Informe Proyecto 2

Kevin David Hernandez Tapiero - 202111724

Análisis Modelo

- Se modifico la clase Cliente removiendo el atributo cobroTotal, pues mantenerlo implicaría que un Cliente solo pudiera estar una sola vez en el hotel o que se iría aumentando el cobro teniendo en cuenta si se quedó anteriormente en el hotel y eso no debería pasar. Así, si queremos obtener lo que gasto un cliente en una estadía o en total en el hotel, se pueden hacer cálculos sobre las tablas de consumo.
- Se modificaron las clases de los servicios para colocar costo como un atributo de todos los servicios, los servicios como por ejemplo Restaurantes, Spa, Tienda tendrán un precio de 0 pues no se puede calcular el precio de estos independientemente como por ejemplo se calcula el precio de una piscina o una sala de reuniones. Además, se le agregan los atributos horaApertura y horaCierre para tener filtros sobre esos atributos como lo señala uno de los requerimientos funcionales
- Se modifico la clase ConsumoServicioCliente cambiando idCliente a idReserva que es una llave foránea a la tabla ReservaHotel para así poder vincular los consumos a la habitación y al cliente al mismo tiempo si se requiere. Además, se añadió el atributo costo que indica cuento se pagó en ese consumo. Los cambios que se hicieron a ConsumoServicioCliente también se realizaron a la clase ClienteConsumeProducto por la misma razón, pero no se agregó el atributo costo, y se le agregaron dos atributos: descripción y fecha. También se cambio la llave primaria de estas tablas, haciendo que todos sus atributos hagan parte de la llave primaria para así permitir varios consumos en el mismo día, consumos con descripciones distintas, etc.

Diseño de la aplicación

Creación de índices

1. RFC1: Para este requerimiento sería útil usar índices debido a que la selectividad de los campos id en ReservaHotel, ReservaHotel_id en ClienteConsumeProducto y ConsumoServicioCliente son altas, y además se van a estar usando consultas que acceden a estos datos frecuentemente, así que sería útil tener índices sobre esas 3 columnas. Estos índices serán índices primarios usando arboles B+.

2. RFC2: Para este requerimiento sería útil usar índices debido a la selectividad de los campos en los que se necesita buscar, es decir en ServicioBasico.id, donde la selectividad es la mejor dado que se trata de la PK de la tabla, y en ConsumoServicioCliente.ServicioBasico_id, siendo la selectividad alta probabilísticamente al ser los datos en ese campo lo suficientemente distintos. En el caso de ServicioBasico y de ConsumoServicioCliente los índices serían primarios usando arboles B+ al tratarse en ambos casos de la PK o parte de ella.

3. RFC3: Para este requerimiento, como se están haciendo consultas comparando los id de las habitaciones en la tabla ReservaHotel, por lo cual sería útil tener un índice sobre esta columna para optimizar las búsquedas, más teniendo en cuenta que se quieren obtener los registros para cada habitación en la tabla ReservaHotel. Este índice sería un índice secundario dado que habitación_id no hace parte de la llave primaria de la tabla ReservaHotel, además, usa arboles B+ como todos los índices.

4. RFC4: Para este requerimiento es complicado saber si se necesitan índices o no, pues las condiciones de filtrado son muy amplias, por lo cual tendríamos que evaluar la selectividad en cada columna. Hablando de manera estadística, si se filtra por un intervalo de id's o si se busca un id especifico, sería útil tener un índice sobre la columna id de la tabla ServicioBasico, dado que al ser la llave primaria de la tabla tendria una selectividad alta y justificaría el uso de índices en esta columna, seria un índice primario y como siempre con arboles B+. Si se filtran por horas de apertura y de cerrado, también valdría la pena usar índices pues la selectividad es extremadamente alta, al ser los valores posibles para las columnas las horas (incluyendo minutos) de 00:00 a 23:59, siendo los valores lo suficientemente distintos y justificando así el uso de un índice sobre esas columnas en la tabla ServicioBasico que es donde están almacenados los servicios, estos índices serian secundarios porque las columnas no hacen parte de la llave primaria y usaría arboles B+. En el caso de filtrar por costo del servicio también seria útil usar un índice puesto que al menos en el contexto de la inserción de datos en el proyecto, el costo de un servicio puede estar entre 10000 y 500000, por lo cual los valores posibles son bastantes y significaría que es una columna con selectividad alta y por lo tanto, elegible para el uso de índices, este índice seria secundario dado que la columna no hace parte de la llave primaria y usaría arboles B+. Si filtráramos por nombre, también seria bueno usar índices, pues los nombres suelen ser valores únicos o rara vez se repiten, por lo cual seria una columna con selectividad alta y elegible para crear un índice, seria secundario y con árboles B+.

5. RFC5: En esta consulta al estarse haciendo búsquedas sobre la columna ReservaHotel_id de las tablas ClienteConsumeProducto y ConsumoServicioCliente, sería útil tener un índice sobre esa columna en ambas tablas, pues lo mas probable es que la Reserva no aparezca muchas veces en las tablas, y así apareciera muchas veces, la densidad de los datos haría que la columna fuera altamente selectiva, por lo que sería apta para la creación de un índice sobre ella, este índice seria primario al ser ReservaHotel_id parte de la llave primaria y usaría arboles B+. Adicional a esto, como se hace un JOIN con la tabla cliente para traer la información de esté, sería útil crear un índice sobre la columna Cliente_id en la tabla ReservaHotel, el índice seria secundario al no ser parte de la llave primaria y usaría arboles B+, bajo la misma lógica, al hacer JOIN con las tablas de consumos, sería útil tener un índice sobre id en ReservaHotel, así será mas eficiente el JOIN, este índice sería primario y usaría arboles B+

6. RFC7: En esta consulta se hace una búsqueda sobre la columna de Cliente_id en ReservaHotel, pues así se sabe cuántos días ha estado un cliente en el hotel en el último año, así que como ya tenemos un índice en esa columna por los RFC anteriores entonces sería útil crear un índice sobre la columna documento en Cliente, así al hacer JOIN entre ReservaHotel y Cliente la operación será mas eficiente, lo cual justificaría la creación del índice, seria primario y usaría arboles B+.

Diseño de las consultas

1. RFC1:
   SELECT habitacion.id, SUM(consumo_servicio_cliente.costo) as total FROM habitacion INNER JOIN reserva_hotel ON habitacion.id = reserva_hotel.habitacion_id INNER JOIN consumo_servicio_cliente ON reserva_hotel.id = consumo_servicio_cliente.Reserva_Hotel_id WHERE consumo_servicio_cliente.fecha >= SYSDATE - INTERVAL '1' YEAR GROUP BY habitacion.id
   Esta consulta da la cantidad de dinero gastada en cada habitación para los consumos de servicios

   SELECT habitacion.id, SUM(cliente_consume_producto.costo) as total FROM habitacion INNER JOIN reserva_hotel ON habitacion.id = reserva_hotel.habitacion_id INNER JOIN cliente_consume_producto ON reserva_hotel.id = cliente_consume_producto.Reserva_Hotel_id WHERE cliente_consume_producto.fecha >= SYSDATE - INTERVAL '1' YEAR GROUP BY habitacion.id
   Esta consulta da la cantidad de dinero gastada en cada habitación para los consumos de productos.

2. RFC2:
   SELECT DISTINCT sb.* FROM (SELECT servicio_basico.* FROM consumo_servicio_cliente INNER JOIN servicio_basico ON servicio_basico.id = consumo_servicio_cliente.servicio_basico_id WHERE fecha >= :fecha_menor AND fecha <= :fecha_mayor ORDER BY fecha) sb WHERE ROWNUM <=20
   Esta consulta da los servicios mas populares en un intervalo de tiempo dado por el usuario basado en la cantidad de veces que fueron consumidos

3. RFC3:
   SELECT habitacion_id, SUM(TO_DATE(FECHA_SALIDA, 'DD-MON-RR') - TO_DATE(FECHA_ENTRADA, 'DD-MON-RR')) AS dias_ocupada FROM reserva_hotel WHERE FECHA_ENTRADA >= TRUNC(SYSDATE - 365) GROUP BY habitacion_id
   Esta consulta da la cantidad de días que cada habitación fue ocupada en el ultimo año desde la fecha actual, para sacar el índice de ocupación se divide cada cantidad entre 365 y se multiplica el resultado por 100 para sacar el porcentaje.

4. RFC4:
   SELECT * FROM servicio_basico WHERE id >= :idMenor AND id <= :idMayor AND capacidad >= :capacidadPiso AND capacidad <= :capacidadTecho AND nombre LIKE '%' || :nombre || '%' AND costo >= :precioPiso AND costo <= :precioTecho AND TO_DATE(hora_apertura, 'HH24:MI') >= TO_DATE(:horaApertura, 'HH24:MI') AND TO_DATE(hora_cierre, 'HH24:MI') >= TO_DATE(:horaCierre, 'HH24:MI')
   Esta consulta da los servicios filtrados por características que da el usuario, como su nombre, su capacidad máxima y mínima, su precio máximo y mínimo, su hora de apertura y de cierre, y su id o un intervalo de id’s

5. RFC5:

SELECT SUM(Cliente_consume_producto.costo) as total FROM Cliente INNER JOIN reserva_hotel ON Cliente.documento = reserva_hotel.Cliente_id INNER JOIN Cliente_consume_producto ON reserva_hotel.id = Cliente_consume_producto.Reserva_Hotel_id WHERE Cliente.documento=:documento AND Cliente_consume_producto.fecha >= :fechaMenor AND Cliente_consume_producto.fecha <= :fechaMayor GROUP BY Cliente.documento
Esta sentencia da el consumo de productos para un usuario dado en un intervalo de fechas

SELECT SUM (consumo_servicio_Cliente.costo) as total FROM Cliente INNER JOIN reserva_hotel ON Cliente.documento = reserva_hotel.Cliente_id INNER JOIN consumo_servicio_Cliente ON reserva_hotel.id = consumo_servicio_Cliente.Reserva_Hotel_id WHERE Cliente.documento=:documento AND consumo_servicio_Cliente.fecha >= :fechaMenor AND consumo_servicio_Cliente.fecha <= :fechaMayor GROUP BY Cliente.documento
Esta sentencia da el consumo de servicios para un usuario dado en un intervalo de fechas

6. RFC7:
   SELECT * FROM (SELECT SUM(TO_DATE(FECHA_SALIDA, 'DD-MON-RR') - TO_DATE(FECHA_ENTRADA, 'DD-MON-RR')) AS diasAlojado FROM reserva_hotel WHERE FECHA_ENTRADA >= TRUNC(SYSDATE - 365) AND reserva_hotel.cliente_id=:clienteId GROUP BY reserva_hotel.cliente_id) WHERE diasAlojado >=14 ORDER BY diasAlojado
   Esta sentencia devuelve los clientes que se han alojado 2 semanas o mas en el hotel en el ultimo año

   SELECT cliente.documento, SUM(cliente_consume_producto.costo) as consumo FROM cliente INNER JOIN reserva_hotel ON reserva_hotel.cliente_id = cliente.documento INNER JOIN cliente_consume_producto ON cliente_consume_producto.reserva_hotel_id = reserva_hotel.id WHERE cliente_consume_producto.fecha >= SYSDATE - INTERVAL '1' YEAR AND cliente.documento = :documento GROUP BY cliente.documento
   SELECT cliente.documento, SUM(consumo_servicio_cliente.costo) as consumo FROM cliente INNER JOIN reserva_hotel ON reserva_hotel.cliente_id = cliente.documento INNER JOIN consumo_servicio_cliente ON consumo_servicio_cliente.reserva_hotel_id = reserva_hotel.id WHERE consumo_servicio_cliente.fecha >= SYSDATE - INTERVAL '1' YEAR AND cliente.documento = :documento GROUP BY cliente.documento
   Estas sentencias dan el consumo de servicios y productos de los clientes en el ultimo año.

Diseño y cargue masivo de datos

- Para generar e ingresar los datos a la BD se decidió crear un script en Java para cada tabla. Se prefirió esta opción sobre crear un script en SQL directamente dado que si se generaban con SQL no se podían verificar las reglas de negocio y no se podía implementar mucha lógica desde ahí, por ejemplo, si se querían insertar datos a la tabla ReservaHotel, era imposible hacer que la fecha_entrada fuera menor siempre que la fecha_salida, así que era necesario usar otros métodos para la generación de datos. La desventaja que trae hacer un script en Java es el tiempo de generación y carga que es considerablemente mas alto, supongo que debido a el tiempo que

necesita para generar los datos, establecer la conexión y después insertar el registro, y aunque esto puede parecer insignificante, en una escala de decenas de miles de datos se nota la diferencia, pues puede demorarse varios minutos en generar e insertar. Estos scripts se encuentran en la carpeta “scripts” en la carpeta donde están los repositorios, el modelo y los controladores.

- Para saber cuantos datos se debian insertar en cada tabla, se decidió que se insertarían mas datos en las tablas con las que mas se fuera a interactuar, así se vería la utilidad de los índices creados, los tiempos diferentes al usar distintas tablas, etc.