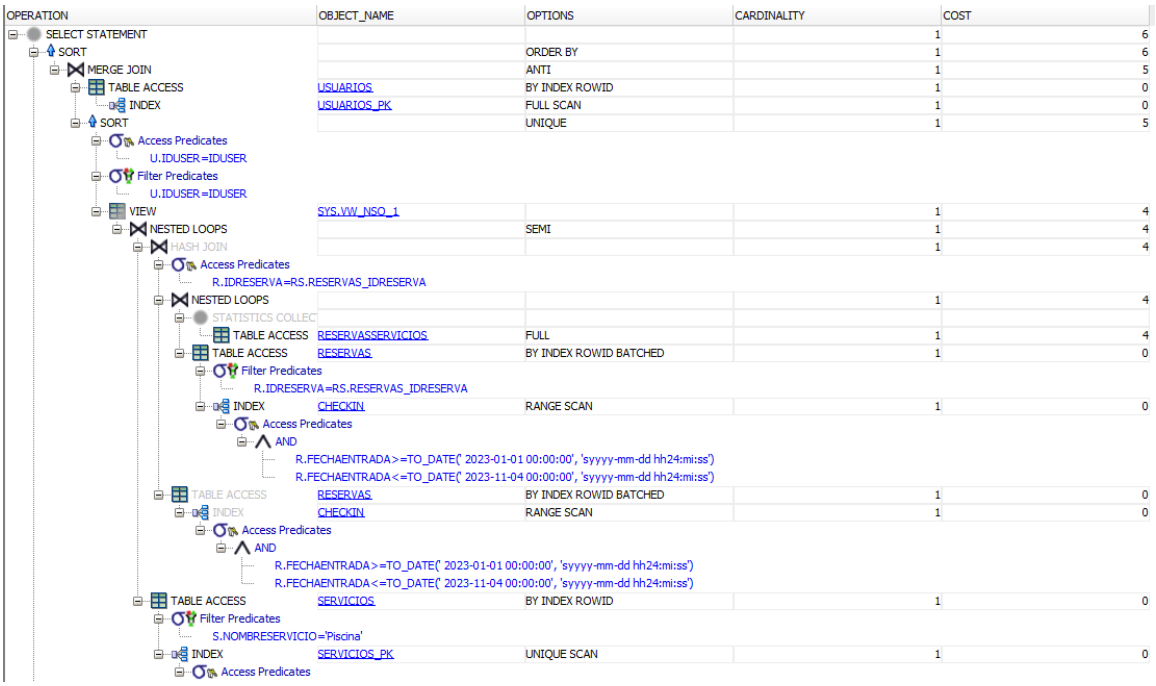
Se utiliza un ORDER BY y GROUP BY para order y agrupar los resultados de la consulta en base a el id del usuario, el tipo de documento, el correo y el nombre

Se utiliza un FULL para acceder a la tabla reservaServicios y consumos, es decir, un recorrido completo de la tabla.

Se utiliza un UNIQUE SCAN para recuperar una fila especifica en la tabla servicio.

Se utiliza 2 BY INDEX ROWID para acceder a la tabla servicios y usuarios, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices.

RF10:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	6
SORT		ORDER BY	1	6
MERGE JOIN		ANTI	1	5
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0
INDEX	USUARIOS_PK	FULL SCAN	1	0
SORT		UNIQUE	1	5
Access Predicates				
U.IDUSER=IDUSER				
Filter Predicates				
U.IDUSER=IDUSER				
VIEW	SYS.VW_NSO_1		1	4
NESTED LOOPS		SEMI	1	4
HASH JOIN			1	4
Access Predicates				
R.IDRESERVA=RS.RESERVAS_IDRESERVA				
NESTED LOOPS			1	4
STATISTICS COLLEC				
TABLE ACCESS	RESERVASSERVICIOS	FULL	1	4
TABLE ACCESS	RESERVAS	BY INDEX ROWID BATCHED	1	0
Filter Predicates				
R.IDRESERVA=RS.RESERVAS_IDRESERVA				
INDEX	CHECKIN	RANGE SCAN	1	0
Access Predicates				
AND				
R.FECHAENTRADA>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
R.FECHAENTRADA<=TO_DATE(' 2023-11-04 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
TABLE ACCESS	RESERVAS	BY INDEX ROWID BATCHED	1	0
INDEX	CHECKIN	RANGE SCAN	1	0
Access Predicates				
AND				
R.FECHAENTRADA>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
R.FECHAENTRADA<=TO_DATE(' 2023-11-04 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0
Filter Predicates				
S.NOMBRESERVICIO='Piscina'				
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				

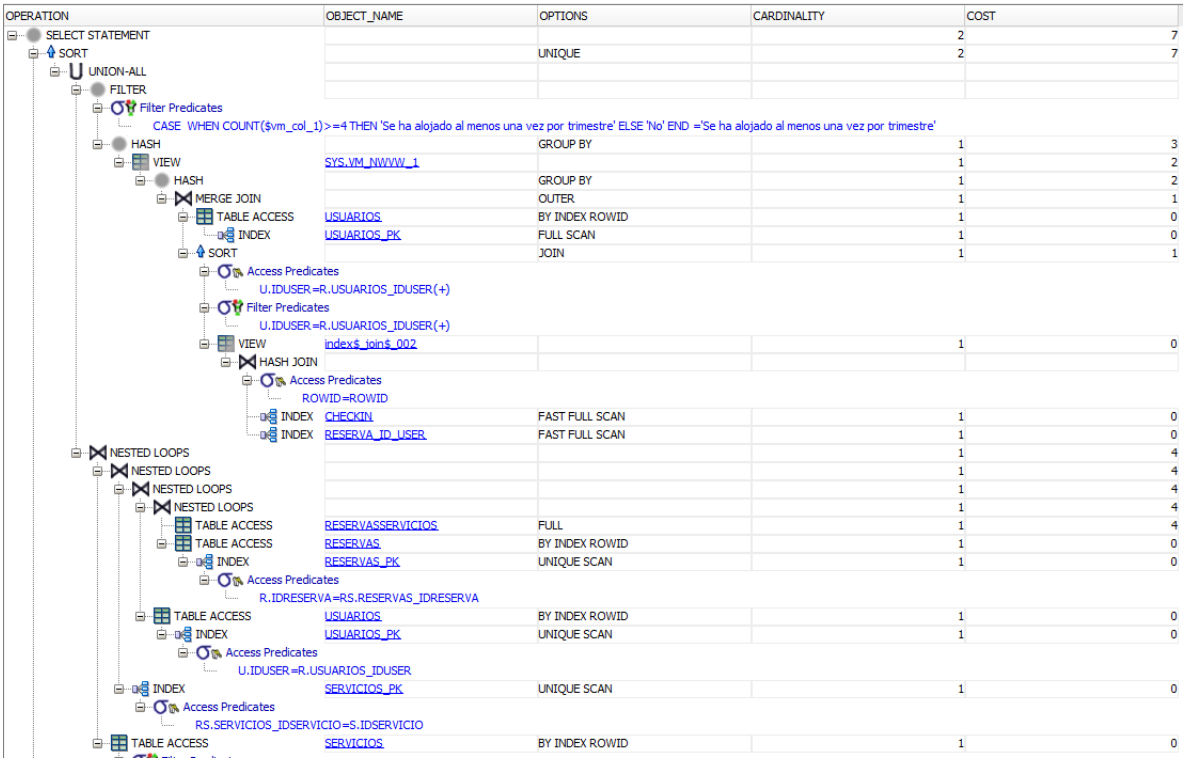
Se utiliza un ORDER BY con el fin de ordenar los resultados en base al nombre del usuario

Se utiliza 2 BY INDEX ROWID BATCHED para acceder a la tabla usuarios y reservas, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

Se utiliza un FULL para acceder a la tabla reservaServicios, es decir, un recorrido completo de la tabla.

Se utiliza un UNIQUE SCAN para recuperar una fila especifica en la tabla servicio

RF12:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	7
SORT		UNIQUE	2	7
UNION-ALL				
FILTER				
Filter Predicates				
CASE WHEN COUNT($vm_col_1)>=4 THEN 'Se ha alojado al menos una vez por trimestre' ELSE 'No' END ='Se ha alojado al menos una vez por trimestre'				
HASH		GROUP BY	1	3
VIEW	SYS.VM_NWVW_1		1	2
HASH		GROUP BY	1	2
MERGE JOIN		OUTER	1	1
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0
INDEX	USUARIOS_PK	FULL SCAN	1	0
SORT		JOIN	1	1
Access Predicates				
U.IDUSER=R.USUARIOS_IDUSER(+)				
Filter Predicates				
U.IDUSER=R.USUARIOS_IDUSER(+)				
VIEW	index$_join$_002		1	0
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	CHECKIN	FAST FULL SCAN	1	0
INDEX	RESERVA_ID_USER	FAST FULL SCAN	1	0
NESTED LOOPS			1	4
NESTED LOOPS			1	4
NESTED LOOPS			1	4
NESTED LOOPS			1	4
TABLE ACCESS	RESERVASSERVICIOS	FULL	1	4
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
R.IDRESERVA=RS.RESERVAS_IDRESERVA				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
U.IDUSER=R.USUARIOS_IDUSER				
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
RS.SERVICIOS_IDSERVICIO=S.IDSERVICIO				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0

Se utiliza un GROUP BY con el fin de agrupar los resultados en base al id del usuario, el nombre del usuario, el tipo de documento y el correo

Se utiliza 3 BY INDEX ROWID para acceder a la tabla usuarios, reservas y servicios, lo que significa que está realizando una operación que implica una búsqueda y recuperación de datos por medio de índices

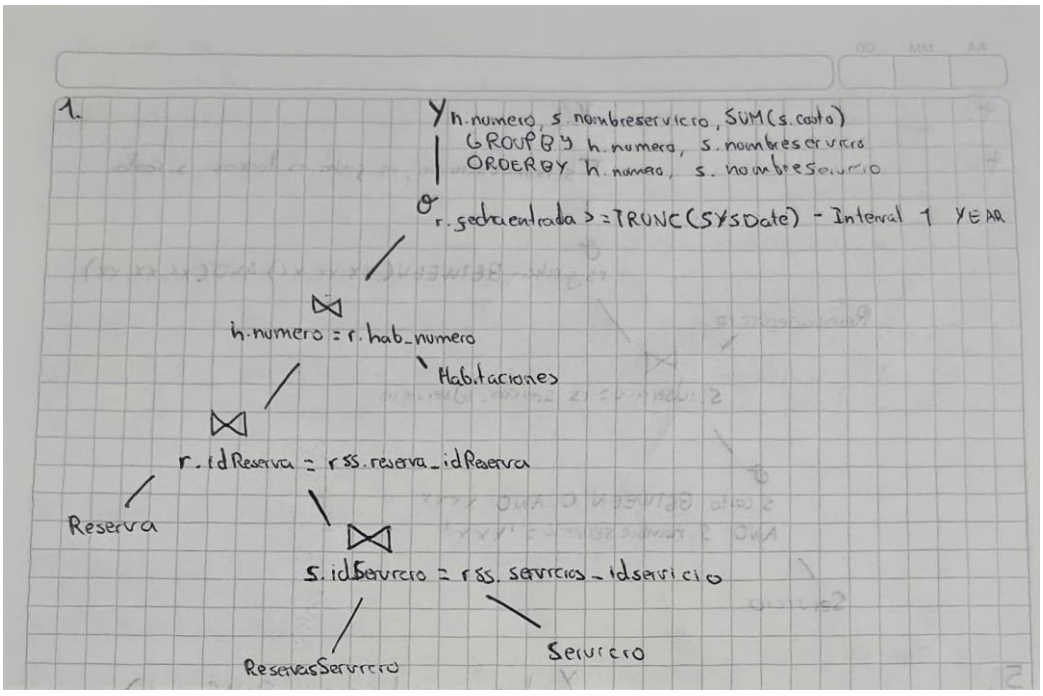
Se utiliza 2 FULL para acceder a la tabla reservaServicios y usuarios, es decir, un recorrido completo de la tabla.

Se utiliza 2 FAST FULL SCAN para acceder a Reserva_idReserva de la tabla ReservaServicio, es decir, un escaneo de una tabla mas eficiente

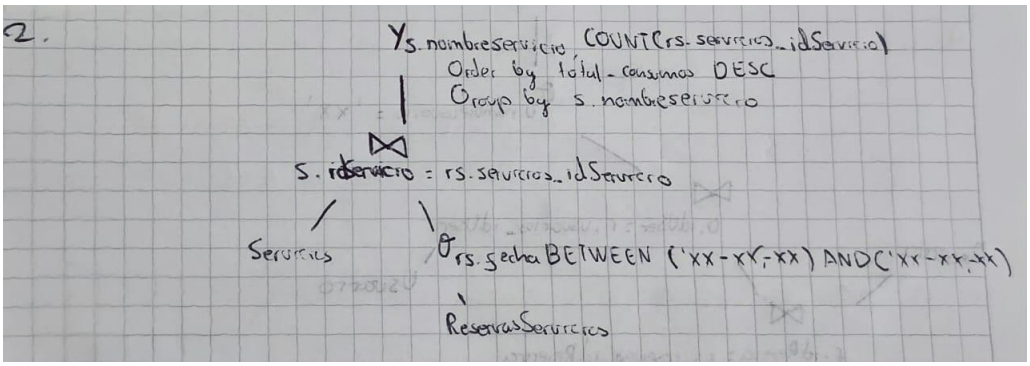
Se utiliza un UNIQUE SCAN para recuperar una fila especifica en la tabla reservas

PLANES DE EJECUCIÓN DE CADA REQUERIMIENTO:

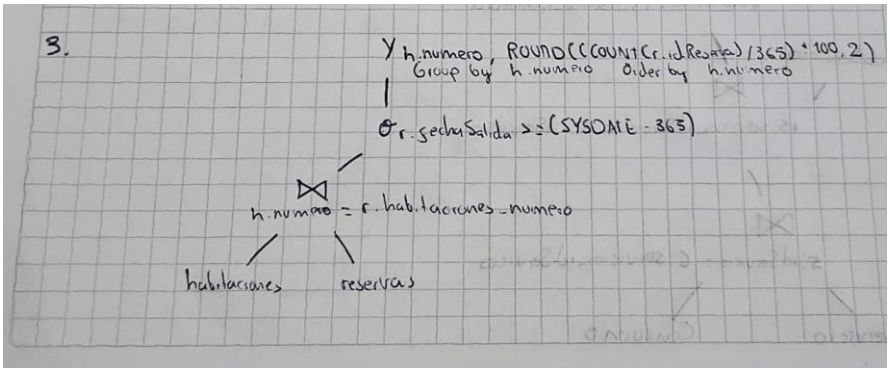
RF1:


1.
γ h.numero, s.nombreservicio, SUM(s.costo)
GROUP BY h.numero, s.nombreservicio
ORDER BY h.numero, s.nombreservicio
σ r.fechaentrada >= TRUNC(SYSDate) - Interval 1 YEAR
⋈
h.numero = r.hab_numero
Habitaciones
⋈
r.idReserva = rss.reserva_idReserva
Reserva
⋈
s.idServicio = rss.servicios_idservicio
ReservasServicios
Servicios

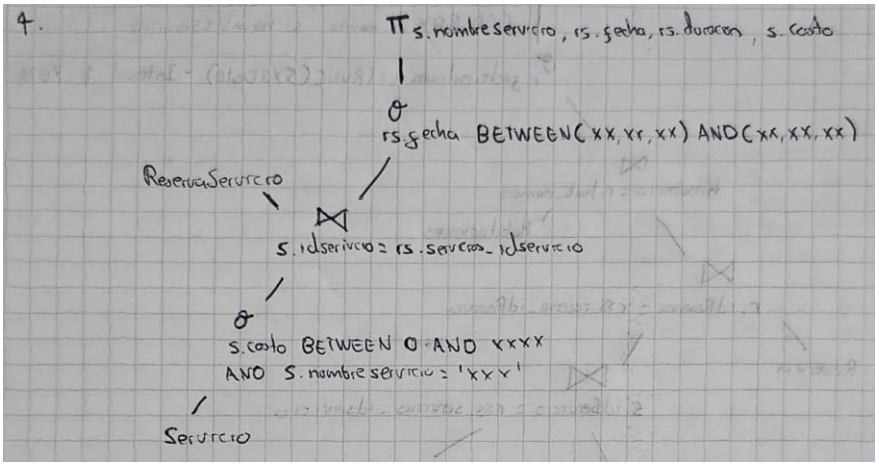
RF2:


2.
γ s.nombreservicio, COUNT(rs.servicios_idServicio)
Order by total_consumos DESC
Group by s.nombreservicio
⋈
s.idservicio = rs.servicios_idServicio
Servicios
σ rs.fecha BETWEEN ('XX-XX-XX') AND ('XX-XX-XX')
ReservasServicios

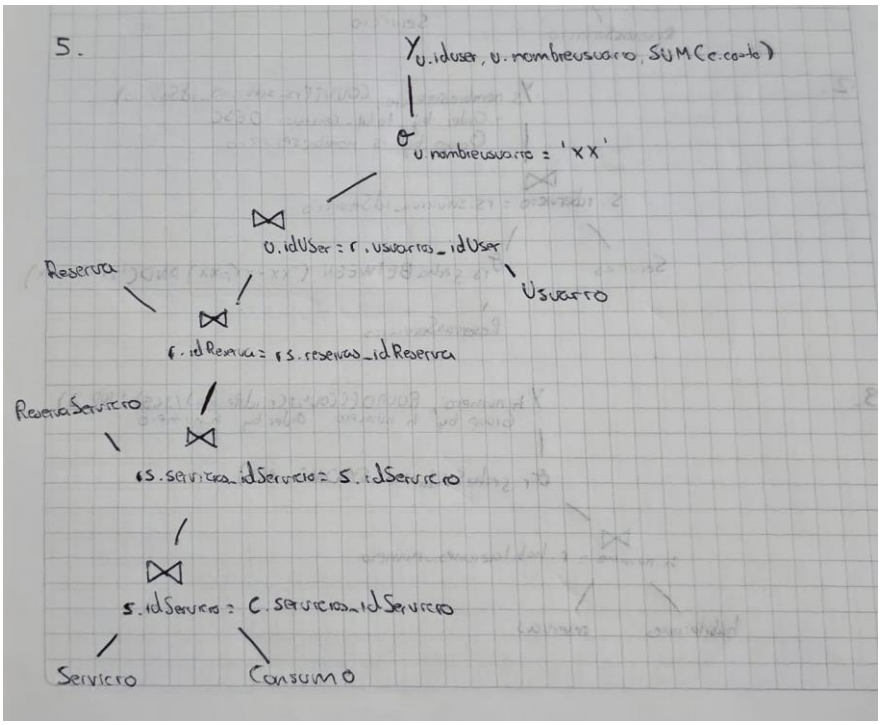
RF3:


3.
γ h.numero, ROUND((COUNT(r.idReserva)/365)*100,2)
Group by h.numero Order by h.numero
σ r.fechaSalida >= (SYSDATE - 365)
⋈
h.numero = r.habitaciones_numero
habitaciones
reservas

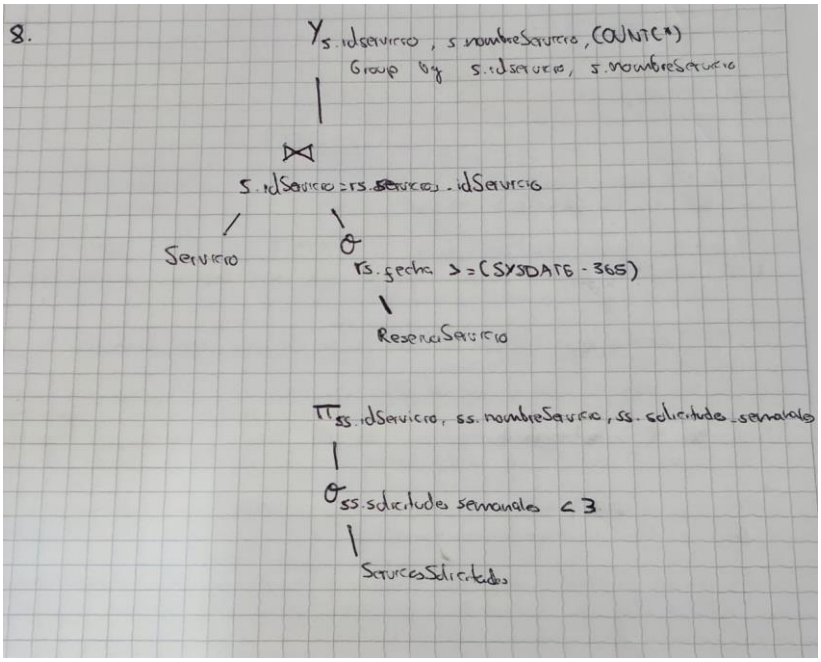
RF4


4.
π s.nombreservicio, rs.fecha, rs.duracion, s.costo
σ
rs.fecha BETWEEN(xx,xx,xx) AND(xx,xx,xx)
ReservaServicio
⋈
s.idservicio = rs.servicio_idservicio
σ
s.costo BETWEEN 0 AND xxxx
AND s.nombreservicio = 'xxx'
Servicio

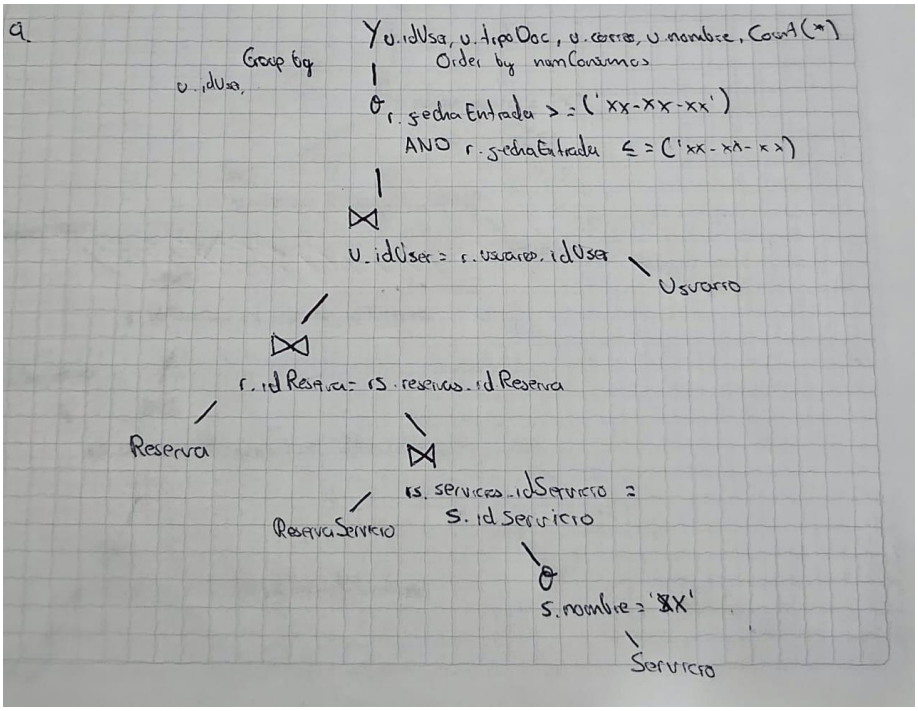
RF5:


5.
Y u.idUser, u.nombreUsuario, SUM(c.costo)
σ u.nombreUsuario = 'XX'
u.idUser = r.usuarios_idUser
Reserva
Usuarios
r.idReserva = rs.reservas_idReserva
ReservaServicio
rs.servicios_idServicio = s.idServicio
s.idServicio = c.servicios_idServicio
Servicio
Consumo

RF8:


8.
Y s.idServicio, s.nombreServicio, (COUNT(*)
Group by s.idServicio, s.nombreServicio
s.idServicio = rs.servicios_idServicio
Servicio
σ
rs.fecha >= (SYSDATE - 365)
ReservaServicio
Π ss.idServicio, ss.nombreServicio, ss.solicitudes_semanales
σ ss.solicitudes semanales < 3
ServiciosSolicitados

RF9:


a
Group by
u.idUser,
γ u.idUser, u.tipoDoc, u.correo, u.nombre, Count(*)
Order by numCanjes
σ r.fechaEntrada >= ('xx-xx-xx')
AND r.fechaEntrada <= ('xx-xx-xx')
u.idUser = r.usuario.idUser
Usuario
r.idReserva = rs.reserva.idReserva
Reserva
rs.servicio.idServicio =
s.idServicio
ReservaServicio
σ
s.nombre = 'X'
Servicio