

**Entrega 2**  
**Hotel Andes**  
**B9**

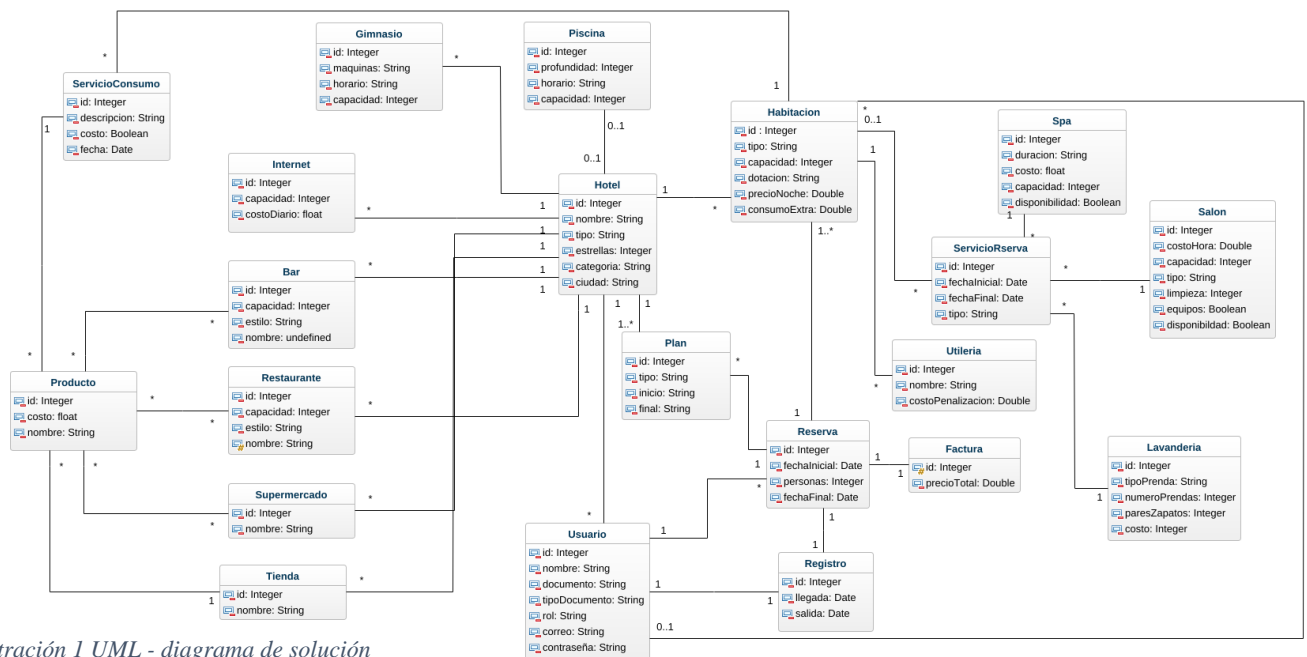
Pablo R Santiago Peñaranda - 201922871

Adriana Sofía Rozo Cepeda – 202211498

Nicolas David Camargo Prieto - 202020782

**Objetivo:** Integrar requerimientos de eficiencia a una aplicación transaccional desarrollada en una arquitectura de tres niveles, en una aplicación cliente/servidor y manejo de persistencia en base de datos relacionales.

## Análisis



*Ilustración 1 UML - diagrama de solución*

Para esta entrega lo único que incluimos fue dos nuevos atributos a la clase ServicioReserva.

## Diseño de la aplicación

## Creación de datos

Para la creación de los 750,000 datos se hizo uso de la página Mockaroo, donde para cada tabla se generaron los siguientes datos por tabla teniendo en cuenta las 14 tablas que usamos:

- -factura 1.000 datos
- -habitación 50.000 datos
- -hotel 1.000 datos
- -lavanderia 1.000 datos

- -plan 1.000 datos
- -producto 150.000 datos
- -registro 150.000 datos
- -reserva 100.000 datos
- -salon 1.000 datos
- -servicioconsumo 100.000 datos
- -servicioreservas 100.000 datos
- -spa 1.000 datos
- -usuario 100.000 datos
- -utilleria 1.000 datos

Se crearon los datos como csv y posteriormente, se juntaron usando mergecsv los de cada tabla puesto que el máximo es de 1.000 datos por generación gratis en la página de Mockaroo. Luego, se procedió a importar los datos de cada tabla por medio de la opción de importar datos por csv en Oracle developer.

### Indices

- Para cada uno de los requerimientos funcionales: Identifiquen si es necesario crear un índice.
- Justifiquen la selección del índice creado desde el punto de vista de la selectividad (concepto visto en clase). En caso de que para algún requerimiento funcional Uds determinen que un índice no es necesario, justifíquelo también desde el punto de vista de la selectividad.
- Indiquen claramente cuál es el tipo de índice utilizado (B+, Hash, ..., primario, secundario).

Según su modelo de datos, para los índices creados de forma automática por Oracle:

- Incluyan una captura de pantalla con la información generada por Oracle asociada a los índices existentes.
- Analicen los índices encontrados. Específicamente, Analicen por qué fueron creados por Oracle y si ayudan al rendimiento de los requerimientos funcionales.

### Requerimientos

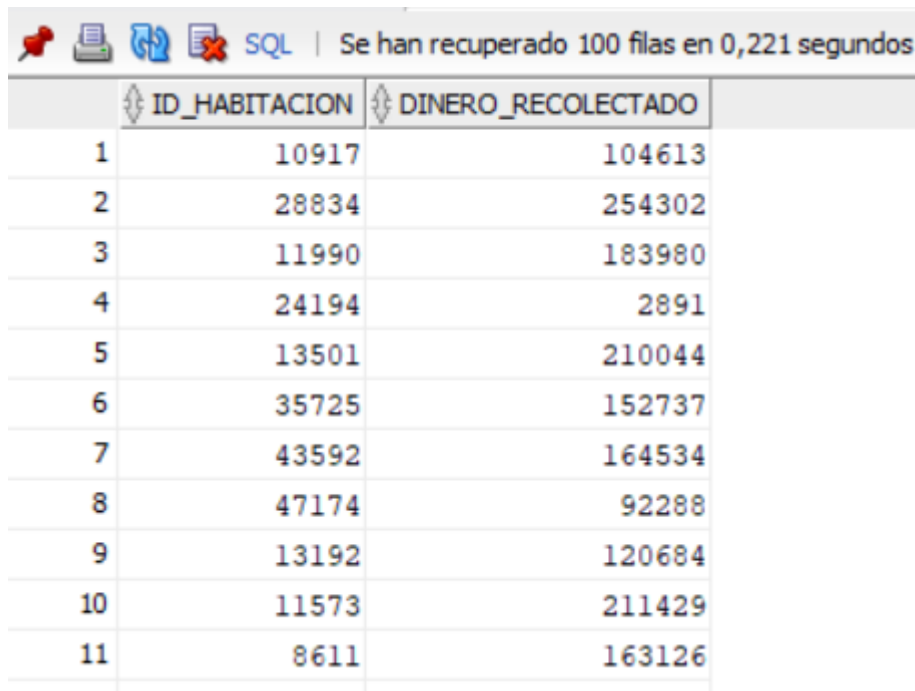
**RFC1 - Mostrar el dinero recolectado por servicios en cada habitación en el último año corrido.**

```
SELECT
    h.id AS id_habitacion,
    SUM(CASE
        WHEN sr.precio IS NOT NULL THEN sr.precio
        ELSE 0
    END) AS dinero_recolectado
FROM HABITACIONES h
LEFT JOIN SERVICIORESERVAS sr ON h.id = sr.idhabitacion
WHERE sr.fechainicial BETWEEN ADD_MONTHS(SYSDATE, -12) AND SYSDATE
GROUP BY h.id;
```

La consulta contiene las condiciones en el where:

Condición	Razonamiento	Necesidad de índice	Tipo índice
WHERE sr.fechainicial BETWEEN ADD_MONTHS(SYSDATE, - 12) AND SYSDATE	obtener la suma de los precios de los servicios relacionados con las habitaciones en un período de 12 meses, de modo que puedas calcular cuánto dinero se ha recolectado en ese período.	Dado que estás realizando una operación de filtrado basada en esta columna, un índice puede acelerar la búsqueda de registros que cumplan con la condición. La necesidad del índice se basa en la frecuencia de las consultas y la cantidad de datos en la tabla.	Un índice B-tree

Sin la creación de los índices devuelve los siguientes resultados:



	ID_HABITACION	DINERO_RECOLECTADO
1	10917	104613
2	28834	254302
3	11990	183980
4	24194	2891
5	13501	210044
6	35725	152737
7	43592	164534
8	47174	92288
9	13192	120684
10	11573	211429
11	8611	163126

Después de la creación de índices:

```
CREATE INDEX idx_fechainicial ON SERVICIORESERVAS (fechainicial);
```

Presenta una eficiencia menor:

SQL   Se han recuperado 50 filas en 0,059 segundos			
	ID_HABITACION	DINERO_RECOLECTADO	
1	10917	104613	
2	28834	254302	
3	11990	183980	
4	24194	2891	
5	13501	210044	
6	35725	152737	
7	43592	164534	
8	47174	92288	
9	13192	120684	
10	11573	211429	
11	8611	163126	
12	23845	1222	
13	33011	135336	

Plan de la consulta:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5	70	5 (20)	00:00:01
1	HASH GROUP BY		5	70	5 (20)	00:00:01
* 2	FILTER					
* 3	TABLE ACCESS FULL	SERVICIORESERVAS	16	224	4 (0)	00:00:01

Predicate Information (identified by operation id):						
-----						
2	filter(SYSDATE@!>=ADD_MONTHS(SYSDATE@!,(-12)))					
3	filter("SR"."FECHAINICIAL"<=SYSDATE@! AND "SR"."FECHAINICIAL">=ADD_MONTHS(SYSDATE@!,(-12)) AND "SR"."IDHABITACION" IS NOT NULL)					

**RFC2 - Mostrar los 20 servicios más populares.**

```
SELECT UPPER(S.TIPO) AS SERVICIO,
       COUNT(*) AS FRECUENCIA_TOTAL
FROM SERVICIORESERVAS S
WHERE S.FECHAINICIAL BETWEEN DATE '2023-10-01' AND DATE '2023-10-15'
GROUP BY UPPER(S.TIPO)
ORDER BY FRECUENCIA_TOTAL DESC
FETCH FIRST 20 ROWS ONLY;
```

Se utilizaron las fechas '2023-10-01' y '2023-10-15' para ejemplificar.

Condición	Razonamiento	Necesidad de índice	Tipo índice
WHERE S.FECHAINICIAL BETWEEN DATE '2023-10-01' AND DATE '2023-10-15'	esta consulta es obtener una lista de servicios junto con su frecuencia de uso (número de reservas) en un período de tiempo específico.	. Dado que estás realizando una operación de filtrado basada en esta columna, un índice puede acelerar la búsqueda de registros que cumplan con la condición.	Un índice B-tree

Sin la creación de índices el tiempo de consulta es el siguiente:

 <span>Todas las Filas Recuperadas: 8 en 0,093 segundos</span>			
	SERVICIO	FRECUENCIA_TOTAL	
1	LAVANDERIA	410	
2	INTERNET	408	
3	SPA	408	
4	GIMNASIO	404	
5	UTENSILIOS	401	
6	PISCINA	399	
7	SALON GRANDE	398	
8	SALON	382	

Luego de crear los índices:

```
CREATE INDEX idx_fechainicial ON SERVICIORESERVAS (FECHAINICIAL);
```

 <span>Todas las Filas Recuperadas: 8 en 0,025 segundos</span>			
	SERVICIO	FRECUENCIA_TOTAL	
1	LAVANDERIA	410	
2	INTERNET	408	
3	SPA	408	
4	GIMNASIO	404	
5	UTENSILIOS	401	
6	PISCINA	399	
7	SALON GRANDE	398	
8	SALON	382	

El plan es el siguiente:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		20	3360	7 (43)	00:00:01
1	SORT ORDER BY		20	3360	7 (43)	00:00:01
* 2	VIEW		20	3360	6 (34)	00:00:01
* 3	WINDOW SORT PUSHED RANK		4	64	6 (34)	00:00:01
4	HASH GROUP BY		4	64	6 (34)	00:00:01
* 5	TABLE ACCESS FULL	SERVICIORESERVAS	13	208	4 (0)	00:00:01
Predicate Information (identified by operation id):						
2 - filter("from\$_subquery\$_002"."rowlimit_\$\$_rownumber"<=20)						
3 - filter(ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC )<=20)						
5 - filter("S"."FECHAINICIAL"<=TO_DATE(' 2023-10-15 00:00:00', 'yyyy-mm-dd hh24:mi:ss') AND "S"."FECHAINICIAL">=TO_DATE(' 2023-10-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss'))						

RFC3 - Mostrar el índice de ocupación de cada una de las habitaciones del hotel

```
SELECT r.habitacionid AS numeroHabitacion,
       ROUND(SUM(MONTHS_BETWEEN(r.fechafinal, r.fecha) / 12.0) * 100, 2) AS porcentajeOcupacion
FROM reservas r
WHERE r.fecha BETWEEN ADD_MONTHS(SYSDATE, -12) AND SYSDATE
GROUP BY r.habitacionid;
```

Condición	Razonamiento	Necesidad de índice	Tipo índice
-----------	--------------	---------------------	-------------

WHERE r.fecha BETWEEN ADD_MONTHS(SYSDATE, - 12) AND SYSDATE	es calcular el porcentaje de ocupación de cada habitación en un período de 12 meses. Esto se hace tomando en cuenta la diferencia entre la fecha de inicio y la fecha de finalización de cada reserva.	Dado que estás realizando una operación de filtrado basada en esta columna, un índice puede acelerar la búsqueda de registros que cumplan con la condición. La necesidad del índice se basa en la frecuencia de las consultas y la cantidad de datos en la tabla.	Un índice B-tree
---	---	---	------------------

La creación del índice genera la siguiente eficiencia en la consulta:

SQL   Se han recuperado 50 filas en 0,108 segundos			
	NUMEROHABITACION	PORCENTAJEOCUPACION	
1	22296	35,48	
2	38182	80,11	
3	35640	17,74	
4	41743	-0,27	
5	34023	12,9	
6	1019	-23,39	
7	28003	-168,55	
8	38194	20,43	
9	34538	-49,19	

El plan corresponde al siguiente:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		8	152	5 (20)	00:00:01
1	HASH GROUP BY		8	152	5 (20)	00:00:01
* 2	FILTER					
* 3	TABLE ACCESS FULL	RESERVAS	10	190	4 (0)	00:00:01
Predicate Information (identified by operation id):						
-----						
2 - filter(SYSDATE@!>=ADD_MONTHS(SYSDATE@!, (-12)))						
3 - filter("R"."FECHA"<=SYSDATE@! AND						
"R"."FECHA">=ADD_MONTHS(SYSDATE@!, (-12)))						

**RFC4 - Mostrar los servicios que cumplen con cierta característica**

```

SELECT S.id as servicio_id,
       S.fecha_inicial as fecha_inicial,
       S.fecha_final as fecha_final,
       S.id_habitacion as id_habitacion,
       S.tipo as tipo_servicio,
       S.precio as precio_servicio
FROM SERVICIORESERVAS S
WHERE (:fechaInicio IS NULL OR S.FECHAINICIAL BETWEEN :fechaInicio AND :fechaFin)
AND (:rangoPrecioMin IS NULL OR S.PRECIO >= :rangoPrecioMin)
AND (:rangoPrecioMax IS NULL OR S.PRECIO <= :rangoPrecioMax)
AND (:tipoServicio IS NULL OR UPPER(S.TIPO) = UPPER(:tipoServicio));


```

Condición	Razonamiento	Necesidad de	Tipo índice
WHERE (:fechaInicio IS NULL OR S.FECHAINICIAL BETWEEN :fechaInicio AND :fechaFin)	El razonamiento o detrás de esta consulta es recuperar registros de la tabla SERVICIORESERVAS que cumplan con los criterios de filtro	Si esta consulta se ejecuta con frecuencia y la tabla contiene una cantidad significativa de datos, podrías considerar crear índices en las columnas utilizadas en las condiciones de filtrado, es decir, FECHAINICIAL, PRECIO y TIPO. La	índices B- tree o índices de rango
AND (:rangoPrecioMin IS	proporciona	dos por el	
(:rangoPrecioMax IS NULL OR	usuario o	que no se	
(:tipoServicio IS NULL OR UPPER(S.TIPO) =	proporcione n, en caso de que los		

Para ejemplificar se realizará la consulta de servicios que estén en precio entre 1 y 100 para el servicio de spa.



Salida de Script x Resultado de la Consulta x

 SQL | Todas las Filas Recuperadas: 20 en 0,066 segundos

SERVICIO_ID	FECHA_INICIAL	FECHA_FINAL	ID_HABITACION	TIPO_SERVICIO	PRECIO_SERVICIO
1	12 01/10/23	05/10/23		2 spa	10
2	13 01/10/23	05/10/23		2 spa	10
3	14 01/10/23	05/10/23		3 spa	10
4	17 01/10/23	05/10/23		10 spa	10
5	20 01/10/23	05/10/23		9 spa	10
6	38048 23/01/23	10/03/23	43912	Spa	27
7	2441 23/11/22	17/05/23	44549	Spa	39
8	10722 25/09/23	10/02/23	13197	Spa	86
9	32918 28/10/23	10/12/22	33507	Spa	61
10	17667 31/03/23	02/03/23	4345	Spa	2
11	24208 07/04/23	04/09/23	17273	Spa	71
12	54373 01/05/23	10/05/23	22392	Spa	32
13	60592 13/04/23	21/01/23	39114	Spa	56
14	41497 31/07/23	03/01/23	2445	Spa	60


Para la creación de índices utilizaremos los siguientes:

```
CREATE INDEX idxPrecio ON SERVICIORESERVAS (PRECIO);
```

```
CREATE INDEX idxTipoServicio ON SERVICIORESERVAS (UPPER(TIPO));
```

Utilizando los índices se obtiene la siguiente eficiencia:

Salida de Script x Resultado de la Consulta x

 | Todas las Filas Recuperadas: 20 en 0,042 segundos

SERVICIO_ID	FECHA_INICIAL	FECHA_FINAL	ID_HABITACION	TIPO_SERVICIO	PRECIO_SERVICIO
1	12 01/10/23	05/10/23	2	spa	10
2	13 01/10/23	05/10/23	2	spa	10
3	14 01/10/23	05/10/23	3	spa	10
4	17 01/10/23	05/10/23	10	spa	10
5	20 01/10/23	05/10/23	9	spa	10
6	38048 23/01/23	10/03/23	43912	Spa	27
7	2441 23/11/22	17/05/23	44549	Spa	39
8	10722 25/09/23	10/02/23	13197	Spa	86
9	32918 28/10/23	10/12/22	33507	Spa	61
10	17667 31/03/23	02/03/23	4345	Spa	2

El plan propuesto es el siguiente:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	32	4 (0)	00:00:01
* 1	TABLE ACCESS FULL	SERVICIORESERVAS	1	32	4 (0)	00:00:01

Predicate Information (identified by operation id):

```

1 - filter((:TIPOSERVICIO IS NULL OR UPPER("TIPO")=UPPER(:TIPOSERVICIO))
AND (:FECHAINICIO IS NULL OR "S"."FECHAINICIAL">=:FECHAINICIO AND
"S"."FECHAINICIAL"<=:FECHAFIN) AND ("S"."PRECIO">=TO_NUMBER(:RANGOPRECIOMIN)
OR :RANGOPRECIOMIN IS NULL) AND ("S"."PRECIO"<=TO_NUMBER(:RANGOPRECIOMAX) OR
:RANGOPRECIOMAX IS NULL))

```

**RFC5 - Mostrar el consumo en hotelandes por un usuario dado, en un rango de fechas indicado. Recuerde que un cliente puede alojarse en el hotel cuantas veces quiera.**

```
SELECT U.NOMBRE AS NOMBRE_USUARIO, P.NOMBRE AS NOMBRE_PRODUCTO, SC.COSTO AS COSTO_CONSUMO, SC.FECHA
FROM SERVICIOSCONSUMO SC
JOIN PRODUCTOS P ON SC.IDPRODUCTO = P.ID
JOIN RESERVAS R ON SC.IDHABITACION = R.HABITACIONID
JOIN USUARIOS U ON R.USUARIOID = U.ID
WHERE U.ID = 3
AND SC.FECHA BETWEEN TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD');
```

Ilustración 2 Img- servicioConsumo

La consulta tiene las siguientes condiciones en el where:

Condición	Razonamiento	Necesidad de índice	Tipo indice
U.ID = 3	Esto filtra los registros de la tabla "USUARIOS" donde el ID del usuario es igual a 3.	Podría ser beneficioso crear un índice en la columna "ID" de la tabla "USUARIOS". Sin embargo, dado que la condición es directa y la columna tiene valores únicos, la ganancia de rendimiento al crear un índice en esta columna es mínima. Ya existe por la PK.	No es necesario.
SC.FECHA BETWEEN TO_DATE('2023-10-01', 'YYYY-MM-DD') AND TO_DATE('2023-10-05', 'YYYY-MM-DD')	Esto filtra los registros en la tabla "SERVICIOSCONSUMO" donde la fecha está en un rango específico.	Sería útil crear un índice en la columna "FECHA" de la tabla "SERVICIOSCONSUMO" ya que se busca un rango específico de fechas. Esto mejoraría el rendimiento al buscar registros que cumplan con esta condición. IDX_SERVICIOSCONSUMO_FECHA	Indice B+ Bitmap

- La condición SC. FECHA BETWEEN TO\_DATE ('2023-10-01', 'YYYY-MM-DD') AND TO\_DATE ('2023-10-05', 'YYYY-MM-DD') filtra los registros de "SERVICIOSCONSUMO" dentro de un rango de fechas específico, lo que significa que esta condición no es altamente selectiva, ya que podría abarcar un número significativo de registros en la tabla.

Para mejorar la selectividad de esta consulta y reducir el número de filas que se deben examinar, la creación de un índice B+ Tree en la columna "FECHA" de la tabla "SERVICIOSCONSUMO" es recomendable. Un índice B+ Tree es eficaz en la aceleración de consultas que buscan dentro de un rango de valores, como fechas, y ayuda a reducir la cantidad de registros que deben ser explorados para satisfacer la condición de rango.

Explicación del Plan x Resultado de la Consulta x				
SQL   Todas las Filas Recuperadas: 18 en 0,097 segundos				
	NOMBRE_USUARIO	NOMBRE_PRODUCTO	COSTO_CONSUMO	FECHA
1	Sofia Rozo	Cerveza	7500	08/10/23
2	Sofia Rozo	Collar de Diamantes	1000000	06/10/23
3	Sofia Rozo	Whisky Escocés	80000	05/10/23
4	Sofia Rozo	Whisky Escocés	80000	11/10/23
5	Sofia Rozo	Whisky Escocés	80000	25/10/23
6	Sofia Rozo	Sushi Variado	60000	03/10/23
7	Sofia Rozo	Filete a la Parrilla	25000	17/10/23
8	Sofia Rozo	Filete a la Parrilla	25000	01/11/23
9	Sofia Rozo	Champú	10000	03/01/23
10	Sofia Rozo	Jersey	2128	20/05/23
11	Sofia Rozo	Paraguas	9520	12/05/23
12	Sofia Rozo	Zapatos	28991	08/02/23
13	Sofia Rozo	Sombrero	37327	28/02/23
14	Sofia Rozo	Calzoncillos	69086	29/04/23

Al crear el índice:

Explicación del Plan x Resultado de la Consulta x				
SQL   Todas las Filas Recuperadas: 18 en 0,066 segundos				
	NOMBRE_USUARIO	NOMBRE_PRODUCTO	COSTO_CONSUMO	FECHA
1	Sofia Rozo	Cerveza	7500	08/10/23
2	Sofia Rozo	Collar de Diamantes	1000000	06/10/23
3	Sofia Rozo	Whisky Escocés	80000	05/10/23
4	Sofia Rozo	Whisky Escocés	80000	11/10/23
5	Sofia Rozo	Whisky Escocés	80000	25/10/23
6	Sofia Rozo	Sushi Variado	60000	03/10/23
7	Sofia Rozo	Filete a la Parrilla	25000	17/10/23
8	Sofia Rozo	Filete a la Parrilla	25000	01/11/23
9	Sofia Rozo	Champú	10000	03/01/23
10	Sofia Rozo	Jersey	2128	20/05/23
11	Sofia Rozo	Paraguas	9520	12/05/23
12	Sofia Rozo	Zapatos	28991	08/02/23
13	Sofia Rozo	Sombrero	37327	28/02/23
14	Sofia Rozo	Calzoncillos	69086	29/04/23
15	Sofia Rozo	Sombrero	17484	04/05/23

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				8 13
HASH JOIN				8 13
Access Predicates SC.IDPRODUCTO=P.ID				
NESTED LOOPS				8 13
NESTED LOOPS				
STATISTICS COLLECTOR				
HASH JOIN				8 9
Access Predicates SC.IDHABITACION=R.HABTACIONID				
NESTED LOOPS				3 5
TABLE ACCESS USUARIOS	USUARIOS	BY INDEX ROWID		1 1
INDEX USUARIOS_PK	USUARIOS_PK	UNIQUE SCAN		1 0
Access Predicates U.ID=3				
TABLE ACCESS RESERVAS	RESERVAS	FULL		3 4
Filter Predicates R.USUARIOID=3				
TABLE ACCESS SERVICIOSCONSUMO	SERVICIOSCONSUMO	FULL		23 4
Filter Predicates AND SC.FECHA>=TO_DATE(' 2023-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss') SC.FECHA<=TO_DATE(' 2023-12-31 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
INDEX PRODUCTOS_PK	PRODUCTOS_PK	UNIQUE SCAN		
Access Predicates SC.IDPRODUCTO=P.ID				
TABLE ACCESS PRODUCTOS	PRODUCTOS	BY INDEX ROWID		1 4
TABLE ACCESS PRODUCTOS	PRODUCTOS	FULL		11 4

## RFC6 - Analizar la operación de hotelandes

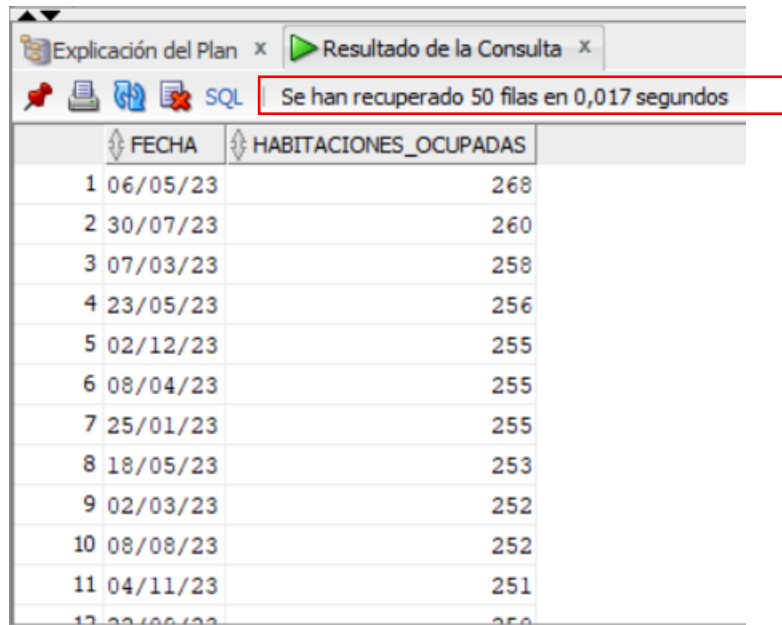
```
--Fechas de Mayor Ocupación:
SELECT FECHA, COUNT(*) AS HABITACIONES_OCUPADAS
FROM RESERVAS
GROUP BY FECHA
ORDER BY HABITACIONES_OCUPADAS DESC;

--Fechas de Mayores Ingresos (Mayor cantidad de consumos realizados):
SELECT SC.FECHA, SUM(SC.COSTO) AS INGRESOS
FROM SERVICIOSCONSUMO SC
GROUP BY SC.FECHA
ORDER BY INGRESOS DESC;

--Fechas de Menor Demanda (Menor ocupación)
SELECT FECHA, COUNT(*) AS HABITACIONES_OCUPADAS
FROM RESERVAS
GROUP BY FECHA
ORDER BY HABITACIONES_OCUPADAS ASC;
```

### *Fechas de Mayor Ocupación:*

Esta consulta involucra una agregación (COUNT) en la tabla RESERVAS y un ordenamiento por la columna COUNT. Si la tabla RESERVAS es grande y esta consulta se ejecuta con frecuencia, podría ser beneficioso tener un índice en la columna FECHA de la tabla RESERVAS.

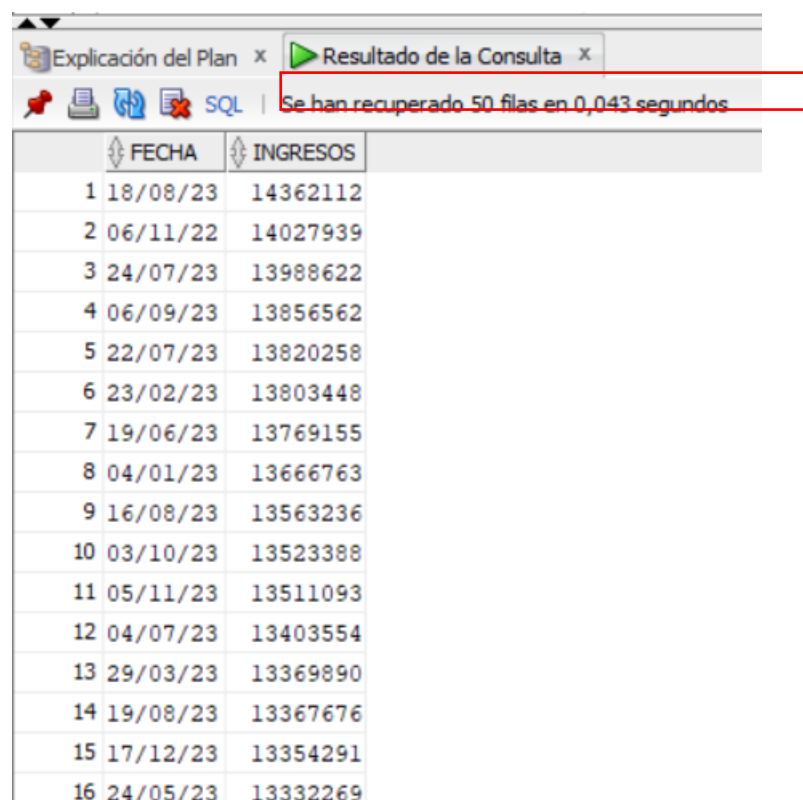


The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'Explicación del Plan' and 'Resultado de la Consulta'. The 'Resultado de la Consulta' tab is active, displaying a table with two columns: 'FECHA' and 'HABITACIONES\_OCUPADAS'. The table contains 17 rows of data, with the first row having an index of 1 and a date of 06/05/23, and the last row having an index of 17 and a date of 22/06/23. A red box highlights the status bar text: 'Se han recuperado 50 filas en 0,017 segundos'.

	FECHA	HABITACIONES_OCUPADAS
1	06/05/23	268
2	30/07/23	260
3	07/03/23	258
4	23/05/23	256
5	02/12/23	255
6	08/04/23	255
7	25/01/23	255
8	18/05/23	253
9	02/03/23	252
10	08/08/23	252
11	04/11/23	251
12	22/06/23	250

### *Fechas de Mayores Ingresos:*

Similar a la primera consulta, esta también implica agregación (SUM) y ordenamiento en la tabla SERVICIOSCONSUMO. Si la tabla SERVICIOSCONSUMO es grande y esta consulta se ejecuta con frecuencia, un índice en la columna FECHA de esa tabla podría ser útil.

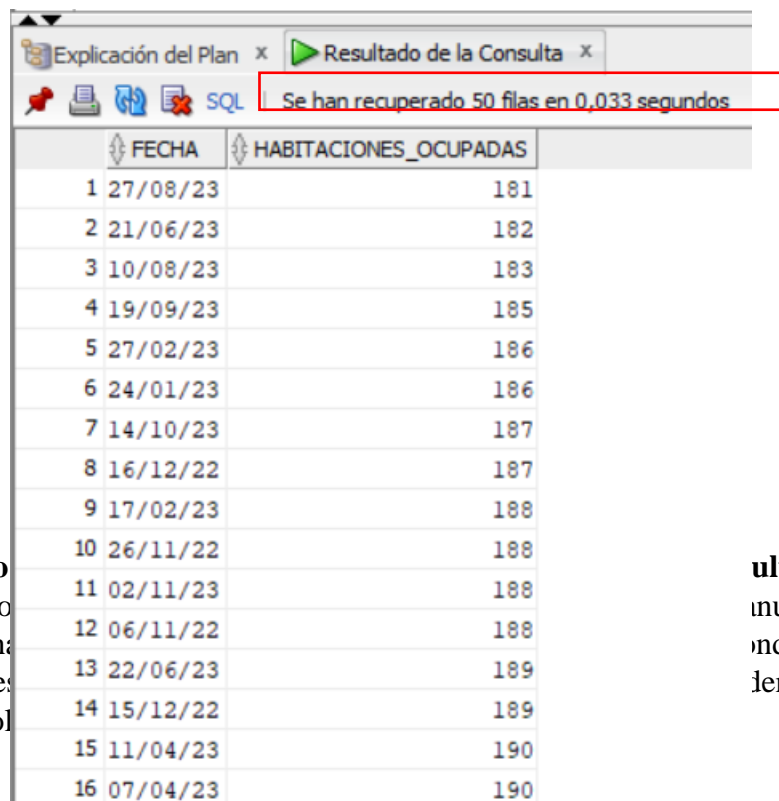


The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'Explicación del Plan' and 'Resultado de la Consulta'. The 'Resultado de la Consulta' tab is active, displaying a table with two columns: 'FECHA' and 'INGRESOS'. The table contains 16 rows of data, with the first row having an index of 1 and a date of 18/08/23, and the last row having an index of 16 and a date of 24/05/23. A red box highlights the status bar text: 'Se han recuperado 50 filas en 0,043 segundos'.

	FECHA	INGRESOS
1	18/08/23	14362112
2	06/11/22	14027939
3	24/07/23	13988622
4	06/09/23	13856562
5	22/07/23	13820258
6	23/02/23	13803448
7	19/06/23	13769155
8	04/01/23	13666763
9	16/08/23	13563236
10	03/10/23	13523388
11	05/11/23	13511093
12	04/07/23	13403554
13	29/03/23	13369890
14	19/08/23	13367676
15	17/12/23	13354291
16	24/05/23	13332269

### *Fechas de Menor Demanda:*

Al igual que las dos consultas anteriores, esta consulta también involucra agregación (COUNT) y ordenamiento en la tabla RESERVAS. Si esta consulta se ejecuta con frecuencia y la tabla RESERVAS es grande, un índice en la columna FECHA podría ayudar a acelerarla.



	FECHA	HABITACIONES_OCUPADAS
1	27/08/23	181
2	21/06/23	182
3	10/08/23	183
4	19/09/23	185
5	27/02/23	186
6	24/01/23	186
7	14/10/23	187
8	16/12/22	187
9	17/02/23	188
10	26/11/22	188
11	02/11/23	188
12	06/11/22	188
13	22/06/23	189
14	15/12/22	189
15	11/04/23	190
16	07/04/23	190

Por lo anterior, no es recomendable que el administrador solo se enfoque en la creación de índices.

Al final, no son las condiciones para crear índices, solo se

### **RFC7 - Encontrar los buenos clientes**

```
--Clientes que han estado en el hotel al menos dos semanas:

SELECT U.NOMBRE AS NOMBRE_CLIENTE, R.USUARIOID, SUM(R.FECHAFINAL - R.FECHA) AS TOTAL DIAS
FROM RESERVAS R
JOIN USUARIOS U ON R.USUARIOID = U.ID
GROUP BY U.NOMBRE, R.USUARIOID
HAVING SUM(R.FECHAFINAL - R.FECHA) >= 14;
```

### *Clientes que han estado en el hotel al menos dos semanas:*

En esta consulta, hay una agregación (SUM) en la columna "FECHAFINAL - FECHA" y un filtro con la cláusula HAVING.



Para mejorar el rendimiento de esta consulta, consideramos un índice en la columna "USUARIOID" de la tabla RESERVAS. Sin embargo, aun con este o sin este la consulta es rápida.

Se han recuperado 50 filas en 0,139 segundos		
NOMBRE_CLIENTE	USUARIOID	TOTAL_DIAS
1 Javier Rodriguez	8	321
2 Tod Sinkin	54	167
3 Violante Peltzer	101	36
4 Shawna Leupold	114	137
5 Massimiliano Savile	169	67
6 Selina Wethered	172	50
7 Aurore Davson	206	147
8 Jany Binstead	213	44
9 Kare Laidel	232	17
10 Layton Tithecott	250	354
11 Karita Marryatt	299	379
12 Dennis Chaise	315	258

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1 9
FILTER				
SUM(R.FECHAFINAL-R.FECHA) >= 14				
HASH				
MERGE JOIN				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	10	3
INDEX	USUARIOS_PK	FULL SCAN	10	1
SORT				
Access Predicates	R.USUARIOID=U.ID			
Filter Predicates	R.USUARIOID=U.ID			
TABLE ACCESS	RESERVAS	FULL	12	4
GROUP BY			1	9
			12	8

*Clientes que han consumido más de \$15,000,000.00 en el último año:*

```
--Clientes que han consumido más de $15,000,000.00 en el último año:

SELECT U.NOMBRE AS NOMBRE_CLIENTE, R.USUARIOID, SUM(SC.COSTO) AS TOTAL_CONSUMO
FROM SERVICIOSCONSUMO SC
JOIN RESERVAS R ON SC.IDHABITACION = R.HABTACIONID
JOIN USUARIOS U ON R.USUARIOID = U.ID
WHERE SC.FECHA >= SYSDATE - 365 -- Para el último año
GROUP BY U.NOMBRE, R.USUARIOID
HAVING SUM(SC.COSTO) > 15000000;
```

Indice	Necesidad de índice	Tipo indice
"IDHABITACION" de la tabla SERVICIOSCONSUMO	Es necesario un índice en esta columna porque las consultas que filtran o agrupan por "IDHABITACION" son comunes y si la tabla SERVICIOSCONSUMO es grande.	En este caso, un índice normal (b-tree) es apropiado, ya que las consultas suelen implicar una igualdad en la columna "IDHABITACION".
FECHA" de la tabla SERVICIOSCONSUMO	Si las consultas filtran o agrupan por la columna "FECHA" con frecuencia y la tabla SERVICIOSCONSUMO es grande, es probable que necesites un índice en esta columna.	Dado que se está filtrando por una columna de fecha, un índice basado en rango (b-tree) es adecuado para esta columna.

### RFC8 – Encontrar los servicios que no tienen mucha demanda

```
SELECT s.TIPO AS SERVICIO, COUNT(*) AS FRECUENCIA_TOTAL
FROM SERVICIORESERVAS s
WHERE FECHAINICIAL >= ADD_MONTHS(SYSDATE, -12) -- Filtrar reservas del último año
GROUP BY s.TIPO
HAVING COUNT(*) < 3;
```

En la consulta SQL, estamos filtrando y agrupando por el campo `TIPO` en la tabla `SERVICIORESERVAS`. Esta tabla es mediana pero la consulta no es tan frecuente. Aun así, un índice en el campo `TIPO` podría mejorar el rendimiento de la consulta.

Además, hay una condición en el campo `FECHAINICIAL` y como varias consultas lo utilizan un índice allí sería apropiado, este fue creado en una consulta anterior.

Considerando lo anterior, decidimos no implementar índices diferentes porque no hay las suficientes condiciones cumpliéndose para utilizarlos.

SERVICIO	FRECUENCIA_TOTAL
1 salon	2
2 LAVANDERIA	2



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				5
HASH				5
Filter Predicates				
COUNT(*) < 3				
TABLE ACCESS	SERVICIORESERVAS	FULL	17	4
Filter Predicates				
FECHAINICIAL >= ADD_MONTHS(SYSDATE@!, (-12))				

## RFC9 - Consultar consumo en hotelandes

### Resultado:

ID	NOMBRE	DOCUMENTO	TIPODOCUMENTO	CORREO	DESCRIPCION	COSTO	FECHA	IDHABITACION	IDPRODUCTO
1	49519 Bethina Bruun	4508039527646860	PASAPORTE	bbruunee@yahoo.co.jp	Piscina	93868	19/04/23	2237	31795
2	32803 Sean Verbruggen	630454739945310025	CE	sverbruggema@trellian.com	Uso jacuzzi 1 hora	37232	19/04/23	33290	10974
3	28064 Sabine Grunwall	3535872466059961	CE	sgrunwall1r@arstechnica.com	Cerveza en el Bar	27664	19/04/23	33964	125974
4	42151 Sheeree Frankiewicz	3557532799223006	PASAPORTE	sfrankiewicz46@adobe.com	Whisky en el Bar	84056	19/04/23	842	79956
5	27406 Phillis Steanyng	5610682100747837	CC	psteanyngb9@clickbank.net	Whisky en el Bar	24614	19/04/23	31343	137445
6	40217 Suki Leates	491171178777637149	PASAPORTE	sleates60@lala.or.jp	Cerveza en el Bar	44610	19/04/23	9600	4986
7	47565 Carzy Hermon	3544583050589359	CC	ckermof@vimeo.com	Piscina	73087	19/04/23	38567	39999
8	26076 Sibby Balleine	3528760110932163	DNI	sballeine23@craigslist.org	Botella de Agua del Supermercado	94215	19/04/23	48510	67067
9	39941 Gusta Estevez	5360573339868571	CE	gestevezq4@arstechnica.com	Cerveza en el Bar	75540	19/04/23	26085	92745
10	46012 Clarke Monnover	4844671918081787	CC	cmonnoverb@salon.com	Chórol de la Casa en el Bar	3962	19/04/23	40382	57596

### Índices:

TABLE ACCESS	USUARIOS	BY INDEX ROWID	6	3	369	18347
Filter Predicates						
U.ROL='Cliente'						
INDEX	USUARIOS_PK	FULL SCAN	10	1	48	4649
SORT		JOIN	4	5	223	25638
Access Predicates						
U.ID=H.USUARIOID						
Filter Predicates						
U.ID=H.USUARIOID						
TABLE ACCESS	HABITACIONES	FULL	4	4	223	7371
Filter Predicates						
H.USUARIOID IS NOT NULL						
INDEX	IDX_SERVICIOSCONSUMO_FECHA	RANGE SCAN	23	1	0	0
Access Predicates						
S.FECHA>=TO_DATE(:FECHAINICIAL, 'DD-MM-YYYY')						
S.FECHA<=TO_DATE(:FECHAFINAL, 'DD-MM-YYYY')						
TABLE ACCESS	SERVICIOSCONSUMO	BY INDEX ROWID	2	3	0	0
Filter Predicates						
H.ID=S.IDHABITACION						
TABLE ACCESS	SERVICIOSCONSUMO	BY INDEX ROWID BATCHED	23	3	108	827
INDEX	IDX_SERVICIOSCONSUMO_FECHA	RANGE SCAN	23	1	2	66

Por lo tanto, se crearon índice simple tipo bitmap para el rol de usuario con el fin de filtrar los clientes, otro índice simple de bitmap para las fechas de servicio consumo y otro simple bitmap para el idusuario en habitaciones, con el fin de mejorar la consulta, puesto que los otros índices creados por oracle eran inefficientes.

## RFC10 - Consultar consumo en hotelandes V2

```
SELECT U.ID, U.NOMBRE, U.DOCUMENTO, U.TIPODOCUMENTO, U.CORREO
FROM USUARIOS U
JOIN HABITACIONES H ON U.ID = H.USUARIOID
LEFT JOIN SERVICIOSCONSUMO S ON H.ID = S.IDHABITACION
AND S.FECHA BETWEEN TO_DATE(:fechainicial, 'DD-MM-YYYY') AND TO_DATE(:fechafinal, 'DD-MM-YYYY')
WHERE U.ROL = 'Cliente'
HAVING COUNT((U.ID)) = 1
GROUP BY U.ID, U.NOMBRE, U.DOCUMENTO, U.TIPODOCUMENTO, U.CORREO;
```

### Resultado:

Salida de Script x Rastreo Automático x Resultado de la Consulta x					
SQL   Se han recuperado 1,600 filas en 0.552 segundos					
ID	NOMBRE	DOCUMENTO	TIPODOCUMENTO	CORREO	
1	73 Rutter Machon	5602248251637753	PASAPORTE	rmachon20@theforest.net	
2	2538 Vincenz Philippault	30577202117884	CE	vphilippaultex@ft.com	
3	26806 Liz Downie	3535117309562239	PASAPORTE	ldowniend@unesco.org	
4	48080 Coral Billing	3589014128228594	PASAPORTE	cbilling27@sourceforge.net	
5	39529 Sapphire Fesby	3553691972828504	CC	sfesbyeo@twitpic.com	
6	46617 Yolanne Dunne	3581011626015398	CC	ydunneh4@elegantthemes.com	
7	36929 Nananne Shalliker	3574797080886618	PASAPORTE	nshallikerps@joomla.org	
8	36357 Shell Sansom	5020589656767338727	CE	ssansom9w@geocities.jp	
9	32335 Torrey Sigg	670949219939416269	CE	tsigg9a@myspace.com	
10	29516 Norina Liccardo	6706471386847369	DNI	nliccardoeb@tuttocitta.it	
11	18547 Aleksandr Strase	3554486593740603	PASAPORTE	astrasef6@hibu.com	
12	18467 Glenda Cutridge	67631496776618585	PASAPORTE	gcutridgcy@bloglines.com	
13	33890 Bernard Coupar	5557348286963565	CC	bcouparop@ning.com	
14	1706 Theresita Trathan	30082903063305	DNI	ttrathanjl@histats.com	
15	14778 Arvin Cogge	67633993334509635	CE	acogge111@umn.edu	
16	18075 Ware Arsmith	3576610472351067	PASAPORTE	warsmith22@ustream.tv	
17	33917 Bobbee Bozward	374288845969731	DNI	bbozwardpg@haol23.com	
18	10999 Kasper Turfes	3571898859760670	CE	kturfesrq@mayoclinic.com	
19	29180 Bartram Nesfield	3536235209947463	CE	bnesfield4z@flickr.com	
20	22112 Herschel Brazier	201838678344367	CC	hbrazier33@hud.gov	

índices:



La misma lógica se realiza en este caso para ambos requerimientos, por lo tanto, Oracle por defecto utiliza los mismos índices que habíamos aplicado anteriormente lo cual es beneficioso y utiliza el índice de la pk de usuarios.

## RFC11 - Consultar funcionamiento

```

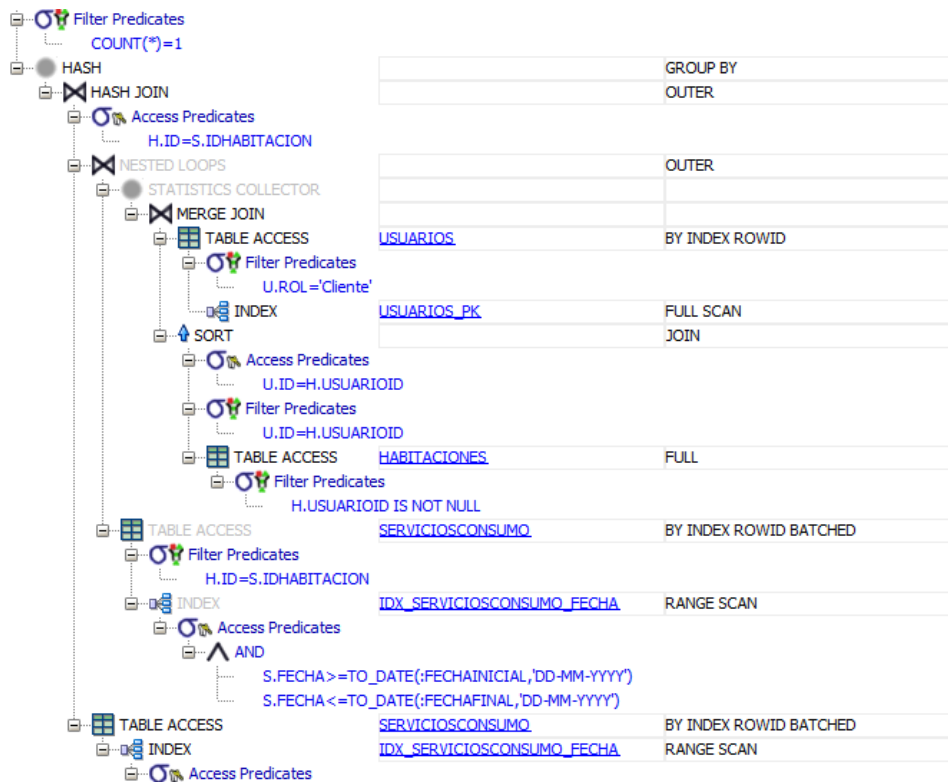
31 WITH WeekNumbers AS (
32     SELECT LEVEL AS WeekNumber
33     FROM DUAL
34     CONNECT BY LEVEL <= 52), WeekDates AS (
35     SELECT
36         TO_DATE('01-01-2023', 'DD-MM-YYYY') + (WeekNumber - 1) * 7 AS StartDate,
37         TO_DATE('01-01-2023', 'DD-MM-YYYY') + (WeekNumber - 1) * 7 + 6 AS EndDate
38     FROM WeekNumbers)
39 SELECT
40     TO_CHAR(StartDate, 'DD-MM-YYYY') AS Inicio,
41     TO_CHAR(EndDate, 'DD-MM-YYYY') AS Fin,
42     (SELECT Descripcion FROM (
43         SELECT s.Descripcion, COUNT(*) AS ConsumptionCount
44         FROM SERVICIOSCONSUMO s
45         WHERE s.FECHA BETWEEN StartDate AND EndDate
46         GROUP BY s.Descripcion
47         ORDER BY COUNT(*) DESC
48     ) WHERE ROWNUM = 1) AS servicioMasConsumido,
49     (SELECT Descripcion FROM (
50         SELECT s.Descripcion, COUNT(*) AS ConsumptionCount
51         FROM SERVICIOSCONSUMO s
52         WHERE s.FECHA BETWEEN StartDate AND EndDate
53         GROUP BY s.Descripcion
54         ORDER BY COUNT(*) ASC
55     ) WHERE ROWNUM = 1) AS servicioMenosConsumido,
56     (SELECT HabitaciónID FROM (
57         SELECT r.HabitacionID, COUNT(*) AS ReservationCount
58         FROM RESERVAS r
59         WHERE r.FECHA BETWEEN StartDate AND EndDate
60         GROUP BY r.HabitacionID
61         ORDER BY COUNT(*) DESC
62     ) WHERE ROWNUM = 1) AS habitacionMasSolicitada,
63     (SELECT HabitaciónID FROM (
64         SELECT r.HabitacionID, COUNT(*) AS ReservationCount
65         FROM RESERVAS r
66         WHERE r.FECHA BETWEEN StartDate AND EndDate
67         GROUP BY r.HabitacionID
68         ORDER BY COUNT(*) ASC
69     ) WHERE ROWNUM = 1) AS habitacionMenosSolicitada
70 FROM WeekDates;

```

## Resultado:

Salida de Script x Rastreo Automático x Resultado de la Consulta x					
Todas las Filas Recuperadas: 52 en 0.346 segundos					
	INICIO	FIN	SERVICIOASMENOSCONSUMIDO	SERVICIOMENOSCONSUMIDO	HABITACIONMÁS SOLICITADA HABITACIONMENOS SOLICITADA
1	01-01-2023	07-01-2023	Pasta Carbonara en el Restaurante	Compra de producto shampoo	40121 27
2	08-01-2023	14-01-2023	Spa	Servicio domicilio a puerta	1034 13
3	15-01-2023	21-01-2023	Collar de Diamantes de la Tienda	Compra producto shampoo	44936 34
4	22-01-2023	28-01-2023	Spa	Uso jacuzzi 1 hora	627 3
5	29-01-2023	04-02-2023	Filete a la Parrilla en el Restaurante	Cóctel de la Casa en el Bar	5007 22
6	05-02-2023	11-02-2023	Collar de Diamantes de la Tienda	Filete a la Parrilla en el Restaurante	1105 9
7	12-02-2023	18-02-2023	Cóctel de la Casa en el Bar	1 hora de sauna	411 31
8	19-02-2023	25-02-2023	Lavandería	Filete a la Parrilla en el Restaurante	248 18
9	26-02-2023	04-03-2023	Filete a la Parrilla en el Restaurante	Whisky en el Bar	3120 54
10	05-03-2023	11-03-2023	Whisky en el Bar	1 hora de sauna	4013 62
11	12-03-2023	18-03-2023	Sushi en el Restaurante	Botella de Agua del Supermercado	3097 32
12	19-03-2023	25-03-2023	Collar de Diamantes de la Tienda	Uso jacuzzi 1 hora	21025 52
13	26-03-2023	01-04-2023	Pasta Carbonara en el Restaurante	1 hora de hidromasaje	13102 91
14	02-04-2023	08-04-2023	Compra producto shampoo	1 hora de sauna	11968 32
15	09-04-2023	15-04-2023	Whisky en el Bar	Cerveza en el Bar	26160 14
16	16-04-2023	22-04-2023	Filete a la Parrilla en el Restaurante	Cóctel de la Casa en el Bar	1047 48
17	23-04-2023	29-04-2023	Uso jacuzzi 1 hora	Whisky en el Bar	476 33
18	30-04-2023	06-05-2023	Servicio domicilio a puerta	Pasta Carbonara en el Restaurante	190 22
19	07-05-2023	13-05-2023	Cerveza en el Bar	Piscina	23764 16
20	14-05-2023	20-05-2023	Cóctel de la Casa en el Bar	1 hora de hidromasaje	23270 7
21	21-05-2023	27-05-2023	Piscina	Cóctel de la Casa en el Bar	1964 30
22	28-05-2023	03-06-2023	Cóctel de la Casa en el Bar	Compra producto shampoo	18340 29

## Índices:



Oracle originalmente no usaba indices mas alla de los ya creados para requerimientos anteriores y indice por primary key en usuarios. Sin embargo, agregamos un indice simple de bitmap en el atributo fecha de reservas, esto con el fin de acelerar la consulta puesto que la fecha es el atributo que usamos en todos los where de la consulta. Inicialmente la consulta duraba 1,5 segundos en total y ahora solo 0,3.

## RFC12 - Consultar los clientes excelentes

```
SELECT
  U.ID AS ID,
  U.NOMBRE AS NOMBRE,
  U.CORREO AS CORREO,
  U.DOCUMENTO AS DOCUMENTO,
  CASE
    WHEN EXISTS (
      SELECT 1
      FROM RESERVAS R
      WHERE R.USUARIOID = U.ID
      AND R.FECHA BETWEEN TRUNC(SYSDATE, 'Q') - INTERVAL '3' MONTH AND SYSDATE
    ) THEN 'Cliente de Estancias Trimestrales'
    WHEN EXISTS (
      SELECT 1
      FROM SERVICIOSCONSUMO SC
      JOIN HABITACIONES ON HABITACIONES.USUARIOID = U.ID
      WHERE HABITACIONES.USUARIOID = U.ID
      AND SC.COSTO > 300000
    ) THEN 'Cliente de Servicio Costoso'
  END AS TipoClienteExcelente
FROM USUARIOS U;
```

**Resultado:**

Se han recuperado 50 filas en 0.867 segundos				
ID	NOMBRE	CORREO	DOCUMENTO	TIPOCLIENTEEXCELENTE
1	1627 Morlee Reany	mreanyhe@printfriendly.com	5602238982361328	(null)
2	1468 Terra Sybe	tsybecs@hexun.com	6377866345116608	Cliente de Estancias Trimestrales
3	1469 Janka Hitter	jhitterd0@census.gov	3552465783841399	Cliente de Estancias Trimestrales
4	1470 Geoffrey Agge	gagged1@pbs.org	3536696872567153	Cliente de Servicio Costoso
5	1471 Randy Vassall	rvassalld2@stanford.edu	490365643474040275	Cliente de Estancias Trimestrales
6	1472 Hayden Sauvan	hsauvand3@upenn.edu	3543157797228687	Cliente de Servicio Costoso
7	1473 Cornell Colebeck	ccolebeck4@cnn.com	3537646302059460	Cliente de Servicio Costoso
8	1474 Ebebe Rippen	prippend5@twitpic.com	3544665909376177	Cliente de Servicio Costoso
9	1475 Myron Vergo	mvergod6@bloglines.com	3560914950192805	Cliente de Servicio Costoso
10	1476 Gusty Sturm	gsturmd7@cbsnews.com	4041590481552	Cliente de Servicio Costoso
11	1477 Lorin Niccols	lniccols8@usatoday.com	3579711345787474	Cliente de Servicio Costoso
12	1478 Suzie Kertess	skertessd9@unicef.org	6762141627712482032	Cliente de Estancias Trimestrales
13	1479 Cybil Freathy	cfreathyda@cbslocal.com	3533075799115953	(null)
14	1480 Eben Deverille	edeverilledb@stumbleupon.com	3564599888086959	Cliente de Servicio Costoso
15	1481 Maddalena Bernette	mbernettedc@twitpic.com	3561246172407151	Cliente de Servicio Costoso
16	1482 Dario Ivermee	divermeedd@de.vu	3589662145230421	Cliente de Servicio Costoso
17	1483 Zedekiah Buffy	zbuffyde@un.org	5048374300631423	Cliente de Estancias Trimestrales
18	1484 Fransisco Wilbraham	fwilbrahamdf@arizona.edu	3539811998543163	Cliente de Servicio Costoso
19	1485 Jarvis Heintsch	jheintschdg@loc.gov	3567566650718272	(null)
20	1486 Jerrold Bennedsen	jbennedsendh@odnoklassniki.ru	30119794419376	Cliente de Estancias Trimestrales
21	1487 Son Claydon	sclaydondi@netscape.com	5602225229449537	Cliente de Servicio Costoso
22	1488 Sarette Bavridge	sbavridgedj@jugem.jp	5602222125365966	(null)
23	1489 Laryssa Oven	lovendk@shared.com	5602211093194870	Cliente de Estancias Trimestrales
24	1490 Jewel Pickett	jpickettdl@telegraph.co.uk	4041370327850	Cliente de Servicio Costoso
25	1491 Amye Zukierman	azukiermandm@yellowpages.com	3532239760694286	Cliente de Estancias Trimestrales
26	1492 Cyrille Hazard	chazarddn@cocolog-nifty.com	30224159616499	(null)
27	1493 Sharia Glawsop	sglawsopdo@auda.org.au	5602229609189531	(null)
28	1494 Daron Petri	dpetridp@dedecms.com	4917927353282229	(null)

## Indices:



La consulta original de Oracle no hacia uso de índices más allá de fecha en reservas que ya fue creado previamente en otro requerimiento y los índices en las primary keys, por lo tanto, la consulta demoraba 1,9 segundos. Sin embargo, se implementaron 2 índices tipo bitmap en el atributo usuarioid de reservas y costo en servicios consumo, con lo cual la consulta baja a 0,816 segundos.