DOCUMENTO DE INFORME

Manuel Carvajal - 202014203

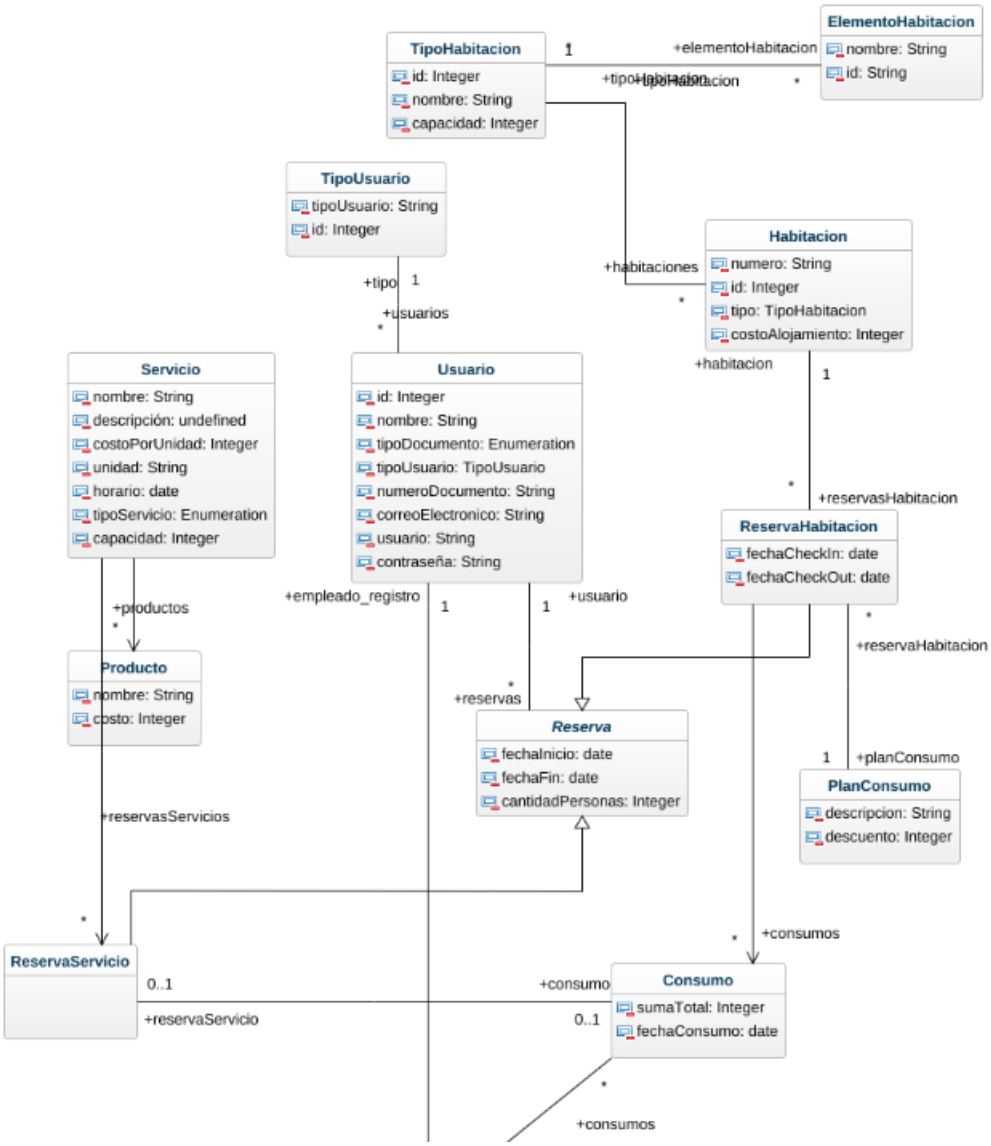
Nicolas Ruiz – 20213608

Juan Pablo Hernández - 202122707

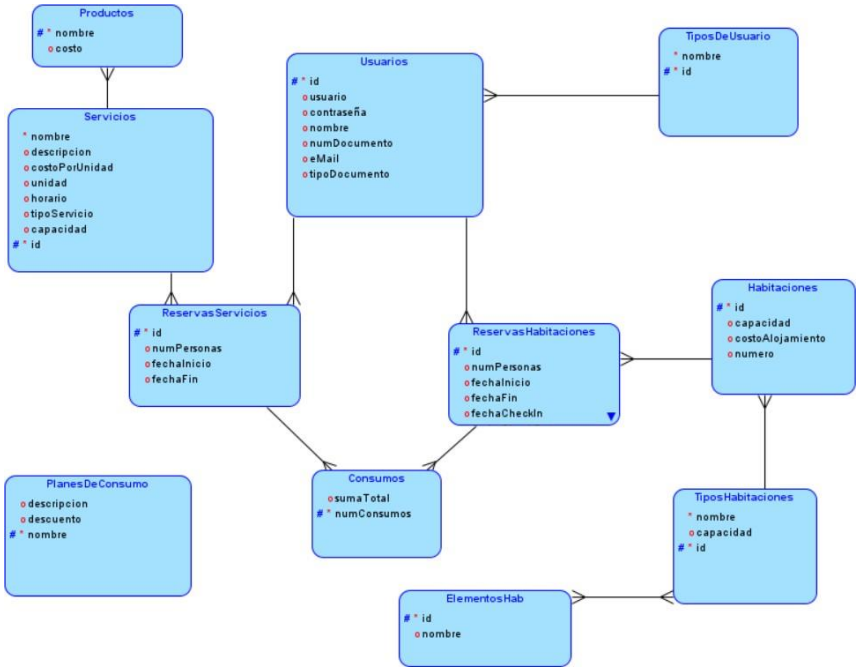
HOTEL DE LOS ANDES – Descripción

Uniandes ha decidido implementar HOTEL DE LOS ANDES, una aplicación que apoye a las cadenas hoteleras en su operación diaria. Cada hotel que opera utilizando HOTEL DE LOS ANDES tiene su propia instancia de la aplicación.

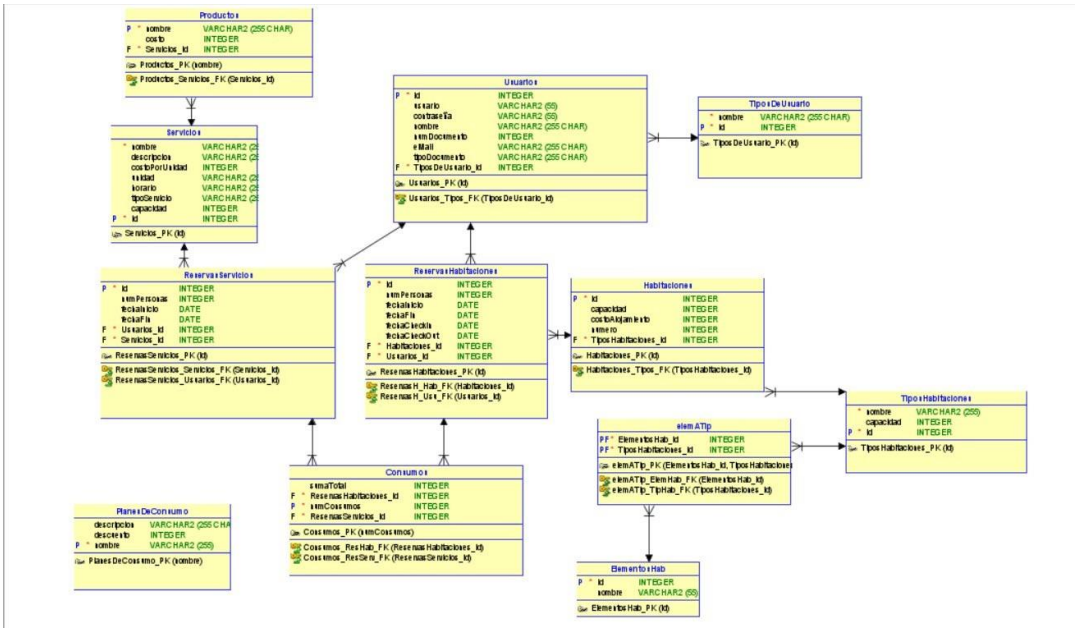
Modelo Conceptual



Modelo E/R



Modelo Relacional



Resultados logrados

- Cobertura optima de todos los requerimientos descritos en el enunciado representados en las entidades mostradas
- Abstracción coherente del mundo del problema para la correcta traducción entre modelos conceptual/E.R/Relacional..
- Se logró implementar el sistema de gestión hotelera de manera efectiva, cumpliendo con los requisitos especificados.
- Los requerimientos funcionales 1-12 funcionan correctamente
- Uso correcto de los índices creados por el equipo de trabajo

Carga Masiva:

- 10000 usuarios
- 4500 habitaciones
- 220200 reservas de alojamiento
- 250200 reservas de servicios
- 250200 registros de consumos
- 5 tipos de usuarios
- 2 tipos de habitaciones
- 2 planes de consumo

Total de datos cargados: 735.109 datos

Generación de Datos para la Carga Masiva:

Para generar los datos a gran escala, se programó y utilizó un algoritmo de Python llamado GenerateData.py, el cual está disponible en el repositorio en la sección de Docs. Dicho algoritmo emplea estrategias de generación de datos sistemática, procesal y aleatoria, para poder generar registros para poblar todas las tablas necesarias de la base de datos de Hotel de los Andes.

Cabe resaltar que el script no se utilizó para poblar todas las tablas, ya que algunas de estas necesitaban tan solo unos cuantos registros. Dichas tablas eran, la de tipos de usuarios, la de tipos de habitaciones, la de planes de consumo y la de servicios; las demás tablas de la base de datos sí utilizan el script de formas específicas.

Para la generación de usuarios, se utilizó una lista predeterminada para nombres y apellidos, y se seleccionan dos elementos aleatoriamente. Para la contraseña y documento, se generan cadenas de caracteres aleatorios de tamaños y cualidades específicas. Para el usuario y correo se utiliza el nombre y apellido escogidos y se concatenan con números aleatorios y un domain al azar. Finalmente, el tipo de documento y el tipo de usuario se obtienen de una lista predefinida.

Para las habitaciones, se hizo dos fors anidados para simular los pisos del hotel y las habitaciones por piso. Para la capacidad de la habitación y el costo, se generó un número aleatorio entre dos intervalos predefinidos. En cuanto al tipo de habitación, se escoje aleatoriamente un valor de los que ya existen en la base datos.

Para la tabla de reservas de habitaciones, se asigna un número de personas aleatoriamente entre dos rangos predefinidos. Para todas las fechas, se generan aleatoriamente valores entre ciertos rangos, pero se asegura que la fecha de inicio y fecha de checkin sea antes que la fecha de fin y fecha de checkout respectivamente. Por último, se le asigna un id de habitación y de usuario aleatorio a partir del número total de cada uno en la base de datos; el plan de consumo se escoge de un grupo predeterminado.

Para las reservas de servicios, se sigue un procedimiento similar. El número de personas se genera aleatoriamente entre un rango definido. La fecha de inicio y la fecha de finalización se generan aleatoriamente, asegurando que la primera sea antes que la segunda. Por último, se le asigna un id de usuario y de servicio al azar, basado en el número de estos que estén cargados en la base de datos.

Por último, para la tabla de consumos, se generan valores aleatorios para la suma total, el id de habitación y el número de consumos. Se obtiene el id del servicio con su respectivo nombre aleatoriamente de todos los servicios cargados en la base de datos. Finalmente, se le asigna una fecha de consumo e id de usuario aleatorios.

Documentación de los requerimientos:

RFC1 - MOSTRAR EL DINERO RECOLECTADO POR SERVICIOS EN CADA HABITACIÓN EN EL ÚLTIMO AÑO CORRIDO.

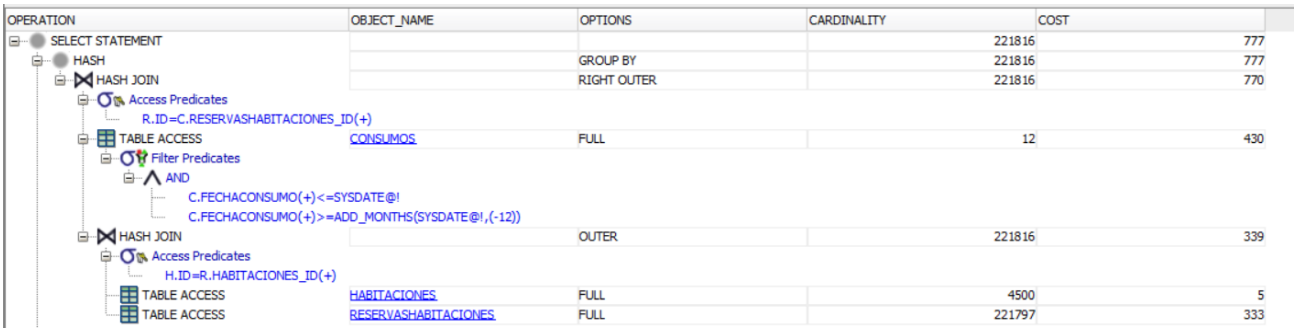
Índices:

- Índices en habitaciones.id, reservashabitaciones.habitaciones_id, y consumos.reservashabitaciones_id debido a los JOINs.
- Un índice en consumos.fechaconsumo para el filtrado por rango de fechas.

CREATE INDEX idx_r_habitaciones_id ON reservashabitaciones(habitaciones_id); CREATE INDEX idx_c_reservashabitaciones_id ON consumos(reservashabitaciones_id); CREATE INDEX idx_c_fechaconsumo ON consumos(fechaconsumo);

Parámetros: La sentencia no tiene parámetros.

Plan de ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			221816	777
HASH		GROUP BY	221816	777
HASH JOIN		RIGHT OUTER	221816	770
Access Predicates				
R.ID=C.RESERVASHABITACIONES_ID(+)				
TABLE ACCESS	CONSUMOS	FULL	12	430
Filter Predicates				
AND				
C.FECHACONSUMO(+)<=SYSDATE@!				
C.FECHACONSUMO(+)>=ADD_MONTHS(SYSDATE@!,(-12))				
HASH JOIN		OUTER	221816	339
Access Predicates				
H.ID=R.HABITACIONES_ID(+)				
TABLE ACCESS	HABITACIONES	FULL	4500	5
TABLE ACCESS	RESERVASHABITACIONES	FULL	221797	333

Como se puede apreciar, Oracle no crea indices para esta consulta. Por lo que probablemente el uso de estos no reduciría de forma considerable el tiempo de ejecución.

Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 5
Transcurrido: 00:00:00.345
```

RFC2 - MOSTRAR LOS 20 SERVICIOS MÁS POPULARES.

Los que fueron más consumidos en un período de tiempo dado

índices

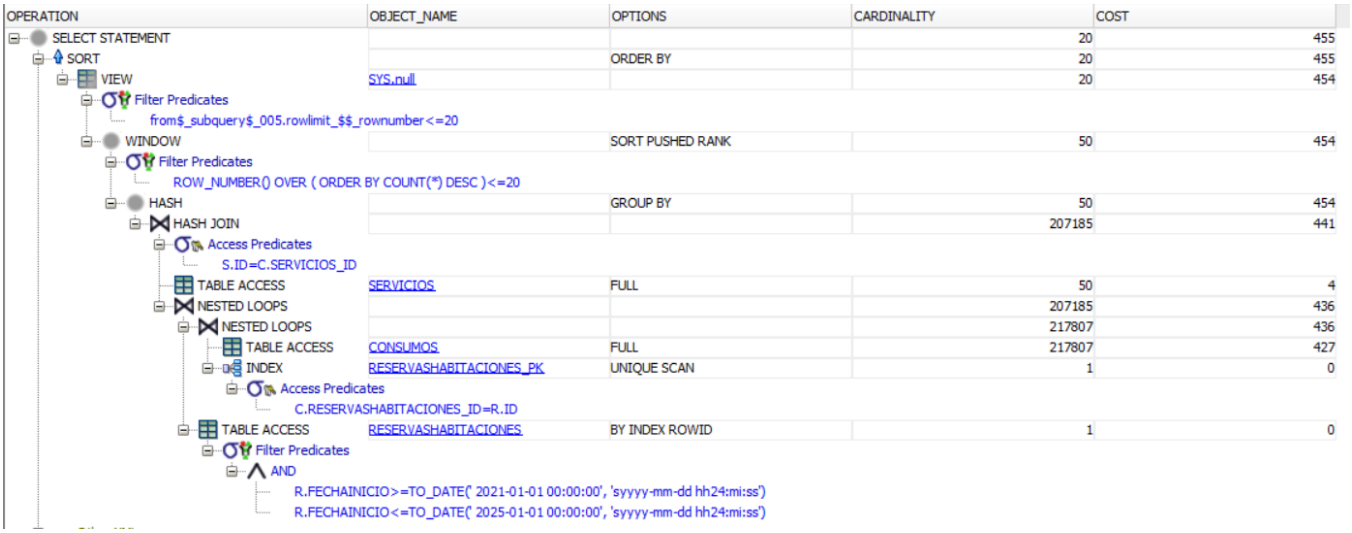
Un índice en servicios.id ya que se usa para el JOIN.
Un índice en consumos.servicios_id y reservashabitaciones.fechainicio debido al JOIN y al subquery WHERE.

CREATE INDEX idx_s_id ON servicios(id);
CREATE INDEX idx_c_servicios_id ON consumos(servicios_id);

CREATE INDEX idx_rh_fechainicio2 ON reservashabitaciones(fechainicio);

Parametros: La sentencia tiene dos parametros, fecha 1 y fecha 2. Donde fecha 1 tiene que ser una fecha anterior a la fecha 2.

Plan de ejecución



Como se puede apreciar, no se crearon indices en esta consulta, esto puede ser debido a que no se hacen Joins. Por lo que no es necesario crearlos para reducir el tiempo de los mismos.

Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 6
Transcurrido: 00:00:00.346
```

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

Se debe mostrar el % de ocupación de cada habitación en el último año

```
SELECT h.id,h.numero AS,COALESCE(ROUND(100 * SUM(NVL(r.fechafin, SYSDATE) -
r.fechainicio) / 365, 2), 0)
FROM habitaciones h
LEFT JOIN reservashabitaciones r ON h.id = r.habitaciones_id AND r.fechainicio
BETWEEN ADD_MONTHS(SYSDATE, -12) AND SYSDATE
GROUP BY  h.id, h.numero
ORDER BY h.id;
```

Índices

Un índice en habitaciones.id ya que se utiliza para el JOIN.
Índices en las columnas reservashabitaciones.fechainicio y reservashabitaciones.habitaciones_id porque se usan para el JOIN y el cálculo de fechas.

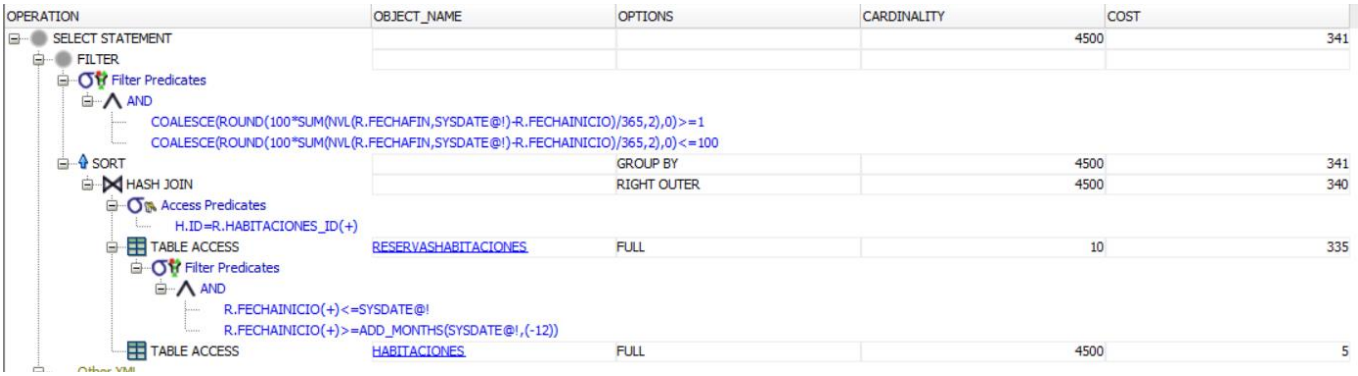
```
CREATE INDEX idx_habitaciones_id ON habitaciones(id); CREATE INDEX
idx_r_fechainicio ON reservashabitaciones(fechainicio);
```

En la consulta, se estan filtrando las reservas con la condición fechainicio BETWEEN

ADD_MONTHS(SYSDATE, -12) AND SYSDATE. Se están buscando en el rango de fechas en la columna fechainicio. Un índice en esta columna facilita la búsqueda y recuperación de las filas que cumplen con esta condición.

Parametros: La sentencia no tiene parametros.

Plan de ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4500	341
FILTER				
Filter Predicates				
AND				
COALESCE(ROUND(100*SUM(NVL(R.FECHAFIN,SYSDATE@!)-R.FECHAINICIO)/365,2),0)>=1				
COALESCE(ROUND(100*SUM(NVL(R.FECHAFIN,SYSDATE@!)-R.FECHAINICIO)/365,2),0)<=100				
SORT		GROUP BY	4500	341
HASH JOIN		RIGHT OUTER	4500	340
Access Predicates				
H.ID=R.HABITACIONES_ID(+)				
TABLE ACCESS	RESERVASHABITACIONES	FULL	10	335
Filter Predicates				
AND				
R.FECHAINICIO(+)<=SYSDATE@!				
R.FECHAINICIO(+)>=ADD_MONTHS(SYSDATE@!,(-12))				
TABLE ACCESS	HABITACIONES	FULL	4500	5

Como se puede apreciar, Oracle no creó indices para esta consulta.

Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 4
Transcurrido: 00:00:00.358
```

RFC4 - MOSTRAR LOS SERVICIOS QUE CUMPLEN CON CIERTA CARACTERÍSTICA

Las características son, por ejemplo, el precio se encuentra en un cierto rango, la fecha de consumo está en un rango de tiempo, servicios que fueron registrados por un cierto empleado, los servicios que son de un cierto tipo o de una cierta categoría. Se pueden aplicar múltiples características en simultaneo. Se debe presentar toda la información de los servicios que cumplen con las características

```
select s.nombre, s.descripcion, s.costoporunidad, s.unidad, s.horario,
s.tiposervicio, s.capacidad from servicios s,consumos c
where costoporunidad Between 0 and 1 and
s.id = c.servicios_id and c.fechaconsumo between to_date('2020-01-01','YYYY-MM-
DD') and to_date('2025-01-01','YYYY-MM-DD') and
c.usuarios_id = 12 and s.tiposervicio = 'joyas'
```

Índices

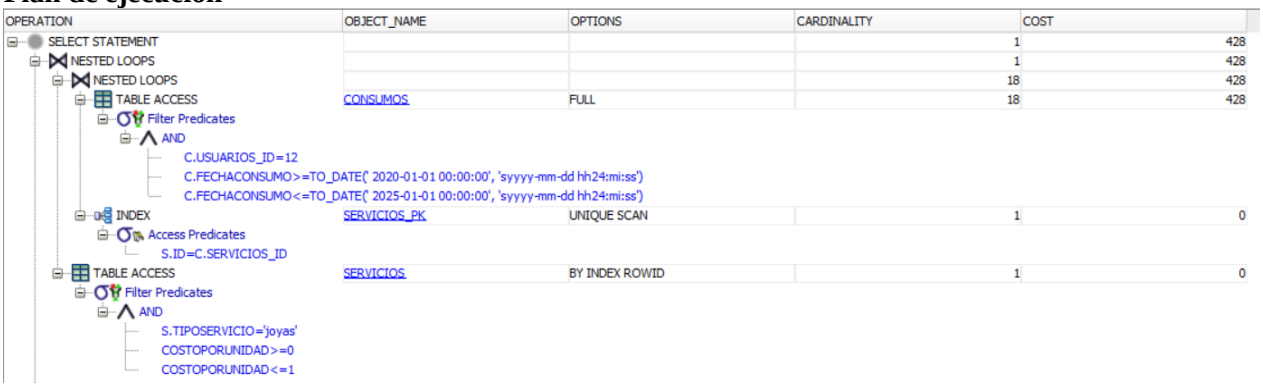
```
CREATE INDEX consumos_index ON consumos (servicios_id, fechaconsumo,
usuarios_id);
```

Este índice es compuesto y se enfoca en las condiciones del WHERE. Al incluir estas columnas en el índice, la base de datos puede realizar búsquedas más eficientes y reducir el tiempo de acceso a las filas relevantes en la tabla consumos. fechaconsumo es util porque se quiere revisar un rango especifico y servicios_id y usuarios_id porque se quieren revisar condiciones de igualdad.
La columna servicios_id se utiliza para el JOIN de las tablas servicios y consumos, el índice facilita la búsqueda y el JOIN.

Parametros: costo 1, costo 2, fecha 1, fecha 2, usuario_id y tipo de servicio. Pero cada par de parametros es opcional, pues los costos, las fechas, el usuario y el tipo de servicio pueden ser null independientemente del valor de los demás. Se cumple que 0 <= costo 1 <= costo 2 y que fecha 1 < = fecha 2 y que 1 <=

usuarios_id

Plan de ejecución



Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 10
Transcurrido: 00:00:00.363
```

Como se puede apreciar en el plan de ejcución, se creó un indice para la llave primaria de la tabla de servicios, esto debido al join que se hace en la consulta, esto permite reducir el tiempo de ejecución al reducir el número de operaciones en el join.

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

Recuerde que un cliente puede alojarse en el hotel cuantas veces quiera.

```
SELECT u.id,u.nombre,s.nombre ,c.sumatotal
FROM consumos c
JOIN reservashabitaciones rh ON c.reservashabitaciones_id = rh.id
JOIN servicios s ON c.servicios_id = s.id
JOIN usuarios u ON rh.usuarios_id = u.id
WHERE  u.id = :usuarios_id
AND rh.fechainicio BETWEEN TO_DATE(:fecha1,'YYYY-MM-DD') AND
TO_DATE(:fecha2,'YYYY-MM-DD')
```

Índices

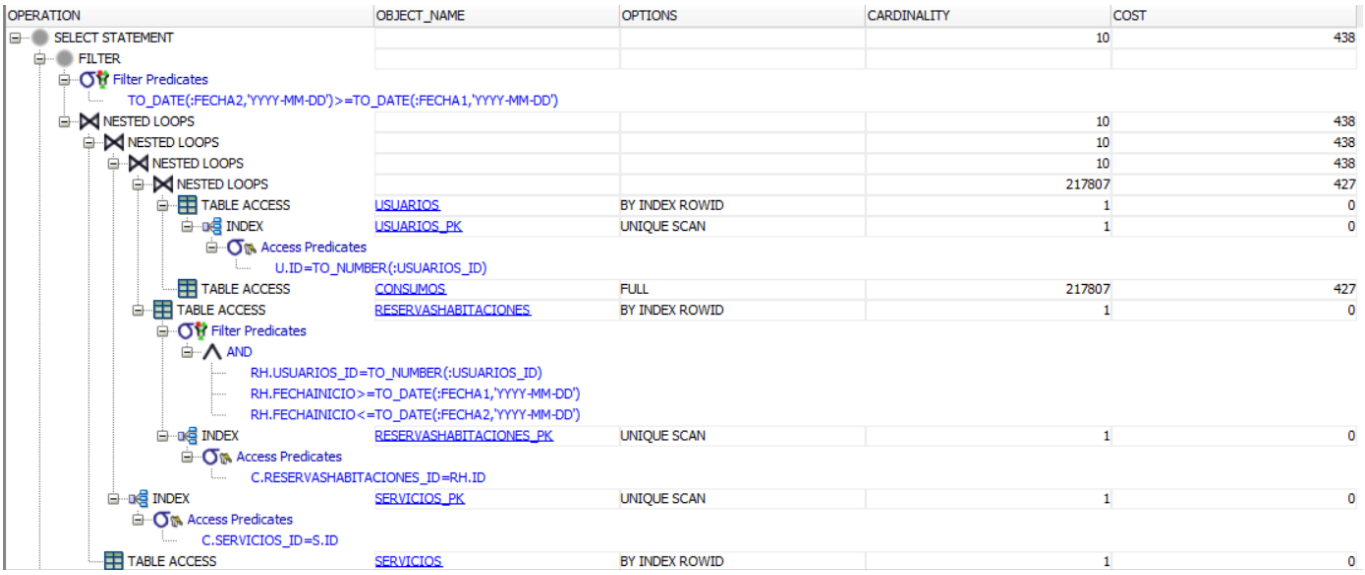
Un índice en la columna usuarios.id porque se utiliza en la cláusula WHERE.
Índices en las columnas reservashabitaciones.fechainicio y reservashabitaciones.usuarios_id ya que se utilizan en la operación JOIN y también en la cláusula WHERE para el rango de fechas.

CREATE INDEX idx_usuarios_id ON usuarios(id);
CREATE INDEX idx_rh_usuarios ON reservashabitaciones(usuarios_id);
CREATE INDEX idx_rh_fechainicio ON reservashabitaciones(fechainicio);

Este índice es compuesto y se enfoca en las condiciones del JOIN y WHERE de la tabla consumos. reservashabitaciones_id se usa para el JOIN con reservashabitaciones. servicios_id se usa para el JOIN con servicios. El índice facilita la búsqueda y el JOIN.

Parametros: fecha 1, fecha 2 y una id de usuario. Se tiene que cumplir que 1 <= usuarios_id y que fecha 1 < fecha 2.

Plan de ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			10	438
FILTER				
Filter Predicates				
TO_DATE(:FECHA2,'YYYY-MM-DD')>=TO_DATE(:FECHA1,'YYYY-MM-DD')				
NESTED LOOPS			10	438
NESTED LOOPS			10	438
NESTED LOOPS			10	438
NESTED LOOPS			217807	427
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
U.ID=TO_NUMBER(:USUARIOS_ID)				
TABLE ACCESS	CONSUMOS	FULL	217807	427
TABLE ACCESS	RESERVASHABITACIONES	BY INDEX ROWID	1	0
Filter Predicates				
AND				
RH.USUARIOS_ID=TO_NUMBER(:USUARIOS_ID)				
RH.FECHAINICIO>=TO_DATE(:FECHA1,'YYYY-MM-DD')				
RH.FECHAINICIO<=TO_DATE(:FECHA2,'YYYY-MM-DD')				
INDEX	RESERVASHABITACIONES_PK	UNIQUE SCAN	1	0
Access Predicates				
C.RESERVASHABITACIONES_ID=RH.ID				
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
C.SERVICIOS_ID=S.ID				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0

Como se puede apreciar, Oracle creó índices para las llaves primarias de las tablas de reservas habitaciones, servicios y usuarios, esto debido a que son utilizadas en los joins dentro de la consulta, por lo que los recorridos del join requieren menos operaciones y reduce el costo de la consulta.

Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 1
Transcurrido: 00:00:00.786
```

RFC6 - ANALIZAR LA OPERACIÓN DE HOTELANDES

Para todo el tiempo de operación de HotelAndes, indicar cuáles fueron las fechas de los días de mayor ocupación (mayor cantidad habitaciones ocupadas), las fechas de mayores ingresos (mayor cantidad de consumos realizados) y también las fechas de menor demanda (menor ocupación).

```
SELECT fechainicio, COUNT(*) AS ocupacion
FROM reservashabitaciones
GROUP BY fechainicio
ORDER BY ocupacion DESC
FETCH FIRST 10 ROWS ONLY;
SELECT c.fechaconsumo, SUM(c.sumatotal) AS ingresos
FROM consumos c
GROUP BY c.fechaconsumo
ORDER BY ingresos DESC
FETCH FIRST 10 ROWS ONLY;
SELECT fechainicio, COUNT(*) AS ocupacion
FROM reservashabitaciones
GROUP BY fechainicio
ORDER BY ocupacion ASC
FETCH FIRST 10 ROWS ONLY;
```

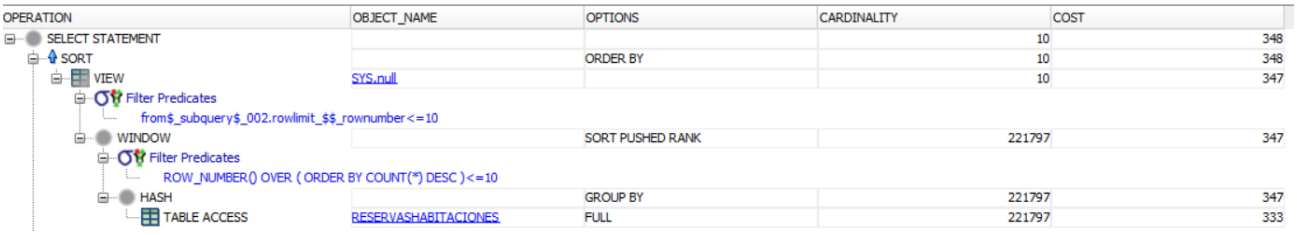
Índices

```
CREATE INDEX idx_rh_fechainicio3 ON reservashabitaciones(fechainicio);
CREATE INDEX idx_c_fechaconsumo2 ON consumos(fechaconsumo);
```

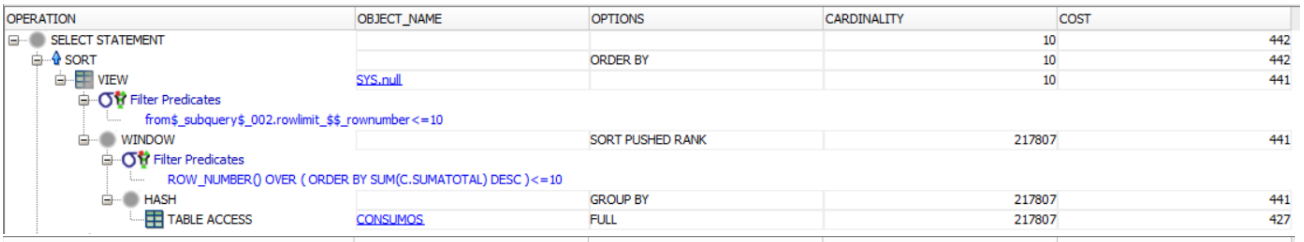
EL índice fechainicio y fecha consumo son útil para las cláusulas GROUP BY y ORDER BY.

Parámetros: La sentencia no tiene parámetros.

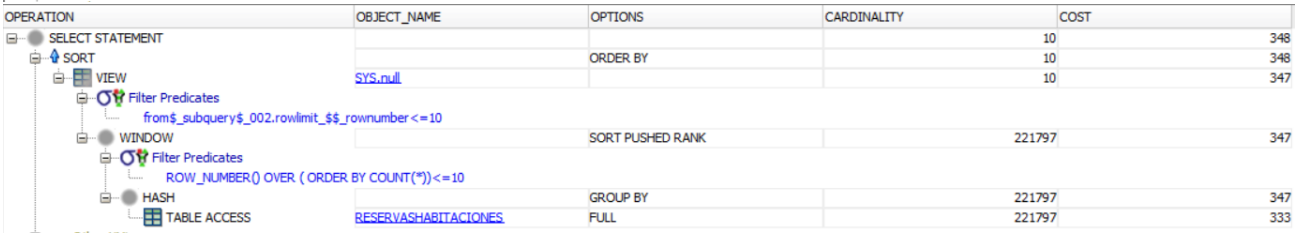
Plan de ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			10	348
SORT		ORDER BY	10	348
VIEW	SYS.null		10	347
Filter Predicates				
from$_subquery$_002.rowlimit_$$_rownumber<=10				
WINDOW		SORT PUSHED RANK	221797	347
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC)<=10				
HASH		GROUP BY	221797	347
TABLE ACCESS	RESERVASHABITACIONES	FULL	221797	333



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			10	442
SORT		ORDER BY	10	442
VIEW	SYS.null		10	441
Filter Predicates				
from$_subquery$_002.rowlimit_$$_rownumber<=10				
WINDOW		SORT PUSHED RANK	217807	441
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY SUM(C.SUMATOTAL) DESC)<=10				
HASH		GROUP BY	217807	441
TABLE ACCESS	CONSUMOS	FULL	217807	427



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			10	348
SORT		ORDER BY	10	348
VIEW	SYS.null		10	347
Filter Predicates				
from$_subquery$_002.rowlimit_$$_rownumber<=10				
WINDOW		SORT PUSHED RANK	221797	347
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*))<=10				
HASH		GROUP BY	221797	347
TABLE ACCESS	RESERVASHABITACIONES	FULL	221797	333

Como se puede apreciar en las imágenes, no se crearon índices para las consultas, esto puede ser debido a que no se hacen joins en ninguna consulta.

Tiempos de ejecución:

```
>>Query Run In:Resultado de la Consulta 8
Transcurrido: 00:00:00.344
```

RFC7 - ENCONTRAR LOS BUENOS CLIENTES

Se considera bueno a un cliente que ha estado en el hotel por lo menos dos semanas (no necesariamente en una sola estadía) o si ha consumido más de $15’000.000.oo, durante el último año de operación de HotelAndes. La información que se muestra en el resultado debe evidenciar el hecho de ser un buen cliente.

```
WITH DIAS AS(
SELECT usuarios.nombre as id, SUM(reservashabitaciones.fechafin -
reservashabitaciones.fechainicio) as diashotel
FROM usuarios, reservashabitaciones
WHERE reservashabitaciones.usuarios_id = usuarios.id
GROUP BY usuarios.nombre), gastos as(
select usuarios.nombre as id, sum(consumos.sumatotal) as gasto from consumos,
usuarios,reservashabitaciones
where usuarios.id = reservashabitaciones.usuarios_id and
consumos.reservashabitaciones_id  = reservashabitaciones.id
group by usuarios.nombre)
Select gastos.id,gastos.gasto,dias.diashotel from gastos,dias
where dias.id = gastos.id and (gastos.gasto >= 15000000 or dias.diashotel >=
14);
```

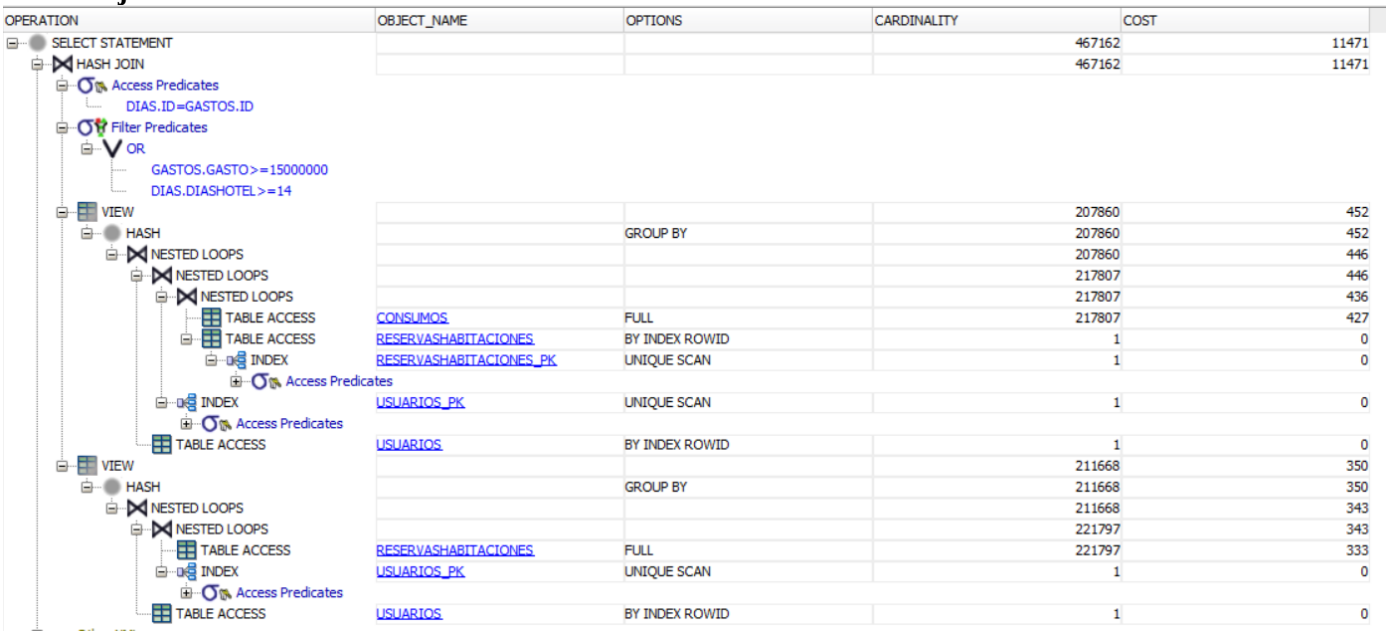
Indices:

Índices en las columnas usuarios.id, reservashabitaciones.usuarios_id, y consumos.reservashabitaciones_id por los JOINs y WHERE.
Posiblemente un índice compuesto en reservashabitaciones.fechainicio y reservashabitaciones.fechafin para cálculos de fechas.

CREATE INDEX idx_usuarios_id2 ON usuarios(id);
CREATE INDEX idx_rh_usuarios_id ON reservashabitaciones(usuarios_id);
CREATE INDEX idx_c_reservashabitaciones_id2 ON consumos(reservashabitaciones_id);

Parámetros: La sentencia no tiene parámetros.

Plan de ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			467162	11471
HASH JOIN			467162	11471
Access Predicates				
DIAS.ID=GASTOS.ID				
Filter Predicates				
OR				
GASTOS.GASTO>=15000000				
DIAS.DIASHOTEL>=14				
VIEW			207860	452
HASH		GROUP BY	207860	452
NESTED LOOPS			207860	446
NESTED LOOPS			217807	446
NESTED LOOPS			217807	436
TABLE ACCESS	CONSUMOS	FULL	217807	427
TABLE ACCESS	RESERVASHABITACIONES	BY INDEX ROWID	1	0
INDEX	RESERVASHABITACIONES_PK	UNIQUE SCAN	1	0
Access Predicates				
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0
VIEW			211668	350
HASH		GROUP BY	211668	350
NESTED LOOPS			211668	343
NESTED LOOPS			221797	343
TABLE ACCESS	RESERVASHABITACIONES	FULL	221797	333
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0

Como se puede apreciar, la aplicación creó el indice para la llave primaria de usuarios, esto debido a que es la columna presente en el join principal de la consulta, por lo que permite optimizar el recorrido por los datos de los usuarios.

Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 7
Transcurrido: 00:00:00.359
```

RFC8 - ENCONTRAR LOS SERVICIOS QUE NO TIENEN MUCHA DEMANDA

Encontrar los servicios que hayan sido solicitados menos de 3 veces semanales, durante el último año de operación de HotelAndes.

```
SELECT s.nombre,  COUNT(*)
FROM Consumos c, servicios s
WHERE (c.fechaconsumo >= ADD_MONTHS(SYSDATE, -12) and c.servicios_id = s.id)
GROUP BY s.nombre, TO_CHAR(c.FECHACONSUMO, 'IYYY-IW')
HAVING COUNT(*) < 3;
```

Indices

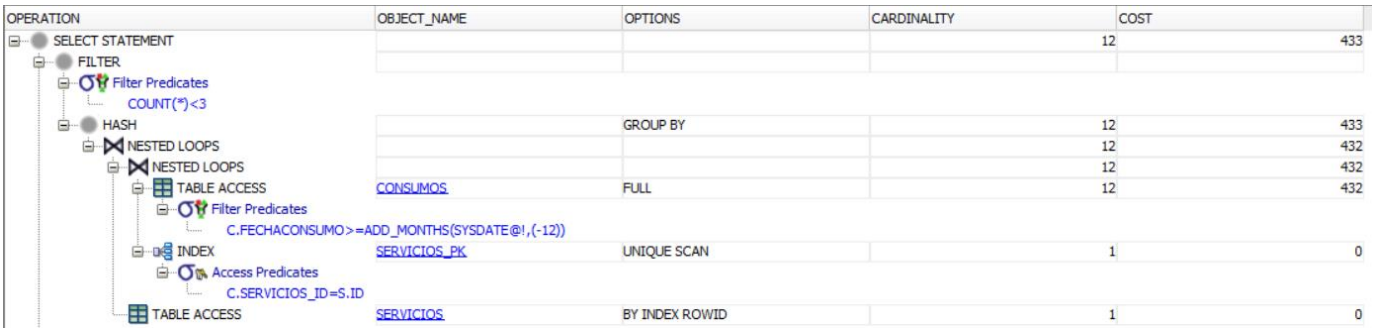
```
CREATE INDEX consumos_servicios_fecha_index ON Consumos (servicios_id,
```

fechaconsumo);

El índice compuesto agiliza la búsqueda de las filas que cumplen con las condiciones de JOIN y GROUP BY.

Parámetros: La sentencia no tiene parámetros.

Plan de ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			12	433
FILTER				
Filter Predicates				
COUNT(*)<3				
HASH		GROUP BY	12	433
NESTED LOOPS			12	432
NESTED LOOPS			12	432
TABLE ACCESS	CONSUMOS	FULL	12	432
Filter Predicates				
C.FECHACONSUMO>=ADD_MONTHS(SYSDATE@!,(-12))				
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
C.SERVICIOS_ID=S.ID				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0

La aplicación creó el indice para la llave primaria de servicios, esto debido a que es la columna presente en el join principal de la consulta, por lo que permite optimizar el recorrido por los datos de los servicios.
Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 12
Transcurrido: 00:00:00.403
```

RFC9 - CONSULTAR CONSUMO EN HOTELANDES

Se quiere conocer la información de los clientes que consumieron al menos una vez un determinado servicio del hotel, en un rango de fechas. Los resultados deben ser clasificados según un criterio deseado por quien realiza la consulta. En la clasificación debe ofrecerse la posibilidad de agrupamiento y ordenamiento de las respuestas según los intereses del usuario que consulta como, por ejemplo, por los datos del cliente, por fechay número de veces que se utilizó el servicio. Esta operación está disponible para el recepcionista y el gerente del hotel.

```
SELECT  u.nombre,u.numdocumento,s.nombre,COUNT(c.numconsumos) AS cuenta
FROM consumos c
JOIN usuarios u ON c.usuarios_id = u.id JOIN servicios s ON c.servicios_id
= s.id
WHERE    c.fechaconsumo    BETWEEN    TO_DATE(:fecha1,'YYYY-MM-DD')    AND
TO_DATE(:fecha2,'YYYY-MM-DD') AND s.id = :servicio_id
GROUP   BY   DECODE(:agrupamiento,   'usuario',   u.nombre,   'documento',
u.numdocumento),u.nombre,s.nombre, u.numdocumento
ORDER    BY    DECODE(:ordenamiento,    'usuario',    u.nombre,'documento',
u.numdocumento, 'count', cuenta);
```

Indices

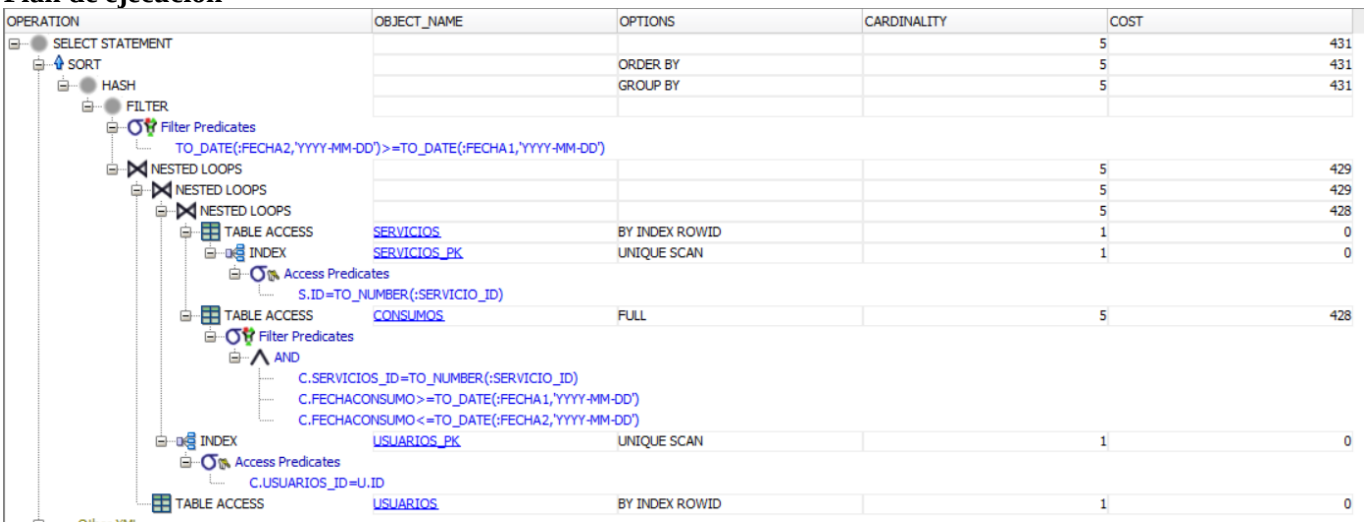
```
CREATE INDEX fechaconsumo _index ON consumos (fechaconsumo);

CREATE INDEX usuarios_servicios_index ON consumos (usuarios_id, servicios_id);
```

Fechaconsumo ayuda a filtrar por fechas con BETWEEN TO_DATE(:fecha1, 'YYYY-MM-DD') AND TO_DATE(:fecha2, 'YYYY-MM-DD'), y usuarios_servicios **f**acilita las operaciones de JOIN entre usuarios y servicios.

Parámetros: Agrupamiento, ordenamiento, fecha 1, fecha 2, servicio. Donde fecha 1 < fecha 2. Los parametros de agrupamiento y ordenamiento son opcionales, pero sus valores pueden ser: null, ‘usuario’, ‘documento’ y ‘count’ solo en el caso del ordenamiento.

Plan de ejecución



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	431
SORT		ORDER BY	5	431
HASH		GROUP BY	5	431
FILTER				
Filter Predicates				
TO_DATE(:FECHA2,'YYYY-MM-DD')>=TO_DATE(:FECHA1,'YYYY-MM-DD')				
NESTED LOOPS			5	429
NESTED LOOPS			5	429
NESTED LOOPS			5	428
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
S.ID=TO_NUMBER(:SERVICIO_ID)				
TABLE ACCESS	CONSUMOS	FULL	5	428
Filter Predicates				
AND				
C.SERVICIOS_ID=TO_NUMBER(:SERVICIO_ID)				
C.FECHACONSUMO>=TO_DATE(:FECHA1,'YYYY-MM-DD')				
C.FECHACONSUMO<=TO_DATE(:FECHA2,'YYYY-MM-DD')				
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
C.USUARIOS_ID=U.ID				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0

Como se puede apreciar en la imagen, la aplicación creó el índice para la llave primaria de usuarios, esto debido a que es la columna presente en el join principal de la consulta, por lo que permite optimizar el recorrido por los datos de los usuarios.
Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 1
Transcurrido: 00:00:00.786
```

RFC10 - CONSULTAR CONSUMO EN HOTELANDES – RFC9-V2

Se quiere conocer la información de los clientes que NO consumieron ninguna vez un determinado servicio del hotel, en un rango de fechas. Los resultados deben ser clasificados según un criterio deseado por quien realiza la consulta. En la clasificación debe ofrecerse la posibilidad de agrupamiento y ordenamiento de las respuestas según los intereses del usuario que consulta como, por ejemplo, por los datos del cliente, por fecha, y por servicio. Esta operación está disponible para el recepcionista y el gerente del hotel.

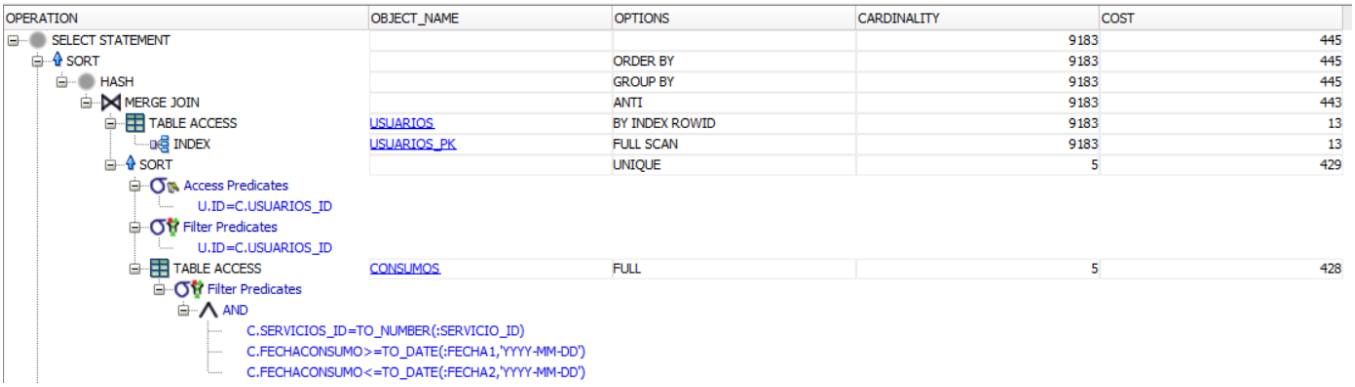
Indices

Índices en las columnas utilizadas para JOINs y WHERE (usuarios.id, consumos.usuarios_id, servicios.id, consumos.servicios_id, y consumos.fechaconsumo).

```
CREATE INDEX idx_c_usuarios_id ON consumos(usuarios_id);
 CREATE INDEX idx_c_servicios_id3 ON consumos(servicios_id);
CREATE INDEX idx_c_fechaconsumo4 ON consumos(fechaconsumo);
```

Parametros: Agrupamiento, ordenamiento, fecha 1, fecha 2, id de servicio. Donde fecha 1 < fecha 2. Los parametros de agrupamiento y ordenamiento son opcionales, pero sus valores pueden ser: null, ‘usuario’, ‘documento’ y ‘count’ solo en el caso del ordenamiento.

Plan de ejecución



Como se puede apreciar en la imagen, para esta consulta no se generan índices automaticamente.

Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 6
Transcurrido: 00:00:00.346
```

RFC11 - CONSULTAR FUNCIONAMIENTO

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. Las respuestas deben ser sustentadas por el detalle de las reservas y consumos correspondiente. Esta operación es realizada el gerente general de HotelAndes.

Indices

Índices en consumos.fechaconsumo y reservashabitaciones.fechainicio para las funciones de agrupación por semana.
Índices en consumos.servicios_id y reservashabitaciones.habitaciones_id para los JOINs.

CREATE INDEX idx_c_fechaconsumo5 ON consumos(fechaconsumo);
CREATE INDEX idx_rh_fechainicio4 ON reservashabitaciones(fechainicio);

Parametros: Razón, que determina la razón que se tiene en cuenta para considerar los consumos, pueden ser por la suma total de dinero si su valor es ‘dinero’, puede ser la cantidad de consumos si es ‘cant’ y la suma total de numero de productos consumidos si es ‘num’.

Plan de ejecución

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1607
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D703D_CF93870	(CURSOR DURATION MEMORY)		
HASH		GROUP BY	217807	434
TABLE ACCESS	CONSUMOS	FULL	217807	427
NESTED LOOPS			1	1173
NESTED LOOPS			1	1173
HASH JOIN			1	1173
Access Predicates				
AND				
NESTED LOOPS			1	784
NESTED LOOPS			1	784
HASH JOIN			1	784
Access Predicates				
AND				
VIEW			1	395
HASH		GROUP BY	1	395
VIEW			217807	388
VIEW			217807	388
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
S.SERVICIOS_ID=SV.ID				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0
VIEW			217807	388
TABLE ACCESS	SYS.SYS_TEMP_0FD9D703D_CF93870	FULL	217807	388
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
S1.SERVICIOS_ID=SV1.ID				
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	0

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	898
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D703E_CF93870	(CURSOR DURATION MEMORY)		
HASH		GROUP BY	221797	340
TABLE ACCESS	RESERVASHABITACIONES	FULL	221797	333
NESTED LOOPS			1	558
NESTED LOOPS			1	558
HASH JOIN			1	558
Access Predicates				
AND				
NESTED LOOPS			1	374
NESTED LOOPS			1	374
HASH JOIN			1	374
Access Predicates				
AND				
VIEW			1	190
HASH		GROUP BY	1	190
VIEW			221797	183
VIEW			221797	183
TABLE ACCESS	SYS.SYS_TEMP_0FD9D703E_CF93870	FULL	221797	183
INDEX	HABITACIONES_PK	UNIQUE SCAN	1	0
Access Predicates				
TABLE ACCESS	HABITACIONES	BY INDEX ROWID	1	0
VIEW			221797	183
TABLE ACCESS	SYS.SYS_TEMP_0FD9D703E_CF93870	FULL	221797	183
INDEX	HABITACIONES_PK	UNIQUE SCAN	1	0
Access Predicates				
H1.HABITACIONES_ID=HAB1.ID				
TABLE ACCESS	HABITACIONES	BY INDEX ROWID	1	0

Índices existentes: Como se puede apreciar en las imágenes, en ambas consultas se crearon índices para las id de las habitaciones, esto debido a que es la columna principal del join de la consulta, por lo que hacer un índice para esta reduce el número de recorridos sobre las tablas.

Tiempo de ejecución:

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 158 en 0,224 segundos

	YEAR	WEEK	START_DATE	END_DATE	SERVICIO_MAS_CONSUMIDO_NOMBRE	MAX_CONSUMO_SERVICIO	SERVICIO_MENOS_CONSUMIDO_NOMBRE	MIN_
1	2021	38	19/09/21	25/09/21	Servicio de Lavanderia	37954530	Servicio de Limpieza en Seco	
2	2020	27	28/06/20	04/07/20	Guarderia Infantil	31018843	Salas de Reuniones	
3	2021	15	11/04/21	17/04/21	Acceso a la Playa	37930047	Espacios con Vistas al Mar	
4	2021	45	07/11/21	13/11/21	Alquiler de Equipo de Golf	37985276	Wifi Gratis	
5	2020	6	02/02/20	08/02/20	Transporte al Aeropuerto	35720729	Centro de Arte y Cultura	
6	2020	49	29/11/20	05/12/20	Tienda de Regalos y Souvenirs	34040549	Servicio de Conserjería	
7	2021	47	21/11/21	27/11/21	Área de Juegos para Niños	40346800	Alquiler de Bicicletas	
8	2021	19	09/05/21	15/05/21	Servicio de Conserjería	40102151	Wifi Gratis	
9	2021	46	14/11/21	20/11/21	Admisión de Mascotas	38652595	Espacios con Vistas al Mar	
10	2020	16	12/04/20	18/04/20	Gimnasio	37812256	Concierge de Playa	
11	2023	19	07/05/23	13/05/23	Centro de Arte y Cultura	1242089	Espacios Adaptadados para Personas con Discapacidad	
12	2023	9	26/02/23	04/03/23	Centro de Negocios	778299	Programa de Recompensas para Huéspedes Frecuentes	

```
>>Query Run In:Resultado de la Consulta 18
Transcurrido: 00:00:01.977
```

RFC12 - CONSULTAR LOS CLIENTES EXCELENTES

Los clientes excelentes son de tres tipos: aquellos que realizan estancias (las estancias están delimitadas por un check in y su respectivo check out) en HotelAndes al menos una vez por trimestre, aquellos que siempre consumen por lo menos un servicio costoso (Entiéndase como costoso, por ejemplo, con un precio mayor a $300.000.oo) y aquellos que en cada estancia consumen servicios de SPA o de salones de reuniones con duración mayor a 4 horas. Esta consulta retorna toda la información de dichos clientes, incluyendo aquella que v justifica su calificación como clientes excelentes. Esta operación es realizada únicamente por el gerente general de HotelAndes

Indices

Índices en las columnas usuarios.id, reservashabitaciones.usuarios_id, consumos.usuarios_id, reservasservicios.usuarios_id por los subqueries y condiciones HAVING.
Índices en servicios.costoporunidad, servicios.nombre, y servicios.tiposervicio debido a las condiciones WHERE.

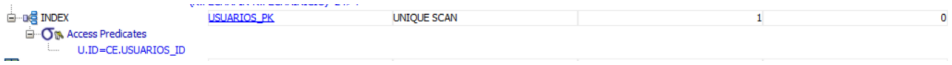
CREATE INDEX idx_s_costoporunidad2 ON servicios(costoporunidad);
CREATE INDEX idx_s_nombre ON servicios(nombre);
CREATE INDEX idx_s_tiposervicio2 ON servicios(tiposervicio);

Parametros: La sentencia no tiene parametros.

Plan de ejecución

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
NESTED LOOPS			260173	30368
VIEW			260173	30357
SORT		UNIQUE	260173	30357
UNION-ALL				
HASH		GROUP BY	221797	1655
Filter Predicates				
VIEW	SYS.VM_NWVW_1		221797	340
HASH		GROUP BY	221797	340
FILTER				
Filter Predicates				
SORT		GROUP BY	13068	673
HASH JOIN			13068	432
Access Predicates				
TABLE ACCE	SERVICIOS	FULL	3	4
TABLE ACCE	CONSUMOS	FULL	217807	428
FILTER				
Filter Predicates				
SORT		GROUP BY	25308	6056
HASH JOIN			25308	248
Access Predicates				
TABLE ACCE	SERVICIOS	FULL	5	4
Filter Predicates				
OR				
TABLE ACCE	RESERVASSERVICIOS	FULL	253075	243
Filter Predicates				
(R.FECHAFIN-R.FECHAINICIO)*24>4				
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	0

Indices existentes:



La aplicación creó el indice para la llave primaria de usuarios, esto debido a que es la columna presente en el join principal de la consulta, por lo que permite optimizar el recorrido por los datos de los usuarios.

Tiempo de ejecución:

```
>>Query Run In:Resultado de la Consulta 19
Transcurrido: 00:00:00.410
```