



# Sistemas Transaccionales

Documento de informe

Grupo C1

Santiago Jaimes

Juan Jose Cediel

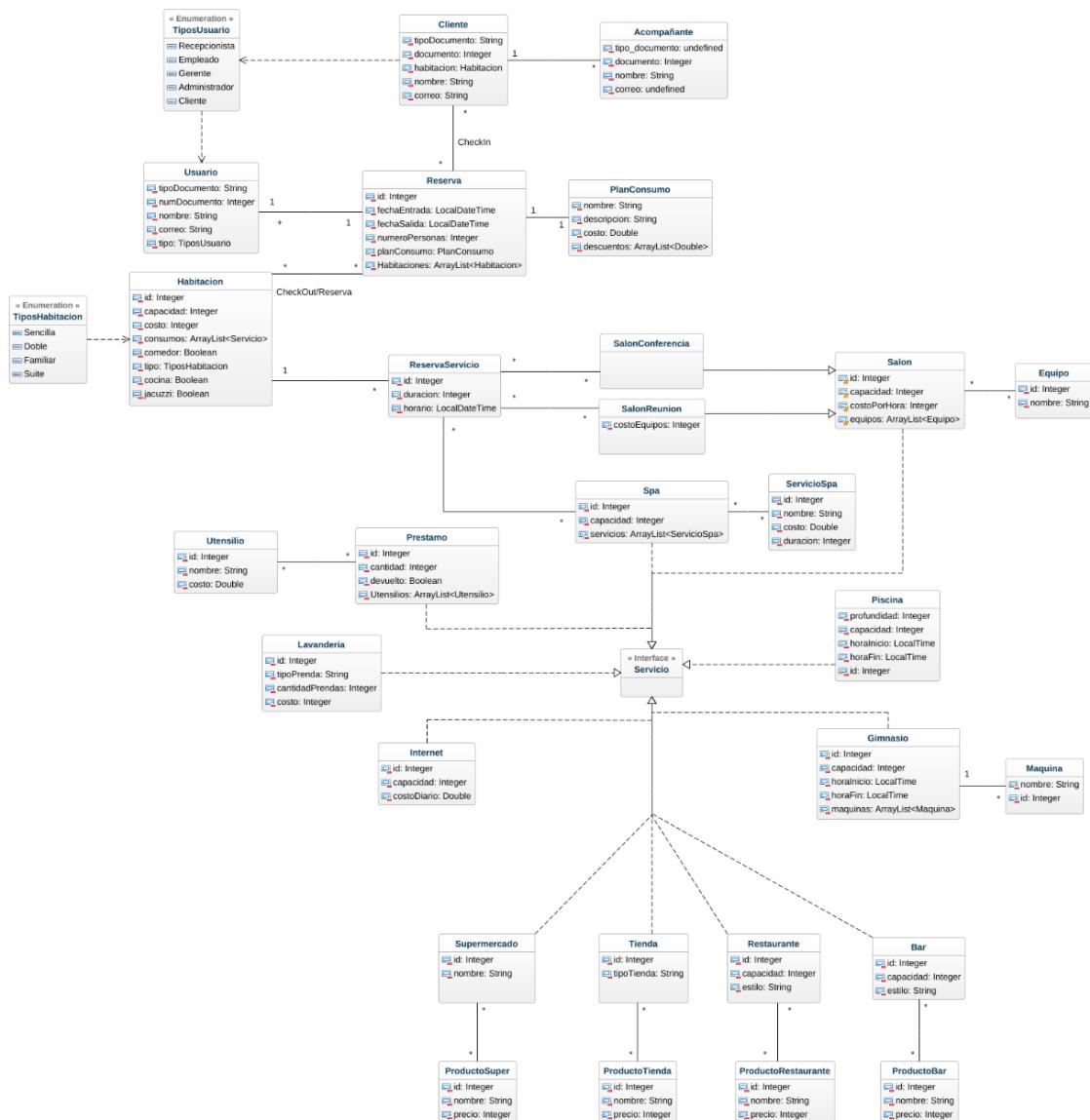
Camilo Mercado

Universidad de Los Andes  
Bogotá – Colombia

7 de Noviembre de 2023

## 1. Análisis

A partir del diseño existente de la anterior entrega se puede decir que el impacto que representa la introducción de los nuevos requerimientos a nivel del modelo UML conceptual propuesto en la entrega anterior implica un cambio completo en el diseño para poder soportar las funcionalidades de la entrega pasada corregidas y las nuevas funcionalidades de consulta, por lo tanto, el modelo conceptual propuesto en esta entrega es el siguiente:



En este modelo se considera la reconfiguración del proyecto a un modelo donde las relaciones UML conectan correctamente las clases para poder hacer consultas avanzadas por medio de clases principales como lo son Usuario, Reserva, Servicio y Habitación, que, con las cardinalidades apropiadas en el modelo E/R generado se logra apreciar las tablas que conectan estas clases, permitiendo consultar consumos y reservas, el modelo Entidad/Relación (E/R) y el modelo relacional están descritos en los archivos modelorelacional.pdf y

modelo.dmd, ubicados en la carpeta 'modelo entidad relación'. No se incluyen en este documento actual por razones de legibilidad. Se sugiere abrir el archivo de Excel que contiene el modelo relacional en lugar del archivo PDF para una mejor visualización de las tablas.

A continuación, se dará una breve descripción del modelo relacional:

Nombre	Campos	Restricciones
bares	id (entero), capacidad (entero), estilo (cadena), tiposervicio_tipo (cadena)	PK: id
checkin	reservas_id (entero), informacionclientes_num_documento (entero), informacionclientes_tipo_documento (entero), fecha_ingreso (fecha)	PK: reservas_id, informacionclientes_num_documento, informacionclientes_tipo_documento
checkouts	reservas_id (entero), habitaciones_id (entero), fecha_salida (fecha)	PK: reservas_id, habitaciones_id
consumos	habitaciones_id (entero), tiposervicio_tipo (cadena), descripcion (cadena), costo (número)	PK: habitaciones_id, tiposervicio_tipo
equipos	id (entero), nombre (cadena)	PK: id
gimnasios	id (entero), capacidad (entero), hora_inicio (hora), hora_fin (hora), tiposervicio_tipo (cadena)	PK: id
habitaciones	id (entero), capacidad (entero), costo (número), tiposhabitacion_tipo (cadena)	PK: id
informacionclientes	tipo_documento (entero), num_documento (entero), nombre (cadena), correo (cadena)	PK: tipo_documento, num_documento
internets	id (entero), capacidad (entero), costo_dia (número), tiposervicio_tipo (cadena)	PK: id
maquinas	id_maquina (entero), nombre (cadena), gimnasios_id (entero)	PK: id_maquina
ofrece	spas_id (entero), serviciosspa_id (entero)	PK: spas_id, serviciosspa_id
piscinas	id (entero), profundidad_m (entero), capacidad (entero), hora_inicio (hora), hora_fin (hora), tiposervicio_tipo (cadena)	PK: id
planesconsumo	id (entero), nombre (cadena), estadia_min (entero), costo (número), desc_reserva (número), desc_bar (número), desc_restaurante (número), desc_servicio (número)	PK: id
presta	serviciosprestamo_id (entero), utensilios_id (entero)	PK: serviciosprestamo_id, utensilios_id
prestanaditional	salonesreunion_id (entero), equipos_id (entero), costo (número)	PK: salonesreunion_id, equipos_id
prestangratis	salonesconferencia_id (entero), equipos_id (entero)	PK: salonesconferencia_id, equipos_id
productosbar	id (entero), nombre (cadena), precio (número)	PK: id
productosrestaurante	id (entero), nombre (cadena), precio (número)	PK: id
productossuper	id (entero), nombre (cadena), precio (número)	PK: id

Nombre	Campos	Restricciones
productostienda	id (entero), nombre (cadena), precio (número)	PK: id
reservan	habitaciones_id (entero), reservas_id (entero) id (entero), fecha_inicio (fecha), fecha_salida (fecha), num_personas (entero), planesconsumo_id (entero), habitaciones_id (entero), tiposhabitacion_tipo (cadena), informacionclientes_num_documento (entero), informacionclientes_tipo_documento (entero)	PK: habitaciones_id, reservas_id
reservas	(entero)	PK: id
salonesconferencia	id (entero), capacidad (entero), hora_inicio (hora), hora_fin (hora), tiposervicio_tipo (cadena)	PK: id
salonesreunion	id (entero), capacidad (entero), hora_inicio (hora), hora_fin (hora), tiposervicio_tipo (cadena)	PK: id
servicios	tipo (cadena), descripcion (cadena), costo (número), tiposervicio_tipo (cadena)	PK: tipo, tiposervicio_tipo
serviciosprestamo	id (entero), nombre (cadena)	PK: id
serviciosspaa	id (entero), nombre (cadena), capacidad (entero), hora_inicio (hora), hora_fin (hora), tiposervicio_tipo (cadena)	PK: id
tiposervicio	tipo (cadena)	PK: tipo
tiposhabitacion	tipo (cadena)	PK: tipo
tiposerviciosspa	tipo (cadena)	PK: tipo
tiposerviciosprestamo	tipo (cadena)	PK: tipo

Cada tabla tiene una serie de campos que almacenan diferentes tipos de información. También hay restricciones que aseguran la integridad de los datos, como las claves primarias y las relaciones entre tablas. Estas relaciones permiten vincular información de diferentes tablas a través de claves que se comparten entre ellas.

Todas las clases son actualizadas respecto a la anterior entrega tanto en diseño de atributos como de relaciones, y con este modelo se ejecutaron todos los requerimientos implementados y las consultas correspondientes.

## 2. Diseño de la aplicación

### Entrega anterior

La aplicación completa de la entrega anterior tenía un enfoque diferente, pues este se enfocaba en la creación de los CRUD de tuplas en las relaciones, por lo tanto, las instrucciones SQL usadas fueron de tipo DML de creación, actualizar y eliminar, por esto en este enfoque lo importante es la funcionalidad, mas no la velocidad de la operación, pues, en estas únicamente modifica la relación añadiendo o eliminando una tupla, sin necesidad de filtrar los resultados ni dar información específica, además, se sabe que no es necesario crear un índice ya que se traen todos los datos al consultar y mostrar la información, por esto no serían eficiente, la selectividad es muy baja para necesitar un índice.

### Nueva entrega

La aplicación completa de la nueva entrega se implementa sobre el nuevo modelo conceptual UML propuesto en el literal 1 en donde las clases y los atributos son los necesarios y suficientes para poder hacer los requerimientos funcionales descritos para la entrega, por lo tanto este modelo incluye las relaciones correctas, y modificaciones en la base de datos como la creación de índices para la optimización de las consultas, teniendo en cuenta los requerimientos no funcionales como la latencia de la aplicación en donde las consultas con la base de datos poblada debe tener tiempos menores a 0,8s, para esto, en el siguiente literal se genera un análisis completo acerca de las decisiones de creación de índices para el cumplimiento de los requerimientos descritos en el enunciado:

#### Diseño de índices

Requerimiento funcional	¿Es necesario crear idx?	Justificación	Tipo de índice usado
RFC1	Sí	Utiliza columnas frecuentemente referenciadas en el WHERE (tiposervicio_tipo), usa columnas frecuentemente usadas para hacer el JOIN para estos se usa índices a diferencia del resto de atributos de las relaciones que si bien o tienen una operación algebraica junto a una cláusula WHERE. Por otra parte, la selectividad supera el 75% al tener valores que se repiten en las diferentes consultas sobre la cantidad de datos existentes.	B+ para todos
RFC2	No	Los índices necesarios se crean para el requerimiento anterior por las mismas razones, los otros atributos no son elegibles ya que son consultas de rango, con condiciones y con operaciones como COUNT	---
RFC3	No	Los índices necesarios se crean para requerimientos anteriores por las mismas razones, los otros atributos no son elegibles ya que participan de operaciones	...

		con operadores aritméticos u operaciones como COUNT con un WHERE	
RFC4	No	Los índices necesarios se crean para requerimientos anteriores por las mismas razones, los otros atributos no son elegibles ya que se descartan por las razones previas	---
RFC5	No	El índice necesario ya está creado que corresponde a tiposervicio_tipo, el resto de los atributos de la consulta no son elegibles ya que participan de operaciones de en donde hacen parte de rangos y en consultas anteriores con operadores. No se crea un índice para consumos.descripcion ya que son únicos para cada consumos.tiposervicio_tipo con el cual ya está vinculado y la operación de filtrado se hace por este criterio	---
RFC6	Sí	Se crea un índice para fecha, ya que tiene selectividad alta > 75%, no se usan operadores aritméticos ni funciones con cláusulas WHERE,	B+
RFC7	No	No se necesitan índices ya que se selecciona toda la información, no se filtra en la consulta principal, en la subconsulta se filtra por la llave primaria de la relación, por lo que ya se tiene el índice creado por defecto en por el SMBD y no es necesario crearlo	---
RFC8	No	Ya están creados los índices necesarios candidatos para la consulta, los demás atributos no son candidatos ya que se selecciona el conteo de todo con condiciones where, también se hacen operaciones aritméticas con estos	---
RFC9	Sí	Ya están creados los índices necesarios candidatos para la consulta, los otros atributos corresponden a llaves primarias, por lo tanto, ya tiene creado un índice por el SMBD, los otros atributos de atributos no son elegibles ya que hacen parte de consultas con operadores aritméticos	---
RFC10	No	La consulta se hace sobre un todo de una subconsulta que filtra por medio atributos de llaves primarias, al estas ser parte de llaves primarias ya tienen un índice por defecto del SMBD, los demás atributos ya habían sido descartados como candidatos y tener un índice.	---

RFC11	No	Todos los índices necesarios ya están creados en consultas anteriores, los demás atributos que participan de las consultas tienen funciones con cláusulas WHERE, por lo que no se debe crear un índice para estos, los otros atributos son creados en ejecución, por lo que no se puede crear un índice sobre estos	---
RFC12	Sí	Se crean índices para r.usuarios_nombre, r.usuarios_tipo_documento, r.usuarios_num_documento, r.usuarios_correo, todos en conjunto, ya que estos tienen nivel de selectividad mayor al 75% y las consultas agrupan continuamente por este atributo, el organizarlo favorece el tiempo de agrupamiento	B+

En resumen, se crean índices para:

consumos.tiposervicio\_tipo, reservan.habitaciones\_id, consumos.habitaciones\_id, fecha, (reservas.usuarios\_nombre, reservas.usuarios\_tipo\_documento, reservas.usuarios\_num\_documento, reservas.usuarios\_correo)

Todos de tipo árboles B+ ya que no es necesario mapear por bits, pues los valores no son encontrados por el valor propio de los bits de los elementos sino por la agrupación de estos por similares en ciertos atributos.

#### Índices automáticos

INDEX_NAME	STATUS	COLUMNS	COMMENTS	UNIQUENESS	DISTINCT_KEYS	NUM_ROWS
1 HABITACIONES_PK	VALID	1 (null)	UNIQUE		9	9
2 USUARIOS_PK	VALID	4 (null)	UNIQUE		12	12
3 TIPOSUSUARIO_PK	VALID	1 (null)	UNIQUE		5	5
4 SERVICIOSSPA_PK	VALID	1 (null)	UNIQUE		2	2
5 SERVICIOSLAVANDERIA_PK	VALID	1 (null)	UNIQUE		4	4
6 INFORMACIONCLIENTES_PK	VALID	2 (null)	UNIQUE		2	2
7 CHECKOUTS_PK	VALID	2 (null)	UNIQUE		1	1
8 PRODUCTOSBAR_PK	VALID	1 (null)	UNIQUE		4	4
9 PISCINAS_PK	VALID	1 (null)	UNIQUE		2	2
10 CHECKIN_PK	VALID	3 (null)	UNIQUE		2	2
11 TIENDAS_PK	VALID	1 (null)	UNIQUE		1	1
12 MAQUINAS_PK	VALID	1 (null)	UNIQUE		4	4
13 SALONESCONFERENCIA_PK	VALID	1 (null)	UNIQUE		3	3
14 BARES_PK	VALID	1 (null)	UNIQUE		2	2
15 UTENSILIOS_PK	VALID	1 (null)	UNIQUE		3	3
16 EQUIPOS_PK	VALID	1 (null)	UNIQUE		3	3
17 SERVICIOSPRESTAMO_PK	VALID	1 (null)	UNIQUE		2	2
18 RESERVASSERVICIO_PK	VALID	1 (null)	UNIQUE		1	1
19 TIPOSERVICIO_PK	VALID	1 (null)	UNIQUE		11	11
20 SUPERMERCADOS_PK	VALID	1 (null)	UNIQUE		2	2
21 SPAS_PK	VALID	1 (null)	UNIQUE		1	1
22 CONSUMOS_PK	VALID	2 (null)	UNIQUE		1	1
23 RESERVASSPA_PK	VALID	2 (null)	UNIQUE		1	1
24 VENDENSUPER_PK	VALID	2 (null)	UNIQUE		5	5
25 OFRECE_PK	VALID	2 (null)	UNIQUE		2	2
26 SIRVEN_BAR_PK	VALID	2 (null)	UNIQUE		6	6
27 INTERNETS_PK	VALID	1 (null)	UNIQUE		1	1
28 PRODUCTOSRESTAURANTE_PK	VALID	1 (null)	UNIQUE		4	4
29 SALONESREUNION_PK	VALID	1 (null)	UNIQUE		5	5
30 RESTAURANTES_PK	VALID	1 (null)	UNIQUE		2	2
31 TIPOSOSHABITACION_PK	VALID	1 (null)	UNIQUE		3	3
32 PRESTANGRATIS_PK	VALID	2 (null)	UNIQUE		3	3
33 RESERVAS_PK	VALID	1 (null)	UNIQUE		1	1

34 PRODUCTOSTIENDA_PK	VALID	1 (null)	UNIQUE	3	3
35 VENDENTIENDA_PK	VALID	2 (null)	UNIQUE	3	3
36 RESERVAN_PK	VALID	2 (null)	UNIQUE	1	1
37 RESERVASSALONCONFERENCIA_PK	VALID	2 (null)	UNIQUE	0	0
38 RESERVASSALONREUNION_PK	VALID	2 (null)	UNIQUE	0	0
39 SIRVEN_RES_PK	VALID	2 (null)	UNIQUE	4	4
40 PRESTANADICIONAL_PK	VALID	2 (null)	UNIQUE	3	3
41 PRESTA_PK	VALID	2 (null)	UNIQUE	1	1
42 PLANESCONSUMO_PK	VALID	1 (null)	UNIQUE	5	5
43 PRODUCTOSUPER_PK	VALID	1 (null)	UNIQUE	3	3
44 GIMNASIOS_PK	VALID	1 (null)	UNIQUE	2	2

Los índices que Oracle genera automáticamente surgen al definir claves primarias o restricciones de unicidad en las tablas. En tu caso, podemos observar que se ha creado un índice correspondiente para cada clave primaria de las tablas, lo que resulta en un total de 44 índices automáticos.

Además, en el campo "columns" se indica un número específico, el cual representa la cantidad de claves primarias que cada una de las 44 tablas posee. Esto subraya la correlación directa entre la cantidad de índices generados y la estructura de las claves primarias en la base de datos.

### 3. Diseño y carga masiva de datos

#### Diseño de datos

Los datos se hacen de tal forma que cumplan con las restricciones de check y de llave primaria, pues para esto los datos siempre van a tener diferente llave primaria con el uso de una generación de datos generada por la librería de Python faker. Las fechas de las reservas corresponden a fechas de la última década hasta este año para todos los consumos, las fechas de check in corresponden a las fechas del inicio de la reserva, las fecha de los consumos corresponden a fechas entre el rango de la reserva, los costos siempre serán menores a 3 millones y mayores a 15000 pesos y la distribución de los consumos será equivalente, cada que se hace un consumo de tipo salón o spa se genera la reserva del mismo y se guarda en el atributo correspondiente.

#### Cómo se realiza

Se realiza por medio de un script en Python que logra generar el script con extensión sql con las sentencias de inserción. Estos scripts se encuentran en la carpeta Poblar de docs del repositorio. Su contenido es el siguiente:

```
import random
from faker import Faker
from datetime import datetime, timedelta

fake = Faker()

tipos_doc = ['CE', 'CC', 'TI']
tipos_hab = ['suite', 'familiar', 'doble']

archivo = open("./Scripts/poblarNuevo.sql", "w")

for _ in range(750000):
    tipo_doc = random.choice(tipos_doc)
    tipo_hab = random.choice(tipos_hab)
    num_doc = random.randint(10000000, 99999999)
    precio_hab = random.randint(100000, 9999999)
    capacidad = random.randint(1, 10)
    nombre = fake.name()
    correo = fake.email()

    fecha_in = fake.date_this_decade(before_today=True)
    dias_rand = random.randint(1, 50)
    fecha_out = fecha_in + timedelta(days=dias_rand)
    fecha_in = fecha_in.strftime('%Y-%m-%d')
    fecha_out = fecha_out.strftime('%Y-%m-%d')

    archivo.write(f"INSERT INTO InformacionClientes(tipo_documento, num_documento, nombre, correo) VALUES ('{tipo_doc}', {num_doc}, '{nombre}', '{correo}');\n")
    archivo.write(f"INSERT INTO Usuarios (tipo_documento, num_documento, nombre, correo, TiposUsuario_tipo) VALUES ('{tipo_doc}', {num_doc}, '{nombre}', '{correo}', 'cliente');\n")
```

```

archivo.write(f"INSERT INTO Reservas (fecha_inicio, fecha_salida, num_personas, Usuarios_tipo_documento,
Usuarios_num_documento, Usuarios_nombre, Usuarios_correo, PlanesConsumo_id) VALUES ({TO_DATE('{fecha_in}', 'YYYY-MM-
DD'), TO_DATE('{fecha_out}', 'YYYY-MM-DD')}, 2, '{tipo_doc}', {num_doc}, '{nombre}', '{correo}', 1);\n")
    archivo.write(f"INSERT INTO Habitaciones (capacidad, costo, TiposHabitacion_tipo) VALUES ({capacidad}, {precio_hab},
'{tipo_hab}')\n")
    archivo.write(f"INSERT INTO Reservan (Habitaciones_id, Reservas_id) VALUES ({_+1},(select id from reservas where
usuarios_num_documento = {num_doc} fetch first 1 row only));\n")
    archivo.write(f"INSERT INTO CheckIn (Reservas_id, informacionclientes_num_documento, informacionclientes_tipo_documento,
fecha_ingreso) VALUES ((select id from reservas where usuarios_num_documento = {num_doc} fetch first 1 row only),{num_doc},
'{tipo_doc}',TO_DATE('{fecha_in}', 'YYYY-MM-DD'));\n")

archivo.close()
print(">> Archivo generado con éxito")

```

El archivo de poblar 2 tiene la misma estructura, pero genera los Scripts necesarios para poblar los consumos y reservas

#### *Cómo se logra el volumen de datos solicitado*

El volumen de datos solicitado se hace ejecutando los scripts generados desde SQL developer hasta lograr que los datos pertenecientes a todas las tablas de los requerimientos sumen 750000 datos en total, luego de esto se deja de ejecutar los pedazos del script. Generados. En caso de pasarse excesivamente del volumen solicitado se hace eliminación desde la tabla Consumos y Usuarios que al tener condiciones de cascada eliminan datos en masa en las demás tablas.

#### *Qué se usó para poblar la BD*

Para esto se usan los programas Python dentro de la carpeta /dosc/Poblar/ y se ejecutan los Scripts que quedan guardados en /docs/Scripts/ como PoblarNuevo y PoblarNuevo2 con extensión SQL dentro de SQL developer y el usuario correspondiente

#### *Qué consideraciones tuvimos*

Las consideraciones tenidas en cuenta es que el volumen generado tuviera datos coherentes de las reservas, usuarios, check in para poder cumplir las restricciones de llave primaria y las llaves de rangos establecidas, los consumos con precios positivos y fechas coherentes. Para evitar confusión todos los registros se insertan con el mismo usuario Empleado por lo que será único que estará registrando los consumos. Igualmente se tiene en cuenta que los datos se distribuyen aleatoriamente buscando aleatoriedad en la cantidad en los registros y simulación de una situación real donde no en todas las fechas hay un mismo número de registros

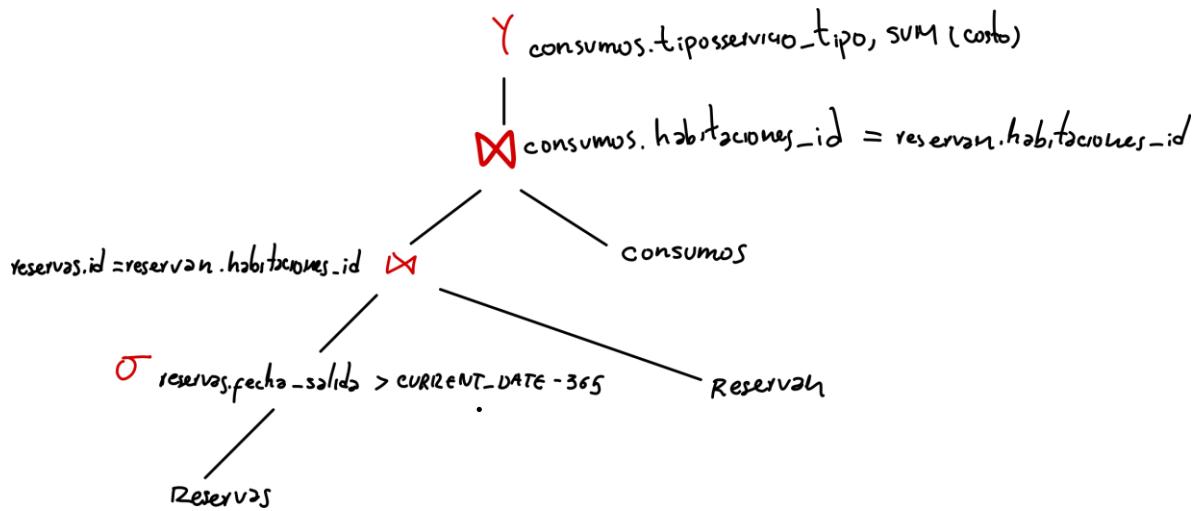
#### *Planes de Consulta Oracle*

- **Requerimiento RFC 1**

#### **Plan de Ejecucion vs Sugerencia de Oracle**

Dado que se quiere mostrar el dinero recolectado por los servicios en cada habitación en el último año corrido, se realizan dos INNER JOIN, el primero es entre las tablas de RESEVAS y RESERVAN sobre el identificador de la habitación y el segundo es entre las tablas de CONSUMOS y RESERVAN, también sobre el identificador de la habitación. Adicionalmente, se filtran los resultados mediante la consulta WHERE y se seleccionan las reservas donde la fecha de salida sea mayor que la fecha actual menos 365 días. En otra palabras, se selecciona las reservas que han tenido en el último año. Además, se agrupan los resultados mediante la cláusula GROUP BY por el ID de la habitación, lo que significa que la consulta calcula la suma de costos para cada habitación y devuelve un resultado por tipo de servicio.

Realizado grupalmente:



Sugerido por Oracle:

Como se observa, el plan de ejecución comienza con un select statement dado que la operación a realizar es una consulta. Seguido a esto se realiza 3 NESTED LOOPS para unir las condiciones entre las tablas de RESERVAS, RESERVAN y CONSUMOS. Este tipo de join es el mas adecuado pues que se esta buscando coincidencias de tipo numero y específicos, específicamente, la llave primaria de las habitaciones(ID). El Filter Predicates se encarga de filtrar los datos en las reservas del ultimo año. El Access predicate corresponde a la coincidencia entre el ID de la habitacion en ambas tablas.

-- Sentencia SQL RFC1- caso agrupar por habitación

```
SELECT consumos.habitaciones_id, sum(consumos.costo) FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
WHERE reservas.fecha_salida > (SELECT CURRENT_DATE-365 from dual)
GROUP BY consumos.habitaciones_id;
```

Plan de Consulta

### Sin índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				
HASH		GROUP BY	2552	1297
HASH JOIN			2552	1297
Access Predicates	CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID		2552	1294
MERGE JOIN			2500	348
TABLE ACCESS	RESERVAS	BY INDEX ROWID	2500	320
Filter Predicates	RESERVAS.FECHA_SALIDA > (SELECT CURRENT_DATE - 365 FROM SYS.DUAL DUAL)			
INDEX	RESERVAS_PK	FULL SCAN	50001	47
FAST DUAL			1	2
SORT		JOIN	50001	27
Access Predicates	RESERVAS.ID=RESERVAN.HABITACIONES_ID			
Filter Predicates	RESERVAS.ID=RESERVAN.HABITACIONES_ID			
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
TABLE ACCESS	CONSUMOS	FULL	51045	946

### Con índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
CREATE INDEX STATEMENT				
INDEX BUILD	TIPOSSERVICIO_IDX	NON UNIQUE	51045	55
SORT		CREATE INDEX	51045	
INDEX	TIPOSSERVICIO_IDX	FAST FULL SCAN	51045	28

### Análisis de Eficiencia

Se evidencia una alta mejora en la consulta al crear el índice:

```
CREATE
INDEX tiposervicio_idx
ON consumos(tiposervicio_tipo);
```

La cual pasa de un costo de 1297 a 55, evidenciando una mejora del 96%.

Tiempo de consulta: 0,70 s

```
-- RFC1 - caso agrupar por habitaci n

SELECT consumos.habitaciones_id, sum(consumos.costo) FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
WHERE reservas.fecha_salida > (SELECT CURRENT_DATE - 365 from dual)
GROUP BY consumos.habitaciones_id;
```

Resultado de la Consulta x

SQL | Se han recuperado 4.000 filas en 0,701 segundos

HABITACIONES_ID	SUM(CONSUMOS.COSTO)
3973	29325
	8242289

## Escenarios de Pruebas

- Caso Exitoso sin índice:

	HABITACIONES_ID	SUM(CONSUMOS.COSTO)
3974	21762	20468922
3975	12564	2001142
3976	5843	5830534
3977	19717	1657683
3978	570	6988803
3979	22019	17665596
3980	1813	5333610
3981	23514	3114385
3982	24882	15569382
3983	20888	9883422
3984	27724	386441
3985	14386	3873891
3986	21260	8031506
3987	15263	12099786
3988	3113	16049714
3989	7428	10363518
3990	27065	15400238
3991	18311	21327894
3992	9387	8600807
3993	5365	6538794
3994	24830	6141492
3995	25482	7884980
3996	29468	17984167
3997	19969	664526
3998	24133	8989579
3999	23795	14949028
4000	2684	3525226

Para este escenario de prueba, que es cuando se ejecuta sobre la base de datos que se cargo a SQL developer, se espera obtener alrededor de 4000 registros que tengan 2 columnas, las cuales corresponden a HABITACIONES\_ID y SUM(CONSUMOS.COSTO), la cual muestran como se agrupan el consumo total de cada habitación

- Caso Exitoso con índice:

	HABITACIONES_ID	SUM(CONSUMOS.COSTO)
3974	21762	20468922
3975	12564	2001142
3976	5843	5830534
3977	19717	1657683
3978	570	6988803
3979	22019	17665596
3980	1813	5333610
3981	23514	3114385
3982	24882	15569382
3983	20888	9883422
3984	27724	386441
3985	14386	3873891
3986	21260	8031506
3987	15263	12099786
3988	3113	16049714
3989	7428	10363518
3990	27065	15400238
3991	18311	21327894
3992	9387	8600807
3993	5365	6538794
3994	24830	6141492
3995	25482	7884980
3996	29468	17984167
3997	19969	664526
3998	24133	8989579
3999	23795	14949028
4000	2684	3525226

- Caso No Exitoso

Este caso consiste en una consulta que no retorna ningún registro, este caso se da cuando en la base de datos presenta problemas en la carga de la BD, haciendo que al tener un error nos dé una consulta vacía.

- Requerimiento 2

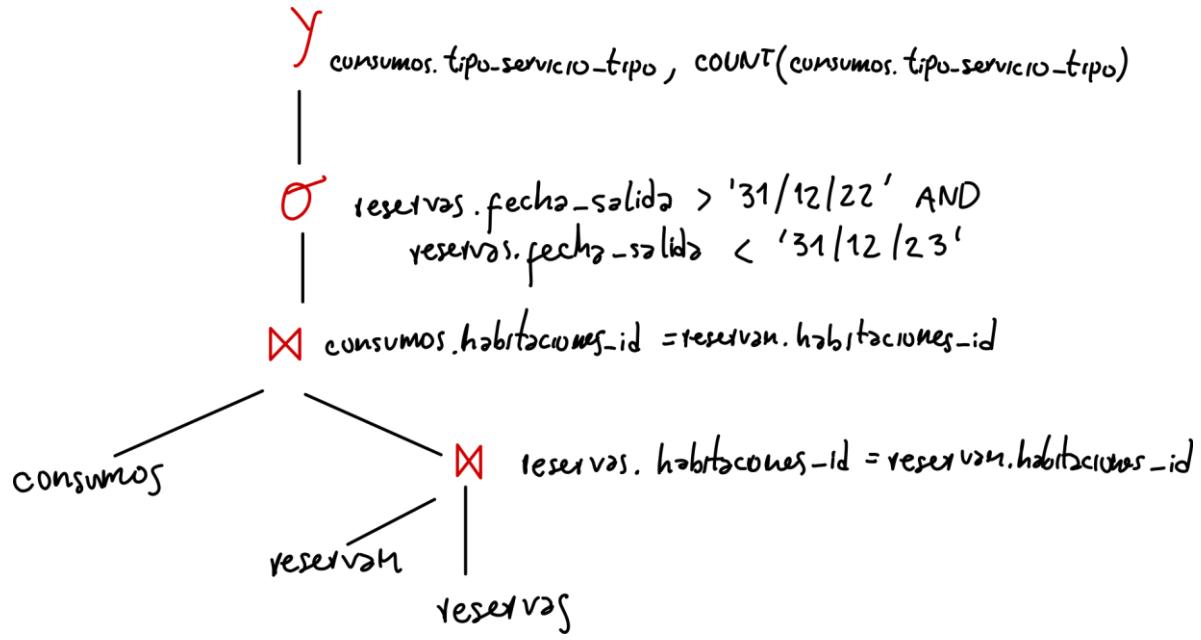
### Plan de Ejecucion vs Sugerencia de Oracle

Dado que se quieren mostrar los 20 servicios más populares determinando un período de tiempo dado, se realiza 2 INNER JOIN, primero es entre las tablas de RESERVAN y RESERVAS, donde la condición para la unión es que el valor de la columna ID en la tabla RESERVAS sea igual al valor de la columna de habitacion\_id de la tabla RESERVAN. El segundo INNER JOIN es entre las tablas de CONSUMOS y RESERVAN, también sobre los identificadores de la habitación. Esto vincula las reservas con las habitaciones correspondientes. Adicionalmente, se filtran los resultados mediante la consulta WHERE y se seleccionan las reservas donde la fecha de salida tiene que estar en el rango que se especifique. después mediante la cláusula GROUP BY se agrupan los resultados por el tipo de servicio. Esto calcula el recuento de consumos para cada tipo de servicio y devolverá un resultado por tipo de servicio. Luego la cláusula ORDER BY COUNT ordena los resultados en función de la cantidad de consumos para cada tipo de servicio en orden ascendente. Finalmente, se usa la CLAUSULA de FETCH FIRST 20 ROWS ONLY, esto hace que la consulta devolverá los 2 tipos de servicios más comunes durante el periodo de tiempo especificado.

#### -- Sentencia SQL RFC2

```
SELECT consumos.tiposervicio_tipo, COUNT(consumos.tiposervicio_tipo) AS cantidad
FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
WHERE reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
GROUP BY consumos.tiposervicio_tipo
ORDER BY COUNT(consumos.tiposervicio_tipo)
FETCH FIRST 20 ROWS ONLY;
```

Realizada grupalmente:



### Plan de Consulta

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			20	479
SORT		ORDER BY	20	479
VIEW	SYS.null		20	478
Filter Predicates				
from\$\$_subquery\$_\$006.rowlimit \$\$_rownumber<=20				
WINDOW		SORT PUSHED RANK	9	478
Filter Predicates				
ROW_NUMBER() OVER ( ORDER BY COUNT(*))<=20				
HASH		GROUP BY	9	478
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
HASH JOIN			12372	475
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
NESTED LOOPS			12372	475
STATISTICS CC				
MERGE JOIN			12119	348
TABLE A(RESERVAS		BY INDEX ROWID	12119	320
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEXRESERVAS_PK		FULL SCAN	50001	47
SORT		JOIN	50001	27
Access Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
Filter Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
INDEXRESERVAN_PK		FAST FULL SCAN	50001	25
CONSUMOS_PK		RANGE SCAN	1	127
INDEX	CONSUMOS_PK	FAST FULL SCAN	51045	127
INDEX	CONSUMOS_PK	FAST FULL SCAN		

Tiempo de ejecución: 0,07s

```
-- RFC2

SELECT consumos.tiposervicio_tipo, COUNT(consumos.tiposervicio_tipo) AS cantidad
FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
WHERE reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
GROUP BY consumos.tiposervicio_tipo
ORDER BY COUNT(consumos.tiposervicio_tipo)
FETCH FIRST 20 ROWS ONLY;
```

resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 9 en 0,067 segundos

TIPOSERVICIO TIPO	CANTIDAD
1 spa	1297
2 restaurante	1306
3 internet	1308
4 salon	1315
5 lavadoSecadoEmbolado	1323
6 bar	1324
7 prestamo	1326
8 supermercado	1331
9 tienda	1357

## Escenarios de Pruebas

- Caso Exitoso entre '31/12/22' AND '31/12/23'

TIPOSERVICIO TIPO	CANTIDAD
1 spa	1297
2 restaurante	1306
3 internet	1308
4 salon	1315
5 lavadoSecadoEmbolado	1323
6 bar	1324
7 prestamo	1326
8 supermercado	1331
9 tienda	1357

- Caso Exitoso entre '31/12/21' AND '31/12/22'

TIPOSERVICIO TIPO	CANTIDAD
1 supermercado	1425
2 salon	1431
3 internet	1447
4 prestamo	1452
5 restaurante	1466
6 spa	1474
7 tienda	1476
8 bar	1503
9 lavadoSecadoEmbolado	1520

- Caso Exitoso entre '31/12/20' AND '31/12/21'

TIPOSSERVICIO_TIPO	CANTIDAD
1 spa	1445
2 bar	1451
3 restaurante	1452
4 lavadoSecadoEmbolado	1482
5 supermercado	1503
6 tienda	1505
7 internet	1509
8 salon	1512
9 prestamo	1545

Para estos escenarios de prueba exitosos, se evidencia una clara correctitud de la consulta, debido a que para los rangos de fechas modificados se obtienen que van variando los resultados lo cual es lógico y es lo esperado en el hotel.

- Caso No Exitoso entre '31/12/20' AND '31/12/19'

TIPOSS...	CANTI...

Para este rango de fecha podemos evidenciar que no encontramos ningún registro debido a que el rango de fechas está mal establecido. Por lo tanto, al momento de filtrar información de las tablas en la consulta quedaría vacía.

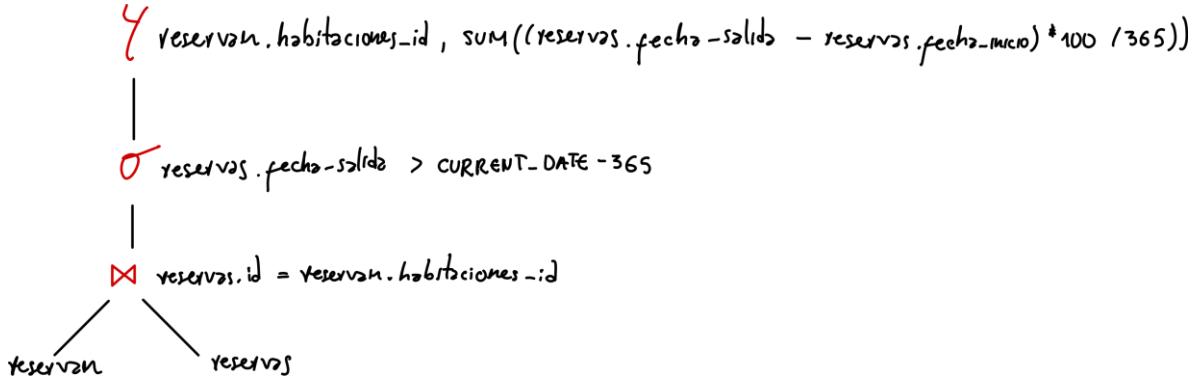
- Requerimiento 3

Dado que se quiere mostrar el índice de ocupación de cada una de las habitaciones del hotel, se realiza primero un SELECT donde se seleccionan las 2 columnas que se necesitan, la primera columna representa los IDs de las habitaciones de la tabla RESERVAN, la segunda columna seleccionada calcula el porcentaje de ocupación para cada habitación en el último año. Después se realiza un INNER JOIN entre las tablas de RESERVAN y RESERVAS sobre el identificador de la habitación. Adicionalmente, se filtran los resultados mediante la consulta WHERE y se seleccionan las reservas donde la fecha de salida sea mayor que la fecha actual menos 365 días. En otra palabras, se selecciona las reservas que han tenido en el último año. Finalmente, se usa la cláusula GRUOP BY, la cual agrupa los resultados por el ID de la habitación. Esta consulta calcula el porcentaje de ocupación para cada habitación del hotel y devolverá un resultado por habitación.

#### -- Sentencia SQL RFC3

```
SELECT reservan.habitaciones_id, ROUND(SUM((reservas.fecha_salida - reservas.fecha_inicio)*100/365), 3) AS
pct_ocupacion
FROM reservas INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
WHERE reservas.fecha_salida > (SELECT CURRENT_DATE-365 from dual)
GROUP BY reservan.habitaciones_id;
```

Realizado grupalmente:



Sugerido por Oracle:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2500	350
SORT		GROUP BY NOSORT	2500	350
MERGE JOIN			2500	348
TABLE ACCESS	RESERVAS	BY INDEX ROWID	2500	320
Filter Predicates				
RESERVAS.FECHA_SALIDA > (SELECT CURRENT_DATE-365 FROM SYS.DUAL DUAL)				
INDEX	RESERVAS_PK	FULL SCAN	50001	47
FAST DUAL			1	2
SORT		JOIN	50001	27
Access Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
Filter Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25

Tiempo de ejecución: 0,63s

```

-- RFC3

SELECT reservan.habitaciones_id, ROUND(SUM((reservas.fecha_salida - reservas.fecha_inicio)*100/365), 3) AS pct_ocupacion
FROM reservas INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
WHERE reservas.fecha_salida > (SELECT CURRENT_DATE-365 from dual)
GROUP BY reservan.habitaciones_id;

```

Resultado de la Consulta

HABITACIONES_ID	PCT_OCUACION
1	5,479
2	12,877
3	11,233
4	10,959
5	5,205
6	2,222

Escenarios de Pruebas

- Caso Exitoso:

HABITACIONES_ID	PCT_OCUACION
4035	3.014
4036	7.123
4037	9.589
4038	0.822
4039	11.233
4040	6.575
4041	3.288
4042	10.411
4043	7.945
4044	6.575
4045	2.192
4046	12.055
4047	1.644
4048	4.658
4049	11.507
4050	9.863

Para este escenario de prueba, que es cuando se ejecuta sobre la base de datos que se cargó a SQL developer, se espera obtener alrededor de 4050 registros que tengan 2 columnas, las cuales corresponden a HABITACIONES\_ID y pct\_ocupacion, la cual muestran el índice de ocupación para cada una de las habitaciones

- Caso No Exitoso

Esta caso consiste en una consulta que no retorna ningún registro, este caso se da cuando en la base de datos presenta problemas en la carga de la BD, haciendo que al tener un error nos dé una consulta vacía.

- Requerimiento 4

La consulta base del RFC4 selecciona datos de las tablas "RESERVAS", "RESERVAN" y "CONSUMOS", uniendo estas tablas mediante operaciones de unión basadas en las relaciones de clave primaria y extranjera. Luego, se eligen tres columnas específicas para mostrar en los resultados: el tipo de servicio asociado a los consumos, el costo de los servicios y la fecha de salida de las reservas. Esta consulta proporciona información general sobre los consumos y reservas, y se utiliza como punto de partida para aplicar condiciones adicionales y filtrar los servicios que cumplen con características específicas, como precio, fecha, tipo de servicio o categoría, según se requiera en las otras partes del RFC4.

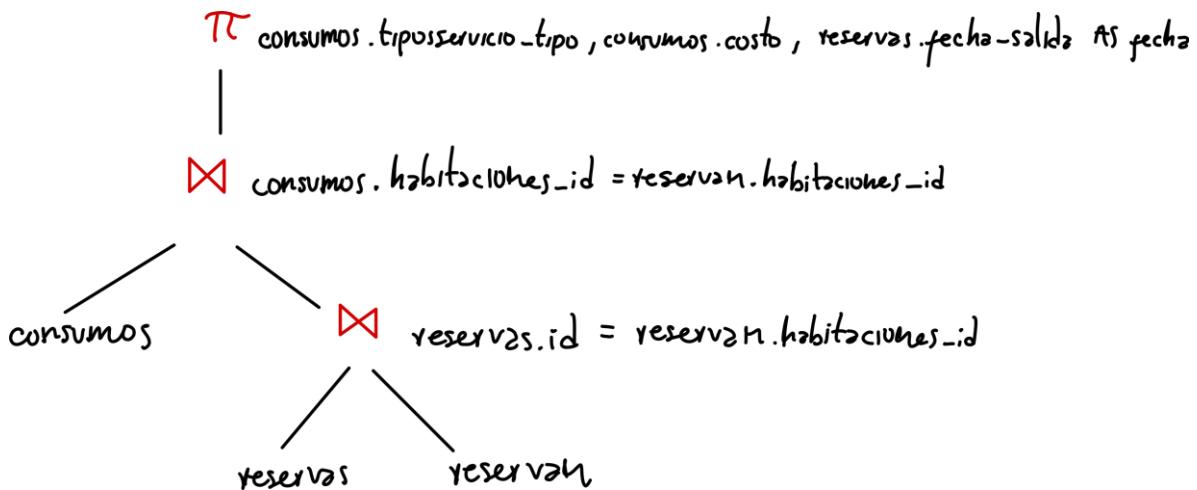
-- Sentencia SQL RFC4

```
-- base
SELECT consumos.tiposervicio_tipo, consumos.costo, reservas.fecha_salida AS fecha
FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id;
```

**Planes de Consulta**

- a) Base

Realizado grupalmente:



Sugerido por Oracle:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			51045	1294
HASH JOIN			51045	1294
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
NESTED LOOPS			51045	1294
NESTED LOOPS				
STATISTICS COLLECTOR				
MERGE JOIN				
TABLE ACCESS RESERVAS	RESERVAS	BY INDEX ROWID	50001	347
INDEX	RESERVAS_PK	FULL SCAN	50001	320
SORT		JOIN	50001	47
Access Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
Filter Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	RANGE SCAN		
INDEX				
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
TABLE ACCESS CONSUMOS	CONSUMOS	BY INDEX ROWID	1	946
TABLE ACCESS CONSUMOS	CONSUMOS	FULL	51045	946

Tiempo de consulta: 0,624s

```

-- base
SELECT consumos.tiposervicio_tipo, consumos.costo, reservas.fecha_salida AS fecha
FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id;

```

resultado de la Consulta x

SQL | Se han recuperado 3.700 filas en 0,624 segundos

TIPOSERVICIO TIPO	COSTO	FECHA
1 restaurante	359017	05/06/22

## Escenario de Prueba

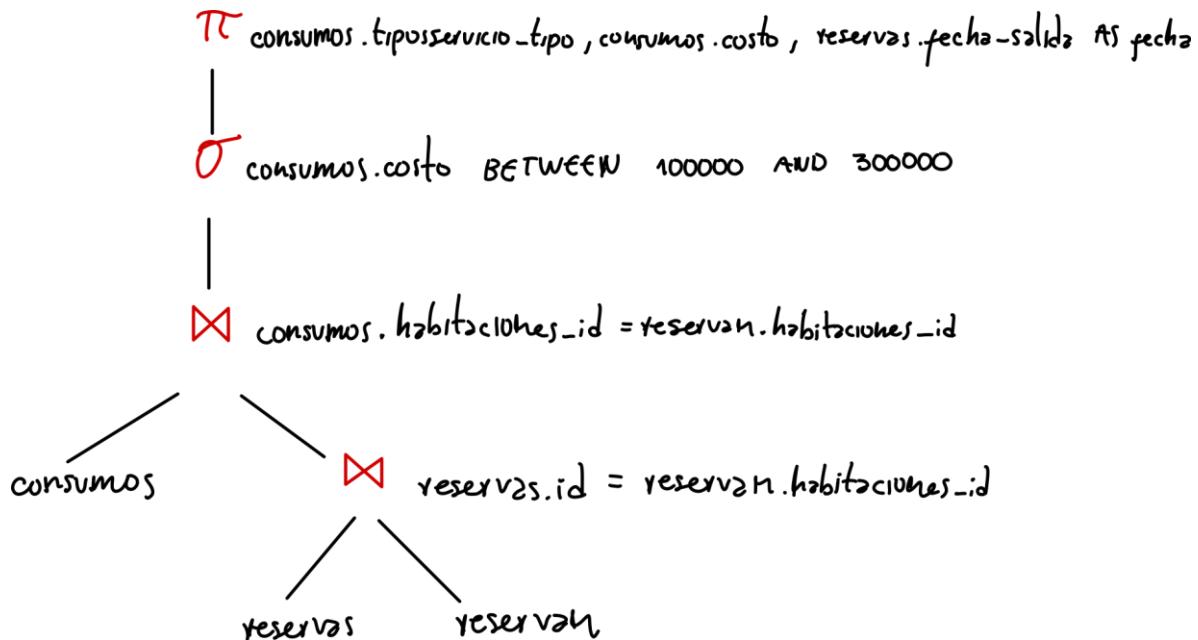
- Caso Exitoso:

SQL | Fetched 100 rows in 0.185 seconds

TIPOSERVICIO_TIPO	COSTO	FECHA
1 spa	150000	11/12/23
2 spa	150000	11/12/23
3 salon	252044	04/01/21
4 salon	106072	09/02/23
5 salon	276426	18/02/23
6 tienda	215113	30/05/22
7 supermercado	163863	15/06/20
8 restaurante	129577	07/03/21
9 spa	123731	31/12/21
10 bar	110450	03/10/20
11 prestamo	124114	27/09/23
12 supermercado	257811	27/09/23
13 spa	199374	10/12/21

- b) Precio

Realizado grupalmente:



*Sugerido por Oracle:*

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1023	1294
HASH JOIN			1023	1294
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
TABLE ACCESS	CONSUMOS	FULL	1023	946
Filter Predicates				
AND				
CONSUMOS.COSTO<=300000				
CONSUMOS.COSTO>=100000				
MERGE JOIN			50001	347
TABLE ACCESS	RESERVAS	BY INDEX ROWID	50001	320
INDEX	RESERVAS_PK	FULL SCAN	50001	47
SORT		JOIN	50001	27
Access Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
Filter Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25

Tiempo de consulta: 0,276s

```
-- precio
SELECT consumos.tiposervicio_tipo, consumos.costo, reservas.fecha_salida AS fecha
FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
WHERE consumos.costo BETWEEN 100000 AND 300000;
```

Resultado de la Consulta x Resultado de la Consulta 1 x

SQL | Todas las Filas Recuperadas: 1035 en 0,276 segundos

TIPOSERVICIO TIPO	COSTO	FECHA
1007 prestamo	241668	15/03/20

## Escenarios de Prueba

- Caso Exitoso para costo entre 5000 y 30000:

All Rows Fetched: 120 in 0.222 seconds

TIPOSERVICIO TIPO	COSTO	FECHA
110 bar	21795	27/03/22
111 bar	27101	08/04/21
112 restaurante	15627	30/07/23
113 restaurante	13639	05/09/23
114 tienda	28928	15/12/20
115 spa	23771	19/09/20
116 tienda	11055	20/04/20
117 internet	9463	06/09/22
118 spa	12757	22/03/22
119 internet	9507	12/02/21
120 internet	8247	27/09/22

- Caso Exitoso para costo entre 50000 y 100000:

SQL | All Rows Fetched: 253 in 0.419 seconds

TIPOSERVICIO_TIPO	COSTO	FECHA
243 internet	60362	06/02/20
244 internet	65115	07/03/22
245 internet	54974	09/04/21
246 bar	91009	16/07/21
247 spa	98552	04/07/21
248 spa	73304	14/12/22
249 bar	93397	08/11/20
250 tienda	68730	18/10/23
251 spa	90251	14/06/23
252 salon	77134	27/05/23
253 restaurante	71822	15/06/21

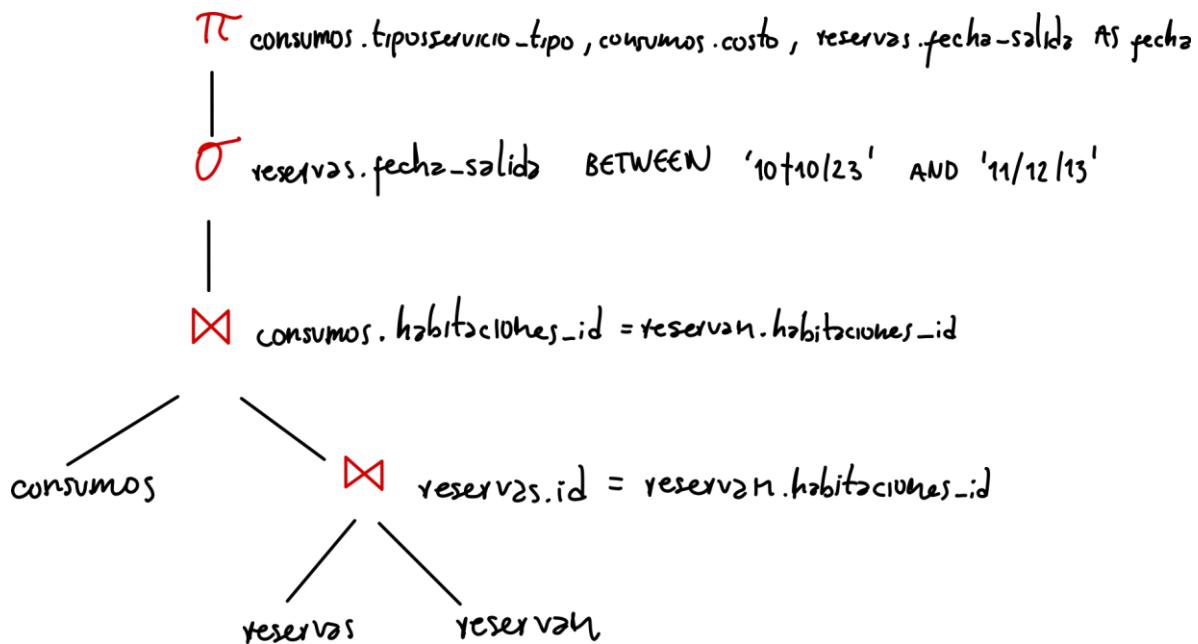
- Caso No Exitoso para costo entre 30000 y 30000:

SQL | All Rows Fetched: 0 in 0.09 seconds

TIPOSERVICIO_TIPO	COSTO	FECHA

c) Fecha

Realizado grupalmente:



Sugerido por Oracle:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2262	1294
FILTER				
Filter Predicates				
TO_DATE('11/12/23')>=TO_DATE('10/10/23')				
HASH JOIN			2262	1294
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
NESTED LOOPS			2262	1294
NESTED LOOPS				
STATISTICS COLLECTOR				
MERGE JOIN				
TABLE ACCESS RESERVAS		BY INDEX ROWID	2216	348
Filter Predicates			2216	320
AND				
RESERVAS.FECHA_SALIDA>='10/10/23'				
RESERVAS.FECHA_SALIDA<='11/12/23'				
INDEX RESERVAS_PK		FULL SCAN	50001	47
SORT		JOIN	50001	27
Access Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
Filter Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
INDEX RESERVAN_PK		FAST FULL SCAN	50001	25
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
TABLE ACCESS CONSUMOS		BY INDEX ROWID	1	946
TABLE ACCESS CONSUMOS		FULL	51045	946

Tiempo de consulta: 0,30s

```
-- fecha
SELECT consumos.tiposervicio_tipo, consumos.costo, reservas.fecha_salida AS fecha
FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
WHERE reservas.fecha_salida BETWEEN '10/10/23' AND '11/12/23';
```

Resultado de la Consulta x   Resultado de la Consulta 1 x		
SQL   Todas las Filas Recuperadas: 1549 en 0,298 segundos		
TIPOSERVICIO_TIPO	COSTO	FECHA
521 tienda	1188635	18/11/23
522 prestamo	6731580	04/11/23

Escenarios de Prueba

- Caso Exitoso para fecha entre '10/10/22' AND '11/12/22':

TIPOSERVICIO_TIPO	COSTO	FECHA
2402 lavadoSecadoEmbolado	8425571	05/12/22
2403 bar	1775526	09/11/22
2404 lavadoSecadoEmbolado	7541376	19/10/22
2405 restaurante	6784966	11/10/22
2406 internet	5948364	25/10/22
2407 internet	5912286	11/10/22
2408 prestamo	9448851	16/11/22
2409 restaurante	108247	26/10/22
2410 restaurante	8651985	28/11/22
2411 tienda	4777414	12/11/22
2412 supermercado	312702	30/10/22
2413 tienda	9877284	20/11/22
2414 prestamo	5964585	03/12/22
2415 prestamo	3281810	15/11/22
2416 spa	5259967	25/11/22
2417 tienda	5269594	29/11/22
2418 internet	1722664	24/10/22
2419 internet	3405684	20/10/22
2420 spa	8422671	13/10/22

- Caso Exitoso para fecha entre '10/10/21' AND '11/12/21':

TIPOSERVICIO_TIPO	COSTO	FECHA
2259 prestamo	3213515	23/10/21
2260 bar	1833142	12/10/21
2261 internet	2992277	15/11/21
2262 salon	4491685	03/12/21
2263 restaurante	3253538	04/12/21
2264 prestamo	2191966	15/11/21
2265 spa	7519822	08/11/21
2266 spa	9525705	26/10/21
2267 supermercado	1454446	25/11/21
2268 lavadoSecadoEmbolado	2617124	23/11/21
2269 bar	8835877	23/10/21
2270 internet	2521496	06/12/21
2271 salon	7882643	04/11/21
2272 tienda	4761743	29/10/21
2273 bar	9416009	04/12/21
2274 spa	8186351	20/10/21
2275 restaurante	4182625	01/12/21
2276 prestamo	4378037	14/10/21
2277 lavadoSecadoEmbolado	3359461	05/12/21

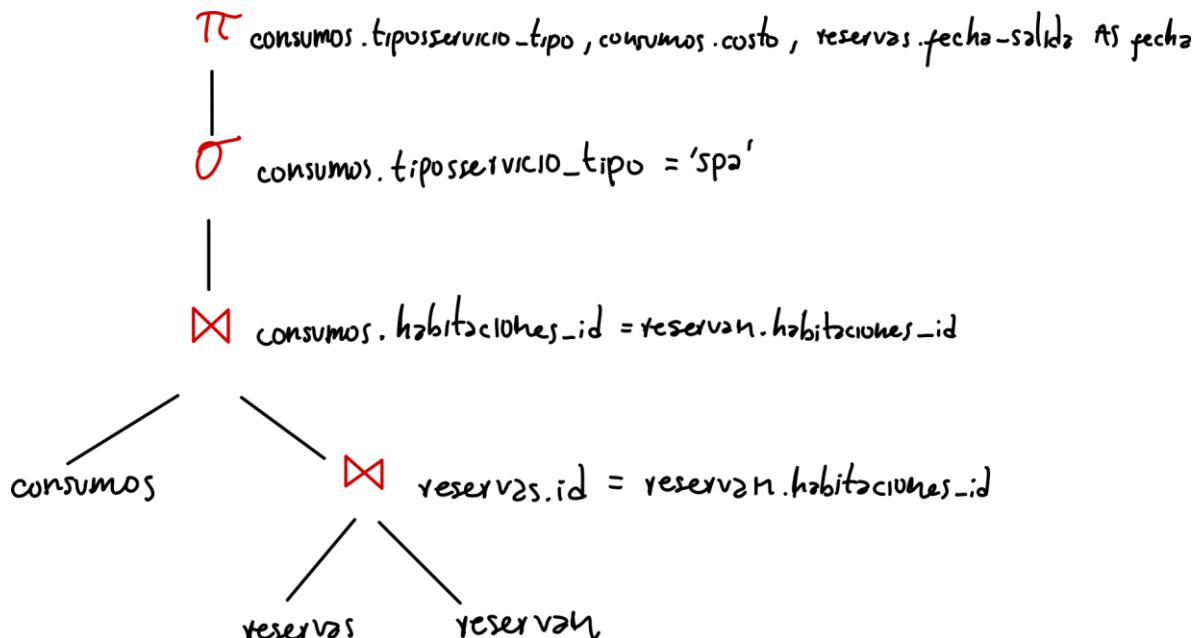
- Caso No Exitoso para fecha entre '10/10/22' AND '10/10/21':

TIPOSS...	COSTO	FECHA

En este caso de prueba se evidencia un error en las fechas límite establecidas y por lo tanto se espera que no devuelva ningún registro, lo cual es correcto.

- d) Tipo

Realizado grupalmente:



Sugerido por Oracle:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5598	1294
HASH JOIN			5598	1294
Access Predicates	CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID			
TABLE ACCESS	CONSUMOS	FULL	5598	946
Filter Predicates	CONSUMOS.TIPOSERVICIO_TIPO='spa'			
MERGE JOIN				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	50001	347
INDEX	RESERVAS_PK	FULL SCAN	50001	320
SORT		JOIN	50001	47
Access Predicates	RESERVAS.ID=RESERVAN.HABITACIONES_ID			
Filter Predicates	RESERVAS.ID=RESERVAN.HABITACIONES_ID			
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	27

Tiempo de consulta: 0,64s

```
-- tipo
SELECT consumos.tiposervicio_tipo, consumos.costo, reservas.fecha_salida AS fecha
FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
WHERE consumos.tiposervicio_tipo = 'spa';
```

resultado de la Consulta x Resultado de la Consulta 1 x  
 SQL | Se han recuperado 3.900 filas en 0,638 segundos  

TIPOSERVICIO_TIPO	COSTO	FECHA
-------------------	-------	-------

### Escenarios de Prueba

- Caso Exitoso para tipo = Restaurante:

TIPOSERVICIO_TIPO	COSTO	FECHA
5582 restaurante	9839207	23/10/20
5583 restaurante	7087684	29/02/20
5584 restaurante	6817690	10/03/21
5585 restaurante	2882024	08/12/20
5586 restaurante	4925069	17/02/23
5587 restaurante	8374928	14/05/20
5588 restaurante	6305489	07/07/23
5589 restaurante	411053	19/10/23
5590 restaurante	9870935	05/07/21
5591 restaurante	5459845	31/12/21
5592 restaurante	5813638	14/03/21
5593 restaurante	9824191	05/10/22
5594 restaurante	2863247	31/03/22
5595 restaurante	8257563	24/01/23
5596 restaurante	3634591	15/03/23
5597 restaurante	13639	05/09/23
5598 restaurante	3579753	25/06/23
5599 restaurante	974068	23/11/22
5600 restaurante	2475318	23/02/20

- Caso Exitoso para tipo = Tienda:

	TIPOSERVICIO_TIPO	COSTO	FECHA
5703	tienda	8668347	12/04/22
5704	tienda	4025242	21/03/22
5705	tienda	9673208	26/07/22
5706	tienda	2711996	17/02/23
5707	tienda	1009526	07/02/23
5708	tienda	6967690	27/05/23
5709	tienda	1981296	07/11/22
5710	tienda	307134	14/10/21
5711	tienda	7763522	03/05/23
5712	tienda	5671142	05/08/20
5713	tienda	1727164	18/10/20
5714	tienda	3645153	09/11/21
5715	tienda	1044708	01/03/23
5716	tienda	4776591	02/01/22
5717	tienda	3574680	04/02/20
5718	tienda	6687857	06/02/23
5719	tienda	9478057	14/08/23
5720	tienda	5234150	10/06/20
5721	tienda	4649205	03/05/22

- Caso No Exitoso para tipo = gym:

				SQL	All Rows Fetched: 0 in 0.247 seconds

En este caso cuando tipo = gym no devuelve ningún registro debido a que no tenemos establecido en la tabla consumos en la columna tiposervicio\_tipo ningún parámetro denominado 'gym'

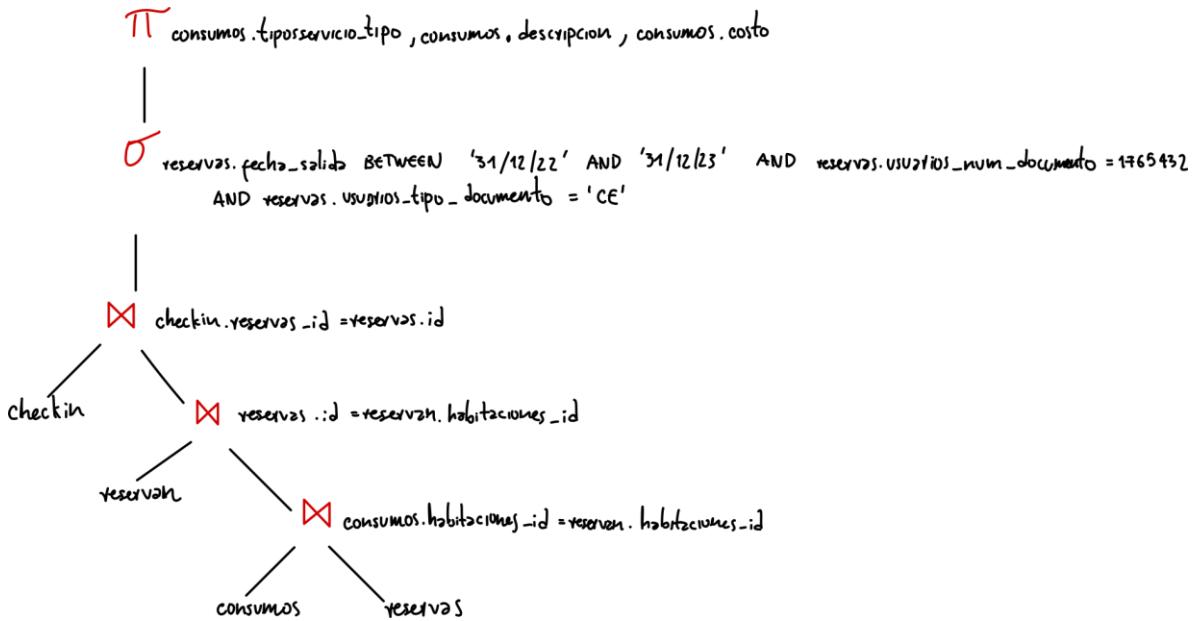
- Requerimiento 5

Dado que se quiere mostrar el consumo en el hotel por un usuario dado, en un rango de fechas indicado, se realizan tres INNER JOIN, el primero es entre las tablas de RESEVAS y RESERVAN sobre el identificador de la habitación, el segundo es entre las tablas de CONSUMOS y RESERVAN, también sobre el identificador de la habitación y el tercero es entre las tablas de CHECKIN y RESERVAS la cual vincula los datos de ID en CHECKIN con las reservas correspondientes. Adicionalmente, se filtran los resultados mediante la consulta WHERE y se seleccionar los servicios relacionados con reservas cuya fecha de salida estén dentro del rango de fecha especificado. Además, se filtra por un usuario específico identificado por su número ID y tipo de documento. Esto permite mostrar los consumos realizados por ese usuario en el periodo de tiempo especificado.

#### -- Sentencia SQL RFC5

```
SELECT consumos.tiposervicio_tipo, consumos.descripcion, consumos.costo FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
INNER JOIN CHECKIN
ON checkin.reservas_id = reservas.id
WHERE reservas.fecha_salida BETWEEN '31/12/22' AND '31/12/23'
AND reservas.usuarios_num_documento = 1765432 AND reservas.usuarios_tipo_documento = 'CE';
```

Realizado grupalmente:



Sugerido por Oracle:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	351
FILTER				
Filter Predicates	TO_DATE('31/12/23')>=TO_DATE('31/12/22')			
HASH JOIN			1	351
Access Predicates	CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID			
NESTED LOOPS			1	351
NESTED LOOPS			2	351
STATISTICS COLLECTOR			1	348
HASH JOIN				
Access Predicates	CHECKIN.RESERVAS_ID=RESERVAS.ID			
NESTED LOOPS			1	348
STATISTICS CC				
MERGE JOIN			1	347
TABLE ARESERVAS		BY INDEX ROWID	1	320
Filter Predicates	RESERVAS.USUARIOS_NUM_DOCUMENTO=1765432			
AND	RESERVAS.USUARIOS_TIPO_DOCUMENTO='CE'			
AND	RESERVAS.FECHA_SALIDA>='31/12/22'			
AND	RESERVAS.FECHA_SALIDA<='31/12/23'			
INDEXRESERVAS_PK		FULL SCAN	50001	47
JOIN			50001	27
SORT				
Access Predicates	RESERVAS.ID=RESERVAN.HABITACIONES_ID			
Filter Predicates	RESERVAS.ID=RESERVAN.HABITACIONES_ID			
INDEXRESERVAN_PK		FAST FULL SCAN	50001	25
CHECKIN_PK		RANGE SCAN	1	1
INDEX			1	1
INDEX			1	1
Access Predicates	CHECKIN.RESERVAS_ID=RESERVAS.ID			
INDEX			1	1
CHECKIN_PK		FAST FULL SCAN	1	1
INDEX			2	1
CONSUMOS_PK		RANGE SCAN		
TABLE ACCESS	CONSUMOS	BY INDEX ROWID	1	3
TABLE ACCESS	CONSUMOS	FULL	1	3

Tiempo de ejecución: 0,03s

```
-- RFC5

SELECT consumos.tiposervicio_tipo, consumos.descripcion, consumos.costo FROM RESERVAS
INNER JOIN RESERVAN
ON reservas.id = reservan.habitaciones_id
INNER JOIN CONSUMOS
ON consumos.habitaciones_id = reservan.habitaciones_id
INNER JOIN CHECKIN
ON checkin.reservas_id = reservas.id
WHERE reservas.fecha_salida BETWEEN '31/12/22' AND '31/12/23'
AND reservas.usuarios_num_documento = 1765432 AND reservas.usuarios_tipo_documento = 'CE';
```

Resultado de la Consulta x | Resultado de la Consulta 1 x | Resultado de la Consulta 2 x

SQL | Todas las Filas Recuperadas: 12 en 0,029 segundos

### Escenarios de Prueba

- Caso Exitoso para el cliente April Decker con TI = 20392673 entre las fechas '31/12/15' AND '31/12/23':

TIPOSERVICIO_TIPO	DESCRIPCION	COSTO
1 prestamo	As night member one quickly. Give environment brother. You news side about identify. Analysis well quite use carry rise. May sign since thought stand by green.	8284448

- Caso Exitoso para el cliente Craig Johnson con CE = 34180417 entre las fechas '31/12/15' AND '31/12/23':

TIPOSERVICIO_TIPO	DESCRIPCION	COSTO
1	Campaign tough law. Lead they answer hard democratic. Her peace buy south today prepare. At talk though collection beat. Ever level feeling. Interview theory group at rather watch guy.	5163836

- Caso No Exitoso para el cliente Craig Johnson con CC = 34180417 entre las fechas '31/12/15' AND '31/12/23':

TIPOSERVICIO_TIPO	DESCRIPCION	COSTO

Se debe a que el cliente esta registrado en nuestra base de datos con CE y no con CC, por lo tanto al momento de realizar la consulta no lo encuentra

- Requerimiento 6

El RFC6 tiene tres partes que buscan analizar la operación del HotelAndes para diferentes aspectos de su funcionamiento a lo largo de su tiempo de operación:

1. **Fechas de Mayor Ocupación:** La primera parte de la consulta utiliza una estructura de consulta recursiva para generar una lista de fechas entre la fecha de inicio y la fecha de salida de las reservas. Luego, se cuenta cuántas veces cada fecha aparece en las reservas y se almacena en la tabla temporal "CountMax". Finalmente, se selecciona la fecha con el recuento máximo de ocupación.

2. **Fechas de Mayores Ingresos por Consumo:** La segunda parte busca las fechas con los mayores ingresos por consumo. Realiza una unión de las tablas "RESERVAS", "RESERVAN" y "CONSUMOS", calcula el costo total de los consumos para cada día y almacena esta información en la tabla temporal "CountMax". Luego, selecciona la fecha con el costo total máximo.
3. **Fechas de Menor Ocupación:** La tercera parte es similar a la primera, pero busca las fechas con la menor ocupación. Utiliza una estructura de consulta recursiva para generar una lista de fechas y calcula el recuento de ocupación para cada fecha, almacenando estos datos en la tabla temporal "CountMin". Luego, selecciona la fecha con el recuento mínimo de ocupación.

Cada parte de la consulta utiliza estructuras de consulta recursiva y subconsultas para realizar los cálculos necesarios. Estas partes se ejecutan de manera independiente para identificar las fechas de interés en cuanto a ocupación, ingresos por consumo y menor demanda en la operación del hotel a lo largo del tiempo.

#### -- Sentencias SQL General RFC6

-- fechas mayor ocupación

```
WITH CountMax AS (
  SELECT fecha, COUNT(*) AS conteo
  FROM (
    SELECT reservas.fecha_inicio + LEVEL- 1 AS fecha
    FROM RESERVAS
    CONNECT BY PRIOR fecha_inicio = fecha_inicio
    AND PRIOR DBMS_RANDOM.VALUE IS NOT NULL
    AND LEVEL <= fecha_salida- fecha_inicio + 1
  )
  GROUP BY fecha
)
SELECT fecha
FROM CountMax
WHERE conteo = (SELECT MAX(conteo) FROM CountMax);
```

-- fechas mayores ingresos por consumo

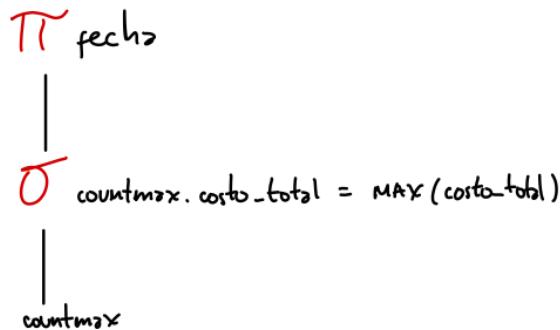
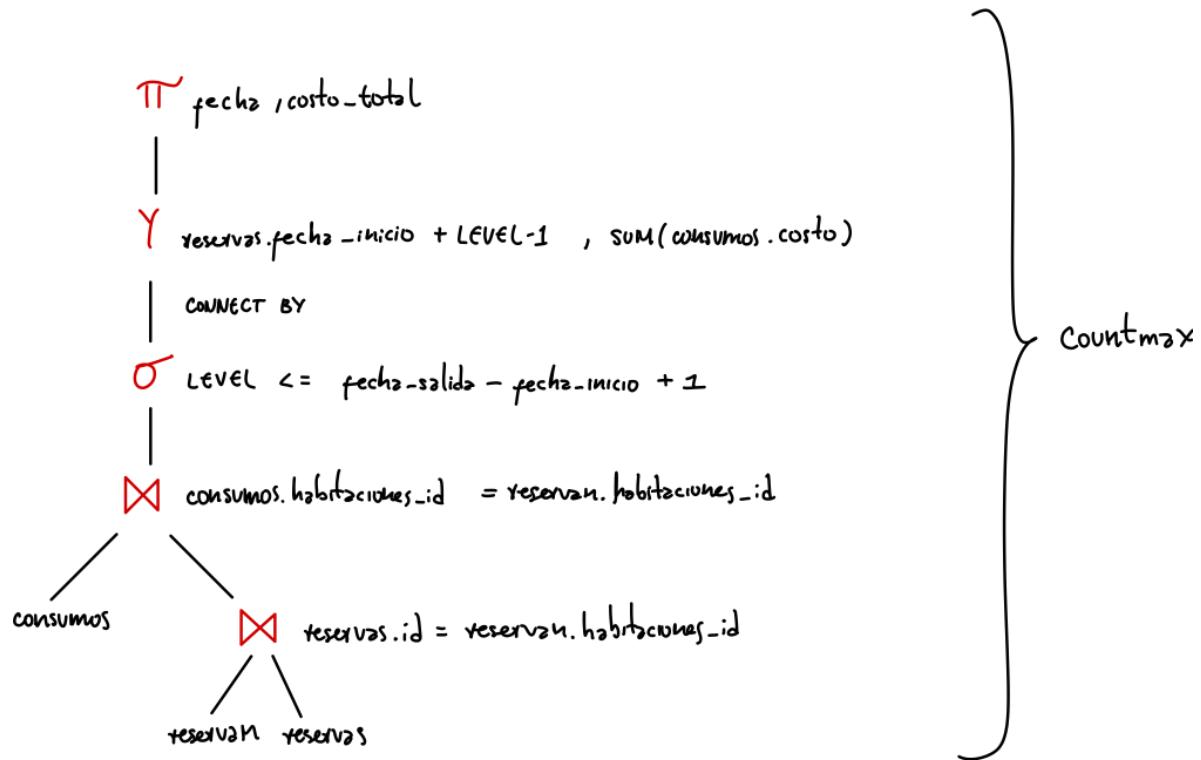
```
WITH CountMax AS (
  SELECT reservas.fecha_inicio + LEVEL- 1 AS fecha, SUM(consumos.costo) AS costo_total
  FROM RESERVAS
  INNER JOIN RESERVAN
  ON reservas.id = reservan.habitaciones_id
  INNER JOIN CONSUMOS
  ON consumos.habitaciones_id = reservan.habitaciones_id
  GROUP BY reservas.fecha_inicio + LEVEL - 1
  CONNECT BY PRIOR fecha_inicio = fecha_inicio
  AND PRIOR DBMS_RANDOM.VALUE IS NOT NULL
  AND LEVEL <= fecha_salida- fecha_inicio + 1
)
SELECT fecha
FROM CountMax
WHERE costo_total = (SELECT MAX(costo_total) FROM CountMax);
```

-- fechas menor ocupacion

```
WITH CountMin AS (
    SELECT fecha, COUNT(*) AS conteo
    FROM (
        SELECT reservas.fecha_inicio + LEVEL- 1 AS fecha
        FROM RESERVAS
        CONNECT BY PRIOR fecha_inicio = fecha_inicio
        AND PRIOR DBMS_RANDOM.VALUE IS NOT NULL
        AND LEVEL <= fecha_salida- fecha_inicio + 1
    )
    GROUP BY fecha
)
SELECT fecha
FROM CountMin
WHERE conteo = (SELECT MIN(conteo) FROM CountMin);
```

#### Planes de Consulta

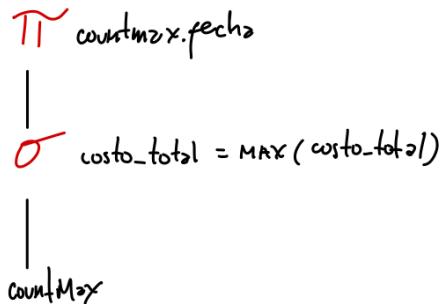
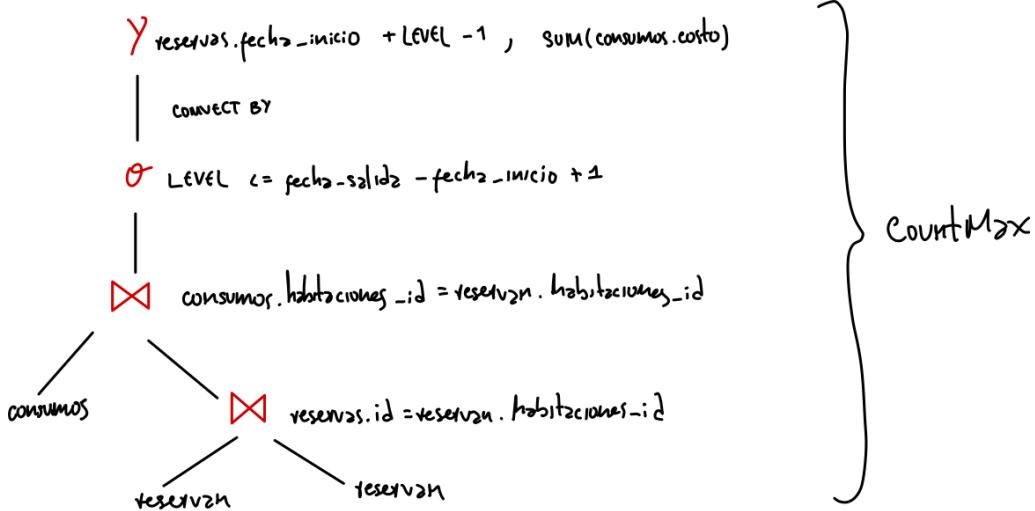
Realizado grupalmente:



a. Fecha mayor ocupación

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			50001	383
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D7234_CF93870	(CURSOR DURATION MEMORY)		
HASH		GROUP BY	50001	357
VIEW		WITHOUT FILTERING	50001	355
CONNECT BY				
Access Predicates	FECHA_INICIO=PRIOR FECHA_INICIO			
Filter Predicates				
AND	PRIOR DBMS_RANDOM.VALUE() IS NOT NULL LEVEL<=FECHA_SALIDA-FECHA_INICIO+1			
TABLE ACCESS	RESERVAS	FULL	50001 50001	13
VIEW				
Filter Predicates	CONTEO=(SELECT MAX( CONTEO ) FROM (SELECT /*+CACHE(T1) */ INTERNAL_FUNCTION(C0) FECHA,C1 CONTEO FROM SYS.SYS_TEMP_0FD9D7234_CF93870 T1) COUNTMAX)			
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7234_CF93870	FULL	50001	13
SORT		AGGREGATE	1	
VIEW			50001	13
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7234_CF93870	FULL	50001	13

Realizado grupalmente:



b. Fecha mayores ingresos por consumo

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1397	1304
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D7238_CF93870	(CURSOR DURATION MEMORY) GROUP BY WITHOUT FILTERING	1397	1296
HASH				
CONNECT BY				
Access Predicates	$\text{RESERVAS.FECHA\_INICIO} = \text{PRIOR RESERVAS.FECHA\_INICIO}$			
Filter Predicates				
AND				
Access Predicates	$\text{PRIOR DBMS\_RANDOM.VALUE()} \text{ IS NOT NULL}$			
	$\text{LEVEL} \leq \text{RESERVAS.FECHA\_SALIDA} - \text{RESERVAS.FECHA\_INICIO} + 1$			
HASH JOIN			51045	1294
Access Predicates	$\text{CONSUMOS.HABITACIONES\_ID} = \text{RESERVAN.HABITACIONES\_ID}$			
TABLE ACCESS	CONSUMOS	FULL	51045	946
MERGE JOIN			50001	347
TABLE ACCESS	RESERVAS	BY INDEX ROWID	50001	320
INDEX	RESERVAS_PK	FULL SCAN	50001	47
SORT		JOIN	50001	27
Access Predicates	$\text{RESERVAS.ID} = \text{RESERVAN.HABITACIONES\_ID}$			
Filter Predicates				
RESERVAS.ID = RESERVAN.HABITACIONES_ID				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
			1397	4
VIEW				
Filter Predicates	$\text{COSTO\_TOTAL} = (\text{SELECT MAX}(\text{COSTO\_TOTAL}) \text{ FROM } (\text{SELECT /*+ CACHE (T1) */ INTERNAL\_FUNCTION(C0) FECHA,C1 COSTO\_TOTAL FROM SYS.SYS\_TEMP\_0FD9D7238\_CF93870 T1}) \text{ COUNTMAX})$			
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7238_CF93870	FULL	1397	4
SORT		AGGREGATE	1	
VIEW			1397	4
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7238_CF93870	FULL	1397	4

c. Fecha menor ocupación

Realizado grupalmente:

```

Y fechas, COUNT(*)
|
T1 reservas.fechas_inicio + LEVEL-1
| CONNECT BY
O level <= fechas_salidas - fechas_inicio + 1
|
O PRIOR DBMS_RANDOM.VALUE IS NOT NULL
|
O PRIOR fechas_inicio = fechas_inicio
|
reservas
    
```

CountMin

```

T1 countmin.fechas
|
O conteo = MIN(conteo)
|
countmin
    
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			50001	383
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D723B_CF93870	(CURSOR DURATION MEMORY)		
HASH		GROUP BY	50001	357
VIEW			50001	355
CONNECT BY		WITHOUT FILTERING		
Access Predicates				
FECHA_INICIO=PRIOR FECHA_INICIO				
Filter Predicates				
AND				
PRIOR DBMS_RANDOM.VALUE() IS NOT NULL				
LEVEL<=FECHA_SALIDA-FECHA_INICIO+1				
TABLE ACCESS	RESERVAS	FULL	50001	13
VIEW			50001	
Filter Predicates				
CONTEO=(SELECT MIN(CONTEO) FROM (SELECT /*+ CACHE (T1) */ INTERNAL_FUNCTION(C0) FECHA,C1 CONTEO FROM SYS.SYS_TEMP_0FD9D723B_CF93870 T1) COUNTMIN)				
TABLE ACCESS	SYS.SYS_TEMP_0FD9D723B_CF93870	FULL	50001	13
SORT		AGGREGATE	1	
VIEW			50001	13
TABLE ACCESS	SYS.SYS_TEMP_0FD9D723B_CF93870	FULL	50001	13

Tiempo de ejecución: (x) +

Escenarios de Prueba:

- Requerimiento 7

La consulta RFC7 busca identificar a los "buenos clientes" del HotelAndes, definidos como aquellos que han pasado al menos dos semanas en el hotel en el último año o que han tenido consumos por un valor total de más de \$15'000.000.00 en el mismo período. La consulta se divide en tres partes:

1. **Estadía:** La primera parte calcula la cantidad total de días que cada cliente ha pasado en el hotel durante el último año, agrupando los datos por tipo de documento, número de documento y nombre del cliente.
2. **Mejores Consumos:** La segunda parte calcula el costo total de los consumos realizados por cada cliente en el último año, nuevamente agrupando los datos por tipo de documento, número de documento y nombre del cliente.
3. **Selección de Buenos Clientes:** Finalmente, la consulta compila los resultados de las dos partes anteriores. Selecciona a los clientes que han tenido la estancia más larga (máximo número de días) o los clientes con el mayor costo total de consumos en el último año. Luego, se agrupan los resultados para mostrar la información de los "buenos clientes".

La consulta utiliza estructuras de consulta comunes (WITH) para calcular la estadía y los mejores consumos antes de combinar los resultados en una lista de "buenos clientes". Esta consulta ayuda a identificar a los clientes que han tenido una larga estadía o han gastado significativamente en consumos, lo que sugiere que son clientes valiosos para el hotel.

### Sentencia SQL RFC7

```

WITH estadia AS (
  SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento,
  reservas.usuarios_nombre, SUM(reservas.fecha_salida - reservas.fecha_inicio) AS dias
  FROM RESERVAS
  GROUP BY reservas.usuarios_num_documento, reservas.usuarios_tipo_documento,
  reservas.usuarios_nombre
)
, mejoresconsumos AS (
  SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento,
  reservas.usuarios_nombre, SUM(costo) AS costo_total
  FROM RESERVAS
  INNER JOIN RESERVAN
  ON reservas.id = reservan.habitaciones_id
  INNER JOIN CONSUMOS
  ON consumos.habitaciones_id = reservan.habitaciones_id
  INNER JOIN CHECKIN
  ON checkin.reservas_id = reservas.id
  WHERE reservas.fecha_salida > (SELECT CURRENT_DATE - 365 from dual)
  GROUP BY reservas.usuarios_tipo_documento, reservas.usuarios_num_documento,
  reservas.usuarios_nombre
)
SELECT * FROM (
  SELECT usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre
  FROM ESTADIA
  WHERE dias = (SELECT MAX(dias) FROM estadia)
  UNION ALL
  SELECT usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre
  FROM mejoresconsumos
  WHERE costo_total = (SELECT MAX(costo_total) FROM mejoresconsumos)
  GROUP BY usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre
)
GROUP BY usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre;

```

Tiempo de consulta: 0.606s

```
-- RFC7

WITH estadia AS (
    SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento,
    reservas.usuarios_nombre, SUM(reservas.fecha_salida - reservas.fecha_inicio) AS dias
    FROM RESERVAS
    GROUP BY reservas.usuarios_num_documento, reservas.usuarios_tipo_documento,
    reservas.usuarios_nombre
)
, mejoresconsumos AS (
    SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento,
    reservas.usuarios_nombre, SUM(costo) AS costo_total
    FROM RESERVAS
    INNER JOIN RESERVAN
    ON reservas.id = reservan.habitaciones_id
    INNER JOIN CONSUMOS
    ON consumos.habitaciones_id = reservan.habitaciones_id
    INNER JOIN CHECKIN
    ON checkin.reservas_id = reservas.id
    WHERE reservas.fecha_salida > (SELECT CURRENT_DATE-365 from dual)
    GROUP BY reservas.usuarios_tipo_documento, reservas.usuarios_num_documento,
    reservas.usuarios_nombre
)
SELECT * FROM (
    SELECT usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre
    FROM ESTADIA
    WHERE dias = (SELECT MAX(dias) FROM estadia)
    UNION ALL
    SELECT usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre
    FROM mejoresconsumos
    WHERE costo_total = (SELECT MAX(costo_total) FROM mejoresconsumos)
    GROUP BY usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre
)
GROUP BY usuarios_tipo_documento, usuarios_num_documento, usuarios_nombre;
```

Query Result x | Query Result 1 x

SQL | All Rows Fetched: 1006 in 0.606 seconds

USUARIOS_TIPO_DOCUMENTO	USUARIOS_NUM_DOCUMENTO	USUARIOS_NOMBRE
1001 TI	35581547	Melissa Lang
1002 CE	29426608	Susan Baker
1003 CE	76655622	Tommy York
1004 TI	94062031	Bryan Sanchez
1005 CE	50463622	Christian Leblanc
1006 CE	1765432	Eduardo benitez

Realizado grupalmente:

$\left\{ \begin{array}{l} \text{π}_{\text{reservas.usuarios\_tipo\_documento}, \text{reservas.usuarios\_num\_documento}, \text{reservas.usuarios\_nombre}, \text{dias}} \\ | \\ \text{Y}_{\text{reservas.usuarios\_tipo\_documento}, \text{reservas.usuarios\_num\_documento}, \text{reservas.usuarios\_nombre}; \text{SUM}(\text{fecha\_salida} - \text{fecha\_inicio})} \\ | \\ \text{reservas} \end{array} \right.$

$\text{Y}_{\text{reservas.usuarios\_tipo\_documento}, \text{reservas.usuarios\_num\_documento}, \text{reservas.usuarios\_nombre}; \text{sum}(\text{costo})}$   
 $\text{O}_{\text{reservas.fecha\_salida} > \text{CURRENT\_DATE} - 365}$   
 $\text{X}_{\text{checkin.reservas\_id} = \text{reservas.id}}$   
 $\text{checkin}$   
 $\text{consumos}$   
 $\text{X}_{\text{consumos.habitaciones\_id} = \text{reservan.habitaciones\_id}}$   
 $\text{reservan}$   
 $\text{X}_{\text{reservas.id} = \text{reservan.habitaciones\_id}}$   
 $\text{reservas}$   
mejores consumos

$\text{π}^*$   
 $\cup$   
 $\text{O}_{\text{estadiz.dias} = \text{MAX}(\text{estadiz.dias})}$   
 $\text{estadiz}$   
 $\text{Y}_{\text{mejores.usuarios\_tipo\_documento}, \text{mejores.usuarios\_num\_documento}, \text{mejores.usuarios\_nombre}}$   
 $\text{O}_{\text{costo\_total} = \text{MAX}(\text{costo\_total})}$   
mejores consumos

## Plan de Consulta

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			52553	2377
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D723C_CF93870	(CURSOR DURATION MEMORY)		
HASH		GROUP BY	50001	868
TABLE ACCESS	RESERVAS	FULL	50001	355
LOAD AS SELECT	SYS_TEMP_0FD9D723D_CF93870	(CURSOR DURATION MEMORY)		
HASH		GROUP BY	2552	1341
HASH JOIN			2552	1338
Access Predicates	CONSUMOS.HABITACIONES_ID = RESERVAN.HABITACIONES_ID			
HASH JOIN			2500	391
Access Predicates	CHECKIN.RESERVAS_ID = RESERVAS.ID			
MERGE JOIN				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	2500	348
Filter Predicates	RESERVAS.FECHA_SALIDA > (SELECT CURRENT_DATE - 365 FROM SYS.DUAL DUAL)		2500	320
INDEX	RESERVAS_PK	FULL SCAN	50001	47
FAST DUAL			1	2
SORT		JOIN	50001	27
Access Predicates	RESERVAS.ID = RESERVAN.HABITACIONES_ID			
Filter Predicates	RESERVAS.ID = RESERVAN.HABITACIONES_ID			
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CHECKIN_PK	FAST FULL SCAN	50042	43
TABLE ACCESS	CONSUMOS	FULL	51045	946
GROUP BY			52553	168
INDEX			52553	166
TABLE ACCESS			50001	75
VIEW				
UNION-ALL				
VIEW				
Filter Predicates	DIAS = (SELECT MAX(DIAS) FROM (SELECT /*+ CACHE (T1) */ C0 USUARIOS_TIPO_DOCUMENTO, C1 USUARIOS_NUM_DOCUMENTO, C2 USUARIOS_NOMBRE, C3 DIAS FROM SYS.SYS_TEMP_0FD9D723C_CF93870)		50001	75
TABLE ACCESS	SYS.SYS_TEMP_0FD9D723C_CF93870	FULL		
SORT		AGGREGATE	1	
VIEW			50001	75
TABLE ACCESS	SYS.SYS_TEMP_0FD9D723C_CF93870	FULL	50001	75
GROUP BY			2552	15
INDEX			2552	7
VIEW				
Filter Predicates	COSTO_TOTAL = (SELECT MAX(COSTO_TOTAL) FROM (SELECT /*+ CACHE (T1) */ C0 USUARIOS_TIPO_DOCUMENTO, C1 USUARIOS_NUM_DOCUMENTO, C2 USUARIOS_NOMBRE, C3 COSTO_TOTAL FROM SYS.SYS_TEMP_0FD9D723D_CF93870)		2552	7
TABLE ACCESS	SYS.SYS_TEMP_0FD9D723D_CF93870	FULL		
AGGREGATE			1	
VIEW			2552	7
TABLE ACCESS	SYS.SYS_TEMP_0FD9D723D_CF93870	FULL		

## Escenarios de Prueba:

- Caso Exitoso

Fetched 100 rows in 0.183 seconds			
	USUARIOS_TIPO_DOCUMENTO	USUARIOS_NUM_DOCUMENTO	USUARIOS_NOMBRE
1	CE	34493805	Sean Valencia
2	CE	77651485	Sean Armstrong
3	CE	46395346	Brittany Shaw
4	CE	55271155	Joan Briggs
5	TI	69001657	Jacob Salazar
6	CC	53294054	Alyssa Gonzales
7	CE	78177632	Dr. William Gutierrez
8	CC	34748622	Albert Bell
9	CE	86106956	Chad Davis
10	TI	61474968	William Reese
11	CC	13222006	Darrell Pennington
12	CC	43904785	Catherine Campbell
13	CC	84798666	Gary Bennett
14	TI	52130918	Tamara Bender
15	CE	70818037	Kelly Roach
16	CC	30409468	Andrew Tate

Para este escenario de prueba, se espera obtener alrededor de 1006 registros que tengan 3 columnas, las cuales corresponden a USUARIOS\_TIPO\_DOCUMENTO, USUARIOS\_NUM\_DOCUMENTO y USUARIOS\_NOMBRE, la cual muestran a los buenos clientes que tiene HOTELANDES

- Caso No Exitoso

Esta caso consiste en una consulta que no retorna ningún registro en el QUERY RESULT, este caso se da cuando en la base de datos presenta problemas en la carga de la BD, haciendo que al tener un error nos dé una consulta vacía pero se tiene igualmente las 3 columnas en el resultado.

- Requerimiento 8

La consulta RFC8 tiene como objetivo identificar los servicios que han sido solicitados menos de tres veces por semana durante el último año de operación de HotelAndes. El plan de consulta se desarrolla en dos partes:

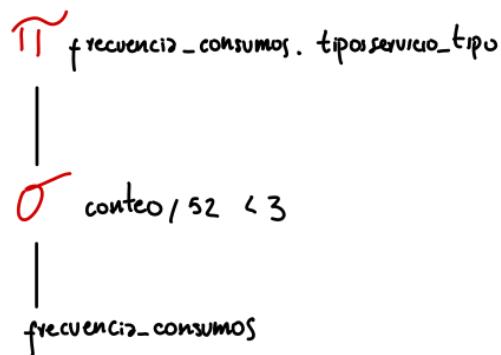
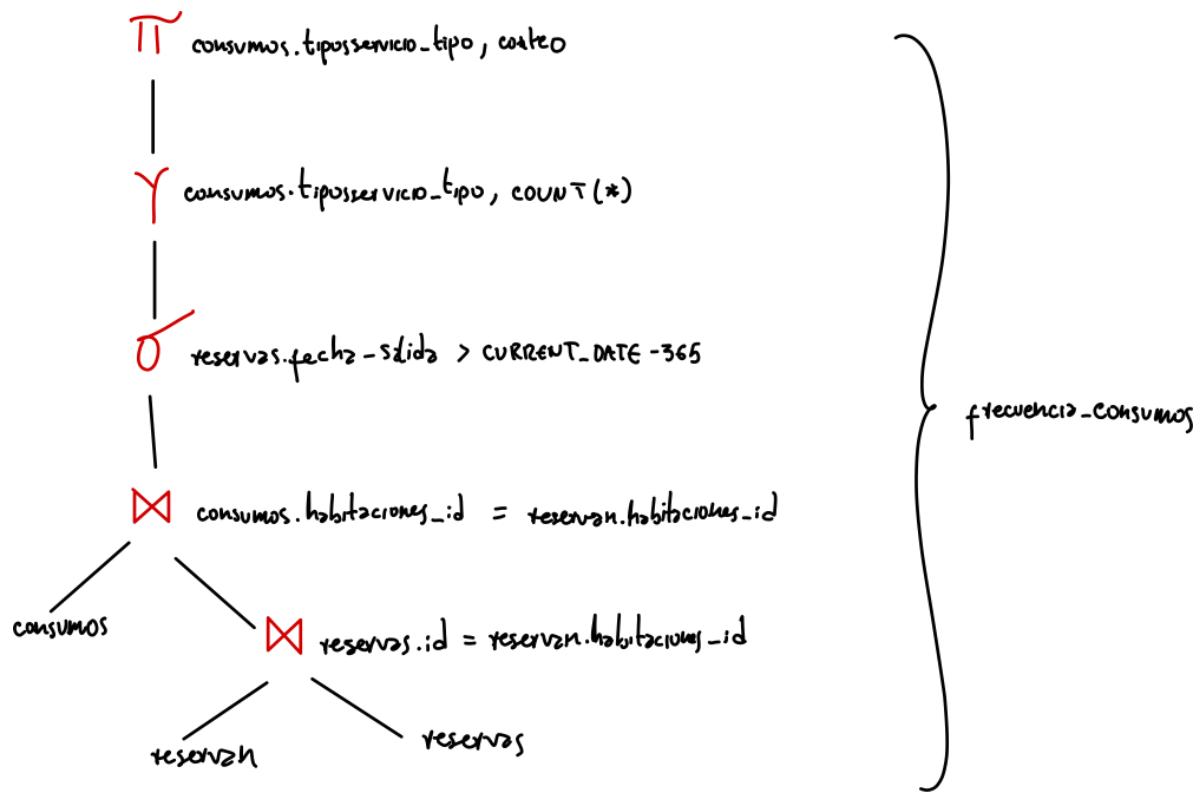
1. Frecuencia de Consumos: La primera parte calcula la frecuencia de consumo para cada tipo de servicio al unir las tablas "RESERVAS," "RESERVAN," y "CONSUMOS." Filtra los registros en función de la fecha de salida dentro del último año y agrupa los resultados por tipo de servicio. Luego, se cuenta cuántas veces se ha solicitado cada tipo de servicio en el período.
2. Selección de Servicios con Poca Demanda: En la segunda parte, la consulta selecciona los tipos de servicio cuya frecuencia semanal (conteo/52) es menor de 3, lo que indica que han sido solicitados menos de tres veces por semana. Esto identifica los servicios con poca demanda en el último año de operación.

En resumen, la consulta busca los servicios con una baja frecuencia de solicitud en el último año y los muestra como resultados. La consulta utiliza una estructura de consulta común (WITH) para calcular la frecuencia de consumo antes de filtrar los servicios con poca demanda. Esto puede ser útil para la gestión y análisis de los servicios que podrían requerir ajustes o promoción para aumentar su demanda en el hotel.

-- Sentencia SQL RFC8

```
WITH frecuencia_consumos AS (
    SELECT consumos.tiposervicio_tipo, COUNT(*) AS conteo FROM RESERVAS
    INNER JOIN RESERVAN
    ON reservas.id = reservan.habitaciones_id
    INNER JOIN CONSUMOS
    ON consumos.habitaciones_id = reservan.habitaciones_id
    WHERE reservas.fecha_salida > (SELECT CURRENT_DATE-365 from dual)
    GROUP BY consumos.tiposervicio_tipo
)
SELECT frecuencia_consumos.tiposervicio_tipo FROM frecuencia_consumos
WHERE conteo/52 < 3;
```

Realizado grupalmente:



Plan de Consulta

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	478
FILTER				
Filter Predicates				
COUNT(*)/52<3				
HASH		GROUP BY	1	478
HASH JOIN			2552	475
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
NESTED LOOPS				
STATISTICS COLLECTOR				
MERGE JOIN				
TABLE ACCESS	RESERVAS	BY INDEX ROWID		
Filter Predicates				
RESERVAS.FECHA_SALIDA > (SELECT CURRENT_DATE-365 FROM SYS.DUAL DUAL)				
INDEX	RESERVAS_PK	FULL SCAN	50001	47
FAST DUAL			1	2
JOIN			50001	27
SORT				
Access Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
Filter Predicates				
RESERVAS.ID=RESERVAN.HABITACIONES_ID				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	RANGE SCAN	1	127
Access Predicates				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
INDEX	CONSUMOS_PK	FAST FULL SCAN	51045	127

Tiempo de consulta: 0,05s

```
-- RFC8

WITH frecuencia_consumos AS (
  SELECT consumos.tiposervicio_tipo, COUNT(*) AS conteo FROM RESERVAS
  INNER JOIN RESERVAN
  ON reservas.id = reservan.habitaciones_id
  INNER JOIN CONSUMOS
  ON consumos.habitaciones_id = reservan.habitaciones_id
  WHERE reservas.fecha_salida > (SELECT CURRENT_DATE-365 from dual)
  GROUP BY consumos.tiposervicio_tipo
)
SELECT frecuencia_consumos.tiposervicio_tipo FROM frecuencia_consumos
WHERE conteo/52 < 3;
```

Resultado de la Consulta x  
 SQL | Todas las Filas Recuperadas: 0 en 0,054 segundos

### Escenarios de Prueba:

- Caso sin Resultados

Para este escenario solamente se tiene contemplado un escenario de prueba y corresponde a una tabla de tiene la columna de todos los servicios que no tienen mucha demanda dentro del hotel durante el ultimo año, esta consulta vacía muestra que el hotel no ha tenido ningún tipo de servicio que halla sido solicitado menos de 3 veces semanales

- Caso No Exitoso

Esta caso consiste en una consulta que no retorna ningnùn registro en el QUERY RESULT, este caso se da cuando en la base de datos presenta problemas en la carga de la BD, haciendo que al tener un error nos dí una consulta vacía pero se tiene igualmente las 3 columnas en el resultado.

- Requerimiento 9

La consulta RFC9 busca proporcionar información sobre los clientes que han consumido al menos una vez un servicio específico en el HotelAndes, dentro de un rango de fechas especificado. La consulta se centra en el

servicio de spa ('spa') y se aplica un filtro temporal para limitar los resultados al período del 31 de diciembre de 2022 al 31 de diciembre de 2023. El plan de consulta implica las siguientes acciones:

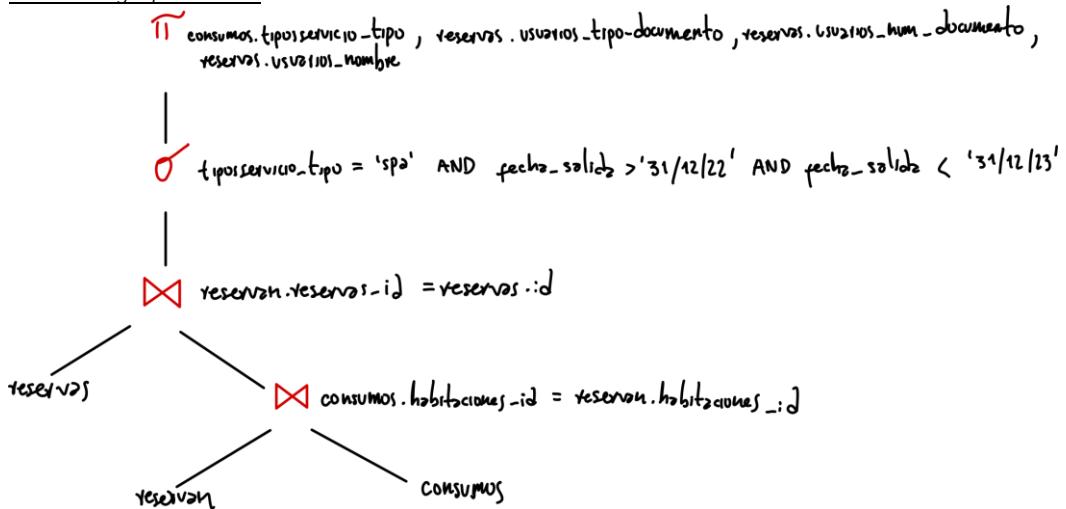
1. **Selección de Datos de Consumo:** La consulta selecciona los datos de consumo del servicio de spa ('spa') desde la tabla "CONSUMOS."
2. **Unión de Tablas:** Se realizan dos operaciones de unión para vincular los datos de consumo con la información de las reservas y los clientes. Primero, se une con la tabla "RESERVAN" utilizando la relación entre las habitaciones, y luego, se une con la tabla "RESERVAS" a través de la relación entre las reservas y las habitaciones.
3. **Filtrado por Servicio y Fecha:** La consulta aplica condiciones de filtro para seleccionar solo los registros relacionados con el servicio de spa y dentro del rango de fechas especificado.

El resultado final mostrará la información de los clientes que han consumido el servicio de spa al menos una vez durante el período establecido. La consulta ofrece la flexibilidad de permitir al usuario clasificar y ordenar los resultados según sus preferencias, lo que puede ser útil tanto para los recepcionistas como para los gerentes del hotel, según sus necesidades de análisis y seguimiento.

#### Sentencia SQL RFC9

```
-- base
SELECT consumos.tiposervicio_tipo AS servicio ,
reservas.usuarios_tipo_documento, reservas.usuarios_num_documento, reservas.usuarios_nombre
FROM CONSUMOS
INNER JOIN RESERVAN
ON consumos.habitaciones_id = reservan.habitaciones_id
INNER JOIN RESERVAS
ON reservan.reservas_id = reservas.id
WHERE consumos.tiposervicio_tipo = 'spa'
AND reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23';
```

Realizado grupalmente:



Sugerido por Oracle:

- Base sin índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS				
HASH JOIN				
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS	RESERVAS	FULL		
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	RESERVAN_PK	FAST FULL SCAN	1357	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
CONSUMOS.TIPOSERVICIO_TIPOT='spa'				

Base con índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS				
HASH JOIN				
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
VIEW	index\$_join\$_004			
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	FECHA_IDX	RANGE SCAN	12119	17
INDEX				
Access Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	USUARIOS_IDX	FAST FULL SCAN	12119	248
INDEX	RESERVAS_PK	FAST FULL SCAN	12119	59
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				

ii. Agrupado sin índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				
HASH		GROUP BY	5598	352
MERGE JOIN			5598	352
TABLE ACCESS	RESERVAS	BY INDEX ROWID	5598	351
Filter Predicates			12119	320
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	RESERVAS_PK	FULL SCAN	50001	47
SORT		JOIN	5598	31
Access Predicates				
ITEM_1=RESERVAS.ID				
Filter Predicates				
ITEM_1=RESERVAS.ID				
VIEW	SYS.VW_GBF_17	GROUP BY	5598	30
HASH			5598	30
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	28
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	25
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
CONSUMOS.TIPOSERVICIO_TIPOT='spa'				

Agrupado con índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				
HASH		GROUP BY	1357	308
FILTER			1357	308
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS			1357	307
HASH JOIN			12119	306
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
VIEW	index\$_join\$_004		12119	280
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	FECHA_IDX	RANGE SCAN	12119	17
Access Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	USUARIOS_IDX	FAST FULL SCAN	12119	248
INDEX	RESERVAS_PK	FAST FULL SCAN	12119	59
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0

iii. Ordenado sin índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1357	383
SORT		ORDER BY	1357	383
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS			1357	382
HASH JOIN			12119	381
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS	RESERVAS	FULL	12119	355
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
CONSUMOS.TIPOSERVICIO_TIPO='spa'				

Ordenado con índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1357	308
SORT		ORDER BY	1357	308
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS			1357	307
HASH JOIN			12119	306
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
VIEW	index\$_join\$_004		12119	280
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	FECHA_IDX	RANGE SCAN	12119	17
Access Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	USUARIOS_IDX	FAST FULL SCAN	12119	248
INDEX	RESERVAS_PK	FAST FULL SCAN	12119	59
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				

iv. Agrupado y Ordenado sin índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5598	352
SORT		GROUP BY	5598	352
MERGE JOIN			5598	351
TABLE ACCESS	RESERVAS	BY INDEX ROWID	12119	320
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	RESERVAS_PK	FULL SCAN	50001	47
SORT		JOIN	5598	31
Access Predicates				
ITEM_1=RESERVAS.ID				
Filter Predicates				
ITEM_1=RESERVAS.ID				
VIEW	SYS_VW_GBF_1Z	GROUP BY	5598	30
HASH			5598	30
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS			5598	28
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
CONSUMOS.TIPOSERVICIO_TIPO='spa'				

Agrupado y ordenado con índice:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SORT		GROUP BY	1357	308
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS			1357	307
HASH JOIN			12119	306
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
VIEW	index5_join\$004		12119	280
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
INDEX	FECHA_IDX	RANGE SCAN	12119	17
Access Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
INDEX	USUARIOS_IDX	FAST FULL SCAN	12119	248
INDEX	RESERVAS_PK	FAST FULL SCAN	12119	59
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				

Tiempo de consulta: 0,33s

El tiempo se toma respecto a la consulta más demorada que es agrupando y ordenando, la última.

```
-- agrupado y ordenado
SELECT consumos.tiposervicio_tipo, COUNT(consumos.tiposervicio_tipo) AS veces_usado,
reservas.usuarios_tipo_documento, reservas.usuarios_num_documento, reservas.usuarios_nombre, reservas.fecha_salida AS fecha
FROM CONSUMOS
INNER JOIN RESERVAN
ON consumos.habitaciones_id = reservan.habitaciones_id
INNER JOIN RESERVAS
ON reservan.reservas_id = reservas.id
WHERE consumos.tiposervicio_tipo = 'spa'
AND reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
GROUP BY consumos.tiposervicio_tipo, reservas.usuarios_tipo_documento,
reservas.usuarios_num_documento, reservas.usuarios_nombre, reservas.fecha_salida
ORDER BY reservas.usuarios_num_documento;
```

Resultado de la Consulta					
TIPOSERVICIO TIPO	VECES_USADO	USUARIOS TIPO DOCUMENTO	USUARIOS_NUM_DOCUMENTO	USUARIOS_NOMBRE	FECHA
spa	1306	TI	1338	Erin White	2023-12-22

#### Escenarios de Prueba:

- Caso Exitoso para tipo = tienda y entre las fechas '31/12/22' AND '31/12/23':

SERVICIO	USUARIOS_TIPO_DOCUMENTO	USUARIOS_NUM_DOCUMENTO	USUARIOS_NOMBRE
tienda	CE	9792427	Erin White
tienda	TI	25356249	Brian McLaughlin
tienda	TI	82812897	Casey Moore
tienda	CC	21981136	Catherine Farmer
tienda	CE	43958770	Jessica Scott
tienda	CE	39734877	Miranda Scott
tienda	TI	23673673	Alicia King
tienda	TI	11339818	Holly Moreno
tienda	TI	15847868	Brandy Costa MD
tienda	CC	83246991	Heather Carrillo
tienda	CE	34548992	Amy Johnson
tienda	CC	53826297	Carmen Chang
tienda	TI	97546467	Malik Wright
tienda	TI	89994729	Joshua Hill
tienda	CC	54997470	Matthew Little
tienda	CC	38206403	Erica Freeman
tienda	CC	43472486	Nicole Crosby MD
tienda	CC	29012982	Jennifer Lucas
tienda	CE	23829367	Eduardo Lang
tienda	CE	75319127	Miguel Herrera
tienda	CE	70862745	Christopher Hamilton
tienda	TI	24564924	Marc Roach
tienda	CE	14113236	Alexander Johns
tienda	CE	66636111	Danny Wright
tienda	TI	73996245	Matthew Evans
tienda	TI	4119856	Daniel Salinas
tienda	CE	33189882	Christopher Proctor
tienda	TI	82189663	James McDowell
tienda	CC	74343435	Donna Miller

- Caso Exitoso para tipo = Restaurante y entre las fechas '31/12/21' AND '31/12/23':

SERVICIO	USUARIOS_TIPO_DOCUMENTO	USUARIOS_NUM_DOCUMENTO	USUARIOS_NOMBRE
restaurante	CC	26699338	Joseph Simpson
restaurante	CE	92285511	Samantha Kim
restaurante	CE	78186290	Angela Bruce
restaurante	CE	10579688	Brittany Berry
restaurante	CC	40510056	David Martin
restaurante	TI	92833392	Frank Adams
restaurante	TI	87585431	Kayla Wilson
restaurante	TI	17915847	Kathryn Perez
restaurante	TI	18288979	Tasha Butler
restaurante	CE	38482422	Tracey Waters
restaurante	CE	55673062	Jill Nelson DDS
restaurante	CC	54624984	Nathan Howell
restaurante	CE	58736810	Tammy Waters
restaurante	CC	80438711	Tim Ballard
restaurante	CE	67988406	Trevor Pierce PhD
restaurante	TI	79886956	Jason White
restaurante	CE	84345434	Jason Hamilton
restaurante	CE	46035947	Robert Richardson
restaurante	CC	21695398	Sean Delgado
restaurante	CC	27936370	Grant Cobb
restaurante	CE	82985164	Kathleen Williams
restaurante	TI	27870650	Lori Hill
restaurante	CC	41583159	Erin Walker
restaurante	CE	88664628	Tina Roberts
restaurante	CE	75823086	Pamela Carter
restaurante	CE	12776451	Jennifer Oliver
restaurante	TI	95201435	Matthew Maldonado

- Caso No Exitoso para tipo = gym y entre las fechas '31/12/21' AND '31/12/23':

				SQL	All Rows Fetched: 0 in 0.123 seconds
				SERVIC...	USUAR...

En este caso, no se evidencia ningún registro debido a que tipo = gym no se encuentra en nuestra base de datos.

- Caso No Exitoso para tipo = restaurante y entre las fechas '31/12/23' AND '31/12/23':

				SQL	All Rows Fetched: 0 in 0.069 seconds
				SERVIC...	USUAR...

En este caso, evidenciamos que tampoco devuelve ningún registro debido a que a pesar de que tipo = restaurante si existe, el rango de fechas está constituido únicamente por un día por lo tanto en ese día el registro queda vacío.

- Requerimiento 10

La consulta RFC10 tiene como objetivo proporcionar información sobre los clientes que no han consumido un servicio específico en el HotelAndes durante un período definido, y ofrece opciones de clasificación y ordenamiento. El plan de consulta se divide en varias partes:

1. **Consulta Base:** La consulta base selecciona toda la información de los clientes desde la tabla "INFORMACIONCLIENTES" y establece un filtro para excluir a los clientes que han consumido el servicio de spa ('spa') dentro del rango de fechas especificado.
2. **Consulta Agrupada:** En la siguiente parte, la consulta agrupa los resultados por tipo de documento de cliente y cuenta cuántos clientes cumplen con el criterio de no haber consumido el servicio de spa en el período. Esto permite clasificar a los clientes por la cantidad de veces que no han consumido el servicio.
3. **Consulta Ordenada:** Luego, se realiza una consulta similar a la consulta base, pero se ordena la información de los clientes por nombre en orden ascendente. Esto permite clasificar a los clientes por nombre en orden alfabético.
4. **Consulta Agrupada y Ordenada:** Finalmente, se combina la agrupación de clientes por tipo de documento con el ordenamiento de la cantidad de veces que no han consumido el servicio. Esto clasifica a los clientes por la cantidad de veces que no han consumido el servicio de spa y, dentro de cada grupo, los ordena alfabéticamente por nombre.

La consulta brinda flexibilidad al usuario, ya que permite seleccionar entre diferentes opciones de clasificación y ordenamiento de los resultados, lo que puede ser útil tanto para los recepcionistas como para los gerentes del hotel, según sus necesidades de análisis y seguimiento de clientes que no han consumido un servicio específico.

### Sentencias SQL RFC10

```
-- base
SELECT * FROM INFORMACIONCLIENTES
WHERE (infoRmacionclientes.tipo_documento, informacionclientes.num_documento) not in (
```

```

SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento
FROM CONSUMOS
INNER JOIN RESERVAN
ON consumos.habitaciones_id = reservan.habitaciones_id
INNER JOIN RESERVAS
ON reservan.reservas_id = reservas.id
WHERE consumos.tiposervicio_tipo = 'spa'
AND reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
);

-- agrupado
SELECT informacionclientes.tipo_documento, COUNT(*) AS cantidad
FROM INFORMACIONCLIENTES
WHERE (infoRmacionclientes.tipo_documento, informacionclientes.num_documento) not in (
    SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento
    FROM CONSUMOS
    INNER JOIN RESERVAN
    ON consumos.habitaciones_id = reservan.habitaciones_id
    INNER JOIN RESERVAS
    ON reservan.reservas_id = reservas.id
    WHERE consumos.tiposervicio_tipo = 'spa'
    AND reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
)
GROUP BY informacionclientes.tipo_documento;

-- ordenado

SELECT * FROM INFORMACIONCLIENTES
WHERE (infoRmacionclientes.tipo_documento, informacionclientes.num_documento) not in (
    SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento
    FROM CONSUMOS
    INNER JOIN RESERVAN
    ON consumos.habitaciones_id = reservan.habitaciones_id
    INNER JOIN RESERVAS
    ON reservan.reservas_id = reservas.id
    WHERE consumos.tiposervicio_tipo = 'spa'
    AND reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
)
ORDER BY informacionclientes.nombre;

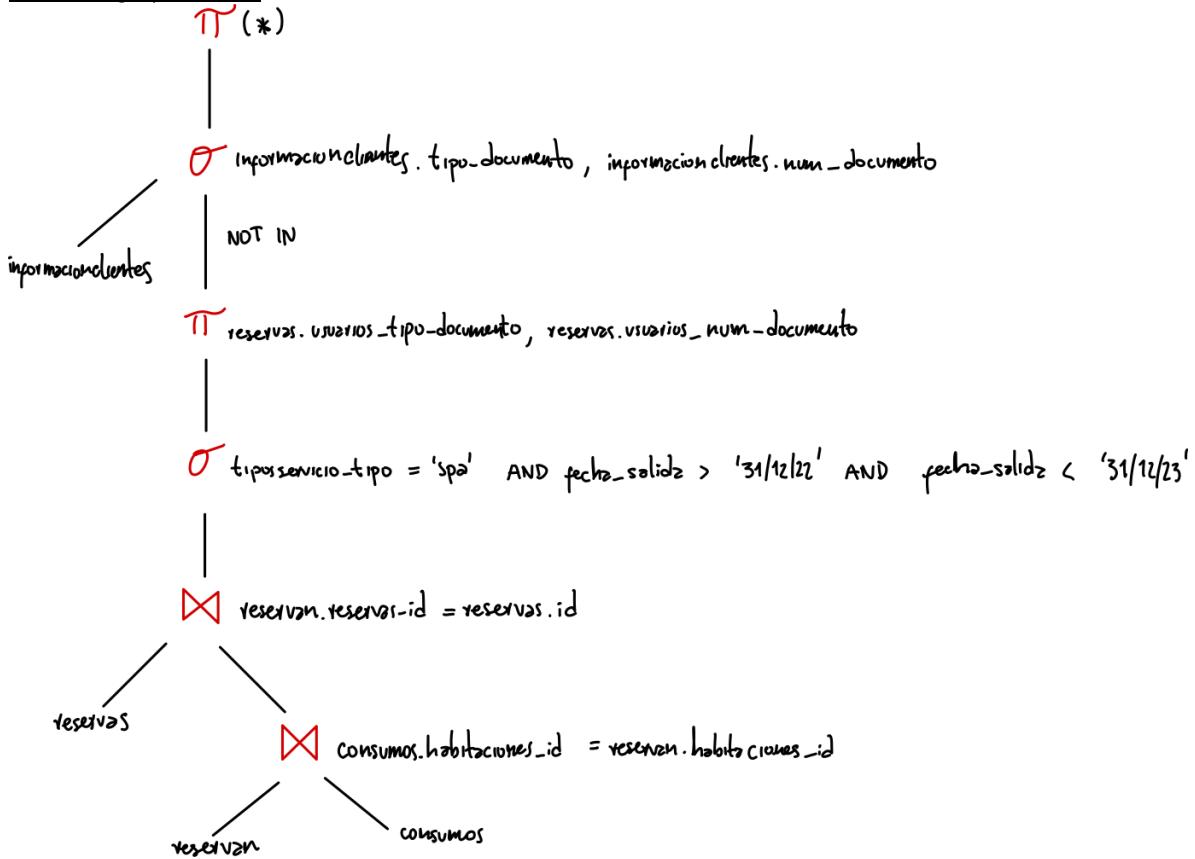
-- agrupado y ordenado

SELECT informacionclientes.tipo_documento, COUNT(*) AS cantidad
FROM INFORMACIONCLIENTES
WHERE (infoRmacionclientes.tipo_documento, informacionclientes.num_documento) not in (
    SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento
    FROM CONSUMOS
    INNER JOIN RESERVAN
    ON consumos.habitaciones_id = reservan.habitaciones_id
    INNER JOIN RESERVAS
    ON reservan.reservas_id = reservas.id
    WHERE consumos.tiposervicio_tipo = 'spa'
    AND reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
)

```

```
)
GROUP BY informacionclientes.tipo_documento
ORDER BY cantidad;
```

Realizado grupalmente:



Sugerido por Oracle:

- I. Base

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			162374	620
HASH JOIN		RIGHT ANTI	162374	620
Access Predicates				
AND				
INFORMACIONCLIENTES.TIPO_DOCUMENTO=USUARIOS_TIPO_DOCUMENTO				
INFORMACIONCLIENTES.NUM_DOCUMENTO=USUARIOS_NUM_DOCUMENTO				
VIEW	SYS.VW_NSO_1		1283	382
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS		SEMI	1283	382
HASH JOIN			12119	381
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS	RESERVAS	FULL	12119	356
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
OR				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CC'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CE'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CIF'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='TI'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='pasaporte'				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	593	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
CONSUMOS.TIPOSERVICIO TIPO='sda'				

## II. Agrupado

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	487
HASH		GROUP BY	3	487
HASH JOIN		RIGHT ANTI	162374	482
Access Predicates				
AND				
INFORMACIONCLIENTES.TIPO_DOCUMENTO=USUARIOS_TIPO_DOCUMENTO				
INFORMACIONCLIENTES.NUM_DOCUMENTO=USUARIOS_NUM_DOCUMENTO				
VIEW	SYS.VW_NSO_1		1283	382
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS		SEMI	1283	382
HASH JOIN			12119	381
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS	RESERVAS	FULL	12119	356
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
OR				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CC'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CE'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CIF'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='TI'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='pasaporte'				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	593	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				

## III. Ordenado

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			162374	2804
SORT		ORDER BY	162374	2804
HASH JOIN		RIGHT ANTI	162374	620
Access Predicates				
AND				
INFORMACIONCLIENTES.TIPO_DOCUMENTO=USUARIOS_TIPO_DOCUMENTO				
INFORMACIONCLIENTES.NUM_DOCUMENTO=USUARIOS_NUM_DOCUMENTO				
VIEW	SYS.VW_NSO_1		1283	382
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS		SEMI	1283	382
HASH JOIN			12119	381
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS RESERVAS		FULL		
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
OR				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CC'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CE'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CIF'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='TI'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='pasaporte'				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	593	0
Access Predicates				
AND				
CONSUMOS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				

#### IV. Agrupado y Ordenado

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	493
SORT		ORDER BY	3	493
HASH		GROUP BY	3	493
HASH JOIN		RIGHT ANTI	162374	482
Access Predicates				
AND				
INFORMACIONCLIENTES.TIPO_DOCUMENTO=USUARIOS_TIPO_DOCUMENTO				
INFORMACIONCLIENTES.NUM_DOCUMENTO=USUARIOS_NUM_DOCUMENTO				
VIEW	SYS.VW_NSO_1		1283	382
FILTER				
Filter Predicates				
TO_DATE('31/12/23')>TO_DATE('31/12/22')				
NESTED LOOPS		SEMI	1283	382
HASH JOIN			12119	381
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS RESERVAS		FULL		
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA>'31/12/22'				
RESERVAS.FECHA_SALIDA<'31/12/23'				
OR				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CC'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CE'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='CIF'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='TI'				
RESERVAS.USUARIOS_TIPO_DOCUMENTO='pasaporte'				
INDEX	RESERVAN_PK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	UNIQUE SCAN	593	0
Access Predicates				
AND				

Tiempo de consulta: 0,13s

El tiempo se toma respecto a la consulta más demorada que agrupa y ordena

```
-- agrupado y ordenado

SELECT informacionclientes.tipo_documento, COUNT(*) AS cantidad
FROM INFORMACIONCLIENTES
WHERE (infoRmacionclientes.tipo_documento, informacionclientes.num_documento) not in (
    SELECT reservas.usuarios_tipo_documento, reservas.usuarios_num_documento
    FROM CONSUMOS
    INNER JOIN RESERVAN
    ON consumos.habitaciones_id = reservan.habitaciones_id
    INNER JOIN RESERVAS
    ON reservan.reservas_id = reservas.id
    WHERE consumos.tiposervicio_tipo = 'spa'
    AND reservas.fecha_salida > '31/12/22' AND reservas.fecha_salida < '31/12/23'
)
GROUP BY informacionclientes.tipo_documento
ORDER BY cantidad;
```

#### Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 3 en 0,131 segundos

TIPO_DOCUMENTO	CANTIDAD
CE	55973
CC	56263
TI	56576

#### Escenarios de Prueba:

- Caso Exitoso para tipo = restaurante y entre las fechas '31/12/22' AND '31/12/23':

TIPO_DOCUMENTO	NUM_DOCUMENTO	NOMBRE	CORREO
138... TI	33281069	Rodney Barber	kimberly12@example.net
138... CE	70686745	Christine Pollard	rthomas@example.net
138... CC	50913962	Kevin Marks	alexisrobinson@example.org
138... CC	11803609	Mrs. Nicole Le	phoover@example.org
138... CC	10276691	Brittany Jordan	jonesrobert@example.org
138... CE	76696308	Kelly Ramos MD	nelsondarrell@example.com
138... TI	87489652	Amber Scott	michaelelliott@example.org
138... TI	82239601	James Harris	thomasbrown@example.org
138... CC	57151011	Patrick Hernandez	kimberlybrooks@example.net
138... TI	91964891	Arthur Davis	emilyhenderson@example.org
138... TI	41165983	Angela Rivera	hwood@example.com
138... CC	12283169	Manuel Scott	caguilar@example.net
138... CC	60658940	Scott Short	shawn51@example.com
138... CC	65905180	Mary Grant	carolwhite@example.org
138... TI	21854930	Jacqueline Keller	matthewwright@example.org
138... CC	43711382	Sean David	williamtate@example.com
138... TI	80485861	Bailey Watson	leeashley@example.com
138... CE	22595712	Valerie Aguilar	ericwilliams@example.com
138... CC	10944057	Sylvia Perez	david20@example.com
138... CC	11833910	Tonya Conner	sherrysmith@example.net
138... CC	77598807	Emily Roy	christinesimmons@example.com
138... CC	88163428	Patricia Simpson	jessica84@example.org
138... CE	68131491	Chris Oneal	hperez@example.com
138... TI	50758741	Nina Hanson	brian26@example.com
138... CC	37336288	Jason Dillon	torrespamela@example.net
138... CE	74887703	Christopher Brown	beardjessica@example.org

- Caso Exitoso para tipo = gym y entre las fechas '31/12/22' AND '31/12/23':

En este evidenciaremos que aparecerá el registro de todos los clientes debido a que gym no se encuentra como un tipo de servicio, pero estamos buscando aquellos que nunca han consumido ese tipo de servicio para esa fecha.

- Requerimiento 11

La consulta RFC11 tiene como objetivo proporcionar información semanal al gerente general de HotelAndes sobre el servicio más consumido, el servicio menos consumido, las habitaciones más solicitadas y las habitaciones menos solicitadas. La consulta se divide en varias partes:

1. **Cálculo de Consumos por Semana:** La primera parte de la consulta crea una tabla temporal que calcula la cantidad de consumos para cada tipo de servicio agrupados por semana. Esto se basa en las reservas y consumos registrados en la base de datos.
2. **Cálculo de Reservas por Semana:** La siguiente parte de la consulta calcula la cantidad de reservas de habitaciones por semana y almacena estos datos en otra tabla temporal.
3. **Selección de Servicios Más y Menos Consumidos:** La consulta luego compara los datos de consumo por semana para determinar el servicio más consumido y el servicio menos consumido en cada semana. Esto se logra mediante la aplicación de funciones de ventana (ROW\_NUMBER) que clasifican los servicios en función de su cantidad de consumos.
4. **Selección de Habitaciones Más y Menos Solicitadas:** De manera similar, la consulta compara los datos de reservas por semana para identificar las habitaciones más solicitadas y las menos solicitadas en cada semana.
5. **Resultados Finales:** Finalmente, la consulta reúne los resultados, mostrando la semana, el servicio más consumido, el servicio menos consumido, la habitación más reservada y la habitación menos reservada. Los resultados se presentan en un orden específico según la semana.

En resumen, esta consulta proporciona al gerente general información detallada y clasificada sobre la operación semanal del hotel, permitiéndole tomar decisiones basadas en el consumo de servicios y la demanda de habitaciones.

### Sentencia SQL RFC11

```
WITH ConsumosPorSemana AS (
    SELECT TO_CHAR(r.fecha_salida, 'IW') AS semana, c.tiposervicio_tipo, COUNT(*) AS cantidad_consumos
    FROM RESERVAS r
    INNER JOIN RESERVAN rv ON r.id = rv.reservas_id
    INNER JOIN CONSUMOS c ON rv.habitaciones_id = c.habitaciones_id
    GROUP BY TO_CHAR(r.fecha_salida, 'IW'), c.tiposervicio_tipo
),
ReservasPorSemana AS (
    SELECT TO_CHAR(r.fecha_salida, 'IW') AS semana, rv.habitaciones_id, COUNT(*) AS cantidad_reservas
    FROM RESERVAS r
    INNER JOIN RESERVAN rv ON r.id = rv.reservas_id
    GROUP BY TO_CHAR(r.fecha_salida, 'IW'), rv.habitaciones_id
)
SELECT
semana,
(SELECT tiposervicio_tipo
    FROM (
```

```

SELECT c.*,
ROW_NUMBER() OVER (PARTITION BY semana, cantidad_consumos ORDER BY semana) AS rn
FROM ConsumosPorSemana c
WHERE cantidad_consumos = (SELECT MAX(cantidad_consumos) FROM ConsumosPorSemana c2 ) and
c.semana = s.semana
)
WHERE rn = 1) AS servicio_mas_consumido,
(SELECT tiposervicio_tipo
FROM (
SELECT c.*,
ROW_NUMBER() OVER (PARTITION BY semana, cantidad_consumos ORDER BY semana) AS rn
FROM ConsumosPorSemana c
WHERE cantidad_consumos = (SELECT MIN(cantidad_consumos) FROM ConsumosPorSemana c2 ) and
c.semana = s.semana
)
WHERE rn = 1) AS servicio_menos_consumido,
(SELECT habitaciones_id
FROM (
SELECT r.*,
ROW_NUMBER() OVER (PARTITION BY semana, cantidad_reservas ORDER BY semana) AS rn
FROM ReservasPorSemana r
WHERE cantidad_reservas = (SELECT MAX(cantidad_reservas) FROM ReservasPorSemana r2 ) and r.semana
= s.semana
)
WHERE rn = 1) AS habitacion_mas_reservada,
(SELECT habitaciones_id
FROM (
SELECT r.*,
ROW_NUMBER() OVER (PARTITION BY semana, cantidad_reservas ORDER BY semana) AS rn
FROM ReservasPorSemana r
WHERE cantidad_reservas = (SELECT MIN(cantidad_reservas) FROM ReservasPorSemana r2 ) and r.semana
= s.semana
)
WHERE rn = 1) AS habitacion_menos_reservada
FROM (SELECT DISTINCT TO_CHAR(fecha_salida, 'IW') AS semana FROM RESERVAS) s
ORDER BY semana;

```

*Realizado grupalmente:*

$\Pi_{\text{reservas.fecha\_salida}, \text{consumos.tiposervicio}, \text{cantidad\_consumo}}$   
 |  
 $\Sigma_{\text{TO-CHAR}(\text{reservas.fecha\_salida}, 'IW')}, \text{consumos.tiposervicio}; \text{COUNT}(\text{a})$   
 |  
 $\bowtie_{\text{reservas.id} = \text{reservan.reservas\_id}}$   
 reserves reservan  
 reserves reservan habitaciones\_id = consumos.habitaciones\_id  
 reservan consumos

$\Pi_{\text{semana}, \text{reservan.habitaciones\_id}, \text{cantidad\_reservas}}$   
 |  
 $\Sigma_{\text{TO-CHAR}(r.\text{fecha\_salida}, 'IW')}, \text{reservan.habitaciones\_id}, \text{COUNT}(\text{a})$   
 |  
 $\bowtie_{\text{reservas.id} = \text{reservan.reservas\_id}}$   
 reserves reservan

Plan De Consulta

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1443	3227
VIEW			9184	325
Filter Predicates				
RN=1				
WINDOW		SORT PUSHED RANK	9184	325
Filter Predicates				
ROW_NUMBER() OVER (PARTITION BY SEMANA,CANTIDAD_CONSUMOS ORDER BY NULL )<=1			9184	14
VIEW				
Filter Predicates				
AND				
C_SEMANA=:B1				
CANTIDAD_CONSUMOS=(SELECT MAX(CANTIDAD_CONSUMOS) FROM (SELECT /*+ CACHE (T1) */ C0 SEMANA,C1 TIPOSERVICIO_TIP0,C2 CANTIDAD_CONSUMOS FROM SYS.SYS_TEMP_0FD9D7251_CF93870)		FULL	9184	14
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7251_CF93870	AGGREGATE	1	
SORT			9184	14
VIEW	SYS.SYS_TEMP_0FD9D7251_CF93870	FULL	9184	14
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7251_CF93870	FULL	9184	325
VIEW				
Filter Predicates				
RN=1				
WINDOW		SORT PUSHED RANK	9184	325
Filter Predicates				
ROW_NUMBER() OVER (PARTITION BY SEMANA,CANTIDAD_CONSUMOS ORDER BY NULL )<=1			9184	14
VIEW				
Filter Predicates				
AND				
C_SEMANA=:B1				
CANTIDAD_CONSUMOS=(SELECT MIN(CANTIDAD_CONSUMOS) FROM (SELECT /*+ CACHE (T1) */ C0 SEMANA,C1 TIPOSERVICIO_TIP0,C2 CANTIDAD_CONSUMOS FROM SYS.SYS_TEMP_0FD9D7251_CF93870)		FULL	9184	14
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7251_CF93870	AGGREGATE	1	
SORT			9184	14
VIEW	SYS.SYS_TEMP_0FD9D7251_CF93870	FULL	9184	14
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7251_CF93870	FULL	50001	491
VIEW				
Filter Predicates				
RN=1				
WINDOW		SORT PUSHED RANK	50001	491
Filter Predicates				
ROW_NUMBER() OVER (PARTITION BY SEMANA,CANTIDAD_RESERVAS ORDER BY NULL )<=1			50001	44
VIEW				
Filter Predicates				
AND				
R_SEMANA=:B1				
CANTIDAD_RESERVAS=(SELECT MAX(CANTIDAD_RESERVAS) FROM (SELECT /*+ CACHE (T1) */ C0 SEMANA,C1 HABITACIONES_ID,C2 CANTIDAD_RESERVAS FROM SYS.SYS_TEMP_0FD9D7252_CF93870)		FULL	50001	44
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7252_CF93870	AGGREGATE	1	
SORT			50001	44
VIEW	SYS.SYS_TEMP_0FD9D7252_CF93870	FULL	50001	44
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7252_CF93870	FULL	50001	491
VIEW				
Filter Predicates				
RN=1				
WINDOW		SORT PUSHED RANK	50001	491
Filter Predicates				
ROW_NUMBER() OVER (PARTITION BY SEMANA,CANTIDAD_RESERVAS ORDER BY NULL )<=1			50001	44
VIEW				
Filter Predicates				
AND				
R_SEMANA=:B1				
CANTIDAD_RESERVAS=(SELECT MIN(CANTIDAD_RESERVAS) FROM (SELECT /*+ CACHE (T1) */ C0 SEMANA,C1 HABITACIONES_ID,C2 CANTIDAD_RESERVAS FROM SYS.SYS_TEMP_0FD9D7252_CF93870)		FULL	50001	44
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7252_CF93870	AGGREGATE	1	
SORT			50001	44
VIEW	SYS.SYS_TEMP_0FD9D7252_CF93870	FULL	50001	44
TABLE ACCESS	SYS.SYS_TEMP_0FD9D7252_CF93870	FULL	50001	491
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT				
HASH				
HASH JOIN				
Access Predicates				
R.ID=RV.RESERVAS_ID				
TABLE ACCESS	RESERVAS	FULL	50001	355
HASH JOIN				
Access Predicates				
RV.HABITACIONES_ID=C.HABITACIONES_ID				
INDEX	RESERVA_NK	FAST FULL SCAN	50001	25
INDEX	CONSUMOS_PK	FAST FULL SCAN	51045	127
LOAD AS SELECT	SYS_TEMP_0FD9D7252_CF93870	(CURSOR DURATION MEMORY)		
HASH		GROUP BY	50001	726
HASH JOIN				
Access Predicates				
R.ID=RV.RESERVAS_ID				
INDEX	RESERVA_NK	FAST FULL SCAN	50001	25
INDEX	RESERVAS	FULL	50001	355
SORT				
HASH		ORDER BY	1443	1991
VIEW				
HASH		UNIQUE	1443	357
TABLE ACCESS	RESERVAS	FULL	50001	355

Tiempo de consulta: 0,48s

No se muestra la consulta ya que es muy extensa y se encuentra en la parte anterior

Todas las Filas Recuperadas: 53 en 0,48 segundos

#### Escenarios de Prueba:

- o Escenario Existente

Para este escenario de prueba, se espera obtener 53 registros que tengan 5 columnas, las cuales corresponden a SEMANA, SERVICIO\_MAS\_CONSUMIDO, SERVICIO\_MENOS\_CONSUMIDO, HABITACION\_MAS\_RESERVADA, HABITACION\_MENOS\_RESERVADA , la cual muestra para cada semana del año (sábado a sábado), el servicio más consumido, el servicio menos consumido, las habitaciones más solicitadas y las habitaciones menos solicitadas del hotel.

- o Escenario No Existente

Este caso consiste en una consulta que no retorna ningún registro en el QUERY RESULT, este caso se da cuando en la base de datos presenta problemas en la carga de la BD, haciendo que al tener un error nos dé una consulta vacía pero se tiene igualmente las 5 columnas en el resultado.

- Requerimiento 12

La consulta RFC12 tiene como objetivo identificar y proporcionar información sobre los "clientes excelentes" del HotelAndes, definidos como aquellos que cumplen con uno de tres criterios: realizar estancias al menos una vez por trimestre, consumir servicios costosos (con un precio mayor a \$300,000), o consumir servicios de SPA o salones de reuniones con una duración mayor a 4 horas en cada estancia. El plan de consulta se desarrolla en varias etapas:

1. **Cálculo de Estancias por Trimestre:** La primera parte de la consulta crea una tabla temporal que calcula cuántas veces cada cliente ha realizado estancias en el hotel durante cada trimestre.
2. **Selección de Consumos Costosos:** Luego, se identifican los clientes que han consumido servicios costosos, definidos por un precio superior a \$300,000. Estos clientes se seleccionan en una tabla temporal separada.
3. **Selección de Reservas con Duración Mayor a 4 Horas:** La consulta también identifica los clientes que han reservado servicios de SPA o salones de reuniones con una duración de al menos 4 horas durante sus estancias.
4. **Resultados Finales:** Finalmente, la consulta combina los resultados de las tres categorías de clientes excelentes: aquellos que cumplen con los criterios de estancias por trimestre, consumos costosos y reservas con duración mayor a 4 horas. Los resultados incluyen información sobre el nombre del cliente, su tipo y número de documento, su dirección de correo electrónico y el motivo por el cual se consideran clientes excelentes (por ejemplo, "Estancias por trimestre" o "Consumos costosos").

En resumen, esta consulta ofrece al gerente general del hotel información detallada sobre los clientes que cumplen con los criterios de excelencia definidos, lo que les permite tomar decisiones basadas en el rendimiento y el valor de los clientes en el hotel.

#### Sentencia SQL RFC12

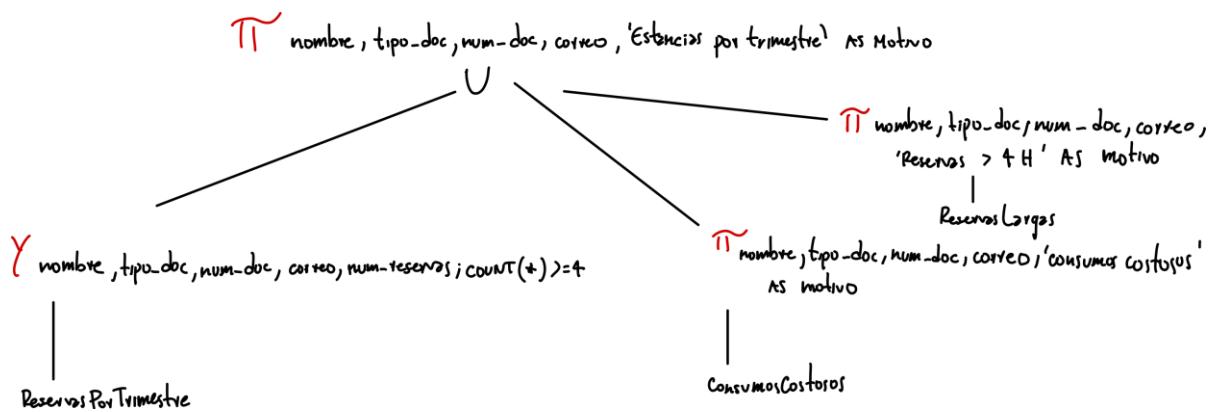
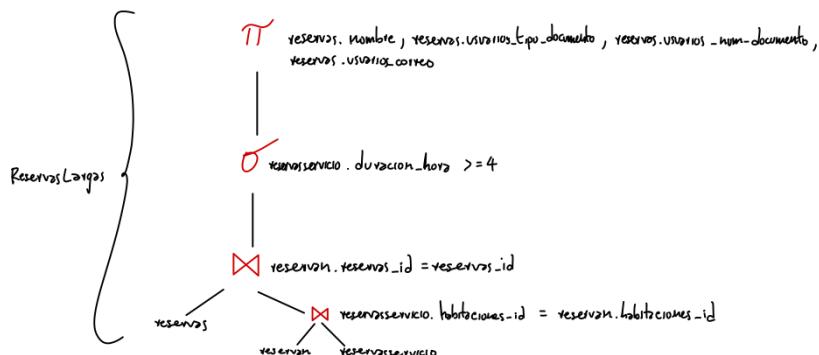
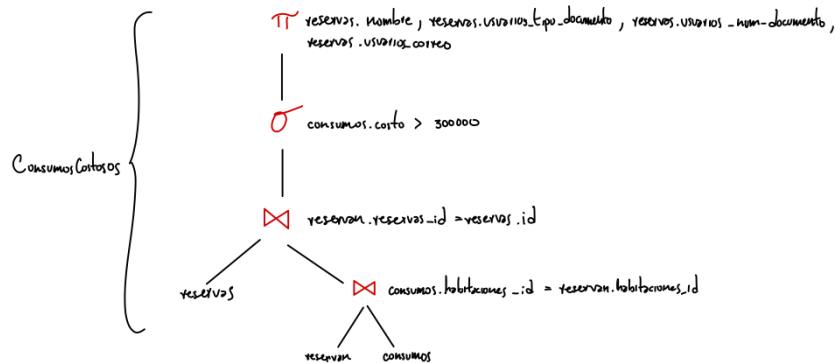
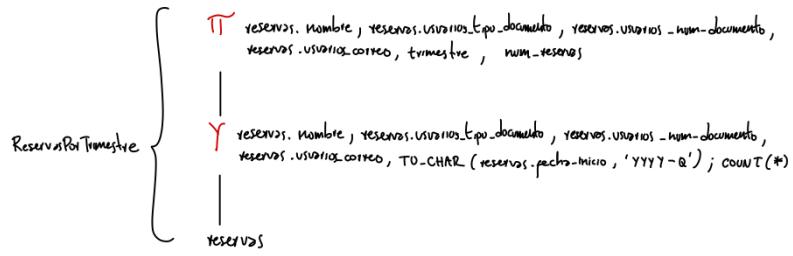
```
WITH ReservasPorTrimestre AS (
    SELECT
```

```

r.usuarios_nombre AS nombre, r.usuarios_tipo_documento AS tipo_doc,
r.usuarios_num_documento AS num_doc, r.usuarios_correo AS correo,
TO_CHAR(r.fecha_inicio, 'YYYY-Q') AS trimestre,
COUNT(*) AS num_reservas
FROM RESERVAS r
GROUP BY r.usuarios_nombre, r.usuarios_tipo_documento,
r.usuarios_num_documento, r.usuarios_correo, TO_CHAR(r.fecha_inicio, 'YYYY-Q')
),
ConsumosCostosos AS (
SELECT r.usuarios_nombre AS nombre, r.usuarios_tipo_documento AS tipo_doc,
r.usuarios_num_documento AS num_doc, r.usuarios_correo AS correo
FROM CONSUMOS c
INNER JOIN RESERVAN
ON c.habitaciones_id = reservan.habitaciones_id
INNER JOIN RESERVAS r
ON reservan.reservas_id = r.id
WHERE c.costo > 300000
GROUP BY r.usuarios_nombre, r.usuarios_tipo_documento,
r.usuarios_num_documento, r.usuarios_correo
),
ReservasLargas AS (
SELECT r.usuarios_nombre AS nombre, r.usuarios_tipo_documento AS tipo_doc,
r.usuarios_num_documento AS num_doc, r.usuarios_correo AS correo
FROM RESERVASSERVICIO rs
INNER JOIN RESERVAN
ON rs.habitaciones_id = reservan.habitaciones_id
INNER JOIN RESERVAS r
ON reservan.reservas_id = r.id
WHERE rs.duracion_hora >= 4
GROUP BY r.usuarios_nombre, r.usuarios_tipo_documento,
r.usuarios_num_documento, r.usuarios_correo
)
SELECT c.nombre, c.tipo_doc, c.num_doc, c.correo, 'Estancias por trimestre' AS motivo
FROM ReservasPorTrimestre c
GROUP BY c.nombre, c.tipo_doc, c.num_doc, c.correo, c.num_reservas
HAVING count(*) >= 4
UNION ALL
SELECT cc.nombre, cc.tipo_doc, cc.num_doc, cc.correo, 'Consumos costosos' AS motivo
FROM ConsumosCostosos cc
UNION ALL
SELECT rl.nombre, rl.tipo_doc, rl.num_doc, rl.correo, 'Reservas mayores a 4H' AS motivo
FROM ReservasLargas rl;

```

Realizado grupalmente:



## Plan de Consulta

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	10
U UNION-ALL				
FILTER				
Filter Predicates				
COUNT(*)>=4				
HASH				
VIEW				
HASH				
TABLE ACCESS				
HASH				
X NESTED LOOPS				
X NESTED LOOPS				
X INDEX				
TABLE ACCESS				
Filter Predicates				
C.COSTO>300000				
INDEX				
Access Predicates				
C.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
INDEX				
Access Predicates				
RESERVAN.RESERVAS_ID=R.ID				
TABLE ACCESS				
HASH				
X NESTED LOOPS				
X NESTED LOOPS				
X TABLE ACCESS				
Filter Predicates				
RS.DURACION_HORA>=4				
INDEX				
Access Predicates				
RS.HABITACIONES_ID=RESERVAN.HABITACIONES_ID				
INDEX				
Access Predicates				
RESERVAN.RESERVAS_ID=R.ID				
TABLE ACCESS				
RESERVAS		BY INDEX ROWID	1	0
		GROUP BY	1	4
			1	3
		FULL	1	2
		GROUP BY	1	1
			1	0
		SEMI	1	0
		FULL SCAN	1	0
		BY INDEX ROWID BATCHED	1	0
	RESERVAS_PK			
	CONSUMOS			
	CONSUMOS_PK	RANGE SCAN	1	0
	RESERVAS			
	RESERVAS	BY INDEX ROWID	1	0
		GROUP BY	1	5
			1	4
		FULL	1	4
	RESERVASSERVICIO	FULL	1	4
	RESERVAS			
	RESERVAS	UNIQUE SCAN	1	0
	RESERVAS			
	RESERVAS	BY INDEX ROWID	1	0

Tiempo de ejecución: 0,64s

Se han recuperado 2.600 filas en 0,637 segundos

### Escenarios de Prueba:

- Caso Exitoso

Para este escenario de prueba, se espera obtener 2600 registros que tengan 5 columnas, las cuales corresponden a NOMBRE, TIPO\_DOC, NUM\_DOC, CORREO, MOTIVO. Esta consulta retorna toda la información de los clientes excelentes, incluyendo aquella que justifica su calificación como clientes excelentes.

- Caso No Exitoso:

Este caso consiste en una consulta que no retorna ningún registro en el QUERY RESULT, este caso se da cuando en la base de datos presenta problemas en la carga de la BD, haciendo que al tener un error nos dé una consulta vacía pero se tiene igualmente las 5 columnas en el resultado.

## 4. Documentación del proyecto java

El proyecto Java con lo corregido de la entrega pasada contiene los nuevos controladores llamados RFC<N>Controller, donde la N corresponde al número del requerimiento funcional, dentro de esto se encuentra el controlador necesario para cada consulta con los parámetros posibles y el manejo de los Repositorios creados para cada requerimiento. En la carpeta Static se encuentran los archivos HTML donde se implementa la interfaz y se presentan los resultados de las consultas.

Para ejecutar el proyecto se ejecuta el proyecto en spring y se abre el local host en el puerto 8080. Los requerimientos todos se encuentran al seleccionar el rol gerente el cuál se debe seleccionar para verlos, el 9 y 10 también se encuentran disponibles para el empleado. En los requerimientos que no necesitan parámetros se abre directamente el resultado de la consulta. Si es un requerimiento donde se puede ordenar y/o filtrar aparecerá en la parte de arriba de la tabla de resultados. Si el requerimiento tiene parámetros de consulta se abre primero el formulario para poder hacer este. Luego se ejecuta la consulta con los parámetros ingresados. Todas las interfaces tienen el botón volver o cancelar que lo devuelve a los parámetros al menú principal.

Las consultas presentadas en los scripts sql son las mismas ejecutadas en los repositorios. Se toma el tiempo de hacer esta operación en el repositorio y se presenta en pantalla.