SNATI
Seminar Nasional
Aplikasi Teknologi Informasi 2004
Yogyakarta, 19 Juni 2004

Optimasi *SQL Query* untuk *Information Retrieval* pada Aplikasi Berbasis Web

Mukhammad Andri Setiawan
Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia
Jl. Kaliurang Km. 14,5 Yogyakarta 55501, Telp. (0274) 895287, Faks. (0274) 895007
e-mail: andri@fti.uii.ac.id

Abstract

Nowadays web has expanded rapidly. It has no longer present static information, but it developed into dinamically information provider which provide a new information in a fast way using what we called web based application. Using this application, web can provides users to the information that they need. In it's expansion, web can hold so much information to maintain, this condition can make the information retrieval effort become a problem. This paper will analyze the use of optimized query for retrieving information from a database using web based application. The result is an understanding of the importance of using optimized query to retrieve information to be presented into a web page.

Keywords: *query optimization, information retrieval, web based application*

1. Pendahuluan

Pada masa sekarang ini internet bukan lagi menjadi hal yang asing dan aneh terutama bagi orang-orang yang berkecimpung di dunia komputer dan teknologi informasi bahkan bagi orang yang awam sekalipun [2]. *World wide web* (www) yang merupakan bagian dari internet telah menjadi bagian dari kehidupan manusia, dimana web menjadi sumber informasi yang sangat dibutuhkan dikarenakan web dapat menyajikan informasi yang diinginkan secara mudah, cepat, dan murah, apalagi ditunjang dengan sifatnya yang mendunia.

Pada perkembangannya, web telah meluas fungsinya dengan adanya aplikasi-aplikasi yang dibangun di atas *platform* berbasis web yang lazim dikenal sebagai *web based application* (aplikasi berbasis web) [4]. Di era sebelumnya penyajian informasi bersifat statis, setelah berkembangnya teknologi aplikasi berbasis web penyajian informasi menjadi bersifat lebih dinamis. Aplikasi berbasis web memungkinkan sebuah proses dinamisasi dengan cara mengambil informasi dari basisdata untuk kemudian ditampilkan ke dalam halaman web. Ketika informasi yang dimiliki relatif kecil, proses pencarian informasi dapat berjalan relatif mudah, akan tetapi ketika jumlah informasi yang disajikan semakin banyak, maka proses pencarian dan penampilan informasi tersebut ke dalam halaman web juga akan menjadi kendala tersendiri dan aplikasi harus dapat merespon akan hal ini [3]. Untuk itu diperlukan sebuah cara dan mekanisme tertentu agar proses *information retrieval* dapat berjalan dengan cepat, karena kecepatan merupakan faktor yang sangat penting dalam proses *information retrieval* [3].

Tulisan berikut berupaya menyajikan sebuah analisis penggunaan *query SQL* yang teroptimasi untuk mempercepat penyajian informasi ke dalam halaman web dan membandingkannya dengan *query* yang tidak teroptimasi.

2. Kajian Pustaka

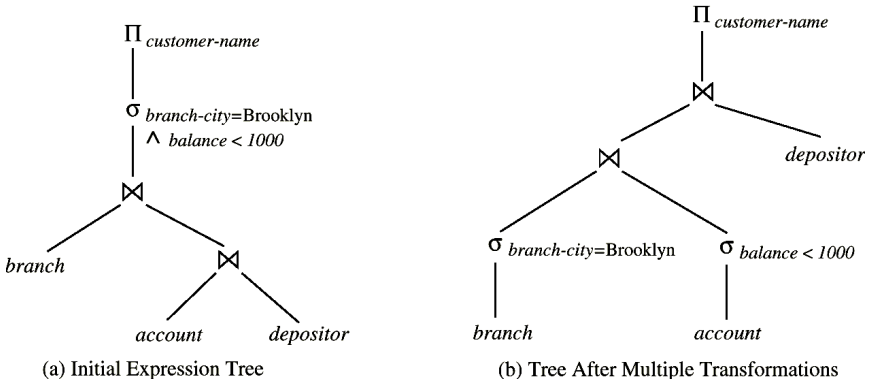
Teknik optimasi dapat dilakukan dengan beberapa cara. Terdapat dua pendekatan optimasi yang umum dipergunakan sebagaimana yang telah dikemukakan oleh Chanowich (2001), yakni:

a. Heuristik atau *rule-based*

Teknik ini mengaplikasikan aturan heuristik untuk mempercepat proses *query*. Optimasi jenis ini mentransformasikan *query* dengan sejumlah aturan yang akan meningkatkan kinerja eksekusi, yakni:

- melakukan operasi *selection* di awal untuk mereduksi jumlah baris,
- melakukan operasi *projection* di awal untuk mengurangi jumlah atribut,
- mengkonversikan *query* dengan banyak *join* menjadi *query* dengan banyak *subquery*, dan
- melakukan operasi *selection* dan *join* yang paling kecil keluarannya sebelum operasi yang lain.

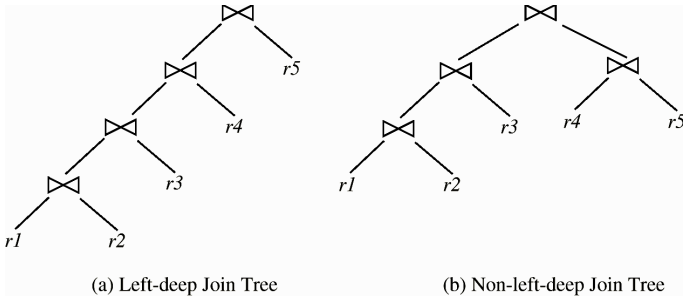
Gambar 1 memperlihatkan bagaimana sebuah *query* ditransformasikan untuk mendapatkan eksekusi yang lebih cepat dengan mengaplikasikan aturan-aturan yang dipergunakan dalam optimasi heuristik.



Gambar 1. Optimasi heuristik pada sebuah *query-tree*

b. *Cost-based*

Teknik ini mengestimasikan *cost* yang dipergunakan dari beberapa alternatif untuk kemudian dipilih salah satu yang memiliki *cost* terendah. Teknik ini mengoptimalkan urutan *join* terbaik yang dimungkinkan pada relasi-relasi $r_1 \bowtie r_2 \bowtie \ldots r_n$. Teknik ini dipergunakan untuk mendapatkan pohon *left-deep join* yang akan menghasilkan sebuah relasi sebenarnya pada *node* sebelah kanan yang bukan hasil dari sebuah *intermediate join* [5]. Gambar 2 memperlihatkan optimasi jenis ini.



Gambar 2. Optimasi *cost-based* pada sebuah *query-tree*

3. Metodologi Penelitian

Penelitian ini dilakukan dengan melakukan pengambilan data melalui sebuah aplikasi berbasis web yang dibangun dengan ASP dan PHP sebagai script *server side* dengan menggunakan IIS 5.1 sebagai web server. RDBMS (*Relational Database Management System*) yang dipergunakan adalah Microsoft Access 2002/XP dan MySQL 4.11. Skema tabel yang dipergunakan adalah sebagai berikut:

tabel1(id, Satu, Dua, Tiga)

tabel2(id, Tiga, Empat, Lima)

Operasi dilakukan dengan cara menginputkan rekord pada dua buah tabel tersebut dengan kenaikan sebanyak 1000 tuple untuk setiap kali operasi. Setelah dilakukan pemasukan data, maka dilanjutkan dengan pengambilan data berupa pencatatan waktu yang dipergunakan atas pengeksekusian *query* SQL yang dipergunakan. Pencatatan dilakukan sebanyak tiga kali untuk tiap *query* baik pada *query* teroptimasi maupun yang tidak teroptimasi dan kemudian diambil rata-ratanya.

Untuk membandingkan kinerja, dibuat dua buah *query* dengan pendekatan yang berbeda. *Query* pertama yang tidak teroptimasi yakni

```
SELECT Satu,Dua
FROM tabel1,tabel2
WHERE tabel1.Tiga = tabel2.Tiga
```

menghasilkan *intermediary table* yang cukup besar. Pada *query* ini, dilakukan terlebih dahulu operasi *cross product* (×) antara *tabel1* dan *tabel2*, kemudian dicari rekord yang bersesuaian dengan kekangan yang diberikan. Operasi *cross product* inilah yang mengakibatkan proses *information retrieval* menjadi lambat. Misal kedua tabel memuat 1000 baris dengan 3 kolom maka paling tidak akan dihasilkan sebuah *intermediary table* sebanyak 3.000.000 rekord. Sementara *query* kedua yang teroptimasi

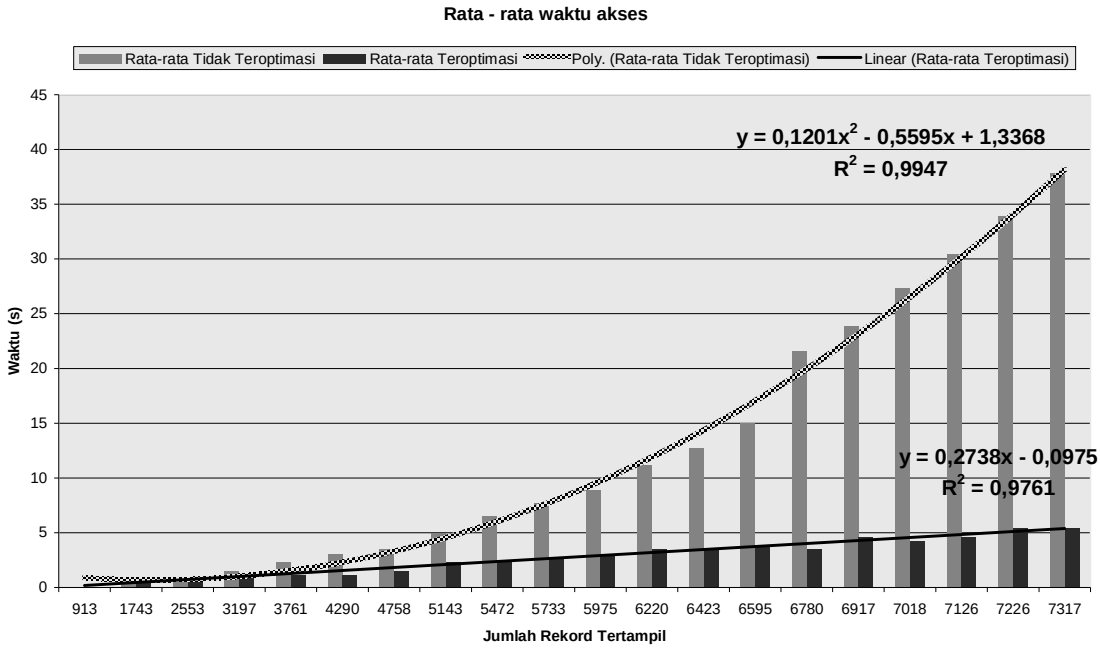
```
SELECT Satu,Dua
FROM tabel1
WHERE Tiga IN (SELECT Tiga FROM tabel2)
```

menghasilkan *intermediary table* yang lebih kecil. *Query* ini menggunakan *subquery* yang menghasilkan tabel yang memuat rekord yang lebih spesifik sesuai dengan kekangan yang diberikan. Misal kedua tabel memuat 1000 baris dengan 3 kolom dan dikenai *query tersebut* hanya akan menghasilkan sebuah *intermediary table* sebanyak 1.000.000 rekord, karena operasi *subquery* hanya menghasilkan 1000 baris dengan satu kolom saja. *Query* ini dioptimasi dengan metode *Simple Heuristic Optimizations* sebagaimana telah dijelaskan di atas.

4. Hasil Penelitian dan Pembahasan

4.1. Pengambilan Data dengan RDBMS Microsoft Access 2002/XP

Pengambilan data pertama pada aplikasi berbasis web menggunakan RDBMS Microsoft Acess 2002/XP yang dihubungkan dengan aplikasi melalui sebuah *connection string*. Dilakukan *entry data* rekord acak yang diinputkan ke dalam setiap tabel dalam basisdata. Kemudian dilakukan pembacaan *field Satu* dan *Dua* dari *tabel1* setelah dilakukan input data, dimana *field Tiga* dari *tabel1* sama dengan *field Tiga* dari *tabel2*. Pengujian dilakukan dalam komputer *localhost* dimana client sekaligus bertindak sebagai server. Proses ini dilakukan sebanyak 20 kali yang berarti dilakukan *entry* data hingga mencapai 20.000 tuple. Hasil pengambilan data ditunjukkan pada Gambar 3.

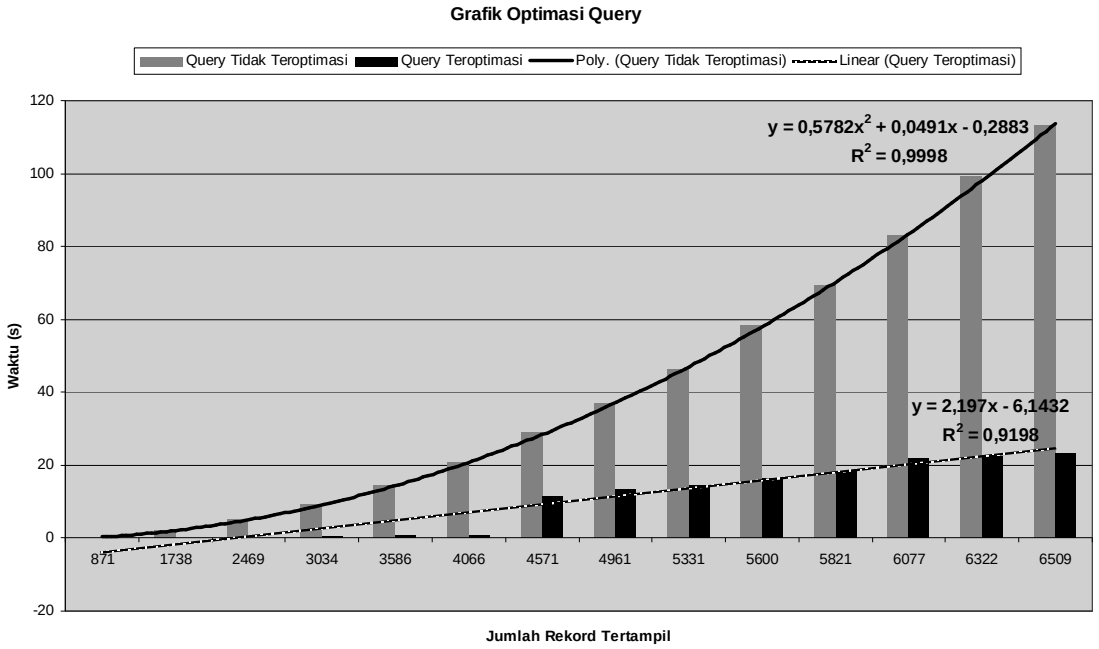


Gambar 3. Rata-rata waktu akses Aplikasi berbasis Web dengan RDBMS Microsoft Access 2002/XP

Gambar 3 menunjukkan tren pengaruh penggunaan *query* yang teroptimasi dan yang tidak. Gambar tersebut memperlihatkan sebuah relasi kuadratik yang kuat, ditunjukkan dengan R mendekati 1, pada *query* yang tidak teroptimasi sebagaimana ditunjukkan secara statistik dengan koefisien korelasi yang tinggi (R^2 = 0,9947 atau R = 0,99). Waktu akses *query* ini berubah menurut polinomial berderajad dua dengan persamaan $y = 0{,}1201x^2 - 0{,}5595x + 1{,}3368$. Untuk *query* yang teroptimasi, Gambar 3 menujukkan relasi linear yang kuat sebagaimana ditunjukkan melalui statistik data yang memperlihatkan koefisien korelasi yang tinggi (R^2 = 0,9761 atau R = 0,99). Waktu akses *query* ini berubah secara linear dengan persamaan $y = 0{,}2738x - 0{,}0975x$.

4.2. Pengambilan Data dengan RDBMS MySQL 4.11

Pengambilan data kedua dilakukan dengan menggunakan RDBMS MySQL 4.11 yang dihubungkan dengan aplikasi melalui koneksitas ODBC. Dilakukan *entry data* dengan rekord acak yang diinputkan ke dalam *tabel1* dan *tabel2*. Kemudian dilakukan pembacaan *field Satu* dan *Dua* dari *tabel1* dimana *field Tiga* dari *tabel1* sama dengan *field Tiga* dari *tabel2* setelah dilakukannya proses input data. Pengujian dilakukan dalam komputer *localhost* dimana client sekaligus bertindak sebagai server. Proses ini dilakukan sebanyak 14 kali yang berarti dilakukan *entry* data hingga mencapai 14.000 tuple. Hasil pengambilan data ditunjukkan pada Gambar 4.



Gambar 4. Rata-rata waktu akses Aplikasi berbasis Web dengan RDBMS MySQL 4.11

Gambar 4 memperlihatkan sebuah relasi polinomial pangkat dua untuk *query* yang tidak teroptimasi sebagaimana ditunjukkan dengan kuat secara statistik melalui koefisien korelasi yang tinggi yakni R mendekati 1 yang berarti tren kenaikan secara polinomial berderajad dua ini cukup terpercaya (R^2 = 0,9998 atau R = 0,9999). Waktu akses *query* tidak teroptimasi ini berubah menurut polinomial berderajad dua dengan persamaan $y = 0{,}5782x^2 + 0{,}0491x - 0{,}2883$. Sementara, untuk *query* yang teroptimasi, ditunjukkan sebuah relasi linear yang cukup kuat sebagaimana ditunjukkan melalui statistik data yang memperlihatkan koefisien korelasi yang tinggi (R^2 = 0,9198 atau R = 0,96). Waktu akses *query* teroptimasi ini ditunjukkan dengan persamaan $y = 2{,}197x - 6{,}1432$.

4.3. Pembahasan

Pengambilan data dengan menggunakan RDBMS Microsoft Access 2002/XP dilakukan sebanyak 20 kali dengan jumlah *tuple* sebanyak 20.000, sementara untuk RDBMS MySQL 4.11 dilakukan sebanyak 14 kali atau sebanyak 14.000 tuple. Dari jumlah baris berisi rekord acak tersebut, telah dapat digambarkan tren kenaikan waktu akses. Perbedaan jumlah input *tuple* antara Microsoft Access 2002/XP dan MySQL 4.11 terjadi karena pada MySQL 4.11 setelah diisikan sebanyak 14.000 rekord skrip dengan *query* tidak teroptimasi menjadi terhenti (*Script Time Out*) karena waktu akses yang terlalu lama melebihi batas waktu akses yang diizinkan oleh web server.

Dari grafik yang ditunjukkan oleh Gambar 3 dan Gambar 4, diperlihatkan sebuah tren yang mengindikasikan perbedaan yang jauh ketika dilakukan operasi *information retrieval* dengan menggunakan *query* teroptimasi dan *query* yang tidak teroptimasi. Dari kedua jenis RDBMS yang dipergunakan didapatkan kecenderungan yang hampir sama, dimana penggunaan *query* teroptimasi dapat mempercepat proses pengambilan data untuk ditampilkan ke dalam browser, hal ini ditunjukkan dengan kenaikan waktu yang bersifat linear tidak seperti halnya pada *query* yang tidak teroptimasi yang menunjukkan kenaikan waktu bersifat kuadratik.

Dari hasil penelitian juga didapat suatu kondisi dimana web server akan memutuskan koneksi sebuah *script* apabila dirasa koneksi tersebut terlalu lama berlangsung dengan server. Dengan dilakukannya optimasi *query*, maka kondisi tersebut dapat diminimalisir.

5. Kesimpulan

Dari hasil penelitian yang telah dilakukan dapat ditarik kesimpulan bahwa penggunaan *query* teroptimasi dapat mempercepat proses *information retrieval* pada aplikasi berbasis web secara signifikan dengan karakteristik yang sama meskipun menggunakan RDBMS yang berbeda-beda. Penggunaan *query* yang teroptimasi juga dapat menjaga keberlangsungan proses *information retrieval* agar dapat tertampil ke dalam halaman web, karena dapat meminimalisir terjadinya *Script Time Out* yang dilakukan oleh web server yang akan mengakibatkan proses penampilan informasi berhenti.

Daftar Pustaka

[1] Chanowich, E. dan Sendelbach, E., "Query Optimization," in *CSE 498G – Advanced Databases*, 2001.

[2] Kurniawan, A. (2000) *Belajar Sendiri Microsoft Active Server Pages*, PT. Elex Media Komputindo, Jakarta.

[3] Savoy, J. (2002) Information Retrieval on the Web: A New Paradigm, *UPGRADE*, Vol. III, No. 3, 9-11.

[4] Setiawan, M.A., "Content Management System Dengan Active Server Pages Pada Situs DPD PK Sleman," in *Teknik Elektro*. Yogyakarta: Gadjah Mada, 2002.

[5] Silberschatz, A., Korth, H.F., dan Sudarshan, S. (2001) *Database Systems Concepts*, McGraw-Hill.