Iteración 3 – Sistemas Transaccionales

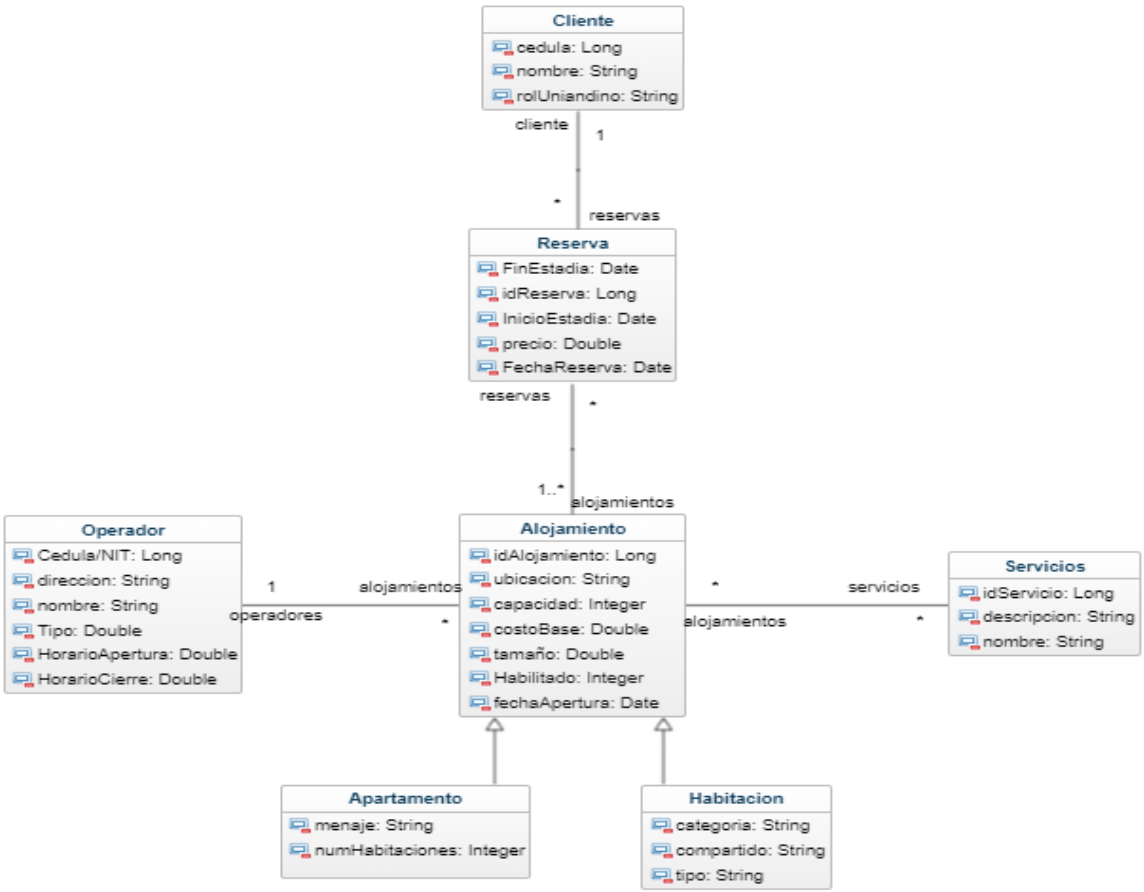
Andrés Javier Ortiz Peña, Carlos Alberto Guerrero Medina
Universidad de los Andes, Bogotá, Colombia
{aj.ortiz10, ca.guerrero} @uniandes.edu.co

Fecha de Presentación: Abril 24 de 2018

Contenido

Análisis

(1 %) 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.



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 del mismo. 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 cambio más importante fue suprimir la clase Oferta establecer una relación de 1 a muchos entre Operador y Alojamiento. Las otras clases siguen funcionando igual que antes con excepción de Alojamiento donde se añadió un atributo "habilitado" para conocer la disponibilidad de los alojamientos.

Diseño Físico

- **(63%) 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.**
    - A) **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.**

RFC10:

Tabla	Índice Secundario
RESERVA	FECHARESERVA

- INDICES ELEGIDOS:

  - Columna FECHARESERVA (secundario), se escoge un índice del tipo B+ ya que se está usando para establecer un rango entre dos fechas. El árbol B+ es idóneo en este caso porque empieza a seleccionar entre las hojas desde la fecha inicial hasta la fecha final del rango (BETWEEN).

RFC11:

Tabla	Índice Secundario
RESERVA	FECHARESERVA

- INDICES ELEGIDOS:

  - Columna FECHARESERVA (secundario), al igual que en el RFC10, aquí tenemos que filtrar por un rango de fechas, luego usamos el mismo índice para FECHARESERVA en la tabla RESERVA

RFC12:

Tabla	Índice Secundario
RESERVA	INICIOESTADIA
RESERVA	FINESTADIA

- INDICES ELEGIDOS:

  - Columnas INICIOESTADIA y FINESTADIA, se escoge in índice tipo B+ para poder acceder rapidamente a estas dos fechas que son necesarias para calcular en que fechas son más o menos ocupados los alojamientos por clientes, al igual que para saber en que fechas son más o menos solicitados los alojamientos de los operadores.

RFC13:

Tabla	Índice Secundario
RESERVA	FECHARESERVA
ALOJAMIENTO	PRECIO

- INDICES ELEGIDOS:

- Columnas FECHARESERVA, PRECIO Y TIPO, se escoge in índice tipo B+ para poder acceder rapidamente a los tres casos de clientes buenos. La FECHARESERVA sedebe acceder rapidamente para poder saber si mensualmente el cliente esta realizando reservas. PRECIO se debe acceder rapidamente para poder saber que clientes siempre pagan más de 150USD (o 450.000 COP), para esto se obtienen los que pagan menos de 150USD y se le quitan al total de clientes. Para el atributo TIPO no se coloca indice ya que para saber cuales clientes siempre utilizan habitaciones tipo "suite" se deben escoger todos los que no y quitarlos del total lo cual representa la gran mayoría de las tuplas.

B) **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.**

En las relaciones que utilizamos para los RFCs se encuentran los siguientes índices:

TABLA CLIENTE:

	INDEX_OWN...	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_S...	JOIN_INDEX	COLUMNS
1	ISIS230...	SYS_C00...	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	CEDULA

TABLA RESERVAS:

	INDEX_O...	INDEX_NAME	UNIQUEN...	STATUS	INDEX_TYPE	T...	PARTITIO...	FUNCIDX...	JOIN_IN...	COLUMNS
1	ISIS230...	RES_FECHA_IDX	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	FECHARE..
2	ISIS230...	SYS_C00338130	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDRESERVA

TABLA RESERVASDEALOJAMIENTO:

	INDEX_OWNER	INDEX_NAME	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN
1	ISIS2304A911810	SYS_C00350756	ISIS2304A911810	RESERVASDEALOJAMIENTO	IDALOJAMIENTO	1	
2	ISIS2304A911810	SYS_C00350756	ISIS2304A911810	RESERVASDEALOJAMIENTO	IDRESERVA	2	

TABLA ALOJAMIENTO:

	INDEX_OWNER	INDEX_NAME	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	C
1	ISIS2304A911810	SYS_C00287587	ISIS2304A911810	ALOJAMIENTO	IDALOJAMIENTO	1	22	

TABLA OPERADORES:

	INDEX_OWNER	INDEX_NAME	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	C
1	ISIS2304A911810	SYS_C00288390	ISIS2304A911810	OPERADORES	CEDULA_NIT	1	22	

TABLA HABITACION:

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	ISIS2304A911810	SYS_C00350568	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDHABITACION

- **Analice los índices encontrados. Específicamente, analice por qué fueron creados por Oracle y si ayudan al rendimiento de los requerimientos funcionales.**

Los índices cuyos nombres comienzan con "SYS_ " son aquellos que creó Oracle. Estos índices se crean en el (los) atributo(s) que forma(n) parte de la Primary Key (PK) de la relación ya que esta PK tiene valores siempre distintos que nunca son nulos y la selectividad es la más baja, entonces es casi obligatorio tener un índice para mejorar el rendimiento de los requerimientos funcionales.

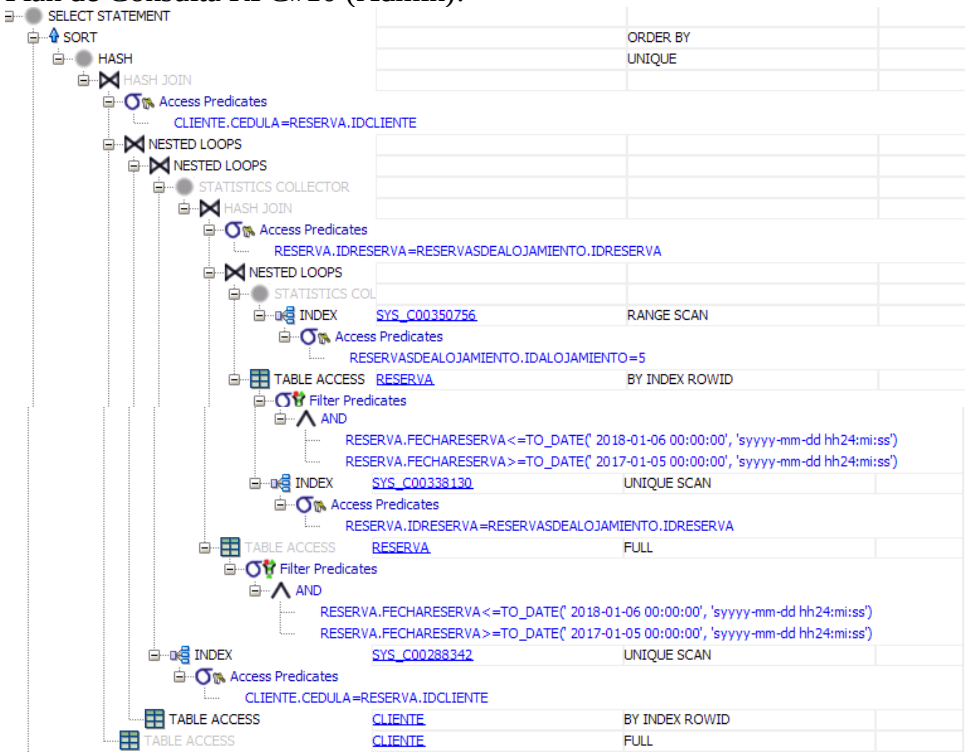
(44%) Documente plenamente el análisis realizado, incluyendo los siguientes aspectos para cada requerimiento funcional de consulta solicitado

*** Documentación del escenario de pruebas**

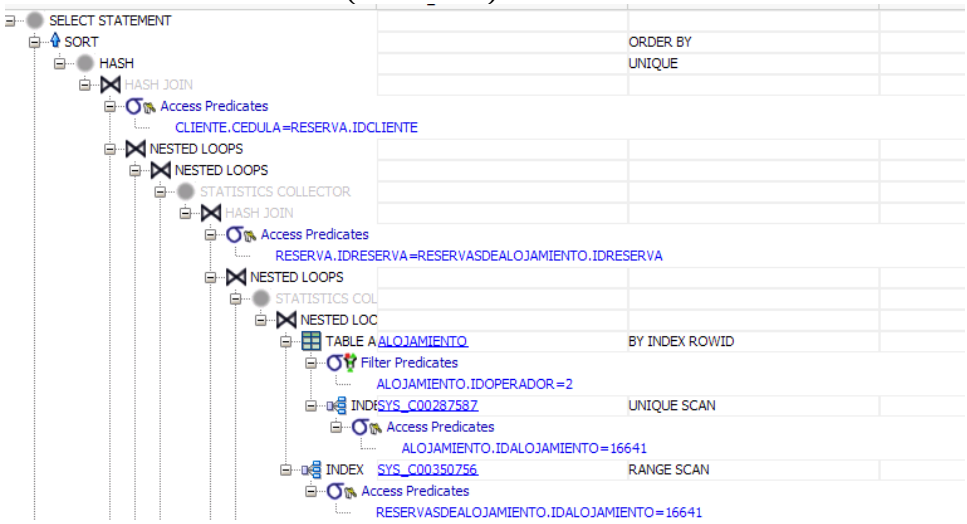
- **Sentencias SQL que responden el requerimiento y que fueron analizadas.**
- **Distribución de los datos con respecto a los parámetros de entrada utilizados en el requerimiento funcional. En particular se quiere un análisis de distribución que permita ver cómo puede cambiar el tamaño de la respuesta según el valor de los parámetros utilizados y la configuración de los datos de prueba.**
- **Valores de los parámetros utilizados en el análisis y que constituyen diferenciadores en los planes de ejecución obtenidos.**

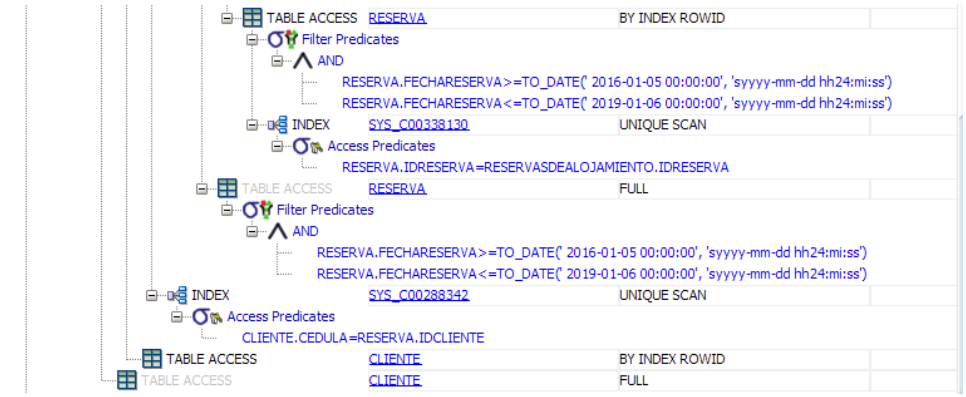
- **Planes de consulta obtenidos en Oracle para la ejecución del requerimiento. Para ello, documente con una foto de pantalla los planes de consulta obtenidos en SQLDevelopper.**

Plan de Consulta RFC#10 (Admin):

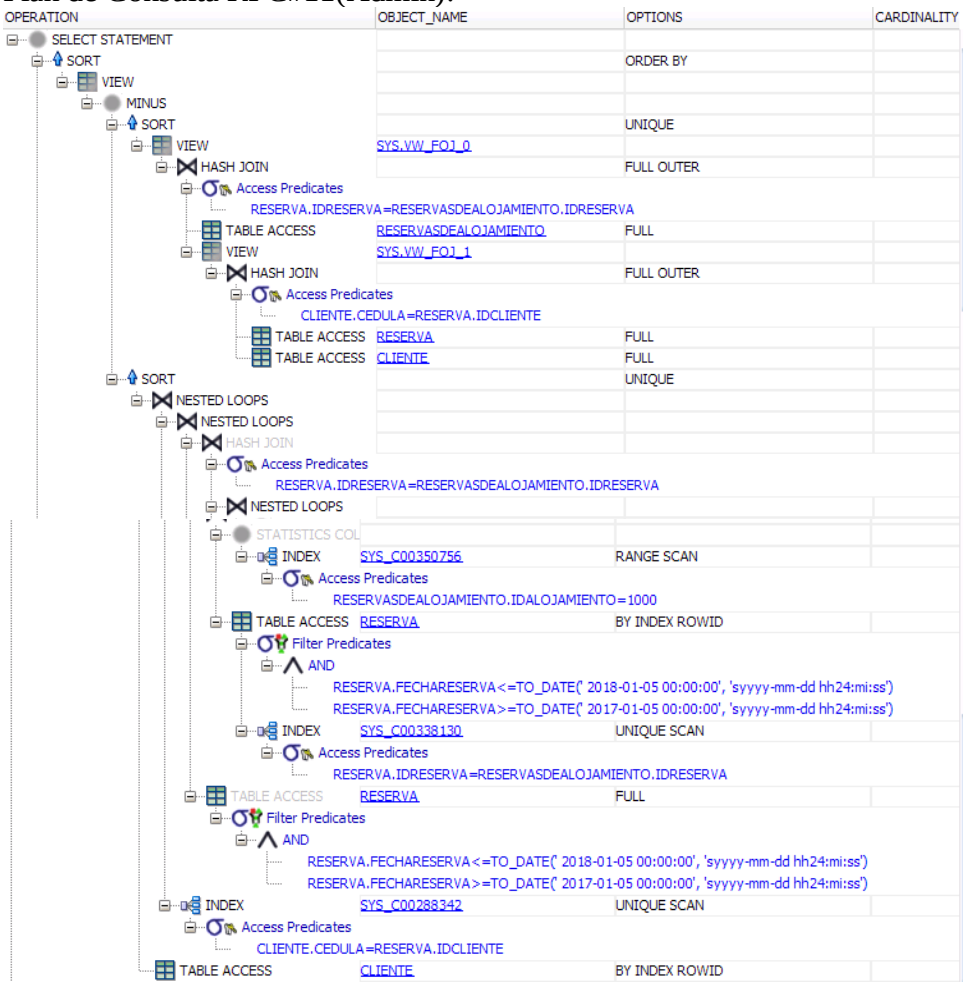


Plan de Consulta RFC#10 (Proveedor):

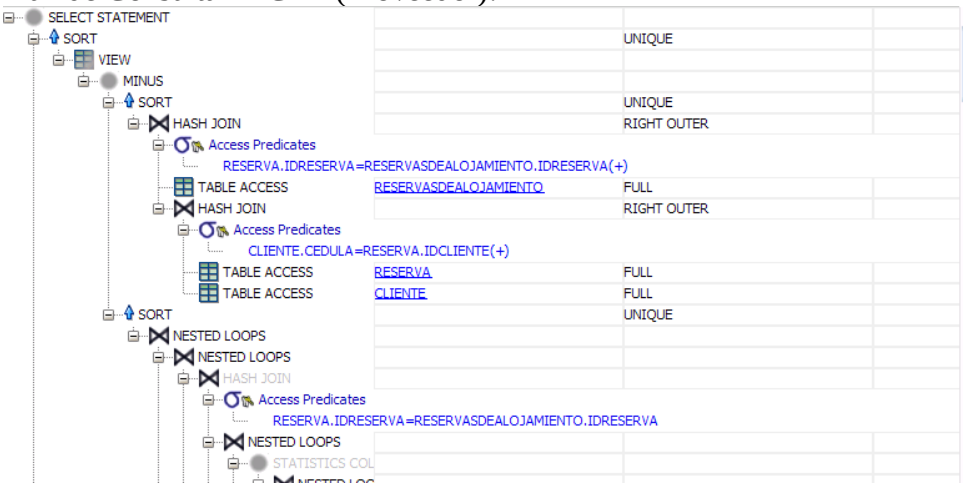


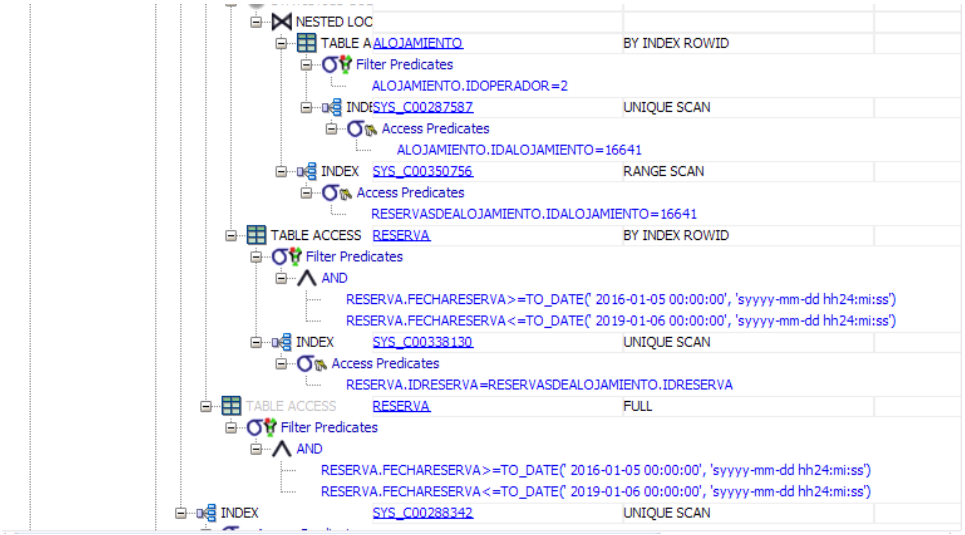


Plan de Consulta RFC#11(Admin):



Plan de Consulta RFC#11(Proveedor):





Plan de Consulta RFC#12:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			134	50590
SORT		UNIQUE	134	50590
UNION-ALL				
HASH JOIN		RIGHT SEMI	32	14551
Access Predicates				
AND				
A.OCUPACION=B.MAXIMO				
A.SEMANA=B.SEMANA				
VIEW			318252	6608
HASH		GROUP BY	318252	6608
VIEW			318252	6600
HASH		GROUP BY	318252	6600
VIEW			318252	4248
HASH		GROUP BY	318252	4248
MERGE JOIN			350482	380
SORT		JOIN	12017	373
HAS			12017	372
Access Predicates				
ALOJAMIENTO.IDALOJAMIENTO=RESERVASDEALOJAMIENTO.IDALOJAMIENTO				
			12017	133
Access Predicates				
RESERVASDEALOJAMIENTO.IDRESERVA=RESERVA.IDRESERVA				
RESERVA		FULL	12017	102
Filter Predicates				
AND				
RESERVA.FINESTADIA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVA.INICIOESTADIA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVASDEALOJAMIENTO		FULL	50000	30
ALOJAMIENTO		FULL	105000	239
FILTER				
Filter Predicates				
FECHA<=RESERVA.FINESTADIA				
SORT		JOIN	32	6
Access Predicates				
FECHA>=RESERVA.INICIOESTADIA				
Filter Predicates				
FECHA>=RESERVA.INICIOESTADIA				
LISTA_DE_FECHAS		FULL	32	5
Filter Predicates				
AND				
FECHA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
FECHA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
VIEW			318252	6600
HASH		GROUP BY	318252	6600
VIEW			318252	4248
HASH		GROUP BY	318252	4248
MERGE JOIN			350482	380
SORT		JOIN	12017	373
HASH JOIN			12017	372
Access Predicates				
ALOJAMIENTO.IDALOJAMIENTO=RESERVASDEALOJAMIENTO.IDALOJAMIENTO				
HASH JC			12017	133
Access Predicates				
RESERVASDEALOJAMIENTO.IDRESERVA=RESERVA.IDRESERVA				

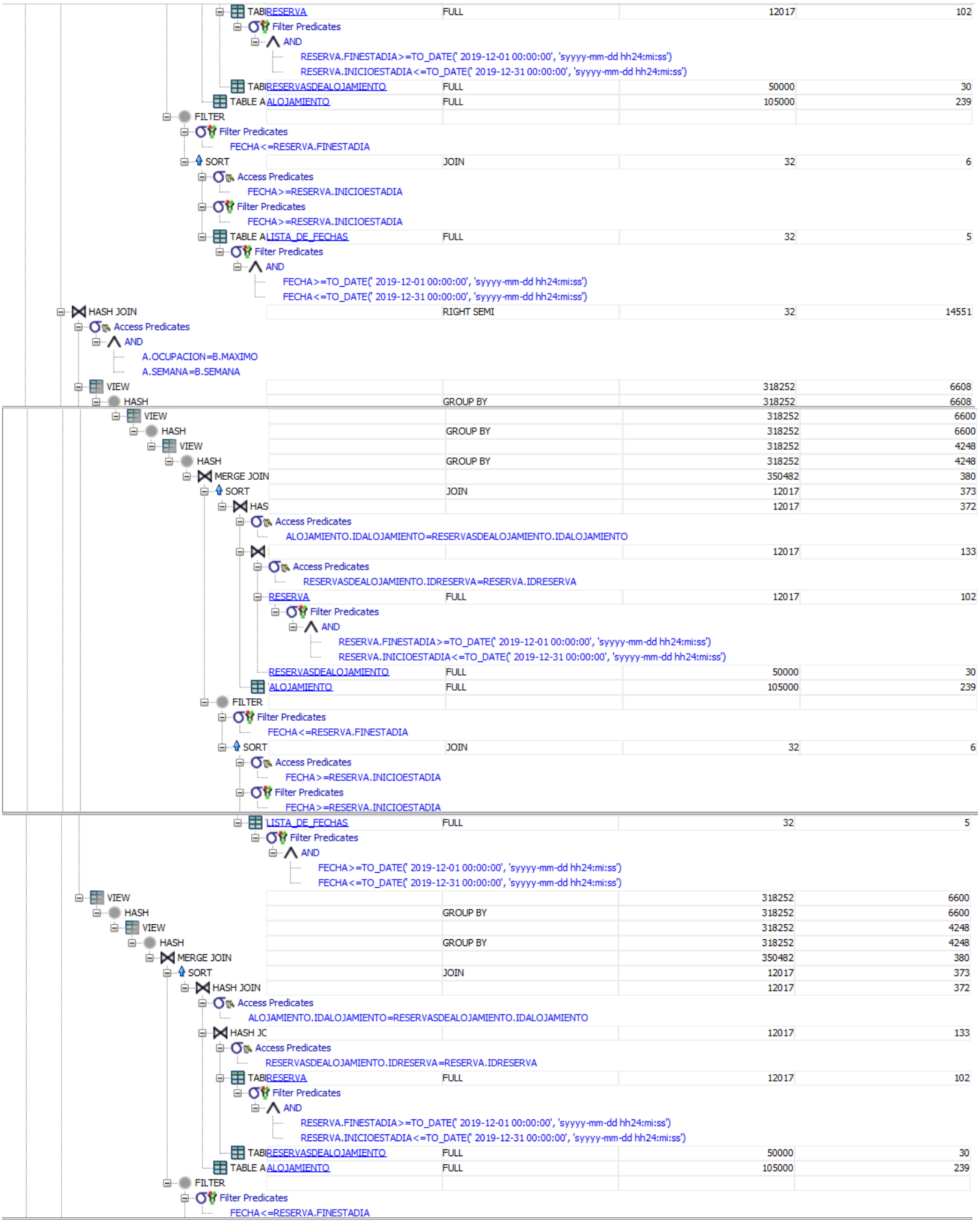
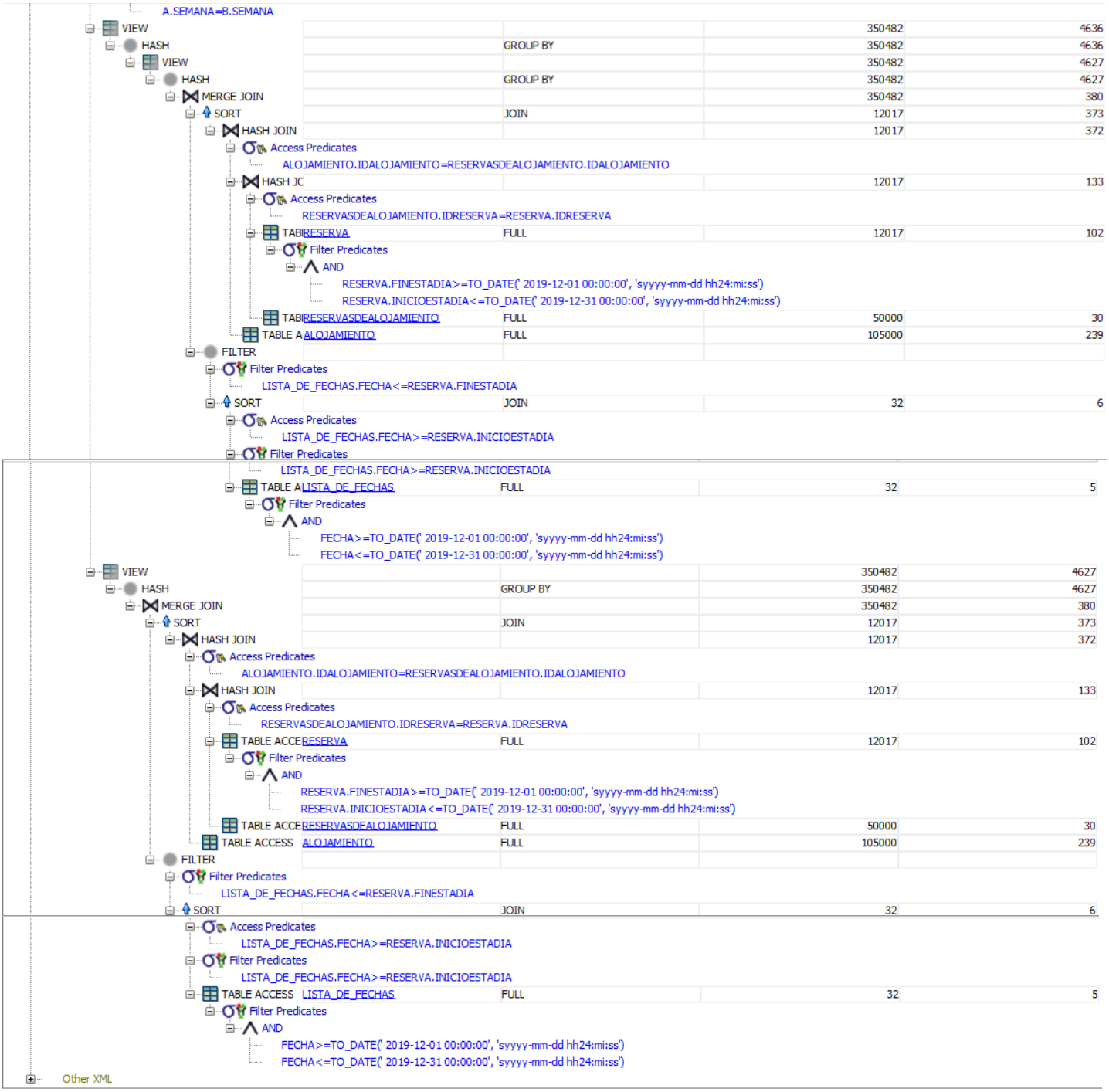


TABLE ACCESS RESERVA FULL 12017 102
Filter Predicates
AND
RESERVA.FINESTADIA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
RESERVA.INICIOESTADIA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
TABLE ACCESS RESERVASDEALOJAMIENTO FULL 50000 30
TABLE ACCESS ALOJAMIENTO FULL 105000 239
FILTER
Filter Predicates
FECHA<=RESERVA.FINESTADIA
SORT JOIN 32 6
Access Predicates
FECHA>=RESERVA.INICIOESTADIA
Filter Predicates
FECHA>=RESERVA.INICIOESTADIA
TABLE ACCESS LISTA_DE_FECHAS FULL 32 5
Filter Predicates
AND
FECHA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
FECHA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
HASH JOIN RIGHT SEMI 32 14551
Access Predicates
AND
A.OCUPACION=B.MAXIMO
A.SEMANA=B.SEMANA
VIEW 318252 6608
HASH GROUP BY 318252 6608
VIEW 318252 6600
HASH GROUP BY 318252 6600
VIEW 318252 4248
HASH GROUP BY 318252 4248
MERGE JOIN 350482 380
SORT JOIN 12017 373
HAS 12017 372
Access Predicates
ALOJAMIENTO.IDALOJAMIENTO=RESERVASDEALOJAMIENTO.IDALOJAMIENTO
12017 133
Access Predicates
RESERVASDEALOJAMIENTO.IDRESERVA=RESERVA.IDRESERVA
RESERVA FULL 12017 102
Filter Predicates
AND
RESERVA.FINESTADIA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
RESERVA.INICIOESTADIA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
RESERVASDEALOJAMIENTO FULL 50000 30
ALOJAMIENTO FULL 105000 239
FILTER
Filter Predicates
FECHA<=RESERVA.FINESTADIA
SORT JOIN 32 6
Access Predicates
FECHA>=RESERVA.INICIOESTADIA
Filter Predicates
FECHA>=RESERVA.INICIOESTADIA
LISTA_DE_FECHAS FULL 32 5
Filter Predicates
AND
FECHA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
FECHA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
VIEW 318252 6600
HASH GROUP BY 318252 6600
VIEW 318252 4248
HASH GROUP BY 318252 4248
MERGE JOIN 350482 380
SORT JOIN 12017 373
HASH JOIN 12017 372
Access Predicates
ALOJAMIENTO.IDALOJAMIENTO=RESERVASDEALOJAMIENTO.IDALOJAMIENTO
HASH JC 12017 133
Access Predicates
RESERVASDEALOJAMIENTO.IDRESERVA=RESERVA.IDRESERVA
TABLE ACCESS RESERVA FULL 12017 102
Filter Predicates
AND
RESERVA.FINESTADIA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
RESERVA.INICIOESTADIA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
TABLE ACCESS RESERVASDEALOJAMIENTO FULL 50000 30
TABLE ACCESS ALOJAMIENTO FULL 105000 239
FILTER
Filter Predicates
FECHA<=RESERVA.FINESTADIA


SORT JOIN 32 6
Access Predicates
FECHA>=RESERVA.INICIOESTADIA
Filter Predicates
FECHA>=RESERVA.INICIOESTADIA
TABLE ALISTA_DE_FECHAS FULL 32 5
Filter Predicates
AND
FECHA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
FECHA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
HASH JOIN RIGHT SEMI 35 10742
Access Predicates
AND
A.SOLICITUDES=B.MAXIMO
A.SEMANA=B.SEMANA
VIEW 350482 4636
HASH GROUP BY 350482 4636
VIEW 350482 4627
HASH GROUP BY 350482 4627
MERGE JOIN 350482 380
SORT JOIN 12017 373
HASH JOIN 12017 372
Access Predicates
ALOJAMIENTO.IDALOJAMIENTO=RESERVASDEALOJAMIENTO.IDALOJAMIENTO
HASH JC 12017 133
Access Predicates
RESERVASDEALOJAMIENTO.IDRESERVA=RESERVA.IDRESERVA
TABLRESERVA FULL 12017 102
Filter Predicates
AND
RESERVA.FINESTADIA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
RESERVA.INICIOESTADIA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
TABLRESERVASDEALOJAMIENTO FULL 50000 30
TABLE ALOJAMIENTO FULL 105000 239
FILTER
Filter Predicates
LISTA_DE_FECHAS.FECHA<=RESERVA.FINESTADIA
SORT JOIN 32 6
Access Predicates
LISTA_DE_FECHAS.FECHA>=RESERVA.INICIOESTADIA
Filter Predicates
LISTA_DE_FECHAS.FECHA>=RESERVA.INICIOESTADIA
TABLE ALISTA_DE_FECHAS FULL 32 5
Filter Predicates
AND
FECHA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
FECHA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
VIEW 350482 4627
HASH GROUP BY 350482 4627
MERGE JOIN 350482 380
SORT JOIN 12017 373
HASH JOIN 12017 372
Access Predicates
ALOJAMIENTO.IDALOJAMIENTO=RESERVASDEALOJAMIENTO.IDALOJAMIENTO
HASH JOIN 12017 133
Access Predicates
RESERVASDEALOJAMIENTO.IDRESERVA=RESERVA.IDRESERVA
TABLE ACCERESERVA FULL 12017 102
Filter Predicates
AND
RESERVA.FINESTADIA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
RESERVA.INICIOESTADIA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
TABLE ACCERESERVASDEALOJAMIENTO FULL 50000 30
TABLE ACCESS ALOJAMIENTO FULL 105000 239
FILTER
Filter Predicates
LISTA_DE_FECHAS.FECHA<=RESERVA.FINESTADIA
SORT JOIN 32 6
Access Predicates
LISTA_DE_FECHAS.FECHA>=RESERVA.INICIOESTADIA
Filter Predicates
LISTA_DE_FECHAS.FECHA>=RESERVA.INICIOESTADIA
TABLE ACCESS LISTA_DE_FECHAS FULL 32 5
Filter Predicates
AND
FECHA>=TO_DATE(' 2019-12-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
FECHA<=TO_DATE(' 2019-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
HASH JOIN RIGHT SEMI 35 10742
Access Predicates
AND
A.SOLICITUDES=B.MAXIMO



- **<u>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.</u>**

RFC#10(Admin)

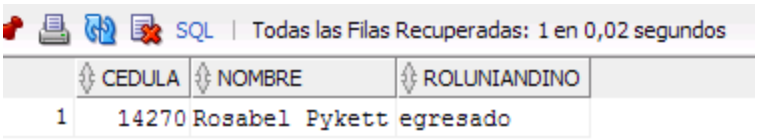
Tiempo ejecución:

SQL | Todas las Filas Recuperadas: 2 en 0,078 segundos

	CEDULA	NOMBRE	ROLUNIANDINO
1	8034	Cchaddie Karpf	egresado
2	9526	Becki Beeston	egresado

RFC#10(Proveedor)

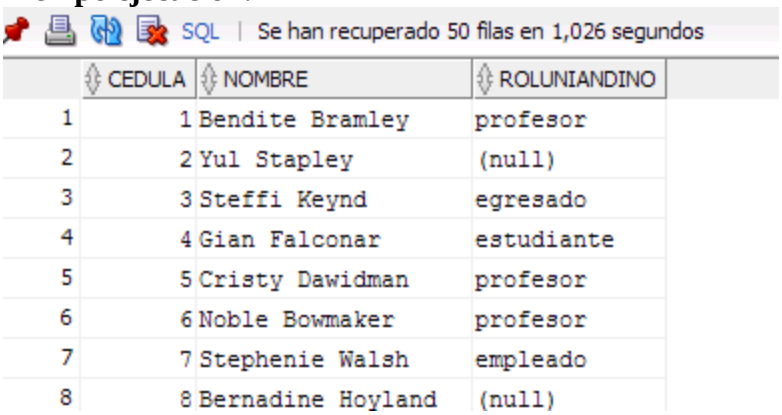
Tiempo ejecución:


SQL | Todas las Filas Recuperadas: 1 en 0,02 segundos

	CEDULA	NOMBRE	ROLUNIANDINO
1	14270	Rosabel Pykett	egresado

RFC#11(Admin)

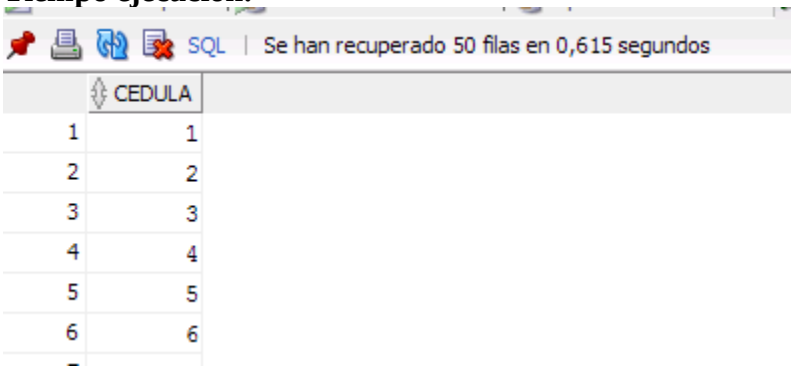
Tiempo ejecución:


SQL | Se han recuperado 50 filas en 1,026 segundos

	CEDULA	NOMBRE	ROLUNIANDINO
1	1	Bendite Bramley	profesor
2	2	Yul Stapley	(null)
3	3	Steffi Keynd	egresado
4	4	Gian Falconar	estudiante
5	5	Cristy Dawidman	profesor
6	6	Noble Bowmaker	profesor
7	7	Stephenie Walsh	empleado
8	8	Bernadine Hoyland	(null)

RFC#11(Proveedor)

Tiempo ejecución:


SQL | Se han recuperado 50 filas en 0,615 segundos

	CEDULA
1	1
2	2
3	3
4	4
5	5
6	6

*** Análisis de eficiencia**

- **Establezca escenarios de datos que le permitan validar diferentes selectividades.**
- **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.**
- **Compare y analice el plan de ejecución propuesto por usted y el obtenido en Oracle.**
  - ✓ **El porcentaje de evaluación correspondiente a cada uno de los requerimientos solicitados es proporcional al número de los requerimientos.**
  - ✓ **La nota para cada uno de los requerimientos depende de los escenarios de ejecución definidos.**

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

(35%)

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

- **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 adecuadamente los datos y para esta iteración lo es también el obtener**

un número muy grande de ellos. Se debe generar un volumen de datos tal que algunas tablas no quepan en la memoria principal de la máquina. El no cumplimiento de este requisito implica la invalidez de este componente de la evaluación.

- **Puede escribir un programa de generación automática de datos acorde al diseño establecido para los mismos.**
- **(5%) Documente claramente el proceso de carga de datos: Cómo fue realizado, cómo logró el volumen de datos solicitado, …**

Se escribió un pequeño programa en Java que utiliza un **api** generada en mockaroo (la cual proporciona datos aleatorios con las características definidas por el usuario) para crear archivos con extensión .csv que luego serán importados en las respectivas relaciones en la base de datos.
El programa necesita que se le proporcione el link del api donde están los datos aleatorios, luego los toma y crea en la carpeta que se le asigne un archivo con la extensión que se le indique (csv, en este caso). Sin embargo, se debieron modificar manualmente los datos para que las PK'S fueran distintas y no hubieran problemas al insertar los datos en las tablas.

- **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 a los servicios REST para cumplir con los nuevos requerimientos.**
  - **(5%) Cambios y desarrollo de las transacciones en AlohAndesMaster**
  - **(5%) Cambios en los DAO.**
- **(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.**
  - **Documente el análisis realizado, de forma clara y concisa.**

La solución con operadores de control es menos eficaz ya que la prueba en postman se demora mucho más (30 segundos). La solución delegada a la base de datos costó 5 segundos:

Una de las razones que hace que la consulta sea más lenta se debe a que hay que agregar la información dos veces en dos listas distintas: en la primera la que no incluye las proyecciones y demás operaciones y luego viene todo el proceso de filtrar para conseguir los datos necesarios que se agregan en la segunda lista (método getClientesConsumoAdminAnalyze() de DAOCliente).