**Informe cuarto trabajo - Bases de Datos II – 2021-1**

***Integrantes:***

*Aristizabal Giraldo Salomé - Rendón Giraldo Sebastián - Valencia Zapata Santiago Alexis*

Para el desarrollo de este trabajo, cada uno de los tres puntos tiene una carpeta llamada PuntoX, dónde X puede ser 1, 2 o 3. Cada una de estas carpetas tiene 3 archivos .sql, uno llamado “Datos”, dónde se guarda el programa que se debe ejecutar para la generación de datos aleatorios en ese punto. “Consultas”, dónde se almacenan las consultas que se proponen para ser analizadas, y “Tablas”, dónde se encuentran las instrucciones para la construcción de las tablas que se van a analizar en cada punto. En el caso de los puntos 1 y 2, las instrucciones de “Tablas” y “Consultas” son las dadas por el profesor en el planteamiento del trabajo, mientras que en el punto 3, son las instrucciones requeridas para correr el ejemplo propuesto para este punto.

Así mismo, en las carpetas de los puntos 1 y 2 también podemos encontrar un archivo .docx dónde se han guardado todos los resultados del Explain Plan del punto respectivo, cada uno de ellos titulado con respecto a la consulta y a la cantidad de datos que pertenece. Para los resultados del TKPROF encontraremos una carpeta llamada Resultados TKPROF, en cuyo interior se encuentran una serie de .txt con los resultados respectivos. Cada archivo de esta carpeta está nombrado como TKPROF\_#.txt, donde el # indica los números de los datos utilizados para esos resultados. De vuelta en la carpeta de punto también encontramos un archivo .xlsx dónde encontraremos las tablas que resumen los resultados requeridos para el análisis posterior.

En la carpeta raíz encontramos el archivo Configuraciones.sql, el cual contiene las instrucciones previas antes de realizar el análisis de rendimiento, además del enunciado del trabajo y del informe del trabajo.

A continuación, se presentarán los resultados y los análisis de cada una de estas situaciones:

1. Punto 1

Para el análisis de este punto, se generaron 5 diferentes volúmenes de datos, los cuales fueron usados de igual manera, a excepción de la consulta D, para obtener los resultados. Luego de creadas las tablas, se generaban los datos de manera aleatoria con la función “datos\_punto\_uno(clientes, facturas, detalles)” que en sus tres parámetros recibe cuántos de cada uno de los ítems se planea generar. Se realizaban las consultas y entre consulta se borraba el caché. Dado que para la consulta D se requiere de la creación de índices para algunas tablas, antes de realizar esta, la tabla se volvía a crear vacía para crear los índices antes de la inserción de los datos. Luego de obtener los resultados requeridos, la tabla se borraba por completo para evitar uso de índices en otras consultas.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Consulta A** | **Volumen de datos** | | | **Explain Plan** | | **TKPROF tasas** | | |
| ***Clientes*** | ***Facturas*** | ***Detalles*** | ***Cost*** | ***Time (s)*** | ***(f+g)/h*** | ***i/j*** | ***k/(f+g)*** |
| 100 | 1000 | 5000 | 13 | 1 | 0,0742 | 14,9254 | 0,0000 |
| 1000 | 10000 | 50000 | 68 | 1 | 0,0714 | 14,9925 | 0,0168 |
| 5000 | 30000 | 100000 | 535 | 7 | 0,0718 | 14,9970 | 0,0149 |
| 10000 | 80000 | 240000 | 1520 | 19 | 0,0718 | 14,9991 | 0,0408 |
| 15000 | 100000 | 480000 | 1591 | 20 | 0,0712 | 14,9995 | 0,0283 |

**Tabla 1. Resultados rendimiento consulta A punto 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Consulta B** | **Volumen de datos** | | | **Explain Plan** | | **TKPROF tasas** | | |
| ***Clientes*** | ***Facturas*** | ***Detalles*** | ***Cost*** | ***Time (s)*** | ***(f+g)/h*** | ***i/j*** | ***k/(f+g)*** |
| 100 | 1000 | 5000 | 13 | 1 | 0,0748 | 14,9254 | 0,0241 |
| 1000 | 10000 | 50000 | 68 | 1 | 0,0714 | 14,9925 | 0,0168 |
| 5000 | 30000 | 100000 | 535 | 7 | 0,0718 | 14,9970 | 0,0149 |
| 10000 | 80000 | 240000 | 1520 | 19 | 0,0718 | 14,9991 | 0,0408 |
| 15000 | 100000 | 480000 | 1591 | 20 | 0,0712 | 14,9995 | 0,0671 |

**Tabla 2. Resultados rendimiento consulta B punto 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Consulta C** | **Volumen de datos** | | | **Explain Plan** | | **TKPROF tasas** | | |
| ***Clientes*** | ***Facturas*** | ***Detalles*** | ***Cost*** | ***Time (s)*** | ***(f+g)/h*** | ***i/j*** | ***k/(f+g)*** |
| 100 | 1000 | 5000 | 16 | 1 | 0,0080 | 14,9254 | 0,2250 |
| 1000 | 10000 | 50000 | 1078 | 13 | 0,0049 | 14,9925 | 0,2439 |
| 5000 | 30000 | 100000 | 2770 | 34 | 0,0053 | 14,9970 | 0,2008 |
| 10000 | 80000 | 240000 | 3484 | 42 | 0,0053 | 14,9991 | 0,9961 |
| 15000 | 100000 | 480000 | 5830 | 70 | 0,0048 | 14,9995 | 0,9979 |

**Tabla 3. Resultados rendimiento consulta C punto 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Consulta D** | **Volumen de datos** | | | **Explain Plan** | | **TKPROF tasas** | | |
| ***Clientes*** | ***Facturas*** | ***Detalles*** | ***Cost*** | ***Time (s)*** | ***(f+g)/h*** | ***i/j*** | ***k/(f+g)*** |
| 100 | 1000 | 5000 | 2 | 1 | 1,4342 | 14,9254 | 0,0046 |
| 1000 | 10000 | 50000 | 3 | 1 | 1,6169 | 14,9925 | 0,0028 |
| 5000 | 30000 | 100000 | 9 | 1 | 1,8881 | 14,9970 | 0,0024 |
| 10000 | 80000 | 240000 | 17 | 1 | 1,9538 | 14,9991 | 0,0000 |
| 15000 | 100000 | 480000 | 23 | 1 | 1,8925 | 14,9995 | 0,0036 |

**Tabla 4. Resultados rendimiento consulta D punto 1**

En las tablas 1 a 4 podemos observar los volúmenes de datos utilizados, y es de resaltar que son idénticos para todas las consultas. También podemos observar que las tablas 1 y 2 son prácticamente idénticas a excepción del dato de la esquina inferior derecha. Esto se debe a que, por defecto, la consulta A, que era ejecutada sin “Hints”, se ejecuta por defecto tal como se indica en la B con el “Hint” “Full” para cada tabla, es decir, realizaba “TABLE ACCESS FULL” acompañado de un “HASH JOIN” en ambos casos, esto se puede ver en el anexo de los resultados del “Explain Plan”.

También podemos identificar que la métrica “i/j” se muestra prácticamente estática en un valor muy cercano a 15, sin importar el volumen de datos generado o la consulta realizada. Lo que quiere decir que en todo momento el uso del “array fetch” fue consistente, retornando un número similar de filas por cada ida y venida de este objeto de la base de datos.

Hablando específicamente de las consultas A y B, cuyos resultados se encuentran en las tablas 1 y 2 respectivamente, podemos ver en el “Explain Plan” que el Costo y el tiempo de ejecución de la consulta crece de manera más o menos lineal con respecto volumen de datos, de manera no muy explosiva, conservando un radio de entre 100 y 1000 datos por cada unidad de costo, y los tiempos de ejecución aún con muchos datos no pasan de los 20 segundos. Por el otro lado, las métricas extraídas del TKPROF, como el “(f+g)/h” nos muestra que es relativamente barato extraer una fila de la tabla, puesto que cada una requiere de 0,07 consultas al buffer aproximadamente, bastante por debajo de 20 que es el valor máximo recomendado, mientras que el “k/(f+g)” nos muestra que las consultas directas al disco, que son las más costosas de realizar, no superan el 7% de las realizadas en buffer, encontrándose por debajo del 10% máximo recomendado.

Pasando a la consulta C, representada en la tabla 3, su “Explain Plan” nos muestra un crecimiento más explosivo del costo y del tiempo requerido para la misma, al aumentar en gran cantidad el volumen de datos. Es así como mientras en las consultas anteriores no pasaban de los 1600 en costo, esta consulta ya tiene un costo de más de 2000 en el tercer volumen de datos, llegando a más de 5800 unidades de costo en el mayor volumen. Así mismo, el tiempo de la consulta se eleva a poco más de medio minuto con el tercer volumen de datos, y a minuto y diez segundos en el volumen más grande. Al observar el TKPROF vemos que el costo de buffer por fila extraída es incluso más bajo que en las consultas anteriores, pero al mirar el porcentaje de consultas a discos con respecto a las de buffer, podemos ver que se encuentra entre un 20% para los volúmenes bajos, y cerca del 100% para los volúmenes más altos de datos, superando con creces el 10% máximo recomendado. Esta consulta, que se compone de varios SORT JOIN y MERGE JOIN según su “Explain Plan” adjunto en la carpeta del punto, da muestras de ser bastante costosa tanto en ejecución como en recursos.

La consulta D por el contrario, como se muestra en la tabla 4, tiene un costo mucho más bajo en comparación con todas las consultas anteriores, mostrando tan solo 23 unidades de costo en el volumen más alto de datos. Esto también se ve reflejado en el costo de tiempo, dónde parece mantenerse prácticamente constante en 1 segundo sin importar el volumen de datos. El TKPROF nos muestra que el costo de buffer de retornar una fila es bastante mayor en comparación a las anteriores consultas, rondando entre 1,5 y 2 consultas a buffer por fila, con tendencia a la alta a la par que el volumen de datos. Aún así, la consulta se mantiene muy por debajo de las 20 consultas por fila recomendadas. Por otra parte, el porcentaje de consultas a disco es menor al 0,5% sin importar el volumen de datos consultado. Recordemos que las consultas a buffer son menos costosas que las que se hacen a disco, y esto pudo ser el factor que determinó el bajo costo de la consulta en el “Explain Plan”. Esta consulta fue realizada apoyado en la creación de índices para las tablas Factura y Detalle, lo que puede ser el factor clave para la reducción de las consultas a disco en favor de las consultas al buffer.

Para finalizar, teniendo en cuenta todo lo anterior expuesto, podemos encontrar que para esta consulta es aceptable realizar una consulta basada en HASH JOIN, como lo son las dos primeras consultas, puesto que presentan un rendimiento y unos costos relativamente buenos, si los comparamos con los valores recomendados. Aún así, es aún más recomendable realizar una consulta como la D, apoyada en la creación de índices para las tablas a consultas, puesto que nos permitirá obtener un rendimiento excepcional en comparación de las otras opciones disponibles. Por otra parte, no es para nada recomendable aplicar una consulta como la C, puesto que presenta costos muy altos, tanto en ejecución, como en recursos.

1. Punto 2

Para realizar el análisis de rendimiento de estas consultas se hizo uso de 5 volúmenes de datos, que se pueden observar en la tabla 5 y 6. Luego de la creación de las tablas, se hizo uso de la función “datos\_punto\_dos(numeros)” dónde en el parámetro “números” se ingresa el número de filas de números que queremos tener en la tabla en ese momento. Para realizar las consultas, de manera previa se implementó el borrado de caché para mantener la igualdad de condiciones entre los estados de arranque de las consultas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Consulta A** | **Volumen de datos** | **Explain plan** | | **TKPROF** |
| ***Cost*** | ***Time (s)*** | ***CPU*** |
| 10000 | 14 | 1 | 0,03 |
| 100000 | 112 | 2 | 0,17 |
| 500000 | 545 | 7 | 0,6 |
| 1000000 | 1088 | 14 | 1,31 |
| 2000000 | 2171 | 27 | 3,14 |

**Tabla 5. Resultados rendimiento consulta A punto 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Consulta B** | **Volumen de datos** | **Explain plan** | | **TKPROF** |
| ***Cost*** | ***Time (s)*** | ***CPU*** |
| 10000 | 14 | 1 | 0,54 |
| 100000 | 112 | 2 | 7,43 |
| 500000 | 545 | 7 | 27,12 |
| 1000000 | 1088 | 14 | 60,25 |
| 2000000 | 2171 | 27 | 118,82 |

**Tabla 6. Resultados rendimiento consulta B punto 2**

Se puede observar al comparar la tabla 5 con la 6, que los resultados de costo en el “Explain Plan” son completamente iguales para ambas consultas, la diferencia entre las tablas está marcada por la información que brinda el TKPROF para el tiempo de uso de la CPU, dónde para la consulta 1, con 2’000.000 de datos, apenas llega a usar 3,14 segundos, mientras que la consulta B, con tan solo 100.000 datos, ya hace uso de 7,43 segundos, y mientras tanto, a los 2’000.000 el uso llega hasta los 118,82 segundos.

La similaridad dentro del “Explain Plan” hace completo sentido al comparar ambas consultas directamente. Podremos observar que ambas hacen un llamado a las filas de la tabla número para aplicar una serie de operaciones matemáticas sobre los números que componen cada una de las filas. En términos de “fetch” de filas, ambas consultas realizan lo mismo.

Sin embargo, podemos decir que el diablo está en los detalles. Mientras para la consulta A, las operaciones matemáticas que se aplican sobre los valores de las filas extraídas son multiplicaciones y divisiones, para la consulta B a cada número antes de multiplicar o dividir debe aplicársele un logaritmo natural, lo que es una operación bastante más compleja para procesar por parte de la CPU. Esto no es una tarea propia de la consulta, sino de los recursos con los que cuente el computador dónde se hospeda el servicio, lo que puede dar cabida a que no se tome en cuenta al momento de hacer estimaciones con respecto a las consultas.

Estos resultados nos muestran que a pesar de que el Explain Plan puede darnos una idea de lo costoso que puede ser una consulta, este tipo de detalles quedan por fuera de sus estimaciones, y es precisamente en situaciones como esta donde la complementariedad entre el Explain Plan y el TKPROF al momento de optimizar una consulta, salta a la vista.

1. Punto 3

Los índices bitmap son índices especiales usan bitmaps o listas bit. Oracle almacena un bitmap por cada clave del índex, cada una de estas claves guarda apuntadores a múltiples filas. Cada bitmap contiene un solo bit (0 o 1) por cada fila de la tabla un 1 indica que la fila tiene el valor especificado por el mapa de bits y un 0 indica que no lo tiene. Esto permite acelerar el tiempo de consulta.

|  |  |
| --- | --- |
| Genero | ID |
| M | 1210 |
| F | 320 |
| F | 875 |
| M | 2398 |
| M | 9304 |

**Tabla 7. Representación de una tabla común en una base de datos**

Un índice bitmap en la columna genero presentaría la siguiente información:

|  |  |
| --- | --- |
| M | F |
| 1 | 0 |
| 0 | 1 |
| 0 | 1 |
| 1 | 0 |
| 1 | 0 |

**Tabla 8. Representación una columna con un índice bitmap para género**

¿Dónde usar índices bitmap? En columnas con baja cardinalidad, las cuales definimos como columnas con menos de 100 valores diferentes (cabe destacar que a menor cardinalidad más efectivo es el uso de índices bitmap para mejorar la eficiencia de las consultas); por ejemplo, una columna de genero solo tendría los valores de “masculino” “femenino”, esta es una columna con baja cardinalidad en la que se pueden usar los índices bitmap. Adicionalmente en tablas de solo lectura, o que se actualizan con poca frecuencia ya que mantener un índice bitmap toma muchos recursos y usarlo en tablas muy utilizadas puede causar bloqueos.

Los índices de mapa de bits se utilizan ampliamente en entornos de almacenamiento de datos. Ya que presentan:

* Tiempo de respuesta reducido para grandes clases de consultas
* Requisitos de almacenamiento reducidos en comparación con otras técnicas de indexación
* Aumentos elevados del rendimiento
* Mantenimiento eficiente

Ejemplo:

Se presenta un ejemplo que posee una tabla con información de posibles arrendadores para una vivienda, la mayoría de las columnas de la tabla son de baja cardinalidad, por lo que a la hora de realizar consultas para escoger el arrendador perfecto para una vivienda es recomendado el uso de índices bitmap ya que mejoraría considerablemente el tiempo de respuesta de las consultas. Este ejemplo lo podemos encontrar en los archivos de la carpeta Punto3 como se explicó al principio del informe.