**Laboratorio de:** Administración de Base de Datos

**Práctica No.:** 6

**Grupo No.:** 2

**Integrantes:** Aguirre Carlos, Pilatuña Isaac

**Tema:**

Gestión de Rendimiento del Sistema

1. **Objetivos:**
   * Analizar y mejorar el desempeño y funcionamiento de la base de datos a través del uso correcto de índices.
   * Analizar el porcentaje de fragmentación en la base de datos y mejora el rendimiento de base de datos.
   * Uso de SHOWPLAN.
2. **Marco teórico:**
   1. **Rendimiento de Base de Datos**

Se enfoca en el ajuste y optimización del diseño, los parámetros y estructura física de los objetos que componen la base de datos, específicamente las tablas, índices y los archivos en los que se almacenan sus datos. Si no se cuenta con un buen diseño de base de datos, el rendimiento de las consultas no va a verse mejorado sin importar cuantas optimizaciones a las consultas o el sistema se hagan. [1]

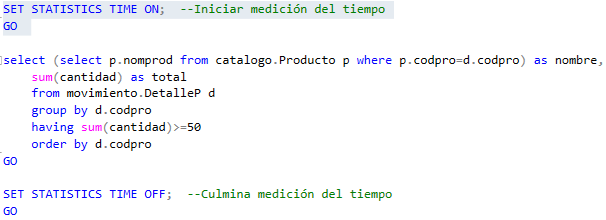
* 1. **Técnicas para la Optimización de Bases de Datos**

Se debe conocer las características de la base de datos para poder aplicar las técnicas adecuadas que optimicen el rendimiento de la base de datos. [1] La mayoría de Sistemas de Bases son compatibles con las siguientes técnicas:

* **Particionamiento:** Dividir una tabla en secciones almacenadas en múltiples archivos.
* **Partición pura frente a sistemas de archivos:** Elegir si el sistema operativo controla o no  los archivos de la base de datos.
* **Indexación:** Elegir índices y opciones que optimicen el rendimiento de las consultas.
* **Desnormalización:** Variar del diseño lógico para conseguir una consultas más eficientes.
* **Agrupación (Clustering):** Hacer cumplir la secuencia física de datos en el disco.
* **Intercalación de datos:** Combinar datos de varias tablas en una sola.
* **Espacio libre:** Dejar espacio para el crecimiento de los datos.
* **Compresión:** Reducir algorítmicamente los requisitos de almacenamiento.
* **Colocación y asignación de archivos:** Colocar los archivos correctos en el lugar correcto.
* **Tamaño de página:** Utilizar el tamaño correcto de página para un almacenamiento eficiente.
* **Reorganización:** Eliminar deficiencias de la bases de datos realineando y reestructurando los objetos de la base de datos.

1. **Desarrollo de la práctica:**
   1. **Listado nombres de artículos con más de 50 ventas y tiempo de ejecución**

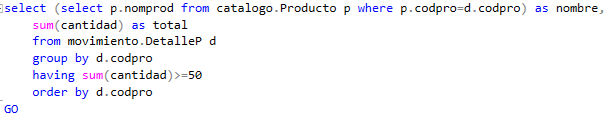
Para realizar este ejercicio se utilizó el script que se puede ver en la Figura 1; el cual se compone de dos funciones del sistema y una consulta a la base de datos compuesta en medio de ambas.



**Figura 1: Script completo de la práctica**

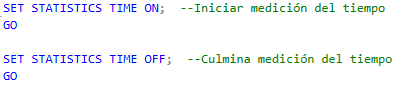
Para conseguir los nombres de los artículos se realizó una consulta compuesta a la base de datos que se muestra en la Figura 2. En primer lugar se realizó un select a los nombres de la tabla Producto; el cual se contuvo en un select con la suma de las cantidades especificadas en cada línea de detalle.

Los resultados se agruparon y ordenaron usando el código del producto de detalle; se especificó que sólo se recuperen los datos cuya suma de cantidades en las líneas de detalle sea mayor a 50.

****

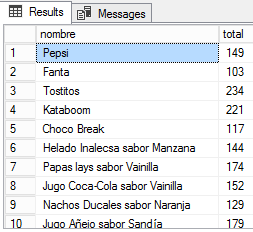
**Figura 2: Query compuesta de dos selects para recuperar los datos**

Para medir el tiempo que se demora en recuperar los datos, se utiliza un comando del DBMS llamado **SET STATISTICS TIME**, que se puede ver en la Figura 3; ésta inicia un contador del sistema para medir el tiempo en mini segundos (ms). Se inició (**ON**) antes de la query compuesta, y se finalizó (**OFF**) tras acabar la query.



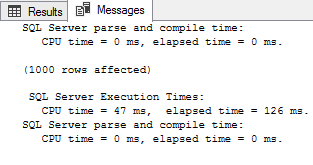
**Figura 3: Funciones para medir tiempo de ejecución del DBMS**

Una vez ejecutado el script, en la sección de resultados se puede ver el nombre de los productos y el total de la suma de las cantidades de las líneas de detalle, comprobando que se recuperaron todos los productos con más de 50 cantidades vendidas en total. En la Figura 4 se expone una parte de estos resultados.

****

**Figura 4: Parte de resultados obtenidos tras ejecutar el script.**

Como se ve en la Figura 5, si se cambia a la pestaña de mensajes se pueden ver los resultados de la medición del tiempo que demoró el DBMS en ejecutar la consulta. Se muestra el tiempo que transcurrido antes de que se inicie la medición, el total de líneas afectadas por la query y los resultados de tiempo de procesamiento (**CPU time**) y el tiempo transcurrido en la ejecución (**elapsed time**).

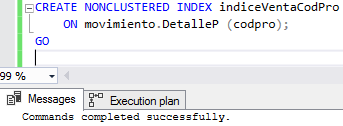
****

**Figura 5: Resultados medición de tiempo**

En este caso el tiempo de procesamiento fue de en promedio 31 mini segundos, mientras que el tiempo transcurrido fue de 214 mini segundos. El total de filas afectadas fue mil (1000), lo que quiere decir que la totalidad de productos han sido vendidos más de 50 veces.

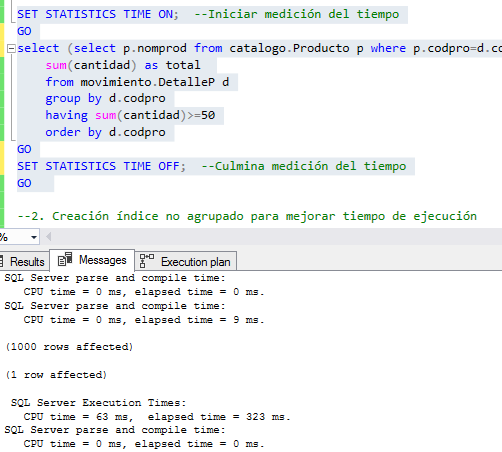
* 1. **Creación de índice no agrupado para disminuir tiempo de respuesta.**

En SQL Server se pueden crear índices agrupados, que son las claves y solo puede existir uno en cada tabla; y los índices no agrupados, que no tienen un límite. Para el caso actual, se decidió que el índice no agrupado fuese creado en el atributo cantidad de las líneas de detalle, pues es la columna más visitada por la consulta que no es una clave primaria o foránea, como se ve en la Figura 6.

****

**Figura 6: Script de creación de índice en atributo “cantidad”**

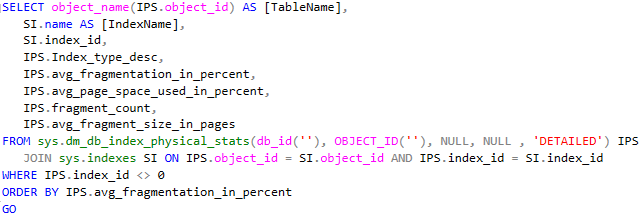
Al realizar exactamente la misma consulta que en el paso uno, pero en esta ocasión con el índice creado, se comprobaron los resultados obtenidos. En la Figura 7 se aprecia que  el DBMS ha ejecutado la consulta en aproximadamente la mitad del tiempo del tiempo de procesamiento y un tercio de tiempo transcurrido, demostrando que la creación del índice no agrupado ha mejorado el rendimiento de la consulta.



**Figura 7: Resultado de consulta tras creación del índice**

* 1. **Determinar y demostrar la cantidad de fragmentación externa e interna**

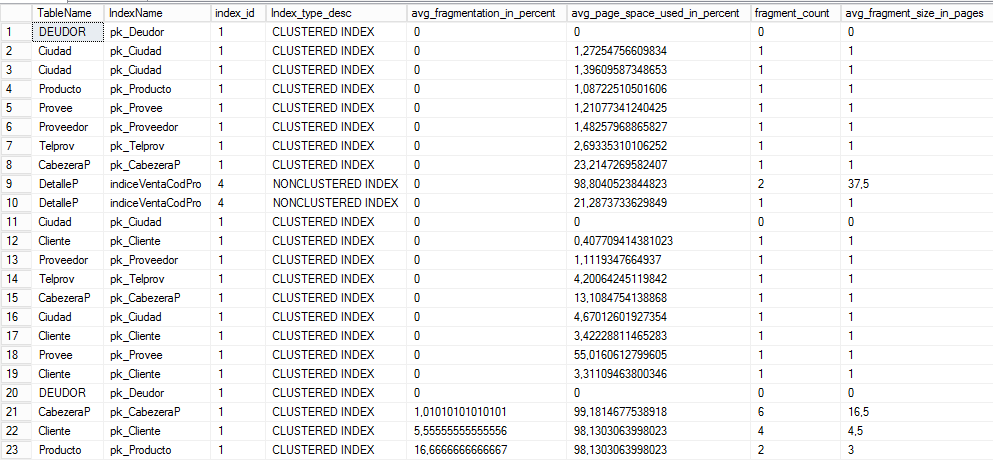
Para saber qué cantidad de fragmentación existe en la base de datos se utilizó una función dinámica que consulta al DBMS características del sistema. En la Figura 10 se muestra el script de la consulta, la cual recupera de la tabla que se indique el índice, su id, tipo, porcentaje de fragmentación y porcentaje de páginas usadas.

****

**Figura 10: Script de función dinámica**

Tras ejecutar la función, se obtuvieron los resultados que se pueden ver en la Figura 11, en este se muestra que la tabla **Producto** es la cual tiene el mayor porcentaje de fragmentación lógica, con un 16%, seguido de la tabla **Cliente** con 5,5% y **CabezeraP** con apenas 1.01%; en otras palabras, estas tablas son las que mayor fragmentación interna poseen en la base de datos.

Por otro lado, a nivel de fragmentación externa se obtuvieron los datos de que la tabla DetalleP cuenta con dos fragmentos, CabezeraP 6, Cliente 4 y Producto dos. Además estos son los únicos que cuentan con valores diferentes a uno para el número de páginas en cada fragmento.

****

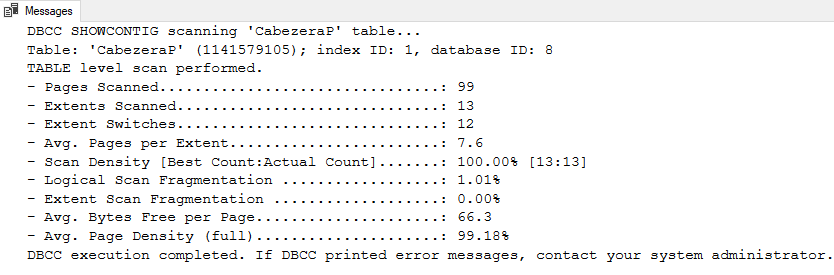
**Figura 11: Resultados de función dinámica**

Además, en la Figura 12 se muestra la función SHOWCONTIG, utilizada para obtener información sobre la fragmentación de tablas específicas. En este caso se desea conocer los datos referentes a la tabla **CabezeraP**.

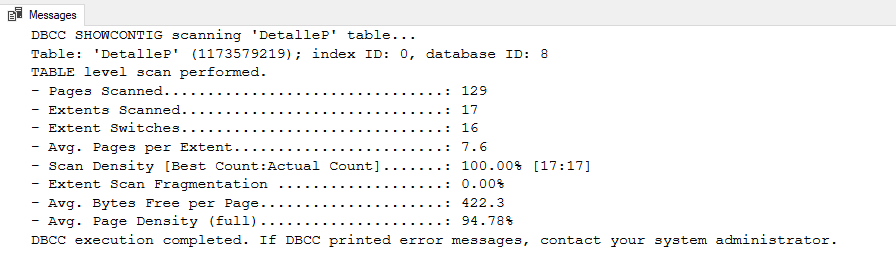
**https://lh3.googleusercontent.com/FAHGHYEuNBL4AQE0uCO77_9CGxoHtK0DfFTrjK2vO_ZlSvm1M3goNxQ4DGfJ_2B76mVE1_n-nlJxeND_U2GldsLRuBJTxlEuNgBQdstPoTGxBVBDbDg0SMrBfh63srQp5UaVDzf3**

**Figura 12: Script información de fragmentación CabezeraP**

En la Figura 13 se detalla los resultados obtenidos de la ejecución del script anterior. En donde se aprecia, gracias al resultado Scan Density, que no existe fragmentación de datos en esta tabla, lo cual coincide con los datos de la función dinámica previamente ejecutada.

**  
Figura 13: Resultados obtenidos de la ejecución del script anterior**

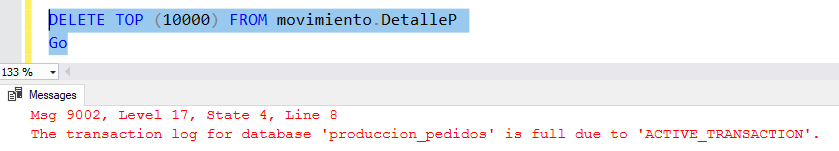
Se repitió el proceso con la tabla DetalleP, y de igualo forma que en el caso anterior, la tabla no presenta partición.

****

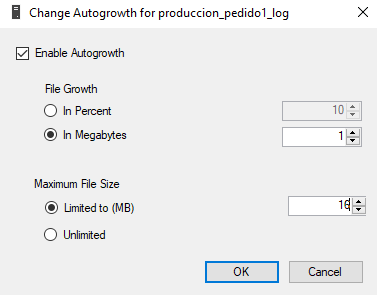
**Figura 14: Resultados obtenidos de la ejecución del script anterior**

* 1. **Eliminar 5000 y 10000 registros aproximadamente delas tablas pedido y detalle respectivamente.**

Se intentó eliminar la cantidad de datos requerida, pero al momento de ejecutar el script para la eliminación de registros el DBMS reportó un mensaje de error debido al límite del tamaño del archivo log, como se ve en la Figura 15.

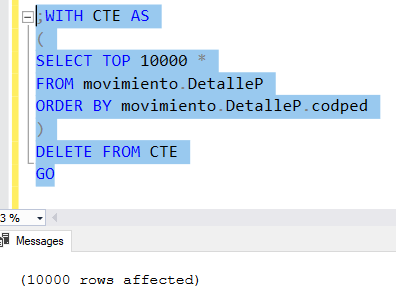
**Figura 15: Error obtenido por la limitación del log**

Para corregir este problema se procedió a incrementar límite máximo del tamaño del archivo log como se muestra en la Figura 16.

****

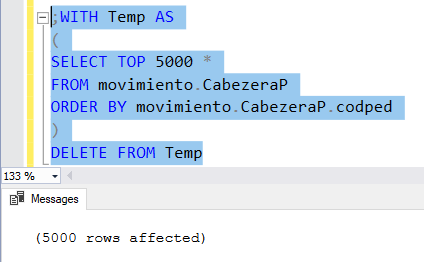
**Figura 16: Incremento del límite del tamaño del archivo log**

El script para la eliminación de los registros de la tabla DetalleP se encuentra en la Figura 17, se procede a eliminarlos de forma ordenada, ya que se encuentran relacionados con las claves primarias de la tabla CabezeraP, por lo cual, para eliminar los registros de la tabla CabezeraP se debe eliminar las llaves foráneas correspondientes.

****

**Figura 17: Script eliminación de 10000 registros DetalleP**

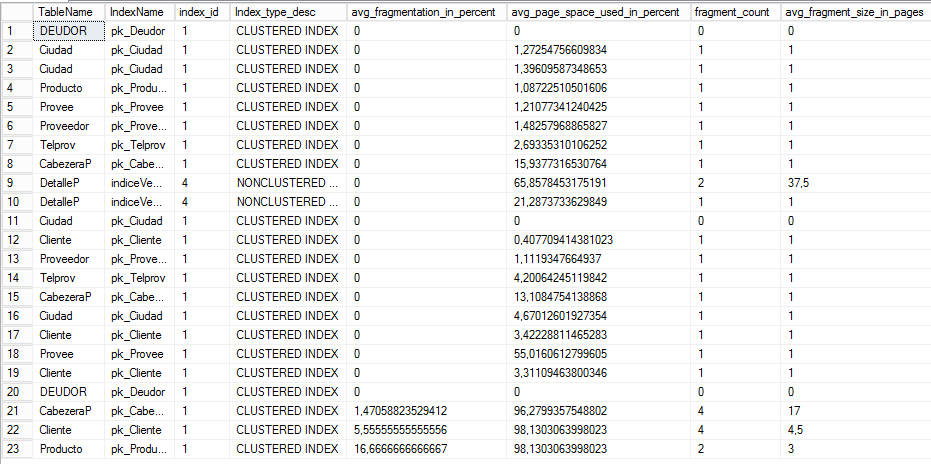
Una vez eliminados los registros relacionados a los 5000 registros que eliminaremos de la tabla CabezeraP, se procede a ejecutar el script de la Figura 18.

**  
Figura 18: Script eliminación de 5000 registros CabezeraP**

* 1. **Confirmar y responder: ¿Cambiaron los valores que corresponden a las tablas pedido y detalle?**

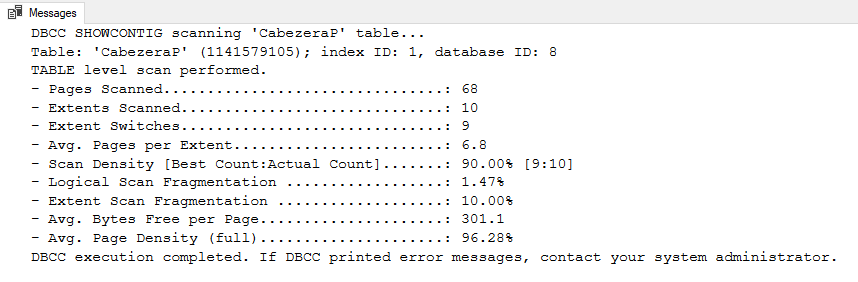
Tras eliminar los registros, se volvieron a ejecutar la función dinámica y las funciones SHOWCONTIG con las mismas tablas que en el paso 3. En la Figura 19 se pueden ver los resultados de la función dinámica. En el caso de la tabla **CabezeraP** se aprecia que el porcentaje de fragmentación se ha incrementado en décimas. Además, se puede observar que la tabla **DetalleP** ha obtenido un porcentaje de fragmentación del 94%.

Además, se puede ver que la cantidad de espacio usado en las páginas ha disminuido del 98% antes de la eliminación a un total de 65%; aunque su fragmentación física y lógica no han cambiado en absoluto.

****

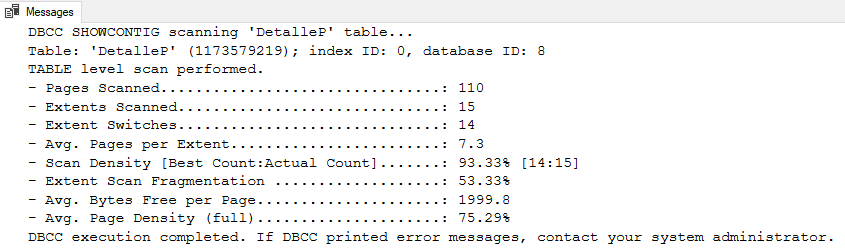
**Figura 19: Resultados de la función dinámica**

En la Figura 20 se puede ver que en comparación con la medición anterior, tras eliminar los registros solicitados, el **Scan Density** ha disminuido a 90% lo que indica que los datos de **CabezeraP** ahora tienen una pequeña fragmentación.

****

**Figura 20: Información de fragmentación de la tabla CabezeraP**

Igualmente en **DetalleP** se comprueba que el porcentaje del mismo resultado ha disminuido a 93%, demostrando la existencia de fragmentación. Como se aprecia en la Figura 21.

****

**Figura 21: Información de fragmentación de la tabla DetalleP**

* 1. **Aplicar la instrucción *ALTER INDEX…REORGANIZE* y/o *ALTER INDEX…REBUILD* basado en los resultados arrojados por la función dinámica anterior.**

**¿En qué casos se aplica cada instrucción basado en la información que se obtiene de la función dinámica anterior?**

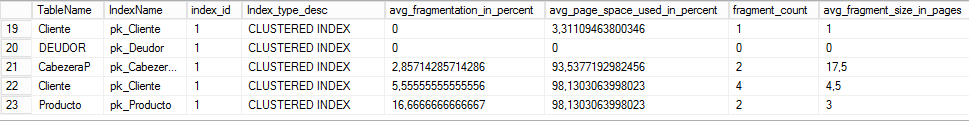
Las funciones requieren conocer el valor del porcentaje de fragmentación de las tablas, obtenido en el paso anterior, con el fin de identificar cuál de ellas ha de usarse. En el caso ***ALTER INDEZ…REORGANIZE***, este debe ser utilizado si el valor de porcentaje de fragmentación se encuentra entre 5 y 30. Si el valor de fragmentación supera el 30%, entonces ha de usarse el ***ALTER INDEX…REBUILD****.*

Como se puede ver en la Figura 19, solo se aplicarían correctamente ***ALTER INDEX…REORGANIZE*** en las tablas **Clientes** (5,5%) y **Producto** (16%). No existe ninguna tabla que amerite que se ejecute la función ***ALTER INDEX...REBUILD***.

Pese a que se ejecutó el script respectivo en las tablas ya mencionadas, como se ve en la Figura 22; no se obtuvo ninguna alteración en los datos sobre la fragmentación de las tablas, como se aprecia en la Figura 23.



**Figura 22: Script *ALTER INDEX…REORGANIZE***

****

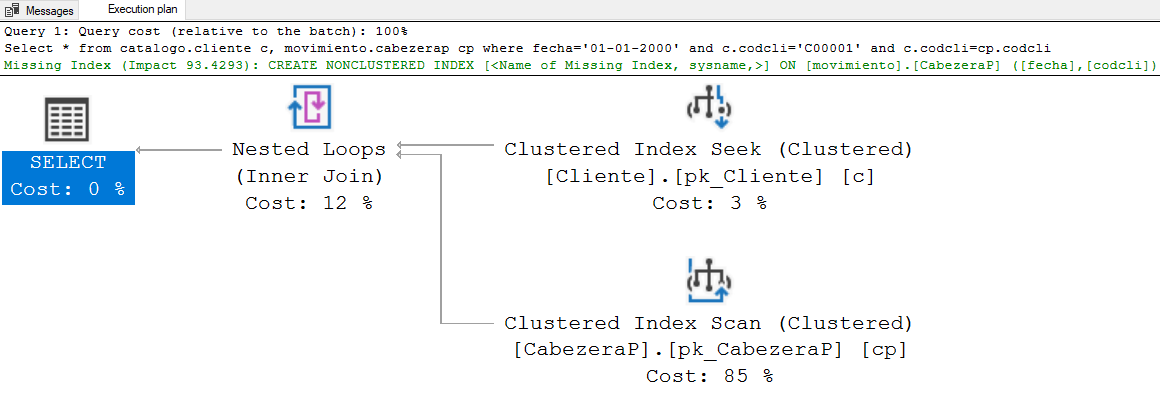
**Figura 23: Datos obtenidos tras función *ORGANIZE***

La tabla DetalleP no fue tomada en cuenta porque el valor de su porcentaje de fragmentación no es apto para ser considerada en el análisis.

* 1. **Utilizando el comando showplan, Interpretar los resultados de la ejecución de las siguientes sentencias:**

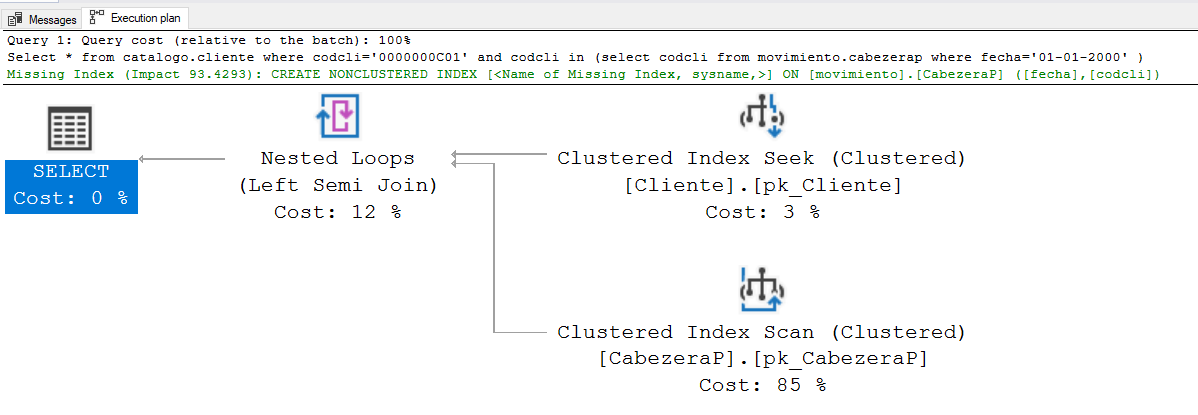
1. Select \* from catalogo.cliente c, movimiento.cabezerap cp where fecha=’01-01-2000’ and c.codcli=‘0000000C01’ and c.codcli=cp.codcli

Al ejecutar el plan de ejecución estimado para la consulta se obtiene como resultado que el mayor costo se generaría en el escaneo de índices agrupados para la llave primaria de la tabla **CabezeraP**, con un costo del 85% del procesamiento en comparación con un 12% de costo debido a los bucles anidados provocados por el Inner Join, finalmente se tiene un costo del 3% de procesamiento para la búsqueda de índices agrupados para la llave primaria  de la tabla Cliente.

**  
Figura 24: Plan de ejecución para la consulta a.**

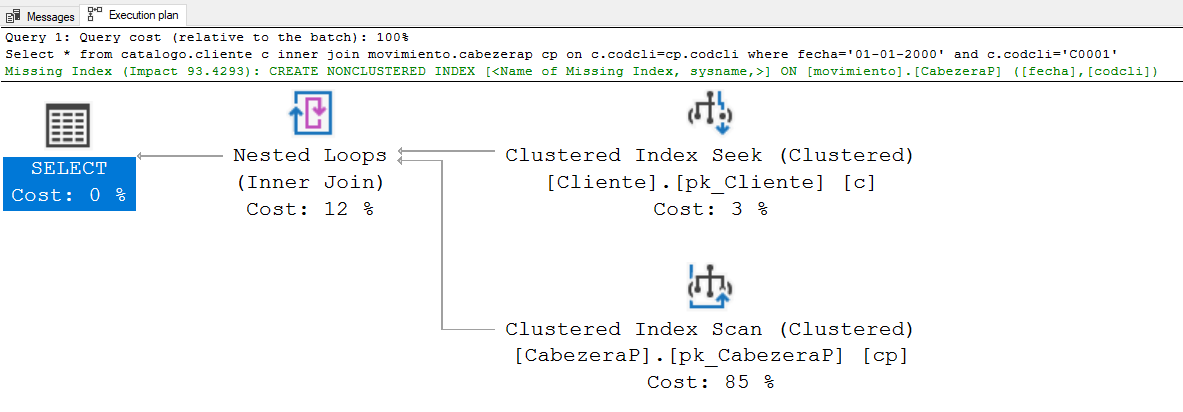
1. Select \* from catalogo.cliente where codcli=‘0000000C01’ and codcli in (select codcli from movimiento.cabezerap where fecha=’01-01-2000’ )

Al ejecutar el plan de ejecución estimado para la consulta se obtiene como resultado que el mayor costo se generaría en el escaneo de índices agrupados para la llave primaria de la tabla **CabezeraP**, con un costo del 85% del procesamiento en comparación con un 12% de costo debido a los bucles anidados provocados por el Semi Join, finalmente se tiene un costo del 3% de procesamiento para la búsqueda de índices agrupados para la llave primaria  de la tabla Cliente.

  
**Figura 25: Plan de ejecución para la consulta b.**

1. Select \* from catalogo.cliente c inner join movimiento.cabezerap cp on c.codcli=cp.codcli where fechapedido=’01-01-2000’ and codcli=‘C01’

Al ejecutar el plan de ejecución estimado para la consulta se obtiene como resultado que el mayor costo se generaría en el escaneo de índices agrupados para la llave primaria de la tabla CabezeraP, con un costo del 85% del procesamiento en comparación con un 12% de costo debido a los bucles anidados provocados por el Inner Join, finalmente se tiene un costo del 3% de procesamiento para la búsqueda de índices agrupados para la llave primaria  de la tabla Cliente.

  
**Figura 26: Plan de ejecución para la consulta c.**

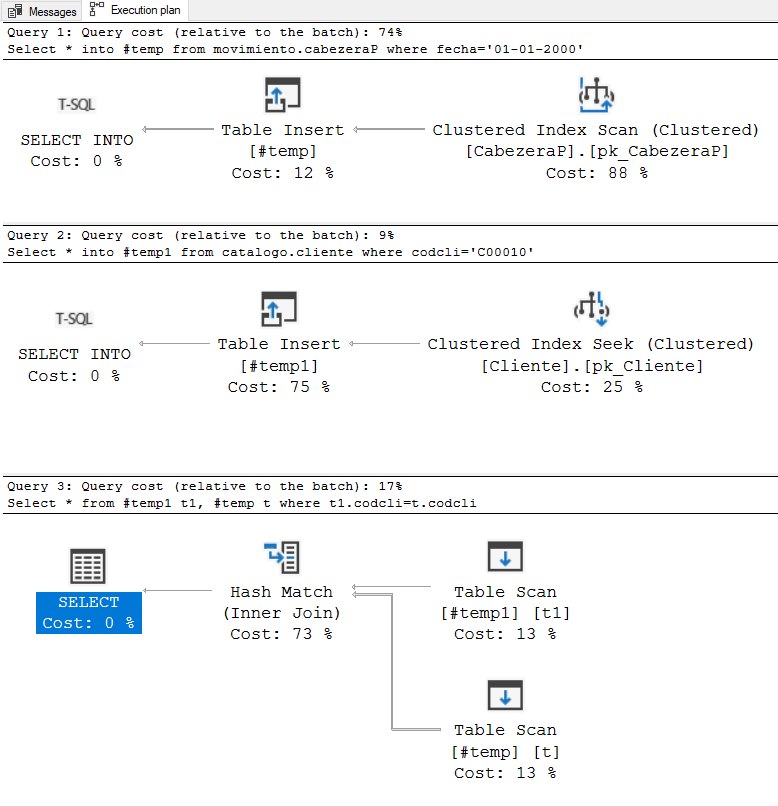
1. Select \* into #temp from movimiento.cabpedido where fechapedido=’01-01-2000’

Select \* into #temp1 from catalogo.cliente where codcli=’0000000C010’

Select \* from #temp1 t1, #temp t where t1.codcli=t.codcli

Go

Como resultado de la primera consulta se tiene un mayor costo en el procesamiento para el escaneo de índices agrupados de la llave primaria de la tabla CabezeraP con un 88% de costo y un 12% de costo para la inserción en la tabla temporal. Para la segunda consulta se tiene un costo del 25% para la búsqueda de índices agrupados de la llave primaria de la tabla Cliente, en comparación con un 75% de costo para la inserción de los registros obtenidos en una segunda tabla temporal. Finalmente, en la ejecución de la visualización del plan para la tercera consulta, se tiene un costo del 73% para la realización del emparejamiento Hash (Inner Join) y un 13% de costo en cada escaneo de las tablas temporales previamente creadas.

****

**Figura 27: Plan de ejecución para la consulta d.**

1. **Conclusiones y Recomendaciones**
   1. **Conclusiones**
2. El rendimiento de una base de datos puede ser mejorado mediante la creación de índices en los atributos más consultados que no son claves primarias. El tiempo de ejecución de una query puede acortarse gracias a la utilización de índices, pero dependerá enteramente del atributo elegido para esta acción.
3. La eliminación de grandes cantidades de registros puede derivar en la formación de una fragmentación de los datos de las tablas. Esto afecta al rendimiento de forma positiva o negativa dependiendo de la cantidad de fragmentos físicos o lógicos de las tablas.
4. La creación de índices no afecta la visualización de los planes de ejecución. Esto se comprobó mediante la función SHOWPLAN del DBMS. Pese a crearse un índice, el DBMS no lo muestra en el plan.
   1. **Recomendaciones**

* Para la creación de índices se debe valorar las columnas en donde se considere necesario la agregación de índices, los factores a tomar en cuenta son: La columna puede ser tomado como parámetro de búsqueda muy frecuentemente como lo son nombres, apellidos, números de cédula, fechas. La columna tiende a ser utilizada como factor de ordenamiento, o si se realizan muchas consultas o actualizaciones en base a cierta columna.
* Previo a la realización de una eliminación o modificación extensiva de registros es recomendable hacer un backup, por si ocurren errores durante la ejecución del script. En el caso del desarrollo de la práctica, se obtuvo el error de la eliminación de diez mil registros aleatorios en lugar de los primeros diez mil como se buscaba.
* Otro paso a tomar en cuenta previo a la realización de una eliminación o modificación extensiva es la configuración del límite máximo del tamaño del archivo log en caso de que se requiera.
* Si se requiere eliminar los registros de una tabla independiente, asegurarse que consten los controles de integridad referencial para evitar mensajes de error.
* En caso de que no se cuente con controles de integridad referencial, es recomendable que los registros de la tabla dependiente a ser eliminados sean ordenados y agrupados, para evitar errores al eliminar los registros de la tabla independiente.

1. **Referencias**

|  |  |
| --- | --- |
| [1] | C. Mullins, Database Administration, New York: Addison-Wesley, 2013. |