PEMANFAATAN ALGORITMA DIJKSTRA PADA RETRIEVAL DATA DARI BASIS DATA TERSEBAR

Muqtafi Akhmad (13508059)

Teknik Informatika ITB
Bandung
e-mail: if18059@students.if.itb.ac.id

ABSTRAK

Dalam makalah ini akan dibahas tentang pemanfaatan metode pencarian *shortest path* menggunakan algoritma Dijkstra pada pencarian jalur *retrieval* data paling pendek (dengan *cost* paling kecil) pada basis data. Pembahasan dimulai dengan pengenalan mengenai basis data tersebar dan algoritma Dijkstra. Kemudian akan disajikan hasil pengujian dari algoritma Dijkstra melalui simulasi sistem basis data tersebar dan menelaah apakah algoritma Dijkstra cukup efektif untuk mencari path paling pendek untuk *retrieval* data sebuah komputer dari komputer pada *node* yang lain.

Kata kunci: basis data tersebar, *record*, *retrieval*, *database*

1. PENDAHULUAN

Kebutuhan penyimpanan dan pengolahan data dari basis data semakin meningkat dengan semakin kompleksnya kebutuhan penyimpanan data dan volume dari data. Sebagai konsekuensi dari semakin tingginya volume data, pembacaan data dari dalam disk dan pengolahannya membutuhkan waktu yang lebih daripada sebelumnya, dan akan sangat membebani jika basis data dalam volume besar hanya ditangani oleh satu mesin. Lebih jauh lagi, jika kita akan membicarakan kinerja, semakin lama sebuah mesin mengeksekusi satu transaksi (permintaan), maka semakin *response time* sistem akan semakin tinggi dan *availability* basis data akan semakin rendah. Oleh karena itu, muncul konsep untuk membagi-bagi basis data dari satu mesin ke beberapa mesin dalam satu jaringan untuk meningkatkan performa dan kapasitas penyimpanan dari basis data sehingga layanan penyimpanan data mampu menangani volume data yang besar. Ide dari sistem basis data terdistribusi didukung dengan semakin berkembangnya kecepatan transfer data dalam satu jaringan, sehingga pengiriman data dari satu *node* ke *node* lain dalam satu jaringan tidak lagi menjadi masalah.

Dengan semakin meningkatnya kebutuhan jumlah penyimpanan dan akses data dari sebuah basis data, tidak cukup kita hanya mengandalkan kecepatan jaringan untuk mencapai kinerja maksimal dari basis data tersebar, sehingga dibutuhkan sebuah algoritma untuk menemukan sebuah lintasan dengan *cost* terkecil (waktu transfer paling kecil) untuk transfer data dari satu *node* ke *node* lain.

Pada makalah ini, akan disimulasikan penggunaan algoritma Dijkstra, salah satu algoritma untuk mencari path terpendek dalam satu graf, dalam menghitung waktu yang dibutuhkan untuk melakukan sebuah *retrieval* data dari dalam sebuah sistem basis data.

2. DASAR TEORI

2.1 Basis Data Tersebar

2.1.1. Pengertian Sistem Basis Data Tersebar

Sebuah basis data tersebar adalah sebuah basis data yang berada di bawah control dari sebuah *database management system (DBMS)* yang memiliki media penyimpanan tersebar dalam beberapa *host* di sebuah jaringan. Sebuah koleksi data dapat didistribusikan secara fisik ke beberapa *host* yang berbeda. Sebuah basis data tersebar dapat bekerja bersama dengan *server* jaringan dalam internet dan jaringan sebuah perusahaan.

Untuk memastikan basis data terdistribusi selalu termutakhirkan dengan keadaan terkini, dalam sebuah DBMS basis data tersebar terdapat dua proses utama : replikasi dan duplikasi. Replikasi menggunakan perangkat lunak khusus untuk perubahan dalam basis data terdistribusi. Setiap kali teridentifikasi perubahan, replikasi akan membuat seluruh salinan dari basis data termutakhirkan dan konsisten. Proses replikasi dapat menjadi sangat kompleks dan membutuhkan waktu yang lama bergantung pada ukuran dan jumlah dari basis data terdistribusi. Proses ini juga membutuhkan waktu dan resource yang banyak. Berbeda dengan replikasi, duplikasi lebih sederhana. Duplikasi mengidentifikasi sebuah basis data sebagai *master* dan menduplikasi basis data. Proses duplikasi biasanya dijalankan beberapa jam sekali untuk menjaga keamanan data dan memastikan bahwa setiap *database* akan berisi data yang sama. Dalam

proses duplikasi, hanya perubahan pada basis data *master* yang diperbolehkan.
Selain replikasi dan duplikasi pada basis data terdistribusi, terdapat banyak teknologi desain sistem basis data tersebar. Sebagai contoh, *local autonomy*, basis data tersebar *synchronous* dan *asynchronous*. Implementasi teknologi tersebut tergantung dari kebutuhan dan sensitivitas/kerahasiaan dari data yang akan disimpan dalam basis data.

2.1.2. Arsitektur Basis Data Tersebar
Berikut akan dibahas sebuah contoh arsitektur sistem basis data tersebar dari Oracle.

2.1.2.1. Homogenous Distributed Database System

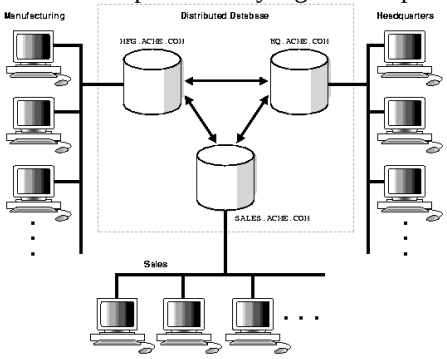
Sebuah *homogenous distributed database system* adalah sebuah jaringan dengan dua atau lebih dari basis data Oracle yang bekerja pada satu atau lebih mesin. Gambar di bawah mengilustrasikan basis data tersebar dengan tiga buah basis data : hq, mfg, dan sales.

Sebua aplikasi dapat secara bersamaan mengakses atau mengubah data dalam beberapa basis data. Sebagai contoh, sebuah query dari manufacturing *client* dalam basis data lokal mfg dapat melakukan *retrieval* data join dari tabel products dan dept dalam basis data hq.

Untuk aplikasi *client*, lokasi dan platform dari basis data terlihat transparan. Anda dapat membuat sebuah *synonyms* untuk objek pada *node* lain dalam sistem basis data terdistribusi sehingga pengguna dapat mengakses data sebagaimana menggunakan/mengakses data lokal. Sebagai contoh, jika Anda terhubung dengan basis data mfg tetapi ingin mengakses data dalam basis data hq, Anda dapat menciptakan synonym pada mfg untuk tabel dept yang memungkinkan kita melakukan *query* :

SELECT * FROM dept;

Dengan cara ini, sebuah basis data terdistribusi dapat menyajikan data dari *host* lain sebagaimana data lokal. Pengguna pada mfg tidak perlu mengetahui bahwa data yang diakses merupakan data yang terletak pada *host* lain.



Gambar 1 Arsitektur sistem basis data tersebar

2.1.2.2. Heterogeneous Distributed Database System

Dalam *heterogeneous distributed database*, paling tidak terdapat satu *host* yang memiliki *database* non-Oracle (memiliki standar *database* yang berbeda). Bagi aplikasi, heterogeneous distributed *database* diperlakukan sebagai satu satu, lokal, basis data Oracle. Basis data Oracle lokal menyembunyikan distribusi dan heterogenitas dari data.

Server basis data Oracle mengakses basis data non-Oracle menggunakan *Oracle heterogenous Service*. Selain itu, dapat digunakan sebuah *generic connectivity* untuk mengakses basis data non-Oracle.

2.1.2.3. *Client*/*Server* Distributed *Database* Architecture

Sebuah *server* basis data Oracle mengatur basis data, dan sebuah *client* adalah sebuah aplikasi yang memberikan request ke *server*. Sebuah *node* dalam basis data terdistribusi dapat bertindak sebagai *client* sekaligus *server*, tergantung pada situasi.

2.1.3. Keuntungan dan Kerugian Penggunaan Basis Data Tersebar

Keuntungan penggunaan dari basis data tersebar :
- Kemudahan manajemen data terdistribusi dengan berbagai level *transparency*
- Menambahkan *realibility* dan *availability*
- Ekspansi basis data lebih mudah
- Manajemen penempatan data, lokasi fragmen data biasanya dekat dengan departemen yang berkaitan.
- Autonomi, masing-masing departemen dapat mengontrol data yang menjadi tanggung jawabnya
- Perlindungan data yang berharga dari kemungkinan kegagalan/ kerusakan basis data dalam sebuah *node*
- Kinerja lebih baik, dengan adanya paralesisasi akses disk, jumlah permintaan yang dapat dilayani basis data meningkat
- Transaksi yang lebih reliable
- Operasi yang kontinu
- Memungkinkan *Distributed Query processing* dan *Distributed Transaction management.*

Sedangkan kerugian (*overhead*) penggunaan basis data tersebar antara lain
- Kompleksitas, usaha lebih harus dilakukan oleh DBA untuk mengatur transparency dari sistem basis data tersebar
- Pembangunan sistem basis data tersebar membutuhkan *cost* yang mahal
- Harus ada penjagaan keamanan dari masing-masing fragmen data dalam sistem basis data tersebar
- Belum adanya standar untuk melakukan konversi basis data terpusat menjadi basis data tersebar

- Desain basis data menjadi lebih kompleks.
- Dibutuhkan perangkat lunak tambahan
- Dibutuhkan sistem operasi yang mendukung pengolahan resource yang terdistribusi

2.2 Algoritma Pencarian Lintasan Terpendek Dijkstra

Algoritma Dijkstra dipublikasikan oleh seorang ilmuwan matematika Belanda, Edsger Dijkstra pada tahun 1956. Dari suatu titik, algoritma Dijkstra dapat menemukan *cost* minimal yang dibutuhkan untuk menuju seluruh titik lain dalam graf. Langkah-langkah dalam algoritma Dijkstra adalah sebagai berikut :

1. *Assign* setiap *node* sebuah nilai jarak (*cost*). Isikan 0 untuk *node* awal dan tidak hingga untuk semua *node* lain.
2. Tandai semua *node* yang belum dikunjungi, jadikan *node* awal sebagai current *node*.
3. Dari current *node*, untuk masing-masing *node* yang belum dikunjungi hitung *cost* total menuju *node* lain dari *node* awal.
4. Ketika *node* sudah menghitung *cost* total untuk masing-masing tetangganya, jadikan current *node* sebagai *visited*, *node* yang sudah *visited* tidak akan dicek lagi, *cost* yang tercatat sudah minimal.
5. Jika semua *node* sudah *visited*, maka selesai. Jika belum, maka jadikan current *node* sebagai *node* un*visited* terdekat dan ulangi dari langkah 3.

Algoritma Dijkstra dapat dinyatakan sebagai strategi greedy sebagai berikut :

"Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih."

```
procedure Dijkstra(input m: matriks, a : simpul awal)
( Mencari lintasan terpendek dari simpul awal a ke semua
simpul lainnya.
Masukan: matriks ketetanggaan (m) dari graf berbobot G
dan simpul awal a
Keluaran: lintasan terpendek dari a ke semua simpul
lainnya
}

Deklarasi
  s : array[1..n] of integer
  d : array[1..n] of integer
  i : integer
```

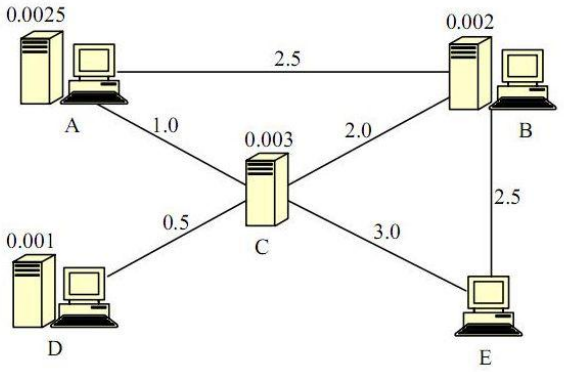
```
Algoritma:
  { Langkah 0 (inisialisasi: }
  for i←1 to n do
    s_i ← 0
    d_i ← m_ai
  endfor

  { Langkah 1: }
  s_a ← 1
  d_a ← ∞
  { Langkah 2, 3, ..., n-1: }
  for i←2 to n-1 do
        cari j  sedemikian sehingga s_j = 0 dan
   d_j = min{d_1, d_2, ..., d_n}
        s_j ← 1  { simpul j sudah terpilih }
        perbarui d_i, untuk i = 1, 2, 3, …, n dengan:
     d_i(baru)=min{d_i(lama), d_j + m_ji }
  endfor
```

Dengan kompleksitas algoritma: $O(n^2)$.

3. ANALISIS

Untuk analisis, akan digunakan sebuah simulasi sistem basis data tersebar yang dirancang sebagai berikut :
Basis data tersebar terdiri dari *node-node* yang dapat hanya berperan sebagai *client*, *server*, atau keduanya sekaligus. Pada gambar di bawah C hanya bertindak sebagai *server*, E hanya sebagai *client*, sedangkan A, B, D sebagai *server* sekaligus *client*



Gambar 2 *Node* dalam sistem basis data tersebar

Pada *node* yang memiliki peran sebagai *server* tersimpan basisdata dalam bentuk tabel yang dapat diakses dari *node* manapun dan masing-masing *server* memiliki CPU *time* (dalam satuan waktu detik), yaitu waktu yang dibutuhkan oleh untuk membaca data dari dalam basis data yang tersimpan dalam *server* agar siap untuk dikirimkan. Satu *node* terhubung dengan *node* yang lain menggunakan koneksi yang memiliki *bandwidth* (dalam ukuran kilo byte per second). Waktu transmisi dari satu *node* ke *node* lain dapat dihitung sebagai jumlah byte data yang dikirimkan dibagi dengan *bandwidth* (dalam satuan byte per second), dan waktu download adalah waktu transmisi dijumlahkan dengan CPU *time* dari *server*.

Basis data yang akan digunakan dalam simulasi adalah memiliki tiga relasi, yaitu Mahasiswa, MataKuliah, dan PengambilanMK. Tiap *node* dalam sistem basis data tersebar dapat berisi salah satu atau beberapa jenis tabel atau tidak menyimpan tabel sama sekali (bukan *server*).

Tabel Mahasiswa

NIM : string *Nama : string* *TglLahir : date* *KotaAsal : string* *IPK : real* *JumlahSKS : integer*

Tabel MataKuliah

KodeMK : string *NamaMK : string* *SKS : integer* *Bidang : string*

Tabel PengambilanMK

NIM : string *KodeMK : string* *Semester : integer* *ThnAkademik : string* *Nilai : char*

Kemudian tetapkan asumsi ukuran dari tiap tipe data field-field dalam tabel :

Type data	Ukuran (byte)
char	1
integer	2
real	3
string	sejumlah karakter dalam string
date	6

Masukan simulasi adalah sebuah file teks yang di dalamnya terdapat keterangan masing-masing *node* dalam sistem basis data tersebar, antara lain :

- Tipe *node*, apakah *server*, *client* atau dapa bertindak sebagai keduanya
- CPU *time*
- Nama tabel yang ada dalam *node*
- Data dalam tabel pada *node* tersebut

Data mengenai *node* diikuti dengan data *bandwidth*

File masukan memiliki format sebagai berikut :

```
@nodes
<jumlah-lokasi> #
@@<kode-lokasi-1>
<server-Y/T> <client-Y/T> <CPU-time> #
<nama-tabel-1> #
<data-tabel-1-kolom-1> <data-tabel-1-kolom-2> <data-tabel-1-kolom-3> ... #
<data-tabel-1-kolom-1> <data-tabel-1-kolom-2> <data-tabel-1-kolom-3> ... #
...
<nama-tabel-2> #
<data-tabel-2-kolom-1> <data-tabel-2-kolom-2> <data-tabel-2-kolom-3> ... #
<data-tabel-2-kolom-1> <data-tabel-2-kolom-2> <data-tabel-2-kolom-3> ... #
...
@@<kode-lokasi-2>
<server-Y/T> <client-Y/T> <CPU-time> #
<nama-tabel-1> #
<data-tabel-1-kolom-1> <data-tabel-1-kolom-2> <data-tabel-1-kolom-3> ... #
<data-tabel-1-kolom-1> <data-tabel-1-kolom-2> <data-tabel-1-kolom-3> ... #
...
<nama-tabel-2> #
<data-tabel-2-kolom-1> <data-tabel-2-kolom-2> <data-tabel-2-kolom-3> ... #
<data-tabel-2-kolom-1> <data-tabel-2-kolom-2> <data-tabel-2-kolom-3> ... #
...
@edges
<jumlah-koneksi> #
<kode-lokasi-1> <kode-lokasi-2> <bandwidth> #
<kode-lokasi-1> <kode-lokasi-2> <bandwidth> #
...
@end
```

Untuk simulasi, akan digunakan contoh file masukan dengan nama *myinput.in*

```
@nodes
6 #
@@A
Y Y 0.0025 #
Mahasiswa  #
19960001 "Amin Badrun" 01-08-1990 Bandung 4.00 12 #
19960002 "Cici Dadan" 12-07-1990 Bandung 3.11 9 #
19960003 "Emi Fitria" 23-06-1990 Surabaya 3.00 5 #
19960004 "Gilang Hamdan" 30-05-1990 Jakarta 0.00 2 #
@@B
Y Y 0.0020 #
MataKuliah  #
IF2030 "Algoritma dan Struktur Data" 4 Pemrograman #
EL2091 "Sistem Digital" 3 Elektronika #
IF2132 "Basisdata" 3 Basisdata #
KU1071 "Pengantar Teknologi Informasi A" 2 Pemrograman #
@@C
Y T 0.0030 #
MataKuliah  #
IF2030 "Algoritma dan Struktur Data" 4 Pemrograman #
EL2091 "Sistem Digital" 3 Elektronika #
IF2132 "Basisdata" 3 Basisdata #
KU1071 "Pengantar Teknologi Informasi A" 2 Pemrograman #
PengambilanMK  #
19960001 IF2030 1 2009-2010 A #
19960001 EL2091 1 2009-2010 A #
19960001 IF2132 2 2009-2010 A #
19960001 KU1071 1 2009-2010 A #
19960002 IF2030 1 2009-2010 B #
19960002 EL2091 1 2009-2010 A #
19960002 KU1071 1 2009-2010 C #
19960003 IF2030 1 2009-2010 B #
19960003 KU1071 1 2009-2010 B #
19960004 KU1071 1 2009-2010 E #
```

```
@@D
Y Y 0.0010 #
Mahasiswa #
19960005 "Iwan Jajang" 04-11-1990 Jakarta NULL NULL #
MataKuliah #
PengambilanMK #
@@E
T Y 0.0000 #
@edges
6 #
B A 2.00 #
C A 1.00 #
C B 2.00 #
D C 0.50 #
E B 2.50 #
E C 3.00 #
@end
```

File masukan memodelkan sistem basis data tersebar dengan lima *node* dengan nama A sampai E. *Node* A, B, dan D bertindak sebagai *server* sekaligus *client*. *Node* C hanya sebagai *server* dan *node* E hanya sebagai *client* (dapat dilihat *node* E tidak menyimpan tabel). Di bawah pendefinisian *node* tabel-tabel dalam masing-masing *node*, setelah @edges didefinisikan besarnya *bandwidth* antar *node* yang dapat digambarkan sebagai berikut :

	A	B	C	D	E
A	∞	2.00	1.00	∞	∞
B	2.00	∞	2.00	0.50	2.50
C	1.00	2.00	∞	∞	3.00
D	∞	∞	0.50	∞	∞
E	∞	2.50	3.00	∞	∞

Pengujian dilakukan dengan melakukan *retrieval* data berbagai macam tabel dari *node* A, berikut adalah hasil pengujian yang dilakukan :

- Melakukan *retrieval* data tabel mahasiswa yang berkota asal Bandung. Semua data tabel berasal dari *node* A sendiri, sehingga waktu download sama dengan CPU *time* dari *node* A

```
Tabel yang tersedia :
        1. Mahasiswa
        2. MataKuliah
        3. PengambilanMK
Masukkan nama tabel : Mahasiswa
Kolom yang tersedia :
        1. NIM
        2. Nama
        3. TglLahir
        4. KotaAsal
        5. IPK
        6. JumlahSKS
Masukkan nama kolom (pisahkan dengan spasi atau * untuk memilih semua) : *
Masukkan kondisi pencarian (gunakan "-" jika tidak ingin menggunakan kondisi pencarian) : kotaasal = Bandung
Tabel Mahasiswa
```

NIM	Nama	Tgl Lahir	Kota Asal	IPK	Jumlah SKS	Lokasi	Waktu	Jalur
19960001	Amin Badrun	01-08-1990	Bandung	4.00	12	A	0.002500	A-A
19960002	Cici Dadan	12-07-1990	Bandung	3.11	9	A	0.002500	A-A

```
Waktu total download = 0.005000
```

Gambar 3 *Retrieval* Mahasiswa kota asal Bandung

- Melakukan *retrieval* semua data tabel MataKuliah, dalam hal ini tabel MataKuliah yang diambil berasal dari *node* B dan C

```
Tabel yang tersedia :
        1. Mahasiswa
        2. MataKuliah
        3. PengambilanMK
Masukkan nama tabel : MataKuliah
Kolom yang tersedia :
        1. KodeMK
        2. NamaMK
        3. SKS
        4. Bidang
Masukkan nama kolom (pisahkan dengan spasi atau * untuk memilih semua) : *
Masukkan kondisi pencarian (gunakan "-" jika tidak ingin menggunakan kondisi pencarian) : -
Tabel Mata Kuliah
```

Kode MK	Nama MK	SKS	Bidang	Lokasi	Waktu	Jalur
IF2030	Algoritma dan Struktur Data	4	Pemrograman	B	0.024500	A-B
EL2091	Sistem Digital	3	Elektronika	B	0.018000	A-B
IF2132	Basisdata	3	Basisdata	B	0.014500	A-B
KU1071	Pengantar Teknologi Informasi A	2	Pemrograman	B	0.026500	A-B
IF2030	Algoritma dan Struktur Data	4	Pemrograman	C	0.048000	A-C
EL2091	Sistem Digital	3	Elektronika	C	0.035000	A-C
IF2132	Basisdata	3	Basisdata	C	0.028000	A-C
KU1071	Pengantar Teknologi Informasi A	2	Pemrograman	C	0.052000	A-C

```
Waktu total download = 0.246500
```

Gambar 4 *Retrieval* semua tabel MataKuliah

- Pengujian *retrieval* tabel PengambilanMK, semuanya dari *node* C

```
Tabel yang tersedia :
        1. Mahasiswa
        2. MataKuliah
        3. PengambilanMK
Masukkan nama tabel : pengambilanMK
Kolom yang tersedia :
        1. NIM
        2. KodeMK
        3. Semester
        4. ThnAkademik
        5. Nilai
Masukkan nama kolom (pisahkan dengan spasi atau * untuk memilih semua) : *
Masukkan kondisi pencarian (gunakan "-" jika tidak ingin menggunakan kondisi pencarian) : -
Tabel Pengambilan Mata Kuliah
```

NIM	Kode MK	Semester	Thn Akademik	Nilai	Lokasi	Waktu	Jalur
19960001	IF2030	1	2009-2010	A	C	0.028000	A-C
19960001	EL2091	1	2009-2010	A	C	0.028000	A-C
19960001	IF2132	2	2009-2010	A	C	0.028000	A-C
19960001	KU1071	1	2009-2010	A	C	0.028000	A-C
19960002	IF2030	1	2009-2010	B	C	0.028000	A-C
19960002	EL2091	1	2009-2010	A	C	0.028000	A-C
19960002	KU1071	1	2009-2010	C	C	0.028000	A-C
19960003	IF2030	1	2009-2010	B	C	0.028000	A-C
19960003	KU1071	1	2009-2010	B	C	0.028000	A-C
19960004	KU1071	1	2009-2010	E	C	0.028000	A-C

```
Waktu total download = 0.280000
```

Gambar 5 *Retrieval* data PengambilanMK

- *Retrieval* data tabel Mahasiswa yang berasal dari Jakarta. Tabel didownload dari *node* A dan D. Data Mahasiswa dari *node* D melalui jalur D → C → A

```
Tabel yang tersedia :
        1. Mahasiswa
        2. MataKuliah
        3. PengambilanMK
Masukkan nama tabel : Mahasiswa
Kolom yang tersedia :
        1. NIM
        2. Nama
        3. TglLahir
        4. KotaAsal
        5. IPK
        6. JumlahSKS
Masukkan nama kolom (pisahkan dengan spasi atau * untuk memilih semua) : *
Masukkan kondisi pencarian (gunakan "-" jika tidak ingin menggunakan kondisi pencarian) : kotaasal = Jakarta
Tabel Mahasiswa
```

NIM	Nama	Tgl Lahir	Kota Asal	IPK	Jumlah SKS	Lokasi	Waktu	Jalur
19960004	Gilang Hamdan	30-05-1990	Jakarta	0.00	2	A	0.002500	A-A
19960005	Iwan Jajang	04-11-1990	Jakarta	-	-	D	0.097000	A-C-D

```
Waktu total download = 0.099500
```

Gambar 6 *Retrieval* data Mahasiswa kota asal Jakarta

4. KESIMPULAN

Dari pengkajian mengenai algoritma Dijkstra dan penerapannya dalam sistem basis data tersebar dan simulasi didapatkan beberapa kesimpulan :

1. Algoritma Dijkstra terbukti dapat diimplementasikan untuk pencarian jalur terpendek untuk *retrieval* data dalam basis data tersebar.
2. Penerapan Algoritma Dijkstra dalam menentukan jalur terpendek transmisi data antar *node* dalam sistem basis data tersebar tidak menimbulkan *overhead* yang besar, karena penghitungan lintasan terpendek dari satu *node* ke *node* yang

lain hanya membutuhkan satu kali penghitungan yang hasilnya dapat dimanfaatkan berkali-kali selama keadaan dari jaringan tidak berubah. Jika ada perubahan jaringan, maka perlu ada kalkulasi ulang jalur terpendek dari satu *node* ke seluruh *node* lainnya dan inilah kelemahan penggunaan dari algoritma Dijkstra, yaitu kurang adaptif terhadap perubahan yang terjadi dalam jaringan.

5.REFERENSI

[1] Wikipedia (2010)
http://en.wikipedia.org/wiki/Distributed_*database*
(4 Desember 2010, pukul 10:28)

[2] Oracle (2010)
http://download.oracle.com/docs/cd/B10501_01/*server*.920/a96521/ds_concepts.htm#20409
(4 Desember 2010, pukul 10:44)

[3] Bahan Kuliah ke-4 Strategi Algoritmik
http://www.informatika.org/~rinaldi/Stmik/2006-2007/Algoritma%20Greedy%20(bagian%202).pdf
(4 Desember 2010, pukul 10:59)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2010

Muqtafi Akhmad
(13508059)