ANALISIS ALAT BANTU *TUNING* FISIKAL BASIS DATA PADA SQL SERVER 2008

Marlene Martani; Hanny Juwitasary; Arya Nata Gani Putra

Information Systems Department, School of Information Systems, Binus University
Jl. K.H. Syahdan No. 9, Palmerah, Jakarta Barat 11480
marlene@binus.edu; hjuwitasary@binus.edu; anputra@binus.edu

ABSTRACT

Nowadays every company has been faced with a business competition that requires the company to survive and be superior to its competitors. One strategy used by many companies is to use information technology to run their business processes. The use of information technology would require a storage which commonly referred to as a database to store and process data into useful information for the company. However, it was found that the greater the amount of data in the database, then the speed of the resulting process will decrease because the time needed to access the data will be much longer. The long process of data can cause a decrease in the company's performance and the length of time needed to make decisions so that this can be a challenge to achieve the company's competitive advantage. In this study performed an analysis of technique to improve the performance of the database system used by the company to perform tuning on SQL Server 2008 database physically. The purpose of this study is to improve the performance of the database used by speeding up the time it takes when doing query processing. The research methodology used was the method of analysis such as literature studies, analysis of the process and the workings of tuning tools that already exist in SQL Server 2008, and evaluation of applications that have been created, and also tuning methods that include query optimization and create index. The results obtained from this study is an evaluation of the physical application tuning tools that can integrate database functionality of other tuning tools such as SQL Profiler and Database Tuning Advisor.

Keywords: *information technology, database, tuning, SQL Server 2008*

ABSTRAK

Saat ini setiap perusahaan telah dihadapkan pada suatu kompetisi bisnis yang menuntut agar perusahaan dapat bertahan dan menjadi lebih unggul dari pesaing-pesaingnya. Salah satu strategi yang digunakan oleh banyak perusahaan adalah dengan menggunakan teknologi informasi dalam menjalankan proses bisnisnya. Penggunaan terhadap teknologi informasi tentu memerlukan suatu media penyimpanan yang biasa disebut dengan basis data untuk menyimpan dan mengolah data menjadi informasi yang berguna bagi perusahaan. Namun, ditemukan bahwa semakin besar jumlah data dalam basis data tersebut, maka kecepatan proses yang dihasilkan akan menurun karena waktu yang diperlukan untuk mengakses data akan menjadi lebih lama. Proses data yang lama dapat menyebabkan penurunan kinerja perusahaan dan lamanya waktu yang dibutuhkan dalam mengambil keputusan sehingga hal tersebut dapat menjadi suatu tantangan dalam mencapai keunggulan kompetitif perusahaan. Pada penelitian ini dilakukan suatu analisis terhadapteknik untuk meningkatkan kinerja dari sistem basis data yang digunakan oleh perusahaan dengan melakukan tuning pada basis data SQL Server 2008 secara fisikal. Tujuan dilakukannya penelitian ini adalah untuk meningkatkan performa dari basis data yang digunakan dengan cara mempercepat waktu yang dibutuhkan ketika melakukan query processing. Metodologi penelitian yang digunakan adalah metode analisis seperti studi literatur, analisis proses dan cara kerja alat bantu tuning yang telah ada dalam SQL Server 2008, dan evaluasi aplikasi yang telah dibuat, serta metode tuning yang meliputi optimasi query dan membuat index. Hasil yang diperoleh dari penelitian ini adalah evaluasi aplikasi alat bantu tuning fisikal basis data yang dapat mengintegrasikan fungsi dari alat bantu tuning lainnya seperti SQL Profiler dan Database Tuning Advisor.

Kata kunci: *teknologi informasi, basis data, tuning, SQL Server 2008*

PENDAHULUAN

Dalam era yang berkembang saat ini, peran teknologi informasi sangatlah penting bagi suatu perusahaan untuk memberi dukungan strategis dalam menjalankan proses bisnis perusahaan. Masing-masing perusahaan bersaing satu dengan yang lainnya menggunakan teknologi tercanggih untuk dapat bertahan dalam dunia bisnis dan mampu mencapai keunggulan kompetitif. Penggunaan teknologi informasi ini tentu memerlukan suatu media penyimpanan yang biasa disebut dengan basis data untuk menyimpan dan mengolah data menjadi informasi yang berguna bagi perusahaan. Semakin banyak data yang diperlukan oleh perusahaan tentu akan memerlukan basis data dalam ukuran yang besar juga sehingga mampu menyimpan dan mengolah data secara cepat dan tepat. Namun, ditemukan bahwa semakin besar jumlah data dalam basis data tersebut, maka kecepatan proses yang dihasilkan akan menurun karena waktu yang diperlukan untuk mengakses data dalam basis data akan menjadi lebih lama. Proses data yang lama dapat menyebabkan penurunan kinerja perusahaan dan lamanya waktu yang dibutuhkan oleh para pimpinan perusahaan dalam mengambil keputusan. Pada akhirnya, hal yang demikian menjadi suatu tantangan bagi perusahaan dalam usaha mencapai keunggulan kompetitif.

Melihat permasalahan yang ada, maka diperlukan sebuah teknik yang dapat meningkatkan kinerja basis data, yang mana untuk selanjutnya teknik ini disebut dengan *tuning* basis data. Berdasarkan penelitian yang telah dilakukan sebelumnya, *tuning* basis data dikembangkan dengan berbagai cara, antara lain melalui perangkat keras, perangkat lunak, DBMS dan perancangan fisik basis data. *Tuning* melalui perangkat keras dilakukan dengan mengganti atau mengubah *processor* pada komputer menjadi versi yang lebih cepat, sehingga kerja komputer dalam mengolah data lebih optimal. *Tuning* melalui perangkat lunak dilakukan dengan meningkatkan versi dari *Operating System* (OS) yang digunakan, misalnya dengan meningkatkan *service pack* dari sistem operasi Windows ke versi yang lebih baru. Cara berikutnya yaitu *tuning* pada DBMS, dimana biasanya *tuning* ini sudah tersedia sebagai salah satu fitur dalam DBMS, yaitu dengan *query* menggunakan *index* dan *statistic* pada basis data sehingga data dapat diakses dengan lebih cepat (Karthik, Reddy, dan Vanan, 2012). Adapun cara lain yang dapat digunakan adalah *tuning* perancangan fisik basis data, yaitu dengan menganalisis serta mengubah susunan *file* fisikal yang memiliki tingkat pertumbuhan sangat tinggi untuk disimpan dalam *disk* yang berbeda. Pada penelitian ini, *tuning* basis data yang diteliti adalah *tuning* pada basis data SQL Server 2008 secara fisikal.

Adapun tujuan dari penelitian ini adalah untuk meningkatkan performa dari basis data yang digunakan dengan cara mempercepat waktu yang dibutuhkan pada saat melakukan *query processing*. Dengan demikian, penggunaan *tuning* basis data ini diharapkan dapat berguna bagi perusahaan-perusahaan dalam mengimplementasi dan meningkatkan kinerja dan kecepatan dalam mengakses, mengolah serta memasukkan data, khususnya bagi perusahaan yang memiliki basis data dengan jumlah data yang sangat besar dan pada akhirnya dapat menjadi sebuah jawaban bagi perusahaan dalam usahanya mencapai keunggulan kompetitif.

METODE

Metodologi penelitian yang digunakan adalah pertama, Metode Analisis yang terdiri dari (a) Studi literatur; Melakukan studi literatur yang berasal dari buku maupun jurnal yang berkaitan dengan *tuning* basis data untuk menjadi bahan pembelajaran serta referensi bagi penelitian ini. (b) Analisis proses dan cara kerja alat bantu *tuning* yang terdapat dalam SQL Server 2008; melakukan analisis dan mempelajari bagaimana kerja/proses alat bantu *tuning* basis data yang telah ada pada SQL Server 2008, seperti SQL Server Profiler dalam memberikan panduan kepada *Database Administrator* (DBA) untuk mengoptimalkan kinerja basis data yang digunakan. Analisis proses ini memberikan

referensi/acuan tentang cara kerja alat bantu *tuning* basis data yang dibuat. (c) Evaluasi aplikasi alat bantu *tuning* fisikal dari basis data yang berkaitan dengan rancangan *file* dan tabel secara fisikal; membahas perancangan aplikasi alat bantu *tuning* basis data yang mencakup analisis terhadap beberapa *query* untuk diberikan rekomendasi dalam melakukan relokasi tabel ke dalam *file system* baru.

Kedua, Metode *Tuning*, Menurut Burleson (2010) metode *tuning* yang dapat digunakan adalah: (a) Optimasi *query*; melakukan perbaikan terhadap *syntax* yang digunakan agar eksekusi *query* menjadi lebih cepat dan menggunakan memori yang lebih sedikit. (b) Membuat *index;* membuat *index* sesuai kebutuhan untuk mempermudah eksekusi *query* sehingga mengurangi waktu respon dan juga menghemat penggunaan memori.

Tinjauan Pustaka

Basis Data

Menurut Whitten (2004) basis data adalah kumpulan *file* yang saling terkait. Connolly dan Begg (2010) menjelaskan bahwa basis data adalah kumpulan data yang terhubung secara logis yang digunakan bersama-sama dan deskripsi dari data tersebut dirancang untuk memenuhi kebutuhan informasi sebuah organisasi. Jadi, basis data adalah kumpulan data atau *file* yang saling terkait. Dengan basis data, data mudah disimpan, diperoleh kembali, dan dimanipulasi untuk dapat memenuhi kebutuhan informasi dari suatu organisasi.

Database Management System (DBMS)

Menurut Whitten (2004) *Database Management System* (DBMS) adalah perangkat lunak khusus yang digunakan untuk membuat, mengontrol, dan mengelola sebuah basis data. Connolly dan Begg (2010) menerangkan bahwa *Database Management System* (DBMS) adalah sistem perangkat lunak yang memungkinkan *user* untuk mendefinisikan, membuat, memelihara, dan mengontrol akses pada basis data. Jadi, *Database Management System* (DBMS) adalah suatu sistem perangkat lunak yang berguna untuk membuat, mengontrol, dan mengelola sebuah basis data.

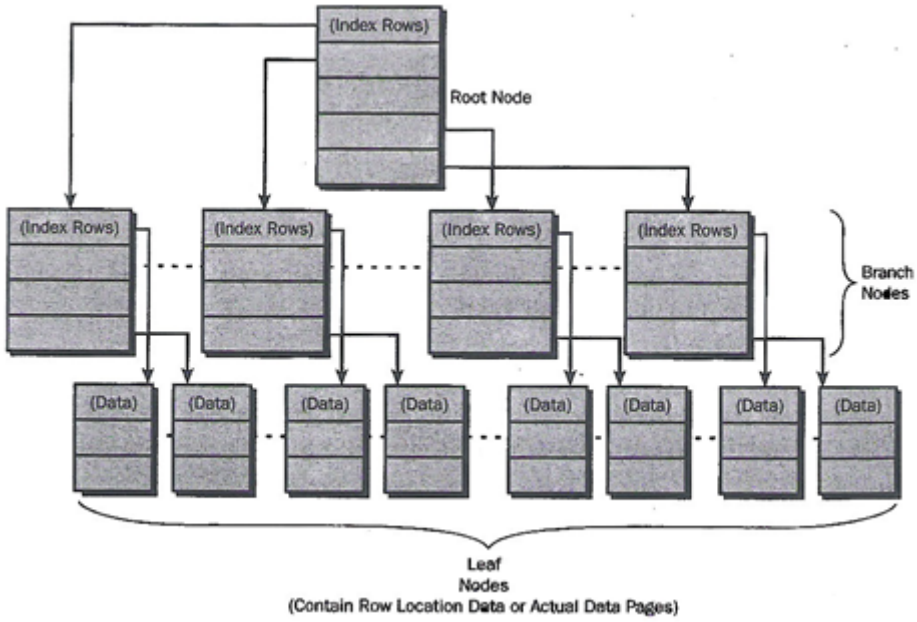
Statistic

Leiter (2009) menjelaskan bahwa *statistic* digunakan oleh SQL Server untuk menemukan cara yang paling efisien untuk mengambil data dari tabel basis data dengan menyimpan informasi tentang data yang disimpan dalam sebuah kolom. Menurut Petković(2008)**,** pertanyaan yang sering muncul ketika *Database Engine* mengeksekusi *query* adalah bagaimana data yang diperlukan untuk di-*query* dapat diakses dan diproses dengan cara yang paling efisien. Komponen dalam sebuah basis data yang bertanggung jawab untuk proses ini disebut *query optimizer*. Tugas *query optimizer* adalah mempertimbangkan berbagai kemungkinan strategi eksekusi untuk *query* data dalam kaitannya dengan *query* yang diberikan dan untuk memilih strategi yang paling efisien. *Query optimizer* membuat keputusan dengan menggunakan pertimbangan seperti seberapa besar tabel yang terlibat dalam *query*, apakah terdapat *index*, dan operator Boolean (AND, OR, NOT) apa yang digunakan dalam klausa WHERE. Secara umum, pertimbangan ini disebut *statistic*.

Index

Menurut Whalen (2001) *index* adalah alat bantu struktur data yang digunakan oleh DBMS dalam pengaksesan data. *Index* akan dibuat di dalam tabel-tabel dalam basis data tersebut. Satu tabel dapat memiliki satu atau lebih *index* yang berguna untuk menunjuk data yang ada dalam tabel tersebut. Berdasarkan tipenya, *index* data bisa bersama dengan *table data* atau secara terpisah.

Tanpa *index*, semua data diperoleh melalui *table scans* yang berarti bahwa semua data yang ada dalam tabel akan dibaca secara sekuensial dan dibandingkan dengan data yang diminta. Hal ini harus dihindari karena *table scans* akan menghasilkan banyak sekali I/O (kecuali pemilihan data dengan persentase tinggi dari tabel). Tabel yang sangat besar mengkonsumsi banyak sekali sumber daya sistem ketika proses *table scan* dilakukan. Dengan menggunakan *index* yang tepat, jumlah operasi I/O yang dibutuhkan untuk mencari data yang tepat bisa dikurangi secara signifikan.



Gambar 1 Struktur *Index*

Secara umum *index* terbagi ke dalam beberapa tipe, diantaranya: (a) Clustered Index; Whalen (2001) menyatakan bahwa *clustered index* merupakan urutan data yang harus disimpan di tabel secara fisikal. Tabel data diurutkan dan disimpan berdasarkan *key column* atau kolom yang dispesifikasikan untuk *clustered index*. Tipe *index* ini dapat disamakan dengan sebuah kamus, di mana informasi disimpan dengan urutan *alphabetic* dan menyediakan informasi untuk mencari data dengan cepat. *Index* ini menyatu dengan datanya, jadi seperti kamus yang menyediakan arti dari sebuah kata tepat di sebelahnya. (b) Nonclustered Index; Whalen (2001) menyatakan bahwa *index* ini berbeda dengan *clustered index* yang menyatu dengan data. *Nonclustered index* dapat dianalogikan seperti buku yang mempunyai *index* di halaman akhirnya. *Index* ini memberi indikasi tentang halaman mana yang akan dituju, tetapi datanya tidak terdapat dalam *index* tersebut. (c) Indexed Views; Whalen (2001) menyatakan bahwa *indexed views* merupakan *index* yang membahayakan, karena dapat meningkatkan jumlah I/O dalam sistem secara signifikan, tetapi *index* ini menawarkan solusi kepada banyak masalah *query* pada umumnya.

Satu kekurangan yang terdapat dalam *indexed views* ialah keharusan untuk membuat *clustered index* pada PRIMARY KEY yang didefinisikan dalam *view*. Ini artinya, semua *view* yang diberikan *index* harus mempunyai kolom atau sekumpulan kolom yang selalu unik. Banyak *view* tidak di desain untuk keperluan ini, jadi untuk memenuhi kebutuhan ini diperlukan pemikiran yang kreatif.

SQL Server 2008

Leiter (2009) menyatakan bahwa SQL Server 2008 sangat dikenal sebagai *Relational Database Management System* (RDBMS). Namun tidak hanya itu saja, tetapi juga dapat lebih akurat digambarkan sebagai *data enterprise platform* yang dibangun di atas banyak fitur yang pertama kali didirikan di SQL Server 2005. Sementara itu, SQL Server 2008 juga memperluas penawaran untuk menyertakan beberapa perbaikan dan penambahan. Karena dikenal dengan peran sebagai RDBMS tradisional, SQL Server 2008 juga menyediakan kemampuan *reporting*, analisis data yang kuat dan *datamining*. SQL Server 2008 tentunya juga mempunyai fitur-fitur yang mendukung aplikasi data yang berbeda, *data-driven event notification* dan masih banyak lagi.

SQL Server 2008 *Database Storage*

Menurut Leiter (2009) semua sistem dan pengguna basis data (termasuk sumber daya basis data) disimpan dalam *file*. Selalu ada minimal dua *file* penyimpanan yaitu satu *file* data dan satu *file* transaksi. Ekstensi *default* untuk *file* data adalah .MDF, dan *default* untuk *filelog* transaksi adalah .LDF.

Data File* dan *Filegroup

Leiter (2009) menjelaskan bahwa ketika *account* sebuah *user* basis data dibuat harus mengandung setidaknya satu *file* data. *File* data yang pertama dikenal sebagai *file* data primer. *File* data primer adalah anggota primer *default filegroup*. Setiap basis data memiliki satu *filegroup* primer ketika dibuat, yang setidaknya terdiri dari *file* data primer. *File* data tambahan juga dapat ditambahkan ke *filegroup* primer. *Filegroup* yang lebih banyak dapat didefinisikan pada pembuatan awal basis data, atau ditambahkan setelah basis data dibuat. *File* data dapat dikelompokkan secara logis untuk meningkatkan performa dan memungkinkan untuk pemeliharaan yang lebih fleksibel.

Log File

Leiter (2009) menjelaskan bahwa setelah membuat suatu basis data, satu transaksi *log* harus ditetapkan. Transaksi *log* digunakan untuk mencatat semua modifikasi terhadap basis data untuk menjamin konsistensi dan kemampuan untuk pulih dari suatu transaksi.

Meskipun menguntungkan untuk membuat beberapa *file* data dan beberapa *filegroup*, adalah jarang diperlukan untuk membuat lebih dari satu *file log*. Hal ini disebabkan bagaimana SQL Server mengakses data. *File* data dapat diakses secara paralel, memungkinkan SQL Server untuk membaca dan menulis ke beberapa *file* dan *filegroup* secara bersamaan. *File log* dijadikan serial untuk menjaga konsistensi transaksional. Setiap transaksi dicatat dalam *log* secara serial dan dalam urutan tersebut transaksi dieksekusi. *File log* yang kedua tidak akan diakses sampai *file log* pertama benar-benar penuh.

Performance Tuning* dan *Optimization

Petković (2008) menjelaskan bahwa kinerja dari sebuah *Database Engine* diukur berdasarkan dua kriteria: (a) Waktu Respon; merupakan panjang suatu waktu pada saat *user* memasukkan sebuah perintah sampai waktu di mana sistem mengindikasikan perintah tersebut telah selesai dilakukan. Waktu respon mengukur performa dari transaksi individual atau program. (b) Hasil Keluaran; merupakan performa keseluruhan dari sistem dengan menghitung jumlah transaksi yang dapat ditangani oleh *Database Engine* dalam satuan waktu yang telah diberikan.

Menurut Whallen (2001) *Performance Tuning* adalah sebuah tindakan mengubah performa dari sebuah sistem dengan memodifikasi parameter dari sistem (*software tuning*) atau dengan merubah

konfigurasi sistem (*hardware tuning*). *Performancetuning* melibatkan suatu analisis detil dari konfigurasi perangkat keras, sistem operasi, dan konfigurasi dari *Relational Database Management System* (RDBMS) dan aplikasi yang mengakses komponen tersebut.

Tujuan utama dari *tuning* adalah untuk menghilangkan *bottlenecks*, atau keterbatasan performa dalam komponen. Komponen yang dimaksud disini dapat berupa perangkat keras atau perangkat lunak yang dapat mempengaruhi performa dari sistem ketika dikonfigurasi dan di-*tuning* secara benar. *Bottlenecks* dapat diartikan sebagai adanya ketidakseimbangan antara peningkatan jumlah data yang akan diproses dengan kapasitas jumlah data yang dapat diproses oleh suatu komponen dalam suatu waktu. Sebagai contoh ialah ketika jumlah data yang akan diproses meningkat sangat pesat, namun kapasitas memori untuk memproses data tersebut tidak mengalami peningkatan. Dengan demikian, maka jumlah data yang dapat diproses dalam suatu waktu oleh memori akan menjadi terbatas sehingga memerlukan waktu lebih banyak untuk memproses data. Hal ini tentu akan berakibat menurunnya performa dari basis data yang digunakan.

Sekarang ini, penerapan SQL *tuning* merupakan parameter penting yang digunakan dalam mengelola *query*. Beberapa alasan terhadap pentingnya SQL *tuning* dikemukakan oleh Karthik, Reddy, dan Vanan (2012): (a) Mengurangi waktu respon (*response time*) ketika melakukan proses SQL. (b) Menemukan cara yang lebih efisien dalam memproses data. (c) Meningkatkan waktu pencarian dengan menggunakan *index*. (d) Memungkinkan proses penggabungan data antara dua tabel atau lebih secara efisien

Meningkatkan performa dari sebuah sistem basis data memerlukan banyak pertimbangan, seperti di mana data harus disimpan dan bagaimana cara menyimpannya. Jika sistem basis data tidak bekerja secara optimal, maka sistem administrator harus melakukan pemeriksaan terhadap banyak faktor dan jika memungkinkan, melakukan *tuning software* terhadap sistem basis data tersebut.

HASIL DAN PEMBAHASAN

Menurut penelitian yang dilakukan oleh Sukheja dan Singh (2011), kebutuhan akan basis data meningkat secara drastis dalam lingkungan pekerjaan saat ini disebabkan oleh adanya desentralisasi dalam infrastruktur IT suatu perusahaan sebagai akibat dari *merger*, akuisisi, dan aplikasi khusus perusahaan. Dengan semakin besar jumlah data dalam basis data, maka kecepatan proses yang dihasilkan akan menurun karena waktu yang diperlukan untuk mengakses data dalam basis data akan menjadi lebih lama. Untuk dapat tetap menjaga bahkan meningkatkan performa basis data, maka diperlukan suatu teknik yang disebut dengan *tuning* basis data. Menurut penelitian yang dilakukan oleh Karthik, Reddy, dan Vanan (2012), dengan melakukan *tuning* maka dapat mengurangi waktu respon untuk pemrosesan SQL, menemukan cara yang lebih efisien untuk memproses beban kerja, meningkatkan waktu pencarian dengan menggunakan index, dan penggabungan data yang lebih efisien. Pada penelitian ini, *tuning* basis data yang diteliti adalah *tuning* pada basis data SQL Server 2008 secara fisikal, yaitu dengan menganalisis serta mengubah susunan *file* fisikal yang memiliki tingkat pertumbuhan sangat tinggi untuk disimpan dalam *disk* yang berbeda.

Tools (alat-alat bantu) yang tersedia pada SQL Server 2008

SQL Server Profiler

SQL Server Profiler merupakan alat bantu grafis yang memungkinkan sistem administrator memonitor dan merekam aktivitas basis data dan *server*, seperti *login*, *user*, dan aplikasi informasi. SQL Server Profiler juga dapat menampilkan informasi terkait kegiatan *server* secara *real time*, dan

fokus pada kegiatan tertentu dari *user*, jenis perintah, atau jenis laporan *Transact SQL*. Fitur yang paling berguna dari SQL Server Profiler adalah kemampuannya untuk menangkap aktivitas yang berkaitan dengan *query*. Aktivitas ini dapat digunakan sebagai masukan untuk Database Engine Tuning Advisor yang memungkinkan untuk memilih *indices* atau *indexedviews*untuk satu atau lebih *query* (Petković, 2008).

Database Engine Tuning Advisor (DTA)

Database Engine Tuning Advisor (DTA) merupakan bagian dari keseluruhan sistem dan memungkinkan untuk otomatisasi desain fisik suatu basis data. DTA berhubungan erat dengan SQL Server Profiler. Fitur spesifik SQL Server Profiler yang digunakan oleh DTA adalah kemampuannya untuk melihat dan merekam aktivitas yang dijalankan oleh *user* dan untuk menyediakan informasi kinerja, seperti penggunaan CPU dan kesesuaian dengan *I/O statistic* (Petković, 2008).

Analisis kinerja basis data berdasarkan alat bantu yang dimiliki oleh SQL Server

Analisis kinerja basis data berdasarkan alat bantu yang dimiliki oleh SQL Server dilakukan untuk mengetahui pengaruh kinerja basis data dengan tabel yang diletakkan dalam *file* fisik yang berbeda, khususnya saat melakukan *query processing*. Analisis dilakukan dengan menggunakan perangkat keras dan perangkat lunak yang sama.

Sebagai contoh, dilakukan analisis pada tabel Person dan Address pada basis data AdventureWorks2008 dengan menjalankan *query select* yang sama dan dilakukan sebanyak sepuluh kali percobaan. Tabel tersebut akan mendapat perlakuan yang terbagi menjadi dua bagian yaitu tabel sebelum diberi *index* (kondisi tabel sebelum dipindahkan ke dalam *file* fisik yang berbeda dengan tabel-tabel lain) dan tabel sesudah diberi *index* (kondisi tabel sesudah dipindahkan ke dalam *file* yang berbeda).

Tabel Person

Query (Query1) yang digunakan untuk menguji tabel yang belum dipindahkan ke dalam *file* fisik yang berbeda dengan tabel-tabel lain (dalam hal ini tabel PersonTest01) adalah:

```
select a.addressID, a.addressID, a.city, a.stateprovinceID, a.postalcode from Person.Address a join Person.BusinessEntityAddress b on a.AddressID = b.addressID join Person.BusinessEntity c on b.BusinessEntityID = c.BusinessEntityID join Person.PersonTest01 d on c.BusinessEntityID = d.BusinessEntityID
```

Sedangkan *query* (Query2) yang digunakan untuk menguji tabel yang telah dipindahkan kedalam *file* fisik yang berbeda (dalam hal ini tabel PersonTest02) adalah:

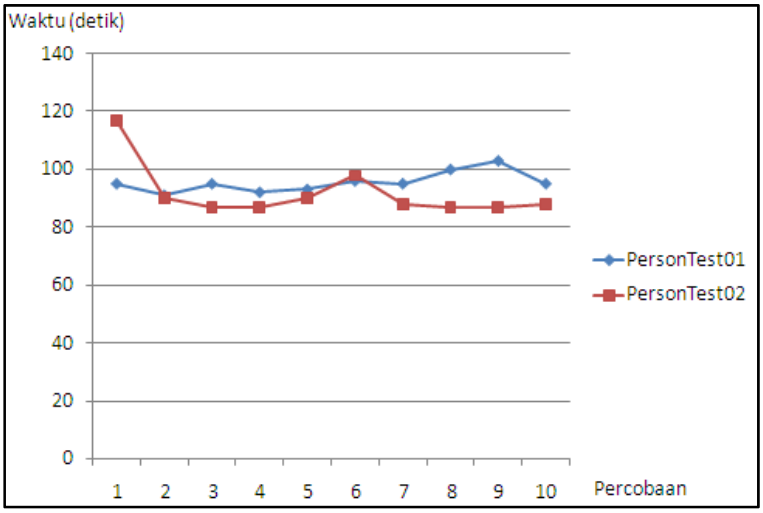
```
select a.addressID, a.addressID, a.city, a.stateprovinceID, a.postalcode from Person.Address a join Person.BusinessEntityAddress b on a.AddressID = b.addressID join Person.BusinessEntity c on b.BusinessEntityID = c.BusinessEntityID join Person.PersonTest02 d on c.BusinessEntityID = d.BusinessEntityID
```

Tabel Person sebelum diberi *index*

Setelah dilakukan pengujian terhadap tabel Person sebelum diberi *index*, hasil yang didapat adalah sebagai berikut:

Tabel 1 Tabel perbandingan antara tabel PersonTest01
dan PersonTest02 sebelum diberi *index*

Percobaan	Tabel PersonTest01		Tabel PersonTest02	
	Query	Waktu(Detik)	*Query*	Waktu(Detik)
1	Query1	95	Query2	117
2		91		90
3		95		87
4		92		87
5		93		90
6		96		98
7		95		88
8		100		87
9		103		87
10		95		88
Rata-rata		**95,5**		**91,9**



Grafik 1 Grafik perbandingan antara tabel PersonTest01
dan PersonTest02 sebelum diberi *index*

Berdasarkan data di atas, dapat dilihat bahwa percobaan 1 dan 6 pada tabel PersonTest02 memiliki waktu respon yang lebih lama daripada tabel PersonTest01. Sedangkan pada percobaan lainnya, tabel PersonTest02 memiliki waktu respon yang lebih cepat. Perbedaan ini dapat terjadi karena adanya proses lain yang sedang berjalan.

Dari data di atas dapat dilihat juga bahwa rata-rata waktu yang diperlukan untuk menjalankan *query* pada tabel PersonTest01 adalah 95.5 detik dan waktu yang diperlukan untuk menjalankan *query* pada tabel PersonTest02 adalah 91.9 detik (terjadi peningkatan sebesar 3.7%).

Dengan demikian terbukti bahwa kinerja dari tabel yang dipindahkan ke dalam *file* fisik yang berbeda (Tabel PersonTest02) lebih cepat dibandingkan dengan tabel yang berada pada *file* fisik yang sama dengan tabel-tabel yang lain (Tabel PersonTest01).

Tabel Person sesudah diberi index

Sebelum pengujian ini dilakukan, perlu dilakukan penambahan *index* pada kedua tabel apabila memungkinkan. Penambahan *index* dilakukan dengan bantuan SQL Server *tools* yaitu Database Engine Tuning Advisor. Caranya adalah dengan melakukan analisis terhadap *file example.sql* dimana isi dari *file* tersebut adalah sebagai berikut:

```
USE AdventureWorks2008;
GO
SELECT *
FROM Person.PersonTest01
ORDER BY FirstName ASC;
```

Rekomendasi *index* yang diberikan oleh Database Engine Tuning Advisor pada tabel PersonTest01 setelah *query* di atas dianalisis adalah sebagai berikut:

```
use [AdventureWorks2008]
go
CREATE NONCLUSTERED INDEX
[_dta_index_PersonTest01_9_214291823__K5_1_2_3_4_6_7_8_9_10_11_12_13] ON
[Person].[PersonTest01]
(
    [FirstName] ASC
)
INCLUDE ( [BusinessEntityID],
[PersonType],
[NameStyle],
[Title],
[MiddleName],
[LastName],
[Suffix],
[EmailPromotion],
[AdditionalContactInfo],
[Demographics],
[rowguid],
[ModifiedDate]) WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING =
OFF, ONLINE = OFF) ON [PRIMARY]
go
```

Query di atas dijalankan pada SQL Server Management Studio untuk membuat *index* pada tabel PersonTest01. Dari hasil yang diperoleh dapat dilihat bahwa *query* berhasil dijalankan pada SQL Server Management Studio dan membuat *non-clustered index* pada tabel PersonTest01.

Setelah *index* pada tabel PersonTest01 dibuat, maka berikutnya adalah membuat *index* pada tabel PersonTest02. *File* yang digunakan untuk dianalisis oleh Database Engine Tuning Advisor adalah *example.sql* yang berisi *query* sebagai berikut:

```
USE AdventureWorks2008;
GO
SELECT *
FROM Person.PersonTest02
ORDER BY FirstName ASC;
```

Berikut adalah *query* yang diberikan oleh Database Engine Tuning Advisor setelah menganalisis *query* di atas:

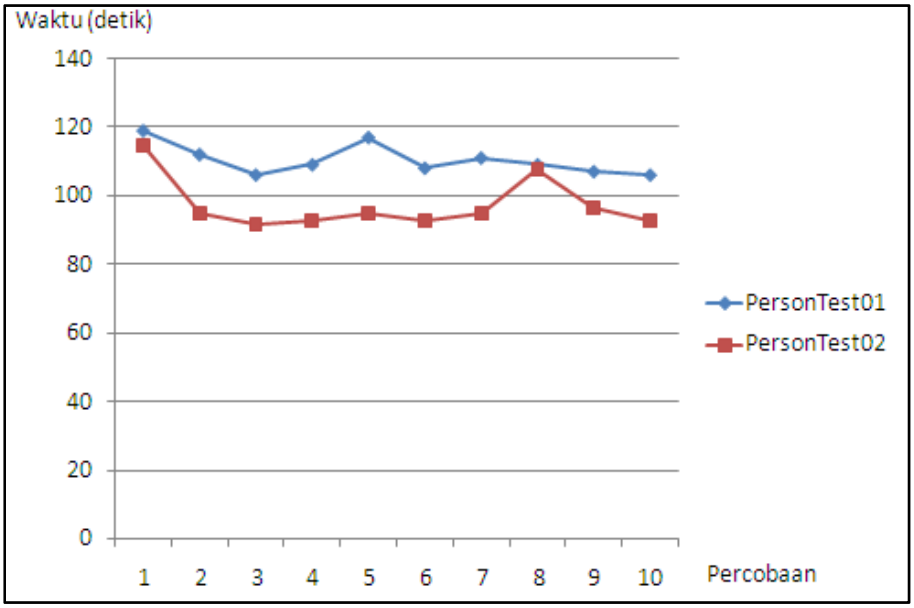
```
use [AdventureWorks2008]
go
CREATE NONCLUSTERED INDEX
[_dta_index_PersonTest02_9_550293020__K5_1_2_3_4_6_7_8_9_10_11_12_13] ON
[Person].[PersonTest02]
(
    [FirstName] ASC
)
INCLUDE ( [BusinessEntityID],
[PersonType],
[NameStyle],
[Title],
[MiddleName],
[LastName],
[Suffix],
[EmailPromotion],
[AdditionalContactInfo],
[Demographics],
[rowguid],
[ModifiedDate]) WITH (SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING =
OFF, ONLINE = OFF) ON [PRIMARY]
go
```

Query di atas dijalankan pada SQL Server Management Studio untuk membuat *index* pada tabel PersonTest02.

Dari hasil yang didapat, diperoleh bahwa *query* berhasil dijalankan pada SQL Server Management Studio dan membuat *non-clustered index* pada tabel PersonTest02. Setelah kedua tabel tersebut diberi *index*, maka dilakukan analisis kembali dengan menggunakan query1 dan query2 pada tabel PersonTest01 dan PersonTest02 yang sudah ditambahkan *index*.Hasil yang didapat adalah sebagai berikut:

Tabel 2 Tabel perbandingan antara tabel PersonTest01 dan PersonTest02 sesudah diberi *index*

Percobaan	**Tabel PersonTest01**		**Tabel PersonTest02**	
	Query	**Waktu(Detik)**	***Query***	**Waktu(Detik)**
1	Query1	119	Query2	115
2		112		95
3		106		92
4		109		93
5		117		95
6		108		93
7		111		95
8		109		108
9		107		97
10		106		93
Rata-rata		**110,4**		**97,6**



Grafik 2 Grafik perbandingan antara tabel PersonTest01
dan PersonTest02 sesudah diberi *index*

Berdasarkan data di atas, maka dapat dilihat bahwa semua percobaan pada tabel PersonTest02 dan rata-rata waktu menjalankan *query*-nya memiliki waktu respon yang lebih cepat daripada tabel PersonTest01. Selain itu, dapat dilihat juga bahwa rata-rata waktu yang diperlukan untuk menjalankan *query* pada tabel PersonTest01 adalah 110.4 detik dan waktu yang diperlukan untuk menjalankan *query* pada tabel PersonTest02 adalah 97.6 detik. Dengan demikian terjadi peningkatan sebesar 11.6%.

Hal ini membuktikan bahwa tabel yang telah diberi *index* dan dipindahkan ke dalam *filegroup* baru dapat lebih meningkatkan kinerja basis data.

Tabel Address

Query (Query1) yang digunakan untuk menguji tabel yang belum dipindahkan ke dalam *file* fisik yang berbeda dengan tabel-tabel lain (dalam hal ini tabel AddressTest01) adalah:

*Select * from Person.AddressTest01*

Sedangkan *query* (Query2) yang digunakan untuk menguji tabel yang telah dipindahkan kedalam *file* fisik yang berbeda (dalam hal ini tabel AddressTest02) adalah:

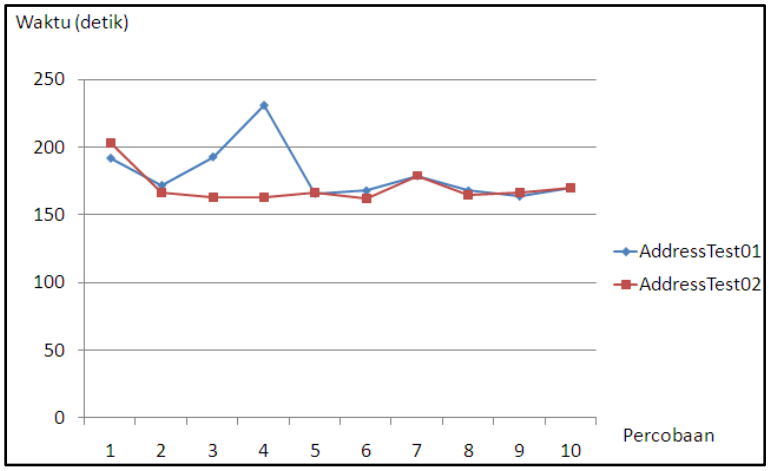
*Select * from Person.AddressTest02*

Tabel Address sebelum diberi *index*

Setelah dilakukan pengujian terhadap tabel Address sebelum diberi *index*, hasil yang didapat adalah sebagai berikut:

Tabel 3 Tabel perbandingan antara tabel AddressTest01 dan AddressTest02 sebelum diberi *index*

Percobaan	Tabel AddressTest01		Tabel AddressTest02	
	Query	Waktu(Detik)	*Query*	Waktu(Detik)
1	Query1	192	Query2	203
2		172		166
3		193		163
4		231		163
5		166		166
6		168		162
7		179		179
8		168		165
9		164		166
10		170		170
Rata-rata		**183,3**		**170,3**



Grafik 3 Grafik perbandingan antara tabel AddressTest01 dan AddressTest02 sebelum diberi *index*

Berdasarkan data di atas, dapat dilihat bahwa percobaan 1 dan 9 pada tabel AddressTest02 memiliki waktu respon yang lebih lama daripada tabel AddressTest01. Sedangkan pada percobaan lainnya, tabel AddressTest02 memiliki waktu respon yang lebih cepat atau sama dengan tabel AddressTest01. Perbedaan ini dapat terjadi karena adanya proses lain yang sedang berjalan.

Dari data di atas dapat dilihat juga bahwa rata-rata waktu yang diperlukan untuk menjalankan *query* pada tabel AddressTest01adalah 183.3 detik dan waktu yang diperlukan untuk menjalankan *query* pada tabel AddressTest02adalah 170.3 detik. Dengan demikian terjadi peningkatan sebesar 6.87%.

Dengan demikian terbukti bahwa kinerja dari tabel yang dipindahkan ke dalam *file* fisik yang berbeda (Tabel AddressTest02) lebih cepat dibandingkan dengan tabel yang berada pada file fisik yang sama dengan tabel-tabel yang lain (Tabel AddressTest01).

Tabel Address sesudah diberi *index*

Sebelum pengujian ini dilakukan, perlu dilakukan penambahan *index* pada kedua tabel tersebut. Untuk menambahkan *index* pada tabel AddressTest01, digunakan *fileexample.sql* yang berisi *query* berikut:

```
USE AdventureWorks2008;
GO
SELECT *
FROM Person.AddressTest01
ORDER BY AddressID ASC;
```

Setelah dianalisis, Database Engine Tuning Advisor tidak memberikan rekomendasi apapun. Ini berarti tabel AddressTest01 tidak perlu dibuatkan *non-clustered index*.

Berikutnya dilakukan percobaan menambah atau membuat *non clustered index* terhadap tabel AddressTest02. Percobaan dilakukan dengan menganalisis *fileexample.sql* yang berisi *query* berikut:

```
USE AdventureWorks2008;
GO
SELECT *
FROM Person.AddressTest02
ORDER BY AddressID ASC;
```

Setelah dianalisis, Database Engine Tuning Advisor tidak memberikan rekomendasi apapun. Ini berarti tabel AddressTest02 pun tidak perlu dibuatkan *non-clustered index*. Dengan tidak adanya penambahan *index* pada kedua tabel, berarti percobaan lebih lanjut tidak perlu dilakukan.

Masalah kinerja SQL Server 2008 yang belum atau tidak mempunyai alat bantu/*tools*

Dari analisis yang telah dilakukan, dapat dilihat bahwa perbaikan performa padaSQL Server dapat dilakukan dengan setidaknya dua cara yaitu dengan membuat sebuah *index* pada tabel dan dengan memindahkan tabel ke dalam *file system* yang berbeda. Dalam hal pembuatan *index*, SQL telah memiliki alat bantu seperti yang telah dijelaskan sebelumnya yaitu dengan menggunakan Database Engine Tuning Advisor. Ketika dijalankan, alat bantu ini akan memberikan sebuah rekomendasi yang dapat dijalankan oleh *user* dan secara otomatis membuatkan *index* pada tabel yang diinginkan.

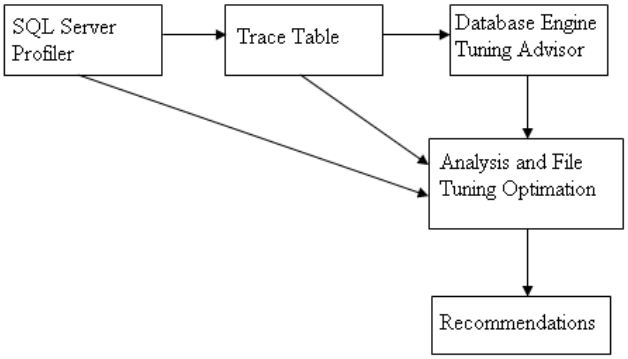
Permasalahan yang dihadapi adalah belum adanya alat bantu dari SQL Server yang mempunyai kemampuan dalam menganalisis kinerja *query* atas tabel antara tabel yang tergabung dengan tabel lain dalam *file system* yang sama dengan tabel yang dipisah ke dalam *file system* yang berbeda secara otomatis dan menghasilkan rekomendasi berdasarkan analisis yang telah dilakukan tersebut.

Usulan Pemecahan Masalah

Untuk menghadapi permasalahan yang ada, maka telah dibuatkan sebuah aplikasi yang dapat mengintegrasikan alat bantu SQLServer Profiler dan Tuning Advisor. Aplikasi yang telah dibuat ini mampu melakukan analisis untuk menentukan tabel-tabel apa saja yang sebaiknya dipindahkan ke dalam *file system* yang berbeda, membuat sebuah *file system* (*file* dan *filegroup)* yang baru, memindahkan tabel-tabel tersebut ke dalam *file system* yang baru dibuat, melakukan analisis kinerja basis data setelah tabel-tabel tersebut dipindahkan, menghasilkan rekomendasi berdasarkan hasil analisis yang telah dilakukan, dan mengeksekusi rekomendasi tersebut sehingga kinerja basis data bisa ditingkatkan.

Implementasi dan Evaluasi

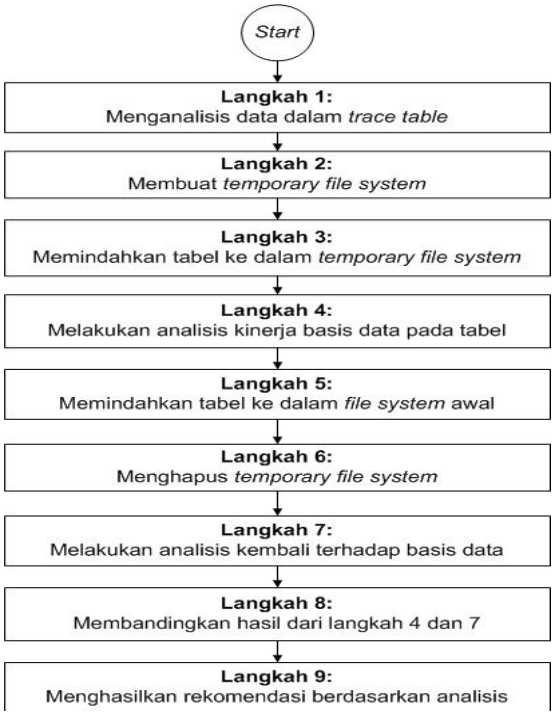
Aplikasi yang dapat mengintegrasikan alat bantu SQLServer Profiler dan Tuning Advisor dinamakan Analysis and File Tuning Optimation memiliki rancangan arsitektur sebagai berikut:



Gambar 2 Arsitektur aplikasi Analysis and File Tuning Optimation

Berdasarkan arsitektur aplikasi yang telah dibuat, maka dapat dilihat bahwa aplikasi mengintegrasikan fungsi dari SQL Server Profiler dan Database Tuning Advisor. Integrasi ini dapat dilakukan karena aplikasi menggunakan hasil keluaran dari SQL Server Profiler (berupa tabel). Dari hasil keluaran tersebut dapat dilakukan analisis dari tabel-tabel yang sudah diberi *index* maupun yang belum diberi *index* oleh Database Tuning Advisor sehingga menjadi sebuah tabel untuk dianalisis lebih lanjut. Melalui *trace table* yang ada, aplikasi tuning fisikal akan menganalisis setiap *query* yang terdapat di dalam *trace table* sehinga menghasilkan beberapa rekomendasi terhadap tabel yang perlu di-*tuning*.Secara garis besar, proses yang dijalankan oleh aplikasi terdiri dari dua proses utama, yaitu:

Proses analisis *query*

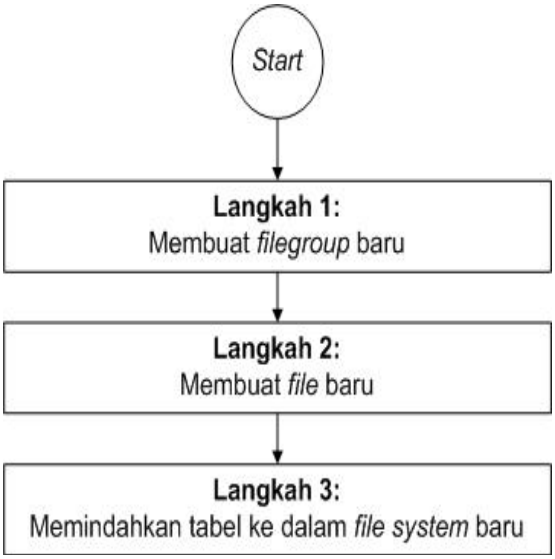


Gambar 3.*Flowchart* proses analisis

Berikut penjelasan gambar 3 di atas, langkah satu adalah melakukan analisis berdasarkan data yang terdapat dalam tabel, yang merupakan hasil keluaran SQL Server Profiler, yang sebelumnya telah diisi oleh *user* untuk menentukan tabel-tabel apa saja yang perlu dipindahkan ke dalam *file system* yang berbeda, guna meningkatkan kinerja sistem basis data. Langkah dua adalah membuat sebuah *temporaryfile system* yang baru (*file* dan *filegroup*) dan langkah tiga adalah memindahkan tabel-tabel hasil analisis awal ke dalam *file system* yang baru dibuat. Langkah empat adalah melakukan analisis kinerja basis data setelah tabel-tabel tersebut dipindahkan ke dalam *temporary file system* untuk selanjutnya langkah lima yaitu Memindahkan kembali tabel-tabel yang sudah dipindahkan ke dalam *file system* awal.

Setelah itu langkah enam yaitu menghapus *temporary file system* dan dilanjutkan dengan langkah tujuh melakukan analisis kembali terhadap kinerja basis data. Sebagai langkah delapan adalah membandingkan hasil pada langkah keempat dan ketujuh dan ditutup dengan langkah Sembilan yaitu menghasilkan rekomendasi berdasarkan analisis-analisis di atas, mengenai tabel mana yang sebaiknya dipindahkan ke dalam *file system* yang berbeda dengan tabel-tabel lainnya dalam basis data.

Proses *tuning* basis data



Gambar 4 *Flowchart* proses *tuning*

Berikut penjelasan gambar 4 di atas, langkah satu adalah membuat sebuah *file group* baru secara otomatis sesuai dengan apa yang diisi oleh *user* dan dilanjutkan dengan langkah dua yaitu membuat *file* baru yang diletakkan pada direktori yang telah diisi oleh *user*. Terakhir ditutup dengan langkah tiga yaitu memindahkan tabel-tabel yang ada pada *list* ke dalam *file system* yang baru dibuat

Untuk menjalankan aplikasi alat bantu *tuning* fisikal basis data pada SQL Server 2008, *user* harus mempunyai data yang berisi histori *query-query* yang pernah dijalankan. Data tersebut dapat diperoleh dengan cara menjalankan SQL Server Profiler terlebih dahulusebelum menjalankan *query-query* tersebut. Dengan demikian, semua data *query* akan tersimpan dalam sebuah tabel yang ditentukan oleh *user* dan bisa digunakan sebagai data awal dalam melakukan analisis.

SIMPULAN

Semakin besar jumlah data dalam suatu basis data menyebabkan kecepatan proses yang dihasilkan menurun karena waktu yang diperlukan untuk mengakses data dalam basis data tersebut juga menjadi lebih lama. Peneliti menyadari bahwa untuk mengatasi permasalahan tersebutdiperlukan suatu teknik yang dapat meningkatkan kinerja sebuah basis data. Hal teknis yang dilakukan dalam usaha meningkatkan kinerja suatu basis data yang lambat adalah dengan menambahkan *index* ke dalam tabel dan dengan memindahkan satu atau beberapa tabel ke dalam *file* fisik yang berbeda. Hal ini terlihat dari hasil waktu respon *query* yang lebih cepat pada tabel yang diletakkan pada *file* fisik yang berbeda dibandingkan dengan waktu respon *query* pada tabel yang diletakkan pada *file* fisik yang sama.Melalui alat bantu *tuning* selain Database Tuning Advisor, makadapat menjadi pertimbangan dan pilihan bagi *Database Engineer*dalam meningkatkan kinerja sebuah basis data perusahaan. Dengan demikian, perusahaan-perusahaan dapat lebih efektif dalam mengakses, mengolah dan memasukkan data, khususnya bagi perusahaan yang memiliki basis data dengan jumlah data yang sangat besar serta membantu dalam pengambilan keputusan yang lebih cepat.

DAFTAR PUSTAKA

Burleson, D. (2010). *Oracle Tuning: The Definitive Reference*.

Connolly, T., Begg, C. (2010). *Database Systems : A Practical Approach To Design, Implementation, and Management.* (5th Edition)**.** New York: Pearson.

Karthik, P., Reddy, G. T., Vanan, E. K. (2012). Tuning the SQL Query in order to Reduce Time Consumption. *IJCSI International Journal of Computer Science Issues, 9*: 418.

Leiter, C., Wood, D., Boettger, A., Cierkowski, M. (2009). *Beginning Microsoft SQL Server 2008 Administration.* Canada: Wiley Publishing.

Petković, D. (2008). *Microsoft SQL Server 2008: A Beginner's Guide.* USA: McGraw-Hill.

Sukheja, D., & Singh, U. K. (2011). A Novel Approach of Query Optimization for Distributed Database Systems. *IJCSI International Journal of Computer Science Issues, 8*: 307.

Whalen, E. (2001). *Microsoft SQL Server 2000 Performance Tuning: Technical reference.* Washington: Microsoft Press.

Whitten, J. L., Bentley, L. D., Dittman, K. C. (2004). *Systems Analysis and Design Methods.* (6th Edition). New York: McGraw-Hill.