INGENIERÍA DE SISTEMAS

ASIGNATURA: BASE DE DATOS I

CONTENIDISTA: Ing. Silvia Cobialca

UNIDAD 7:

Almacenamiento en disco y performance

INDICE UNIDAD 7

MAPA DE LA UNIDAD 7

PROPÓSITOS

En esta unidad nos proponemos explicarle algunos conceptos más avanzados del almacenamiento de los objetos que conforman las bases de datos, y cómo a partir de su gestión se puede mejorar el tiempo de respuesta de las consultas. Estos son conceptos avanzados que le permitirán optimizar tanto la gestión de los datos como su recuperación .

OBJETIVOS

- ✓ Aprender la técnica necesaria para organizar el almacenamiento de los distintos objetos de las bases de datos
- ✓ Conocer cómo armar las estructuras de archivos a utilizar en una base de datos.
- ✓ Diferentes técnicas de optimización del almacenamiento físico.

CONTENIDOS

Para que alcance los objetivos, los contenidos que abordará son los siguientes:

1) Almacenamiento de datos en disco
2) Estructuras de archivos
3) Diseños orientados a la performance
4) Ejemplos típicos.

PALABRAS CLAVES

Archivos, performance, diseño físico, estructuras de archivos

BIBLIOGRAFÍA DE CONSULTA

Elmasri, Ramez/Navathe, Shamkant (2011). *Fundamentos de Sistemas de Base de Datos.* 6ta Ed, EEUU: Pearson / Addison Wesley.

EVALUACIÓN

A lo largo de cada unidad, encontrará actividades de autoevaluación que te permitirán hacer un seguimiento del aprendizaje.

INTRODUCCIÓN

Para comenzar con esta unidad, partimos desde las Unidades 3 y 4, donde ya estuvimos interactuando con la base de datos utilizando el lenguaje SQL, y vimos cómo se creaban consultas para obtener información. Entonces aprenderemos cómo armar la estructura física de la base de datos para mejorar la velocidad a la que se obtienen los resultados de dichas consultas.

Comencemos…

ABORDEMOS EL LOGRO DEL PRIMER OBJETIVO DE LA UNIDAD:

- ✓ Aprender la técnica necesaria para organizar el almacenamiento de los distintos objetos de las bases de datos.

1.Almacenamiento de datos en disco

En este apartado veremos formas en que podemos guardar dentro de los discos o del almacenamiento disponible los diferentes objetos de la base de datos.

Recordará que en la Unidad 3 habíamos visto consultas y subconsultas que nos servían para obtener información de nuestra base de datos.

Cuando el motor resuelve dichas consultas, el primer paso es ir a buscar los datos a la estructura física creada para contenerlos (el archivo). Durante este proceso, el motor de base de datos interactúa con el sistema operativo para obtener acceso al disco, y le pasa la orden de lectura, se da entre ellos entonces una especie de "conversación" cuyo resultado es que los datos almacenados en el archivo pasan al motor de base de datos y se muestran los resultados (o se los utiliza en otra consulta).
Las operaciones de acceso al disco son las más caras desde el punto de vista del tiempo que insumen para el DBMS. Esto es porque los discos se manejan en tiempos medidos en milisegundos (1.10^{-3} segs) cuando el resto de las operaciones realizadas por los DBMSs se realizan en memoria, cuyos tiempos son medidos en nanosegundos (1.10^{-9} segs).

Podemos ver más información a este respecto en wikipedia:
Discos: https://en.wikipedia.org/wiki/Hard_disk_drive_performance_characteristics

Como vimos en la Unidad 4, en todas las bases de datos, hay una serie de consultas que son más frecuentemente solicitadas que otras. Hay algunas que se correrán todos los días y quizás haya algunas consultas que solo se necesiten a fin de mes, todos los meses o solamente a fin de año.

Surgiran luego las siguientes preguntas ¿Esta frecuencia en que se ejecutan las consultan incide en el diseño físico? O más bien ¿Qué tiene que ver la frecuencia en que se ejecutan las consultas con todo lo demás? O tal vez una pregunta más de fondo ¿Debemos conocer las consultas más frecuentes en el momento de diseñar la estructura física de la base de datos?

Vamos a ir respondiendo a todas estas preguntas a continuación:

Veamos la respuesta a la primera pregunta:
¿Esta frecuencia en que se ejecutan las consultan incide en el diseño físico?

Recordemos que las bases de datos son las que alojan los datos correspondientes a las aplicaciones. Si en las operaciones diarias de nuestra aplicación detectamos que se van a repetir siempre las mismas consultas, por ejemplo, cada usuario consultará su saldo antes de realizar una compra, y luego al realizar una compra se le mostrará la lista de sus últimas operaciones, entonces sabemos que será necesario que estas consultas se resuelvan en el menor tiempo posible para poder permitir que muchos más usuarios realicen operaciones todos los días.
Claramente entonces, debemos mejorar la operación de acceso al disco específicamente para estas consultas, que son las que impactan más en el negocio y en sus ganancias.

Esto nos ayuda a responder tambien la segunda pregunta:
¿Qué tiene que ver la frecuencia en que se ejecutan las consultas con todo lo demás?
Tiene todo que ver! Porque si logramos que el tiempo de respuesta de esas consultas más frecuentes sea menor, los clientes van a poder hacer más compras al día.
Entonces…
¿Debemos conocer las consultas más frecuentes en el momento de diseñar la estructura física de la base de datos?
Claramente que sí!!!

Cuando se reúnen los usuarios con los desarrolladores y comienzan a explicarles como debe ser el funcionamiento de la aplicación, se identifican aquellas funcionalidades claves del día a día y aquellas que impactarán en la facturación.
A partir de este conocimiento, y teniendo en cuenta el diseño que van haciendo de la base de datos, los desarrolladores y diseñadores deben analizar la distribución de los datos de la base de datos que beneficie dichas consultas.

Veamos un ejemplo utilizando la base de datos de Northwind:

En general los usuarios siempre consultan sus pedidos pendientes, lo cual harían con una consulta como la siguiente:

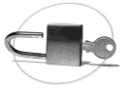
```
select c.ContactName, o.*,od.*
 from orders o join customers c on o.customerid=c.customerid
 join [order details] od on od.orderid=o.orderid
 where ShippedDate is null
```

Para ejecutar esta cosulta, el DBMS define primero si es una consulta que va a demandar mucho trabajo, y si es asi, la separa en partes, las cuales reparte a diferentes procesadores (CPUs) del servidor, luego cada uno debe:

1) Acceder a lo datos más recientes de la tabla orders (los que aun no se han enviado), pero tengamos en cuenta que la tabla orders suele tener los datos de todas las ventas de la empresa (miles y miles de filas de datos)
2) Acceder a los datos más recientes de la tabla [order details], esta tabla contiene tambien miles y miles de filas de datos
3) Acceder a los datos de la tabla customers (como es un maestro, es muy pequeña y su lectura es rápida)
4) Juntar los resultados de 1) , 2) y 3)

El problema sobreviene cuando tenemos mal organizados los archivos que corresponden a las tablas de datos. Lo que corresponde a la "capa física" de la base de datos.
Para acceder a los datos de cada tabla, el DBMS le envía al sistema operativo el nombre del archivo donde están los datos. EL sistema operativo abre el archivo y lee su contenido, enviando al DBMS bloques de datos hasta completar toda la lectura.
Si los archivos que tienen los datos de las tablas están en el mismo disco (o storage), el sistema operativo debe esperar a terminar de leer una tabla para comenzar a leer la otra. Sin embargo, si los archivos estuvieran separados en diferentes discos, esta lectura se puede hacer en paralelo, porque algunas de las CPUs pueden ocuparse de unas tablas y otras de las demás, y siempre habrá una CPU que coordina la separación y junta de los datos.

Cuando ejecutamos esta consulta, el DBMS busca la definición del objeto ventas_por_mes y además, como la tiene guardada en el catálogo ya precompilada, la ejecución es más rápida.



ABORDEMOS EL LOGRO DEL SEGUNDO OBJETIVO DE LA UNIDAD:

- ✓ Conocer cómo armar las estructuras de archivos a utilizar en una base de datos.

2. Estructuras de archivos

Ahora ya podemos pasar a conceptos más concretos ya que hemos comprendido el problema que deben resolver los diseñadores de la base de datos ya que desde el momento en que conocen los requerimientos saben que hay ciertas restricciones de performance asociadas a los mismos. Puntualmente podemos pensar en el ejemplo que vimos en el apartado anterior, donde se necesita leer los datos de las órdenes pendientes de cada cliente muchas veces al día (y para muchos clientes diferentes).

Usted se preguntará entonces, ¿Cómo se implementa en la práctica una lógica que permita que la consulta se pueda resolver más rápidamente?

Justamente de eso vamos a hablar ahora. Veamos para ese ejemplo puntual:

Lo primero que debemos hacer para organizar físicamente una base de datos, de manera de mejorar su performance, es definir todas las consultas habituales a las que vamos a enfocar nuestro diseño. Para nuestro ejemplo solamente elegimos la consulta que utiliza las tablas orders, [order details] y customers, pero podriamos haber elegido otra consulta enfocada en los productos pendientes o en los productos y su stock, en cuyo caso el diseño resultante sería diferente.

Como mencionamos anteriormente, la tabla orders tiene muchos datos, en consecuencia sería clave poder hacer una lectura más rápida de la misma. Igualmente debemos considerar [order details] ya que tiene aun más datos que orders (tiene n productos por cada orden). La forma más lógica que encontramos para mejorar la performance de la lectura es separando los datos en diferentes discos físicos. Si se utiliza un storage más avanzado tal el caso de los storage distribuídos en una red de discos, aplica el mismo concepto.

Lo mejor que podemos hacer para definir la distribución de los datos de la tabla es elegir una lógica que nos permita realizar dicha separación, por ejemplo utilizando la fecha de la orden (orderDate), aunque podríamos haber utilizado otro dato, como el numero de la orden (orderID), etc.

Teniendo en cuenta el orderDate, podemos definir la siguiente lógica, suponiendo que la tabla tiene datos desde 1995 hasta la fecha actual:

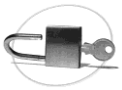
Vamos a repartir los datos en períodos de 5 años a saber: 1990 a 1995 en un disco, 1995 a 2000 en otro, 2000 a 2005 en un tercer disco, 2005 a 2010 en un cuarto disco y asi sucesivamente.

ACTIVIDAD 1

A continuación le pedimos que realice la Actividad 1 que le proponemos en el foro del mismo nombre en el campus. Podrá ver la solución de este ejercicio en los videos titulados “Actividad 1 - archivos - Parte 1” y “Actividad 1 - archivos - Parte 2” en el campus.

Se deberia realizar lo mismo con los datos de la tabla [order details] teniendo especial cuidado de no colocarlos en los mismos discos que los datos de orders. Si lo hiciéramos tambien habría que esperar a que termine una lectura para comenzar la otra. Esto se denomina contención en disco.

Entonces, una vez realizado este trabajo con los archivos involucrados en la consulta, será posible medir el impacto de la mejora en el tiempo de resolución de la consulta.
Si aplicamos esta lógica en el diseño de las bases de datos podremos evitar problemas posteriores que serían bastante complejos de afrontar y más aún resolver posteriormente cuando tengamos nuestro sistema en producción.



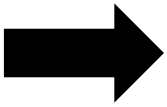
ABORDEMOS EL LOGRO DEL TERCER OBJETIVO DE LA UNIDAD:

- ✓ Diferentes técnicas de optimización del almacenamiento físico.

3. Diseños orientados a la performance

Además de utilizar archivos en diferentes discos para optimizar la performance de las lecturas, debemos tambien tener en cuenta otras actividades que debe realizar la base de datos en forma habitual.
Una de estas, que es muy importante que la base de datos la pueda realizar en todo momento, es la escritura en el log de transacciones. Todas las bases de datos deben realizar la escritura en el log de transacciones, su begin y su commit, o el checkpoint.
Es por eso que al mismo tiempo que la base de datos lee archivos de datos para una transacción, debe poder excribir en el log de transacciones para commitear otra transacción que se está ejecutando simultáneamente.
Queda claro entonces que, tal como era tan importante que los datos se encuentren distribuídos en diferentes discos para permitir procesamiento en paralelo de los datos, es mucho más importante que el log de transacciones este disponible para escrituras el 100% del tiempo. Entonces, es necesario que el log de transacciones esté alojado en un disco separado de los discos donde se realizan las demás actividades de la base de datos (lecturas y escrituras).
Además de los aspectos importantes que ya hemos mencionado, existen otras técnicas avanzadas que sirven para mejorar la performance del acceso a discos.
Una de estas técnicas se denomina ***particionado de tablas:***
A través del particionado podemos decidir separar los datos de una tabla muy consultada para dejar solamente los datos que se desean ver mas a menudo y el resto colocarlos en una tabla separada, utilizando la misma PK para ambas.
A esta operación se la denomina ***particionado vertical***, a continuación ilustramos un ejemplo de la misma con una tabla de empleados:

IDEmpleado
Nombre
Apellido
Dirección
TE Interno
TE part
Sueldo
IDDepto
ID OSoc



IDEmpleado
Nombre
Apellido

IDEmpleado
Dirección
TE Interno
TE Part
Sueldo
IDDepto
ID OSoc

Esta técnica logra acelerar el tiempo de acceso a los datos ya que la tabla, al tener menos columnas es más rápida de acceder y de leer, ocupa menos espacio tanto en disco como en memoria, etc.
También existe la técnica de ***particionado horizontal***, que de manera similar lo que pretende es partir la tabla en diferentes porciones aplicando una función y ubicando las diferentes partes en distintos archivos. Luego, cuando se realizan consultas sobre la tabla, se utiliza la función de partición para ubicar el (o los) archivo/s correspondiente a los datos que se desean acceder, por lo que siempre se va a acceder a una menor cantidad de datos, solo los afectados por la consulta.
De esta manera la lectura de las tablas particionadas horizontalmente accederá siempre a una menor cantidad de datos que la consulta original.
Los motores de base de datos traen herramientas especificas que ayudan a los DBAs a realizar el particionamiento de tablas con mucha facilidad y generalmente utilizando un asistente gráfico bastante simple. Sin embargo es una tarea que debe ser pensada con mucho criterio, sobre todo la elección de los campos que se utilizarán en la creación de la función de partición.

Pueden ver un ejemplo con detalle de esta técnica, utilizando un motor SQL Server en el siguiente link:
https://www.sqlshack.com/es/particionamiento-de-tablas-de-bases-de-datos-en-sql-server/

ACTIVIDAD 2-FORO

A continuación le pedimos que replique los pasos indicados para realizar la partición de la tabla orders utilizando la fecha orderDate. Comente su experiencia y los resultados obtenidos con sus compañeros en el foro abierto a tal efecto. Su tutor estará pendiente de los comentarios e irá corrigiendo si fuera necesario.

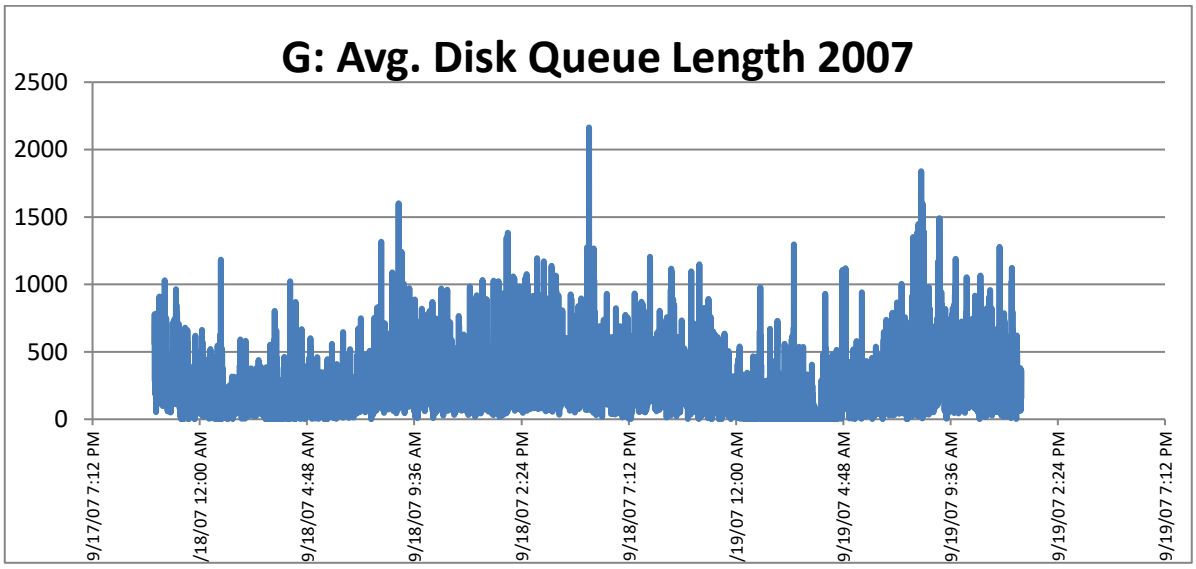
4. Ejemplos típicos.

A continuación vamos a repasar los cambios realizados sobre una base de datos real, que habia sido diseñada originalmente sin tener en cuenta los parámetros estudiados en los apartados anteriores, y que luego, al evidenciar problemas de performance en las consultas y en general, sus diseñadores decidieron aplicar estos principios de diseño. Veremos el impacto de los cambios realizados en las métricas de performance realizadas durante un estudio comparativo. Los cambios fueron realizados mientras la base de datos estaba en producción, utilizando técnicas avanzadas de replicación de datos.

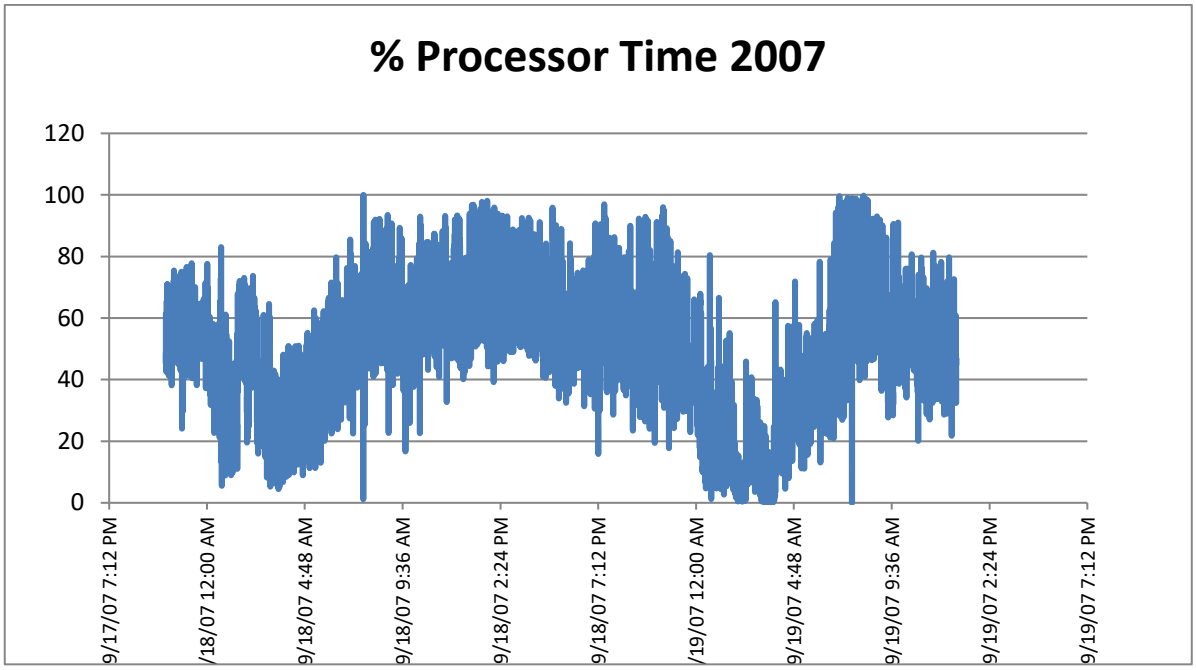
Veremos a traves de las métricas cómo impactaron los cambios realizados en ella a la vez que el crecimiento de la base de datos.

ETAPA INICIAL – Año 2007:

La aplicación inicia su actividad con una base de datos cuyo tamaño era de aproximadamente 50 GB. Los archivos de la base de datos estaban todos juntos en el mismo disco pero el log de transacciones se encontraba ya separado de ellos.
El disco G: era el disco de los archivos de datos.
En el gráfico siguiente se puede ver la longitud de la cola del disco durante un día. Esta medida indica cuanto debe esperar cada proceso hasta que es atendido por el disco. Se considera que valores mayores a 20 son indeseables.



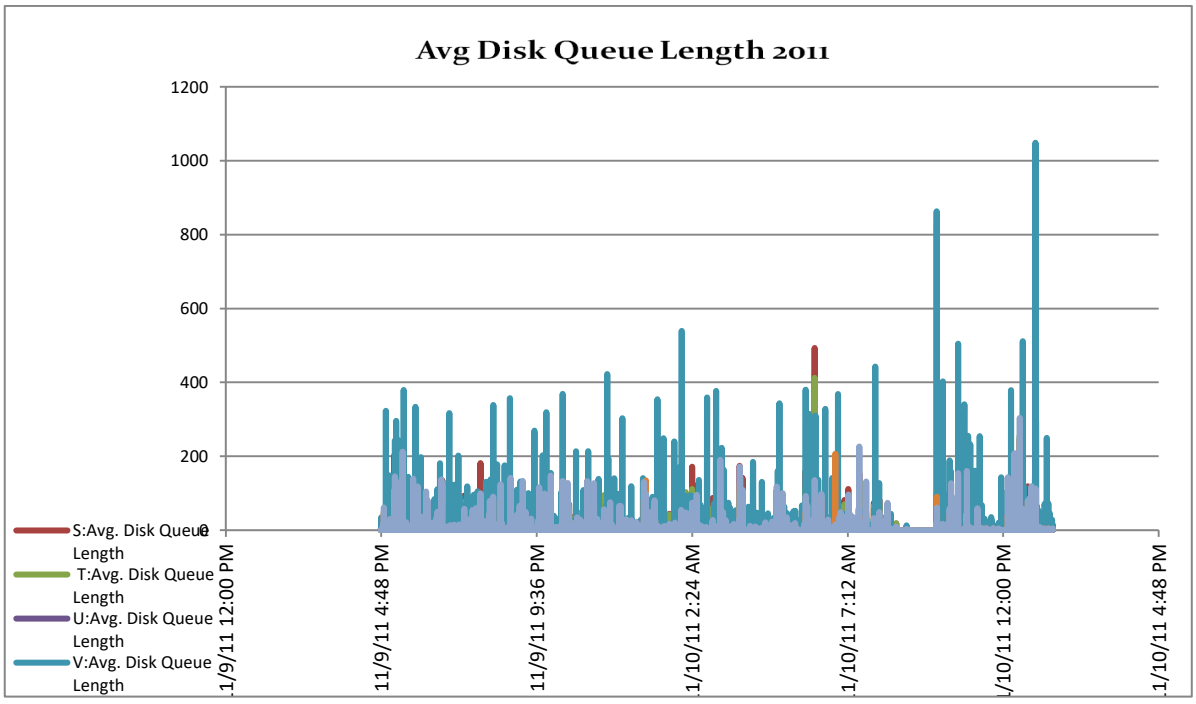
También se puede ver en el siguiente gráfico el uso de CPU para el mismo período de tiempo:



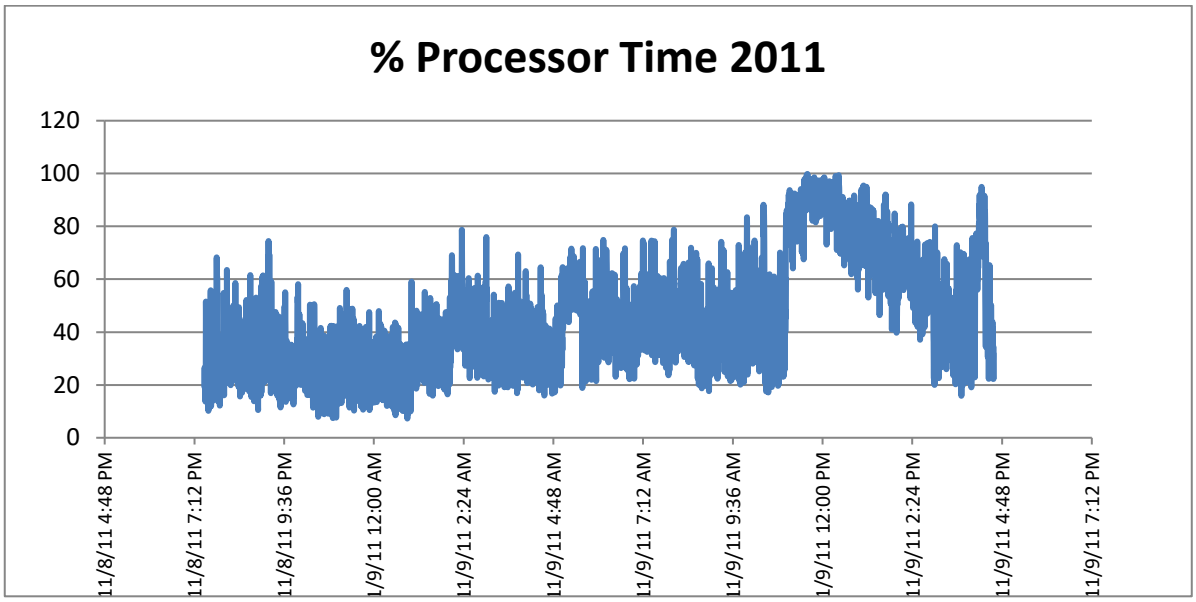
Se observa que si bien la CPU parece estar ocupada parte del tiempo también esta "esperando" por el disco para comenzar a procesar los datos recibidos.

ETAPA FINAL – Año 2011:

Luego de realizar la modificación en la distribución de los archivos, se volvieron a tomar las mismas métricas y estos fueron los resultados obtenidos:



Los datos se distribuyeron en varios discos: S, T, U y V de manera que pudieran beneficiar a ciertas consultas más importantes para la operatoria de la aplicación.
Se puede ver cómo a pesar de ser alta la cola de disco en algunos momentos, el promedio es bastante bajo en todos los discos. Para los usuarios el cambio significó que no tuvieran que esperar por la aplicación y que pudiera trabajar mayor cantidad de usuarios simultáneamente.
En cuanto al uso de CPU vemos en el gráfico siguiente que la CPU ya no está esperando sino que está trabajando todo el tiempo y que el promedio de utilización es menor que en 2007.



Un dato muy importante es que en 2011 el tamaño de la base de datos era de 700GB, es decir que estamos viendo mediciones en las cuales el comportamiento de la base de datos de 700 GB es mucho mejor que el de la misma base de datos cuando tenía 50 GB.

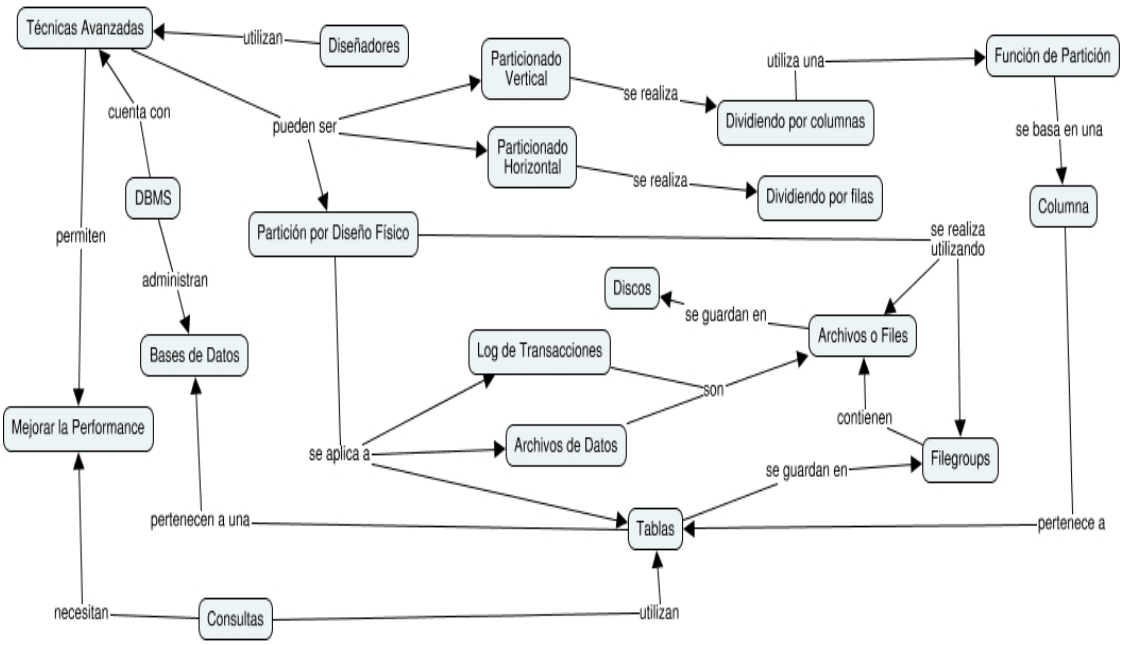
ACTIVIDAD 3- FORO

Comente sus impresiones al respecto de los análisis realizados con esta base de datos con sus compañeros en el foro abierta para esta actividad. Puede aprovechar el foro para preguntar a su tutor más sobre las técnicas aplicadas para realizar este trabajo de migración de datos.

SÍNTESIS DE LA UNIDAD

Hagamos un resumen de todo lo visto en esta Unidad.

Las TECNICAS AVANZADAS de los DBMS permiten asistir a los DISEÑADORES para MEJORAR LA PERFORMANCE de las CONSULTAS realizadas sobre las BASES DE DATOS.
Las CONSULTAS se pueden OPTIMIZAR mediante técnicas de PARTICIONADO que puede ser HORIZONTAL o VERTICAL.
En el PARTICIONADO VERTICAL se DIVIDE la TABLA separando COLUMNAS.
En el PARTICIONADO HORIZONTAL se DIVIDE la TABLA separando FILAS utilizando una FUNCION DE PARTICION basada en una COLUMNA.
Tambien vimos que se puede realizar la PARTICION de una TABLA utilizando el DISEÑO FISICO de la TABLA a través del uso de FILES y FILEGROUPS.
Tambien es importante recordar que el LOG DE TRANSACCIONES debe estar siempre SEPARADO FISICAMENTE de los ARCHIVOS DE DATOS para evitar la CONTENCION EN EL DISCO que provoca la mayoria de los PROBLEMAS DE PERFORMANCE de las BASES DE DATOS.



AUTO-EVALUACIÓN

Aquí le ofrecemos la posibilidad de reflexionar sobre su aprendizaje y sobre la forma como está organizando y llevando a cabo su trabajo. Es una herramienta muy importante. Úsela y si tiene comentarios para compartir, póngalos en el Foro de Interacción con los Tutores o si no son consultas y sólo quiere compartir, en el Bar.

Aspecto a evaluar	MB	B	R	M
1. Mi distribución del tiempo de estudio y trabajo.				
2. Mi entrenamiento en técnicas de estudio en esta unidad. (¿Lo hice, aprendí, usé lo que aprendí?)				
3. Nuevos aprendizajes.				
4. Mi participación en los foros.				
5. Mi participación en el Glosario.				
6. Mi manejo del campus y medios técnicos.				
7. Mi contacto con los compañeros. (¿Intercambio ideas, socializo en el bar?)				
8. Mi contacto con los tutores (¿Pregunto, comento, pido aclaraciones?)				

Considere: ¿Qué debe mejorar?