**“Iteración 4 – Diseño físico u optimización de consultas”**

Claudia D. Bedoya Motta, Johnathan S. Salamanca Lancheros

Diseño físico y optimización de consultas

Universidad de los Andes, Bogotá, Colombia

{cd.bedoya212, js.salamanca1967 [}@uniandes.edu.co](mailto:%7d@uniandes.edu.co)

Fecha de presentación: Mayo 4 de 2014

Tabla de contenido

[1 Introducción 1](#_Toc386977570)

[2 Análisis 1](#_Toc386977571)

[3 Diseño de la aplicación 1](#_Toc386977572)

[4 Diseño físico 2](#_Toc386977573)

[5 Construcción de la aplicación 21](#_Toc386977574)

# Introducción

El presente documento tiene como objetivo revisar los conceptos de vistos en clase con el fin de profundizar y poner en práctica lo aprendido. Por otro lado, también se busca integrar requerimientos de eficiencia a una aplicación transaccional desarrollada en una arquitectura de tres niveles, con interfaz web y manejo de persistencia en base de datos.

# Análisis

Para cumplir con lo requerido en esta iteración se creó una nueva clase en el modelo del mundo llamada movimiento, la cual representa los movimientos de productos de una bodega a un local o de una bodega a otra. Tiene como atributos la presentación, la cantidad de producto que se movió, el producto, la bodega destino y el lugar de almacenamiento de origen. El diagrama UML se puede encontrar en la carpeta docs del proyecto.

# Diseño de la aplicación

Luego de haber leído los nuevos requerimientos que se pidieron para esta entrega, vimos que era necesario realizar cambios significativos tanto al modelo de tablas como a la aplicación. El impacto de los requerimientos es realmente importante, ya que se creó una nueva tabla llamada MOVIMIENTO. Esta tabla, descrita anteriormente, hace que sea más fácil realizar los requerimientos y sus respectivas consultas, sin afectar la consistencia de los datos.

# Diseño físico

**4.1 Documentación del diseño físico**

Luego de realizar los requerimientos, se analizaron las consultas con el fin de determinar qué índices resultaban pertinentes para optimizar el tiempo de respuesta. Después de esto concluimos que lo mejor era crear los siguientes índices:

Índices asociados al RF12:

**INDEX\_LC\_COD\_ADMINISTRADOR**: Índice en la tabla LOCAL sobre la columna COD\_ADMINISTRADOR. Este índice fue creado para optimizar el tiempo del JOIN que realiza la sentencia SQL de este requerimiento (pd.COD\_USUARIO=lc.COD\_ADMINISTRADOR), con el fin de que se ejecute un TWO-PASS JOIN BASED ON SORTED INDEX.

**INDEX\_PD\_COD\_USUARIO**: Índice en la tabla PEDIDO, sobre la columna COD\_USUARIO. Este índice fue creado para optimizar el tiempo del JOIN que realiza la sentencia SQL de este requerimiento (pd.COD\_USUARIO=lc.COD\_ADMINISTRADOR)

**INDEX\_PD\_FECHA\_CREACION**: Índice en la tabla PEDIDO sobre la columna FECHA\_CREACION. Este índice fue creado para optimizar la parte de la consulta pd.FECHA\_CREACION>=?, con el fin de que se ejecute una búsqueda por rangos en el correspondiente árbol B+.

**INDEX\_PD\_FECHA\_ENTREGA**: Índice en la tabla PEDIDO, sobre la columna FECHA\_ENTREGA. Este índice fue creado para optimizar el tiempo de la parte de la cónsulta pd.FECHA\_ENTREGA>=?, con el fin de que se ejecute una búsqueda por rangos en el correspondiente árbol B+.

Índices asociados al RF13:

**INDEX\_PL\_COD\_LUGAR\_ALMAC**: Índice en la tabla PRODUCTO\_LUGAR\_ALMAC, sobre la columna COD\_LUGAR\_ALMAC. Se creó con el fin de optimizar el JOIN en la parte de la sentencia li.ID=pl.COD\_LUGAR\_ALMAC, para que se ejecute un TWO-PASS JOIN BASED ON SORTED INDEX. Cabe resaltar que la otra columna ya tiene un índice creado por Oracle al ser una PK.

**INDEX\_PL\_COD\_PRESENTACION**: Índice en la tabla PRODUCTO\_LUGAR\_ALMAC, sobre la columna COD\_PRESENTACION. Este índice fue creado para optimizar la eficiencia del JOIN sobre las columnas pe.ID=pl.COD\_PRESENTACION, con el fin de que se ejecute un TWO-PASS JOIN BASED ON SORTED INDEX. Cabe resaltar que el índice correspondiente a ID en la tabla PRESENTACION fue creado automáticamente por Oracle ya que es una PK.

**INDEX\_LA\_COD\_TIPO\_PRODUCTO**: Índice en la tabla LUGAR\_ALMACENAMIENTO, sobre la columna COD\_TIPO\_PRODUCTO. Este índice fue creado con el fin de optimizar la eficiencia de la parte de la consulta li.COD\_TIPO\_PRODUCTO=?, con el fin de que se realice una eficiente en el respectivo árbol B+.

Índices asociados al RF14 y RF15

**INDEX\_MB\_COD\_LUGAR\_ORIGEN:** Índice en la tabla MOVIMIENTO, sobre la columna COD\_LUGAR\_ORIGEN. Este índice fue creado con el fin de optimizar la parte de la consulta mo.COD\_LUGAR\_ORIGEN=?, para hacer más eficiente la búsqueda con el respectivo árbol B+.

**INDEX\_MB\_COD\_LUGAR\_DESTINO:** Índice en la tabla MOVIMIENTO, sobre la columna COD\_LUGAR\_DESTINO. Este índice fue creado con el fin de optimizar la parte de la consulta mo.COD\_LUGAR\_DESTINO=?, para hacer más eficiente la búsqueda con el respectivo árbol B+.

**INDEX\_MB\_FECHA\_MOVIMIENTO:** Índice en la tabla MOVIMIENTO sobre la columna FECHA\_MOVIMIENTO. Este índice fue creado con el fin de optimizar la parte de la consulta mo.FECHA\_MOVIMIENTO<?,con el fín de que se realiza una búsqueda por rango en el respectivo árbol B+

**INDEX\_MB\_COD\_PRODUCTO:** Índice en la tabla MOVIMIENTO sobre la columna COD\_PRODUCTO. Este índice fue creado con el fin de optimizar la parte de la consulta mo.COD\_PRODUCTO=?, para hacer más eficiente la búsqueda con el respectivo árbol B+.

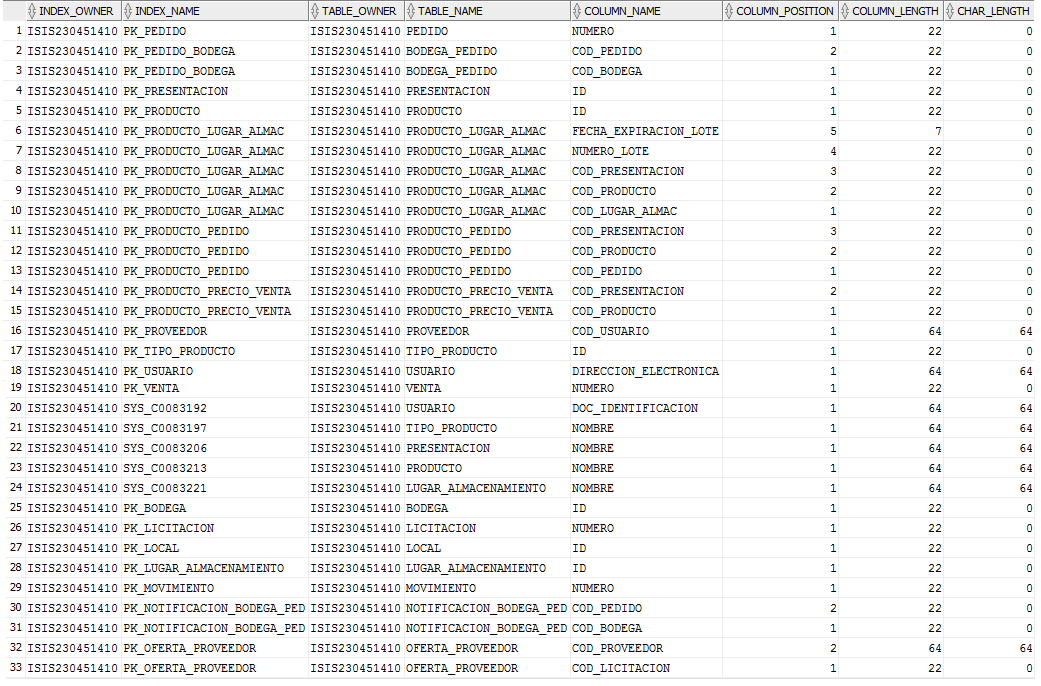
**INDEX\_MB\_COD\_PRESENTACION:** Índice en la tabla MOVIMIENTO sobre la columna COD\_PRESENTACION, Este índice fue creado con el fin de optimizar la parte de la consulta mo.COD\_PRESENTACION=?, para hacer más eficiente la búsqueda con el respectivo árbol B+.

**INDEX\_MB\_CANTIDAD**: Índice en la tabla MOVIMIENTO sobre la columna CANTIDAD. Este índice fue creado con el fin de optimizar la parte de la consulta mo.CANTIDAD<>?, para hacer más eficiente la búsqueda con el respectivo árbol B+.

(Para el otro requerimiento en lugar de = es <>)

Como podemos ver, estos índices nos van a ayudar a que el tiempo de ejecución de las consultas sea corto y se realicen de forma eficiente. Sin embargo es importante tener en cuenta el costo que tiene tener estos índices, por lo cual fueron creados los índices que son estrictamente necesarios para nuestras consultas, analizando si son realmente efectivos. Entre algunos de estos costos de almacenamiento y mantenimiento está el hecho de que las estructuras de datos sobre las cuales están los índices ocupan espacio. También está el hecho de que las operaciones de inserción, eliminación y actualización ocupan recursos, por lo cual es costoso el mantenimiento de estas estructuras de datos donde se guardan los índices.

A continuación mostramos los índices generados automáticamente por Oracle:



Como se puede ver estos índices fueron creados sobre las llaves primarias automáticamente por Oracle, lo cual nos muestra que tener índices por llaves primarias hace que todo sea muy eficiente, ya que son estos registros los que determinan cualquier tupla de la tabla, es decir, si necesitamos la tupla XXXX y tenemos la llave primaria con un índice, el tiempo de búsqueda de esta tupla es muy rápido, contando con que las PK son únicas (no se repiten en la tabla). Si no estuvieran estos índices se debería leer toda la tabla con el fín de encontrar dicha información. Por otro lado, podemos ver que también se crearon índices sobre las columnas que fueron creadas con la condición UNIQUE, que en la imagen anterior tienen el INDEX\_NAME comenzando por ‘SYS\_’. Esto se da ya que estas columnas fueron identificadas como únicas, es decir, sus registros no se repiten, por lo cual Oracle determina que generar índices sobre estas agilizaría de manera increíble las consultas sobre estas tablas, y es por esto que crea índices.

* 1. **Documentación del análisis realizado**

**Requerimiento Funcional:** *Visualizar movimientos de productos 1*

* + **Documentación escenarios de prueba:**

**Sentencia SQL:**

SELECT \* FROM MOVIMIENTO mo WHERE

mo.COD\_PRODUCTO=? AND

mo.CANTIDAD=? AND

mo.COD\_PRESENTACION=? AND

mo.FECHA\_MOVIMIENTO>=? AND

mo.FECHA\_MOVIMIENTO<=? AND

mo.COD\_LUGAR\_ORIGEN=? AND

mo.COD\_LUGAR\_DESTINO=?;

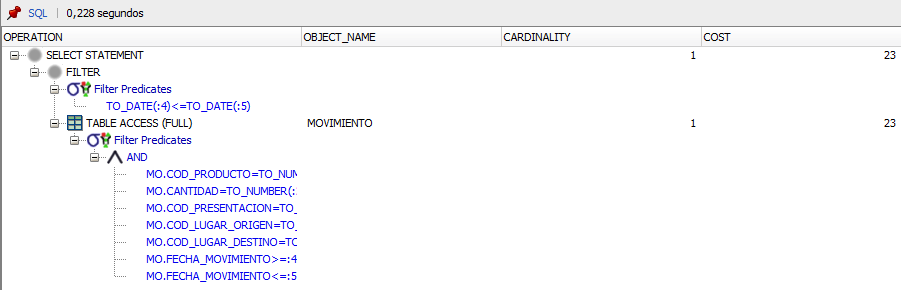
**Distribución datos respecto a los parámetros entrada:**

Teniendo en cuenta que el nivel de astringencia de este requerimiento es bastante exigente, se escogieron los parámetros Bodega de origen, fecha mínima y fecha máxima para garantizar un buen filtro de resultados.

|  |  |  |  |
| --- | --- | --- | --- |
| Bodega | Fecha Min | Fecha Max | Tamaño resultado |
| 1 | 14/02/2010 | 14/02/2012 | 0 |
| 1 | 14/02/2010 | 14/02/2014 | 28 |
| 1 | 14/02/2010 | 14/02/2016 | 33 |
| 2 | 14/02/2010 | 14/02/2012 | 0 |
| 2 | 14/02/2010 | 14/02/2014 | 0 |
| 2 | 14/02/2010 | 14/02/2016 | 0 |
| 17 | 14/02/2010 | 14/02/2012 | 0 |
| 17 | 14/02/2010 | 14/02/2014 | 7 |
| 17 | 14/02/2010 | 14/02/2016 | 7 |
| 19 | 14/02/2010 | 14/02/2012 | 0 |
| 19 | 14/02/2010 | 14/02/2014 | 21 |
| 19 | 14/02/2010 | 14/02/2016 | 21 |
| 20 | 14/02/2010 | 14/02/2012 | 0 |
| 20 | 14/02/2010 | 14/02/2014 | 4 |
| 20 | 14/02/2010 | 14/02/2016 | 6 |
| 22 | 14/02/2010 | 14/02/2012 | 0 |
| 22 | 14/02/2010 | 14/02/2014 | 0 |
| 22 | 14/02/2010 | 14/02/2016 | 8 |
| 44 | 14/02/2010 | 14/02/2012 | 0 |
| 44 | 14/02/2010 | 14/02/2014 | 29 |
| 44 | 14/02/2010 | 14/02/2016 | 45 |

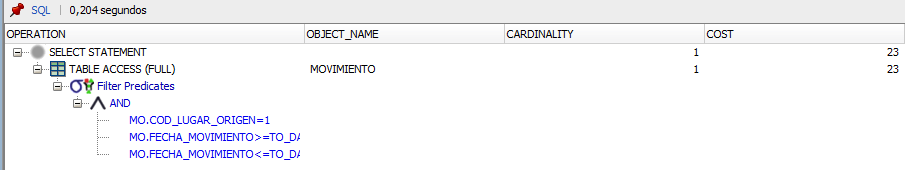
Asimismo, analizando los tests desarrollados el tiempo máximo en que se desarrolló las consultas fue

**Plan de acción general dado por Oracle (sin índices):**



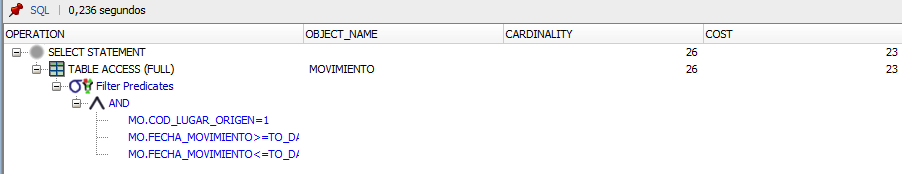
**Caso 1 (sin indices):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bodega | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 1 | 14/02/2010 | 14/02/2012 | 0 | 80 |



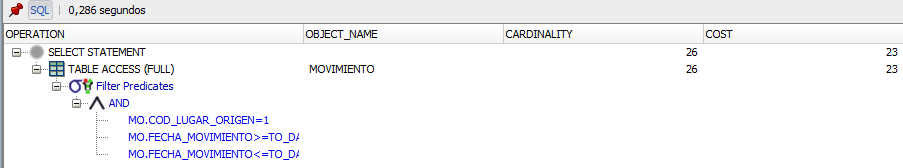
**Caso 2 (sin indices):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bodega | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 1 | 14/02/2010 | 14/02/2014 | 28 | 100 |



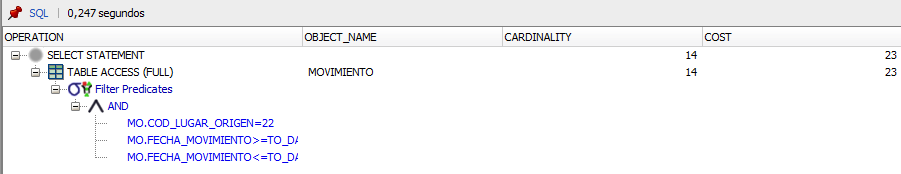
**Caso 3 (sin indices):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bodega | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 1 | 14/02/2010 | 14/02/2016 | 33 | 20 |



**Caso 4 (sin indices):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bodega | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 22 | 14/02/2010 | 14/02/2016 | 8 | 17 |



* **Análisis de eficiencia:**

**Índices planteados:**

INDEX\_MB\_COD\_LUGAR\_ORIGEN

INDEX\_MB\_COD\_LUGAR\_DESTINO

INDEX\_MB\_FECHA\_MOVIMIENTO

INDEX\_MB\_COD\_PRODUCTO

INDEX\_MB\_COD\_PRESENTACION

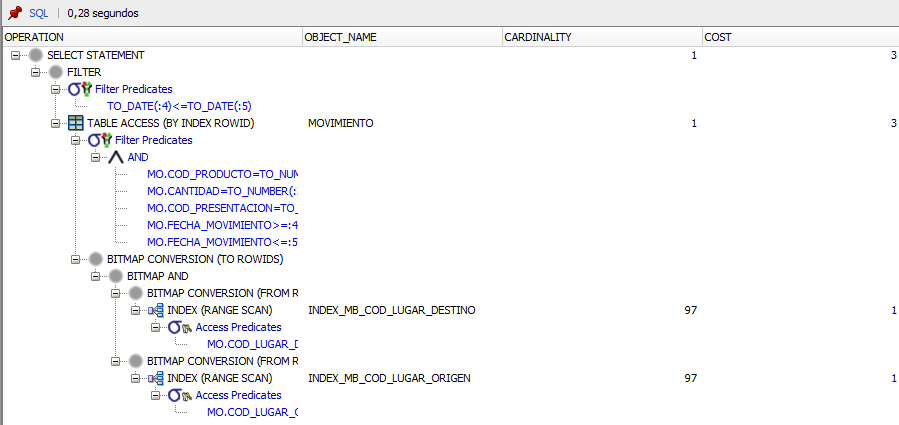
INDEX\_MB\_CANTIDAD

Para mejorar la eficiencia de la búsqueda se plantearon los anteriores índices tal que la búsqueda pase a ser por medio de los índices propuestos y no solamente por la PK que tomó Oracle.

**Plan de ejecución planteado:**

Atendiendo a lo anterior se plantea que el plan de ejecución a seguir debe ser, en primer lugar, un recorrido en la tabla por los índices creados, para lo cual se espera que recorra normalmente (en el árbol B+ de los índices) para los valores de FECHA\_MOVIMIENTO, COD\_PRODUCTO, CANTIDAD y COD\_PRESENTACION. Asimismo, teniendo en cuenta que los valores de COD\_LUGAR\_ORIGEN y COD\_LUGAR\_DESTINO son valores diferentes, puesto que no se encuentra un valor igual en toda la tabla, se espera que para estos dos valores Oracle realice un ordenamiento en un BitMap para ambos atributos de la tabla y realizará una búsqueda por rangos en el respectivo árbol B+, lo cual mejorará enormemente el tiempo.

**Plan de ejecución general planteado por los índices creados:**



Se puede observar que la predicción ha sido bastante acertada puesto que el acceso a toda la tabla se da por medio de los índices y, además, se observa un ordenamiento de los dos atributos planteados (COD\_LUGAR\_ORIGEN y COD\_LUGAR\_DESTINO) por BitMap y ealizando la búsqueda por rangos.

**Comparación con el plan de ejecución default de Oracle:**

En primer lugar se puede observar claramente que los índices generados han reducido enormemente el costo de realizar la consulta dado las unidades que muestra el plan de ejecución. Esto se observa teniendo en cuenta que ahora en vez de realizar un recorrido total sobre la tabla ahora debe realizar simplemente un acceso a la tabla por los índices.

* **Visualizar movimientos de productos 2:**
  + **Documentación escenarios de prueba:**

**Sentencia SQL:**

SELECT \* FROM MOVIMIENTO mo WHERE

mo.COD\_PRODUCTO<>? AND

mo.CANTIDAD<>? AND

mo.COD\_PRESENTACION<>? AND

mo.FECHA\_MOVIMIENTO<? AND

mo.FECHA\_MOVIMIENTO>? AND

(mo.FECHA\_MOVIMIENTO<? OR mo.FECHA\_MOVIMIENTO>?) AND

mo.COD\_LUGAR\_ORIGEN<>? AND

mo.COD\_LUGAR\_DESTINO<>?;

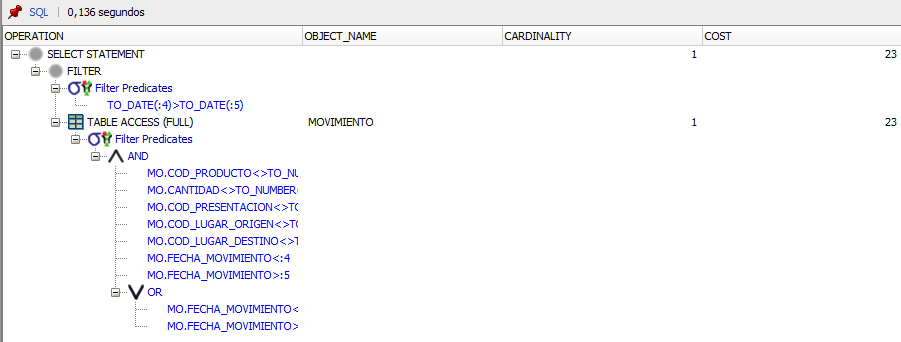
**Distribución datos respecto a los parámetros entrada:**

Teniendo en cuenta que el nivel de astringencia de este requerimiento es bastante exigente, se escogieron los parámetros Bodega de origen, producto, fecha mínima y fecha máxima para garantizar un buen filtro de resultados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bodega | Producto | Fecha Min | Fecha Max | Tamaño resultado |
| 1 | NA | 14/02/2010 | 14/02/2012 | 22123 |
| 1 | NA | 14/02/2010 | 14/02/2014 | 4495 |
| 1 | NA | 14/02/2010 | 14/02/2016 | 0 |
| 1 | 1 | 14/02/2010 | 14/02/2012 | 21908 |
| 1 | 1 | 14/02/2010 | 14/02/2014 | 4449 |
| 1 | 1 | 14/02/2010 | 14/02/2016 | 0 |
| 1 | 2 | 14/02/2010 | 14/02/2012 | 21822 |
| 1 | 2 | 14/02/2010 | 14/02/2014 | 4444 |
| 1 | 2 | 14/02/2010 | 14/02/2016 | 0 |
| 1 | 3 | 14/02/2010 | 14/02/2012 | 21912 |
| 1 | 3 | 14/02/2010 | 14/02/2014 | 4457 |
| 1 | 3 | 14/02/2010 | 14/02/2016 | 0 |
| 1 | 4 | 14/02/2010 | 14/02/2012 | 21907 |
| 1 | 4 | 14/02/2010 | 14/02/2014 | 4443 |
| 1 | 4 | 14/02/2010 | 14/02/2016 | 0 |
| 1 | 5 | 14/02/2010 | 14/02/2012 | 21921 |
| 1 | 5 | 14/02/2010 | 14/02/2014 | 4457 |
| 1 | 5 | 14/02/2010 | 14/02/2016 | 0 |
| 2 | NA | 14/02/2010 | 14/02/2012 | 22156 |
| 2 | NA | 14/02/2010 | 14/02/2014 | 4500 |
| 2 | NA | 14/02/2010 | 14/02/2016 | 0 |
| 2 | 1 | 14/02/2010 | 14/02/2012 | 21941 |
| 2 | 1 | 14/02/2010 | 14/02/2014 | 4454 |
| 2 | 1 | 14/02/2010 | 14/02/2016 | 0 |
| 2 | 2 | 14/02/2010 | 14/02/2012 | 21855 |
| 2 | 2 | 14/02/2010 | 14/02/2014 | 4449 |
| 2 | 2 | 14/02/2010 | 14/02/2016 | 0 |
| 2 | 3 | 14/02/2010 | 14/02/2012 | 21945 |
| 2 | 3 | 14/02/2010 | 14/02/2014 | 4462 |
| 2 | 3 | 14/02/2010 | 14/02/2016 | 0 |
| 2 | 4 | 14/02/2010 | 14/02/2012 | 21940 |
| 2 | 4 | 14/02/2010 | 14/02/2014 | 4448 |
| 2 | 4 | 14/02/2010 | 14/02/2016 | 0 |
| 2 | 5 | 14/02/2010 | 14/02/2012 | 21954 |
| 2 | 5 | 14/02/2010 | 14/02/2014 | 4462 |
| 2 | 5 | 14/02/2010 | 14/02/2016 | 0 |

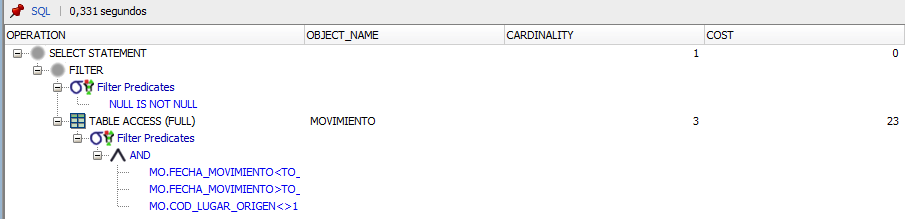
Asimismo, analizando los tests desarrollados el tiempo máximo en que se desarrolló las consultas fue

**Plan de acción general dado por Oracle (sin índices):**



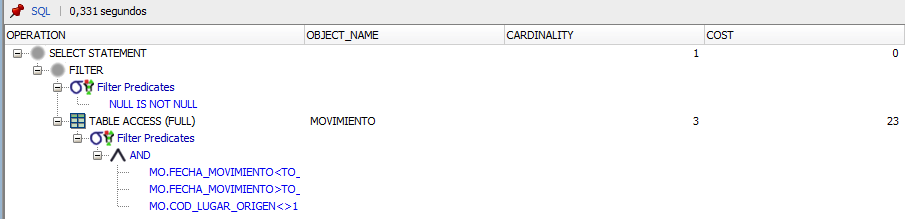
**Caso 1 (sin índices):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bodega | Producto | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 1 | NA | 14/02/2010 | 14/02/2012 | 22123 | 20 |



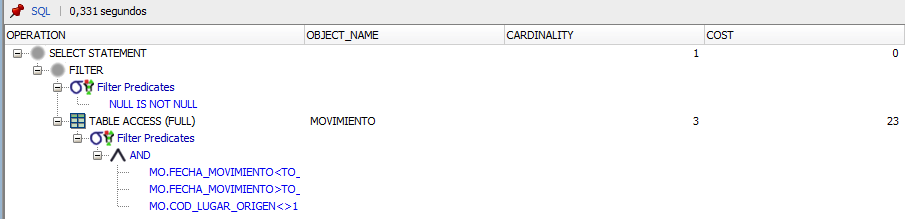
**Caso 2 (sin índices):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bodega | Producto | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 1 | NA | 14/02/2010 | 14/02/2014 | 4495 | 20 |



**Caso 3 (sin índices):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bodega | Producto | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 1 | 1 | 14/02/2010 | 14/02/2016 | 0 | 20 |



* **Análisis de eficiencia:**

**Índices planteados:**

INDEX INDEX\_MB\_COD\_LUGAR\_ORIGEN

INDEX\_MB\_COD\_LUGAR\_DESTINO

INDEX\_MB\_FECHA\_MOVIMIENTO

INDEX\_MB\_COD\_PRODUCTO

INDEX\_MB\_COD\_PRESENTACION

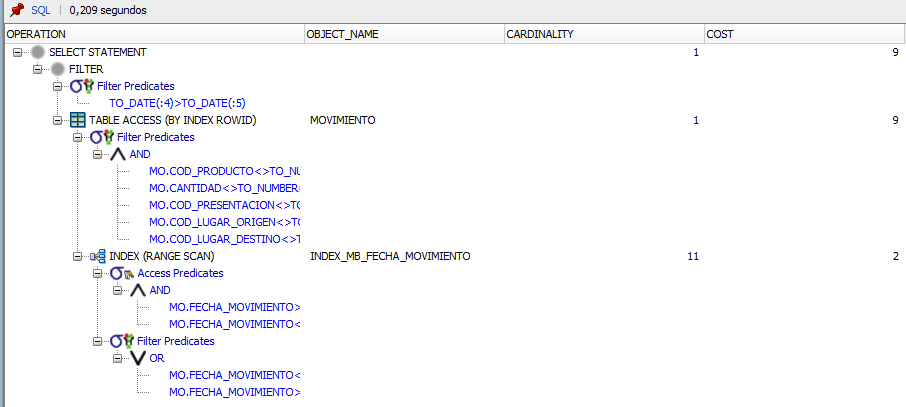
INDEX\_MB\_CANTIDAD

Para mejorar la eficiencia de la búsqueda se plantearon los anteriores índices tal que la búsqueda pase a ser por medio de los índices propuestos y no solamente por la PK que tomó Oracle.

**Plan de ejecución planteado:**

Teniendo en cuenta que este requerimiento es similar al anterior se puede esperar que el plan de ejecución resultante sea similar al mismo. Atendiendo a lo anterior se plantea que el plan de ejecución a seguir debe ser, en primer lugar, un recorrido en la tabla por los índices creados, para lo cual se espera que recorra normalmente para los valores de COD\_PRODUCTO, CANTIDAD y COD\_PRESENTACION. Sin embargo, atendiendo a que ahora se busca diferencia y no igualdad el ordenamiento de los atributos COD\_LUGAR\_ORIGEN y COD\_LUGAR\_DESTINO se espera que se dé de manera normal (INDEXROWID). De la misma manera, para el atributo FECHA\_MOVIMIENTO se espera que realice un Range Scan por el índice puesto que el rango que se está teniendo ahora es un rango que no es acotado.

**Plan de ejecución general planteado por los índices creados:**



Se puede detallar que el plan de ejecución obtenido está de acuerdo con el plan de ejecución planteado lo cual implica que se realizó un buen análisis de los índices y sus efectos sobre el plan de ejecución.

**Comparación con el plan de ejecución default de Oracle:**

En primer lugar se puede observar claramente que los índices planteados han reducido enormemente el costo de realizar la consulta dado las unidades que muestra el plan de ejecución. Esto se observa teniendo en cuenta que ahora en vez de realizar un recorrido total sobre la tabla ahora debe realizar simplemente un acceso a la tabla por los índices.

Teniendo en cuenta lo anterior se puede observar que ahora en vez de realizar un recorrido total de todos los registros de la tabla lo realiza a través de los índices lo cual reduce en gran forma el costo asociado a recorrer los atributos COD\_PRODUCTO, CANTIDAD,COD\_PRESENTACION, COD\_LUGAR\_ORIGEN y COD\_LUGAR\_DESTINO. A la hora de recorrer el atributo FECHA\_MOVIMIENTO se observa un buen ahorro de costo al realizar la búsqueda por escaneo de rangos de los índices.

* **Visualizar Bodegas V2:**
  + **Documentación escenarios de prueba:**

**Sentencia SQL:**

WITH TAB AS (

SELECT pl.COD\_LUGAR\_ALMAC, pl.CANTIDAD\*pe.CAPACIDAD AS OCUPACION

FROM PRODUCTO\_LUGAR\_ALMAC pl

INNER JOIN PRESENTACION pe ON pe.ID=pl.COD\_PRESENTACION

INNER JOIN LUGAR\_ALMACENAMIENTO li ON li.ID=pl.COD\_LUGAR\_ALMAC

INNER JOIN BODEGA bi ON bi.ID=li.ID

WHERE li.COD\_TIPO\_PRODUCTO=?),

TABL AS (

SELECT ta.COD\_LUGAR\_ALMAC, sum(ta.OCUPACION) AS TOTAL\_ALMACENADO

FROM TAB ta

GROUP BY ta.COD\_LUGAR\_ALMAC

)

SELECT ti.COD\_LUGAR\_ALMAC, ti.TOTAL\_ALMACENADO, li.CAPACIDAD, bi.ESTADO FROM TABL ti INNER JOIN LUGAR\_ALMACENAMIENTO li ON ti.COD\_LUGAR\_ALMAC=li.ID INNER JOIN BODEGA bi ON bi.ID=li.ID WHERE (ti.TOTAL\_ALMACENADO >= li.CAPACIDAD \* ?) AND li.COD\_TIPO\_PRODUCTO= ?;

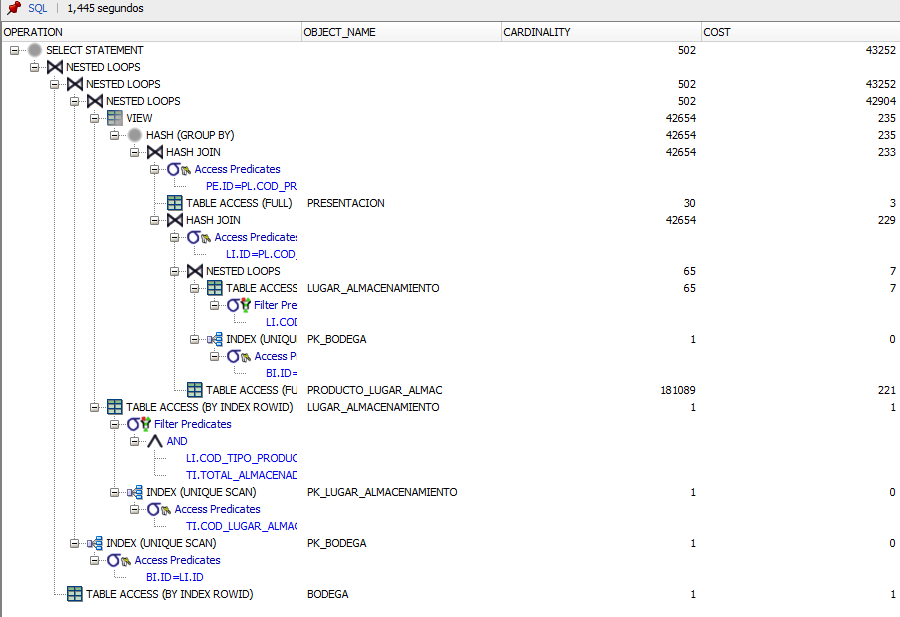
**Distribución datos respecto a los parámetros entrada:**

Teniendo en cuenta que ahora solamente se requieren 2 parámetros: Porcentaje y Tipo producto. Se evalúan todos los resultados para unos porcentajes estándar mostrados a continuación:

|  |  |  |
| --- | --- | --- |
| Porcentaje | Tipo Producto | Tamaño resultado |
| 1 | 1 | 1375 |
| 5 | 1 | 1375 |
| 25 | 1 | 675 |
| 50 | 1 | 16 |
| 75 | 1 | 0 |
| 1 | 2 | 1357 |
| 5 | 2 | 1068 |
| 25 | 2 | 3 |
| 50 | 2 | 0 |
| 75 | 2 | 0 |
| 1 | 3 | 704 |
| 5 | 3 | 52 |
| 25 | 3 | 0 |
| 50 | 3 | 0 |
| 75 | 3 | 0 |

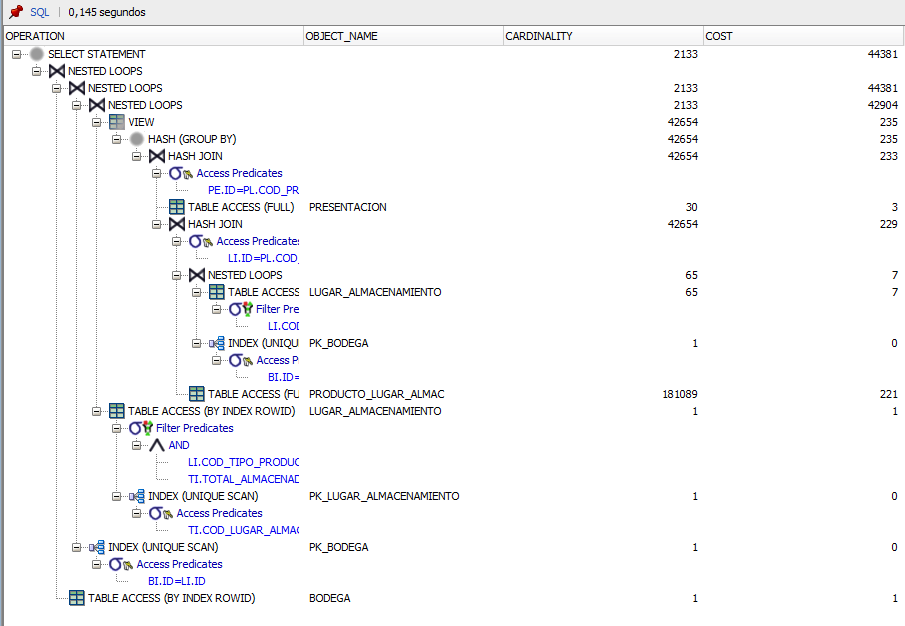
Asimismo, analizando los tests desarrollados el tiempo máximo en que se desarrolló las consultas fue

**Plan de acción general dado por Oracle (sin nuestros índices)**



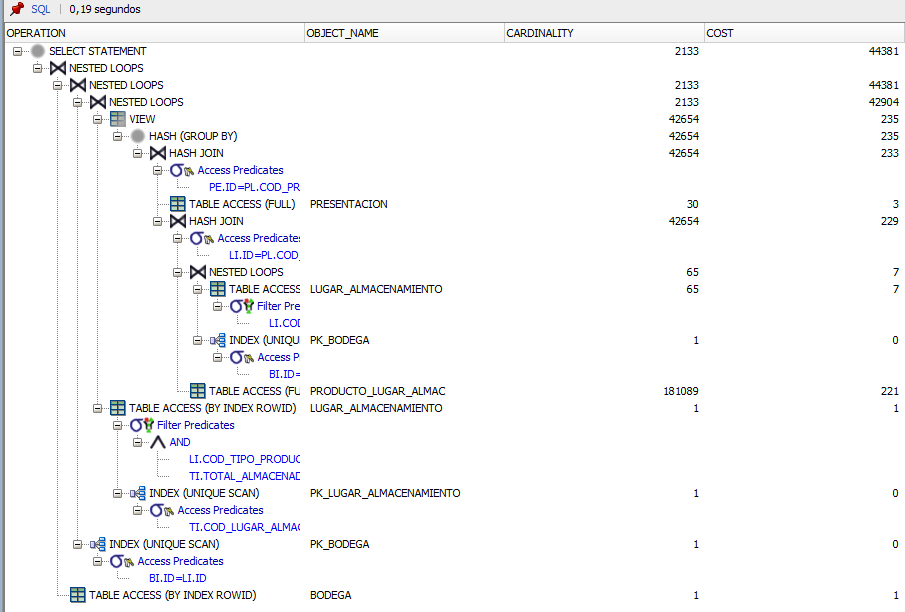
**Caso 1 (sin nuestros índices):**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje | Tipo Producto | Tamaño resultado | Tiempo (ms) |
| 1 | 1 | 1375 | 127 |



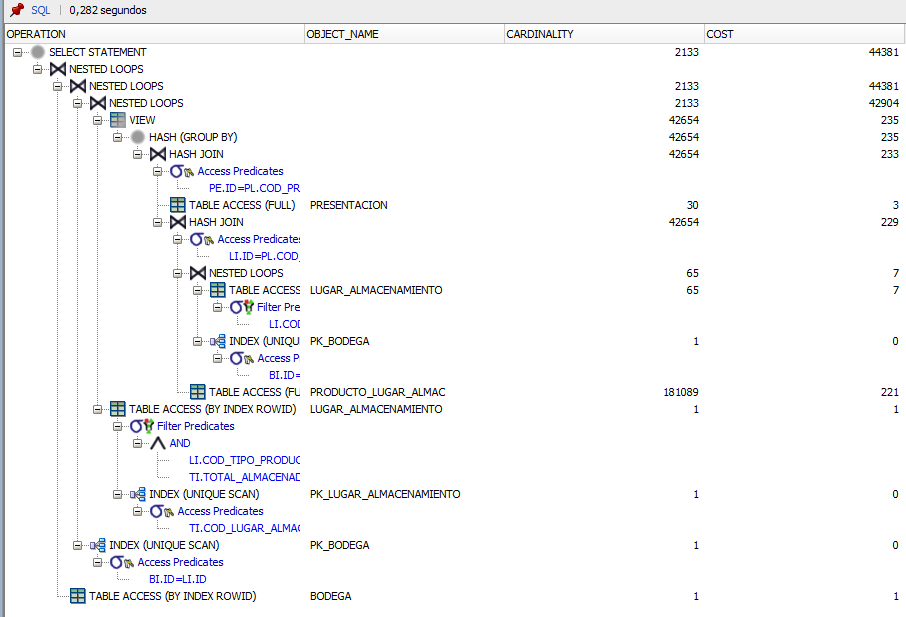
**Caso 2 (sin nuestros índices):**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje | Tipo Producto | Tamaño resultado | Tiempo (ms) |
| 5 | 2 | 1068 | 24 |



**Caso 3 (sin nuestros índices):**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje | Tipo Producto | Tamaño resultado | Tiempo (ms) |
| 25 | 1 | 675 | 29 |



* **Análisis de eficiencia:**

**Índices planteados:**

INDEX\_PL\_COD\_LUGAR\_ALMAC

INDEX\_PL\_COD\_PRESENTACION

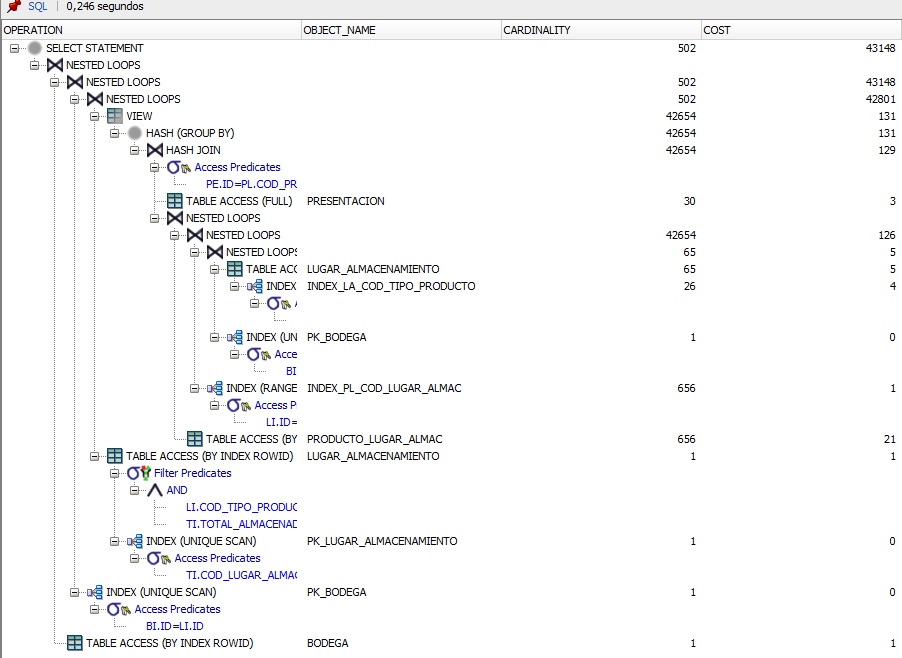
INDEX\_LA\_COD\_TIPO\_PRODUCTO

Para mejorar la eficiencia de la búsqueda se plantearon los anteriores índices tal que la búsqueda pase a ser por medio de los índices propuestos y no solamente por la PK que tomó Oracle.

**Plan de ejecución planteado:**

Atendiendo al hecho que de todas formas debe realizar todos los Joins y ordenamientos que realizaba antes, se plantea que los Nested Loops se mantendrán iguales, al igual que el Hash que realiza para hacer el order by y el Hash Join. La diferencia se verá en el desarrollo interno de las comparaciones y recorridos de la nueva tabla creada, lo que genera un pequeño decrecimiento el costo al realizar las comparaciones por medio de los índices.

**Plan de ejecución general planteado por los índices creados:**



Se puede observar que el plan de ejecución dado por Oracle se ajusta al plan predicho.

**Comparación con el plan de ejecución default de Oracle:**

En primer lugar se puede observar claramente que los índices planteados han reducido enormemente el costo de realizar la consulta dado las unidades que muestra el plan de ejecución. Esto se observa teniendo en cuenta que ahora en vez de realizar un recorrido total sobre la tabla ahora debe realizar simplemente un acceso a la tabla por los índices.

* **Pedidos de los locales**

**Sentencia SQL:**

SELECT pd.NUMERO,pd.COD\_USUARIO,pd.FECHA\_CREACION,pd.FECHA\_ESPERADA\_ENTREGA, pd.FECHA\_ENTREGA,pd.PRECIO\_CABANDES,pd.HA\_SIDO\_CANCELADO,pd.HA\_SIDO\_ENTREGADO FROM PEDIDO pd INNER JOIN LOCAL lc ON pd.COD\_USUARIO=lc.COD\_ADMINISTRADOR WHERE lc.ID=? AND (pd.FECHA\_CREACION>=? OR pd.FECHA\_ENTREGA>=?) AND (pd.FECHA\_CREACION<=? OR pd.FECHA\_ENTREGA<=?);

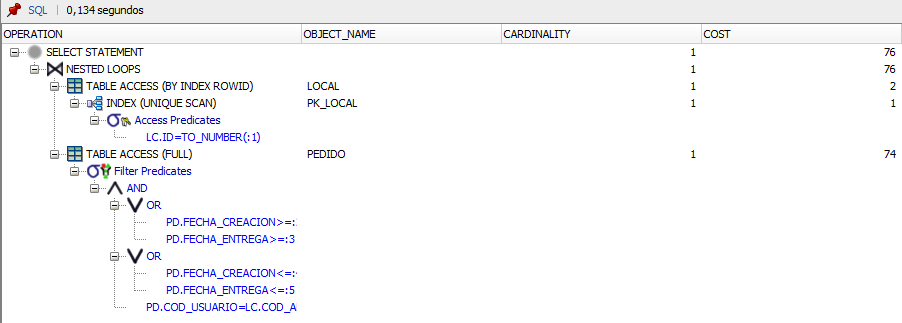
**Distribución datos respecto a los parámetros entrada:**

Teniendo en cuenta que ahora solamente se requieren 3 parámetros: Id Local y el rango de fechas. Se evalúan varios resultados variando el local con un rango de fechas predeterminado para cada uno:

|  |  |  |  |
| --- | --- | --- | --- |
| Local | Fecha Min | Fecha Max | Tamaño resultado |
| 4501 | 14/02/2010 | 14/02/2011 | 0 |
| 4501 | 14/02/2010 | 14/02/2012 | 0 |
| 4501 | 14/02/2010 | 14/02/2014 | 9 |
| 4501 | 14/02/2010 | 14/02/2016 | 10 |
| 4502 | 14/02/2010 | 14/02/2011 | 0 |
| 4502 | 14/02/2010 | 14/02/2012 | 0 |
| 4502 | 14/02/2010 | 14/02/2014 | 5 |
| 4502 | 14/02/2010 | 14/02/2016 | 6 |
| 4503 | 14/02/2010 | 14/02/2011 | 0 |
| 4503 | 14/02/2010 | 14/02/2012 | 0 |
| 4503 | 14/02/2010 | 14/02/2014 | 6 |
| 4503 | 14/02/2010 | 14/02/2016 | 8 |
| 4504 | 14/02/2010 | 14/02/2011 | 0 |
| 4504 | 14/02/2010 | 14/02/2012 | 0 |
| 4504 | 14/02/2010 | 14/02/2014 | 8 |
| 4504 | 14/02/2010 | 14/02/2016 | 8 |
| 4505 | 14/02/2010 | 14/02/2011 | 0 |
| 4505 | 14/02/2010 | 14/02/2012 | 0 |
| 4505 | 14/02/2010 | 14/02/2014 | 5 |
| 4505 | 14/02/2010 | 14/02/2016 | 6 |
| 4506 | 14/02/2010 | 14/02/2011 | 0 |
| 4506 | 14/02/2010 | 14/02/2012 | 0 |
| 4506 | 14/02/2010 | 14/02/2014 | 7 |
| 4506 | 14/02/2010 | 14/02/2016 | 7 |
| 4507 | 14/02/2010 | 14/02/2011 | 0 |
| 4507 | 14/02/2010 | 14/02/2012 | 0 |
| 4507 | 14/02/2010 | 14/02/2014 | 9 |
| 4507 | 14/02/2010 | 14/02/2016 | 10 |
| 4508 | 14/02/2010 | 14/02/2011 | 0 |
| 4508 | 14/02/2010 | 14/02/2012 | 0 |
| 4508 | 14/02/2010 | 14/02/2014 | 1 |
| 4508 | 14/02/2010 | 14/02/2016 | 4 |
| 4509 | 14/02/2010 | 14/02/2011 | 0 |
| 4509 | 14/02/2010 | 14/02/2012 | 0 |
| 4509 | 14/02/2010 | 14/02/2014 | 5 |
| 4509 | 14/02/2010 | 14/02/2016 | 7 |
| 4510 | 14/02/2010 | 14/02/2011 | 0 |
| 4510 | 14/02/2010 | 14/02/2012 | 0 |
| 4510 | 14/02/2010 | 14/02/2014 | 5 |
| 4510 | 14/02/2010 | 14/02/2016 | 11 |
| 4511 | 14/02/2010 | 14/02/2011 | 0 |
| 4511 | 14/02/2010 | 14/02/2012 | 0 |
| 4511 | 14/02/2010 | 14/02/2014 | 4 |
| 4511 | 14/02/2010 | 14/02/2016 | 6 |

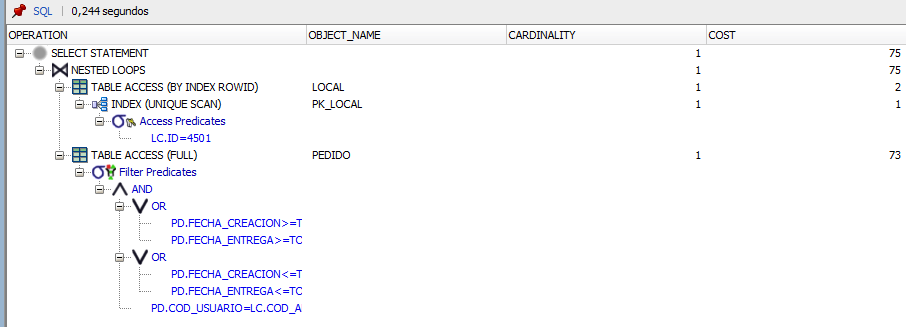
Asimismo, analizando los tests desarrollados el tiempo máximo en que se desarrolló las consultas fue

**Plan de acción dado por Oracle (Sin nuestros índices):**



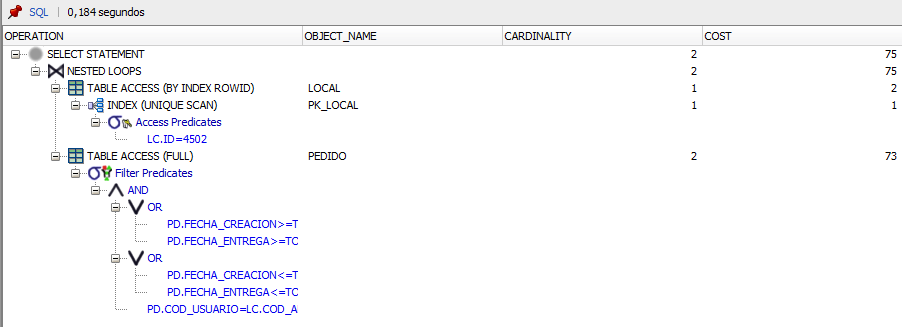
**Caso 1 (Sin nuestros índices):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Local | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 4501 | 14/02/2010 | 14/02/2011 | 0 | 18 |



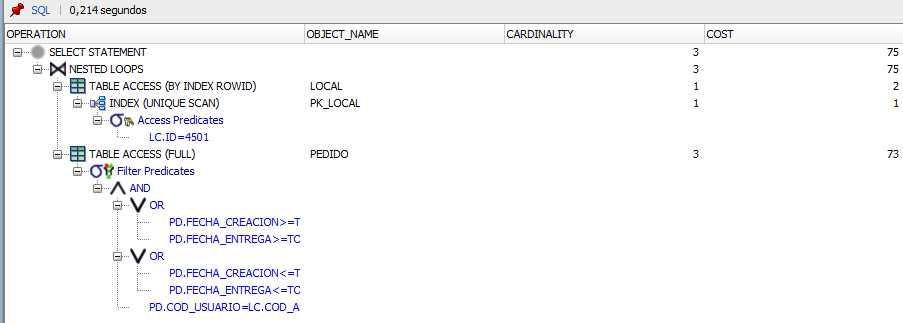
**Caso 2 (Sin nuestros índices):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Local | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 4502 | 14/02/2010 | 14/02/2014 | 5 | 33 |



**Caso 3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Local | Fecha Min | Fecha Max | Tamaño resultado | Tiempo (ms) |
| 4501 | 14/02/2010 | 14/02/2016 | 10 | 25 |



* **Análisis de eficiencia:**

**Índices planteados:**

INDEX\_LC\_COD\_ADMINISTRADOR

INDEX\_PD\_COD\_USUAR

INDEX\_PD\_FECHA\_CREACION

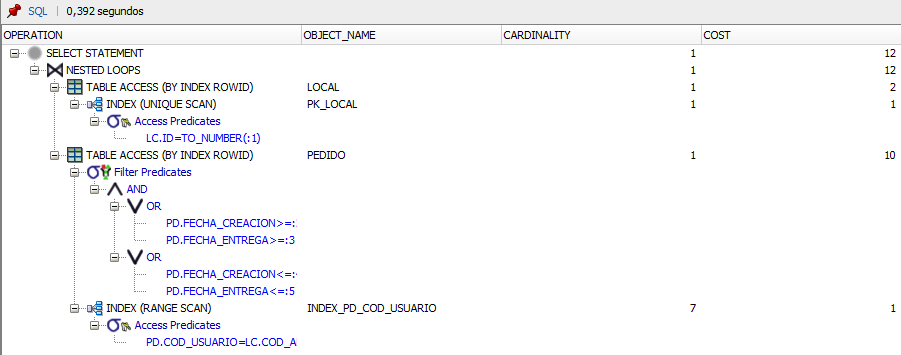
INDEX\_PD\_FECHA\_ENTREGA

Para mejorar la eficiencia de la búsqueda se plantearon los anteriores índices tal que la búsqueda pase a ser por medio de los índices propuestos y no solamente por la PK que tomó Oracle.

**Plan de ejecución planteado:**

Atendiendo al hecho que de todas formas debe realizar todos los Joins y ordenamientos que realizaba antes, se plantea que los Nested Loops se mantendrán iguales, el cambio se dará en el hecho de realizar las comparaciones y filtros al interior de la tabla puesto que ahora se poseerán más índices que los generados por default por parte de Oracle lo cual hará que las búsquedas dentro de la tabla generada ahora se hagan varias por rango de índices y por Index RowId en los atributos restantes lo que generará una reducción el costo de realizar la consulta.

**Plan de ejecución general planteado por los índices creados:**



Se puede observar que se cumple el plan de ejecución predicho.

**Comparación con el plan de ejecución default de Oracle:**

En primer lugar se puede observar claramente que los índices planteados han reducido enormemente el costo de realizar la consulta dado las unidades que muestra el plan de ejecución mostrando una reducción en más de la tercera parte del costo original. Esto se observa teniendo en cuenta que ahora en vez de realizar un recorrido total sobre la tabla ahora debe realizar simplemente un acceso a la tabla por los índices tanto por Index RowID como por Range Scan.

# Construcción de la aplicación

Los cambios efectuados en la aplicación se pueden ver en el archivo Cabandes.

**4.1 Análisis del proceso de optimización**

Si bien el hecho de que el manejo dado por parte del SMBD de Oracle puede volver las consultas algo más lentas que el hecho de traer todo a memoria principal y organizarlo en una estructura de datos diferente para realizar posteriormente el ordenamiento y filtros para realizar los Joins con operaciones lógicas, todo cambia al momento en que se entra a trabajar con volúmenes enormes de datos de forma que no caben en memoria principal y se vuelve inviable realizar un acceso a memoria secundaria cada vez que se desee traer una nueva tupla para agregarla a la base de datos.

En cambio, el optimizador maneja de manera adecuada los atributos y bloques de tuplas de cada una de las tablas utilizadas de tal forma que pueda utilizar diversos algoritmos dependiendo e caso para organizar la información en memoria primaria y garantice el mínimo de accesos a memoria secundaria para optimizar el tiempo de realización de la consulta.

Por ejemplo, supongamos que poseemos dos tablas COMPRADORES y VENTAS, cada una con más de 130’000.000 de tuplas y se pretende realizar la siguiente consulta:

SELECT \* FROM COMPRADORES c INNER JOIN VENTAS v ON c.id\_comprador=v.id\_comprador WHERE v.precio\_venta>=12000;

Se debe observar que, en primer lugar, ninguna de las dos tablas cabe en memoria principal puesto que la cantidad de tuplas que posee cada una es bastante enorme se deben cargar parcialmente en memoria principal, organizar en la estructura de datos determinada y realizar un doble recorrido en pro de encontrar las tuplas que posean el valor del atributo deseado igual. El problema con el que se encuentra el programador aquí es el hecho que no se puede cargar todas las tuplas de las tablas en memoria principal, razón por la cual se deberá importar una a una cada una de las tuplas de memoria secundaria lo cual incurre en un costo 1000 veces mayor que simplemente trabajar todo en memoria principal.

Otro problema también es el hecho de que, una vez leídas las tuplas, se deben meter estas en ciclos (WHILE) con el fin de tratar de simular un JOIN, comparando si los datos hacen match. En el ejemplo propuesto podríamos ver que se debería realizar un doble recorrido para simular el join, en el cual se debe comparar dato por dato para ver si c.id\_comprador==v.id\_comprador. Esto tendría una complejidad de n2, y sabiendo que son más de 130.000.000 registros por tabla, sería algo muy ineficiente, sin contar con el costo del resto de operaciones dependiendo de la consulta que se realice.

Por esta razón es que es mejor utilizar el optimizador de Oracle y confiar en los algoritmos que éste desarrolla (Nested Loops, Hash joins, Index Based Row Id, etc) teniendo en cuenta que pretende optimizar el número de importaciones de memoria secundaria de tal manera que se demore el menor tiempo posible.