

# “Iteración 4 Sistemas Transaccionales”

Andres Felipe Molina Mahecha, Brian Manuel Rivera Hernández

Contexto de Presentación del documento Universidad de los Andes, Bogotá, Colombia

{a.molinam, b.riverah}@uniandes.edu.co

Fecha de presentación: 29 de abril del 2022

Departamento de Ingeniería de Sistemas y Computación, Universidad de los Andes

## (1 %) Análisis

Ajuste el modelo del mundo (modelo conceptual: diagrama de clases UML) propuesto en la iteración anterior, si lo requiere. Indique cuáles clases del modelo del mundo fueron actualizadas o creadas en esta iteración.

- Para esta iteración la modificación de nuestro diagrama UML fue muy sencillo, ya que solo tuvimos que agregarle la clase mantenimiento, y relacionar las clases Habitación y Servicio. Esto no implicó mayores cambios y por lo general una inserción de este estilo, en este punto del desarrollo no es para nada costosa.

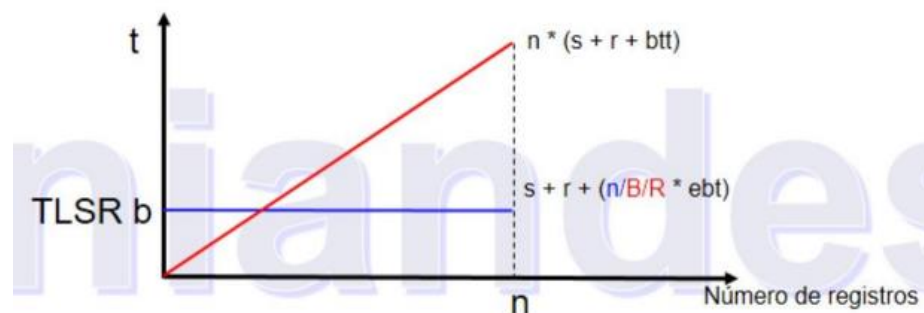
## (64 %) Diseño de la aplicación

- (1%) A partir del diseño existente, analice el impacto que representa la introducción de los nuevos requerimientos y restricciones a nivel del modelo conceptual. Realice los cambios necesarios en su modelo relacional para respetar las reglas de negocio y asegurar la calidad de este. Tenga en cuenta los comentarios recibidos en la sustentación de los talleres anteriores. Documente el diseño y las decisiones tomadas para crear los elementos de la base de datos que da el respaldo de persistencia a la aplicación, a partir del modelo conceptual. Sea claro en mencionar explícitamente los cambios relevantes entre su diseño entregado en iteraciones anteriores y el actual.
- El impacto que generarían nuevos requerimientos funcionales y restricciones depende de lo que aquellos requerimientos y restricciones solicitan. En el caso de tener que agregar nuevas clases y tuplas en clases ya existentes, se vuelve supremamente costoso en este punto del desarrollo, ya que existen muchas conexiones entre múltiples tablas, lo que cambiar una tabla podría implicar cambiar muchas más en el proceso. Al final del día, todos estos cambios se reducen a costos, por lo que podemos afirmar que sería muy costoso implementar requerimientos y restricciones, teniendo en cuenta que sean densos en su implementación. Si las restricciones o los requerimientos son muy simples, la implementación de eso lo será también, por lo que el costo de implementación será bajo. Con respecto al diseño de nuestra aplicación, el diseño empezó desde el diagrama UML y el modelo relacional, donde verificamos de manera cuidadosa de que la base de datos estuviera en forma normal FNBC, al igual que en las iteraciones pasadas. Por otro lado, a la hora de probar la inserción y creación de tablas tuvimos que ser muy cuidadosos para evitar errores, también creamos un script que borraba todas las tablas en orden, de la que menos conexiones tenía a la que más, para evitar errores a la hora del borrado. Estas sentencias nos servían para probar la base de datos a la hora de ejecutar los requerimientos funcionales de consulta, ya que se necesitaba una base de datos poblada

de la manera correcta para la verificación adecuada de los requerimientos. Teniendo en cuenta que en esta iteración solo se agregaron requerimientos funcionales de consulta, el cambio de nuestros modelos con respecto a las iteraciones pasadas no es mayor. Si comparamos nuestros modelos de la iteración 2 a la iteración 3 podemos ver un cambio considerable ya que se necesitaba la implementación de varios requerimientos funcionales, que modificaban clases y restricciones. De la iteración 3 a esta iteración, no vemos muchos cambios a nivel de clases en UML y modelo relacional, sin embargo, los cambios en el código de la aplicación si se encuentran. Tuvimos que implementar clases para las consultas SQL, donde existen parámetros para poder cumplir con dichas consultas. Por otro lado, también existen cambios en la interfaz, para que el usuario pueda generar dichas consultas y ver los resultados por medio de la interfaz gráfica.

- (63 %) Diseño físico. Analice la aplicación completa resultante de la iteración anterior y de los nuevos requerimientos para realizar el diseño físico correspondiente. En particular, diseñe los índices necesarios para el adecuado rendimiento global de la aplicación
  - (19%) Documente su diseño físico.
    - Justifique la selección de índices desde el punto de vista de cada uno de los requerimientos funcionales. Indique claramente cuál es el tipo de índice utilizado (B+, Hash, ..., primario, secundario) y tenga en cuenta el costo de almacenamiento y mantenimiento asociado a los índices.

Para la creación de índices de esta iteración debemos tener en cuenta varios factores dichos en el curso. El primero es, cuantas veces se harán dichas consultas, y teniendo en cuenta cuantas consultas se hacen al día ver si es más eficiente utilizar índices o si de lo contrario el usar índices no cambia la eficiencia de manera significativa. Una vez que se tiene en cuenta la cantidad de consultas, tenemos que ver la selectividad de las tuplas a las que se esta accediendo en dicha consulta. En caso de que la selectividad supere la intersección entre la línea roja y la línea azul de la tabla que se mostrara a continuación, se debe de utilizar un índice.



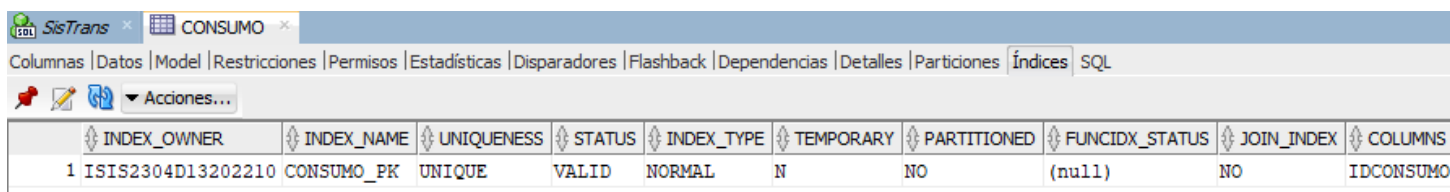
Para esta iteración, tuvimos que completar los requerimientos funcionales de consulta del RFC-7 al RFC-10. Cada uno de estos requerimientos requiere de índices distintos dependiendo de sus requerimientos y restricciones. Así que lo más sencillo es explicar cual seria nuestro diseño y específicamente en que momentos se utilizaran índices, porque, y de que tipo. Empezando por el requerimiento funcional de consulta 7. En este requerimiento nos piden lo siguiente, “*Encontrar la información de sus 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, durante el último año de operación de HotelAndes. La información en el resultado debe evidenciar el hecho de ser buen cliente.* ". Para esta consulta en específico, decidimos utilizar índices para los consumos que suman mas de 15 millones de pesos, ya que consideramos que la selectividad de la consulta es  $\leq$  a 25%, por lo que utilizar un índice es beneficioso. El tipo de índice que se puede utilizar es el B+ ya que la información no se actualizara constantemente, ya que esos consumos son cosa del pasado. Este índice se encargará de buscar a todos los clientes que hayan consumido mas de 15 millones en el último año y a partir de ahí busca el cliente deseado, esto optimiza la consulta de la manera deseada. Por otro lado, utilizaríamos un índice para todos los clientes que se hayan hospedado en el hotel por mas de 2 semanas en el año. Utilizaríamos también un índice B+ ya que cumpliría con el uso solicitado y no tiene la debilidad de la inserción de datos que se hablo anteriormente. Por lo que la búsqueda seria encontrar el índice de las personas que se han hospedado mas de 2 semanas en el hotel y posteriormente buscar el nombre del cliente. Continuando con el siguiente requerimiento funcional de consulta (RFC 8) que dice lo siguiente, *"Encontrar los servicios que hayan sido solicitados menos de 3 veces semanales, durante el último año de operación de HotelAndes."* Para este caso, pensamos que seria pertinente añadir dos índices compuestos de tipo B+, tanto para el servicio como para el tiempo de búsqueda, ya que la selectividad, bajo nuestros cálculos cumpliría con lo que requerimos. En este caso, el sistema no tendría que requerir buscar por todos los datos aquellos servicios que fueron solicitados menos de 3 veces semanales en el ultimo año, sino que simplemente se iría al índice y los encontraría. El RFC 9 dice lo siguiente, *"Se quiere conocer la información de los clientes que consumieron al menos una vez un determinado servicio del hotel, en un rango de fechas."* Para este requerimiento consideramos que un índice que puede ser muy útil para la eficiencia de la consulta es el nombre del servicio, ya que, en su gran mayoría, la selectividad de la consulta favorece la utilización del índice y por lo tanto la eficiencia de la consulta. En este caso usaríamos un índice de tipo B+ ya que no cambia la información, y se usa como un rango de fechas. Por lo que favorece al índice B+ y su utilización. Por último, el RFC 10 es la v2 del RFC donde se pide conocer la información de los clientes que NO consumieron ninguna vez un determinado servicio del hotel, en un rango de fechas. Para este requerimiento final, consideramos que la utilización de índices debe de ser la misma que el requerimiento funcional 9 ya que son casi lo mismo. Por lo que la implementación va a ser idónea y servirá para la optimización de las dos consultas solicitadas.

- Según su modelo de datos, para los índices creados de forma automática por Oracle. Incluya una foto de pantalla con la información generada por Oracle asociada a los índices existentes. Analice los índices encontrados. Específicamente, analice por qué fueron creados por Oracle y si ayudan al rendimiento de los requerimientos funcionales.

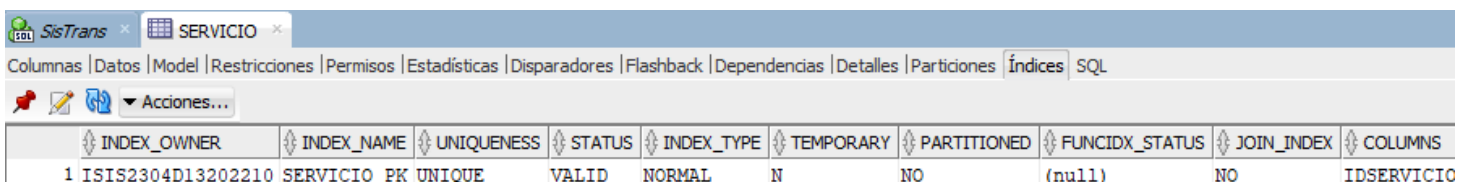
Oracle empieza a utilizar índices si la selectividad es mayor al 25%. Oracle ya tiene un sistema que intenta hacer los query de la manera mas optimizada posible, por lo que la GRAN mayoría de veces ayudan en el rendimiento de los requerimientos funcionales. A continuación, mostraremos los índices indicados por Oracle que

benefician al rendimiento de los requerimientos funcionales de consulta solicitados (RFC7 – RFC10). En este caso, encontramos que los índices que generaba Oracle para los requerimientos fueron las llaves primarias de todos los FROM de las sentencias SQL.



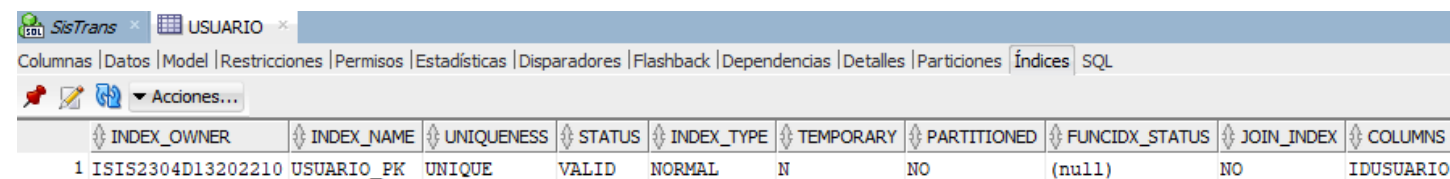
	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304D13202210	CONSUMO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDCONSUMO

Índices tabla Consumo



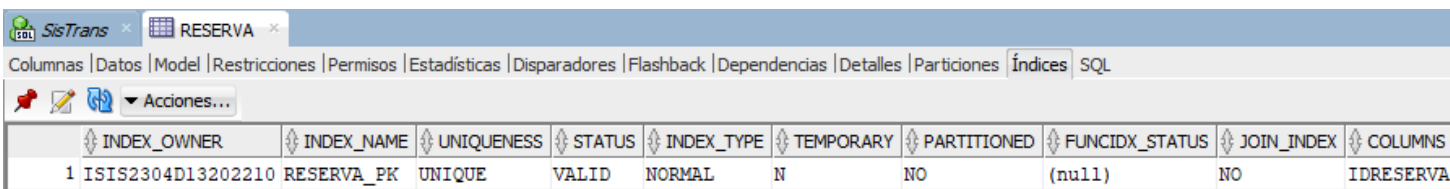
	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304D13202210	SERVICIO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDSERVICIO

Índices tabla Servicio



	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304D13202210	USUARIO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDUSUARIO

Índices tabla Usuario



	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304D13202210	RESERVA_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDRESERVA

Índices tabla Reserva

Estos índices son seleccionados por Oracle por default, ya que se trata de llaves primarias. Estos índices podrían variar dependiendo de las consultas que se hagan a la base de datos, siempre buscando optimizar al máximo las búsquedas.

- (44%) Documente plenamente el análisis realizado, incluyendo los siguientes aspectos para cada requerimiento funcional de consulta solicitado
  - Documentación del escenario de pruebas
    - Sentencias SQL que responden el requerimiento y que fueron analizadas. (.SQL)

A continuación, se muestran las sentencias .SQL que generan de manera exitosa cada uno de los requerimientos funcionales de consulta solicitados para esta iteración,

## RFC7

```
SELECT *  
FROM USUARIO  
WHERE (estadia >= 15 OR gastosHotel > 15000000)  
AND TIPOUSUARIO = 3;
```

## RFC8

```
WITH  
  anio  
AS  
  (SELECT EXTRACT(YEAR  
    FROM (SELECT fecha  
    FROM CONSUMO  
    ORDER BY consumo.fecha desc  
    FETCH FIRST 1 ROWS ONLY)) as anios FROM DUAL)  
SELECT *  
FROM SERVICIO  
WHERE SERVICIO.idservicio  
IN (SELECT Unique(servicio.idServicio)  
    from RESERVA, SERVICIO  
    where EXTRACT(YEAR FROM RESERVA.diaHora)=(select * from anio) and RESERVA.SERVICIO=SERVICIO.idServicio  
    group by trunc(RESERVA.diaHora, 'IW'), servicio.idServicio  
    having COUNT(servicio.idServicio)<3);
```

## RFC9

```
SELECT USUARIO.*  
FROM SERVICIO, USUARIO, RESERVA  
WHERE reserva.cliente=usuario.idUsuario and reserva.servicio=Servicio.idServicio;
```

## RFC10

```
SELECT *  
FROM USUARIO us  
WHERE NOT EXISTS(SELECT s.*  
    FROM SERVICIO, USUARIO s, RESERVA  
    WHERE reserva.cliente=s.idUsuario and reserva.servicio=Servicio.idServicio and us.idUsuario=s.idUsuario);
```

- Distribución de los datos con respecto a los parámetros de entrada utilizados en el requerimiento funcional. En particular se quiere un análisis de distribución que permita ver cómo puede cambiar el tamaño de la respuesta según el valor de los parámetros utilizados y la configuración de los datos de prueba.

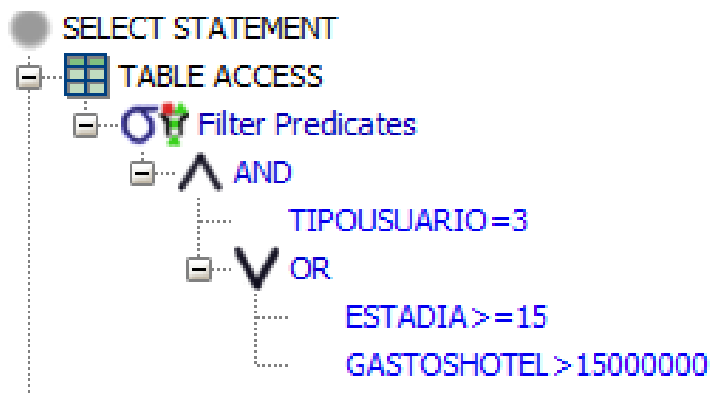
Para casos como este, nos damos cuenta de que los tiempos de búsqueda dependen de múltiples factores. Entre ellos se encuentra la complejidad del query o los datos de las tablas. Encontramos que a medida que el query se vuelve más complejo no necesariamente aumenta de sobre manera el tiempo de búsqueda. Lo que aumenta el tiempo de búsqueda es el numero de datos, sobre todo si los datos que se están solicitando mediante el query no tienen la selectividad necesaria para utilizar un índice ya que el recorrido que hay que hacer por los datos es mucho mayor a si vale la pena o no utilizar un índice, todo esto si claramente existe un volumen de datos significativo.

- Valores de los parámetros utilizados en el análisis y que constituyen diferenciadores en los planes de ejecución obtenidos.

En nuestro caso hicimos consultas sobre tablas pequeñas primero y después sobre tablas con una cantidad de datos considerable. Nos dimos cuenta de que, aunque el query puede ser el mismo el plan de ejecución cambia con el volumen de datos que exista en la base de datos. En el caso de una tabla con un numero de datos bajo almacenado, en un query en el cual se hagan joins se usaron joins de tipo Nested Loop. Pero, cuando el volumen de datos fue mucho mayor Oracle decidió cambiar el plan de ejecución y usar Merge Joins ya que este tipo de Joins se desenvuelve mejor en donde el volumen de datos es más amplio.

- Planes de consulta obtenidos en Oracle para la ejecución del requerimiento. Para ello, documente con una foto de pantalla los planes de consulta obtenidos en SQLDeveloper.

## RFC7



## RFC8

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
VIEW	<a href="#">SYS.VM_NWWW_2</a>			1
HASH		UNIQUE		1
FILTER				
Filter Predicates				
COUNT(RESERVA.SERVICIO)<3				
HASH		GROUP BY		1
NESTED LOOPS				1
NESTED LOOPS				1
TABLE ACCESS	<a href="#">RESERVA</a>	FULL		1
Filter Predicates				
AND				
RESERVA.SERVICIO IS NOT NULL				
EXTRACT(YEAR FROM INTERNAL_FUNCTION(RESERVA.DIAHORA))= (SELECT EXTRACT(YEAR FROM (SELECT from\$_subquery\$_002.FECHA FROM (SELECT FECHA FECHA,CON				
FAST DUAL				1
VIEW	<a href="#">SYS.null</a>			1
Filter Predicates				
from\$_subquery\$_002.rowlimit_\$\$_rownumber<=1				
WINDOW		SORT PUSHED RANK		1
Filter Predicates				
ROW_NUMBER() OVER ( ORDER BY INTERNAL_FUNCTION(CONSUMO.FECHA) DESC )<=1				
TABLE ACCESS	<a href="#">CONSUMO</a>	FULL		1
INDEX	<a href="#">SERVICIO_PK</a>	UNIQUE SCAN		1
SERVICIO.IDSERVICIO=RESERVA.SERVICIO				
TABLE ACCESS	<a href="#">SERVICIO</a>	BY INDEX ROWID		1

## RFC9

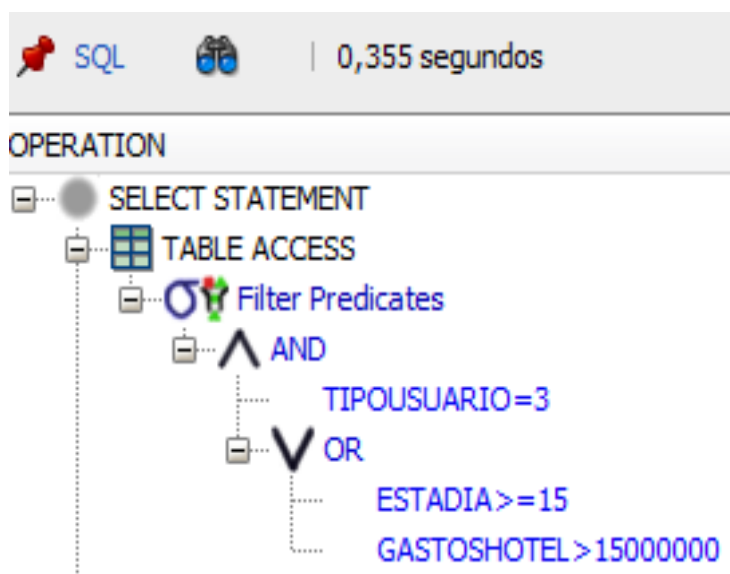
OPERATION	OBJECT_NAME
SELECT STATEMENT	
NESTED LOOPS	
NESTED LOOPS	
TABLE ACCESS	<a href="#">RESERVA</a>
Filter Predicates	
RESERVA.SERVICIO IS NOT NULL	
INDEX	<a href="#">USUARIO_PK</a>
Access Predicates	
RESERVA.CLIENTE=USUARIO.IDUSUARIO	
TABLE ACCESS	<a href="#">USUARIO</a>

## RFC 10

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
MERGE JOIN		ANTI
TABLE ACCESS	USUARIO	BY INDEX ROWID
INDEX	USUARIO_PK	FULL SCAN
SORT		UNIQUE
Access Predicates		
US.IDUSUARIO=RESERVA.CLIENTE		
Filter Predicates		
US.IDUSUARIO=RESERVA.CLIENTE		
TABLE ACCESS	RESERVA	COST=4 FULL
Filter Predicates		
AND		
RESERVA.SERVICIO IS NOT NULL		
RESERVA.CLIENTE IS NOT NULL		

- Tiempos obtenidos con la ejecución de cada uno de los planes. Estos tiempos son medidos desde el núcleo de la aplicación, es decir, no incluyen la parte de interacción con el usuario, ingreso de datos ni despliegue de resultados.

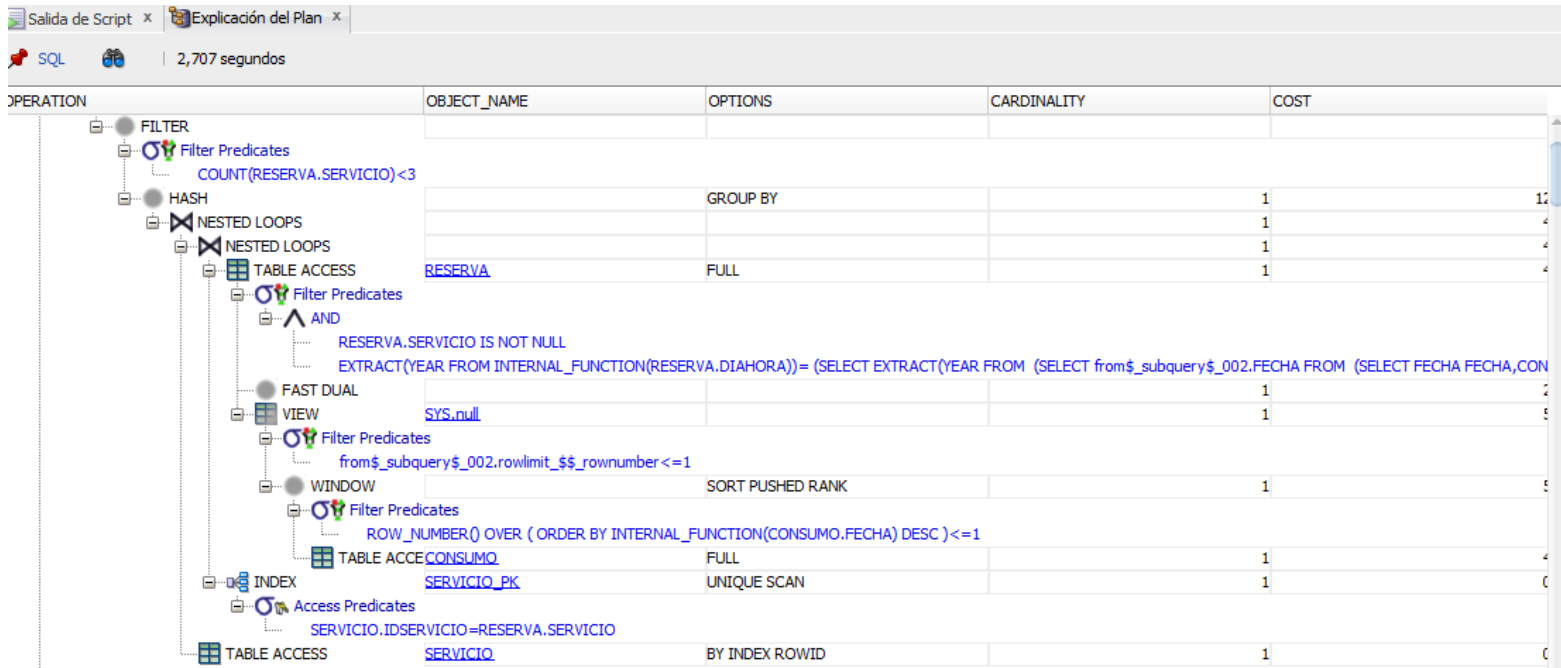
## RFC7



Como se puede apreciar, el query se demoró 0,355 segundos con el plan de ejecución de Oracle mostrado en el punto anterior



## RFC8



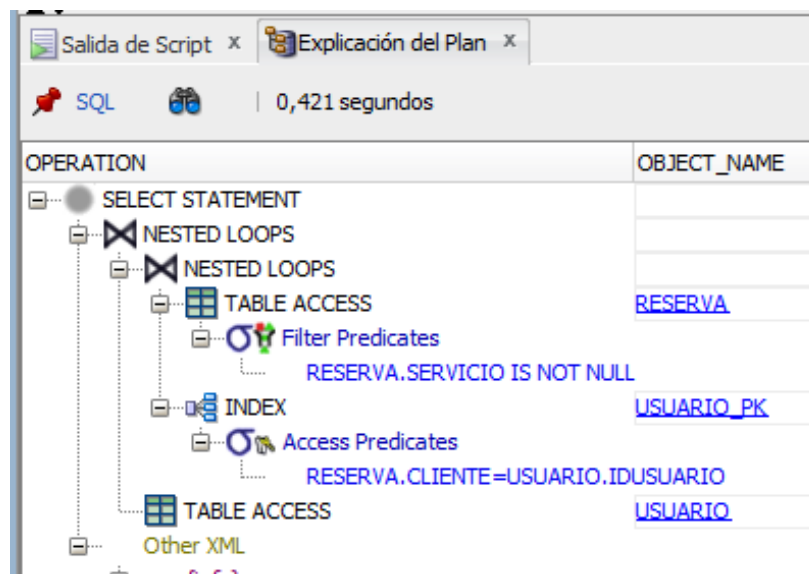
Salida de Script x Explicación del Plan x

SQL | 2,707 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
FILTER				
Filter Predicates				
COUNT(RESERVA.SERVICIO)<3				
HASH		GROUP BY	1	12
NESTED LOOPS			1	4
NESTED LOOPS			1	4
TABLE ACCESS	RESERVA	FULL	1	4
Filter Predicates				
AND				
RESERVA.SERVICIO IS NOT NULL				
EXTRACT(YEAR FROM INTERNAL_FUNCTION(RESERVA.DIAHORA))= (SELECT EXTRACT(YEAR FROM (SELECT from\$_subquery\$_002.FECHA FROM (SELECT FECHA FECHA,CON				
FAST DUAL			1	2
VIEW	SYS.null		1	5
Filter Predicates				
from\$_subquery\$_002.rowlimit_\$\$_rownumber<=1				
WINDOW		SORT PUSHED RANK	1	5
Filter Predicates				
ROW_NUMBER() OVER ( ORDER BY INTERNAL_FUNCTION(CONSUMO.FECHA) DESC )<=1				
TABLE ACCESS	CONSUMO	FULL	1	4
INDEX	SERVICIO_PK	UNIQUE SCAN	1	0
Access Predicates				
SERVICIO.IDSERVICIO=RESERVA.SERVICIO				
TABLE ACCESS	SERVICIO	BY INDEX ROWID	1	0

Como se puede apreciar, el query se demoro 2,707 segundos con el plan de ejecución de Oracle mostrado en el punto anterior

## RFC9



Salida de Script x Explicación del Plan x

SQL | 0,421 segundos

OPERATION	OBJECT_NAME
SELECT STATEMENT	
NESTED LOOPS	
NESTED LOOPS	
TABLE ACCESS	RESERVA
Filter Predicates	
RESERVA.SERVICIO IS NOT NULL	
INDEX	USUARIO_PK
Access Predicates	
RESERVA.CLIENTE=USUARIO.IDUSUARIO	
TABLE ACCESS	USUARIO
Other XML	

Como se puede apreciar, el query se demoró 0,421 segundos con el plan de

ejecución de Oracle mostrado en el punto anterior

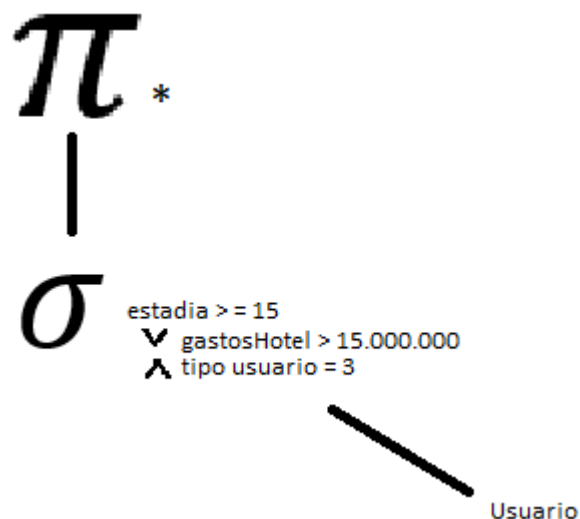
## RFC10

Salida de Script x Explicación del Plan x		
SQL   0,412 segundos		
OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
MERGE JOIN		ANTI
TABLE ACCESS	USUARIO	BY INDEX ROWID
INDEX	USUARIO_PK	FULL SCAN
SORT		UNIQUE
Access Predicates		
US.IDUSUARIO=RESERVA.CLIENTE		
Filter Predicates		
US.IDUSUARIO=RESERVA.CLIENTE		
TABLE ACCESS	RESERVA	COST=4 FULL
Filter Predicates		
AND		
RESERVA.SERVICIO IS NOT NULL		
RESERVA.CLIENTE IS NOT NULL		

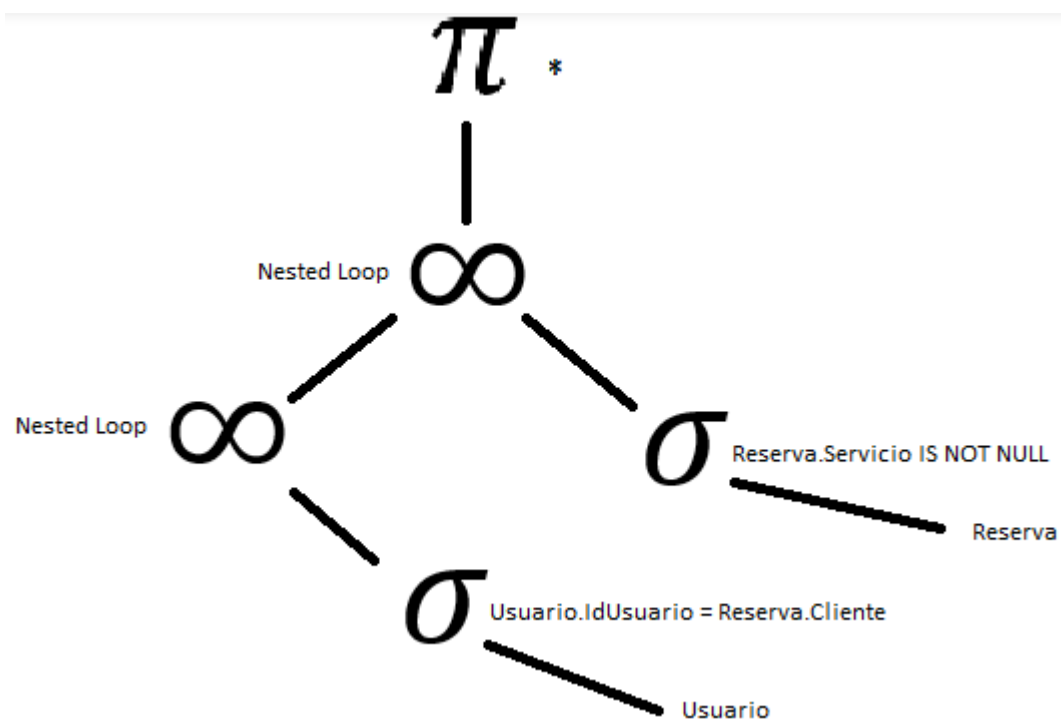
Como se puede apreciar, el query se demoró 0,412 segundos con el plan de ejecución de Oracle mostrado en el punto anterior

- Análisis de eficiencia
- Para cada requerimiento funcional, seleccione un escenario de análisis y diseñe el plan de ejecución de consulta propuesto por el grupo, de acuerdo con su conocimiento del modelo y de la aplicación.

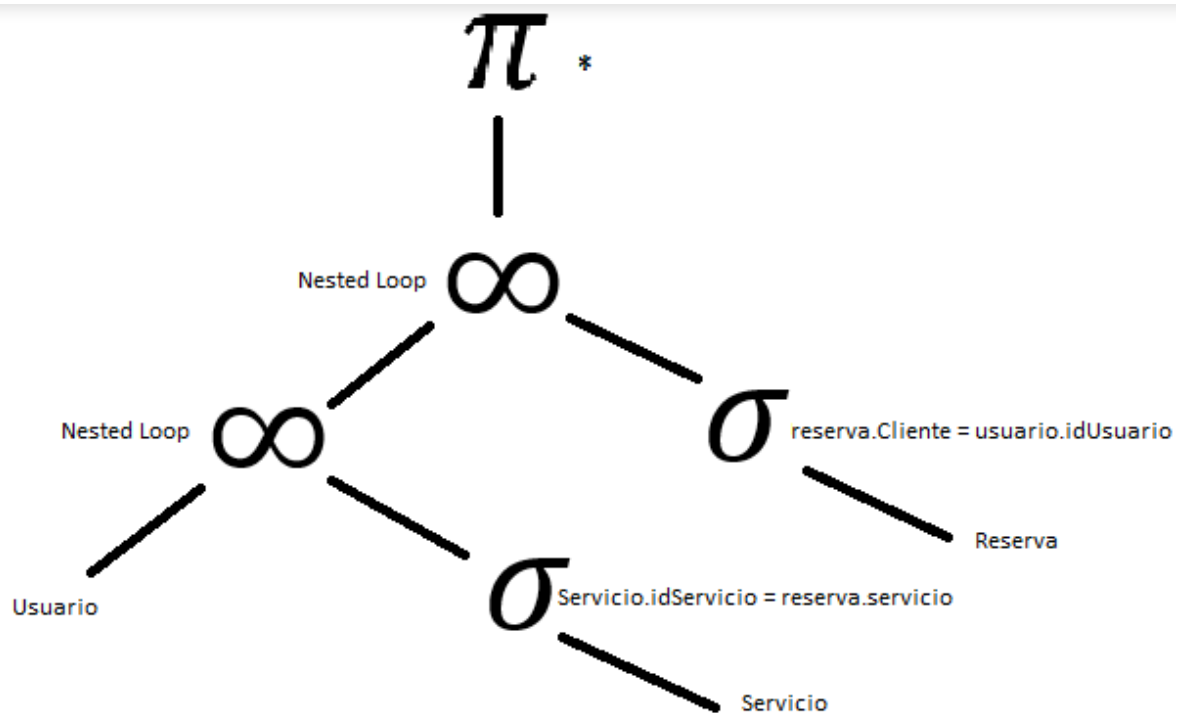
## RFC7



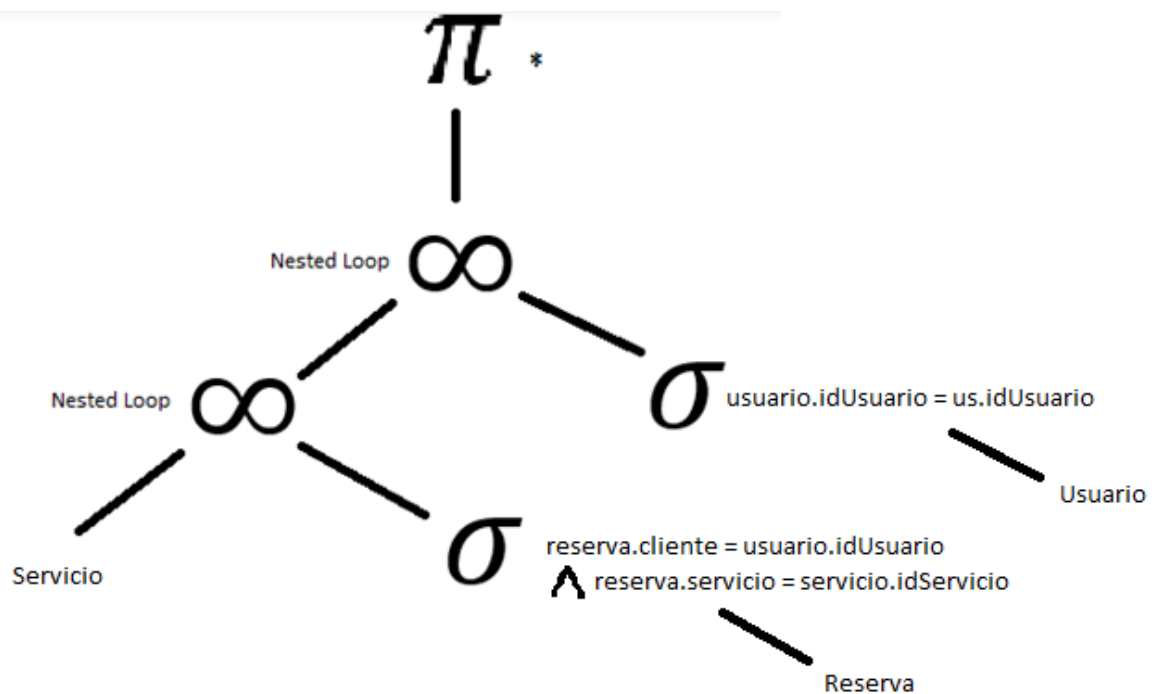
## RFC8



## RFC9



## RFC10



- Establezca escenarios de datos que le permitan validar diferentes selectividades

Existen varios escenarios que permiten validar las selectividades. En un caso podríamos hacer un query en el que se pida el ID específico de un usuario de Hotel Andes, en este caso la selectividad no es favorable para hacer la consulta ya que en caso de utilizar un índice para el ID de esa persona sería tan ineficiente como buscar Id por Id buscando el nombre, debido a la selectividad de este. Todo esto cambia si empezáramos a buscar clientes con una

nacionalidad específica, siempre y cuando la selectividad sea  $\leq$  a 25%. Si buscamos clientes con nacionalidad China, utilizar un índice no sería del todo eficiente ya que hay demasiados chinos en el planeta por lo que utilizar ese índice no sería del todo efectivo. Pero, si utilizamos un índice para los uruguayos, que son solo 3.000.000 de personas, la cosa cambia, porque cumple con los criterios de selectividad propuestos por Oracle y a la hora de buscar la gente de nacionalidad uruguaya solo habría que irnos al índice propuesto y buscar al individuo deseado.

- Compare y analice el plan de ejecución propuesto por usted y el obtenido en Oracle.

A la hora de comparar nuestros planes con los planes que nos propone Oracle podemos concluir varias cosas. En primer lugar, podemos ver que el plan de ejecución del RFC 7 tanto de Oracle como el nuestro es casi igual, esto debido a que al no ser una consulta tan compleja no existen tantas formas tanto de optimizar como de realizar el plan de ejecución. Por otra parte, en el RFC 8, 9, 10 encontramos algunas diferencias significativas. Por un lado, nuestros planes se realizaron de formas más simples y menos específicas, ya que Oracle es muy específico en su plan de ejecución. También, nos dimos cuenta de que la eficacia en los planes de Oracle es muchísimo mayor que la nuestra ya que, aunque el tipo de Join puede ser el mismo que el de nuestro plan de ejecución, el acceso a la información es diferente y nos dimos cuenta que mucho más eficiente. Finalmente, podríamos decir que a pesar de que el plan de ejecución de Oracle es más específico y un poco más efectivo no existen diferencias significativas en caso de que nuestro plan fuera el que se ejecutara en Oracle.

- El porcentaje de evaluación correspondiente a cada uno de los requerimientos solicitados se distribuye de manera uniforme sobre ellos.
- La evaluación para cada uno de los requerimientos depende de los escenarios de ejecución definidos y sus resultados.

## **(35%) Construcción de la aplicación, ejecución de pruebas y análisis de resultados**

- Ajuste las tablas creadas en Oracle de acuerdo con las decisiones del punto anterior.

Las tablas ya fueron ajustadas, se les hicieron las respectivas inserciones y pruebas para comprobar el buen funcionamiento de estas. También se hizo la carga de datos solicitada para poder verificar el buen funcionamiento de los requerimientos funcionales de consulta con una muestra de datos decente.

- Diseño del escenario de pruebas de eficiencia. Cargue de datos necesarios para hacer el estudio de eficiencia de la aplicación.
  - Diseñe los datos que le permitan verificar adecuadamente las reglas de negocio. Note que es importante generar correctamente los datos y para esta iteración lo es también el obtener un número muy grande de ellos.

El diseño de datos se generó mediante un script que generaba un archivo csv con los datos deseados, siguiendo los requerimientos de cada tabla. Para que a la hora de la consulta no existieran errores de tipado ni de otro tipo. Por lo que se puede afirmar que el

diseño de datos es correcto y permite la buena implementación y funcionamiento de los requerimientos funcionales de consulta solicitados. Generamos un volumen de datos considerable en las tablas que lo ameritaban, como la de consumos. Ya que no tiene ningún sentido generar un volumen masivo de datos en las tablas como TipoHabitacion, ya que solo existen 3 tipos de habitaciones en nuestro modelo, por lo que generar más era una repetición de datos ineficiente.

- Se debe generar un volumen de datos tal que algunas tablas no quepan en la memoria principal de la máquina.

Según las pruebas hechas, se determinó que el volumen de datos que la memoria principal de la máquina no pudo soportar fue de 900,000 por lo que, según lo que pide la iteración, es un volumen de datos considerable y que permite el buen funcionamiento y desarrollo de los requerimientos funcionales de consulta solicitados.

- Usualmente, alrededor de 1 millón de registros en su base de datos puede ser adecuado. Demasiados datos sobre este valor pueden colapsar la infraestructura, que es compartida y se considera una mala práctica. Haga pruebas que le permitan experimentar con un valor adecuado. Utilice un 20% de los datos y haga una experimentación que le permita analizar cómo se comportan cuando son muchos datos para insertar, en términos de eficiencia. Documente

En nuestro caso en específico, intentamos probar insertando 300.000 datos de manera manual. En donde el SQL Developer nos permitía importar datos desde un archivo csv, lo que esto hacía era mecanizar las sentencias INSERT en el Developer, pero, al ser 300.000 sentencias de INSERT se demoraba cerca de 1 hora en insertar la totalidad de los datos por lo que en términos de eficiencia fue supremamente malo. Si se hubiera querido cargar el volumen de datos de manera efectiva se tenía que usar el SQL LOADER, que cargaba los datos casi que de manera inmediata. Nosotros usamos SQLLoader para cargar la totalidad de nuestros 900.000 datos, y funciona a la perfección y en un tiempo muy deseable.

- El incumplimiento de este requisito tiene penalización del 20% sobre la evaluación completa de la iteración
- Puede escribir un programa de generación automática de datos acorde al diseño establecido para los mismos.
- Para la población de las tablas utilice la herramienta de carga masiva SQLLoader. Consulte el tutorial disponible en la página del curso sobre SQLLoader y utilice el usuario y la llave que le fueron entregadas al comienzo del semestre. Por medio de SQL developer también existe la posibilidad de cargar información por medio de SQLLoader.
- (5%) Documente claramente el proceso de carga de datos: Cómo fue realizado, cómo logró el volumen de datos solicitado.

En nuestro caso, usamos los tutoriales que el curso nos brindó desde el inicio del semestre, como lo fueron 1-Descarga y conexión inicial, 2-Carga inicial de archivos, Tutorial de

conexión SSH a máquinas Linux a través de connect y SQL\*Loader - Step by Step Guide How to Load a Datafile Into a Table. Lo que hicimos fue crear un script que creara los datos deseados en el formato deseado en un archivo de formato csv. Posteriormente, usamos el SQL Loader para cargar los datos del archivo csv a nuestra base de datos en SQL Developer. Para poder crear los archivos csv, creamos un script por cada tabla a la que queríamos insertar la carga masiva de datos, y después, al ejecutar el script, se genera un archivo csv por clase, que después al usar los tutoriales, se cargan los datos a nuestra base de datos. La carga de datos ocurrió de manera exitosa y se pudo realizar con el volumen de datos que nos permitió nuestra infraestructura.

- Desarrolle o ajuste las clases involucradas en los nuevos requerimientos, de forma que complete o modifique los requerimientos funcionales y cumpla con las restricciones de negocio. Realice los cambios sobre las clases que corresponden a:

- **(5%) Desarrollo y/o ajustes en la interfaz y en el control de la aplicación para cumplir con los nuevos requerimientos.**

En este caso, modificamos la interfaz para realizar las consultas solicitadas en este requerimiento. Por otro lado, tuvimos que ajustar la aplicación para que por medio de la clase SQL se accediera a la persistencia, donde se encuentra toda la lógica de la aplicación y consultara directamente en la base de datos lo que se solicitaba desde la clase SQL. Posteriormente, el resultado se mostraba en consola. La implementación de esto sería costoso si la aplicación ya estaba en el punto de desarrollo sobre el cual nosotros estábamos. Esto ocurrió en todos los requerimientos funcionales de consulta, ya que como todos los requerimientos son de consulta, siguen la misma estructura, por lo que no existen mayores cambios que los descritos anteriormente.

### **(5%) Cambios y desarrollo de las transacciones en PersistenciaHotelAndes**

Para el buen desarrollo e implementación de la iteración, tuvimos que modificar algunas clases SQL en donde nos dimos cuenta de que algunas dependencias y conexiones entre tablas estaban mal implementadas. Por otro lado, tuvimos que implementar las clases SQL sobre las cuales se desarrollan los requerimientos funcionales de consulta que nos pide la iteración. A continuación, se muestran las nuevas clases que se crearon para poder realizar las consultas de manera exitosa:

La creación de nuevos métodos para los req de consulta, además de eliminarMantenimientoServicio, eliminarMantenimientoHabitacion ,adicionarMantenimientoHabitacion, adicionarMantenimientoServicio que sirven para la solución de los RF.

### **(5%) Cambios en las clases SQL.**

En este caso, no tuvimos que cambiar ninguna clase SQL en sí. Añadimos unas clases y cargamos los datos previamente mencionados en el documento. Mas no tuvimos que hacer ningún cambio mayor. En este caso en específico y debido a los requerimientos y restricciones que se pedían, no nos vimos obligados a cambiar las clases, aunque, puede que, si los cambios fueran otros, así como las restricciones, podrían llegar a ser muy costosos los cambios ya que puede que cada cambio que se haga dentro de las clases SQL afecte el modelo relacional y por ende, afecte otras clases, generando un efecto en cadena.

- Tenga en cuenta que los cambios en su modelo impactan directamente JDO, y pueden llegar a causar errores en requerimientos previos.
- Todos los requerimientos tienen el mismo valor.
- **(15 %) Análisis del proceso de optimización y el modelo de ejecución de consultas.**
  - Analice la diferencia entre la ejecución de consultas delegada al manejador de bases de datos como Oracle y compárelo con una ejecución donde la aplicación trae los datos a memoria principal y resuelve con instrucciones de control (if, while, etc.), los operadores involucrados en las consultas como joins, selecciones y proyecciones.
    - La diferencia radica en la cantidad de accesos a la memoria secundaria, ya que cuando se hace un (if, while, etc.), las operaciones son con datos que están en memoria principal y si esta no cabe, procede a realizar algoritmos que no están preparados para funcionar con grandes volúmenes de información, en cambio los join selecciones y proyecciones realizan un manejo de la información a álgebra relacional y posteriormente decide cuál es la mejor solución. Algo que no ocurre con un if o un while que sus operaciones se hacen greedy y no son optimizadas.