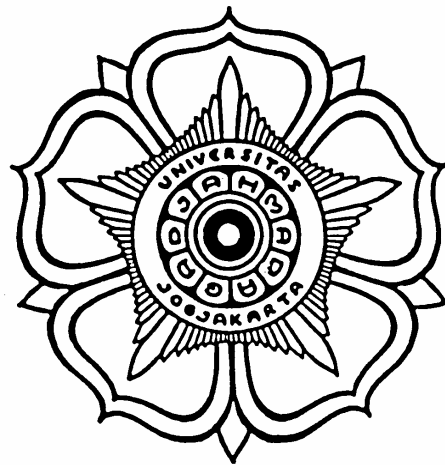


**PERANCANGAN dan PEMBUATAN
APLIKASI PENGOPTIMAL SQL *QUERY***

Naskah Publikasi

Program Studi Magister Teknologi Informasi
Jurusan Teknik Elektro
Fakultas Teknik



diajukan oleh
Ogi Sigit Pornawan
17930/PS/MTI/05

kepada
SEKOLAH PASCASARJANA
UNIVERSITAS GADJAH MADA
2007

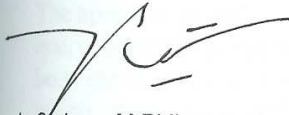
NASKAH PUBLIKASI

PERANCANGAN dan PEMBUATAN
APLIKASI PENGOPTIMAL SQL QUERY

Ogi Sigit Pornawan, ST.
17930/PS/MTI/05

Telah disetujui untuk diajukan sebagai naskah publikasi oleh :

Dosen Pembimbing I



Ir. Surjono, M.Phil.

NIP.

tanggal

Dosen Pembimbing II



Ir. Abdul Kadir, M.M, M.T.

NIP.

tanggal

PERANCANGAN dan PEMBUATAN APLIKASI PENGOPTIMAL SQL QUERY

1. PENDAHULUAN

Pengoptimasian SQL *query* dengan baik memerlukan keahlian khusus dan analisa secara mendalam pada setiap *query*. Karenanya diperlukan suatu alat untuk dapat mengoptimasi SQL secara otomatis, sehingga dapat menyingkat waktu dalam mengoptimasi SQL dan juga mengurangi keahlian khusus yang diperlukan manusia untuk mengoptimasi SQL.

Dalam penelitian ini akan dirancang dan dibuat aplikasi pengoptimal SQL *query* untuk mempermudah dan mempercepat pengoptimalan SQL *query*, sehingga diharapkan orang tanpa keahlian khusus dibidang optimasi SQL *query* dapat mengoptimasi SQL *query* dengan menggunakan aplikasi yang akan dibuat.

2. CARA PENELITIAN

Pembangunan Aplikasi Pengoptimal SQL *Query* menggunakan model pengembangan software Systems Development Life Cycle.

Penelitian dilakukan dengan langkah-langkah sebagai berikut :

Tahap studi literatur

Tahap studi literatur dilakukan dengan mengumpulkan metode optimasi SQL *query* yang pernah diteliti oleh peneliti sebelumnya. Kemudian metode tersebut diterjemahkan kedalam bahasa matematika atau bahasa pemrograman.

Setelah diterjemahkan kedalam bahasa matematika atau bahasa pemrograman, kemudian di metode dan algoritma optimasi SQL *query* yang memungkinkan untuk diimplementasikan pada aplikasi pengoptimal SQL *query* yang akan dibuat.

Analisa

- Penentuan masalah
- Penentuan sasaran pembuatan aplikasi
- Pengidentifikasian pengguna
- Penentuan lingkup aplikasi

Desain Aplikasi

- *Data Flow Diagrams*
 - Diagram konteks

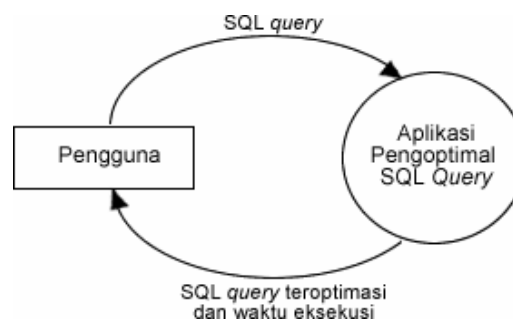
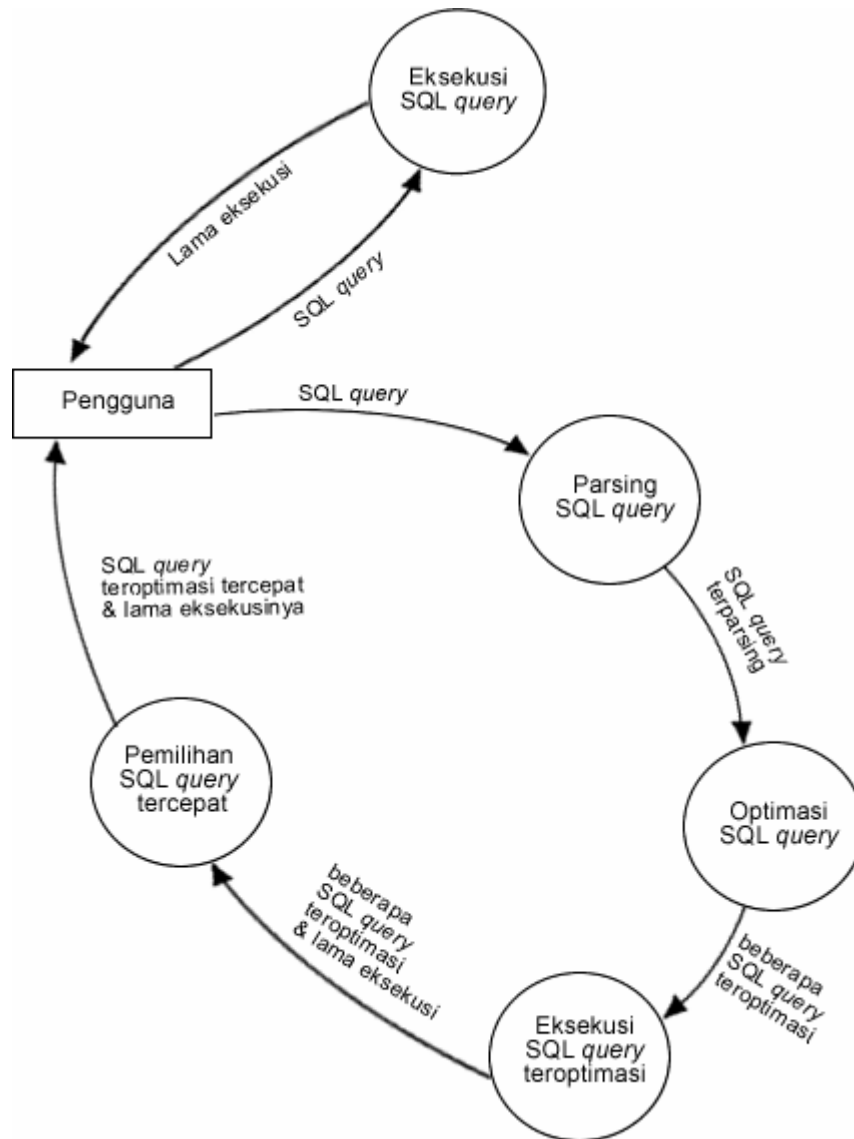


Diagram Konteks

Diagram konteks pada gambar di atas menggambarkan Aplikasi Pengoptimal SQL *Query* secara umum yang memperlihatkan hubungannya dengan pengguna beserta alur data yang masuk dan keluar dari aplikasi.

- DFD Level I

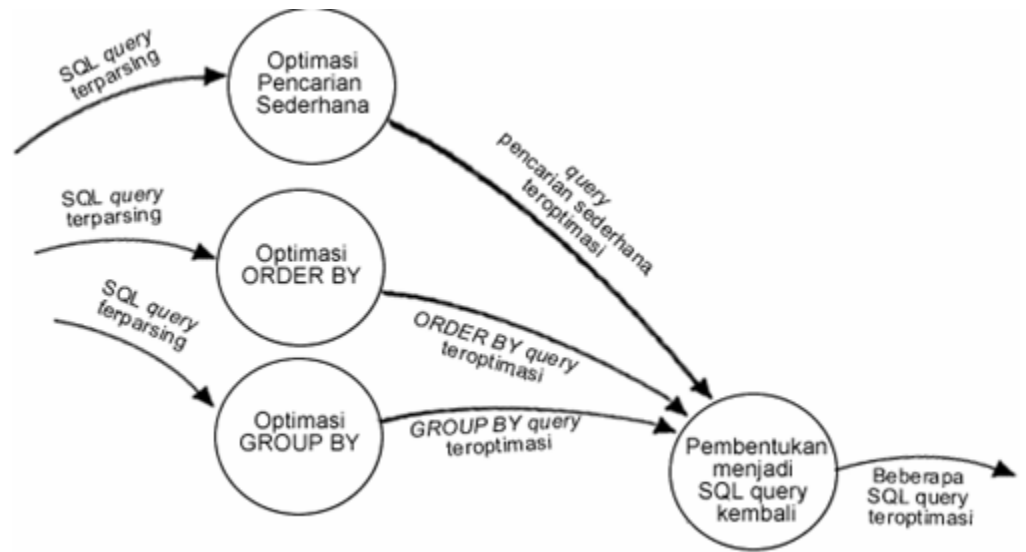


DFD Level I

Diagram konteks kemudian dikembangkan menjadi DFD level I yang dapat dilihat pada gambar diatas.

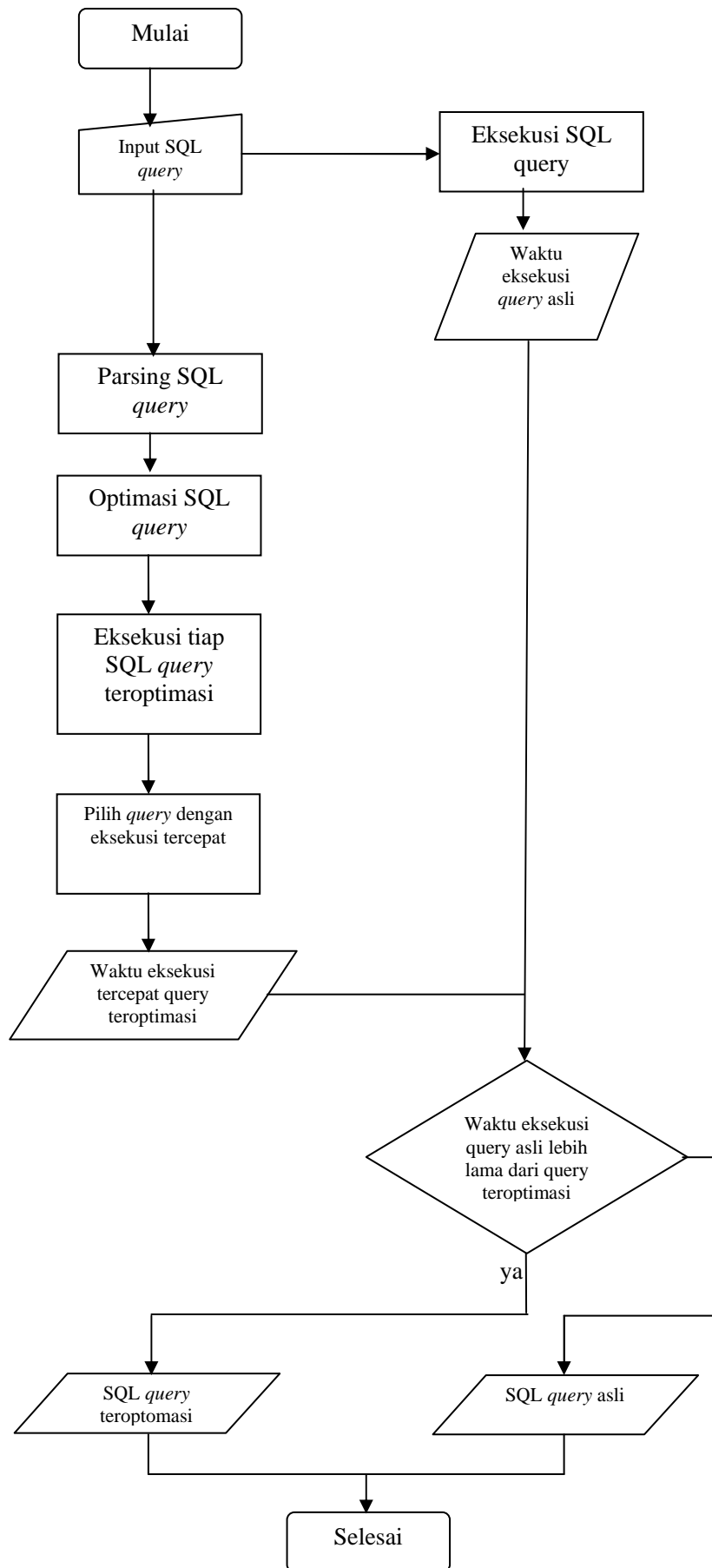
iii. DFD Level II

Proses optimasi SQL query pada gambar di atas dijabarkan secara lebih mendetail pada DFD Level II pada dibawah



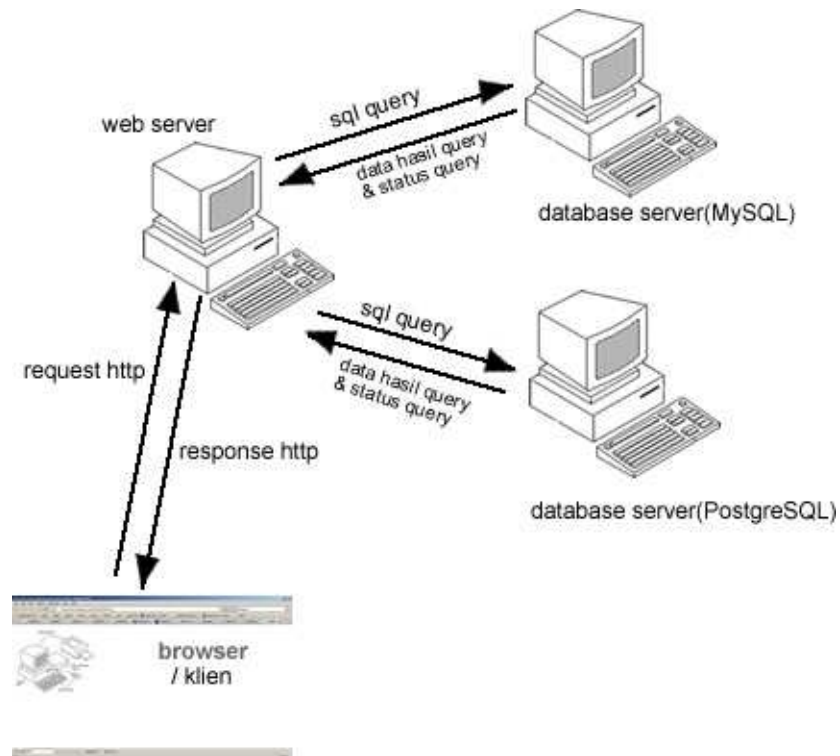
DFD Level II

Diagram alur aplikasi



Flowchart Aplikasi Pengoptimal SQL query

- Perancangan antar muka aplikasi
 - i. Form Input Pengguna
 - ii. Output Aplikasi
- Perancangan skema kerja aplikasi



Gambar Skema Kerja Aplikasi

Implementasi

Pada tahapan ini akan dibuat Aplikasi Pengoptimal SQL *Query* dengan menggunakan bahasa pemrograman PHP.

Ujicoba

Ujicoba akan dilakukan pada 2 macam database yaitu MySQL dan PostgreSQL, dan dilakukan pada berbagai variasi jumlah data.

3. HASIL PENELITIAN

Pembuatan Algoritma dari Tiap Metode Optimasi

- Metode Propagasi Konstanta

Pengecekan:

```
Untuk setiap operator pada WHERE
    jika operator = AND
        lanjutkan
```

```
Jika ada kolom pada WHERE muncul 2 kali dengan operator ( = )
    berhenti
```

Pengoptimasian:

```
Target Kolom =
    Dapatkan nama kolom yang muncul 2 kali dengan operator ( = )
```

```
Nilai Konstanta =
    Dapatkan nilai konstanta pada Target Kolom dengan operator ( = )
```

```
Kolom Dioptimasi =
    Dapatkan nama kolom beroperator ( = ) dengan Target Kolom
```

```
Hasil Optimasi = "Kolom dioptimasi = Nilai Konstanta"
```

```
Hapus kondisi WHERE yang mengandung Target Kolom
```

```
SQL query baru = SQL query + Hasil Optimasi
```

- *Constant Folding*

Pengecekan:

jika jumlah nilai konstanta pada klausa WHERE pada kolom yang sama < 2
berhenti

Pengoptimasian:

Target Kolom = Dapatkan nama kolom yang mempunyai nilai konstanta >= 2

Hasil Konstanta = 0

Untuk tiap kondisi dg Target Kolom dan mempunyai nilai konstanta

Hasil Konstanta = Hasil Kontanta + nilai konstanta

Hasil Optimasi = "Target Kolom = Hasil Konstanta"

Hapus kondisi WHERE dengan Target Kolom dan mempunyai nilai konstanta

SQL query baru = SQL query + Hasil Optimasi

- *Sargability*

Pengecekan:

Untuk setiap kondisi pada klausa WHERE

Jika di sebelah kiri operator bukan nama kolom

Lanjutkan Optimasi

Pengoptimasian:

Target Kondisi =

Dapatkan kondisi yang sebelah kiri opetator bukan nama kolom

Operator Target Kondisi = Dapatkan operator dari Target Kondisi

Kolom Index = ""

Untuk setiap nama kolom pada Target Kondisi

Jika kolom terindex

Kolom Index = nama kolom

Jika Kolom Index == ""

Untuk setiap nama kolom pada Target Kondisi

Kolom Index = nama kolom

break

Untuk setiap entitas pada Target Kondisi sebelah kiri

Jika entitas != Kolom Index

entitas = -entitas

Tambahkan entitas ke Kondisi Kanan

Untuk setiap entitas pada Target Kondisi sebelah kanan

Jika entitas != Kolom Index

Tambahkan entitas ke Kondisi Kanan

Hasil Optimasi =

"Kolom Index + Operator Target Kondisi + Kondisi Kanan"

Hapus kondisi WHERE yang mengandung Target Kondisi

SQL query baru = SQL query + Hasil Optimasi

- Alternatif Sintaks OR

Pengecekan:

Untuk setiap kondisi pada klausa WHERE dengan operator OR

Jika operator pada kondisi tersebut == " = "

Lanjutkan Optimasi

Pengoptimasian:

Target Kolom = Dapatkan nama kolom dengan operator " = "

Untuk tiap kondisi dengan Target Kolom

Isi IN = isi IN + isi kondisi sebelah kanan operator " = "

Hasil Optimasi = "Target Kolom + ' IN (' + isi IN + ') ' "

Hapus kondisi WHERE yang mengandung Target Kolom

SQL query baru = SQL query + Hasil Optimasi

- Optimasi Sintaks IN

Pengecekan:

Jika ada duplikasi nilai pada isi IN pada klausa WHERE

Lanjutkan Optimasi

Pengoptimasian:

Target Kondisi =

Dapatkan kondisi IN yang mempunyai duplikasi nilai

Target Kolom =

Dapatkan nama kolom sebelah kiri IN pada Target Kondisi

Untuk tiap nilai pada Target Kondisi

Jika nilai tidak terdapat pada Array Nilai Baru

Tambahkan nilai pada Array Nilai Baru

Isi IN Baru =

Ubah Array Nilai Baru ke string dengan separator ","

Hasil Optimasi = "Target Kolom + ' IN (' + Isi IN Baru + ') '

Hapus kondisi WHERE yang mengandung Target Kondisi

SQL query baru = SQL query + Hasil Optimasi

- Alternatif Sintaks IN

Pengecekan:

Jika pada nilai IN,

terdapat perbandingan yang tak terurut dan yang terurut $\geq 3/5$

Lanjutkan Optimasi

Pengoptimasian:

Target Kondisi =

Dapatkan kondisi IN yang mempunyai nilai hampir terurut

Tarket Kolom =

Dapatkan nama kolom sebelah kiri IN pada Target Kondisi

Untuk tiap nilai pada Target Kondisi

Tambahkan nilai ke Array Nilai

Array Nilai = urutkan Array Nilai

Nilai Terkecil = Array Nilai [pertama]

Nilai Terbesar = Array Nilai [terakhir]

Untuk setiap Array Nilai

jika nilai \neq temp + 1

Untuk $i=temp+1$ selama $i<nilai-1$

Tambahkan i kedalam Array Negasi

i = i+1

temp = nilai

Untuk setiap Array Negasi

Hasil Negasi =

Hasil Negasi + " AND " + Target Kolom + ' <> ' + nilai negasi

Hasil Optimasi =

"Target Kolom + 'BETWEEN ' + Nilai Terkecil
+ ' AND ' + Nilai Terbesar + Hasil Negasi

Hapus kondisi WHERE yang mengandung Target Kondisi

SQL query baru = SQL query + Hasil Optimasi

- **Penyederhanaan Fungsi**

Pengecekan:

Jika terdapat fungsi database dipanggil

lebih dari satu kali pada kolom yang sama

Jika kondisi tersebut disyaratkan hasilnya

sama dengan suatu nilai konstanta.

Lanjutkan Optimasi

Pengoptimasian:

Array Target Kondisi =

Dapatkan kondisi dg fungsi yang dipanggil lebih dari 1 kali

Target Fungsi =

Dapatkan fungsi dari Array Target Kondisi

Target Kolom =

Dapatkan nama kolom yang dijadikan argumen Target Fungsi

Array Konstanta =

Dapatkan nilai-nilai konstanta dari tiap Array Target Kondisi

Untuk setiap Array Konstanta

Hasil When = Hasil When + nilai konstanta + ' THEN 1 '

Hasil Optimasi =

' 1 = CASE ' + Target Fungsi + Target Kolom
+ Hasil When + ' End'

Hapus kondisi WHERE yang mengandung Array Target Kondisi

SQL query baru = SQL query + Hasil Optimasi

Pembuatan Antar Muka Aplikasi

- Form Input Pengguna

APLIKASI PENGOPTIMAL SQL QUERY

Form Input

Database:

Username Database:

Database:

Database:

SQL Query:

Copyright © 2007

- Output Aplikasi

APLIKASI PENGOPTIMAL SQL QUERY

Hasil Optimasi SQL Query

Database : **Oracle**

SQL Query Sebelum Dioptimalisasi:

SQL Query Sesudah Dioptimalisasi:

Tabel Perbandingan lama eksekusi query :

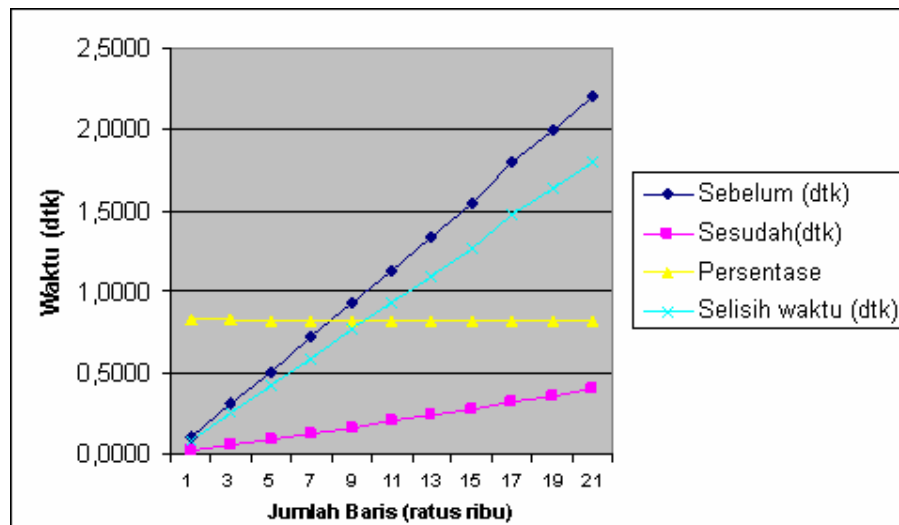
Eksekusi ke	Sebelum dioptmasi	Sesudah dioptmasi
1	0.012 dtk	0.011 dtk
2	0.012 dtk	0.011 dtk
3	0.012 dtk	0.011 dtk
Rata-rata	0.012 dtk	0.011 dtk
Selisih	0.012 dtk	

[<< Kembali](#)

Copyright © 2007

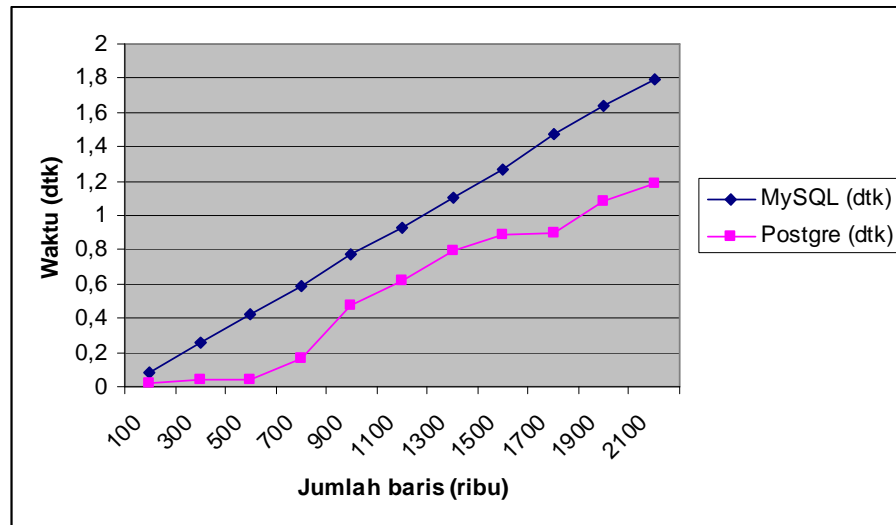
Ujicoba Aplikasi

- Percobaan I



Grafik Statistik Percobaan I pada Database MySQL

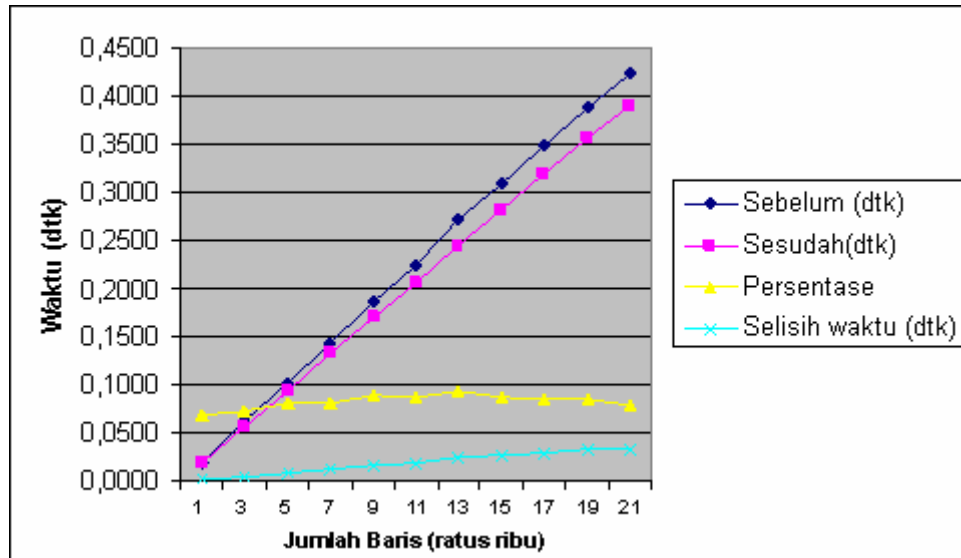
Dari pengamatan statistik percobaan pertama pada database MySQL, persentase pengurangan waktu eksekusi SQL *query* dengan menggunakan Aplikasi Pengoptimal SQL *Query* pada tabel dengan berbagai variasi jumlah data cenderung tetap dan membentuk garis lurus horisontal. Sedangkan selisih waktu lama eksekusi sebelum dan sesudah dioptimasi selalu bertambah dengan semakin banyaknya data pada suatu tabel dan membuat garis lurus naik secara konstan.



Grafik Perbandingan Selisih Waktu Eksekusi SQL query pada MySQL dan PostgreSQL

Percobaan pada database MySQL dan PostgreSQL diatas kemudian digrafikkan pada satu gambar grafik pada Gambar di atas. Dari pengamatan grafik tersebut terlihat bahwa pada jumlah data yang semakin besar, percepatan waktu eksekusi SQL *query* pada database MySQL lebih besar jika dibandingkan dengan percepatan waktu eksekusi SQL query pada database PostgreSQL.

- **Percobaan II**

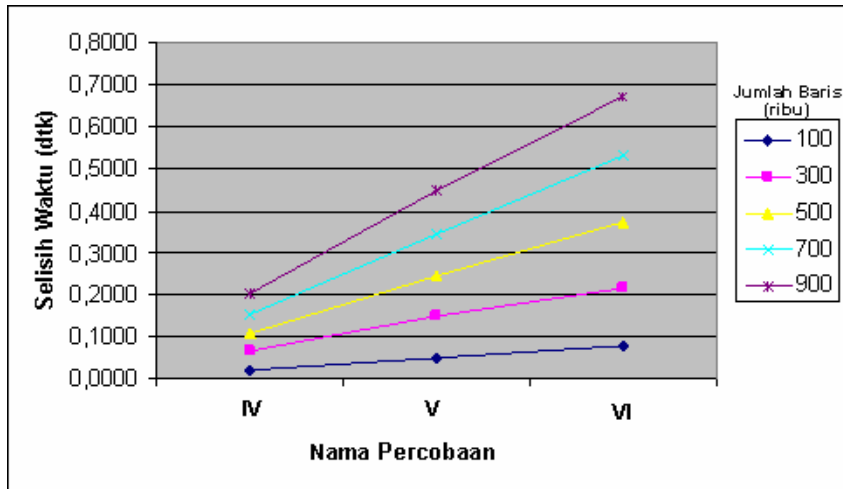


Grafik Statistik Percobaan II pada Database MySQL

Dari pengamatan statistik percobaan kedua, persentase pengurangan waktu eksekusi SQL *query* dengan menggunakan Aplikasi Pengoptimal SQL *Query* pada tabel dengan berbagai variasi jumlah data cenderung tetap dan membentuk garis lurus horisontal. Sedangkan selisih waktu lama eksekusi sebelum dan sesudah dioptimasi selalu bertambah dengan semakin banyaknya data pada suatu tabel dan membuat garis lurus naik secara konstan.

- **Percobaan IV, V dan VI**

Pada Gambar di bawah, dapat diamati bahwa SQL *query* yang dihasilkan pada jumlah baris yang sama dengan perbedaan jumlah kolom pada klausa WHERE yang semakin besar mempunyai selisih waktu eksekusi yang semakin membesar juga.



Grafik Statistik Percobaan IV, V, VI pada Database MySQL dengan Perbandingan Jumlah Data pada Tiap Percobaan

4. KESIMPULAN

1. Aplikasi Pengoptimal SQL *Query* menghasilkan SQL *query* dengan waktu eksekusi sama dengan atau lebih cepat dari input SQL *query*-nya.
2. Percepatan waktu eksekusi SQL *query* yang dihasilkan Aplikasi Pengoptimal SQL *Query* bergantung salah satunya pada jumlah kolom pada SQL *query* yang dapat dioptimasi.
3. Percepatan waktu eksekusi SQL *query* yang dihasilkan oleh Aplikasi Pengoptimal SQL *Query* semakin besar secara linier pada jumlah data yang lebih besar pada database MySQL.
4. Pada data yang lebih banyak, percepatan waktu eksekusi SQL *query* pada database MySQL lebih besar dibandingkan pada database PostgreSQL.
5. Rata-rata percepatan waktu eksekusi SQL *query* yang dihasilkan oleh Aplikasi Pengoptimal SQL *Query* lebih besar dari 3%, sehingga hipotesa dari penelitian ini tercapai.

5. DAFTAR PUSTAKA

- Axmark, David; & Widenius, Michael. 2003. *MYSQL HELP 4.0.11*. Swedish.
- Balling, Derek J.; & Zawodny, Jeremy. 2004. *High Performance MySQL*. California: O'Reilly Publishing.
- England, Ken. 2001. *Microsoft SQL Server 2000 Performance Optimization and Tuning Handbook*. USA: Digital Press.
- Gulutzan, Peter; & Pelzer, Trudy. 2002. *SQL Performance Tuning*. USA: Addison Wesley
- Mata-toledo, Ramon A.; & Cushman, Pauline K. 2000. *Fundamentals of Relational Databases*. : McGraw-Hill Companies.
- Mishra, Sanjay; & Beaulieu, Alan. 2004. *Mastering Oracle SQL, 2nd Edition*. California: O'Reilly Media.
- Silberschatz, Abraham, et. al. *Database System Concepts (Fourth Edition)*. : McGraw-Hill Companies.
- Tow, Dan. 2003. *SQL Tuning*. California: O'Reilly Publishing.
- Ueberhuber, Christoph W. 1997. *Numerical Computation 1: Methods, Software, and Analysis*. Vienna: Springer-Verlag.
- Wordnet. 2005. *Princeton University Cognitive Science Laboratory*.
<http://wordnet.princeton.edu/perl/webwn?s=application> . Diakses tanggal 15 Desember 2006.