María Alejandra Estrada García - 202021060

Marilyn Stephany Joven Fonseca – 202021346

Santiago Martínez Novoa - 202112020

Informe de documentación entrega 2

Correcciones entrega 1:

Normalización:

Relaciones

Hotel (nombre, ciudad)

Usuario (id_usuario, nombre, correo, rol,nombreHotel)

Reserva (idReserva, fechainicio, fechafin, duracion, numAcompanantes, idUsuario, idHabitacion,idPlanConsumo,idCuenta)

Habitacion (idHabitacion, capacidad,disponible, tipo, dotacion, precioNoche,nombreHotel)

Servicio (idServicio, nombre, descripción, costoTotal, nombreHotel, idPlanConsumo)

PlanesC (idPlanConsumo, nombre, descuentoAlojamiento, descuentoBar, descuentoRestaurante, descuentoServicio, costoFijo, fechaInicial, duracion, valorFinal, valido)

Producto (idProducto, precio)

CuentaConsumo (idCuenta, estado, checkIn, checkOut, idHabitacion)

S_Consumidos (idServicio, idCuenta)

Reservas_Servicios (idServicio, idHabitacion, fechaReserva, duracion)

Productos_C (idCuenta, idProducto)

Presatamos (idServicio, utensilio, cantidad, idCuenta, enPrestamo, costoPenalizacion, estado)

ProductosPlan (idPlanConsumo, idProducto, capacidad)

Justificación

- 1FN: El modelo se encuentra en primera forma normal debido a que ninguna de las relaciones presenta atributos que puedan tomar más de un valor.

- 2FN: El modelo se encuentra en segunda forma normal, porque está en 1FN y además los atributos no primos dependen completamente de las llaves candidatas.

- 3FN: El modelo está en 3FN, porque no se presentan relaciones transitivas de manera que no hay atributos no primos que no dependen directamente de las llaves candidatas.

- FNBC: Porque está en 3FN, y las llaves son simples. Todas las relaciones cuentan con una llave primaria, por ende, se encuentran en FNBC. Además, las llaves candidatas no son compuestas ni se sobreponen.

Análisis

El diseño de la aplicación debe permitir la introducción de nuevos requerimientos, gracias a las relaciones generadas es posible realizar operaciones tales como mostrar, analizar y consultar diferentes datos que deben estar interconectados entre sí. Se puede considerar como un buen diseño al tener en cuenta la clasificación de las diferentes entidades, pues con las entidades principales se puede almacenar correctamente la información referente a los diferentes sectores que pueden manejar información del Hotel. La correcta definición de las entidades permite la interconexión de los datos para realizar consultas en las que se requiere información de más de una entidad. Se puede considerar un modelo flexible al abarcar diferentes funcionalidades, de manera que se puede agregar la ejecución de operaciones extra. Por último, se considera un diagrama completo al poder almacenar la información necesaria y exigida por los usuarios.

Diseño de la aplicación

Índices

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

- Necesidad de índices: Sí, es necesario crear un índice en la tabla de cuentas_c para acelerar las consultas que suman el dinero recolectado por servicios en cada habitación.
- Justificación: La consulta busca obtener información sobre el dinero recolectado en cada habitación, y esta operación generalmente involucra búsquedas en función de la habitación. Por lo tanto, un índice secundario en la columna idhabitacion de la tabla cuentas_c ayudará a acelerar estas búsquedas.
- Tipo de índice: Índice secundario.

CREATE INDEX cuentas_habit_idx ON

cuentas_c (

idhabitacion

);

RFC2 - MOSTRAR LOS 20 SERVICIOS MÁS POPULARES:

- Necesidad de índices: Sí, es necesario crear un índice en las tablas relevantes para acelerar la consulta que busca los servicios más populares.
- Justificación: Para determinar los servicios más populares, se requerirá información sobre el consumo de servicios. Un índice en la columna relevante (como el identificador del servicio) en las tablas de consumos será útil para agilizar la consulta.
- Tipo de índice: Índice secundario en la tabla de consumos o en la tabla de servicios, dependiendo de dónde se realice la búsqueda.

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

- Necesidad de índices: Sí, es necesario crear un índice en la tabla de habitaciones para acelerar la consulta que calcula el índice de ocupación.
- Justificación: La consulta busca calcular el índice de ocupación de cada habitación, lo que involucra búsquedas en función de la habitación y fechas. Un índice secundario en la columna idhabitacion de la tabla cuentas_c ayudará a acelerar estas búsquedas.
- Tipo de índice: Índice secundario.

RFC4 - MOSTRAR LOS SERVICIOS QUE CUMPLEN CON CIERTA CARACTERÍSTICA:

- Necesidad de índices: Sí, puede ser necesario crear índices en función de las características específicas utilizadas en la consulta. Por ejemplo, si se filtra por precio o fecha, se pueden crear índices en esas columnas.
- Justificación: La necesidad de índices dependerá de las características utilizadas en la consulta. Si se filtra por características que involucran columnas en las tablas, se pueden crear índices en esas columnas para acelerar la búsqueda.
- Tipo de índice: Índice secundario en las columnas relevantes según las características utilizadas.

CREATE INDEX reservas_fechas_idx ON

reservas (

fechainicio,

fechafin

);

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

- Necesidad de índices: Sí, es necesario crear índices en las tablas relevantes para acelerar la consulta que busca el consumo de un usuario en un rango de fechas.
- Justificación: La consulta busca obtener información sobre el consumo de un usuario en función de su identificación y fechas. Un índice secundario en la columna de identificación de usuario y en la columna de fecha en las tablas de consumo ayudará a acelerar estas búsquedas.
- Tipo de índice: Índice secundario en las tablas de consumo.

```
CREATE INDEX usuarios_nombre_idx ON
  usuarios (
    nombre
  );
```

RFC6 - ANALIZAR LA OPERACIÓN DE HOTELANDES:

- Necesidad de índices: Puede no ser necesario crear índices específicos para este requerimiento, ya que se trata de operaciones de análisis histórico y estadísticas.
- Justificación: Las fechas de mayor ocupación, ingresos y menor demanda se pueden calcular a partir de los datos históricos sin la necesidad de índices adicionales, ya que estas operaciones son consultas ocasionales.
- Tipo de índice: No se requieren índices específicos.

RFC7 - ENCONTRAR LOS BUENOS CLIENTES:

- Necesidad de índices: Sí, es necesario crear índices en las tablas relevantes para acelerar la consulta que busca a los "buenos clientes".
- Justificación: La consulta busca identificar a los "buenos clientes" en función de criterios específicos, como estadías largas o alto consumo. Un índice secundario en las columnas relevantes, como el ID de cliente y el consumo, ayudará a acelerar estas búsquedas.
- Tipo de índice: Índice secundario en las tablas de cuentas_c y clientes.

```
CREATE INDEX r_servicio_fecha_idx ON
  reservas_servicios (
    fechareserva
  );
```

RFC8 - ENCONTRAR LOS SERVICIOS QUE NO TIENEN MUCHA DEMANDA:

- Necesidad de índices: Sí, es necesario crear índices en las tablas relevantes para acelerar la consulta que busca los servicios con poca demanda.

- Justificación: La consulta busca encontrar servicios con demanda baja, lo que generalmente implica buscar registros de consumo. Un índice secundario en la columna relevante (como el ID de servicio) en la tabla de consumos ayudará a acelerar la búsqueda.
- Tipo de índice: Índice secundario en la tabla de consumos.

```
CREATE INDEX r_servicio_fecha_idx ON
  reservas_servicios (
    fechareserva
  );
```

RFC9 - CONSULTAR CONSUMO EN HOTELANDES:

- Necesidad de índices: Sí, se necesitan índices en las tablas relevantes para acelerar la consulta que busca información sobre los clientes que consumieron un servicio específico en un rango de fechas.
- Justificación: La consulta busca información de los clientes que consumieron un servicio específico, lo que generalmente implica buscar registros de consumo. Se requiere un índice secundario en la columna relevante (como el ID del servicio) en la tabla de consumos para acelerar la búsqueda.
- Tipo de índice: Índice secundario en la tabla de consumos.

RFC10 - CONSULTAR CONSUMO EN HOTELANDES – RFC9-V2:

- Necesidad de índices: Sí, se necesitan índices en las tablas relevantes para acelerar la consulta que busca información sobre los clientes que NO consumieron un servicio específico en un rango de fechas.
- Justificación: Al igual que en RFC9, esta consulta busca información de los clientes en función de registros de consumo. Se requiere un índice secundario en la columna relevante (como el ID del servicio) en la tabla de consumos para acelerar la búsqueda.
- Tipo de índice: Índice secundario en la tabla de consumos.

RFC11 - CONSULTAR FUNCIONAMIENTO:

- Necesidad de índices: Sí, es necesario crear índices en las tablas relevantes para acelerar la consulta que busca el servicio más consumido, el servicio menos consumido, las habitaciones más solicitadas y las habitaciones menos solicitadas para cada semana del año.
- Justificación: La consulta busca información detallada sobre la ocupación de las habitaciones y el consumo de servicios en función de las semanas del año. Se requieren índices secundarios en las columnas relevantes (por ejemplo, ID de servicio y ID de habitación) en las tablas de consumos y reservas para acelerar la búsqueda.
- Tipo de índice: Índice secundario en las tablas de consumos y reservas.

RFC12 - CONSULTAR LOS CLIENTES EXCELENTES:

- Necesidad de índices: Sí, es necesario crear índices en las tablas relevantes para acelerar la consulta que busca a los clientes excelentes en función de los criterios especificados.
- Justificación: La consulta busca información sobre los clientes excelentes según varios criterios, como estadías trimestrales, consumo costoso y duración de servicios de SPA o salones de reuniones. Se requieren índices secundarios en las columnas relevantes, como el ID del cliente y las columnas relacionadas con los criterios, para acelerar la búsqueda.
- Tipo de índice: Índice secundario en las tablas de clientes, consumos y reservas según los criterios utilizados.

Los índices automáticos son creados por Oracle generalmente en las claves primarias y únicas de las tablas. Esto tiene utilidad tanto para mantener la integridad de los datos como para acelerar las consultas de búsqueda por clave primaria. Adicionalmente, Oracle implementa la creación de índices en las claves foráneas, lo cual optimiza el desempeño de las operaciones de JOIN en tablas relacionadas.

Finalmente, los índices automáticos generados por Oracle desempeñan un papel vital al mantener la integridad de la base de datos y al mejorar el rendimiento de las operaciones de búsqueda y JOIN. Además, para agilizar las búsquedas relacionadas con los requerimientos funcionales identificados, se hace imprescindible crear índices adicionales en campos específicos. La elección del tipo de índice (primario o secundario) dependerá del tipo de consulta y los campos utilizados en la búsqueda.

Diseño y cargue masivo de datos

El diseño y carga de datos fue realizado usando la aplicación de EXCEL, el cual se logran crear datos pertinentes y automáticos de acuerdo a los atributos de cada una de las tablas. Los datos de los atributos se pusieron en cada una de las columnas del insert_x (donde x significa el nombre de la tabla -relación- de la base de datos), posteriormente se obtuvieron las inserciones por medio de la concatenación de los datos de cada columna y se subieron a SQLDeveloper. Los datos fueron creados para verificar adecuadamente las reglas de negocio del proyecto.

Las inserciones de cada una de las tablas se encuentran en la carpeta Datos dentro del repositorio del proyecto. Se crearon aproximadamente 750.000 datos y confirmados en la base de datos.

Datos>insertBase.xlsx.

Construcción de la aplicación, ejecución de pruebas y análisis de resultados

Pruebas en interfaz de cada requerimiento

Requerimiento 1

IdHabitacion	Dinero Recolectado
1	15001000
4	15000000

Requerimiento 3

IdHabitacion	Usuarios En Habitacion	Indice de Ocupacion
1	1	0.27

Requerimiento 6

Fecha Mayor Demanda	Cantidad demanda
2023-11-28 00:00:00.0	1

Requerimiento 2

Nombre	Veces Consumido
Piscina	1
Gimnasio	1
Spa	1

REQ 5

idCuenta	estado	checkin	checkout	
1	true	2023-11-28 00:00:00.0	2023-11-28 00:00:00.0	Editar Borrar
2	false	2023-06-29 00:00:00.0	2023-07-29 00:00:00.0	Editar Borrar
2	false	2023-06-29 00:00:00.0	2023-07-29 00:00:00.0	Editar Borrar
1	true	2023-11-28 00:00:00.0	2023-11-28 00:00:00.0	Editar Borrar
1	true	2023-11-28 00:00:00.0	2023-11-28 00:00:00.0	Editar Borrar
1	true	2023-11-28 00:00:00.0	2023-11-28 00:00:00.0	Editar Borrar
2	false	2023-06-29 00:00:00.0	2023-07-29 00:00:00.0	Editar Borrar

Requerimiento 8

Nombre Servicio	Veces Solicitado
Gimnasio	1
Piscina	2

Requerimiento 9

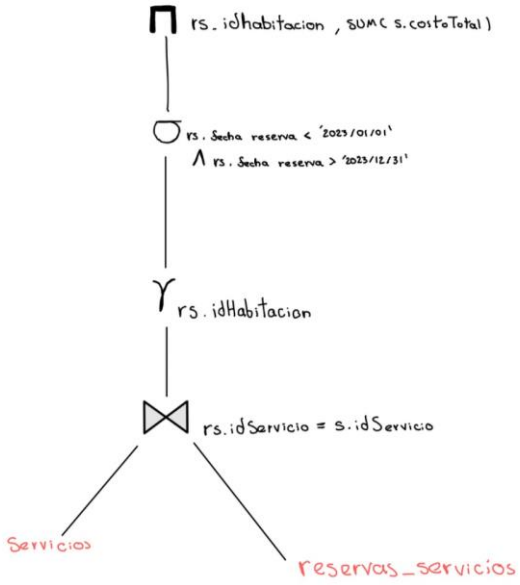
idUsuario	Nombre	Servicio	Veces consumido
1	Carolina Bennedeti	Gimnasio	1
1	Carolina Bennedeti	Piscina	1

Requerimiento 11

Id Servicio más consumido	Id Servicio menos consumido	Id habitacion más opcupada	Id habitacion menos opcupada
1	500	501	1

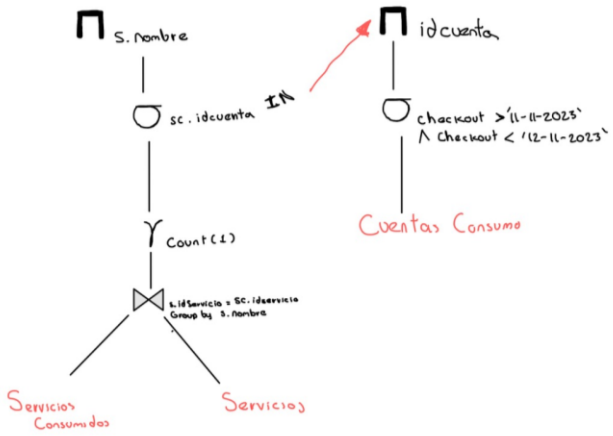
Planes de ejecución Planeado vs Oracle

Req 1:



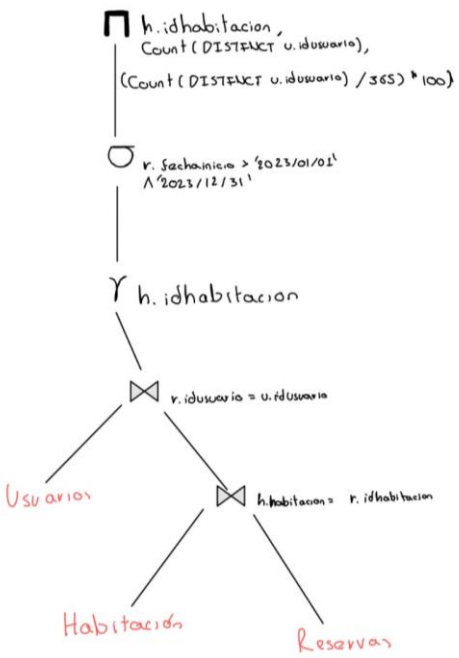
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	6
HASH		GROUP BY	2	6
MERGE JOIN			3	5
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	9	2
INDEX	SERVICIOS_PK	FULL SCAN	9	1
SORT		JOIN	3	3
Access Predicates				
ITEM_1=S.IDSERVICIO				
Filter Predicates				
ITEM_1=S.IDSERVICIO				
VIEW	SYS.VW_GBF_5		3	2
HASH		GROUP BY	3	2
TABLE ACCESS	RESERVAS_SERVICIOS	BY INDEX ROWID ...	3	2
Filter Predicates				
AND				
RS.FECHARESERVA<=TIMESTAMP' 2023-12-31 12:00:00'				
RS.FECHARESERVA>=TIMESTAMP' 2023-01-01 12:00:00'				
INDEX	RESERVAS_SERVICIOS_PK	FULL SCAN	3	1

Req 2:



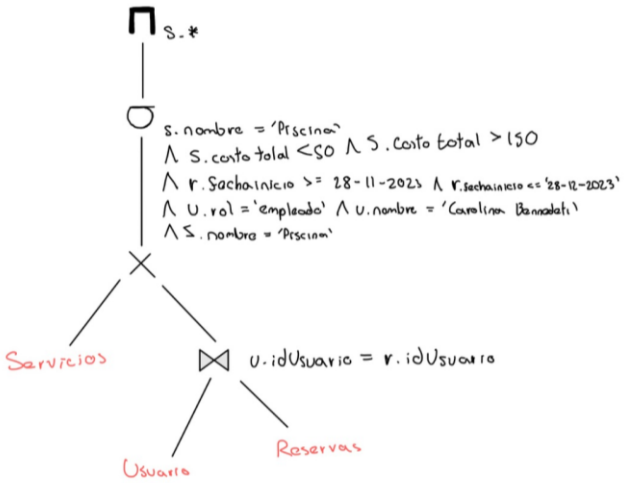
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			20	10
SORT		ORDER BY	20	10
VIEW	SYS.null		20	9
Filter Predicates				
from$_subquery$_005.rowlimit_$$_rownumber<=20				
WINDOW		SORT PUSHED RANK	1	9
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC)<=20				
HASH		GROUP BY	1	9
HASH JOIN			3	7
Access Predicates				
S.IDSERVICIO=SC.IDSERVICIO				
NESTED LOOPS			3	7
NESTED LOOPS			3	7
STATISTICS COLLEC				
MERGE JOIN			3	4
TABLE ACCE	CUENTAS_C	BY INDEX ROWID	1	2
Filter Predicates				
AND				
CHECKOUT>=TIMESTAMP' 2023-11-11 00:00:00.000000000'				
CHECKOUT<=TIMESTAMP' 2023-11-12 00:00:00.000000000'				
INDEX	CUENTAS_C_PK	FULL SCAN	1	1
SORT		JOIN	3	2
Access Predicates				
SC.IDCUENTA=IDCUENTA				
Filter Predicates				
SC.IDCUENTA=IDCUENTA				

Req 3:



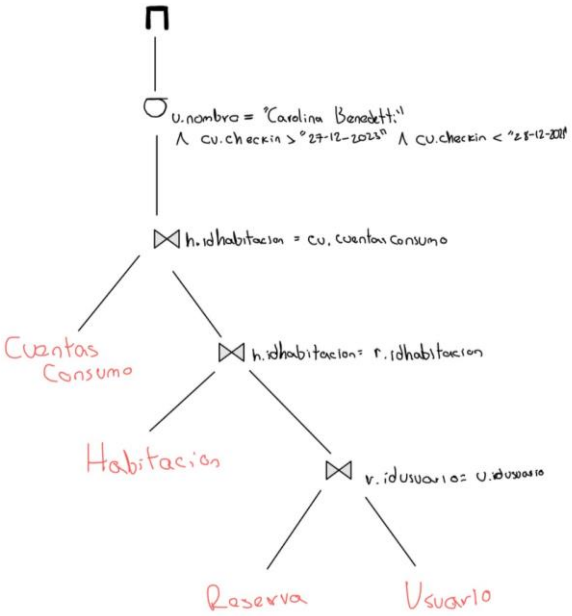
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	5
HASH		GROUP BY	1	5
VIEW	SYS.VM_NWVW_1		1	5
HASH		GROUP BY	1	5
TABLE ACCESS	RESERVAS	FULL	1	4
Filter Predicates				
AND				
R.FECHAINICIO>=TO_DATE(' 2023-01-01 12:00:00', 'syyyy-mm-dd hh24:mi:ss')				
R.FECHAINICIO<=TO_DATE(' 2023-12-31 12:00:00', 'syyyy-mm-dd hh24:mi:ss')				

Req 4:



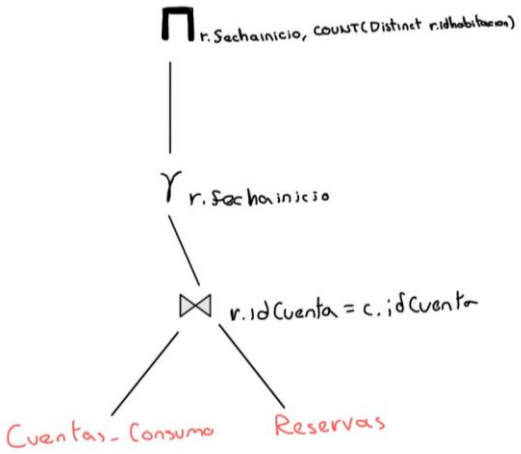
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	9
MERGE JOIN		CARTESIAN	1	9
HASH JOIN			1	5
Access Predicates				
R.IDUSUARIO=U.IDUSUARIO				
NESTED LOOPS			1	5
NESTED LOOPS			1	5
STATISTICS COLLECTOR				
TABLE ACCESS	RESERVAS	FULL	1	4
Filter Predicates				
AND				
R.FECHAINICIO>=TO_DATE(' 2023-11-28 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
R.FECHAFIN<=TO_DATE(' 2023-12-28 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
R.IDUSUARIO=U.IDUSUARIO				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	1
Filter Predicates				
AND				
U.NOMBRE='Carolina Bennedeti'				
U.ROL='empleado'				
TABLE ACCESS	USUARIOS	FULL	1	1
Filter Predicates				
AND				
U.NOMBRE='Carolina Bennedeti'				
U.ROL='empleado'				
BUFFER		SORT	1	8
TABLE ACCESS	SERVICIOS	FULL	1	4

Req 5:



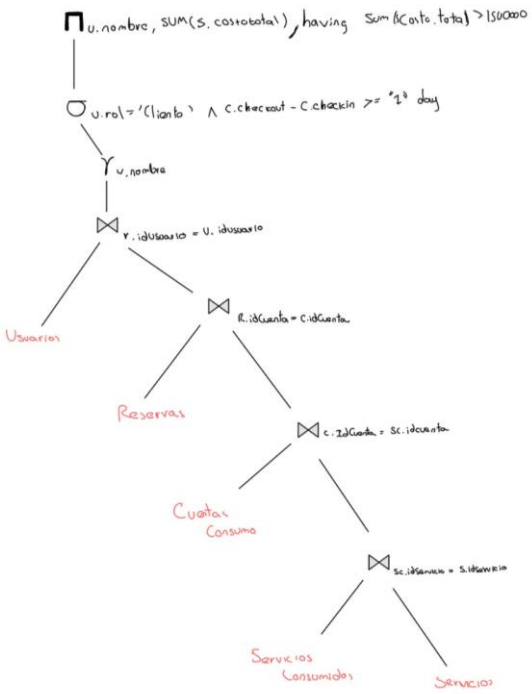
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	9
HASH JOIN			1	9
Access Predicates				
R.IDUSUARIO=U.IDUSUARIO				
NESTED LOOPS			1	9
NESTED LOOPS			1	9
STATISTICS COLLECTOR				
HASH JOIN			1	8
Access Predicates				
CU.IDHABITACION=R.IDHABITACION				
TABLE ACCESS	CUENTAS_C	FULL	1	4
Filter Predicates				
AND				
CU.CHECKIN>=TIMESTAMP' 2023-01-01 00:00:00'				
CU.CHECKIN<=TIMESTAMP' 2023-12-12 00:00:00'				
TABLE ACCESS	RESERVAS	FULL	1	4
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
R.IDUSUARIO=U.IDUSUARIO				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	1
Filter Predicates				
U.NOMBRE='RAQUEL'				
TABLE ACCESS	USUARIOS	FULL	1	1
Filter Predicates				
U.NOMBRE='RAQUEL'				

Req 6:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
SORT		ORDER BY	1	7
VIEW	SYS.null		1	6
Filter Predicates				
from$_subquery$_004.rowlimit_$$_rownumber<=1				
WINDOW		SORT PUSHED RANK	1	6
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT($vm_col_2) DESC)<=1				
HASH		GROUP BY	1	6
VIEW	SYS.VM_NWVW_1		1	5
HASH		GROUP BY	1	5
TABLE ACCESS	RESERVAS	FULL	1	4

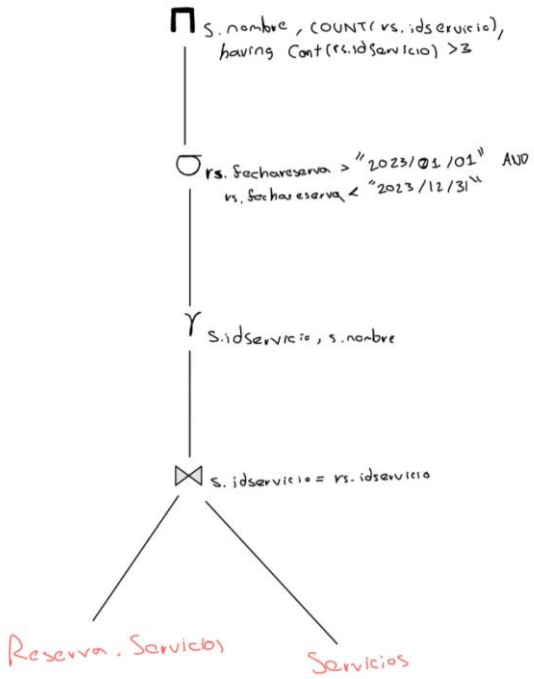
Req 7:



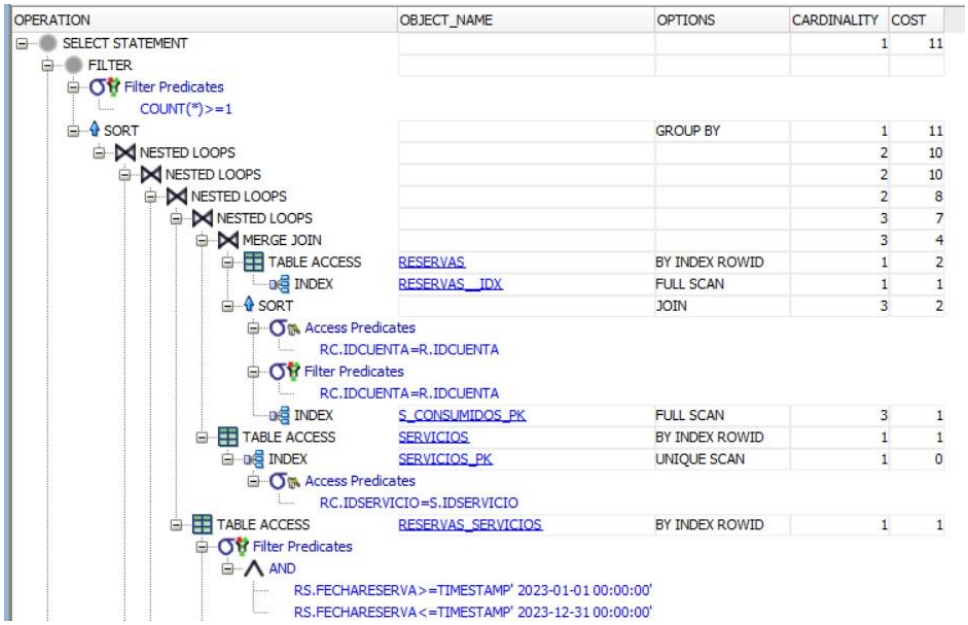
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	10
HASH JOIN			3	10
Access Predicates				
SC.IDSERVICIO=S.IDSERVICIO				
NESTED LOOPS			3	10
NESTED LOOPS			3	10
STATISTICS COLLECTOR				
HASH JOIN			3	7
Access Predicates				
C.IDCUENTA=SC.IDCUENTA				
NESTED LOOPS			3	7
STATISTICS COLLECTOR				
HASH JOIN			1	6
Access Predicates				
U.IDUSUARIO=R.IDUSUARIO				
NESTED LOOPS			1	6
STATISTICS				
HASH JC			1	5
Access Predicates				
R.IDCUENTA=C.IDCUENTA				
NES			1	5
	RESERVAS	FULL	1	4
	CUENTAS_C	BY INDEX ROWID	1	1
	CUENTAS_C_PK	UNIQUE SCAN	1	0
Access Predicates				
R.IDCUENTA=C.IDCUENTA				
TAB	CUENTAS_C	FULL	1	1
TABLE ACCE	USUARIOS	BY INDEX ROWID	1	1
Filter Predicates				
U.ROL='empleado'				


INDEX USUARIOS_PK UNIQUE SCAN 1 0
Access Predicates
U.IDUSUARIO=R.IDUSUARIO
TABLE ACCESS USUARIOS FULL 1 1
Filter Predicates
U.ROL='empleado'
INDEX S_CONSUMIDOS_PK FULL SCAN 3 1
Access Predicates
C.IDCUENTA=SC.IDCUENTA
Filter Predicates
C.IDCUENTA=SC.IDCUENTA
INDEX S_CONSUMIDOS_PK FULL SCAN 3 1
INDEX SERVICIOS_PK UNIQUE SCAN 1 0
Access Predicates
SC.IDSERVICIO=S.IDSERVICIO
TABLE ACCESS SERVICIOS BY INDEX ROWID 1 1
TABLE ACCESS SERVICIOS FULL 1 1

Req 8:

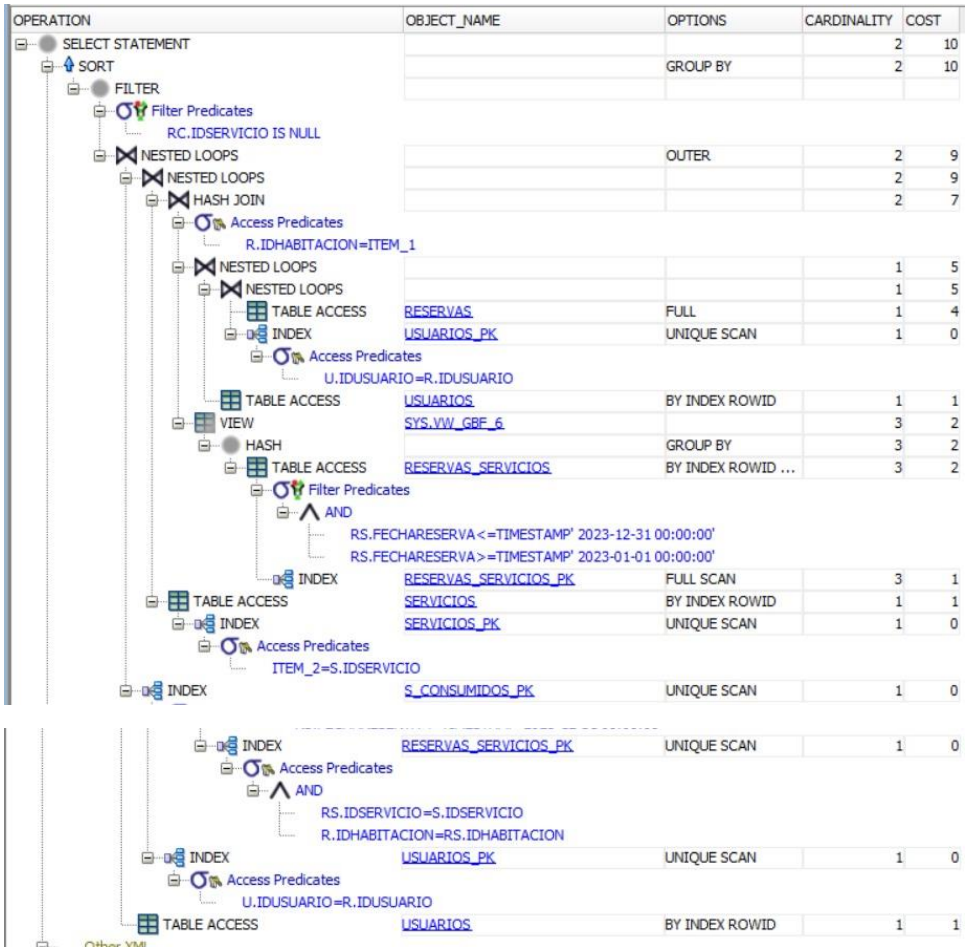

S.nombre, COUNT(rs.idservicio),
having Cont(rs.idservicio) >3
rs.fechareserva > "2023/01/01" AND
rs.fechareserva < "2023/12/31"
S.idservicio, s.nombre
s.idservicio = rs.idservicio
Reserva, Servicios
Servicios

Req 9:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	11
FILTER				
Filter Predicates				
COUNT(*)>=1				
SORT		GROUP BY	1	11
NESTED LOOPS			2	10
NESTED LOOPS			2	10
NESTED LOOPS			2	8
NESTED LOOPS			3	7
MERGE JOIN			3	4
TABLE ACCESS	RESERVAS	BY INDEX ROWID	1	2
INDEX	RESERVAS_IDX	FULL SCAN	1	1
SORT		JOIN	3	2
Access Predicates				
RC.IDCUENTA=R.IDCUENTA				
Filter Predicates				
RC.IDCUENTA=R.IDCUENTA				
INDEX	S_CONSUMIDOS_PK	FULL SCAN	3	1
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	1
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
RC.IDSERVICIO=S.IDSERVICIO				
TABLE ACCESS	RESERVAS_SERVICIOS	BY INDEX ROWID	1	1
Filter Predicates				
AND				
RS.FECHARESERVA>=TIMESTAMP' 2023-01-01 00:00:00'				
RS.FECHARESERVA<=TIMESTAMP' 2023-12-31 00:00:00'				

Req 10:

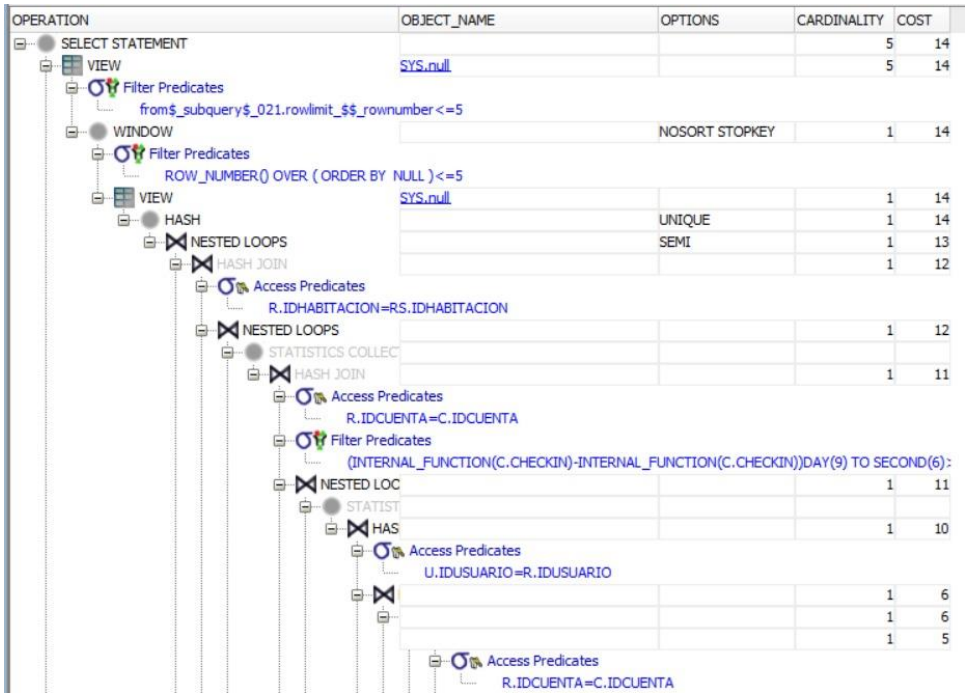


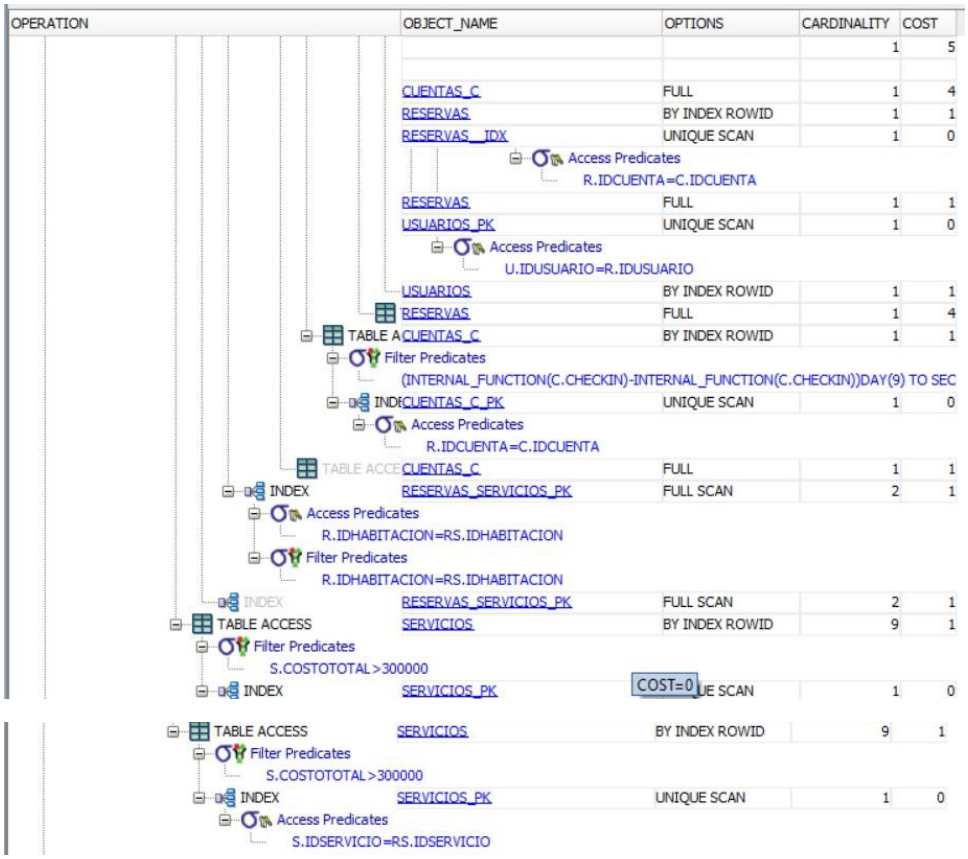
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	10
SORT		GROUP BY	2	10
FILTER				
Filter Predicates				
RC.IDSERVICIO IS NULL				
NESTED LOOPS		OUTER	2	9
NESTED LOOPS			2	9
HASH JOIN			2	7
Access Predicates				
R.IDHABITACION=ITEM_1				
NESTED LOOPS			1	5
NESTED LOOPS			1	5
TABLE ACCESS	RESERVAS	FULL	1	4
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
U.IDUSUARIO=R.IDUSUARIO				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	1
VIEW	SYS.VW_GBF_6		3	2
HASH		GROUP BY	3	2
TABLE ACCESS	RESERVAS_SERVICIOS	BY INDEX ROWID ...	3	2
Filter Predicates				
AND				
RS.FECHARESERVA<=TIMESTAMP' 2023-12-31 00:00:00'				
RS.FECHARESERVA>=TIMESTAMP' 2023-01-01 00:00:00'				
INDEX	RESERVAS_SERVICIOS_PK	FULL SCAN	3	1
TABLE ACCESS	SERVICIOS	BY INDEX ROWID	1	1
INDEX	SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
ITEM_2=S.IDSERVICIO				
INDEX	S_CONSUMIDOS_PK	UNIQUE SCAN	1	0
INDEX	RESERVAS_SERVICIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
RS.IDSERVICIO=S.IDSERVICIO				
R.IDHABITACION=RS.IDHABITACION				
INDEX	USUARIOS_PK	UNIQUE SCAN	1	0
Access Predicates				
U.IDUSUARIO=R.IDUSUARIO				
TABLE ACCESS	USUARIOS	BY INDEX ROWID	1	1

Req 11:


OPERATION
OBJECT_NAME
OPTIONS
CARDINALITY
COST
SELECT STATEMENT
HASH JOIN
Access Predicates
SC.IDSERVICIO=S.IDSERVICIO
NESTED LOOPS
NESTED LOOPS
STATISTICS COLLECTOR
HASH JOIN
Access Predicates
C.IDCUENTA=SC.IDCUENTA
NESTED LOOPS
STATISTICS COLLECTOR
HASH JOIN
Access Predicates
U.IDUSUARIO=R.IDUSUARIO
NESTED LOOPS
STATISTICS
HASH JC
Access Predicates
R.IDCUENTA=C.IDCUENTA
NES
RESERVAS
FULL
CUENTAS_C
BY INDEX ROWID
CUENTAS_C_PK
UNIQUE SCAN
Access Predicates
R.IDCUENTA=C.IDCUENTA
TABLE ACCE
USUARIOS
Filter Predicates
U.ROL='empleado'
INDEX
USUARIOS_PK
Access Predicates
U.IDUSUARIO=R.IDUSUARIO
TABLE ACCESS
S_CONSUMIDOS_PK
FULL SCAN
SERVICIOS_PK
SERVICIOS
BY INDEX ROWID
TABLE ACCESS
SERVICIOS
FULL

Req 12:


OPERATION
OBJECT_NAME
OPTIONS
CARDINALITY
COST
SELECT STATEMENT
VIEW
SYS.null
Filter Predicates
from$_subquery$_021.rowlimit_$$_rownumber<=5
WINDOW
NOSORT STOPKEY
Filter Predicates
ROW_NUMBER() OVER (ORDER BY NULL)<=5
VIEW
SYS.null
HASH
UNIQUE
NESTED LOOPS
SEMI
HASH JOIN
Access Predicates
R.IDHABITACION=RS.IDHABITACION
NESTED LOOPS
HASH JOIN
Access Predicates
R.IDCUENTA=C.IDCUENTA
Filter Predicates
NESTED LOC
HAS
Access Predicates
U.IDUSUARIO=R.IDUSUARIO
Access Predicates
R.IDCUENTA=C.IDCUENTA



Tiempos de respuesta

Número de requerimiento	Tiempo de respuesta en segundos
1	0,114
2	0,056
3	0,085
4	0,029
5	0,053
6	0,075
7	0,224
8	0,029
9	0,505
10	0,606
11	0,133
12	0.965