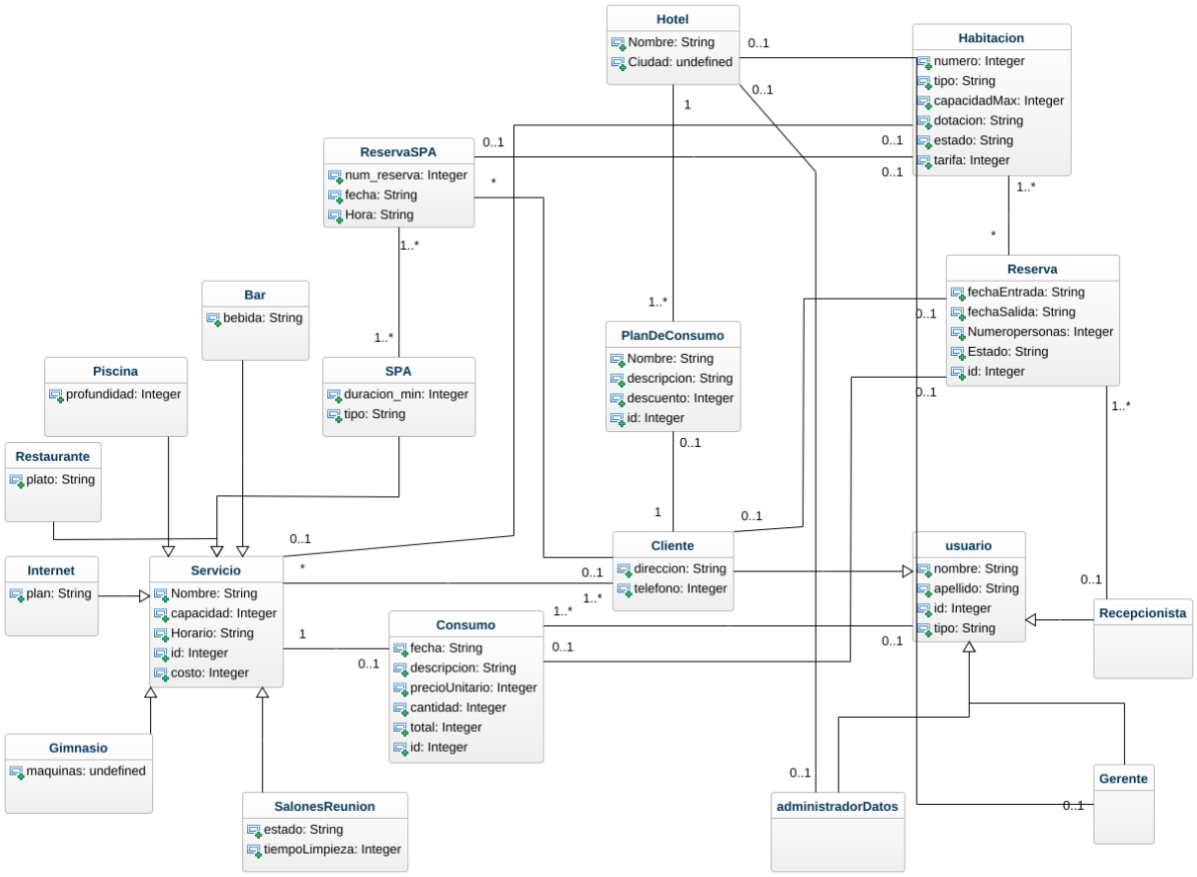
C10
Andres Serrano
Samuel Ramirez
Juana Mejia

Analisis

Al analizar el impacto que representa la introducción de los nuevos atributos se encontró que en el diseño inicial falto la clase de salones de reunión/conferencia, estos se volvieron una clase ya que lo único que los diferenciaba son la capacidad que tiene y el tiempo de limpieza luego de uso. El resto de clases y relaciones se mantuvieron igual debido a que no eran afectados por los nuevos requerimientos.



Diseño de Aplicación

Indices

1. Para el requerimiento funcional de “mostrar el dinero recolectado por servicios en cada habitación en el último año corrido” se podría crear un índice sobre las fechas de servicio para así poder encontrar los resultados que se cumplen con el rango presentado de un año y mejorar significativamente el rendimiento. También se podría poner un índice sobre los id de las habitaciones ya que se optimizaría el agrupamiento este atributo. Al ser un índice sobre el id de la habitación, esto implica que habrá alta selectividad dependiendo de la cantidad de cuartos que haya en el hotel. El tipo de índice para los mencionados anteriormente puede ser B+

2. EN el siguiente requerimiento “MOSTRAR LOS 20 SERVICIOS MÁS POPULARES.” Se puede crear un índice sobre la columna de los servicios, esto se debe a que con este se puede optimizar el rendimiento de la consulta y se acelerara la clasificación y recuento de los servicios. En términos de selectividad, como se tienen tantos servicios diferentes, este tiene un alto porcentaje y puede permitir que menos registros sean involucrados en la operación. Este índice puede ser de tipo B+
3. MOSTRAR EL ÍNDICE DE OCUPACIÓN DE CADA UNA DE LAS HABITACIONES DEL HOTEL: para este requerimiento se puede crear sobre los id de las habitaciones, esto ayudara a mejorar el rendimiento para la operación de agrupamiento por id de habitación y calcular el índice de ocupación de manera más rápida. El tipo de índice seria B+.
4. MOSTRAR LOS SERVICIOS QUE CUMPLEN CON CIERTA CARACTERÍSTICA: en este caso se deben identificar las consultas realizadas con mayor frecuencia, se podría hacer un índice para los precios de los servicios para así buscar en un rango, también se podría hacer un índice para la fecha de consumo que permitirá definir si hace parte del rango. Estos dos índices tendrán un alto nivel de selectividad debido a que hay una gran cantidad de precios y fechas por lo que la selectividad será bastante alta y permitirá reducir la cantidad de registros en la consulta. Estos índices podrán ser de tipo B+. Otro índice podría ser sobre empleado id, para definir cual empleado hizo el registro del servicio. Este puede no beneficiar el tiempo de la transacción debido a que pueden haber pocos empleados por lo que la selectividad será baja.
5. MOSTRAR EL CONSUMO EN HOTELANDES POR UN USUARIO DADO, EN UN RANGO DE FECHAS INDICADO: para este requerimiento se debe encontrar el consumo de un usuario en un rango de fechas, por esto sería beneficioso hacer un índice en la fecha de servicio para poder revisar si hace parte del rango, también se podría hacer un índice sobre el id del usuario que consumió el servicio. Estos dos índices tienen un alto nivel de selectividad
6. ANALIZAR LA OPERACIÓN DE HOTELANDES: Para este requerimiento se pueden crear los índices en la fecha de servicio y en el id de las habitaciones, estos permiten identificar rápidamente los días en los que el hotel tuvo la mayor ocupación. La mayoría del análisis implica encontrar fechas y al tener un índice sobre estas, se podrán hacer las consultas más rápido y optimo. Debido a que las fechas son la mayoría de los resultados para analizar la operación de hotelandes, es de suma importancia que sea un proceso rápido, esto se puede hacer en un árbol b+.
7. ENCONTRAR LOS BUENOS CLIENTES: Debido a que se deben conocer el número de días de la estadía y el costo consumido, es importante tener índices sobre la columna de clientes, para identificarlos más fácil, también un índice para fecha de servicio y para los consumos de clientes. Estos índices tienen altos niveles de selectividad debido a la cantidad de valores que pueden tomar y el hecho de que el hotel tiene bastantes clientes diferentes. Pueden ser de tipo B+
8. ENCONTRAR LOS SERVICIOS QUE NO TIENEN MUCHA DEMANDA: Para encontrar estos servicios se debe tener un índice sobre la columna de fecha del servicio debido a que se debe saber cuándo fue tomado el servicio para así determinar si fue utilizado menos de 3 veces. Al ser una fecha, habrán muchísimas posibilidades y permitirá que la selectividad sea de un porcentaje alto. Este índice puede ser de tipo B+.
9. CONSULTAR CONSUMO EN HOTELANDES: Se necesitarían los índices en el servicio para encontrar rápidamente servicios específicos, uno para el id del cliente y uno para la fecha del

servicio. Estos tres índices tienen una alta selectividad debido a que pueden tomar demasiados valores distintos. Pueden ser de árbol B+.

10. CONSULTAR CONSUMO EN HOTELANDES – RFC9-V2: para este requerimiento se debe tener un índice para el id del servicio, un índice para el id del cliente que toma el servicio, uno para la columna de la fecha de servicio. Todos estos para poder encontrar la información de los clientes que no han consumido un servicio determinado
11. CONSULTAR FUNCIONAMIENTO: se debe hacer un índice para la fecha de servicio, para el id de servicio y el numero de la habitación. Estos serán índices con alta selectividad debido a que existen muchos servicios, fechas y numero de habitaciones. Se podrá usar árbol B+
12. CONSULTAR LOS CLIENTES EXCELENTES: Para realizar esta consulta se podrían necesitar índices en el id del cliente, la fecha del checkin, checkout, costo de servicios y de id de servicio. Estos índices van a ayudar a filtrar las condiciones necesarias para encontrar a los clientes "excelentes". Estos índices serán de tipo B+

Carga de Datos

La carga de datos para esta iteración es un reto bastante importante, ya que el volumen que se solicita es alto. Se solicitan 750000 tuplas en las tablas. Claro que, este volumen no se logra en todas las tuplas. Ya que en ciertas tablas no tiene sentido alguno implementar 750000 tuplas, porque no se ajusta al modelo de la vida real. Estas tablas son: Tipo_usuario, Tipo_habitacion, Servicios, Bar, Restaurante, Spa, Empleado y Habitación. En las demás tablas si se logró llegar a tal cifra de 750000 tuplas. A continuación vamos a explicar cómo.

Insertar tupla por tupla es un proceso bastante demorado en el que no solo nos demoraríamos pensando cada valor individual. Aun así, si tuviéramos las 750000 sentencias sql, ejecutar un archivo de tal tamaño sentencia por sentencia es muy demorado. Por lo que para la creación de los datos nos apoyamos en el lenguaje Python con el editor VSCO.

En este creamos un archivo el cual nos iba permitir crear archivos csv. Estos archivos csv son una manera fácil de importar datos de gran cantidad en sqldeveloper. El archivo que se utilizó será adjuntado en el repositorio. Pero explicaremos su funcionamiento de manera breve.

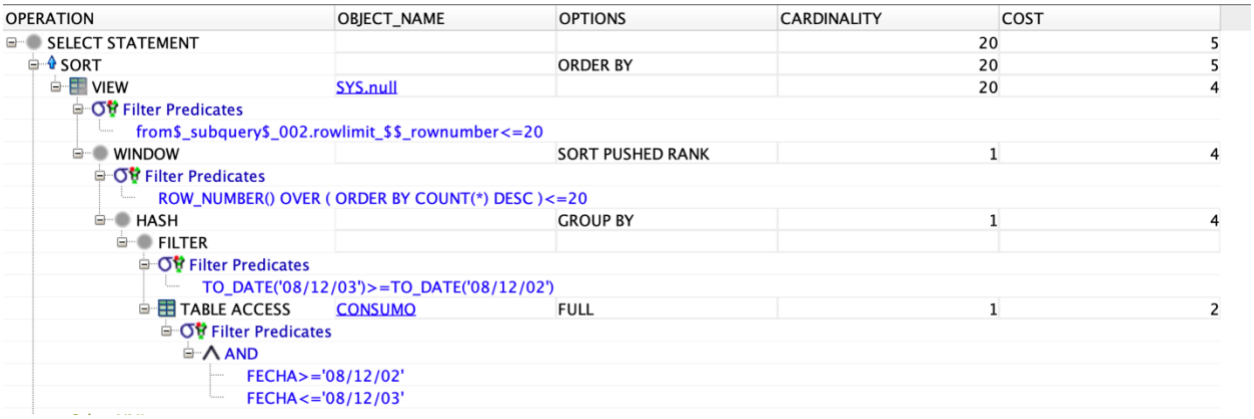
En total se crearon 16 archivos CSV, ya que se tenían 16 tablas y por cada tabla se creaba un archivo CSV. Cada tabla poseía una lista con los campos. Luego para crear cada una de las tuplas se hizo un for. Ese for iba desde 1 hasta 750000. Por cada iteración se generaba un dato aleatorio (esto teniendo en cuenta el tipo de dato) y se anexaba a una lista que contenía una tupla de datos (la tupla varía dependiendo de la tabla) y cada tupla de datos era anexada a una lista de tuplas, ya con toda la lista de tuplas se utilizó la función para crear, abrir y escribir en un csv para crear los 16 CSV. Ya luego de tener los 16 archivos CSV creados, en sql developer. Por cada tabla había la opción de importar los datos. Se importaron los datos y finalmente quedaron en la base de datos.

Diseño de las consultas

RFC2 - MOSTRAR LOS 20 SERVICIOS MÁS POPULARES.

```
SELECT SERVICIO_ID, COUNT(*) AS CANT_CONSUMOS

FROM CONSUMO

WHERE FECHA BETWEEN '08/12/02' AND '08/12/03'

GROUP BY SERVICIO_ID

ORDER BY CANT_CONSUMOS DESC;
```

Plan de ejecucion



Tiempo de la consulta:

```
-------------------------------------------------------------------------------------------
| Id  | Operation                | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |         |    20 |  1040 |     5  (60)| 00:00:01 |
|   1 |  SORT ORDER BY           |         |    20 |  1040 |     5  (60)| 00:00:01 |
|*  2 |   VIEW                   |         |    20 |  1040 |     4  (50)| 00:00:01 |
|*  3 |    WINDOW SORT PUSHED RANK|         |     1 |    22 |     4  (50)| 00:00:01 |
|   4 |     HASH GROUP BY        |         |     1 |    22 |     4  (50)| 00:00:01 |
|*  5 |      FILTER              |         |       |       |            |          |
|*  6 |       TABLE ACCESS FULL  | CONSUMO |     1 |    22 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

RFC3 - MOSTRAR EL ÍNDICE DE OCUPACIÓN DE CADA UNA DE LAS HABITACIONES DEL HOTEL

```
SELECT Reservanhabitaciones.habitacion_numero,((COUNT(FECHASALIDA-
FECHAENTRADA))/365)
```

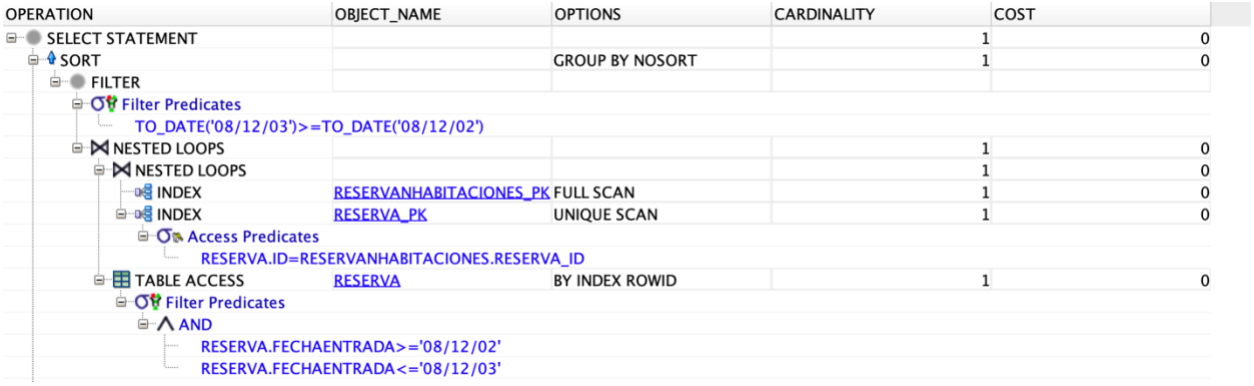
FROM RESERVA

INNER JOIN RESERVANHABITACIONES ON (RESERVA.ID = RESERVANHABITACIONES.RESERVA_ID)

WHERE reserva.fechaentrada BETWEEN '08/12/02' AND '08/12/03'

GROUP BY Reservanhabitaciones.habitacion_numero;

Plan de ejecucion



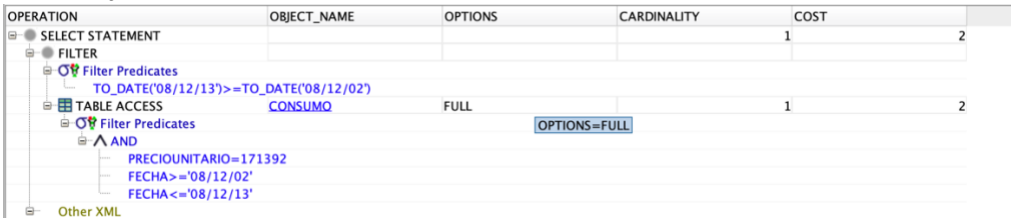
Tiempo de ejecución:

```
-----------------------------------------------------------------------------------------------------------
 Id  | Operation                     | Name                     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
   0 | SELECT STATEMENT              |                          |     1 |    57 |     0   (0)| 00:00:01 |
   1 |  SORT GROUP BY NOSORT         |                          |     1 |    57 |     0   (0)| 00:00:01 |
*  2 |   FILTER                      |                          |       |       |            |          |
   3 |    NESTED LOOPS               |                          |     1 |    57 |     0   (0)| 00:00:01 |
   4 |     NESTED LOOPS              |                          |     1 |    57 |     0   (0)| 00:00:01 |
   5 |      INDEX FULL SCAN          | RESERVANHABITACIONES_PK  |     1 |    26 |     0   (0)| 00:00:01 |
*  6 |      INDEX UNIQUE SCAN        | RESERVA_PK               |     1 |       |     0   (0)| 00:00:01 |
*  7 |     TABLE ACCESS BY INDEX ROWID| RESERVA                  |     1 |    31 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------
```

RFC4 - MOSTRAR LOS SERVICIOS QUE CUMPLEN CON CIERTA CARACTERÍSTICA

SELECT SERVICIO_ID
FROM CONSUMO
WHERE (FECHA BETWEEN '08/12/02' AND '08/12/13') AND (PRECIOUNITARIO = '171392');

Plan de ejecucion



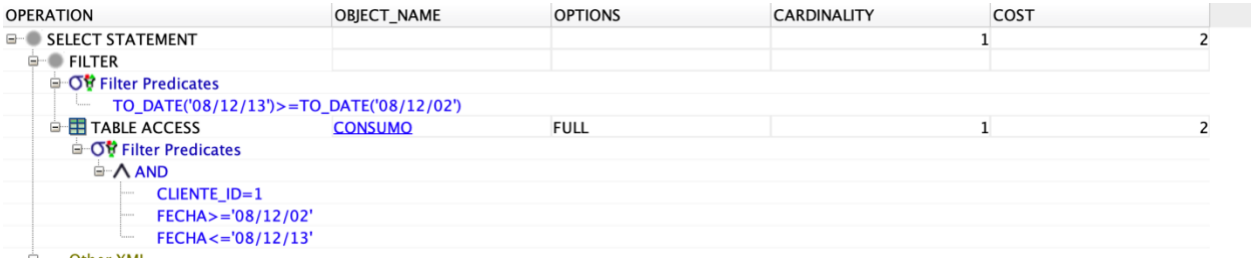
Tiempo de ejecusion:

```
-----------------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     1 |    35 |     2   (0)| 00:00:01 |
|*  1 |  FILTER            |          |       |       |            |          |
|*  2 |   TABLE ACCESS FULL| CONSUMO  |     1 |    35 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

RFC5 - MOSTRAR EL CONSUMO EN HOTELANDES POR UN USUARIO DADO, EN UN RANGO DE FECHAS INDICADO.

```
SELECT ID, DESCRIPCION, TOTAL, CLIENTE_ID, SERVICIO_ID
FROM CONSUMO
WHERE (CLIENTE_ID = 1) AND FECHA BETWEEN '08/12/02' AND '08/12/13';
```

Plan de ejecusion



Tiempo de ejecución

```
-----------------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     1 |   113 |     2   (0)| 00:00:01 |
|*  1 |  FILTER            |          |       |       |            |          |
|*  2 |   TABLE ACCESS FULL| CONSUMO  |     1 |   113 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

RFC6 - ANALIZAR LA OPERACIÓN DE HOTELANDES

```
//1
SELECT FECHAENTRADA, SUM(NUMEROPERSONAS) AS TOTAL_PERSONAS
FROM reserva
GROUP BY FECHAENTRADA
ORDER BY TOTAL_PERSONAS DESC;
```

Plan de ejecución:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	4
SORT		ORDER BY	1	4
HASH		GROUP BY	1	4
TABLE ACCESS	RESERVA	FULL	1	2

Tiempo:

```
-------------------------------------------------------------------------------------
| Id  | Operation           | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |         |     1 |    22 |     4  (50)| 00:00:01 |
|   1 |  SORT ORDER BY      |         |     1 |    22 |     4  (50)| 00:00:01 |
|   2 |   HASH GROUP BY     |         |     1 |    22 |     4  (50)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| RESERVA |     1 |    22 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

//2

```
SELECT FECHA, SUM(TOTAL) AS TOTAL_INGRESOS
FROM consumo
GROUP BY FECHA
ORDER BY TOTAL_INGRESOS DESC;
```

Plan de ejecución :

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	4
SORT		ORDER BY	1	4
HASH		GROUP BY	1	4
TABLE ACCESS	CONSUMO	FULL	1	2

Tiempo:

```
-------------------------------------------------------------------------------------
| Id  | Operation           | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |         |     1 |    22 |     4  (50)| 00:00:01 |
|   1 |  SORT ORDER BY      |         |     1 |    22 |     4  (50)| 00:00:01 |
|   2 |   HASH GROUP BY     |         |     1 |    22 |     4  (50)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| CONSUMO |     1 |    22 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

//3

```
SELECT FECHAENTRADA, SUM(NUMEROPERSONAS) AS TOTAL_PERSONAS
FROM reserva
GROUP BY FECHAENTRADA
ORDER BY TOTAL_PERSONAS Asc;
```

Plan de ejecución:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	4
SORT		ORDER BY	1	4
HASH		GROUP BY	1	4
TABLE ACCESS	RESERVA	FULL	1	2
Other XML				

Tiempo de ejecución:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	22	4 (50)	00:00:01
1	SORT ORDER BY		1	22	4 (50)	00:00:01
2	HASH GROUP BY		1	22	4 (50)	00:00:01
3	TABLE ACCESS FULL	RESERVA	1	22	2 (0)	00:00:01

RFC7 - ENCONTRAR LOS BUENOS CLIENTES

SELECT c.CLIENTE_ID, SUM(DISTINCT r.FECHASALIDA - r.FECHAENTRADA) AS TOTAL_DIAS,SUM(c.TOTAL) AS TOTAL_CONSUMO

FROM reserva r

JOIN consumo c ON r.ID = c.RESERVA_ID

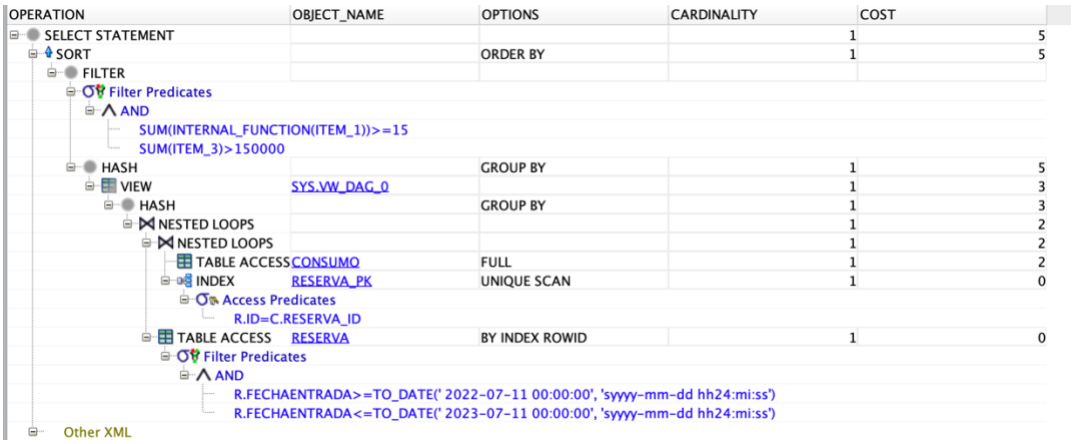
WHERE r.FECHAENTRADA BETWEEN '11/7/2022' AND '11/7/2023'

GROUP BY c.CLIENTE_ID

HAVING SUM(DISTINCT r.FECHASALIDA - r.FECHAENTRADA)>=15 AND SUM(c.TOTAL)> 150000

ORDER BY TOTAL_CONSUMO DESC, TOTAL_DIAS DESC;

Plan ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	5
SORT		ORDER BY	1	5
FILTER				
Filter Predicates				
AND				
SUM(INTERNAL_FUNCTION(ITEM_1))>=15				
SUM(ITEM_3)>150000				
HASH		GROUP BY	1	5
VIEW	SYS.VW_DAG_0		1	3
HASH		GROUP BY	1	3
NESTED LOOPS			1	2
NESTED LOOPS			1	2
TABLE ACCESS	CONSUMO	FULL	1	2
INDEX	RESERVA_PK	UNIQUE SCAN	1	0
Access Predicates				
R.ID=C.RESERVA_ID				
TABLE ACCESS	RESERVA	BY INDEX ROWID	1	0
Filter Predicates				
AND				
R.FECHAENTRADA>=TO_DATE(' 2022-07-11 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
R.FECHAENTRADA<=TO_DATE(' 2023-07-11 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
Other XML				

Tiempo Ejecución

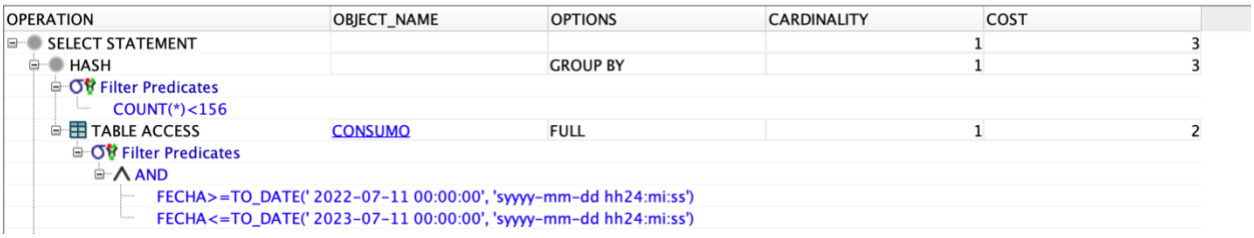
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	32	5 (60)	00:00:01
1	SORT ORDER BY		1	32	5 (60)	00:00:01
* 2	FILTER					
3	HASH GROUP BY		1	32	5 (60)	00:00:01
4	VIEW	VW_DAG_0	1	32	3 (34)	00:00:01
5	HASH GROUP BY		1	70	3 (34)	00:00:01
6	NESTED LOOPS		1	70	2 (0)	00:00:01
7	NESTED LOOPS		1	70	2 (0)	00:00:01
8	TABLE ACCESS FULL	CONSUMO	1	39	2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	RESERVA_PK	1		0 (0)	00:00:01
* 10	TABLE ACCESS BY INDEX ROWID	RESERVA	1	31	0 (0)	00:00:01

RFC8 - ENCONTRAR LOS SERVICIOS QUE NO TIENEN MUCHA DEMANDA

SELECT SERVICIO_ID, COUNT(*) AS NUMERO_VECES_SOLICITADO

FROM consumo
WHERE FECHA BETWEEN '11/7/2022' AND '11/7/2023'
GROUP BY SERVICIO_ID
HAVING COUNT(*) < 3 * 52;

Plan de ejecución:



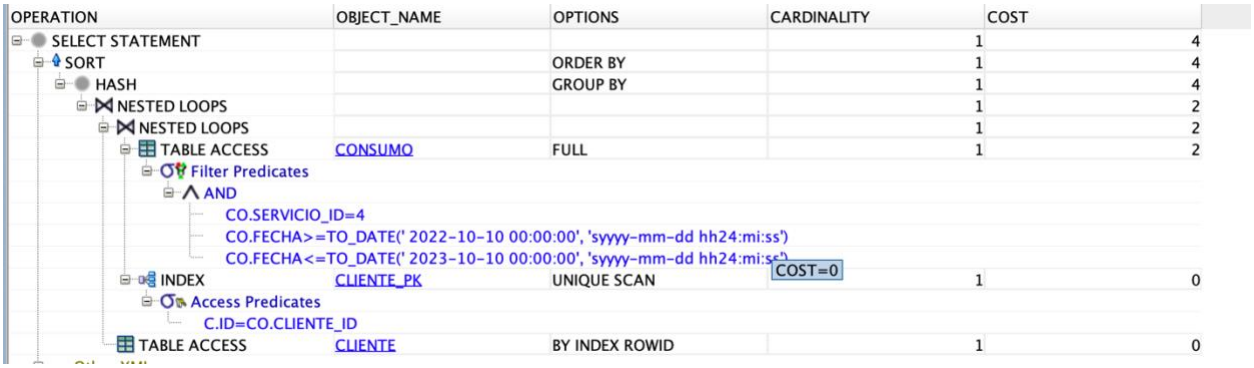
Tiempo de Ejecución:

```
-------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |     1 |    22 |     3  (34)| 00:00:01 |
|*  1 |  HASH GROUP BY     |         |     1 |    22 |     3  (34)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| CONSUMO |     1 |    22 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

RFC9 - CONSULTAR CONSUMO EN HOTELANDES

```
SELECT c.ID AS CLIENTE_ID,c.DIRECCION, c.TELEFONO, COUNT(*) AS NUMERO_CONSUMOS
FROM cliente c
LEFT JOIN consumo co ON c.ID = co.CLIENTE_ID
WHERE co.SERVICIO_ID = 4
AND co.FECHA  BETWEEN '10/10/2022' AND '10/10/2023'
GROUP BY c.ID, c.DIRECCION, c.TELEFONO
ORDER BY COUNT(*) DESC;
```

Plan de ejecución



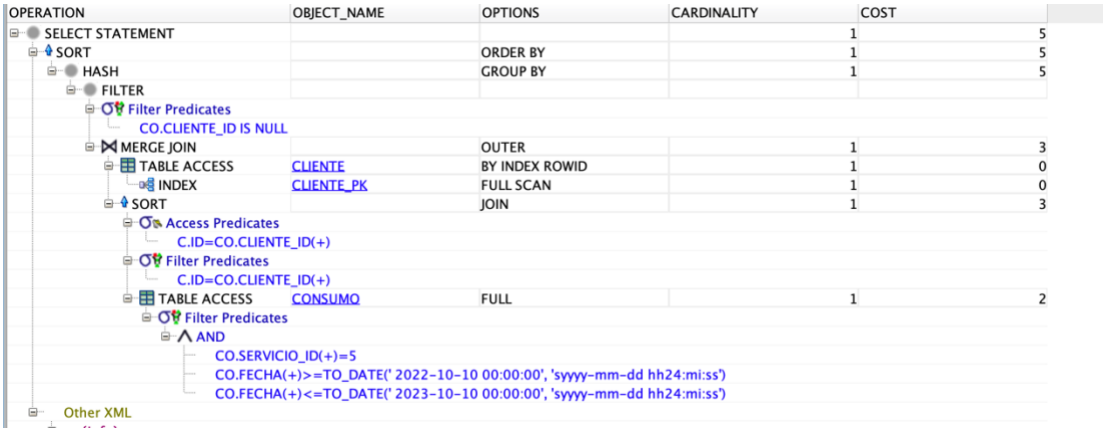
Tiempo de ejecución:

```
-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |   113 |     4  (50)| 00:00:01 |
|   1 |  SORT ORDER BY               |             |     1 |   113 |     4  (50)| 00:00:01 |
|   2 |   HASH GROUP BY              |             |     1 |   113 |     4  (50)| 00:00:01 |
|   3 |    NESTED LOOPS              |             |     1 |   113 |     2   (0)| 00:00:01 |
|   4 |     NESTED LOOPS             |             |     1 |   113 |     2   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL       | CONSUMO     |     1 |    35 |     2   (0)| 00:00:01 |
|*  6 |      INDEX UNIQUE SCAN       | CLIENTE_PK  |     1 |       |     0   (0)| 00:00:01 |
|   7 |     TABLE ACCESS BY INDEX ROWID| CLIENTE   |     1 |    78 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

RFC10 - CONSULTAR CONSUMO EN HOTELANDES – RFC9-V2

```
SELECT c.ID AS CLIENTE_ID, c.DIRECCION, c.TELEFONO, COUNT(*)
FROM cliente c
LEFT JOIN consumo co ON c.ID = co.CLIENTE_ID
AND co.SERVICIO_ID = 5
AND co.FECHA  BETWEEN '10/10/2022' AND '10/10/2023'
WHERE co.CLIENTE_ID IS NULL
GROUP BY c.ID, c.DIRECCION, c.TELEFONO
ORDER BY COUNT(*) ;
```

Plan de ejecución:



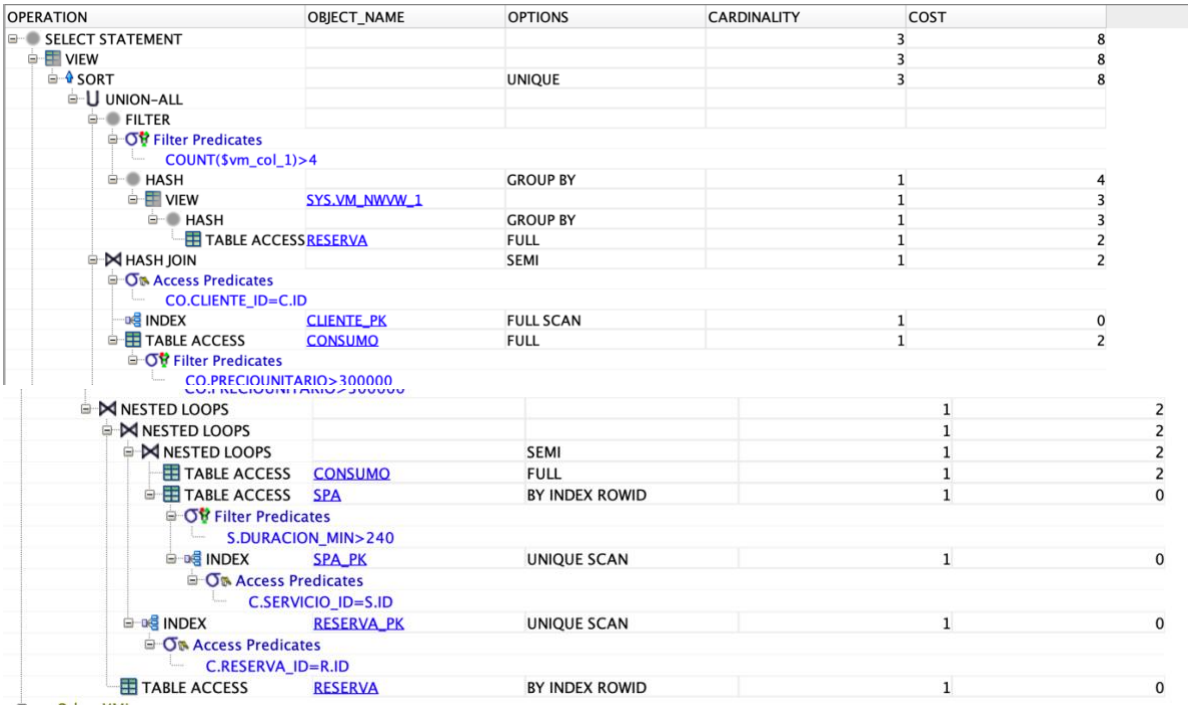
Tiempo de ejecución:

```
-------------------------------------------------------------------------------------------
| Id  | Operation                      | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |            |     1 |   113 |     5  (60)| 00:00:01 |
|   1 |  SORT ORDER BY                 |            |     1 |   113 |     5  (60)| 00:00:01 |
|   2 |   HASH GROUP BY                |            |     1 |   113 |     5  (60)| 00:00:01 |
|*  3 |    FILTER                      |            |       |       |            |          |
|   4 |     MERGE JOIN OUTER           |            |     1 |   113 |     3  (34)| 00:00:01 |
|   5 |      TABLE ACCESS BY INDEX ROWID| CLIENTE    |     1 |    78 |     0   (0)| 00:00:01 |
|   6 |       INDEX FULL SCAN          | CLIENTE_PK |     1 |       |     0   (0)| 00:00:01 |
|*  7 |      SORT JOIN                 |            |     1 |    35 |     3  (34)| 00:00:01 |
|*  8 |       TABLE ACCESS FULL        | CONSUMO    |     1 |    35 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

RFC12 - CONSULTAR LOS CLIENTES EXCELENTES

```
SELECT cliente_id
FROM ((
   SELECT r.cliente_id
   FROM reserva r
   GROUP BY r.cliente_id, EXTRACT(YEAR FROM r.fechaentrada)
   HAVING COUNT(DISTINCT EXTRACT(YEAR FROM r.fechaentrada)) > 4
)
   UNION
   SELECT c.ID AS cliente_id
   FROM cliente c
   WHERE EXISTS (
      SELECT 1
      FROM consumo co
      WHERE co.CLIENTE_ID = c.ID
      AND co.PRECIOUNITARIO > 300000
   )
   UNION
   SELECT r.cliente_id
   FROM reserva r
   JOIN consumo c ON c.RESERVA_ID = r.ID
   JOIN spa s ON c.SERVICIO_ID = s.ID
   WHERE s.DURACION_MIN > (4 * 60)
);
```

Plan de ejecución:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	8
VIEW			3	8
SORT		UNIQUE	3	8
UNION-ALL				
FILTER				
Filter Predicates				
COUNT($vm_col_1)>4				
HASH		GROUP BY	1	4
VIEW	SYS.VM_NWVW_1		1	3
HASH		GROUP BY	1	3
TABLE ACCESS	RESERVA	FULL	1	2
HASH JOIN		SEMI	1	2
Access Predicates				
CO.CLIENTE_ID=C.ID				
INDEX	CLIENTE_PK	FULL SCAN	1	0
TABLE ACCESS	CONSUMO	FULL	1	2
Filter Predicates				
CO.PRECIOUNITARIO>300000				
NESTED LOOPS			1	2
NESTED LOOPS			1	2
NESTED LOOPS		SEMI	1	2
TABLE ACCESS	CONSUMO	FULL	1	2
TABLE ACCESS	SPA	BY INDEX ROWID	1	0
Filter Predicates				
S.DURACION_MIN>240				
INDEX	SPA_PK	UNIQUE SCAN	1	0
Access Predicates				
C.SERVICIO_ID=S.ID				
INDEX	RESERVA_PK	UNIQUE SCAN	1	0
Access Predicates				
C.RESERVA_ID=R.ID				
TABLE ACCESS	RESERVA	BY INDEX ROWID	1	0

Tiempo de ejecución:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	39	8 (25)	00:00:01
1	VIEW		3	39	8 (25)	00:00:01
2	SORT UNIQUE		3	143	8 (25)	00:00:01
3	UNION-ALL					
* 4	FILTER					
5	HASH GROUP BY		1	26	4 (50)	00:00:01
6	VIEW	VM_NWVW_1	1	26	3 (34)	00:00:01
7	HASH GROUP BY		1	22	3 (34)	00:00:01
8	TABLE ACCESS FULL	RESERVA	1	22	2 (0)	00:00:01
* 9	HASH JOIN SEMI		1	39	2 (0)	00:00:01
10	INDEX FULL SCAN	CLIENTE_PK	1	13	0 (0)	00:00:01
* 11	TABLE ACCESS FULL	CONSUMO	1	26	2 (0)	00:00:01
12	NESTED LOOPS		1	78	2 (0)	00:00:01
13	NESTED LOOPS		1	78	2 (0)	00:00:01
14	NESTED LOOPS SEMI		1	52	2 (0)	00:00:01
15	TABLE ACCESS FULL	CONSUMO	1	26	2 (0)	00:00:01
* 16	TABLE ACCESS BY INDEX ROWID	SPA	1	26	0 (0)	00:00:01
* 17	INDEX UNIQUE SCAN	SPA_PK	1		0 (0)	00:00:01
* 18	INDEX UNIQUE SCAN	RESERVA_PK	1		0 (0)	00:00:01
19	TABLE ACCESS BY INDEX ROWID	RESERVA	1	26	0 (0)	00:00:01