TABLA DE CONTENIDOS.

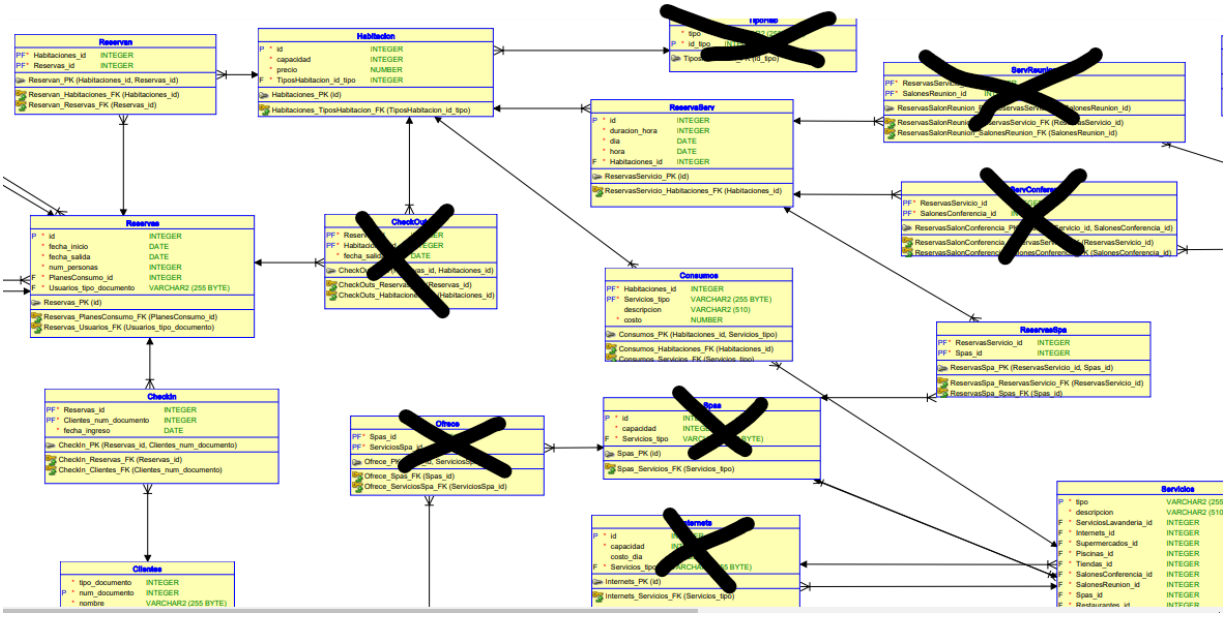
DISEÑO DE LA APLICACION 2
Índices 2
Clientes-Numero documento 2
Reservas-fecha_inicio 3
Habitación-id 3
Reservas y Consumos-Habitacion_id 3
Servicios-tipo 3
Diseño de consultas 4
RFC1 - Mostrar el dinero recolectado por servicios en cada habitación en el último año corrido. 4
RFC2 - Mostrar los 20 servicios más populares. 4
RFC3 - Mostrar el índice de ocupación de cada una de las habitaciones del hotel 5
RFC4 - Mostrar los servicios que cumplen con cierta característica 5
RFC5 - Mostrar el consumo en HotelAndes por un usuario dado, en un rango de fechas indicado. 6
RFC6 - Analizar la operación de HotelAndes 7
RFC7 - Encontrar los buenos clientes 8
RFC8 - Encontrar los servicios que no tienen mucha demanda 9
COMPARACIÓN CON PLANES DE EJECUCIÓN 14
Para RC1: 14
Para RFC2: 14
Para RFC8: 15
Para RFC12: 15
CARGA DE DATOS MASIVA 16
Organización de carpetas y archivos 16
datosAleatorios.py 16
data\cargaMasivaDatos\csv 16
data\cargaMasivaDatos\ctl 17

Alejandro Cataño
Juanita Gil
Santiago Paez

DISEÑO DE LA APLICACION

Índices

Se hará un análisis de los requerimientos funcionales de consulta para la entrega 2. Primero, se descartarán aquellas tablas que no se usan en las consultas.



En el fragmento del modelo relacional de la imagen, aquellas que no están ralladas, son las necesarias para realizar todas las consultas ya que posee la mayoría de los atributos de interés:

- Servicios: Sirve el tipo de servicio
- Cliente: Sirven todos los atributos y se conecta con los demás por el número del documento
- Reservan: Relaciona las habitaciones con las reservas
- Reservas: Sirve para conectar el cliente con las habitaciones
- CheckIn: Sirve para encontrar la fecha de ingreso la cual es fundamental para las consultas
- Habitaciones: Ayuda a relacionar los clientes con el consumo que hacen que se registra en la habitación reservada
- Consumos: Sirven todos los atributos y es el filtrador principal porque contiene muchos de los atributos que piden las consultas.
- Reservaserv: Sirven las duraciones de los servicios
- Reservaspa: Sirve para comprobar que el servicio lo haya usado un cliente

Con esto en cuenta y las consultas SQL realizadas, se mostrará el por qué **NO** deberían tener índices

Clientes-Numero documento

Desde el punto de vista de selectividad es perfecto ya que todos los números de documentos son diferentes por lo que podría facilitar la búsqueda. Sin embargo:

```
--RFC10 - CONSULTAR CONSUMO EN HOTELANDES - RFC9-V2
SELECT clientes.nombre AS nombre_cliente,clientes.num_documento AS num_documento_cliente
FROM clientes
WHERE clientes.num_documento NOT IN (
    SELECT DISTINCT reservas.usuarios_num_documento
    FROM reservas
    INNER JOIN reservan ON reservas.id = reservan.reservas_id
    INNER JOIN consumos ON reservan.habitacion_id = consumos.habitacion_id
    WHERE
        reservas.fecha_inicio BETWEEN TO_DATE(:fecha_inicial, 'YYYY-MM-DD') AND TO_DATE(:fecha_final, 'YYYY-MM-DD')
```

Como se ve en la imagen, una de las condiciones de la columna es NOT IN por lo que no se debería usar un índice en el número de documento. No se puede arriesgar la efectividad de una consulta por beneficio de las demás.

Reservas-fecha_inicio

A pesar de ser muy usado en los where, tenemos una selectividad baja ya que muchos usuarios pueden hacer una reserva el mismo día.

Habitación-id

No tiene una selectividad tan alta ya que en diferentes épocas del año se hace una reserva de esa habitación lo que hace que se repitan varios valores. Además de esto, no es tan usado en los join o where debido a que no se hacen casi condiciones con esto y las uniones se usan con el id de las habitaciones, pero desde otras tablas.

Reservas y Consumos-Habitacion_id

No puede ya que habitaciones.id de la tabla de las habitaciones no es tan usada en los where. Además, la selectividad seria baja ya que históricamente las habitaciones se repiten mucho ya que se hacen varias reservas de la misma habitación, así como los consumos quedan a nombre de este.

Todas las demás tablas no nos conviene colocar índices ya que la selectividad es baja al repetirse muchas veces el mismo valor en el atributo o no son tan utilizados en el join y el where.

Ahora se mostrar las tablas que **SI** se les agregara índice

Servicios-tipo

En cuanto a la selectividad, es demasiado alta ya que los tipos de servicios no se repite, está incluido muchas veces en el where y los join porque se clasifican muchos atributos por esta columna.

Ahora para el tipo de índice, toca tener en cuenta que tiene muchos valores distintos, y es de búsqueda exacta al saber específicamente el servicio. Por lo que podemos decir que se **creara un índice de Hash** ya que este es más efectivo que otros tipos de índices como el B+, que al permitir rangos es menos efectivo, y se enfoca en búsquedas exactas

Para las tablas generadas a partir del modelo de entidades en DataModeler:

```
-- Informe de Resumen de Oracle SQL Developer Data Modeler:
--
-- CREATE TABLE                            43
-- CREATE INDEX                             0
-- ALTER TABLE                            113
-- CREATE VIEW                              0
```

Se puede ver Oracle no creo índices, de igual manera el código generado por este fue modificado en gran parte debido a errores que se generaban al crear as tablas.

Aun con los cambios manuales de la base de datos, no se crearon índices hasta ahora.

```
CREATE INDEX idx_servicios_tipo_hash ON servicios (tipo) ;
```

Con este codigo, se crearia el indice pero al ejecutar esta línea de código, sale que ay esta creada. Esto pasa porque al ser una llave primaria, la propia base de datos le identifica un índice con el fin de acelerar consultas, ya que en el código de crear tablas no aparece

Alejandro Cataño
Juanita Gil
Santiago Paez

Diseño de consultas

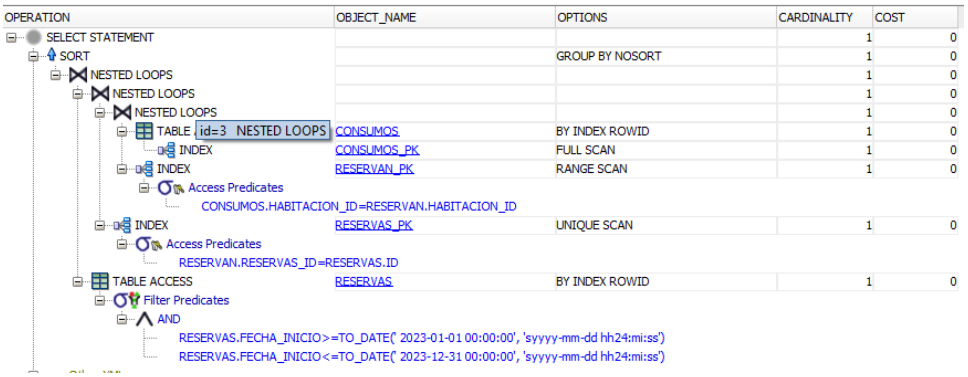
Todas las consultas poseen diferentes parámetros para que aparezcan resultados

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

```
--RFC1 - MOSTRAR EL DINERO RECOLECTADO POR SERVICIOS EN CADA HABITACIÓN EN EL ÚLTIMO AÑO CORRIDO(2023).
SELECT habitaciones.id as habitacion_id , servicios.tipo as servicio, SUM(consumos.costo)dinero_recolectado
FROM consumos
INNER JOIN habitaciones ON consumos.habitacion_id = habitaciones.id
INNER JOIN reservan ON habitaciones.id = reservan.habitacion_id
INNER JOIN servicios ON consumos.servicios_tipo = servicios.tipo
INNER JOIN reservas ON reservan.reservas_id = reservas.id
where reservas.fecha_inicio between TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD')
GROUP BY habitaciones.id, servicios.tipo;
```

El tamaño de respuesta depende de la cantidad y el tipo de servicios consumidos en el último año. A medida que existe más servicios y habitaciones relacionados, mayor será el tamaño de respuesta.

	HABITACION_ID	SERVICIO	DINERO_RECOLECTADO
1	101	bar	5
2	102	restaurante	40



RFC2 - Mostrar los 20 servicios más populares.

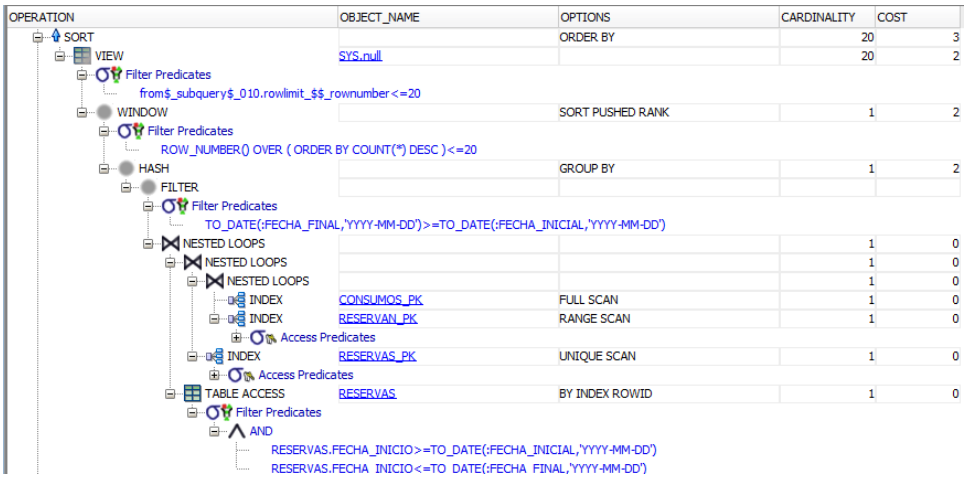
```
--RFC2 - MOSTRAR LOS 20 SERVICIOS MÁS POPULARES.
SELECT servicios.tipo AS servicio_tipo, COUNT(consumos.servicios_tipo) AS cantidad_consumos
FROM consumos
INNER JOIN habitaciones ON consumos.habitacion_id = habitaciones.id
INNER JOIN reservan ON habitaciones.id = reservan.habitacion_id
INNER JOIN servicios ON consumos.servicios_tipo = servicios.tipo
INNER JOIN reservas ON reservan.reservas_id = reservas.id
where reservas.fecha_inicio between TO_DATE(:fecha_inicial, 'YYYY-MM-DD') AND TO_DATE(:fecha_final, 'YYYY-MM-DD')
GROUP BY servicios.tipo
ORDER BY cantidad_consumos DESC
FETCH FIRST 20 ROWS ONLY;
```

La fecha inicial y final corresponden al de todo el año 2023

La distribución de datos depende de la popularidad de los servicios y cuantos consumos se hacen por servicio. A medida que aumente el número de servicios, mayor será el tamaño de respuesta

	SERVICIO_TIPO	CANTIDAD_CONSUMOS
1	restaurante	1
2	bar	1

Alejandro Cataño
Juanita Gil
Santiago Paez



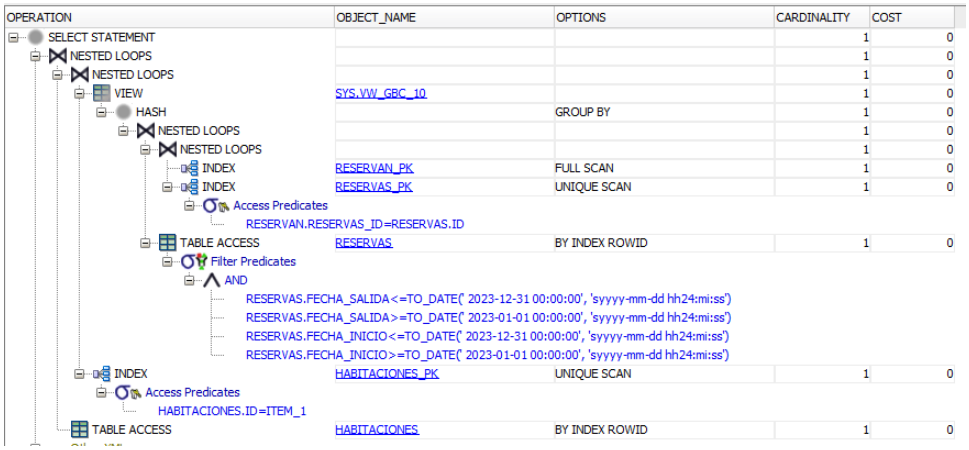
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SORT		ORDER BY	20	3
VIEW	SYS.null		20	2
Filter Predicates				
from$_subquery$_010.rowlimit_$$_rownumber<=20				
WINDOW		SORT PUSHED RANK	1	2
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC)<=20				
HASH		GROUP BY	1	2
FILTER				
Filter Predicates				
TO_DATE(:FECHA_FINAL,'YYYY-MM-DD')>=TO_DATE(:FECHA_INICIAL,'YYYY-MM-DD')				
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
INDEX	CONSUMOS_PK	FULL SCAN	1	0
INDEX	RESERVAN_PK	RANGE SCAN	1	0
Access Predicates				
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
RESERVAS.FECHA_INICIO>=TO_DATE(:FECHA_INICIAL,'YYYY-MM-DD')				
RESERVAS.FECHA_INICIO<=TO_DATE(:FECHA_FINAL,'YYYY-MM-DD')				

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

```
--RFC3 MOSTRAR EL ÍNDICE DE OCUPACIÓN DE CADA UNA DE LAS HABITACIONES DEL HOTEL
SELECT habitaciones.id AS habitacion_id,
       habitaciones.capacidad AS capacidad_habitacion,
       COUNT(reservan.reservas_id) AS total_reservas,
       SUM(reservas.fecha_salida - reservas.fecha_inicio) AS total_dias_ocupados,
       (SUM(reservas.fecha_salida - reservas.fecha_inicio) / 365) * 100 AS porcentaje_ocupacion
FROM habitaciones
LEFT JOIN reservan ON habitaciones.id = reservan.habitacion_id
LEFT JOIN reservas ON reservan.reservas_id = reservas.id
where reservas.fecha_inicio between TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD') and
reservas.fecha_salida between TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD')
GROUP BY habitaciones.id, habitaciones.capacidad;
```

La distribución depende del número de habitaciones que tenga el hotel, a mayor número de habitaciones, mayor será el tamaño de respuesta

	HABITACION_ID	CAPACIDAD_HABITACION	TOTAL_RESERVAS	TOTAL_DIAS_OCUPADOS	PORCENTAJE_OCUPACION
1	101	1	1	4	1,095890410958904109589041095890410958904109589
2	102	2	1	4	1,095890410958904109589041095890410958904109589



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
VIEW	SYS.VW_GBC_10		1	0
HASH		GROUP BY	1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
INDEX	RESERVAN_PK	FULL SCAN	1	0
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
RESERVAS.FECHA_SALIDA<=TO_DATE(' 2023-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVAS.FECHA_SALIDA>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVAS.FECHA_INICIO<=TO_DATE(' 2023-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVAS.FECHA_INICIO>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	HABITACIONES_PK	UNIQUE SCAN	1	0
Access Predicates				
HABITACIONES.ID=ITEM_1				
TABLE ACCESS	HABITACIONES	BY INDEX ROWID	1	0

RFC4 - Mostrar los servicios que cumplen con cierta característica

```
--RFC4 - MOSTRAR LOS SERVICIOS QUE CUMPLEN CON CIERTA CARACTERÍSTICA
SELECT *
FROM servicios
INNER JOIN consumos ON consumos.servicios_tipo = servicios.tipo
INNER JOIN habitaciones ON consumos.habitacion_id = habitaciones.id
INNER JOIN reservan ON habitaciones.id = reservan.habitacion_id
INNER JOIN reservas ON reservan.reservas_id = reservas.id
WHERE
  consumos.costo BETWEEN : costo_minimo AND : costo_maximo
  AND reservas.fecha_inicio BETWEEN TO_DATE(:fecha_inicial, 'YYYY-MM-DD') AND TO_DATE(:fecha_final, 'YYYY-MM-DD')
  AND servicios.tipo = : tipo_servicio;
```

Los costos son entre 1 y 40, las fechas son de todo el año 2023 y el servicio es el de bar.

En este caso, depende de las condiciones que se colocan, para este caso se agregaron rangos de costos, rangos de fechas de reservas de servicios y el tipo de servicio. El tamaño de respuesta depende de los consumos del servicio en esos rangos de horas y fechas.

	TIPO	DESCRIPCION	HABITACION_ID	SERVICIOS_TIPO	DESCRIPCION_1	COSTO	ID	CAPACIDAD	PRECIO	TIPOHAB_ID_TIPO	HABITACION_ID_1	RESERVAS_ID	ID_1	FECHA_INICIO
1	bar	Bar con bebidas y cocteles	101	bar	Cerveza	5	101	1	100	1	101	1	1	01/11/23

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	0
FILTER				
Filter Predicates				
AND				
TO_DATE(:FECHA_FINAL,'YYYY-MM-DD')>=TO_DATE(:FECHA_INICIAL,'YYYY-MM-DD')				
TO_NUMBER(:COSTO_MAXIMO)>=TO_NUMBER(:COSTO_MINIMO)				
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
SERVICIOS.TIPO=:TIPO_SERVICIO				
INDEX	RESERVAN_PK	FULL SCAN	1	0
TABLE ACCESS	HABITACIONES	BY INDEX ROWID	1	0
INDEX	HABITACIONES_PK	UNIQUE SCAN	1	0
Access Predicates				
HABITACIONES.ID=RESERVAN.HABITACION_ID				
TABLE ACCESS	CONSUMOS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
CONSUMOS.COSTO>=TO_NUMBER(:COSTO_MINIMO)				
CONSUMOS.COSTO<=TO_NUMBER(:COSTO_MAXIMO)				
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
CONSUMOS.HABITACION_ID=HABITACIONES.ID				
CONSUMOS.SERVICIOS_TIPO=:TIPO_SERVICIO				
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
RESERVAS.FECHA_INICIO>=TO_DATE(:FECHA_INICIAL,'YYYY-MM-DD')				
RESERVAS.FECHA_INICIO<=TO_DATE(:FECHA_FINAL,'YYYY-MM-DD')				

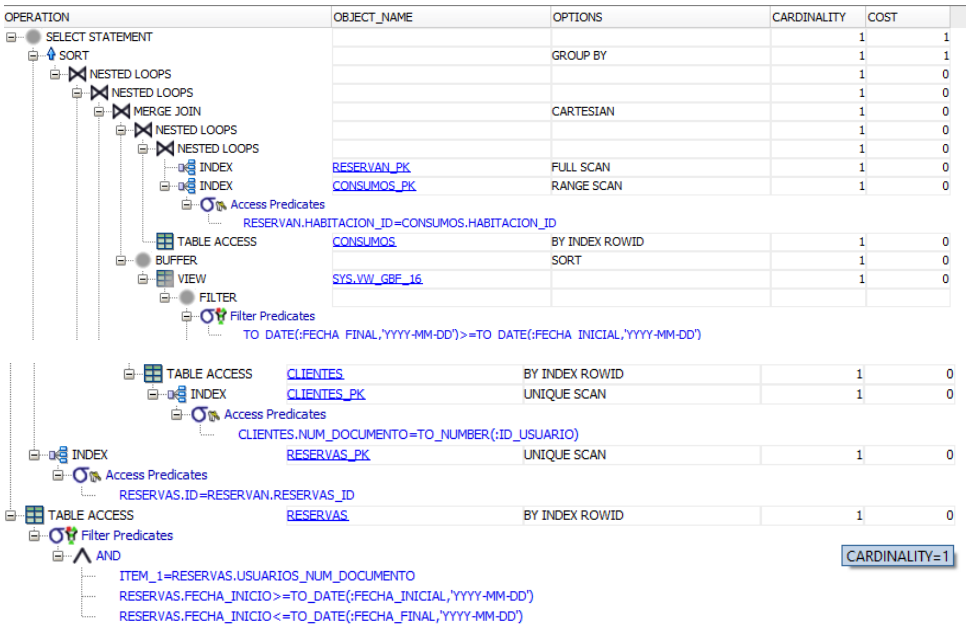
RFC5 - Mostrar el consumo en HotelAndes por un usuario dado, en un rango de fechas indicado.

```
--RFC5 - MOSTRAR EL CONSUMO EN HOTELANDES POR UN USUARIO DADO, EN UN RANGO DE FECHAS INDICADO
SELECT
    clientes.nombre AS nombre_cliente,
    reservas.id AS reserva_id,
    SUM(consumos.costo) AS total_consumo
FROM
    clientes
    JOIN reservas ON clientes.num_documento = reservas.usuarios_num_documento
    JOIN reservan ON reservas.id = reservan.reservas_id
    JOIN consumos ON reservan.habitacion_id = consumos.habitacion_id
WHERE
    clientes.num_documento = : id_usuario
    AND reservas.fecha_inicio BETWEEN TO_DATE(:fecha_inicial, 'YYYY-MM-DD') AND TO_DATE(:fecha_final, 'YYYY-MM-DD')
GROUP BY
    clientes.nombre,
    reservas.id,
    reservas.fecha_inicio,
    reservas.fecha_salida
ORDER BY
    reservas.fecha_inicio;
```

El id del usuario el 1001 y las fechas son del 2023.

El tamaño de respuesta depende de los consumos realizados por el usuario. Mientras más consumos haga, más resultados dará la consulta

	NOMBRE_CLIENTE	RESERVA_ID	TOTAL_CONSUMO
1	Juan Perez	1	5



RFC6 - Analizar la operación de HotelAndes

```
--RFC6 - ANALIZAR LA OPERACIÓN DE HOTELANDES

--Mayor ocupacion
SELECT checkin.fecha_ingreso AS fecha, COUNT(distinct checkin.reservas_id) AS habitaciones_ocupadas
FROM checkin
GROUP BY checkin.fecha_ingreso
ORDER BY habitaciones_ocupadas DESC;

--Mayores ingresos
SELECT checkin.fecha_ingreso AS fecha, SUM(consumos.costo) AS ingresos
FROM checkin
JOIN reservan ON checkin.reservas_id = reservan.reservas_id
JOIN consumos ON consumos.habitacion_id = reservan.habitacion_id
GROUP BY checkin.fecha_ingreso
ORDER BY ingresos DESC;

--Menor ocupacion
SELECT checkin.fecha_ingreso AS fecha, COUNT(distinct checkin.reservas_id) AS habitaciones_ocupadas
FROM checkin
GROUP BY checkin.fecha_ingreso
ORDER BY habitaciones_ocupadas ASC;
```

Para la mayor ocupación, el tamaño de respuesta se refleja en las fechas que se registran habitaciones ocupadas. Mientras habitaciones estén ocupadas, mayores fechas aparecerán

	FECHA	HABITACIONES_OCUPADAS
1	01/11/23	1
2	02/11/23	1

Para los mayores ingresos, la distribución depende de las fechas las cuales se realizaron los consumos. Mientras más consumos hay en diferentes fechas, mayor es el tamaño de respuesta

	FECHA	INGRESOS
1	02/11/23	40
2	01/11/23	5

Para la menor ocupación, el tamaño de respuesta se refleja en las fechas que se registran habitaciones ocupadas. Mientras habitaciones estén ocupadas, mayores fechas aparecerán

	FECHA	HABITACIONES_OCUPADAS
1	01/11/23	1
2	02/11/23	1

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
SORT		ORDER BY	1	7
HASH		GROUP BY	1	7
VIEW	SYS.VM_NWVW_1		1	5
HASH		GROUP BY	1	5
TABLE ACCESS	CHECKIN	FULL	1	4

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
SORT		ORDER BY	1	2
HASH		GROUP BY	1	2
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
INDEX	RESERVAN_PK	FULL SCAN	1	0
TABLE ACCESS	CHECKIN	BY INDEX ROWID BATCHED	1	0
INDEX	CHECKIN_PK	RANGE SCAN	1	0
Access Predicates				
CHECKIN.RESERVAS_ID=RESERVAN.RESERVAS_ID				
INDEX	CONSUMOS_PK	RANGE SCAN	1	0
Access Predicates				
CONSUMOS.HABITACION_ID=RESERVAN.HABITACION_ID				
TABLE ACCESS	CONSUMOS	BY INDEX ROWID	1	0

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
SORT		ORDER BY	1	7
HASH		GROUP BY	1	7
VIEW	SYS.VM_NWVW_1		1	5
HASH		GROUP BY	1	5
TABLE ACCESS	CHECKIN	FULL	1	4

RFC7 - Encontrar los buenos clientes

```
--RFC7 - ENCONTRAR LOS BUENOS CLIENTES
SELECT
    clientes.nombre AS nombre_cliente,
    clientes.correo AS correo_cliente,
    SUM(DISTINCT (reservas.fecha_salida - reservas.fecha_inicio)) AS total_dias_estadia,
    SUM(consumos.costo) AS total_consumos
FROM clientes
inner JOIN reservas ON clientes.num_documento = reservas.usuarios_num_documento
inner JOIN checkin ON clientes.num_documento = checkin.clientes_num_documento
INNER JOIN reservan ON reservan.reservas_id = reservas.id
INNER JOIN habitaciones ON habitaciones.id = reservan.habitacion_id
INNER JOIN consumos ON consumos.habitacion_id = habitaciones.id

WHERE reservas.fecha_inicio between TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD')
GROUP BY clientes.num_documento, clientes.nombre, clientes.correo
HAVING SUM(DISTINCT (reservas.fecha_salida - reservas.fecha_inicio)) >= 14 OR SUM(consumos.costo) > 1;
```

El tamaño de respuesta aumentara a medida que aparezcan más clientes que cumplan con los requisitos de ser un buen cliente

	NOMBRE_CLIENTE	CORREO_CLIENTE	TOTAL_DIAS_ESTADIA	TOTAL_CONSUMOS
1	Juan Perez	juan.perez@example.com	4	5
2	Maria Lopez	maria.lopez@example.com	4	40

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
FILTER				
Filter Predicates				
OR				
SUM(INTERNAL_FUNCTION(ITEM_1))>=14				
SUM(ITEM_5)>1				
HASH		GROUP BY	1	1
VIEW	SYS.VW_DAG_0		1	1
HASH		GROUP BY	1	1
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
MERGE JOIN		CARTESIAN	1	0
NESTED LOOPS			1	0
NESTED LOC			1	0
INDEX	RESERVAN_PK	FULL SCAN	1	0
INDEX	CONSUMOS_PK	RANGE SCAN	1	0
Access Predicates				
CONSUMOS.HABITACION_ID=RESERVAN.HABITACION_ID				
CONSUMOS.HABITACION_ID=RESERVAN.HABITACION_ID				
TABLE ACCE	CONSUMOS	BY INDEX ROWID	1	0
BUFFER		SORT	1	0
INDEX	CHECKIN_PK	FULL SCAN	1	0
TABLE ACCESS	CLIENTES	BY INDEX ROWID	1	0
INDEX	CLIENTES_PK	UNIQUE SCAN	1	0
Access Predicates				
CLIENTES.NUM_DOCUMENTO=CHECKIN.CLIENTES_NUM_DOCUMENTO				
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
RESERVAN.RESERVAS_ID=RESERVAS.ID				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
CLIENTES.NUM_DOCUMENTO=RESERVAS.USUARIOS_NUM_DOCUMENTO				
RESERVAS.FECHA_INICIO>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVAS.FECHA_INICIO<=TO_DATE(' 2023-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				

RFC8 - Encontrar los servicios que no tienen mucha demanda

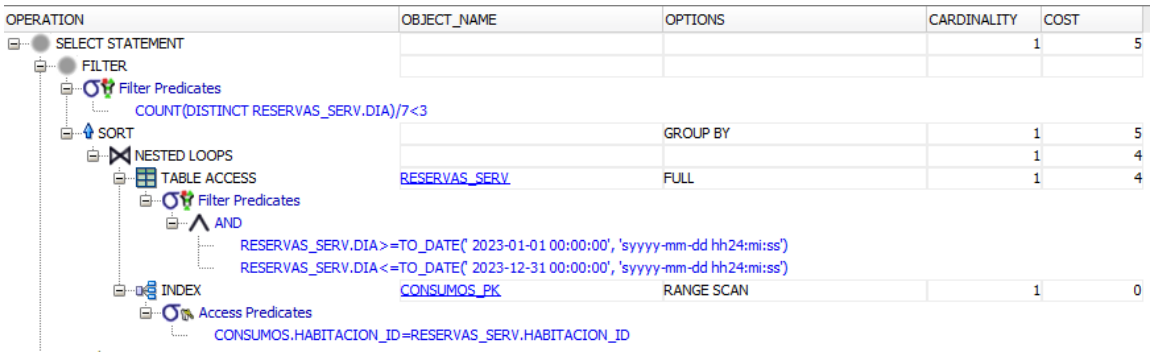
```
--RFC8 - ENCONTRAR LOS SERVICIOS QUE NO TIENEN MUCHA DEMANDA

SELECT servicios.tipo AS tipo_servicio, COUNT(DISTINCT TO_DATE(reservas_serv.dia, 'YYYY-MM-DD')) / 7 AS solicitudes_semanales
FROM servicios
LEFT JOIN consumos ON servicios.tipo = consumos.servicios_tipo
left join habitaciones on consumos.habitacion_id=habitaciones.id
left join reservas_serv on reservas_serv.habitacion_id = habitaciones.id

WHERE reservas_serv.dia between TO_DATE('2023-01-01', 'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD')
GROUP BY servicios.tipo
HAVING (COUNT(DISTINCT reservas_serv.dia) / 7) < 3;
```

El tamaño de respuesta aumenta a medida que disminuyan las peticiones semanales de servicios.

	TIPO_SERVICIO	SOLICITUDES_SEMANALES
1	bar	0,1428571428571428571428571428571428571429
2	restaurante	0,1428571428571428571428571428571428571429



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	5
FILTER				
Filter Predicates				
COUNT(DISTINCT RESERVAS_SERV.DIA)/7<3				
SORT		GROUP BY	1	5
NESTED LOOPS			1	4
TABLE ACCESS	RESERVAS_SERV	FULL	1	4
Filter Predicates				
AND				
RESERVAS_SERV.DIA>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVAS_SERV.DIA<=TO_DATE(' 2023-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	CONSUMOS_PK	RANGE SCAN	1	0
Access Predicates				
CONSUMOS.HABITACION_ID=RESERVAS_SERV.HABITACION_ID				

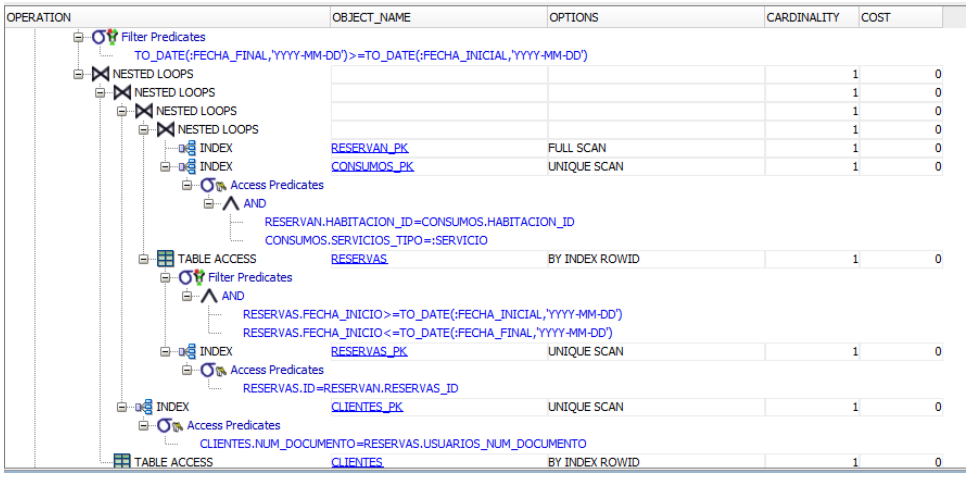
RFC9 - Consultar consumo en HotelAndes

```
--RFC9 - CONSULTAR CONSUMO EN HOTELANDES
SELECT clientes.nombre AS nombre_cliente, clientes.num_documento AS num_documento_cliente, servicios.tipo AS servicio_consumido,
COUNT(consumos.habitacion_id) AS cantidad_consumos, reservas.fecha_inicio as fecha
FROM clientes
INNER JOIN reservas ON clientes.num_documento = reservas.usuarios_num_documento
INNER JOIN reservan ON reservas.id = reservan.reservas_id
INNER JOIN consumos ON reservan.habitacion_id = consumos.habitacion_id
INNER JOIN servicios ON consumos.servicios_tipo = servicios.tipo
WHERE
    reservas.fecha_inicio BETWEEN TO_DATE(:fecha_inicial, 'YYYY-MM-DD') AND TO_DATE(:fecha_final, 'YYYY-MM-DD')
    and servicios.tipo= : servicio --Este tambien varia
GROUP BY
    clientes.nombre,clientes.num_documento,servicios.tipo, reservas.fecha_inicio
ORDER BY
    --nombre_cliente
    --cantidas_consumos
    fecha;
```

El servicio es bar y las fechas son del 2023.

El tamaño de respuesta depende de los clientes que cumplan con las condiciones del requerimiento. A medida que clientes consumen un servicio en esas fechas, mayor será el tamaño.

	NOMBRE_CLIENTE	NUM_DOCUMENTO_CLIENTE	SERVICIO_CONSUMIDO	CANTIDAD_CONSUMOS	FECHA
1	Juan Perez	1001	bar	1	01/11/23



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
Filter Predicates				
TO_DATE(:FECHA_FINAL,'YYYY-MM-DD')>=TO_DATE(:FECHA_INICIAL,'YYYY-MM-DD')				
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
INDEX	RESERVAN_PK	FULL SCAN	1	0
INDEX	CONSUMOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
RESERVAN.HABITACION_ID=CONSUMOS.HABITACION_ID				
CONSUMOS.SERVICIOS_TIPO=:SERVICIO				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
RESERVAS.FECHA_INICIO>=TO_DATE(:FECHA_INICIAL,'YYYY-MM-DD')				
RESERVAS.FECHA_INICIO<=TO_DATE(:FECHA_FINAL,'YYYY-MM-DD')				
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
RESERVAS.ID=RESERVAN.RESERVAS_ID				
INDEX	CLIENTES_PK	UNIQUE SCAN	1	0
Access Predicates				
CLIENTES.NUM_DOCUMENTO=RESERVAS.USUARIOS_NUM_DOCUMENTO				
TABLE ACCESS	CLIENTES	BY INDEX ROWID	1	0

RFC10 - Consultar consumo en HotelAndes – RFC9-V2

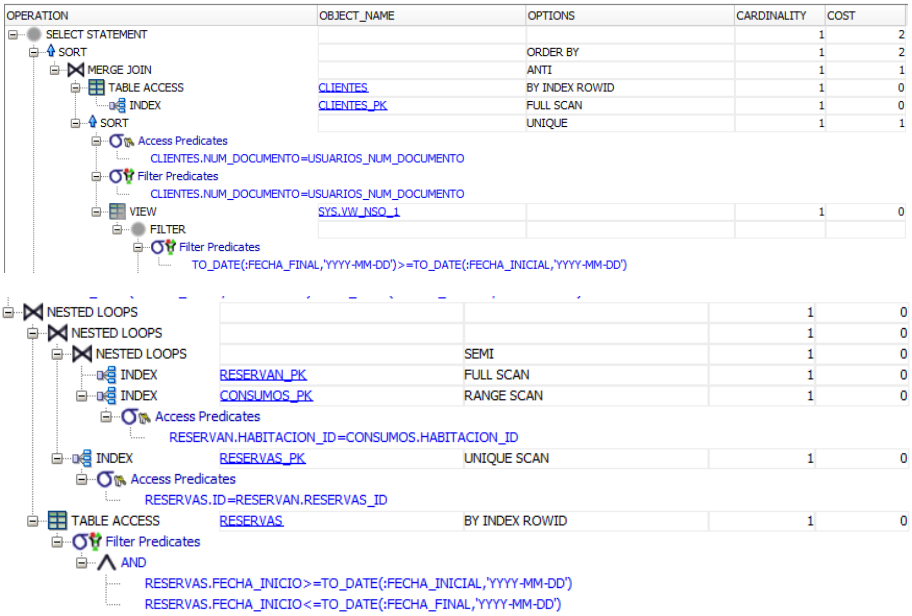
```
--RFC10 - CONSULTAR CONSUMO EN HOTELANDES - RFC9-V2
SELECT clientes.nombre AS nombre_cliente,clientes.num_documento AS num_documento_cliente
FROM clientes
WHERE clientes.num_documento NOT IN (
    SELECT DISTINCT reservas.usuarios_num_documento
    FROM reservas
    INNER JOIN reservan ON reservas.id = reservan.reservas_id
    INNER JOIN consumos ON reservan.habitacion_id = consumos.habitacion_id
    WHERE
        reservas.fecha_inicio BETWEEN TO_DATE(:fecha_inicial, 'YYYY-MM-DD') AND TO_DATE(:fecha_final, 'YYYY-MM-DD')

)
ORDER BY
    nombre_cliente;
```

Las fechas y el servicio son los mismos al requerimiento anterior.

El tamaño de respuesta depende de aquellos usuarios que no consuman el servicio en esas fechas.

	NOMBRE_CLIENTE	NUM_DOCUMENTO_CLIENTE
1	Ana Lopez	1004
2	Pedro Gomez	1003



RFC11 - Consultar funcionamiento

```
--RFC11 - CONSULTAR FUNCIONAMIENTO

SELECT
  TO_CHAR(reservas.fecha_inicio, 'IYYY-IW') AS semana,
  MAX(servicios.tipo) AS servicio_mas_consumido,
  MIN(servicios.tipo) AS servicio_menos_consumido,
  MAX(habitaciones.id) AS habitacion_mas_solicitada,
  MIN(habitaciones.id) AS habitacion_menos_solicitada
FROM reservas
JOIN reservan ON reservas.id = reservan.reservas_id
JOIN consumos ON reservan.habitacion_id = consumos.habitacion_id
JOIN servicios ON consumos.servicios_tipo = servicios.tipo
JOIN habitaciones ON reservan.habitacion_id = habitaciones.id
WHERE reservas.fecha_inicio >= TO_DATE('2023-01-01', 'YYYY-MM-DD')
  AND reservas.fecha_inicio <= TO_DATE('2023-12-31', 'YYYY-MM-DD')
GROUP BY TO_CHAR(reservas.fecha_inicio, 'IYYY-IW')
ORDER BY semana;
```

El tamaño depende de las semanas que son tenidas en cuenta donde existan servicios y habitaciones reservados.

	SEMANA	SERVICIO_MAS_CONSUMIDO	SERVICIO_MENOS_CONSUMIDO	HABITACION_MAS_SOLICITADA	HABITACION_MENOS_SOLICITADA
1	2023-44	restaurante	bar	102	101

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
SORT		ORDER BY	1	2
HASH		GROUP BY	1	2
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
INDEX	RESERVAN_PK	FULL SCAN	1	0
INDEX	CONSUMOS_PK	RANGE SCAN	1	0
Access Predicates				
RESERVAN.HABITACION_ID=CONSUMOS.HABITACION_ID				
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
RESERVAS.ID=RESERVAN.RESERVAS_ID				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
Filter Predicates				
AND				
RESERVAS.FECHA_INICIO>=TO_DATE(' 2023-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVAS.FECHA_INICIO<=TO_DATE(' 2023-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				

RFC12 - Consultar los clientes excelentes

```
--RFC12 - CONSULTAR LOS CLIENTES EXCELENTES
SELECT DISTINCT
    clientes.num_documento AS numero_documento,
    clientes.nombre AS nombre_cliente,
    clientes.correo AS correo_cliente,
    consumos.costo as gastos
FROM clientes
inner JOIN reservas ON clientes.num_documento = reservas.usuarios_num_documento
inner JOIN reservan ON reservas.id = reservan.reservas_id
inner JOIN reservas_serv ON reservan.habitacion_id = reservas_serv.habitacion_id
inner JOIN reservas_spa on reservas_spa.reservaserv_id = reservas_serv.id
inner JOIN consumos ON consumos.habitacion_id = reservan.habitacion_id
WHERE
       (TRUNC(reservas.fecha_salida) - TRUNC(reservas.fecha_inicio)) <= 90 OR
       consumos.costo > 300000 OR

    (
        consumos.servicios_tipo IN ('SPA', 'Salones de Reuniones')
        AND reservas_serv.duracion_hora > 4
    )


    order by consumos.costo;
```

El tamaño se distribuye en cuantos clientes cumples con los requerimientos para ser excelentes.

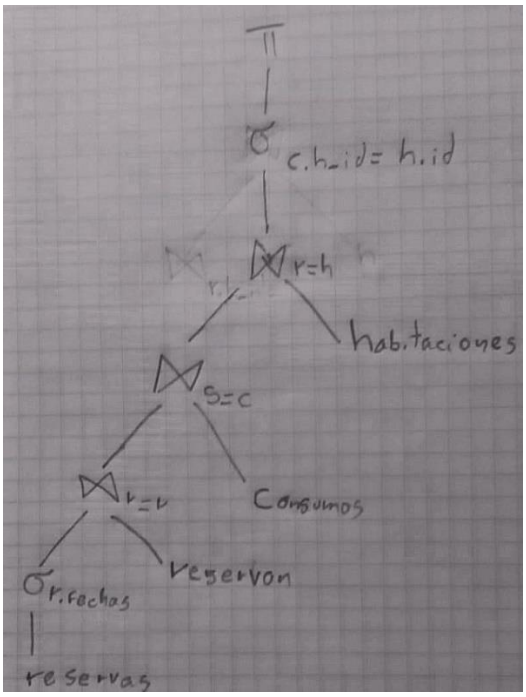
	NUMERO_DOCUMENTO	NOMBRE_CLIENTE	CORREO_CLIENTE	GASTOS
1	1001	Juan Perez	juan.perez@example.com	5
2	1002	Maria Lopez	maria.lopez@example.com	40

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
SORT		UNIQUE	1	1
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
NESTED LOOPS			1	0
INDEX	RESERVAS_SPA_PK	FULL SCAN	1	0
TABLE ACCESS	RESERVAS_SERV	BY INDEX ROWID	1	0
INDEX	RESERVAS_SERV_PK	UNIQUE SCAN	1	0
Access Predicates				
RESERVAS_SPA.RESERVASERV_ID=RESERVAS_SERV.ID				
INDEX	RESERVAN_PK	RANGE SCAN	1	0
Access Predicates				
RESERVAN.HABITACION_ID=RESERVAS_SERV.HABITACION_ID				
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	0
INDEX	RESERVAS_PK	UNIQUE SCAN	1	0
Access Predicates				
RESERVAS.ID=RESERVAN.RESERVAS_ID				
TABLE ACCESS	CLIENTES	BY INDEX ROWID	1	0
INDEX	CLIENTES_PK	UNIQUE SCAN	1	0
Access Predicates				
CLIENTES.NUM_DOCUMENTO=RESERVAS.USUARIOS_NUM_DOCUMENTO				
INDEX	CONSUMOS_PK	RANGE SCAN	1	0
Access Predicates				
CONSUMOS.HABITACION_ID=RESERVAN.HABITACION_ID				
TABLE ACCESS	CONSUMOS	BY INDEX ROWID	1	0
Filter Predicates				
OR				
TRUNC(INTERNAL_FUNCTION(RESERVAS.FECHA_SALIDA))-TRUNC(INTERNAL_FUNCTION(RESERVAS.FECHA_INICIO))<=90				
CONSUMOS.COSTO>300000				
AND				
OR				
CONSUMOS.SERVICIOS_TIPO='SPA'				
CONSUMOS.SERVICIOS_TIPO='Salones de Reuniones'				
RESERVAS_SERV.DURACION_HORA>4				

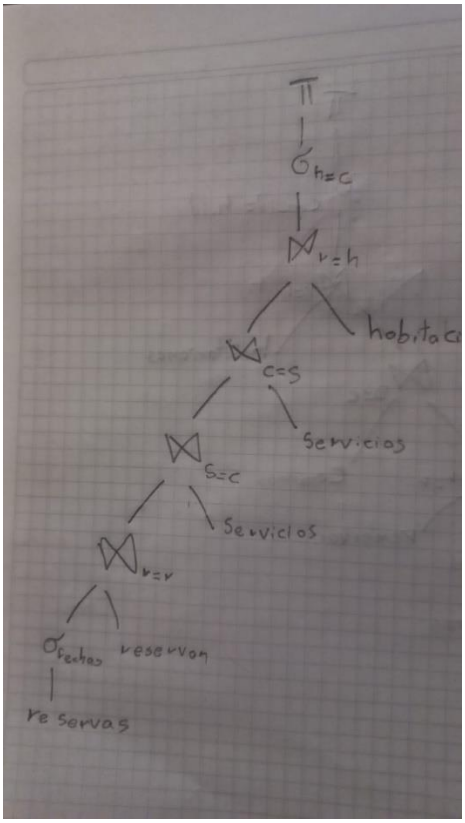
COMPARACIÓN CON PLANES DE EJECUCIÓN

Casi todas las consultas se hacen de la misma manera por lo que se mostraran algunos requerimientos para comprobarlo:

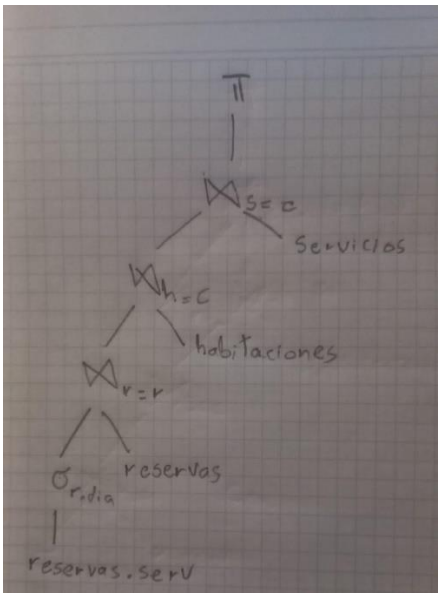
Para RC1:



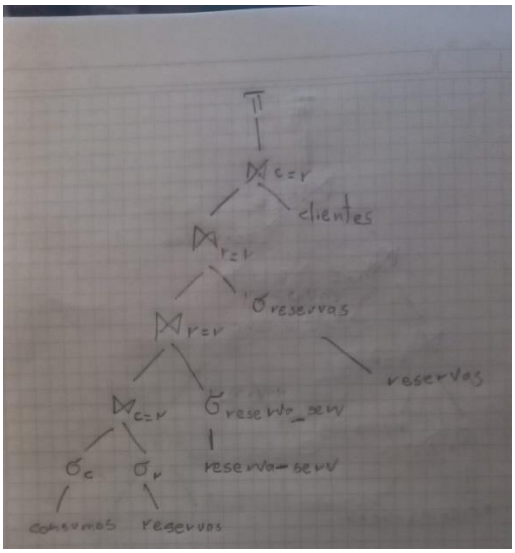
Para RFC2:



Para RFC8:



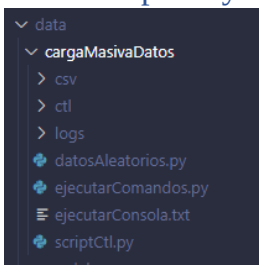
Para RFC12:



Se puede evidenciar que, tanto en los planes del SQL como en los propuestos, se realizan primero joins que junten las tablas por atributos que tengan en común. Para estos casos son los identificadores. Del join desprendían otros join con las siguientes uniones de tablas y las tablas que se van usando. Después de las uniones aparece la sigma donde van las condiciones del where hasta cumplir lo que pide la consulta.

CARGA DE DATOS MASIVA

Organización de carpetas y archivos

- 

datosAleatorios.py

- Este archivo se encarga de generar datos aleatorios para todas las tablas de nuestra base de datos. Además, toma en cuenta las restricciones de negocio que fueron definidas en el script de creación de la BD. Estos datos son guardados en archivos .csv, los cuales se encuentran en la ruta *"data\cargaMasivaDatos\csv"*.
- Se realiza un único archivo el cual se encarga de generar datos para todas las tablas.
- Acá también se define la cantidad de registros que queremos generar para cada tabla. A continuación, mostraremos un gráfico para ilustrar la cantidad de registros que solicitamos para cada tabla mediante el script:

Nombre tabla	# registros
adicionales	100
bares	7,500
checkin	400,000
checkouts	190,000
clientes	90,000
conferencias	5
consumos	100
equipos	100
gimnasios	100
gratis	100
habitaciones	10,000
internets	100
lavanderias	100
ofrecen	100

Nombre tabla	# registros
piscinas	100
planes_consumo	100
prestan	100
productos_bar	100
productos_res	100
productos_super	100
productos_t	100
reservan	15,000
reservas	15,000
reservas_serv	100
reservas_spa	100
restaurantes	100
reuniones	5
servicios	11

Nombre tabla	# registros
servicios_prestamo	100
servicios_spa	100
serv_reuniones	100
sirven_bares	100
sirven_res	1,000
spas	100
supermercados	300
tiendas	5
tipo_hab	4
tipo_usuarios	4
usuarios	3,000
utensilios	100
venden_super	100

TOTAL	734,334

-

data\cargaMasivaDatos\csv

- Acá se almacenan los datos que el archivo *datosAleatorios.py* genera. La primera línea se refiere a las columnas de las tablas de la BD en su respectivo orden, luego los datos siguen este mismo orden. Cada columna es separada por una ",".
- Estos archivos son usados luego por los archivos que se encuentran en la ruta *"data\cargaMasivaDatos\ctl"*.
- Se realiza un archivo por cada tabla.

scriptCtl.py

- Este archivo se encarga de crear los ctl automáticamente. Se define una lista con los nombres de todas las tablas y para crear los atributos, se lee la primera línea de los .csv y se definen en los ctl.
- Este script es de gran ayuda para automatizar el proceso de la creación de ctl. En nuestro caso, son 43 archivos .ctl, lo cual tomaría mucho tiempo de realizar uno por uno, por ello decidimos automatizarlo de esta manera.

data\cargaMasivaDatos\ctl

- Acá se encuentran los archivos que leen los .csv, separan los datos por la "," para luego insertarlos en las respectivas tablas respetando el orden de los atributos.
- Se realiza un archivo por cada tabla.

ejecutarConsola.txt

- Este archivo contiene todos los comandos que se deben ejecutar en la terminal, los cuales tienen la siguiente estructura:
  - sqlldr {usuario}/{contraseña}@{host}{puerto}/{nombreServicio} control=data/cargaMasivaDatos/ctl/cargaMasiva_{nombre_tabla}.ctl log=data/cargaMasivaDatos/logs/cargaMasiva_{nombre_tabla}.log
- Cada línea del archivo es un comando que se debe ejecutar uno por uno en la terminal, pues de esta manera las tablas de la base de datos son pobladas con la data que se tiene en los archivos .csv
- También genera un log en la ruta *"data/cargaMasivaDatos/logs/"*, acá se registra el estado de la ejecución de los scripts. Nos ayuda a identificar errores, la cantidad de tuplas insertadas, etc.

ejecutarComandos.py

- Este script accede al archivo *"ejecutarConsola.txt"*, lee cada línea del archivo y ejecuta línea por línea en la terminal.
- Este archivo este gran ayuda, pues también nos ayuda a automatizar el proceso, reduce el tiempo que este conlleva e incluso posibles errores humanos que pueda haber.