PENCARIAN RUTE TERCEPAT MEMANFAATKAN TEKNOLOGI *SHORT MESSAGE SERVICE* (SMS) STUDI KASUS KOTA SURABAYA

TUGAS AKHIR

Disusun Oleh :

Mas'ud Ulum
NRP. 5100 100 066

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

PENCARIAN RUTE TERCEPAT MEMANFAATKAN TEKNOLOGI *SHORT MESSAGE SERVICE* (SMS) STUDI KASUS KOTA SURABAYA

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Komputer

Pada

Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Ir. F.X. Arunanto, M.Sc	Faizal Johan A, S.Kom
NIP : 131 285 253	NIP : 132 300 414

SURABAYA

PEBRUARI 2005

ABSTRAK

Software-software untuk pencarian rute biasanya hanya dapat diakses menggunakan PC (Personal Computer) atau menggunakan alat GPS (Global Positioning System) jika ingin mengaksesnya di jalanan dan alat tersebut mempunyai harga yang mahal. Biasanya untuk mencari rute, para pengguna berada di jalan/luar ruangan dan jarang yang memiliki alat GPS. Dengan menggabungkan PC dan mobile communication, pencarian rute dapat diimplementasikan dengan memanfaatkan teknologi SMS. Teknologi tersebut dipilih karena hampir semua mobile communication mempunyai fitur SMS dan ketika pengguna berada di jalan, mereka tidak akan kesulitan mencari rute, hanya cukup dengan mengirimkan asal dan tujuan serta parameter-parameter yang lain, maka aplikasi akan mengirimkan hasil pencariannya.

Algoritma yang digunakan dalam pencarian rute tersebut adalah algoritma Dijkstra. Pencarian rute dibagi menjadi dua kategori yaitu pencarian rute menggunakan kendaraan pribadi dan angkutan umum, studi kasus yang diambil adalah peta Surabaya. Proses pembuatan aplikasi dibagi menjadi tiga bagian yaitu proses konversi data, proses pencarian rute dan proses pengiriman dan penerimaaan SMS. Hasil pencarian rute untuk kendaraan pribadi yaitu berupa keterangan arah belok atau tidaknya pada jalan-jalan dan adanya pedoman-pedoman pada setiap jalan berbelok, dan hasil pencarian untuk angkutan umum berupa keterangan dimana harus naik angkutan, dimana harus ganti angkutan dan dimana harus turun.

Uji coba yang dilakukan adalah uji coba untuk konversi data shapefile(SHP) menjadi data graph, uji coba pencarian rute dengan tiga skenario, skenario pertama uji coba untuk kendaraan pribadi, skenario kedua untuk angkutan umum dan skenario ketiga pembuktian rute tercepat dengan memberikan tingkat kemacetan pada jalan-jalan tertentu. Uji coba selanjutnya yaitu uji coba pengiriman dan penerimaan SMS. Evaluasi uji coba tersebut yaitu Pengkonversian menjadi data graph tersebut dapat digunakan untuk pencarian rute tercepat dengan mengimplementasikan algoritma Dijkstra.

Kata Kunci : *Algoritma Dijkstra, GPS, SMS,shapefile, graph*

KATA PENGANTAR

Syukur Alhamdulillah, atas berkat rahmat dan hanya kuasa-Nyalah penulis dapat menyelesaikan Tugas Akhir yang berjudul:

PENCARIAN RUTE TERCEPAT MEMANFAATKAN TEKNOLOGI *SHORT MESSAGE SERVICE* (SMS) STUDI KASUS KOTA SURABAYA

Tugas Akhir ini dibuat guna memenuhi persyaratan akademik dalam rangka ujian akhir bagi mahasiswa Strata 1 (S1) Jurusan Teknik Informatika , Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam penyusunan Tugas Akhir ini, Penulis berusaha untuk menerapkan ilmu yang telah didapat selama menjalani perkuliahan dengan tidak terlepas dari petunjuk, bimbingan, bantuan, dan dukungan berbagai pihak.

Dengan tidak lupa akan kodratnya sebagai manusia, Penulis menyadari bahwa dalam karya Tugas Akhir ini masih mengandung kekurangan sehingga dengan segala kerendahan hati Penulis masih dan insya Allah akan tetap terus masih mengharapkan saran serta kritik yang membangun dari rekan-rekan pembaca.

Surabaya, Pebruari 2005

Penulis

UCAPAN TERIMA KASIH

Dengan tak bosannya mengucapkan syukur Alhamdulillah kepada **Allah SWT**, yang telah memberi terlalu banyak dari yang layak penulis terima, telah penulis curi sekelumit misteri-Mu, dan penulis kembalikan dalam bentuk buku ini. Di kesempatan ini, Penulis hendak menyampaikan rasa penghormatan yang setingginya serta rasa terima kasih kepada pihak-pihak yang telah memberi bantuan baik itu berupa moril maupun material dan langsung maupun tidak langsung kepada:

1. Terima kasih kepada bapak, ibu, adik, kakak, kakek, nenek, paman, bibi, dan keluarga yang telah memberi doa dan dukungan yang sangat besar kepada penulis selama ini hingga dapat menyelesaikan tugas akhir.
2. Bapak Dr. Ir. Arif Djunaidi, selaku Dekan Fakultas Teknologi Informasi, semoga FTIF dapat terus maju dan berkembang.
3. Bapak Yudhi Purwananto S.Kom, M.Kom, selaku Ketua Jurusan. Semoga Teknik Informatika semakin maju dan berkembang di bawah kepemimpinan beliau.
4. Bapak Ir. F.X. Arunanto. M.Sc, selaku dosen pembimbing pertama yang telah memberi banyak masukan bagi penulis juga bimbingan dalam menyelesaikan tugas akhir.
5. Bapak Faizal Johan A S.Kom, selaku dosen pembimbing kedua yang telah memberikan ide bagi pengerjaan tugas akhir serta memberikan banyak bimbingan dan masukan bagi tugas akhir ini.
6. Bapak Ir. Suhadi Lili, selaku dosen wali penulis yang telah membimbing penulis selama menjalani perkuliahan di jurusan ini.
7. Semua dosen jurusan Teknik Informatika yang telah membagikan ilmunya selama penulis menjalani masa perkuliahan.

8. Seluruh staf dan karyawan jurusan Teknik Informatika yang selalu siap membantu penulis dalam hal administrasi perkuliahan.
9. Rekan-rekan di TRASPAC Teknologi Informasi Pak Purnadi, Mas Anib, Mas Fauzi, Mas Ryan, Mas Ndaru, Mas Alimin, Mas Muda, Mas Eko yang selalu memberi dukungan kepada penulis, terima kasih atas semuanya.
10. Mas Agung99 yang bersedia mengajari penulis tentang pencarian rute serta rela memberikan data peta Surabaya, terima kasih Mas.
11. Rizqi00 yang rela meminjamkan handphone Siemens MC60, printer XNU beserta tinta dan kertas A4 nya, terima kasih *Pen*.
12. Tri00, Deka00, Gedhe00, Okhi00, Sinan00 dan mbak Ulfa00 yang setia menemani penulis bersama-sama mengerjakan tugas akhir di Laboratorium Perangkat Lunak.
13. Teman-teman PPL terutama Nindi00 yang rela *coding* dibawah tekanan dan terima kasih juga untuk buku struktur datanya.
14. Teman-teman seangkatan yang bersama-sama mengerjakan tugas akhir Adit00, Puspa00,Arie00 dan masih banyak lagi, terima kasih atas dukungan ketika bersama-sama mengerjakan tugas akhir.
15. Bawenang00 yang telah 'mengajari' penulis betapa pentingnya bahasa inggris, Tyarso00 yang rela meminjamkan sepatu ketika sidang proposal tugas akhir dan Gunawan00 yang rela meminjamkan ikat pinggang ketika sidang proposal tugas akhir.
16. Teman-teman Darul Ulum di Surabaya *Menying, Bajoel, Pukon , Tombil, Botol, Preti*,Ridwan dan tidak lupa Adin '*nDower*' Suhada yang meminjamkan peta Surabaya.
17. Teman-teman c10 lainnya yang selalu memberikan doa dan dukungan kepada penulis. Tetap kompak selalu.
18. Taqi02, Yunan00 dan Junaidi selaku administrator Lab RPL yang sudah banyak membantu penulis dengan menyediakan fasilitas laboratorium dalam menyelesaikan tugas akhirnya.
19. Teman-teman lain yang tidak dapat penulis sebutkan satu persatu dalam halaman ini.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang dan tujuan dari pembuatan tugas akhir , menentukan permasalahan yang akan dipecahkan beserta batasan dari penyelesaiannya. Kemudian metodologi yang digunakan dalam penyelesaian permasalahan serta sistematika dalam penulisannya.

1.1 LATAR BELAKANG

Saat ini teknologi *Short Message Service* (SMS) menjadi sebuah teknologi alternatif, sudah banyak aplikasi-aplikasi menggunakan SMS sebagai media komunikasinya. Misalnya, SMS untuk jajak pendapat (voting), SMS kuis-kuis yang sekarang sedang ramai, atau dikombinasikan dengan *hardware* untuk mengontrol sebuah sistem.

Dengan perkembangan teknologi yang demikian pesat, teknologi juga harus dapat memberikan solusi serta kontribusi dalam bidang sistem informasi geografis. Oleh karena itu penulis mencoba mengembangkan sistem dengan mengkombinasikan sistem informasi geografis dan teknologi SMS. Karena rute merupakan alat penunjuk yang sangat penting untuk mencari lokasi dan saat ini banyak software-software untuk pencarian rute tetapi software tersebut beroperasi di PC (*Personal Computer*) atau menggunakan alat GPS (*Global Positioning*

System), sedangkan kebiasaan seorang pencari rute tidak berada di depan komputer atau dalam perjalanan.

Dengan dasar tersebut, SMS lah yang sesuai dijadikan sebagai media untuk memberikan informasi dalam pencarian rute, disamping biaya SMS murah, juga hampir semua *mobile communication* mempunyai fitur SMS, sehingga SMS rute ini bisa dijadikan alternatif daripada harus membuka berlembar-lembar peta yang digunakan dalam mencari rute.

1.2 Tujuan Pembuatan Tugas Akhir

Tujuan pembuatan tugas akhir ini adalah merancang dan membuat Aplikasi dengan memanfaatkan teknologi SMS untuk pencarian rute tercepat dengan studi kasus kota Surabaya.

1.3 Permasalahan

Berdasarkan latar belakang yang telah diuraikan sebelumnya, permasalahan yang akan diselesaikan dalam tugas akhir ini adalah sebagai berikut:

1. Pencarian rute tercepat pada studi kasus kota Surabaya dengan menggunakan kendaraan pribadi atau angkutan umum.
2. Implementasi algoritma Dijkstra sebagai algoritma yang digunakan untuk mencari rute tercepat.
3. Dalam pencarian rute memperhatikan
    - Arah jalan, yaitu jalan satu arah dan jalan dua arah.

- Jalan diperbolehkan belok pada sebuah persimpangan.
- Tingkat kemacetan lalu lintas pada jalan tertentu dan jam tertentu.

1.4 Batasan Permasalahan

Dari permasalahan-permasalahan di atas, maka batasan dalam tugas akhir ini adalah:

1. Dalam Pencarian rute tidak memperhatikan jalan kecil atau jalan raya
2. Pencarian Rute angkutan umum hanya dibatasi untuk rute lin/mikrolet dan bus kota yang mempunyai waktu operasi 24 jam.
3. Data peta yang akan diolah adalah data atribut dari peta geografis Shapefile (SHP) dari ESRI™
4. Rute yang dapat dicari adalah kombinasi antara jalan dan lokasi, yaitu
   a. Pencarian rute dari jalan tertentu menuju ke lokasi tertentu.
   b. Pencarian rute dari jalan tertentu menuju ke jalan tertentu
   c. Pencarian rute dari lokasi tertentu menuju ke lokasi tertentu
   d. Pencarian rute dari lokasi tertentu menuju ke jalan tertentu.

1.5 Metodologi Tugas Akhir

Metodologi yang digunakan dalam penyusunan tugas akhir ini diantaranya:

1. Studi literatur

   Sebagai langkah awal, pada tahap ini dilakukan pengumpulan data peta Surabaya mulai dari data digital peta Surabaya, rute-rute untuk angkutan umum dan tempat-tempat penting di Surabaya, serta mempelajari konsep dan teori tentang sistem informasi geografis dan algoritma Dijkstra.

2. Perumusan masalah dan perumusan penyelesaiannya

   Langkah berikutnya adalah pendefinisian masalah, batasan-batasan, serta langkah-langkah penyelesaian yang mengarah pada perangkat lunak yang akan dikembangkan.

3. Perancangan perangkat lunak

   Langkah yang dilakukan pada tahap ini adalah melakukan perancangan basis data, struktur data, proses dan antar muka.

4. Pembuatan perangkat lunak

   Pada tahap ini dilakukan proses implementasi dari perancangan perangkat lunak yang telah dibuat pada tahap sebelumnya.

5. Pengujian dan revisi program

   Pada tahap ini dilakukan evaluasi dan perbaikan dari program yang telah dibuat.

6. Penulisan naskah Tugas Akhir

   Pada tahap akhir dari serangkaian metodologi ini, dilakukan penulisan dokumentasi dari tahapan konsep, perancangan, sampai tahap akhir, yaitu penerapannya.

1.6 SISTEMATIKA PENULISAN

Metodologi yang digunakan dalam Penyusunan Tugas akhir ini adalah sebagai berikut :

1. **BAB I Pendahuluan**

   Bab ini menjelaskan mengenai latar belakang permasalahan, batasan masalah, tujuan serta metodologi penyusunan tugas akhir ini.

2. **BAB II Dasar Teori**

   Bab ini membahas mengenai dasar teori yang digunakan pada tugas akhir ini diantaranya mengenai algoritma yang digunakan yaitu algoritma Dijkstra, pemodelan jalan, sumber data dan dasar-dasar teknologi SMS beserta cara-cara pengoperasian SMS melalui komputer.

3. **BAB III Perancangan Perangkat Lunak**

   Bab ini membahas mengenai perancangan sistem untuk aplikasi pencarian rute tercepat meliputi perancangan data, perancangan proses dan perancangan antar muka.

4. **BAB IV Implementasi Perangkat Lunak.**

   Bab ini membahas mengenai implementasi dari perancangan yang telah dibuat pada bab III meliputi implementasi struktur data, implementasi proses dan implementasi antar muka.

5. **BAB V Uji Coba dan Evaluasi**

   Bab ini membahas mengenai uji coba dan evaluasi yang dilakukan terhadap perangkat lunak yang telah dibangun.

6. **BAB VI Penutup**

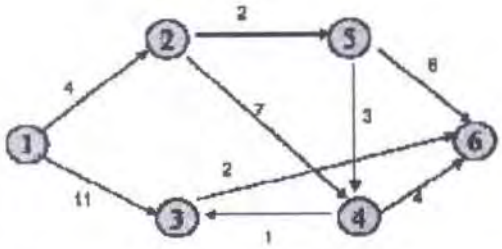
   Bab ini menjelaskan mengenai kesimpulan dan saran dari keseluruhan Tugas Akhir ini.

BAB II
DASAR TEORI

2.1 TEORI DASAR *GRAPH*

Model jaringan fisik seperti jalan, rel kereta api atau rute pesawat terbang, dapat direpresentasikan sebagai *graph* garis yang dibentuk dari tautan yang merepresentasikan *arc/edge* (segmen garis/jalan) serta *node* (interseksi jalan) yang merepresentasikan koneksi antar segmen garis tersebut . *Graph* berarah G=(N,A) terdiri dari sekumpulan *node*, N , dan sekumpulan *edge*, A , dengan nilai-nilai yang berasosiasi dengan setiap *node*. Nilai-nilai yang berasosiasi itu adalah jumlah *node*, jumlah *edge*, dan panjang dari *edge* yang menghubungkan antara *node* i dan j yang dinotasikan sebagai d(i,j). *Node* adalah titik interseksi antara dua garis *edge* atau lebih.

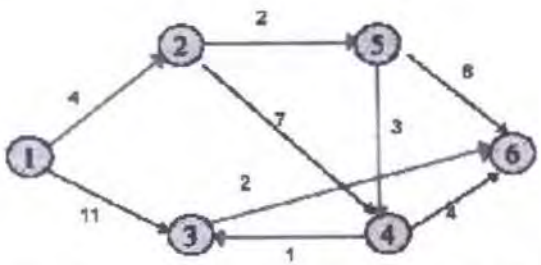
Gambar 2.1 menunjukkan *graph* berarah G yang terdiri dari enam *node* 1,2,3,4,5 dan *node* 6, serta sembilan buah *edge* berarah (1,2), (1,3), (2,5), (2,4), (3,6), (4,3), (4,6), (5,4) dan (5,6). Sebuah *graph* dikatakan berarah jika *node-node* anggota *graph* tersebut tersusun dengan mempunyai urutan tertentu. Tanda panah diantara *node-node* didalam *graph* adalah yang disebut dengan *edge*. Masing-masing *edge* dari gambar berikut mempunyai nilai bobot tersendiri yang dilambangkan dengan nilai angka. 4, 11, 2, 7, 2, 1, 4, 3 dan 8. *Node* yang berada dibagian belakang tanda panah dianggap sebagai awal *node* dan *node* yang berada dibagian depan *node* disebut dengan *node* tujuan.



Gambar II-1 Graph berarah dengan 6 buah node dan 9 buah edge

Derajat konektivitas (*degree*) suatu *node* i dihitung dari banyaknya *edge* yang terhubung dengan *node* i tersebut. *Node* i dianggap *adjacent* dengan *node* j jika terdapat *edge* dari *node* j menuju *node* i. Jika *node* i *adjancent* terhadap *node* j maka *node* i dianggap sebagai *successor node* j, dan *node* j adalah *predecessor* dari *node* i.

Proses penghitungan rute terpendek adalah proses mencari jarak terpendek atau biaya terkecil, d(i) , suatu rute dari *node* awal *s* ke *node* tujuan i , didalam *network G*. Nilai d(i) merupakan nilai jarak total dari suatu rute terpendek. Pada *gambar II*-2 rute terpendek dari *node* 1 ke *node* 6 melalui , *node* 2, 5, 4, dan *node* 3 direpresentasikan dengan garis merah dengan jarak total d(i) adalah 12.

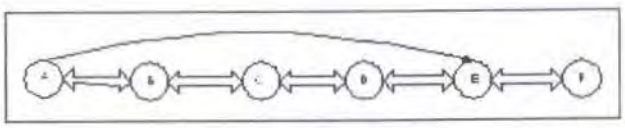


Gambar II-2 Rute terpendek dari node 1 ke node 6

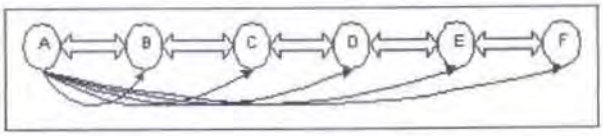
Ada empat versi proses penghitungan rute terpendek yaitu :

- Penghitungan rute terpendek antara satu *node* awal *s* yang spesifik dengan satu *node* tujuan *i* yang juga spesifik (*one-to-one shortest path*)
- Penghitungan rute terpendek antara satu *node* awal *s* yang spesifik dengan semua *node* lain (*one-to-all shortest path*)
- Penghitungan rute terpendek antara semua *node* pada *network* (*all pairs shortest path*)
- Penghitungan rute terpendek ke satu *node* tujuan i yang spesifik dari semua *node* yang ada pada *network* (*all-to-one shortest path*). Penghitungan rute terpendek ini mempunyai kesamaan proses pencarian dengan *one-to-all shortest path*

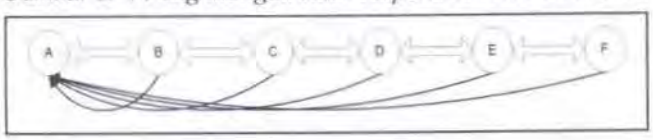
Berikut adalah gambar versi penghitungan rute terpendek pada *graph* berarah G yang terdiri dari *node* A, B, C, D, E dan F yang saling berhubungan satu dengan yang lain :



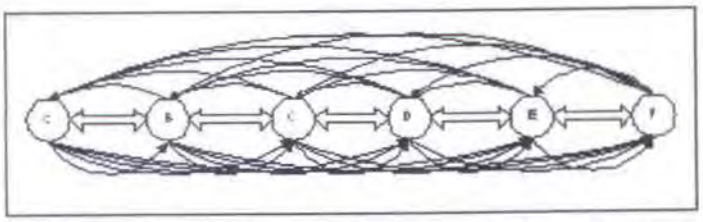
Gambar II-3 Penghitungan Rute Terpendek Versi One-to-one



Gambar II-4 Penghitungan Rute Terpendek Versi One-to-all



Gambar II-5 Penghitungan Rute Terpendek Versi All-to-one



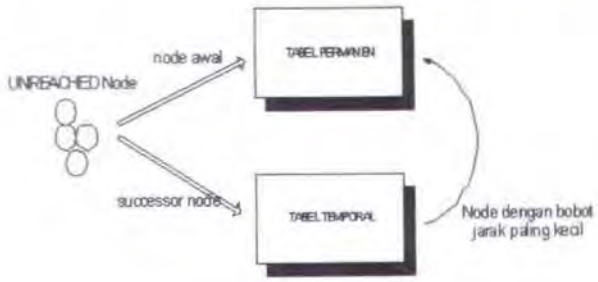
Gambar II-6 Penghitungan Rute Terpendek Versi All-pairs

Terdapat beberapa asumsi pada penghitungan rute terpendek yaitu :

- Bobot *edge* mempunyai nilai integer
- Semua *node* pada network saling terhubung antara satu dengan yang lain (jika tidak terhubung maka akan ditambahkan *edge* virtual dengan nilai bobot yang sangat besar)
- Jaringan yang akan dihitung tidak mengandung nilai bobot negatif
- *Edge* yang menghubungkan antar *node* mempunyai arah

2.2 ALGORITMA DIJKSTRA

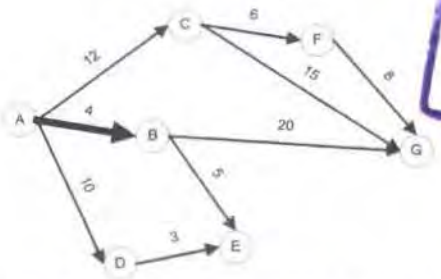
Algoritma Dijkstra pertama kali dikembangkan oleh E.W. Dijksta ["A Note on Two Problems in Connexion with Graphs, " Numeriske Mathematik, 1 (1959)]. Pada perkembangannya algoritma ini menggunakan struktur data yang berbeda-beda dan menghasilkan algoritma turunan yang bermacam-macam,seperti menggunakan *buckets*, *double buckets*, dan *approximate buckets*.



Gambar II-7 Ilustrasi dari Algortima Dijkstra

Algoritma Dijkstra menggunakan adjancency list untuk merepresentasikan sebuah network. Secara garis besar Algortima Dijkstra membagi semua *node* menjadi dua dan memasukkan kedalam tabel yang berbeda yaitu tabel permanen dan tabel temporal. Tabel permanen berisi *node* awal dan *node-node* yang telah melalui proses pemeriksaan dan labelnya telah diubah dari temporal menjadi permanen. Sedangkan tabel temporal berisi *node-node* yang berhubungan dengan *node-node* yang ada pada tabel permanen dan mempunyai status label temporal, *node-node* ini telah melalui proses pemberian label dari unreached menjadi temporal. *Gambar II*-8 adalah ilustrasi dari jalannya Algoritma Dijkstra :

Pemilihan rule Algoritma Dijkstra dilakukan dengan *Best First Seedgeh* (BFS). Pada proses BFS, sebuah rute akan dihitung jaraknya dari *node* awal ke *node* lain dalam suatu network, kemudian rute-rute ini akan dibandingkan dan rute dengan jarak yang paling pendek akan dipilih sebagai rute terpendek. Berikut adalah gambar ilustrasi pemilihan rule pada Algortima Dijkstra dengan menggunakan metode BFS :



Gambar II-8 Ilustrasi pemilihan rule dengan menggunakan BFS

Pada *gambar II*-8 *node* A berhubungan dengan *node* B,C dan D yang dilambangkan dengan tanda panah yang masing-masing mempunyai bobot tersendiri. Jika *node* A dianggap sebagai *node* awal dan *node* G sebagai *node* tujuan maka dengan menggunakan metode BFS *node* A akan memilih *node* B

sebagai jalur yang akan dilalui, karena *node* B mempunyai bobot paling kecil dibandingkan dengan bobot *node* lain. Pada algoritma Dijkstra proses pemilihan ini akan dilakukan secara iterasi pada tabel temporal sampai *node* tujuan tercapai.

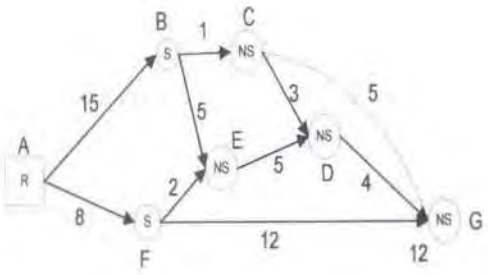
Berikut adalah proses yang terjadi pada setiap tahap penghitungan rute terpendek pada *graph* G = (N,A) berarah menggunakan algoritma Dijkstra :

- Inisialisasi variabel yang akan digunakan pada perhitungan rute terpendek
- Memberikan nilai jarak awal untuk setiap *node*, *node* selain *node* awal *s* label jaraknya diberi nilai tak terbatas, *d(i)* =infinity dan label jarak *no+de* awal, *d(s)* akan diberi nilai 0.
- Memeriksa jarak *node* j, *d(j)* yang berhubungan (*successor*)dengan *node* i pada tabel temporal
- Untuk *node* dengan nilai jarak *d(j)* yang paling kecil akan disimpan nilai jaraknya pada tabel permanen.
- Setiap *node* dengan nilai jarak d(j) terkecil yang dipilih, labelnya akan diubah dari *sementara* menjadi *permanen S(j) = permanent* dan label *parent*, *p(j)* diubah menjadi i.
- Proses ini akan terus berlangsung secara iterasi sampai *node* tujuan tercapai.

2.2.1 Contoh Penghitungan Rute Terpendek

Pada Algoritma Dijkstra proses penghitungan rute terpendeknya membagi *node-node* menjadi 2 tabel yaitu permanen dan temporal. Setiap *node* yang ada pada bagian tabel temporal akan diperiksa dan diubah statusnya dari temporal menjadi permanen. Ketika sudah tidak ada lagi *node* pada tabel temporal maka

proses penghitungan rute terpendek akan berhenti. *Gambar II*-9 adalah network K yang akan dicari rute terpendek dari *node* A sebagi *node* awal ke *node* G sebagai *node* tujuan.



Gambar II-9 Graph K yang Akan Dihitung dan Dicari Rute Terpendeknya Menggunakan Algoritma Dijkstra

Node yang labelnya R adalah *node* yang telah diberi label secara permanent, *node* dengan label S adalah *node* yang telah diberi label tetapi belum dilakukan proses pemeriksaan dan *node* dengan label NS adalah *node* yang belum melalui proses pemberian label dan proses pemeriksaan. Algoritma Dijkstra akan mencari rute mulai dari *node* awal yaitu **A.** Ketika mencapai *node* selanjutnya yang mempunyai biaya terkecil dari *node* awal maka *node* tersebut akan diberi label permanen. Berikut adalah langkah-langkah pencarian rute terpendeknya :

1. Algoritma memberi label awal = permanen dan meletakkan *node* awal pada table permanen. Selanjutnya akan melakukan proses pemeriksaan pada successor *node* awal A yaitu *node* B dan F, dan meletakkannya pada tabel temporal.

B	:15	A
F	:08	A

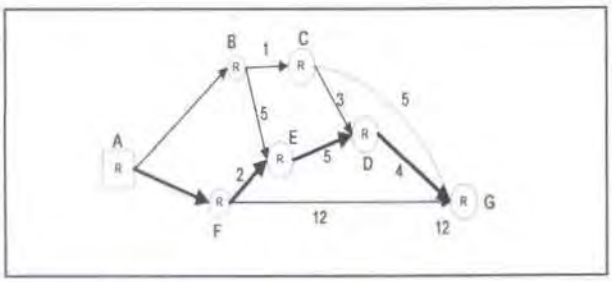
Tabel II-1 Tabel Temporal

NODE	**CUMULATIVE COST**	**PREVIOUS *NODE***
A	:00	none
B		
C		
D		
E		
F		
G		

Tabel II-2 Tabel Permanen

2. Pada tabel temporal akan dilakukan pemilihan *node* dengan bobot jarak paling kecil dan meletakkaknnya pada tabel permanen. Dalam hal ini *node* F adalah *node* yang akan dipindahkan ke tabel permanen.
3. Langkah selanjutnya adalah mencari successor node yang berada pada tabel permanen dan meletakkan pada tabel temporal.
4. Proses 2 dan 3 akan diulang secara iterasi sampai node G sebagai node tujuan tercapai.
5. Hasil rute yang didapat dari node A ke node G sebagi node tujuan adalah A, F, E, D dan G. Jarak total yang ditempuh dari node awal ke node tujuan adalah 19.

Gambar II-10 adalah gambar dan tabel hasil penghitungan rute terpendek menggunakan Algoritma Dijkstra dari *graph* K berarah.



Gambar II-10 Rute terpendek graph K hasil penghitungan menggunakan Algoritma Dijkstra

NODE	CUMULATIVE COST	PREVIOUS NODE
A	:00	none
B	:15	E
C	:16	B
D	:15	E
E	:10	F
F	:08	A
G	:19	D

Tabel II-3 Permanen

NODE	CUMULATIVE COST	PREVIOUS NODE
G	:20	F
G	:19	D
G	:21	C

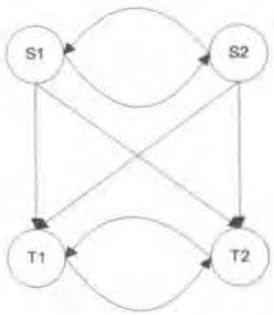
Tabel II-4 Temporal

2.2.2 Pencarian Rute Antara 2 *Edge*

Pada dasarnya, pencarian rute adalah pencarian rute dari *node* sumber menuju *node* tujuan, bukan dari *edge* sumber menuju *edge* tujuan. Untuk pencarian rute dari *edge* sumber menuju *edge* tujuan harus dimodifikasi terlebih dahulu untuk mencari *node* terdekat pada *edge* tersebut, terdapat 4 kondisi untuk pencarian rute dari *edge* menuju *edge*, yaitu :

1. *Edge* Sumber 2 Arah dan *Edge* Tujuan 2 Arah

   Jika *Edge* sumber dan *Edge* tujuan masing-masing mempunyai 2 arah, maka dilakukan 4 kali pencarian rute tercepat. Kemudian bobot terkecilnya adalah rute tercepat diantara dua *edge* tersebut, untuk lebih jelasnya pada gambar di bawah ini :

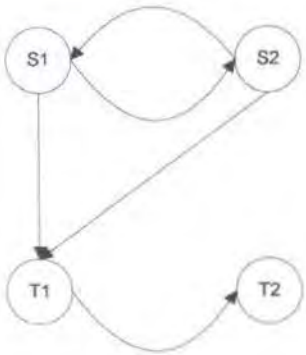


Gambar II-11 Edge Sumber 2 Arah, Edge Tujuan 2 Arah

Pada gambar diatas, maka *node* yang dicari yaitu :

- Dari *node* S1 menuju *node* T1
- Dari *node* S1 menuju *node* T2
- Dari *node* S2 menuju *node* T1
- Dari *node* S2 menuju *node* T2

2. *Edge Sumber* 2 Arah dan *Edge* Tujuan 1 Arah

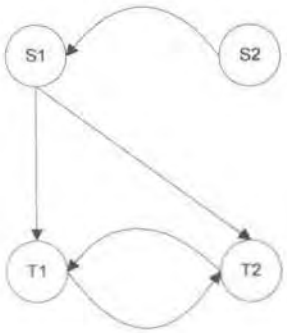


Gambar II-12 Edge Sumber 2 Arah, Edge Tujuan 1 Arah

Pada gambar diatas, maka *node* yang dicari yaitu :

- Dari *node* S1 menuju *node* T1
- Dari *node* S2 menuju *node* T1

3. *Edge* Sumber 1 Arah dan *Edge* Tujuan 2 Arah

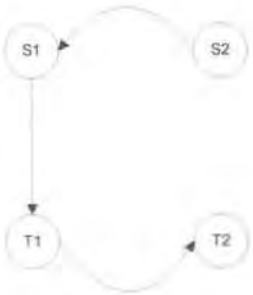


Gambar II-13 Edge Sumber 1 Arah, Edge Tujuan 2 Arah

Pada gambar diatas, maka *node* yang dicari yaitu :

- Dari *node* S1 menuju *node* T1
- Dari *node* S1 menuju *node* T2

4. *Edge* Sumber 1 Arah dan *Edge* Tujuan 1 Arah



Gambar II-14 Edge Sumber 1 Arah, Edge Tujuan 1 Arah

Pada gambar diatas, maka *node* yang dicari yaitu :

- Dari *node* S1 menuju *node* T1

Setelah salah satu pencarian di atas selesai dilakukan, maka hal selanjutnya yaitu menggabungkan bobot dari *node* yang terhubung untuk menyempurnakan hasil pencarian. Cara penggabungannya adalah sebagai berikut :

1. Jika bobot terkecil adalah Dari S1 menuju T1, maka hasil pencarian yang terjadi, *node* paling depan adalah *node* S2 dan *node* paling akhir yaitu *node* T2. pada saat tersebut bobot antara S1 dan S2 ataupun T1 dan T2 juga dijumlahkan untuk menghasilkan bobot terkecil yang baru.
2. Jika bobot terkecil adalah Dari S1 menuju T2, maka hasil pencarian yang terjadi, *node* paling depan adalah *node* S2 dan *node* paling akhir yaitu *node* T1. pada saat tersebut bobot antara S1 dan S2 ataupun T2 dan T1 juga dijumlahkan untuk menghasilkan bobot terkecil yang baru.
3. Jika bobot terkecil adalah Dari S2 menuju T1, maka hasil pencarian yang terjadi, *node* paling depan adalah *node* S1 dan *node* paling akhir yaitu *node* T2. pada saat tersebut bobot antara S2 dan S1 ataupun T1 dan T2 juga dijumlahkan untuk menghasilkan bobot terkecil yang baru.
4. Jika bobot terkecil adalah Dari S2 menuju T2, maka hasil pencarian yang terjadi, *node* paling depan adalah *node* S1 dan *node* paling akhir yaitu *node* T1. pada saat tersebut bobot antara S2 dan S1 ataupun T2 dan T1 juga dijumlahkan untuk menghasilkan bobot terkecil yang baru.

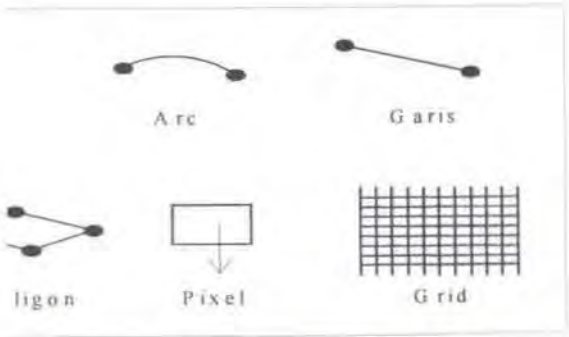
2.3 SUMBER DATA

Agar dapat berfungsi dan memberikan informasi hasil, suatu algoritma penghitungan rute terpendek memerlukan data masukan. Data masukkan yang

diperlukan berupa data jaringan jalan yang direpresentasikan sebagai *graph*. Data masukkan ini mempunyai jenis data sebagai berikut :

1. **Data spasial.** Data yang menunjukkan lokasi dan bentuk permukaan bumi yang terdiri dari titik, garis, dan poligon.
2. **Data atribut.** Data yang menggambarkan karakteristik suatu lokasi. Data atribut memberikan informasi tambahan terhadap data spasial. Contoh dari data atribut adalah nama jalan, nama simpangan, id jalan dan sebagainya, dimana data atribut ini memberikan informasi tambahan terhadap data spasial jalan.
3. **Data numerik.** Data berupa angka-angka yang menyatakan jumlah atau kuantitas.

Untuk menyusun data spasial diperlukan beberapa elemen, seperti elemen titik, garis, poligon, grid, *edge*, pixel dan lain-lain. Berikut adalah gambar elemennya :



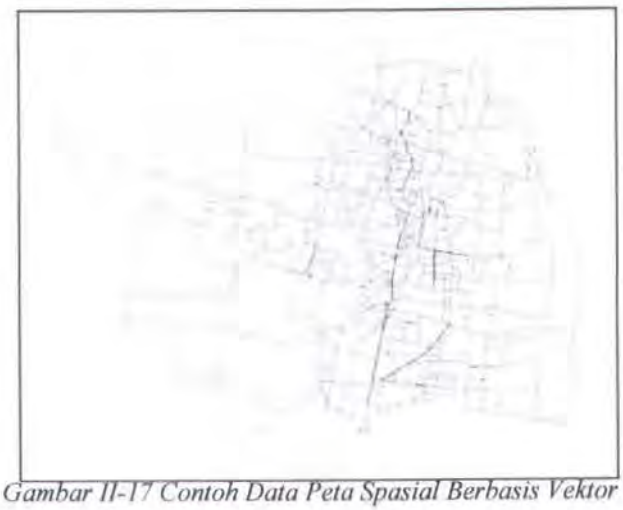
Gambar II-15 Enam Elemen Grafik Penyusun Peta

Untuk merepresentasikan dunia nyata data spatial dibagi menjadi dua yaitu data spasial berbasis raster dan data spasial berbasis vektor.Ciri dari data spasial

raster adalah sel-sel yang terbentuk mempunyai bentuk dan ukuran yang sama, dimana sel-sel tersebut mewakili sebuah fitur, sehingga peta akan terbentuk dengan pola susunan ubin, dengan ukurannya sering disebut sebagai resolusi. Model data raster dikenal sebagai Grid, dengan dua dimensi ruang yang terdiri dari kumpulan piksel. Keakuratan pemodelan dengan raster banyak bergantung pada resolusinya. Pada umumnya digunakan ukuran 30 x 30 meter untuk mendekati ukuran luas $1000m^2$ setiap selnya. Objek Titik, garis atau polygon akan diperlakukan sama pada data berbasis raster. Tiap sel file hanya menyimpan satu nilai, jadi untuk atribut yang berbeda, akan disimpan di file lain. Output grafiknya kurang memuaskan karena tampilannya tidak seperti garis-garis halus pada peta. Hal ini dapat diperbaiki dengan menambah jumlah sel, namun dapat menghasilkan file yang ukurannya besar. *Gambar II*-16 adalah contoh gambar data spasial berbasis raster :

Gambar II-16 Data Peta Spasial Berbasis Raster

Data spasial berbasis vektor ditampilkan sebagai objek titik, garis dan poligon yang mempunyai atribut dalam bentuk koordinat kartesian (X dan Y) dan algoritma perhitungannya dilihat dari titik yang berada pada sistem vektor tersebut. Posisi dari fitur berupa titik, misalnya letak kota dalam sebuah peta disimpan sebagai sebuah koordinat x dan y tunggal. Fitur garis, seperti jalan atau sungai dapat disimpan sebagai himpunan koordinat-koordinat titik. Fitur berupa poligon, misalnya dalam menggambarkan sebuah pulau, dapat disimpan sebagai sebuah himpunan koordinat kurva tertutup. Data yang disimpan adalah informasi tentang letak suatu obyek berada, topologi dan detail atribut. Pada masing-masing fitur (titik, garis atau polygon) disimpan dalam tabel yang berbeda, ditambah sebuah tabel lagi untuk menyimpan koordinat data. Jenis vektor ini lebih mudah dilakukan proses analisa penghitungan dibanding dengan model raster dan output grafiknya memuaskan karena lebih mendekati tampilan peta, namun tidak bagus untuk menggambarkan fitur-fitur yang bervariasi secara kontinyu, misalnya dalam menggambarkan jenis tanah. *Gambar II*-17 adalah contoh gambar peta berbasis vektor :



Gambar II-17 Contoh Data Peta Spasial Berbasis Vektor

Dalam melakukan analisa rute dalam model network seperti jalan dan sungai, maka akan lebih tepat bila digunakan data spatial berbasis vektor, dimana model ini dapat menangani entitas diskrit, contoh menyimpan panjang yang tepat dari sebuah *edge*, sedangkan model raster hanya akan mengira-ngirakan saja, tergantung dari resolusi grid. Data spasial yang digunakan sebagai bahan penghitungan algoritma penghitungan rute terpendek adalah peta jaringan jalan Kota Surabaya. Data spasial ini berbasis vektor yang kemudian diubah menjadi *database* berbasis teks

2.4 PEMODELAN JALAN

Suatu pemodelan jalan memerlukan aturan-aturan tertentu agar sesuai dengan kondisi jalan yang direpresentasikannya. Berikut adalah contoh aturan yang berlaku pada kondisi jalan sesungguhnya yang harus direpresentasikan pada pemodelan jalan :

2.4.1 Model Interseksi Jalan

Ketika suatu segmen garis bertemu dengan segmen garis yang lain dan saling berinterseksi, maka terdapat tiga kemungkinan dari interseksi tersebut :

1. Segmen-segmen garis tersebut benar-benar saling berinterseksi dan diperbolehkan untuk melakukan sembarang belokan dari segmen-segmen garis yang berinterseksi tersebut.
2. Segmen-segmen garis tersebut benar-benar saling berinterseksi, namun tidak diperbolehkan antar segmen garis tersebut untuk saling berhubungan, misal melakukan belokan dari segmen garis yang saling

berinterseksi. Hal ini dapat terjadi karena adanya aturan-aturan lalu lintas yang harus dipatuhi.

3. Segmen-segmen garis tersebut tidak benar-benar saling berinterseksi, hal ini karena suatu segmen garis dalam kondisi riilnya berada di atas segmen garis yang saling berinterseksi tersebut (misal dalam memodelkan jalan tol yang berada di atas jalan yang lain).

Gambar II-18 adalah contoh gambar interseksi, node 1 adalah model interseksi ketiga karena jalan tol tidak benar-benar berinterseksi dengan jalan lainnya yang berada dibawahnya. node 2 adalah model interseksi kedua, karena di perempatan dari arah node 3 menuju ke node 2 tidak diperbolehkan belok kanan. Sedangkan node 3 adalah model interseksi pertama.

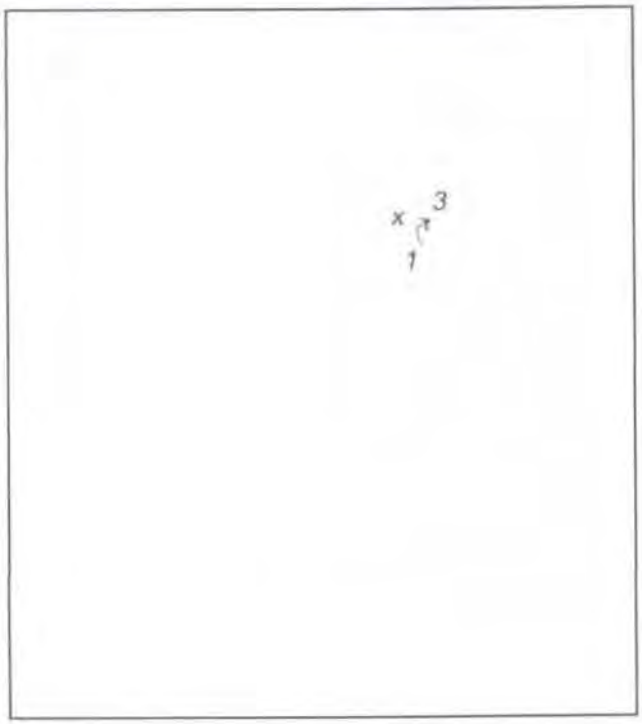
Gambar II-18 Model Interseksi Pada Salah Satu Ruas Jalan di Surabaya

2.4.2 Pemodelan Belokan Jalan

Dalam pemodelan jaringan jalan, memodelkan belokan yang terjadi akan menjadi lebih kompleks. Hal ini terjadi karena suatu belokan terdiri atas node dan

segmen-segmen garis yang berhubungan dengan node tersebut. Segmen-segmen garis yang dihubungkan oleh node-node tersebut dapat saling berhubungan melalui arah manapun.

Misal seperti dalam *gambar II*-19, suatu node x menghubungkan empat segmen jalan, maka keempat segmen jalan ini dapat saling menuju ke segmen jalan yang lain melalui arah manapun. Sedangkan dalam dunia nyata hanya boleh dilakukan belokan dari segmen jalan satu (dikenal dengan istilah *From Edge*) ke segmen jalan tiga (dikenal dengan istilah *To Edge*).



Gambar II-19 Pemodelan Belokan Pada Salah Satu Ruas Jalan di Kota Surabaya

2.5 ArcView GIS

ArcView GIS merupakan sebuah software yang memudahkan user untuk melihat, mengeksplorasi dan melakukan analisa data spasial. Fungsi-fungsi pemetaan dasar dan kemampuan dalam pengolahan data spatial yang komplek

yang disediakan oleh ArcView GIS memudahkan user dalam membuat peta untuk menampilkan, mengintegrasikan serta melihat data-data dalam berbagai bentuk. ArcView GIS memberikan kemudahan dalam menampilkan data-data spatial secara interaktif.

2.5.1 Konsep Avenue

Avenue merupakan sebuah bahasa scripting berorientasi obyek yang berada di dalam ArcView GIS dan dapat digunakan untuk membantu aplikasi ArcView GIS dalam melakukan pengolahan data.

Penggunaan script Avenue sebagai bahasa pemrograman di lingkungan ArcView sangat membantu untuk melakukan kustomisasi data, meminta ArcView untuk langsung melakukan suatu perintah tertentu tanpa perlu melalui menu-menu yang tersedia atau bahkan untuk mengembangkan suatu aplikasi yang lengkap dengan ArcView user interface.

Penekanan Avenue sebagaimana pada bahasa pemrograman berorientasi obyek yang lain adalah pada pengidentifikasian obyek dan kemudian mengirimkan request pada obyek tersebut. Ketika obyek menerima request ini, kemudian ia akan melakukan beberapa tindakan untuk merespon request tersebut.

Obyek ArcView merupakan anggota dari sebuah hirarki class yang diorganisasikan kedalam kategori fungsional yang berhubungan dengan seluruh aspek dari aplikasi.

Untuk mengorganisasikan kapan dan bagaimana request dibuat dapat digunakan statement Avenue. Request menspesifikasikan apa yang akan dilakukan oleh instance dari class yang diberikan dan sebuah method yang

menspesifikasikan bagaiman request diselesaikan. Oleh karena itu pemrograman Avenue lebih banyak berupa penulisan obyek request dibanding pemanggilan fungsi. Dengan mengirimkan request ke obyek maka akan diaktifkan pula method yang sesuai pada class dimana instance dari class adalah object tersebut. Jumlah request dan obyek yang dapat digunakan oleh Avenue tergantung dari aplikasi yang diintegrasikan ke dalam ArcView GIS. Misal request **FindPath** pada obyek **Network** hanya dapat digunakan jika ekstension ArcView Network Analyst diintegrasikan dalam project ArcView GIS.

Script Avenue menyediakan fleksibilitas yang tinggi serta melakukan kustomisasi analisa serta queri. Sebagaimana penelitian yang dilakukan oleh David Theobald[4]. avenue juga dapat digunakan untuk mengintegrasikan berbagai bahasa pemrograman, seperti HTML, Java, ASP serta Javascript hingga dapat dihasilkan aplikasi berbasis web yang diinginkan. Integrasi ini dapat dilakukan dengan menggunakan request **WriteString** atau **WriteeIt** pada obyek **Weblink.** Selain itu untuk pengiriman URL juga dapat dimanfaatkan tag Form HTML dan menyediakan input tipe hidden yang berisi script Avenue yang akan dijalankan bila form tersebut dikirim beserta informasi-informasi yang dibutuhkan oleh script Avenue tersebut untuk melakukan pengolahan data yang terdapat dalam script tersebut.

Kekompleksitasan integrasi berbagai bahasa pemrograman kedalam Avenue merupakan salah satu permasalahan yang penting untuk diperhatikan [2]. Programmer harus berhati-hati dalam menentukan variabel atau token yang digunakan. Misal dalam penggunaan tanda petik dua, request Writestring tidak

mengenali tanda petik dua didalamnya, kecuali diapit oleh tanda petik dua yang lain. Oleh karena itu penggunaan tanda petik dua harus diperhatikan. Jika tidak program tidak akan berjalan seperti yang diharapkan.

2.5.2 ArcView *Shapefile*

ArcView *shapefile* digunakan untuk menyimpan lokasi secara geometris dan informasi tentang atribut-atribut pada objek geografis.

Format *shapefile* ini memiliki 5 buah file yang kelimanya tidak harus dimiliki oleh suatu peta tematik. Kelima jenis file ini memiliki ekstensi yang berbeda tetapi harus disimpan dalam satu ruang kerja proyek. Kelima jenis file ini adalah

- **.shp** – file yang menyimpan informasi geometri.
- **.shx** – file yang menyimpan index dari informasi geometri
- **.dbf** – file dBASE yang menyimpan atribut objek geografis. Ketika sebuah *shapefile* dibuat sebagai sebuah *theme* dalam sebuah *view*, file ini ditampilkan sebagai tabel fitur.
- **.sbn dan .sbx** – file-file yang menyimpan index spasial dari fitur-fitur geografis. Kedua jenis file ini hanya dibuat, jika dilakukan operasi seleksi antar *theme*, spatial join, dan pembuatan indeks pada *field Shape* pada sebuah *theme*. File ini bersifat opsional.
- **.ain dan .aih** – file-file ini menyimpan indeks atribut dari *field-field* yang aktif dalam tabel atau pada tabel fitur sebuah *theme*. File ini bersifat opsional.

2.5.3 Organisasi *Shapefile*

Shapefile menyimpan data geometri yang bersifat non topologis dan informasi atribut untuk data spasial dalam satu kumpulan data. Data geometri tersebut disimpan sebagai sebuah *shape* yang terdiri dari himpunan kordinat-kordinat vektor. *Shapefile* dapat memuat fitur-fitur tunggal yang saling *overlap* maupun yang terpisah.

Shapefile mendukung bentuk *shape* berupa *point, line,* dan fitur area. Fitur area direpresentasikan sebagai sebuah kurva tertutup ataupun poligon.

Data atribut disimpan dalam format file dBASE. Setiap *record* atribut memiliki relasi *one-to-one* dengan *record shape* yang bersesuaian.

Shapefile dapat dibuat dengan berbagai cara, antara lain:

- **Export.** *Shapefile* dapat dibuat dengan mengekspor sumber data ke dalam sebuah *shapefile* dengan menggunakan software-software ARC/INFO, PC ARC/INFO, Spatial Database Engine (SDE), ArcView SIG, atau BusinessMAP.

- ***Digitizing.*** *Shapefile* dapat dibuat secara langsung dengan melakukan *digitizing shape* dengan menggunakan *tool* pembuatan fitur ArcView SIG.

- **Pemrograman.** Dengan menggunakan Avenue (ArcView SIG), MapObjects, ARC Macro Language (AML) (ARC/INFO), atau Simple Macro Language (SML), *shapefile* dapat dijadikan melalui pemrograman. Juga dapat memodifikasi kandungan *shapefile* yang sudah ada.

Sebuah *shapefile* terdiri dari sebuah file utama, sebuah file index, dan sebuah tabel dBASE. File utama adalah sebuah file dengan *record-record* yang setiap rekordnya mewakili sebuah *shape* dengan serangkaian titik-titiknya. Dalam file index, setiap *record* mengandung *offset* dari *record* file utama mulai dari *record* awal. Tabel dBASE mengandung atribut-atribut *fitur* dengan satu *record* untuk setiap *fitur*. Relasi *one-to-one* antara data geometri dan data atribut didasarkan pada nomor *record*. *Record-record* atribut dalam file dBASE harus dalam urutan yang sama dengan *record-record* dalam file utama.

2.5.4 View

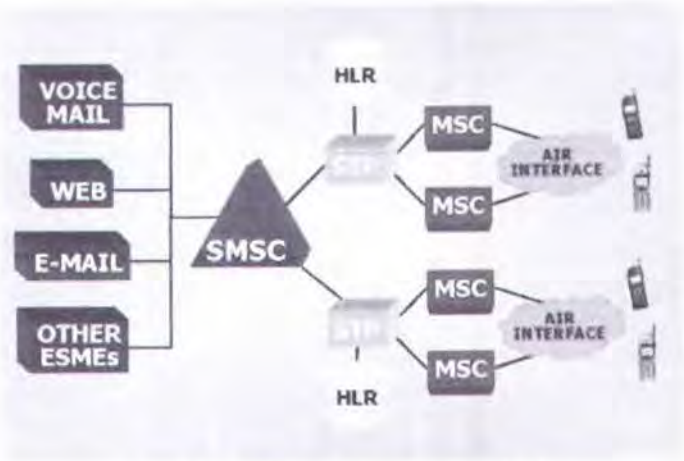
View adalah sebuah peta interaktif yang membantu untuk menampilkan, mengorganisir, melakukan *query*, dan menganalisa data geografis dalam ArcView. *View* menentukan data geografis yang akan digunakan dan mengatur bagaimana ia akan ditampilkan, akan tetapi *view* sendiri tidak memiliki data tersebut. Melainkan, *view* akan merujuk ke beberapa file sumber data. Hal ini berarti *view* bersifat dinamis karena *view* hanya menampilkan status data pada file-file tersebut. Keuntungan lainnya data yang ada dapat ditampilkan pada beberapa *view* berbeda sekaligus.

View pada dasarnya merupakan sebuah kumpulan *theme*. Sebuah *theme* mewakili sebuah set terbatas fitur-fitur geografis dari data-data geografis yang ada.Setiap *shapefile* dapat dibentuk menjadi sebuah *theme* atau sebuah *layer* pada dokumen *view*. Gabungan beberapa file, yaitu .shp, .shx, dan .dbf akan memberikan informasi yang cukup untuk dapat menampilkan sebuah *layer/theme* dalam dokumen *view* beserta data-data atributnya.

2.6 TEKNOLOGI SMS (SHORT MESSAGE SERVICE)

Teknologi SMS dimunculkan pertama kali di Eropa sekitar tahun 1991, oleh suatu badan standard Eropa untuk digital wireless yang disebut dengan *Global System For Mobile Communications (GSM)*. SMS menyediakan mekanisme untuk melakukan transmisi short message dari dan ke *wireless device* . Wireless network memberikan mekanisme yang dibutuhkan untuk menemukan stations tujuan dan mengirimkan short message antara SMSC(Short Messaging Service Center) dengan *Wireless Station.*

Bentuk jaringan *Wireless* dapat digambarkan seperti dibawah ini:



Gambar II-20 Jaringan Wireless SMS

2.6.1 Elemen Jaringan dan Arsitektur dari SMS

2.6.1.1 External Short Messaging Entities

External Messaging Entities (ESME) adalah sebuah device yang memungkinkan untuk menerima atau mengirim pesan pendek. Sebuah short message entity (SME) harus ditaruh di dalam jaringan, mobil device atau service center yang lain.

SMSC adalah kombinasi dari *hardware* dan *software* yang bertanggung jawab untuk me-relay dan menyimpan pesan antara SME dan mobile device. SMSC harus mempunyai reliabilitas yang tinggi sehingga dapat mengakomodasi permintaan SMS yang semakin berkembang.

STP (Signal Transfer Point) adalah elemen jaringan yang tersedia pada deployment IN (*Inteligent Network*) untuk menghubungkan *link* dengan elemen jaringan yang *multiple*.

HLR (Home Location Register) adalah database yang digunakan untuk penyimpanan permanen dan manajemen *subscription* dan *service* profile. jika station tujuan tidak ada maka HLR akan menginformasikan pada SMSC bahwa station diperlukan untuk diakses oleh jaringan mobile.

VLR (Visitor Location Register) adalah database yang berisi informasi sementara tentang *subscriber* yang masuk .

MSC membentuk fungsi switch dari sistem dan melakukan kontrol panggilan dari dan ke telepon serta sistem data.MSC akan menyampaikan data pesan ke mobile *subscriber* yang dituju melalui base station.

Air Interface didefinisikan pada setiap satu teknologi wireless yang berbeda (GSM,TDMA,CDMA). *Air interface* menspesifikasikan bagaimana suara atau signal data ditransfer dari MSc ke *handset*.

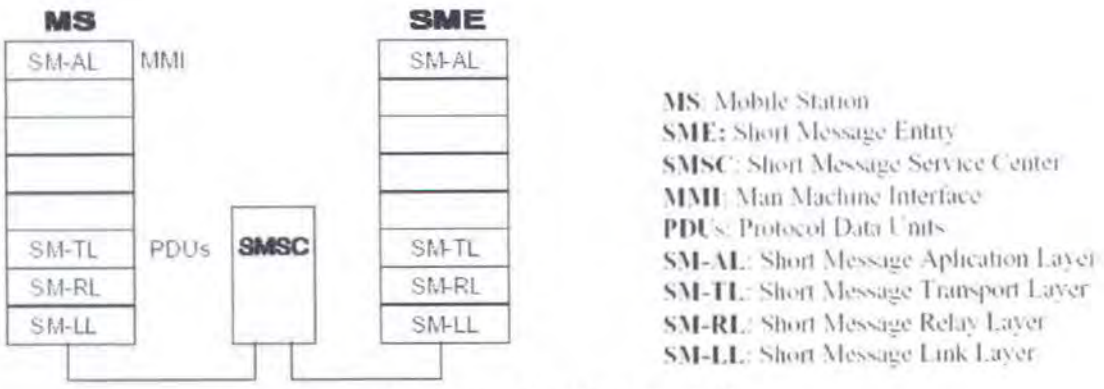
Base Station System terdiri dari *base station controllers* (BSCs) dan Base transceiver stations (BTSs) yang berfungsi untuk mengatur transmisi elektromagnetik antar MSC dan mobile devices.

Mobile Device terminal wireless yang mampu untuk menerima dan mengirimkan pesan (Short message) melalui jaringan wireless.

2.6.2 SMS PDU Mode

Pdu Mode merupakan suatu cara untuk mengirim informasidalam format 7 bit atau 8 bit. Pada pesan SMS yang dipesifikasikan oleh *Etsi organization* setiap pesan dapat mempunyai panjang karakter lebih dari 60 karakter dimana masing-masing karakter adalah 7 bit.

Untuk menggunakan SMS harus dideklarasikan nomor dari SMSC1 (*Short Message Service*) di dalam MS(*Mobile Station*) yang disediakan untuk dukungan MS yaitu:SMS-MO (*Short Message Service - Mobile Originated*).



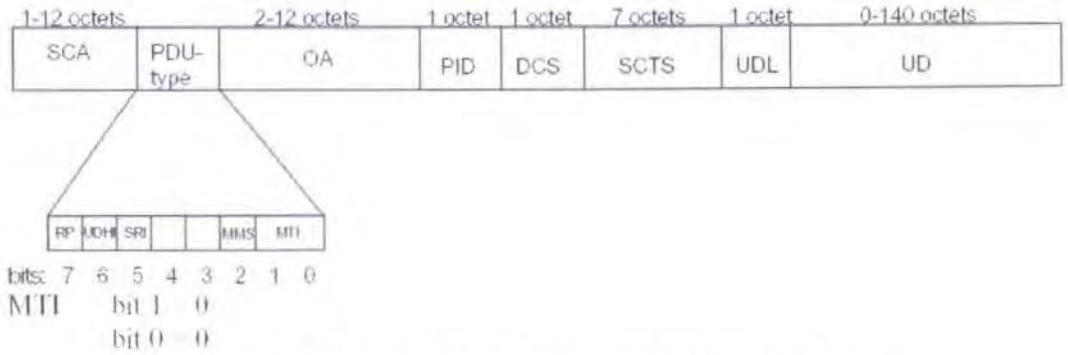
Gambar II-21 Arsitektur MS, SMSC dan SME

MMI (*Man Machine Interface*) didasarkan pada perintah AT+Cellular dan dapat direalisasikan dengan sebuah terminal (sebagai contoh Triodata,Telix, atau Windows-Hyperterminal).

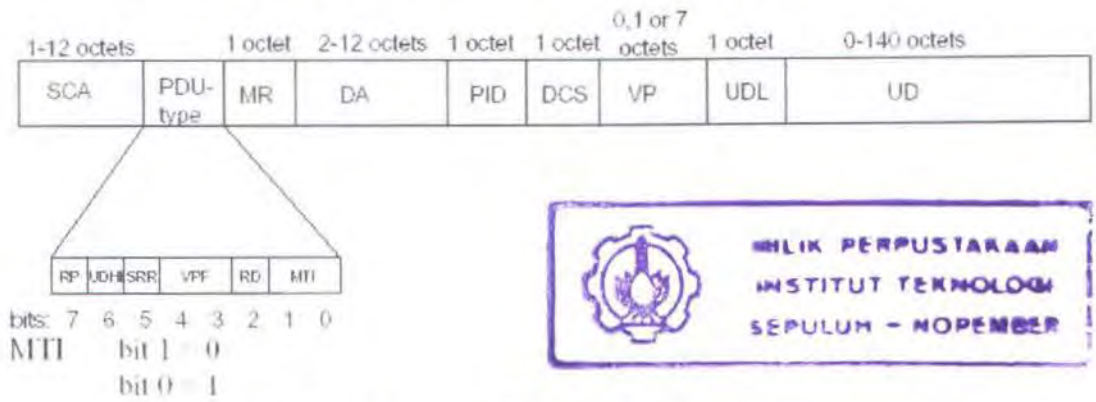
SM-TL(*Short Message Transport Layer*) menyediakan pelayanan sebuah service untuk layer aplikasi Short Message. Pelayanan ini memungkinakn untuk SM-AL(*Short Message Application Layer*) untuk mengirimkan dan menerima Short Message dari *Peer Entity*. SM-TL berkomunikasi dengan peer entitynya dengan 6 PDUs (Protocol Data Units)

- SMS-DELIVER: Menyampaikan *short message* dari SMSC ke MS.
- SMS-DELIVER-REPORT: Menyampaikan pesan error deliver jika diperlukan.
- SMS-SUBMIT: Menyampaikan *short message* dari MS ke SMSC.
- SMS-SUBMIT-REPORT: Menyampaikan pesan error submit jika diperlukan.
- SMS-STATUS-REPORT: Menyampaikan Status laporan (*report*) dari SMSC ke MS.
- SMS-COMMAND: Menyampaikan perintah (*command*) dari MS ke SMSC.

SMS-DELIVER dan SMS-SUBMIT secara jelas dapat digambarkan sebagai berikut:



Gambar II-22 Diagram SMS-DELIVER(Mobile-Terminated)



Gambar II-23 Diagram SMS-SUBMIT(Mobile-Originated)

Berikut ini tabel untuk parameter-parameter yang ada di dalam SMS-DELIVER dan SMS-SUBMIT:

SCA	Service Center Address - information element	Nomor telepon Service Center
PDU Type	Protocol Data Unit Type	
MR	Message Reference	Nomor Urut (0..255) dari semua Frame SMS-SUBMIT dengan MOBILE

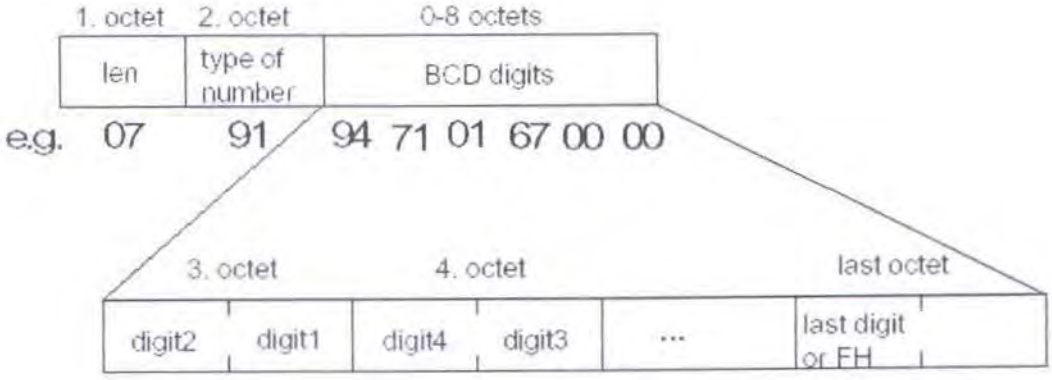
OA	Originator Adress	alamat SME asli (Originating)
DA	Destination Adress	Alamat tujuan SME (Destination)
PID	Protocol Identifier	Parameter untuk menunjukkan bagaimana SMSC memproses SM (sebagai FAX, Suara dan lain-lain)
DCS	Data Coding Scheme	Parameter yang mengidentifikasikan coding scheme dalam User Data (UD)
SCTS	Service Center Time Stamp	Parameter yang mengidentifikasikan waktu ketika SMSC menerima pesan
VP	Validity Period	Parameter yang mengidentifikasikan waktu bahwa kapan pesan tidak akan valid lagi di MSC
UDL	User Data Length	Parameter yang mengidentifikasikan panjang dari field UD
UD	User Data	Data dari SM
RP	Reply Path	Parameter yang mengidentifikasikan ada tidaknya Reply Path
UDHI	User Data Header	Parameter yang

	Indicator	mengidentifikasikan bahwa field UD berisi Header
SRI	Status Report Indication	Parameter yang mengidentifikasikan bahwa SME diminta untuk status report
SRR	Status Report Request	Parameter yang mengidentifikasikan bahwa MS diminta untuk status report
VPF	Validity Period Format	Parameter yang mengidentifikasikan ada tidaknya field VP di waktu sekarang
MMS	More Messages to Send	Parameter yang mengidentifikasikan ada tidaknya pesan yang akan dikirim
RD	Reject Duplicate	
MTI	Message Type Indicator	Parameter yang menggambarkan tipe dari pesan 00 berarti SMS-DELIVER 01 berarti SMS-SUBMIT

Tabel II-5 Istilah-istilah SMS-DELIVER dan SMS-SUBMIT

2.6.2.1 Deskripsi parameter

2.6.2.1.1 Service Center Address Information element (SCA info element)



Gambar II-24 SCA Info Elemen

len:

Octet "len" berisi nomor dari octet yang dibutuhkan untuk nomor dari service center ditambah dengan 1 byte, tipe dari nomor.

type of number:

81H: Nomor ini adalah nomor untuk national.

91H: Nomor ini adalah nomor untuk international.

octet:

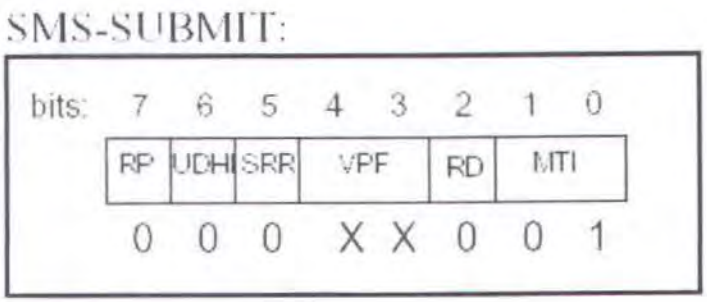
Satu octet didalamnya ada dua *BCD-digit Fields*, JIka nomor BCD berisi nomor ganjil maka digit terakhir akan diisi dengan kode "FH".

Contoh:

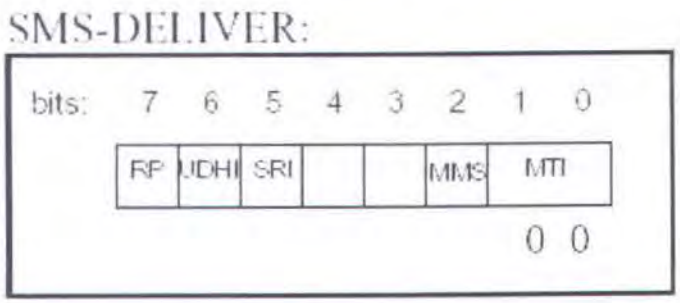
Jika mempunyai nomor-SC +49 171 0760000 maka mempunyai tipe:

0791947101670000

2.6.2.2 Protocol Data Unit Type



Gambar II-25 PDU Type Submit



Gambar II-26 PDU Type Deliver

PDU type harus ditulis dalam bentuk bilangan Hexadecimal. Berikut ini keterangan dari gambar diatas:

RP: 0 Parameter Reply tidak di set dalam PDU ini.

1 Parameter Reply di set dalam PDU ini.

UDHI: 0 *Field* UD hanya berisi *short message*.

1 *Field* awal dari UD berisi *header* tamabhan dari *short message*

SRI: (hanya di-set oleh SMSC)

0 Status *report* tidak akan dikembalikan ke SME.

1 Status *report* akan dikembalikan ke SME.

SRR: 0 status report tidak diminta

1 status report diminta

VPF: bit4 bit3

0 0 *Field* VP bukan waktu sekarang

0 1 *Reserved*

1 0 *Field* VP sekarang digambarkan oleh integer (relatif)

1 1 Field VP sekarang digambarkan oleh semi-octet (absolute)

Nilai VP yang lain akan di tolak oleh SMSC.

MMS: (hanya di-set oleh SMSC)

0 ada pesan lain untuk MS yang menunggu di SMSC

1 tidak ada pesan untuk MS yang menunggu di SMSC

RD:

0 Menginstruksikan SMSC untuk **menerima** sebuah SMS-SUBMIT untuk sebuah short message yang masih ada di SMSC yang mempunyai MR dan DA sama dan sebelumnya disubmit sebagaishort message dari OA yang sama.

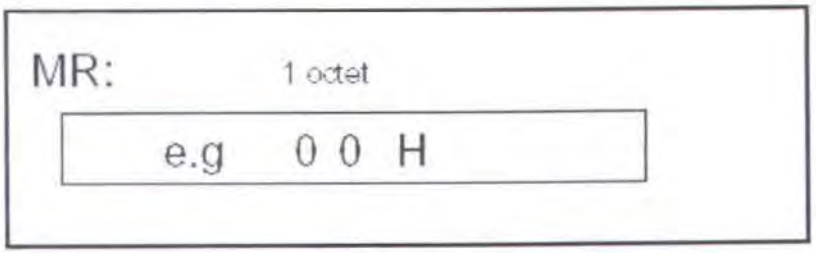
1 Menginstruksikan SMSC untuk **menolak** sebuah SMS-SUBMIT untuk sebuah short message yang masih ada di SMSC yang mempunyai MR dan DA sama dan sebelumnya disubmit sebagaishort message dari OA yang sama.

MTI: bit1 bit0 tipe pesan

0 0 SMS-DELIVER (SMSC ==> MS)

0	0	SMS-DELIVER REPORT (MS ==> SMSC, yang digenerate secara otomatis oleh MOBILE, setelah menerima SMS-DELIVER
0	1	SMS-SUBMIT (MS ==> SMSC)
0	1	SMS-SUBMIT REPORT (SMSC ==> MS)
1	0	SMS-STATUS REPORT (SMSC ==> MS)
1	0	SMS-COMMAND (MS ==> SMSC)
1	1	Reserved

2.6.2.3 Message Reference (MR)

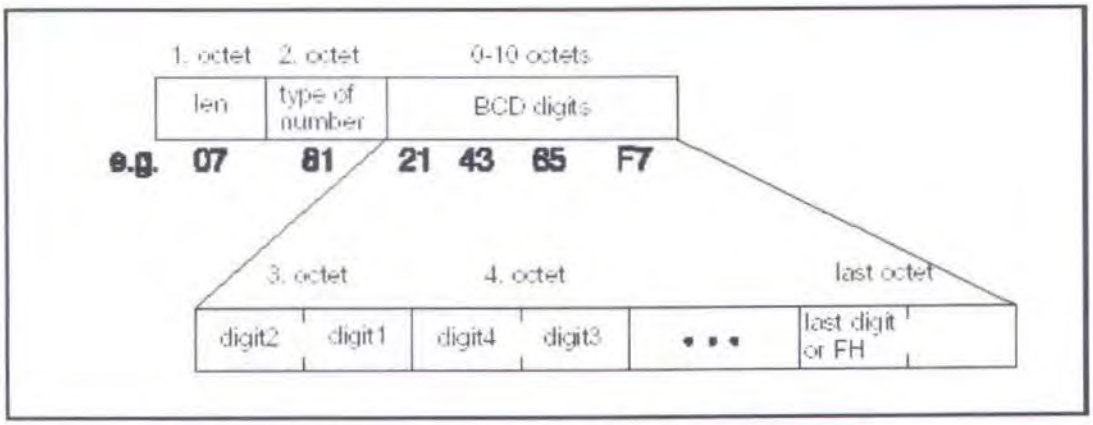


Gambar II-27 Message Reference

Field MR merepresentasikan sebuah nilai integer (0...255) dari nomor referensi SMS-SUBMIT yang disubmit ke SMSC oleh MS.

2.6.2.4 Originator Address OA Destination Address DA

OA dan DA mempunyai format yang sama yang dapat dijelaskan seperti berikut ini :



Gambar II-28 Originator Address OA dan Destination Address DA

len:

Octet "len" berisi nomor dari digit BCD

type of number:

81H: nomor ini adalah nomor National

91H: nomor ini adalah nomor International

BCD-digits:

Fields digit-BCD berisi nomor-BCD dari Destination contohnya originator, Dan jika nomor BCD berisi bilangan ganjil maka digit terakhir akan diisi dengan "FH"

Contoh:

Jika mempunyai nomor nasional 1234567 maka mempunyai tipe:

0781214365F7

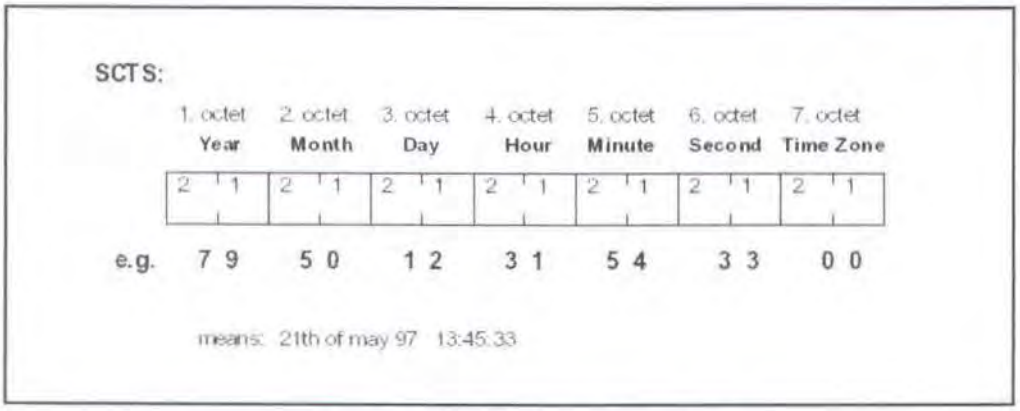
Field DCS mengindikasikan data coding scheme dari *field* UD (User Data). Octed digunakan untuk mengacu pada *coding group* yang diindikasikan dalam bit 7...4 ,Octet dapat dibentuk ke dalam kode seperti berikut ini:

Coding group: bits 7..4	bits 3..0

Gambar II-31 Coding Group

2.6.2.7 Service Center Time Stamp (SCTS)

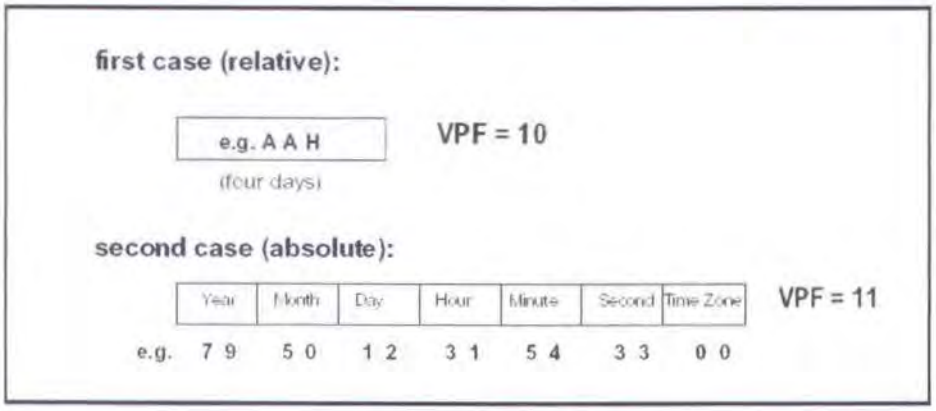
SCTS adalah elemen informasi SMSC yang menginformasikan penerima MS tentang waktu kedatangan *short message* pada entitas transport layer di SMSC. Nilai waktu yang dimasukkan di setiap SMS-DELIVER merepresentasikan waktu local dari komputer. Secara lengkap SCTS dapat digambarkan seperti berikut ini:



Gambar II-32 Service Center Time Stamp

2.6.2.8 Validity Period VP

Validity-Period adalah elemen informasi pada MS yang mengirimkan SMS-SUBMIT ke SMSC dengan memasukkan periode waktu yang spesifik ke dalam short message. Parameter validity period mengindikasikan periode waktu untuk sebuah short message masih tetap valid .



Gambar II-33 Validity Period

Pada Kasus pertama, VP terdiri dari 1 octet yang berisi panjang dari *validity period* yang dihitung ketika SMS-SUBMIT diterima oleh SMSC. Dalam kasus kedua VP terdiri dari 7 octet yang berisi waktu absolute dari terminasi validity period.

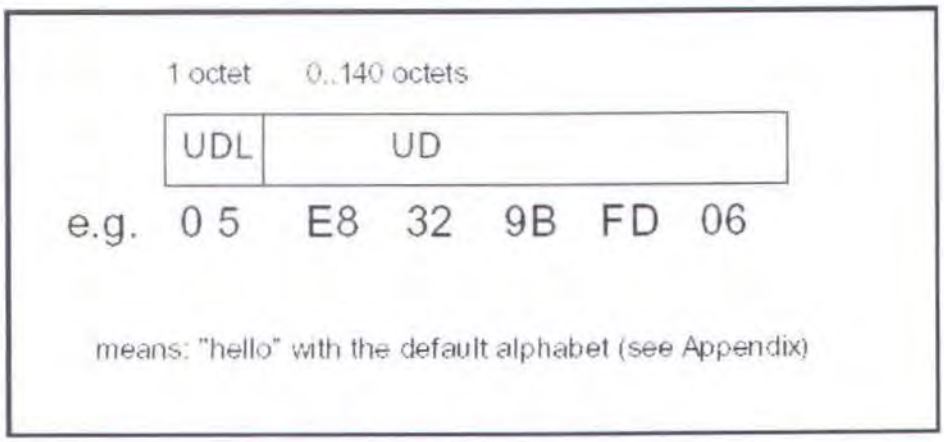
Untuk kasus pertama dapat dijelaskan seperti berikut ini:

VP Value	Validity period value
0-143	(VP + 1) x 5 minutes (i.e 5 minutes intervals up to 12 hours)
144-167	12 hours + ((VP-143) x 30 minutes)
168-196	(VP-166) x 1 day
197-255	(VP - 192) x 1 week

Gambar II-34 Penghitungan Validity Period

Untuk kasus kedua representasi waktu sama dengan representasi waktu pada Service Center Time Stamp (SCTS).

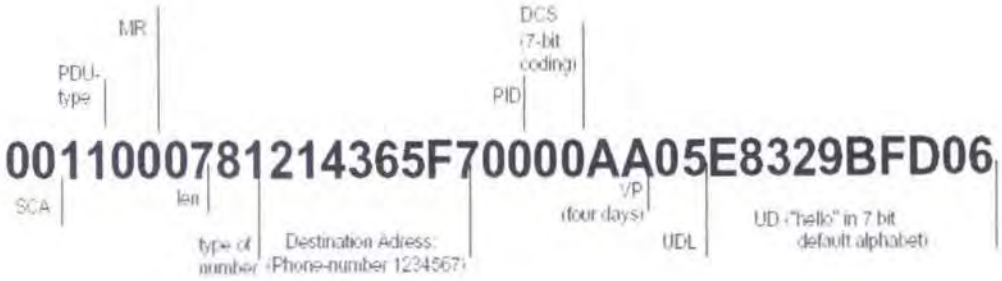
2.6.2.9 User Data Length UDL dan User Data UD



Gambar II-35 UDL dan UD

Field UDL memberikan representasi integer dari jumlah karakter yang ada di dalam Field User Data .

2.6.2.10 Contoh PDU



Gambar II-36 PDU untuk Kata "hello" dalam Format 7 bit.

Gambar II-37 PDU untuk Kata "hello" dalam Format 8 bit.

2.6.3 AT COMMAND

AT COMMAND merupakan perintah untuk melakukan control terhadap mobile telephone GSM melalui interface serial (baik melalui kabel data atau koneksi infrared).

Perintah harus dimulai dengan "AT" dan diakhiri dengan "<CR>" (=0x0D). Input dari perintah akan mendapatkan respon balik (*acknowledge*) dengan "OK" atau "ERROR".

2.6.3.1 Hayes Standard Command

Hayes-standard Command merupakan command dari AT Hayes yang compatible. Hayes standard command dapat dilihat pada tabel berikut ini:

Command	Function
A/	Mengulangi perintah yang terakhir
AT...	Awalan Untuk Semua Perintah yang Lain
ATD<str>;	Dial untuk *dialing string* <str> dengan menggunakan utilitas suara (*voice*) *Valid dial*: "T" (*Tone dialing*) dan jika gagal "P" (Pulse dialing) *ignored.*
ATD><n>;	Memanggil (*Dial*) nomor telepon dari lokasi buku telepon saat ini <n> dan buku telepon diseleksi dengan perintah at+cpbs atau (at^spbs).
ATE0	Menonaktifkan perintah echo
ATE1	Mengaktifkan perintah echo

ATZ	Set Ke konfigurasi default

Tabel II-6 Hayes Standard Command

2.6.3.2 Acknowledge untuk Komunikasi Data Normal

Response	Numeric	Meaning
OK	0	Perintah dieksekusi tidak ada Error
RING	2	Ring terdeteksi
NO CARRIER	3	Link Tidak ada(Disconnect)
ERROR	4	Perintah Salah
NO DIAL TONE	6	Tidak ada dial (mode salah)
BUSY	7	*Remote Station* sedang sibuk (*Busy*)

Tabel II-7 Tabel Acknowledge untuk Komunikasi Data Normal

2.6.3.3 AT Command dan Response

Berikut ini merupakan beberapa cara untuk mengeksekusi AT Command dalam berbagai bentuk yaitu:

Test command	AT+CXXX=?	Telepon akan merespond dengan mengirimkan list parameter dan range nilai
Read command	AT+CXXX?	Perintah ini akan menginformasikan bahwa setting nilai sekarang dari parameter (s).
Write command	AT+CXXX=<...>	Perintah ini digunakan untuk menge-set parameter yang bias di-set.

Execute command AT+CXXX	Perintah execute membaca parameter yang tidak di-set tetapi berpengaruh pada proses internal dalam telepon.

2.6.3.4 Beberapa Perintah AT Cellular Pada GSM

2.6.3.4.1 Perintah untuk memeriksa SMS pada handphone

AT+CMGL	LIST SMS
Test Command AT+CMGL=?	Response +CMGL: (List of supported <stat>s) parameter <stat> 0 "REC UNREAD" ,menerima semua pesan yang tidak dibaca. 1 "REC READ", menerima semua pesan yang dibaca. 2 "STO UNSENT", menyimpan pesan yang tidak dikirim. 3 "STO SENT", menyimpan pesan yang dikirim. 4 "ALL", Untuk semua pesan.

Tabel II-8 AT Command untuk memeriksa SMS

2.6.3.4.2 Perintah untuk mengirim SMS

AT+CMGS	***Send an SMS***
Test command AT+CMGS=?	*Response* OK

Tabel II-9 AT Command untuk mengirim SMS

2.6.3.4.3 Perintah untuk menghapus SMS

AT+CMGD	***Delete an SMS in SMS memory***
Test command AT+CMGD=?	*Response* OK

Tabel II-10 AT Command untuk menghapus SMS

2.6.3.4.4 Contoh untuk mengirimkan SMS dengan perintah AT+Cellular

`at+cmgs=140`	*memasukkan "send message", 140 adalah panjang maksimum (dalam byte).*
`0011000781214365F` `70000AA05E8329BFD06`	*Tipe PDU (SMS SUBMIT) dan diakhiri dengan "Ctrl Z") dan angka yang tidak dicetak tebal merupakan alamat tujuan dari SMS.*
`+CMGS: 0` `OK`	
`at+cpms?`	*Apakah pesan disimpan di dalam* `SIM-Card?`
`+CPMS: "SM" , 1 , 7 , "SM" , 1 , 7`	Di dalam SIM-Card ini ada 1 pesan yang disimpan
`OK`	Dapat menyimpan pesan dengan jumlah maksimal 7 pesan.

at+cmgr=1	*Membaca pesan yang disimpan dalam lokasi 1*
+CMGR: 0 ,, 24 00040C91947182152192000069 30824161840005E8329BFD06 OK	ini adalah PDU (SMS-DELIVER) yang dikirim oleh Service Center

Tabel II-11 Contoh Pengiriman SMS dengan AT Command

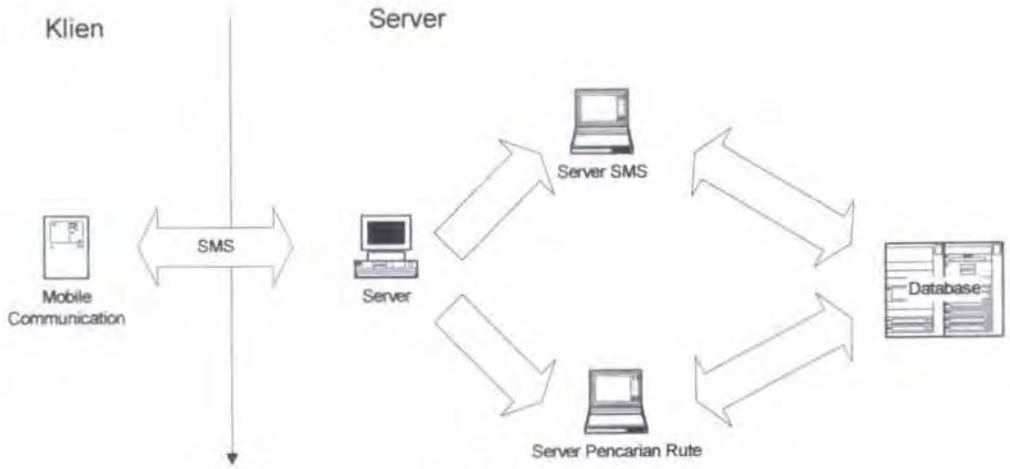
BAB III
PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan membahas mengenai perancangan perangkat lunak. Pembahasan ini meliputi arsitektur sistem secara umum, perancangan basis data yang digunakan untuk pengolahan rute yaitu data atribut peta dan data untuk pengolahan SMS, perancangan struktur data aplikasi, perancangan proses berupa perancangan struktur data dan proses pencarian rute tercepat serta proses pengolahan data SMS, yang terakhir adalah perancangan desain antar muka. Untuk perancangan basis data digunakan perangkat lunak Power Designer 9.0 dan perancangan prosesnya menggunakan pendekatan desain berorientasi objek yang direpresentasikan dengan mengunakan **UML**(*Unified Modeling Language*) dengan perangkat lunak Rational Rose Enterprise Edition 2000.

3.1 ARSITEKTUR SISTEM

Arsitektur sistem SMS Rute ini terbagi menjadi dua sisi, yaitu sisi *server* dan sisi *client*. Sisi server dibagi lagi menjadi dua yaitu server untuk mengirim dan menerima SMS dan server untuk melakukan pencarian rute, sedangkan untuk sisi *client* hanya berupa *mobile communication,* dan media komunikasi antara client dan server memanfaatkan teknologi SMS. Interaksi antara SMS server dan server pencari rute dengan memanfaatkan database, bahwa data yang diterima oleh Server SMS dimasukkan ke database, maka server pencari rute dapat mengolah data SMS tersebut dan hasil olahannya disimpan lagi ke dalam database sehingga server SMS dapat mengirimkan kembali ke pengguna yang

telah meminta. Adapaun gambaran dari arsitektur sistem SMS rute ini adalah sebagai berikut :



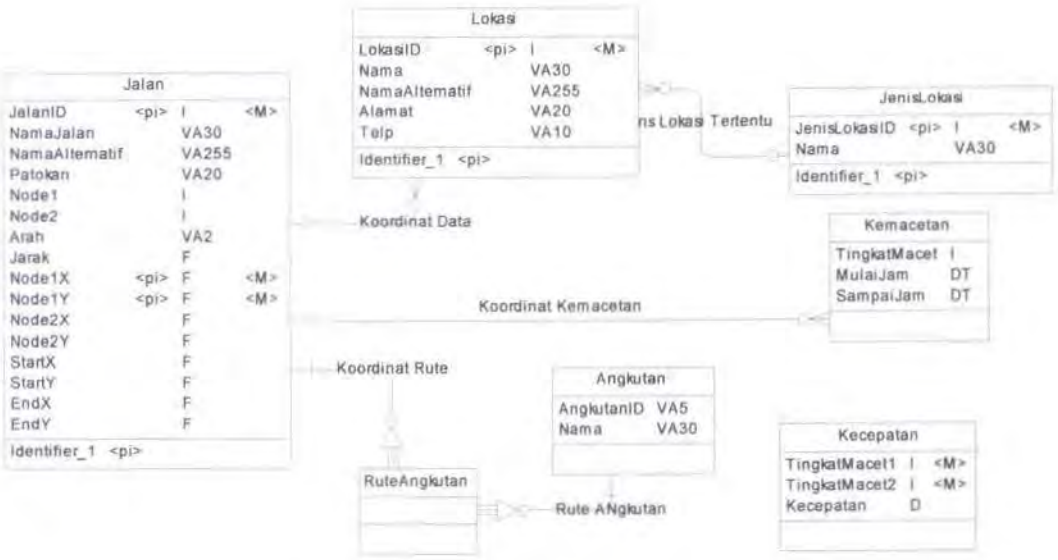
Gambar III-1 Arsitektur Sistem

3.2 PERANCANGAN BASIS DATA

Dalam perangkat lunak ini, data yang akan diolah, sebelumnya berada didalam basis data, data-data tersebut meliputi data atribut dan data untuk pengolahan SMS.

3.2.1 Perancangan Data Atribut

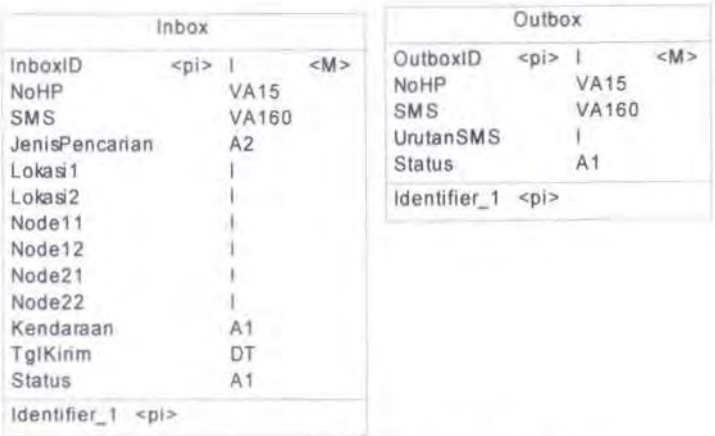
Data atribut yang akan diolah terdiri dari beberapa macam data, yaitu data informasi jalan, lokasi tertentu, tingkat kemacetan dan angkutan umum. *Conceptual Data Model* (CDM) untuk perancangan data atribut adalah sebagai berikut :



Gambar III-2 Conceptual Data Model Data Atribut

3.2.2 Perancangan Data Pengolahan SMS

Dalam pengiriman dan penerimaan SMS, data-data yang akan dikirimkan merupakan data yang sudah terformat dan siap dikirimkan, begitu juga penerimaan SMS, bahwa data yang diterima dimasukkan terlebih dahulu kedalam basis data kemudian diproses. CDM untuk pengolahan SMS adalah sebagai berikut :



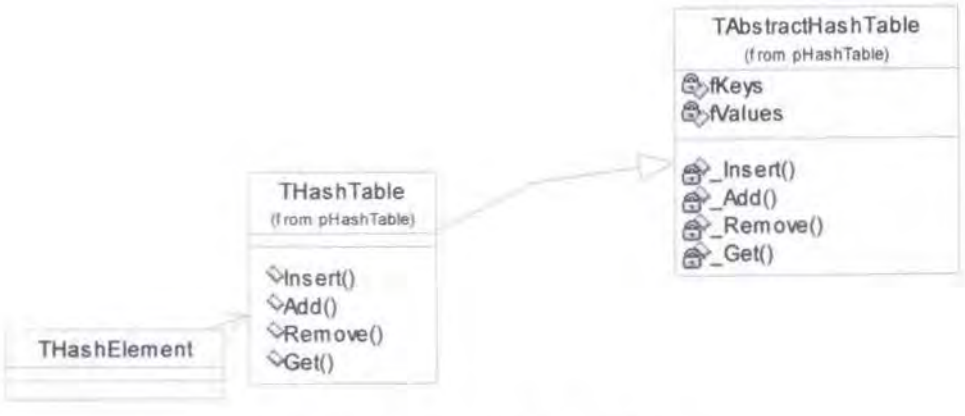
Gambar III-3 CDM Pengolahan SMS

3.3 Perancangan Struktur Data

Struktur data dalam perangkat lunak ini, dikelompokkan menjadi 3 bagian, yaitu data masukan, data proses dan data keluaran, tetapi sebelum menjelaskan ketiga data tersebut terdapat data *collection* yang berfungsi sebagai *array* untuk objek-objek dari kelas yang digunakan.

3.3.1 Perancangan Data *Collection*

Data *collection* berfungsi sebagai penyimpan objek-objek yang sejenis atau tidak sejenis, kelas untuk data *collection* tersebut adalah **THashTable**, kelas tersebut merupakan kelas induk dari kelas-kelas lain untuk data masukan ataupun data proses. Kemampuan kelas ini adalah bisa menambah, menyimpan, menghapus atau mensisipkan objek-objek yang akan dimasukkan ke dalam kelas ini. Sebelum objek dimasukkan ke dalam kelas ini, sebelumnya adalah objek tersebut dikonversi menjadi objek dari kelas **THashElement**.



Gambar III-4 THashTable dan THashElement

3.3.2 Perancangan Data Masukan

Data masukan dari perangkat lunak ini adalah berasal dari tabel, ketika perangkat lunak dioperasikan, semua record dari tabel dipindahkan kedalam data masukan tersebut. Perancangan data masukan ini dibagi menjadi 3 bagian utama yaitu perancangan data masukan untuk jalan, lokasi, kemacetan dan data angkutan umum.

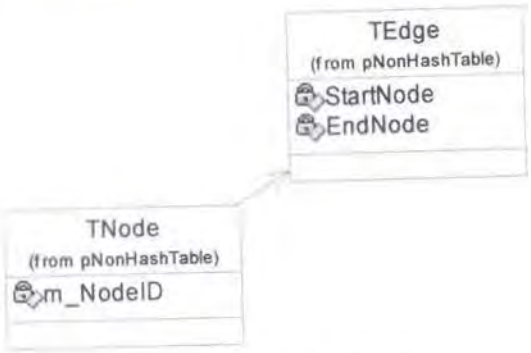
3.3.2.1 Data Jalan

Setiap record yang berada pada tabel jalan, akan dimasukkan kedalam kelas **TEdge** yang mempunyai struktur data hampir sama dengan tabel jalan, hanya pada kelas **TEdge** mempunyai tambahan struktur tentang kemacetan, lokasi dan angkutan umum.

Setelah pemindahan data dari tabel jalan ke objek dari kelas **TEdge**, objek **TEdge** tersebut disimpan ke dalam objek dari kelas **TEdges,** dimana kelas **TEdges** adalah kelas *collection* turunan dari **TAbstractEdges**, dan kelas **TAbstractEdges** adalah turunan dari kelas **THashtable**

Gambar III-5 Kelas TEdge,TEdge

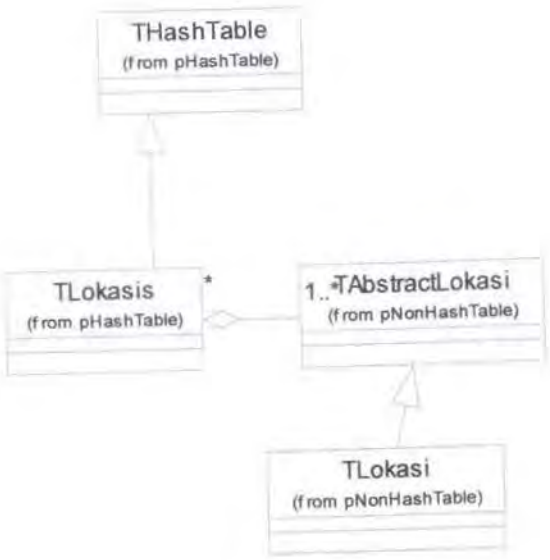
Edge terdiri dari 2 *node* yang saling berhubungan, oleh karena itu dalam kelas **TEdge** terdapat kelas **TNode**, dimana kelas **TEdge** terdapat 2 kelas **TNode** yaitu node awal dan node akhir.



Gambar III-6 Kelas TNode

3.3.2.2 Data Lokasi

Seperti dijelaskan pada data masukan jalan, bahwa setiap *edge* mempunyai informasi lokasi. Satu lokasi digambarkan dengan kelas **TLokasi,** dimana kelas tersebut mempunyai struktur data yang sama dengan tabel lokasi. Semua record pada tabel lokasi akan dimasukkan ke dalam objek dari kelas **TLokasi**. Setelah pemasukan data dari tabel ke objek dari kelas **TLokasi** selesai, maka objek tersebut akan dipindahkan ke objek **TLokasis**, yang merupakan kelas *collection* turunan dari kelas **THashtable**.



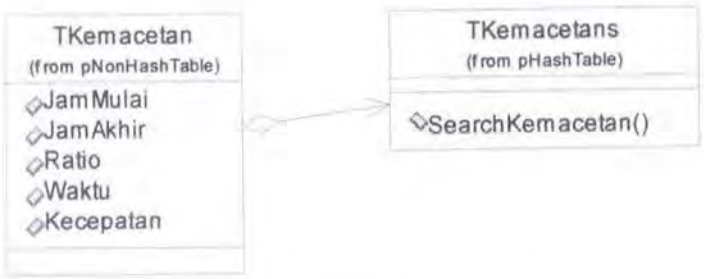
Gambar III-7 Kelas TLokasi dan TLokasis

3.3.2.3 Data Kemacetan

Tingkat kemacetan suatu jalan berubah-ubah menurut waktu atau kejadian yang terjadi pada jalan tersebut, misalkan pada saat jam pulang kerja dan jam berangkat kerja, biasanya jalan raya tersebut memiliki kemacetan yang tinggi, dan ketika terdapat jalan yang dilalui sedang ada galian pipa atau perbaikan jalan, maka kemacetan juga akan meningkat. Oleh karena itu, kemacetan dapat digambarkan sebagai kelas tersendiri dalam sistem ini, karena kemacetan tersebut mempunyai sifat-sifat yang telah dijelaskan di atas.

Kelas dari kemacetan tersebut adalah **TKemacetan**, setiap *edge* mempunyai tingkat kemacetan tertentu, maka objek dari kelas **TKemacetan** berada pada setiap kelas **TEdge,** dan karena setiap *edge* mempunyai kemacetan-kemacetan pada saat-saat tertentu, maka objek dari kelas **TKemacetan** tersebut disimpan terlebih dahulu ke dalam kelas **TKemacetans** yang merupakan kelas *collection* turunan dari kelas **THashTable**. Data awal dari kemacetan berada

pada tabel kemacetan dan kecepatan, tabel kemacetan menjelaskan tingkat kemacetan pada saat tertentu pada jalan tertentu dan tabel kecepatan untuk menentukan kecepatan rata-rata pada tingkat kemacetan tertentu.



Gambar III-8 TKemacetan dan TKemacetans

3.3.2.4 Data Angkutan Umum

Angkutan Umum dalam sistem ini adalah angkutan umum yang mempunyai trayek, angkutan umum tersebut mempunyai rute yang jelas pada setiap jalan-jalan yang akan dilaluinya. Seperti pada gambar berikut ini :

Rute untuk Angkutan A *Rute untuk Angkutan B*

Gambar III-9 Rute Angkutan

Pada gambar di atas, bahwa *edge* antara *node* 1 dan *node* 2 merupakan rute untuk dua angkutan yaitu angkutan A dan angkutan B, hal ini menunjukkan bahwa setiap *edge* dapat mempunyai 1 angkutan umum atau lebih. Pada kelas dalam perangkat lunak, angkutan digambarkan dengan kelas **TAngkutan**, kelas tersebut akan berasosiasi dengan kelas **TEdge** sebagai wakil dari *edge*, karena

setiap *edge* dapat memiliki banyak angkutan umum atau lebih maka objek dari kelas **TAngkutan** akan dimasukkan terlebih kedalam kelas *collection* **TAngkutans** turunan dari kelas **THashtable**. Data dari angkutan tersebut berasal dari tabel angkutan dan rute angkutan, tabel angkutan menjelaskan nama-nama angkutan dan tabel rute angkutan adalah berisi informasi *edge-edge* yang dituju pada masing-masing angkutan.

Gambar III-10 TAngkutan dan TAngkutans

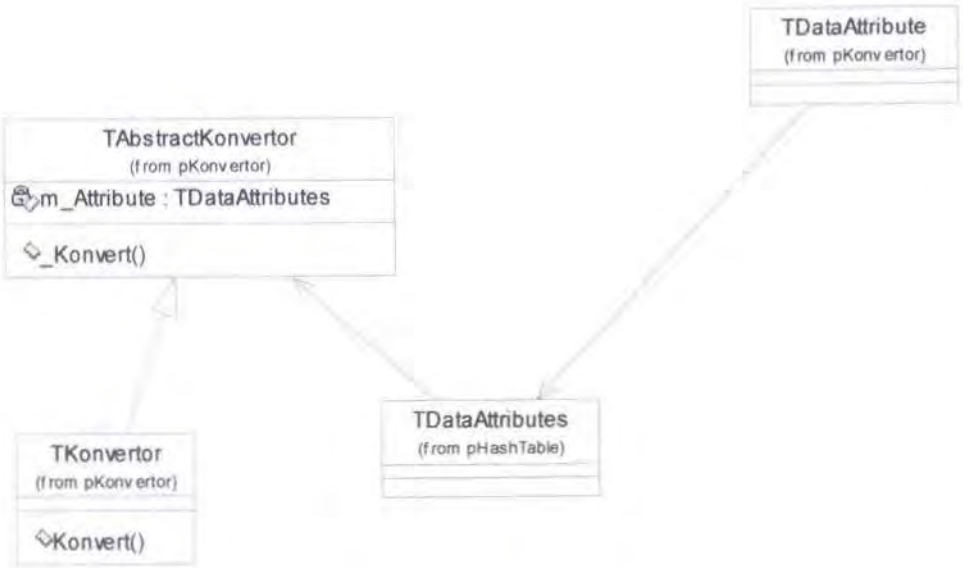
3.3.3 Perancangan Data Proses dan Data Keluaran

Proses-proses yang terjadi pada perangkat lunak ini adalah proses konversi dan pencarian rute.

3.3.3.1 Data Konversi

Dalam pengkonversian data, terdapat beberapa kelas yang terlibat. **TKonvertor** adalah kelas utama dalam melakukan pengkonversian data, merupakan turunan dari kelas **TAbstractKonvertor**, sebelum melakukan pengkonversian data, data yang akan diproses dimasukkan terlebih dahulu kedalam kelas **TDataAttribute**, secara akumulatif dimasukkan ke dalam kelas **TDataAttributes** turunan dari kelas **THashtable**, setelah semua data

dimasukkan ke dalam kelas **TDataAttribute**, maka pengkonversian data baru terjadi dan hasil dari konversi data akan dimasukkan ke dalam tabel jalan.



Gambar III-11 Kelas TKonvertor, TDataAttribute dan TDataAttributes

3.3.3.2 Data Pencarian Rute

Pencarian rute terbagi menjadi dua, yaitu pencarian rute menggunakan kendaraan pribadi dan angkutan umum. Hasil dari pencarian tersebut berupa *edge*, adapun desain kelas dari pencarian rute adalah sebagai berikut :

Gambar III-12 TRuteUmum, TRuteAngkutan, TRutePribadi, TScanEdges dan TScanAngkutans

TRuteUmum adalah kelas untuk mencari rute jika menggunakan kendaraan umum, hasil dari pencarian rute tersebut berupa *edge* turunan dari kelas **TScanEdges**, jika pencarian rute ditemukan langkah selanjutnya yaitu mencari angkutan paling minimal yang akan digunakan, dan kelas yang digunakan adalah **TRuteAngkutan**, hasil dari pencarian angkutan minimal kelas yang digunakan yaitu **TScanAngkutans**. Untuk mencari rute menggunakan kendaraan pribadi, maka kelas yang digunakan adalah **TRutePribadi**.

3.3.3.3 Data Pengiriman dan Penerimaan SMS

Untuk melakukan proses pengiriman dan penerimaan SMS menggunakan kelas **TSMS**, kelas tersebut didesain hanya untuk mengirim dan menerima SMS, pendekodean PDU atau pengiriman data menggunakan *AT Command* sudah termasuk di dalam kelas tersebut, sehingga penggunaan kelas tersebut lebih mudah karena proses pengiriman SMS menggunakan *AT Command* merupakan bagian dari kelas tersebut.

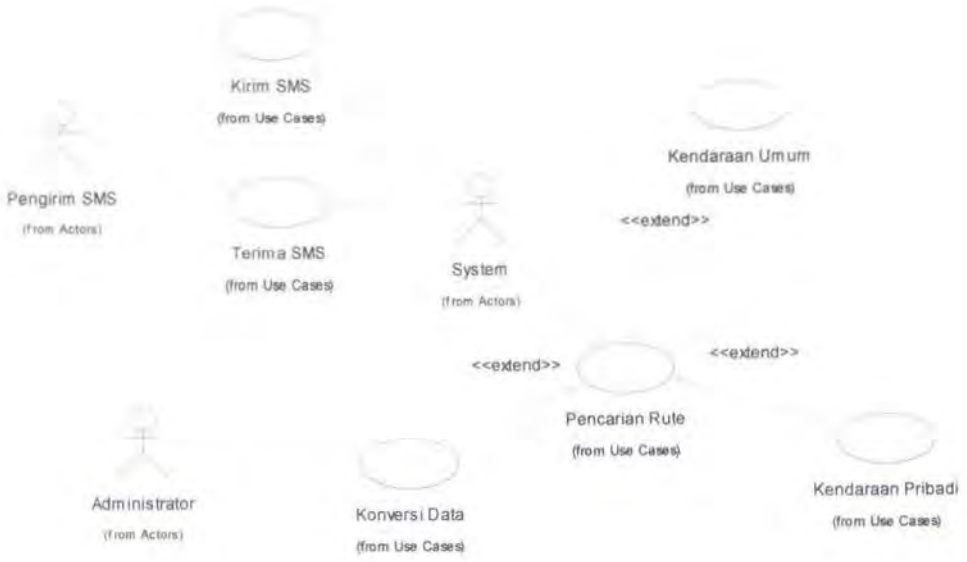
Kelas **TReceiveSMS** adalah kelas untuk mengambil data SMS dari *mobile communication*, kelas **TReceiveSMS** dapat menampung penerimaan SMS secara akumulatif yang dioperasikan oleh kelas **TSMS**.

Gambar III-13 TSMS

3.4 PERANCANGAN PROSES

Dalam perangkat lunak ini, terdapat beberapa proses yang akan dilakukan. Diantaranya adalah:

1. Konversi Data dari data atribut SHP menjadi data *graph* untuk proses pencarian rute.
2. Pencarian rute tercepat.
3. Pengiriman dan Penerimaan SMS.



Gambar III-14 Proses yang Dapat Dilakukan

Untuk lebih jelasnya, proses – proses tersebut akan dibahas berikut ini.

3.4.1 Proses Konversi Data

Konversi data adalah perubahan data dari data atribut SHP menjadi data *graph* yang terdiri dari *node-node* dan *edge-edge*. Konversi data hanya sekali saja

dilakukan ketika data *graph* tidak ada atau belum dikonversi. Data yang akan dikonversi harus mempunyai struktur data sebagai berikut :

Field	Tipe Data	Keterangan
JalanID	Int	Kode Jalan
Nama	String	Nama Jalan
Jenis	Int	Jika jenis=1 maka 1 arah, jika 2 maka 2 arah
Startx_Coord	Double	Koordinat X1
Starty_Coord	Double	Koordinat Y1
EndX_Coord	Double	Koordinat X2
EndY_Coord	Double	Koordinat Y2
Intersect	String	Berisi titik simpang setiap jalan
Simpang	String	Yang menjadi simpangan pada jalan tersebut

Tabel III-1 Data Atribut

Dari tabel tersebut diatas, untuk field *intersect* pengisiannya adalah dengan menjalankan script *avenue* dari perangkat lunak ArcView dan selanjutnya akan dikonversi menjadi data *graph* dan dimasukkan kedalam tabel jalan, adapun perancangan proses konversi dalam bentuk diagram aktivitas adalah sebagai berikut :

Gambar III-15 Diagram Aktivitas Konversi Data

3.4.2 Proses Pencarian Rute

Pencarian rute untuk kendaraan pribadi dan pencarian rute untuk kendaraan umum mempunyai kesamaan proses, perbedaannya pada bobot yang digunakan, jika angkutan umum memperhatikan bobot apakah ada angkutan umum yang

melalui jalan tersebut, sedangkan pada kendaraan pribadi tidak memperhatikan hal tersebut.

Pada perancangan data lokasi dijelaskan bahwa dalam mencari rute tercepat, tidak hanya mencari jalan saja tetapi juga dapat mencari lokasi-lokasi tertentu, juga dijelaskan bahwa setiap *edge* mengandung lokasi-lokasi tertentu, sehingga pencarian antara lokasi dan *edge* merupakan pencarian antara *edge* dan *edge*. Gambar dibawah ini menunjukkan hubungan antara lokasi dan *edge*.

Gambar III-16 Hubungan Antara Edge dan Lokasi

Algoritma untuk pencarian rute yang digunakan adalah algoritma *Dijkstra*, karena algoritma tersebut sesuai diimplementasikan sebagai algoritma pencarian rute terpendek wilayah Surabaya dan bobot yang digunakan adalah jarak. Pencarian rute tercepat untuk sistem ini, bobot jarak tidak dapat diimplementasikan, karena tidak bisa memberikan informasi rute tercepat, hanya untuk rute terpendek. Oleh karena itu, bobot pada rute tercepat digunakan bobot waktu, dimana waktu adalah hasil bagi antara jarak dan kecepatan ($t = S/v$). Jarak adalah jarak jalan, sedangkan kecepatan adalah kecepatan rata-rata pada tingkat kemacetan tertentu. Secara umum setiap jalan tidak mempunyai tingkat kemacetan(*kemacetan* '0'), tetapi pada jam-jam tertentu, tingkat kemacetan suatu

jalan dapat bertambah atau berkurang. Hal ini akan mempengaruhi kecepatan kendaraan.

3.4.2.1 Pencarian Rute Menggunakan Kendaraan Pribadi

Pencarian rute menggunakan kendaraan pribadi dijadikan sebagai standar untuk mencari rute dalam perangkat lunak ini. Dalam mencari rute *node* sumber dan *node* tujuan harus didefinisikan terlebih dahulu dan semua *edge* sudah tersimpan di dalam objek **TEdges**.

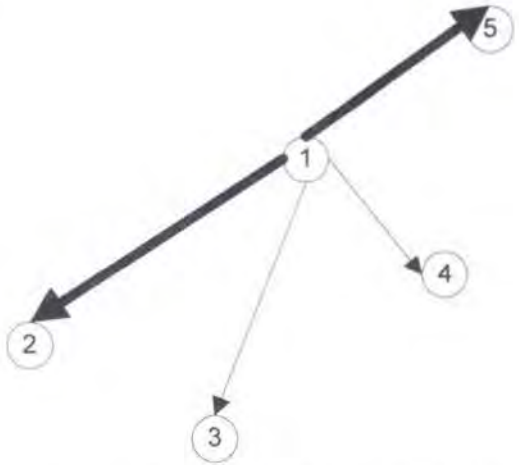
Langkah awal dari pencarian rute adalah inisialisasi awal objek pemindai permanen untuk *edge-edge*, inisialisasi tersebut *node* awal sebagai *node* inisialisasi kemudian inisialisasi bobot dengan bobot tidak terhingga, selanjutnya dilakukan pemindaian *edge* yang menjadikan *node* awal sebagai *node* sumber, hasil *edge* yang telah dipindai dicari bobot yang paling kecil dimasukkan ke dalam pemindai permanen dan *edge* yang lainnya dimasukkan ke dalam pemindai sementara, kemudian pencarian total bobot terkecil antara pemindai sementara dan pemindai permanen, hasil bobot terkecil dipindah ke pemindai permanen dan hasil dari pemindai permanen sebelumnya dipindahkan ke dalam pemindai sementara, hal ini dilakukan terus menerus sampai *node* yang dipindai adalah *node* akhir, jika pencarian rute ketemu maka hasilnya dipindah terlebih dahulu ke dalam rute sementara, langkah selanjutnya bobot dari rute sementara dibandingkan dengan pemindai sementara, jika ternyata bobot rute sementara lebih kecil maka *edge-edge* yang berada pada pemindai sementara dihapus untuk optimasi pencarian karena *edge-edge* tersebut sudah tidak diperlukan lagi,

langkah-langkah diatas dilakukan terus-menerus hingga *node* yang berada pada pemindai permanen kosong. Untuk lebih jelasnya, dapat dilihat pada diagram aktivitas dibawah ini:

Gambar III-17 Diagram Aktivitas Pencarian Rute

3.4.2.2 Pencarian Rute Menggunakan Angkutan Umum

Pencarian untuk rute angkutan umum yang diutamakan adalah pencarian rute tercepat bukan biaya minimal penggunaan angkutan umum. Langkah pencarian rute tersebut sama dengan pencarian rute untuk angkutan pribadi, perbedaannya yaitu dalam pemindaian *edge* yang objek angkutan umumnya ada, jika dalam *edge* tersebut tidak terdapat objek angkutan, maka *edge* tersebut tidak dianggap sebagai *edge*.



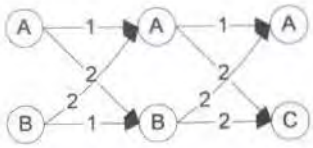
Gambar III-18 Edge untuk Angkutan Umum

Misalkan *edge-edge* pada gambar di atas adalah hasil pemindaian *edge*, karena *edge* 1-2 dan *edge* 1-5 terdapat angkutan umum, maka *edge* tersebut adalah *edge* yang dianggap sedangkan *edge* 1-3 dan *edge* 1-4 dianggap tidak ada.

Gambar III-19 Diagram Aktivitas Pencarian Rute

Langkah selanjutnya adalah mencari angkutan minimal yang akan digunakan, untuk mencari angkutan minimal menggunakan algoritma dijkstra. Penentuan *edge-edge* yang akan digunakan adalah dengan memberi bobot nilai 1

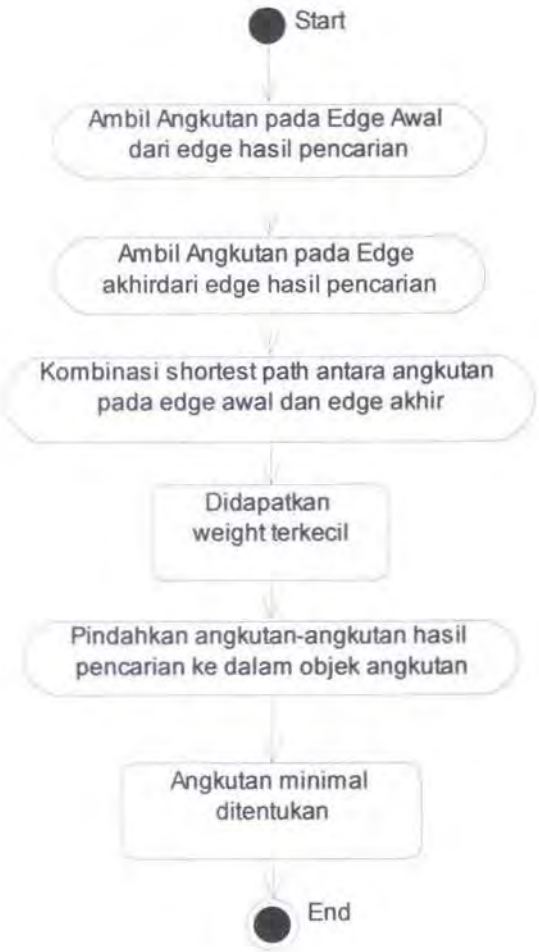
untuk *node* awal dan *node* akhir adalah sama, dan bobot dengan nilai 2 jika *node* awal dan *node* akhir berbeda. *Node* dari pencarian angkutan minimal tersebut merupakan gambaran dari angkutan umum.



Gambar III-20 Penentuan Bobot Pada Edge

Gambar III-21 Diagram aktivitas untuk Membuat Edge

Misalkan *node* A adalah angkutan A, *node* B adalah angkutan B dan *node* C adalah angkutan C, maka minimal angkutan yang akan diperoleh adalah angkutan A saja, karena hanya membutuhkan 1 angkutan(total bobot terkecil).



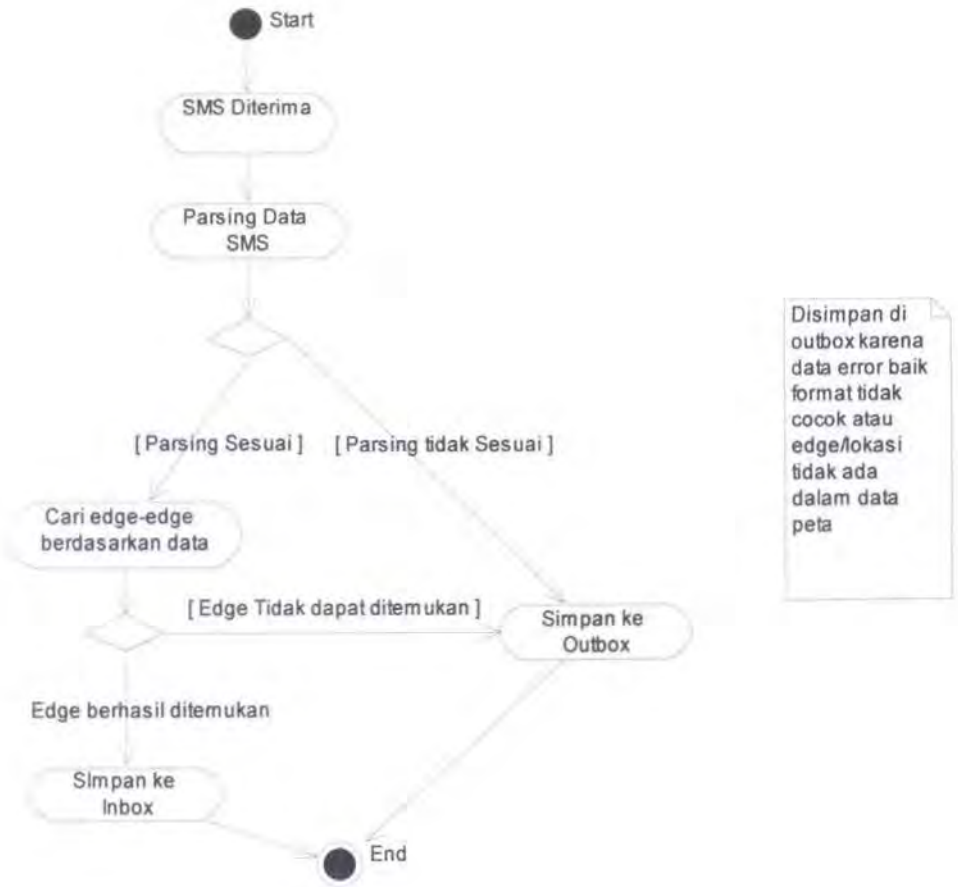
Gambar III-22 Diagram Aktivitas untuk Pencarian Angkutan Minimal

3.4.3 Perancangan Proses Pengiriman dan Penerimaan SMS

Proses pengiriman dan penerimaan SMS dengan mengecek data-data SMS yang masuk atau yang akan dikirim. Setiap data dari tabel Inbox setelah dilakukan pencarian rute akan dimasukkan ke dalama tabel outbox. Proses pengiriman dijalankan berdasarkan data dari tabel outbox dari status yang belum dikirim, proses tersebut dilakukan secara terus-menerus dengan mengecek tabel *outbox*. Untuk proses penerimaan SMS, setiap data yang masuk melalui SMS

akan di*parsing* terlebih dahulu jika data tersebut tidak sesuai format maka pesan error akan dimasukkan ke tabel outbox supaya error tersebut dikirimkan ke pengguna, jika format sesuai dan edge atau lokasi ada dalam database, maka data edge atau lokasi tersebut akan di*queue* ke dalam tabel inbox untuk dilakukan proses pencarian rute. Diagram aktivitas dari kedua proses tersebut adalah sebagai berikut :

Gambar III-23 Diagram Aktivitas Proses Pengiriman SMS



Gambar III-24 Diagram Aktivitas Proses Penerimaan SMS

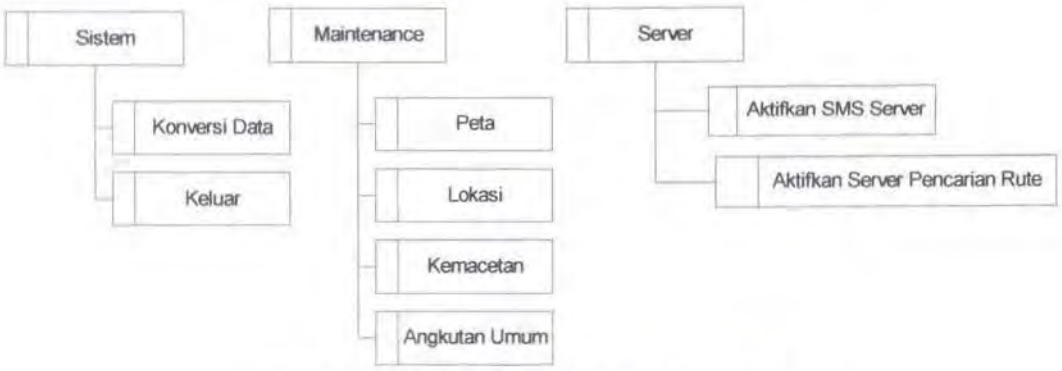
3.5 Perancangan Antar Muka

Interaksi antara pengguna terbagi menjadi dua, pengguna sebagai administrator terdapat aplikasi desktop, sedangkan dari pengguna menggunakan SMS sebagai media interaksi antara server dengan klien. Interaksi kedua pengguna tersebut akan dijelaskan satu-persatu.

3.5.1 Interaksi Administrator

Interaksi Administrator dengan perangkat lunak menggunakan *mouse* dan beberapa tombol pada *Keyboard*. Interaksi ini sebagian besar adalah untuk

melakukan proses konversi data, *maintenance* data dan mengaktifkan sevrer pencarian rute dan server SMS. interaksi-interaksi yang dapat dilakukan administrator dapat digambarkan dari *drop down menu* sebagai berikut :



Gambar III-25 Rancangan dari Dropdown Menu

Rancangan menu di atas adalah rancangan menu pada aplikasi desktop yang akan dioperasikan oleh administrator, menu sistem mempunyai sub menu konversi data untuk mengkonversi data dari data attribut SHP menjadi data *graph* dan sub menu keluar untuk keluar dari aplikasi.

Menu *Maintenance* terdiri dari sub menu peta untuk me*maintenance* data peta, maintenance peta tersebut untuk mengubah informasi jalan. Sub menu lokasi untuk me*maintenance* lokasi yang ada pada edge tertentu, lokasi tersebut dapat ditambah,edit dan hapus. Submenu kemacetan untuk me*maintenance* tingkat kemacetan pada edge tertentu pada saat tertentu, tingkat kemacetan dapat ditambah, edit dan hapus. Dan yang terakhir adalah sub menu angkutan untuk me*maintenance* angkutan yang ada pada edge-edge tertentu,*maintenance* dari angkutan yaitu menentukan rute dari angkutan.

Menu server untuk mengaktifkan server, baik server untuk SMS dan server untuk pencarian rute.

3.5.2 Interaksi Pengguna

Seorang pengguna hanya dapat mengakses server melalui SMS, maka format bahasa dan hasil dari pencarian harus dirancang terlebih dahulu, untuk pengiriman dari seorang pengguna adalah sebagai berikut :

(SUMBER) <SPASI> KE <SPASI> (TUJUAN) <SPASI>
KND <SPASI> (U/P) [<SPASI>JAM<SPASI>(JAM)]

Gambar III-26 Format SMS Pencarian untuk Pengguna

Sumber menjelaskan sumber lokasi atau jalan dari pengguna dan tujuan adalah lokasi atau jalan yang ingin dicari pengguna dan KND menjelaskan kendaraan yang dipakai, jika jenis kendaraan tidak disertakan dalam pengitiman SMS maka kendaraan yang digunakan adalah kendaraan pribadi dan jam adalah waktu pengguna ingin melakukan pencarian ke lokasi atau jalan tersebut, secara default jam akan mengambil data jam pengiriman SMS.

Format SMS untuk hasil dari pencarian rute oleh pengguna yang menjelaskan simbol-simbol untuk berbelok dan naik turun angkutan umum serta arah-arah pada jalan adalah sebagai berikut:

Tikungan dan naik turun

- Belok Kanan =>
- Belok Kiri <=
- Lurus /\
- Balik \/
- Turun \/
- Naik /\

Arah

- Utara=U
- Selatan=S
- Timur=T
- Barat=B
- Timur Laut=TL
- Tenggara=TG
- Barat Laut=BL
- Barat Daya=BD

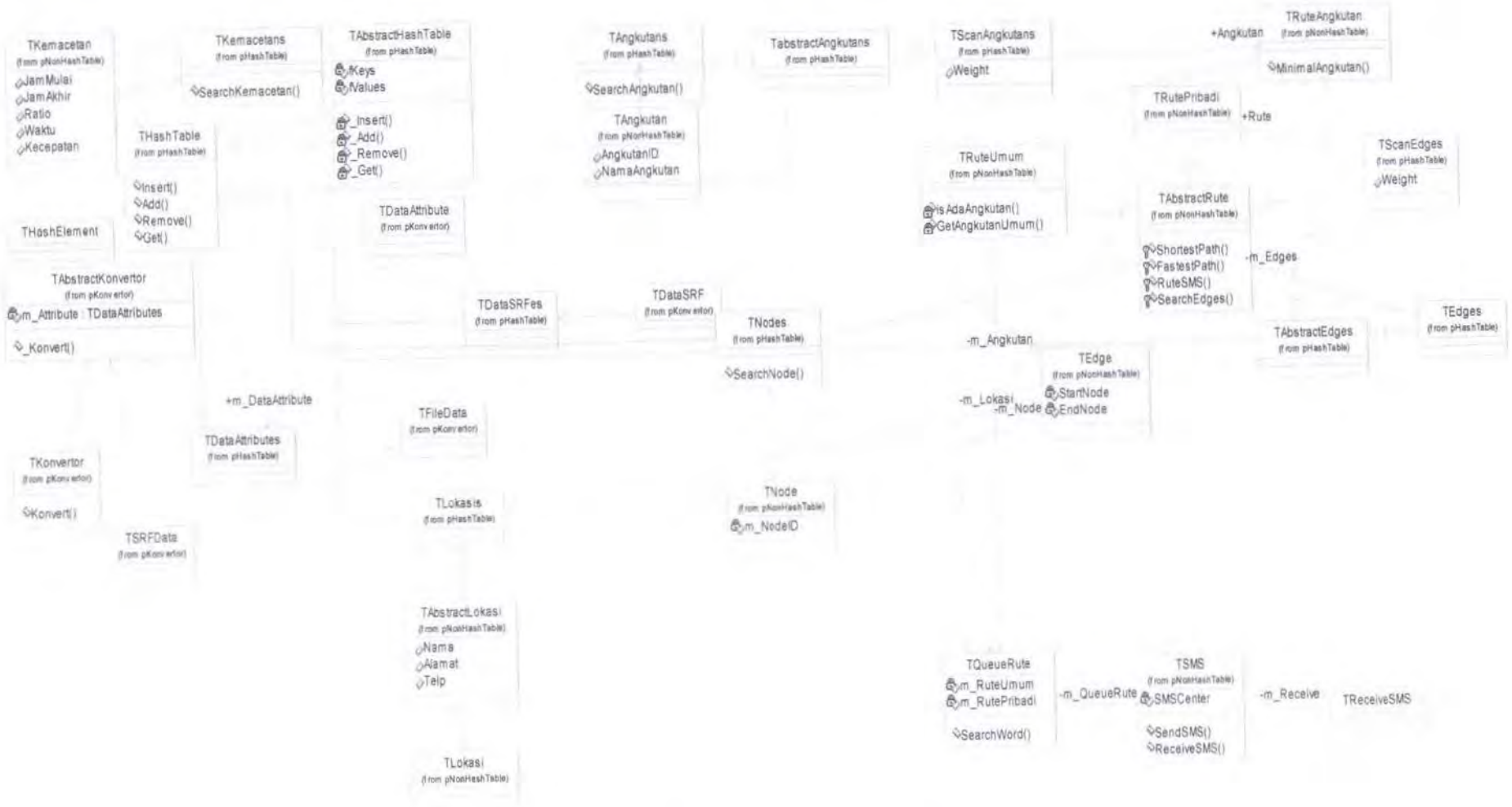
Misalkan dari hasil rute yang dicari adalah sebagai berikut:

/\ Mulyorejo(T)
=> dr KampusC UNAIR ke Kt. Jaya Indah Tmr
<= dr Gdg Mobile8 ke Kt. Jaya Indah
<= BndaranITS ke Raya ITS

Arti dari SMS di atas adalah, dari mulyorejo lurus ke arah timur, jika ada kampus C maka belok kiri menuju kertajaya indah timur, jika di kertajaya indah timur terdapat Mobile8 maka belok kiri menuju kertajaya indah, dan jika di kertajaya indah terdapat bundaran ITS, maka belok kiri menuju jalan raya ITS.

/\ Lyn T2 dr Mulyorejo
\/ Mulyosari
/\ Lyn P(gebang) TRN Raya ITS

Dari mulyorejo naik angkutan umum T2 dan turun di mulyosari dan ganti lin P(Gebang) menuju raya ITS dan turun di jalan raya ITS.



Gambar III-27 Rancangan Aplikasi Keseluruhan

BAB IV
IMPLEMENTASI PERANGKAT LUNAK

Bab ini menguraikan tentang implementasi dari rancangan perangkat lunak yang telah dibuat pada bab III. Pembahasan meliputi lingkungan pembangunan perangkat lunak , implementasi basis data , implementasi proses dan implementasi antar muka.

4.1 Lingkungan Pembangunan Perangkat Lunak

Lingkungan pembangunan aplikasi ini meliputi perangkat keras dan perangkat lunak yang digunakan. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pembangunan aplikasi ini dapat dilihat pada tabel dibawah ini.

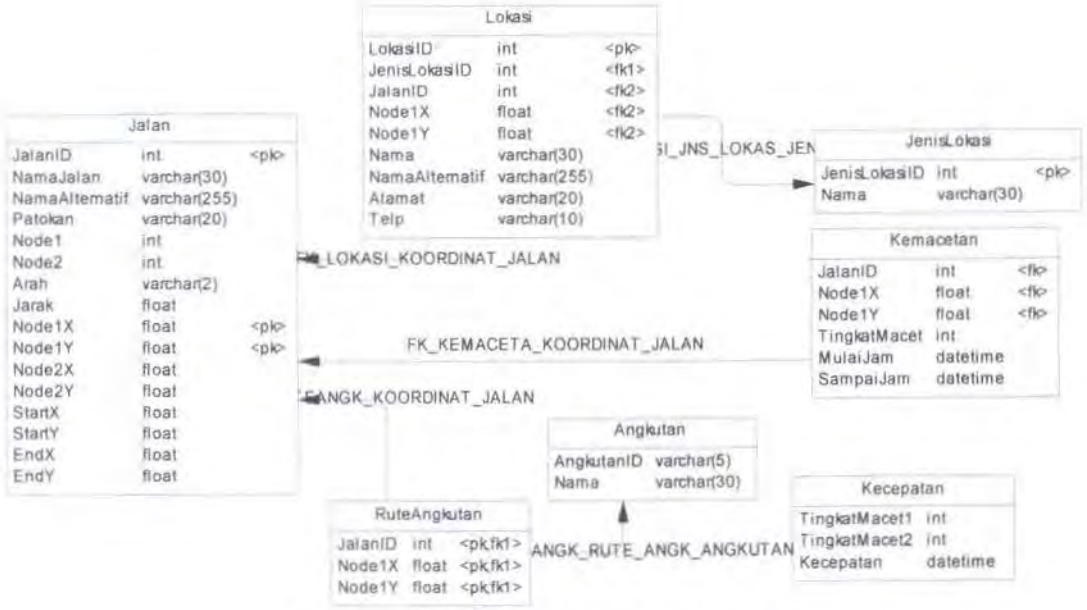
Perangkat Keras	Prosesor	: Intel Pentium IV 2400 MHz
	Memory	: 480 MB
	Modem	: GSM Modem
Perangkat Lunak	Sistem Operasi	: Microsoft Windows XP
	Bahasa Pemrograman	: Pascal
	Compiler & Tools	: Borland Delphi 7.0

Tabel IV-1 Lingkungan Pembangunan Aplikasi

4.2 Implementasi Basis Data

4.2.1 Implementasi Data Atribut

Setelah CDM pada perancangan data dibuat, CDM akan di*generate* menjadi *physical data model* (PDM) sebagai berikut :



Gambar IV-1 Physical Data Model Data Atribut

Penjelasan tabel-tabel di PDM adalah sebagai berikut :

1. Tabel Jalan

Tabel jalan menerangkan tentang informasi jalan tertentu, setiap jalan mempunyai koordinat awal dan koordinat akhir, koordinat tersebut berfungsi sebagai penentu jalan tersebut, jalan juga mengandung informasi arah dan pedoman, informasi tersebut berfungsi untuk mengetahui arah jalan beserta pedoman jika seseorang harus berbelok, dan karena sistem ini akan dioperasikan menggunakan SMS, maka terdapat nama alternatif yang berfungsi sebagai nama lain dari suatu jalan, misalnya tambaksari, nama alternatifnya tbsari Informasi kolom pada tabel jalan adalah sebagai berikut :

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
JALANID	int	Y	T	Y
NAMAJALAN	varchar(30)	T	T	T
PATOKAN	varchar(20)	T	T	T
NAMAALTERNATIF	Varchar(255)	T	T	T
NODE1	int	T	T	T
NODE2	int	T	T	T
ARAH	varchar(2)	T	T	T
JARAK	float	T	T	T
NODE1X	float	Y	T	Y
NODE1Y	float	Y	T	Y
NODE2X	float	T	T	T
NODE2Y	float	T	T	T
STARTX	float	T	T	T
STARTY	float	T	T	T
ENDX	float	T	T	T
ENDY	float	T	T	T
LARANGAN	Varchar(255)	T	T	T

Tabel IV-2 Jalan

2. ***Tabel Jenis Lokasi***

Tabel jenis lokasi digunakan sebagai *master* dari tabel lokasi, tabel tersebut untuk mengelompokkan lokasi-lokasi yang ada di dalam basis data, misal: jenis lokasi untuk kampus, mall, rumah sakit dan lain-lain. Informasi kolom pada tabel jenis lokasi adalah sebagai berikut :

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
JENISLOKASIID	int	Y	T	Y
NAMA	varchar(30)	T	T	T

Tabel IV-3 JenisLokasi

3. ***Tabel Lokasi***

Tabel lokasi menyimpan informasi lokasi-lokasi tertentu dari setiap jalan. Informasi tersebut meliputi nama lokasi, alamat dan nomor telepon, seperti tabel jalan lokasi juga mempunyai nama alternatif yang

mempunyai fungsi sama dengan fungsi nama alternatif pada tabel jalan. Informasi kolom pada tabel lokasi adalah sebagai berikut :

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
LOKASIID	int	Y	T	Y
JENISLOKASIID	int	T	Y	T
JALANID	int	T	Y	T
NODE1X	float	T	Y	T
NODE1Y	float	T	Y	T
NAMA	varchar(30)	T	T	T
ALAMAT	varchar(20)	T	T	T
NAMAALTERNATIF	varchar(255)	T	T	T
TELP	varchar(10)	T	T	T

Tabel IV-4 Lokasi

4. *Tabel Angkutan*

Tabel angkutan menjelaskan informasi angkutan umum meliputi mikrolet dan bus kota. Informasi kolom pada tabel angkutan adalah sebagai berikut :

Field	Tipe Data	Primary key	Foreign Key	Mandatory
ANGKUTANID	varchar(5)	Y	T	Y
NAMA	varchar(30)	T	T	T

Tabel IV-5 Angkutan

5. *Tabel Rute Angkutan*

Tabel rute angkutan menjelaskan tentang rute yang akan dilalui oleh angkutan umum, rute tersebut disimpan dalam koordinat-koordinat dari jalan yang dilalui. Informasi kolom pada tabel rute angkutan adalah sebagai berikut :

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
JALANID	int	Y	Y	Y
NODE1X	float	Y	Y	Y
NODE1Y	float	Y	Y	Y
ANGKUTANID	varchar(5)	Y	Y	Y

Tabel IV-6 RuteAngkutan

6. *Tabel Kemacetan*

Tabel kemacetan menjelaskan tentang kemacetan pada jalan tertentu dan pada saat-saat tertentu, misalnya jalan A antara jam 16:00 – jam 19:00 mempunyai tingkat kemacetan tertentu, fungsi dari tingkat kemacetan tersebut untuk mencari kecepatan rata-rata pada jalan yang dilalui, kecepatan rata-rata didapatkan dari tabel kecepatan. Informasi kolom pada tabel kemacetan adalah sebagai berikut :

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
JALANID	int	T	Y	T
NODE1X	float	T	Y	T
NODE1Y	float	T	Y	T
TINGKATMACET	int	T	T	T
MULAIJAM	datetime	T	T	T
SAMPAIJAM	datetime	T	T	T

Tabel IV-7 Kemacetan

7. *Tabel Kecepatan*

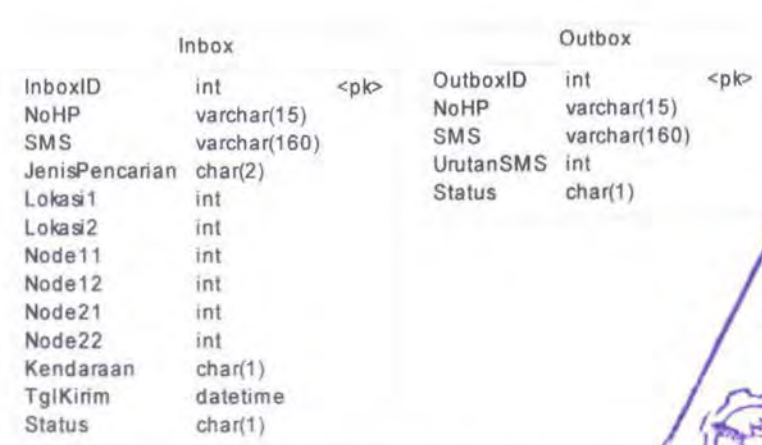
Tabel kecepatan merupakan informasi kecepatan rata-rata pada tingkat kemacetan tertentu. Informasi kolom pada tabel kecepatan adalah sebagai berikut :

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
TINGKATMACET1	int	T	T	Y
TINGKATMACET2	int	T	T	Y
KECEPATAN	datetime	T	T	T

Tabel IV-8 Kecepatan

4.2.2 Implementasi Data Pengolahan SMS

Setelah CDM tersebut selesai didesain, dengan menggunakan power designer 9, CDM akan di*generate* menjadi *physical data model* (PDM) sebagai berikut :



Gambar IV-2 PDM Pengolahan SMS

Penjelasan tabel pada PDM adalah sebagai berikut :

1. ***Tabel Inbox***

   Tabel inbox berfungsi sebagai penyimpan SMS yang diterima oleh *server*, setiap SMS yang diterima akan diproses sebelumnya untuk mengetahui permintaan yang diinginkan oleh pengirim, kemudian hasil dari SMS dimasukkan ke dalam tabel inbox. Informasi kolom pada tabel jalan adalah sebagai berikut :

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
INBOXID	Int	Y	T	Y
NOHP	varchar(15)	T	T	T
SMS	varchar(160)	T	T	T
JENISPENCARIAN	char(2)	T	T	T
LOKASI1	int	T	T	T
LOKASI2	int	T	T	T
NODE11	int	T	T	T
NODE12	int	T	T	T
NODE21	int	T	T	T
NODE22	int	T	T	T

KENDARAAN	char(1)	T	T	T
TGLKIRIM	datetime	T	T	T
STATUS	char(1)	T	T	T

Tabel IV-9 Inbox

2. *Tabel Outbox*

Tabel outbox berfungsi sebagai *queue* untuk pengiriman hasil rute yang telah diminta kepada penyimpan SMS yang diterima oleh *server*, setiap hasil proses pencarian terlebih dahulu disimpan ke dalam tabel outbox, kemudian *server* SMS akan mengirimkan hasil pencariannya ke pengguna yang meminta informasi. Informasi kolom pada tabel jalan adalah sebagai berikut:

Field	Tipe Data	Primary Key	Foreign Key	Mandatory
INBOXID	Int	Y	T	Y
NOHP	varchar(15)	T	T	T
SMS	varchar(160)	T	T	T
JENISPENCARIAN	char(2)	T	T	T
LOKASI1	int	T	T	T
LOKASI2	int	T	T	T
NODE11	int	T	T	T
NODE12	int	T	T	T
NODE21	int	T	T	T
NODE22	int	T	T	T
KENDARAAN	char(1)	T	T	T
TGLKIRIM	datetime	T	T	T
STATUS	char(1)	T	T	T

Tabel IV-10 Outbox

4.3 Implementasi Struktur Data

Pada sub bab ini akan dijelaskan mengenai implementasi dari perancangan data yang telah dibuat pada bab III. Masing – masing kelas akan diperlihatkan atribut yang dimiliki dan fungsi untuk masing – masing atribut.

4.3.1 Kelas TAbstractHashtable dan THashTable

```
type TAbstractHashtable = class(TObject)
      fkeys     : TStrings;
      fvalues   : TList;
end;
type THashElement = class(TObject)
end;
```

Pada kelas TAbstractHashtable terdapat dua variable yaitu fkeys dan fvalues, fkeys berfungsi sebagai penyimpan index dari objek yang disimpan sedangkan fvalues menyimpan objek yang akan disimpan, seperti dijelaskan pada perancangan data bahwa objek yang akan disimpan akan dikonversi terlebih dahulu menjadi kelas THashElement.

```
type THashtable = class(TAbstractHashtable)
end;
```

Kelas THashtable adalah kelas *collection* digunakan untuk menyimpan objek-objek dari kelas yang lain baik sejenis maupun tidak sejenis. pada kelas ini disediakan fungsi-fungsi menambah, mencari, menghapus dan mensisipkan objek untuk mempermudah proses perhitungan daripada menggunakan array.

4.3.2 Kelas TEdge, TEdges dan TScanEdges

```
Type TEdge=class
       StartNode:TNode;
       EndNode:TNode;
       Jarak:double;
       Jalan:String;
       Patokan:String;
       Arah:String[2];
       Kemacetan:TKemacetans;
       Lokasi:TLokasis;
       Angkutan:Tangkutans;
       NamaAlternatif:string;
       Larangan:TEdges;
end;
```

Kelas TEdge adalah kelas yang menerangkan sebuah *edge*, Pada kelas ini juga terdapat kelas-kelas yang menerangkan kemacetan, lokasi atau angkutan

umum, juga informasi *node-node* yang menghubungkannya menjadi suatu *edge*. Pada kelas TEdge terdapat variabel larangan, variabel tersebut menyimpan *edge-edge* yang tidak diperbolehkan belok.

```
Type TEdges=class(TAbstractEdges)

end;
```

Kelas TEdges merupakan kelas turunan dari TAbstractEdges, kelas ini berfungsi sebagai penyimpan objek dari kelas TEdge karena kelas TAbstractEdges adalah kelas *collection*, TEdges juga berfungsi untuk mencari lokasi, tingkat kemacetan dan angkutan umum yang berada pada *edge* tertentu

```
Type TScanEdges=class(TAbstractEdges)
  Bobot:Double;
 end;
```

Kelas TScanEdges merupakan kelas turunan dari TAbstractEdges, pada kelas ini terdapat bobot yang berfungsi untuk menentukan bobot pada hasil pencarian rute, kelas ini juga untuk pemindai sementara dan pemindai permanen.

4.3.3 Kelas TNode dan TNodes

```
Type TNode=class
       NodeID:integer;
end;
```

Variabel NodeID merupakan variabel untuk menyimpan index dari node tertentu.

```
Type TNodes=class(THashtable)
end;
```

Kelas TNodes berfungsi sebagai penyimpan *node-node*, pada kelas ini juga dapat mencari *node-node* tertentu yang berada di dalamnya.

4.3.4 Kelas TLokasi dan TLokasis

```
Type TAbstractLokasi=class
public
  JenisLokasi:String;
  LokasiID:Integer;
  Nama:String;
  Alamat:String;
  Telp:String;
  NamaAlternatif:string;
  Node1,Node2:integer;
end;
Type TLokasi=class(TAbstractLokasi);
```

Kelas TLokasi menyimpan informasi-informasi lokasi tertentu, pada TLokasi terdapat variabel Node1 dan Node2 berguna untuk menjelaskan bahwa lokasi tersebut berada antara Node1 dan Node2.

```
Type TLokasis=class
End;
```

Lokasi dari kelas TLokasi akan disimpan ke dalam kelas TLokasis, pada kelas ini terdapat fungsi untuk mencari lokasi yang ada pada kelas tersebut.

4.3.5 Kelas TKemacetan dan TKemacetans

```
Type TKemacetan=class
  MulaiJam:TDateTime;
  SampaiJam:TDateTime;
  Ratio:integer;
  Kecepatan:double;
  Waktu:double;
end;
```

Variabel Waktu adalah hasil bagi antara variabel Jarak dari edge dan kecepatan berdasarkan Ratio tertentu, variabel tersebut akan digunakan untuk bobot pada pencarian rute. Setiap kecepatan mempunyai waktu tertentu, ini dijelaskan pada variabel MulaiJam yang menjelaskan waktu awal kemacetan dan SampaiJam menjelaskan waktu akhir kemacetan.

```
Type TKemacetans=class(THashtable)
end;
```

Kelas TKemacetans untuk menyimpan objek-objek dari kelas TKemacetan, kelas ini dapat mencari kemacetan dengan parameter waktu tertentu.

4.3.6 Kelas TAngkutan, TAngkutans dan TScanAngkutans

```
Type TAngkutan=class
  AngkutanID:string;
  Nama:string;
  Keterangan:String;
end;
```

Kelas TAngkutan menyimpan informasi-informasi Angkutan tertentu, objek dari kelas ini berada pada objek kelas **TEdge**.

```
Type TAngkutans =class(THashtable)
end;
```

Kelas TAngkutans menyimpan objek dari kelas TAngkutan, kelas ini dapat mencari angkutan berdasarkan kode dari angkutan.

```
Type TScanAngkutans=class(TAbstractAngkutans)
private
    Weight:Integer;
end;
```

Kelas TScanAngkutans digunakan untuk memindai pencarian untuk mendapatkan angkutan minimal, untuk mengetahui apakah angkutan tersebut adalah angkutan minimal yang digunakan dapat dilihat pada variabel *weight*, nilai paling kecil dari weight adalah angkutan minimal yang akan digunakan.

4.3.7 Kelas TAbstractKonvertor dan TKonvertor

```
Type TAbstractKonvertor=class
  Attribut:TDataAttributes;
end;

Type TKonvertor=class(TAbstractKonvertor)
end;
```

Kelas TAbstractKonvertor merupakan kelas dasar untuk melakukan konversi data menjadi data *graph*, sedangkan kelas TKonvertor digunakan untuk mengkonversi data atribut SHP menjadi data *graph*.

4.3.8 Kelas TAbstractRute,TRutePribadi,TRuteUmum dan TRuteAngkutan

```
Type TAbstractRute=class
 Edges:TEdges;
end;
```

Kelas TAbstractRute merupakan kelas dasar untuk mencari rute baik menggunakan kendaraan pribadi ataupun angkutan umum, pada kelas tersebut terdapat variabel edges yang menjelaskan bahwa semua *edge* disimpan pada variabel tersebut. Terdapat dua fungsi utama pada kelas tersebut yaitu mencari rute tercepat dan rute terpendek, rute tercepat digunakan untuk mencari rute berdasarkan kendaraan, sedangkan rute terpendek digunakan untuk mencari rute angkutan minimal yang digunakan.

```
Type TRutePribadi=class(TAbstractRute)
end;
```

Kelas TRutePribadi adalah kelas untuk mencari rute menggunakan kendaraan pribadi, fungsi RuteSMS untuk merubah data *edge* menjadi bentuk *string* hasil pencarian.

```
Type TRuteUmum=class(TAbstractRute)
end;
```

Kelas TRuteUmum untuk mencari rute menggunakan kendaraan umum, fungsi RuteSMS pada kelas tersebut berbeda dengan kelas TRutePribadi, hasil dari RuteSMS pada kelas tersebut yaitu hasil rute tercepat dan angkutan umum minimal yang akan digunakan.

4.3.9 Kelas TSMS dan TReceiveSMS

```
Type TSMS=class
  SMSCenter:String;
  DataSMS:TStrings;
end;
```

Kelas TSMS berfungsi sebagai pengirim dan penerima SMS, pada kelas tersebut terdapat variabel SMSCenter digunakan untuk menyimpan SMS *Center* dalam pengiriman SMS dan dalam penerimaan SMS, variabel DataSMS adalah variabel untuk menyimpan SMS yang diterima, dalam variabel tersebut data penerimaan SMS masih dalam *encode* PDU.

```
type TReceiveSMS=class
  sms:array of TSMSRec;
end;
```

Untuk menggunakan data pada DataSMS harus diubah terlebih dahulu menjadi kelas TReceiveSMS, kelas tersebut memberikan informasi isi SMS, pengirim dan waktu kirim.

4.4 Implementasi Proses

Pada sub bab ini akan dijelaskan mengenai implementasi dari perancangan proses yang telah dibuat pada bab III. Implementasi ini meliputi implementasi proses konversi data, pencarian rute dan pengiriman atau penerimaan SMS.

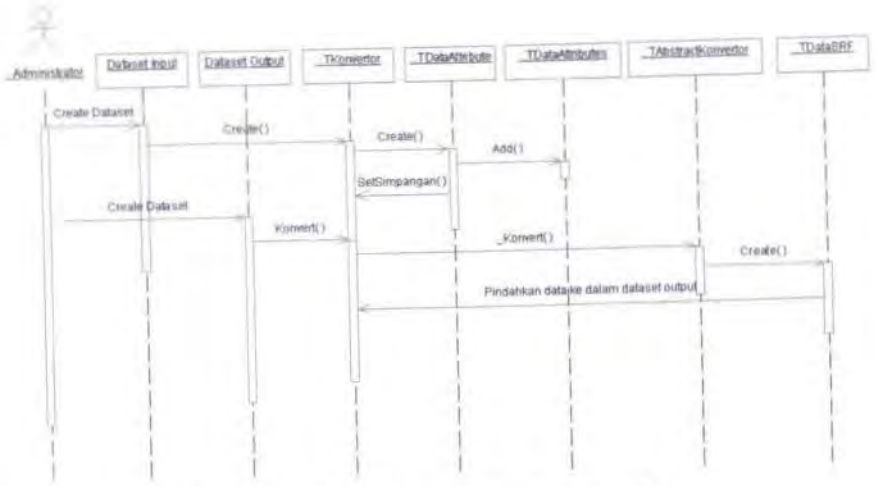
4.4.1 Proses Konversi Data

Sebelum melakukan pengkonversian data, harus menjalankan script *avenue* terlebih dahulu untuk menginputkan nilai pada field *intersect*. Adapun script *avenue* untuk menginput data intersect adalah sebagai berikut :

```
  if (theThemeName = "Jalan") then
        'Check if fields named "Intersects and Simpang" exist
        Intersects_exists = (theFTab.FindField("Intersects") = NIL).Not
        Simpang_exists = (theFtab.FindField("Simpang") = NIL).Not

        if (Intersects_exists or Simpang_exists) then
          if (MsgBox.YesNo("Overwrite field yang ada?",
          "Field Intersects dan Simpang sudah ada!", false)) then
            'if ok to overwrite, delete the fields as they may not be defined
            'as required by this script (eg., created from another script).
            if (Intersects_exists) then
                theFTab.RemoveFields({theFTab.FindField("Intersects")})
            end
            if (Simpang_exists) then
              theFTab.RemoveFields({theFTab.FindField("Simpang")})
            end
          else
            return nil
          end  'if (MsgBox...)
        end  'if

          aNewField = Field.Make ("Intersects", #FIELD_VCHAR, 1000, 0)
          theFtab.addFields({aNewField})
          aNewField = Field.Make ("Simpang", #FIELD_VCHAR, 255, 0)
          theFtab.addFields({aNewField})

       'Check if fields named "X-coord" and Y-coord" exist
        x1_exists = (theFTab.FindField("StartX-coord") = NIL).Not
        y1_exists = (theFtab.FindField("StartY-coord") = NIL).Not
        x2_exists = (theFTab.FindField("EndX-coord") = NIL).Not
        y2_exists = (theFtab.FindField("EndY-coord") = NIL).Not

        if (x1_exists or y1_exists or x2_exists or y2_exists) then
          if (MsgBox.YesNo("Overwrite existing fields?",
          "StartX-coord, StartY-coord or EndX-coord, EndY-coord fields already
exist", false)) then
            'if ok to overwrite, delete the fields as they may not be defined
            'as required by this script (eg., created from another script).
            if (x1_exists) then

              theFTab.RemoveFields({theFTab.FindField("StartX-coord")})
            end
            if (y1_exists) then
              theFTab.RemoveFields({theFTab.FindField("StartY-coord")})
```

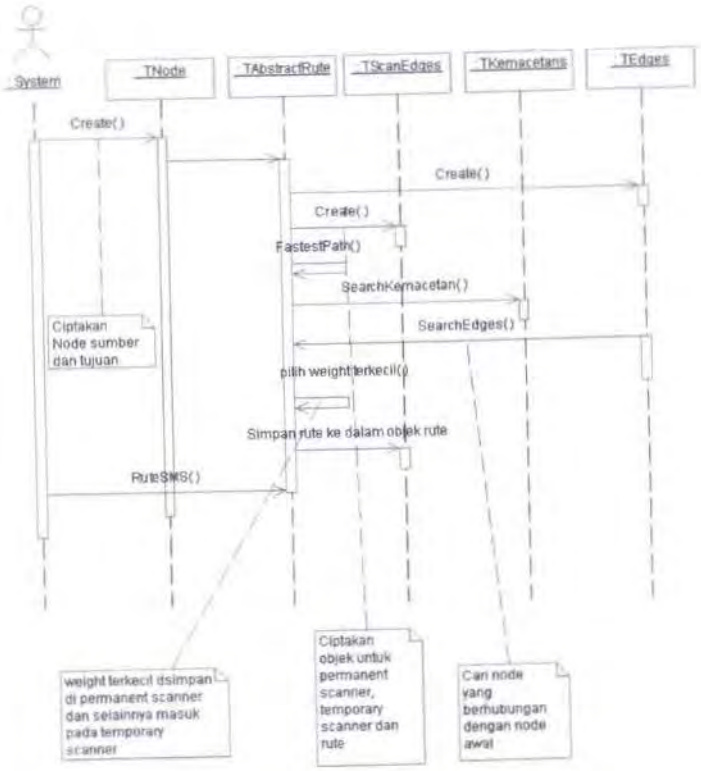
```
            end
            if (x2_exists) then
              theFTab.RemoveFields({theFTab.FindField("EndX-coord")})
            end
            if (y2_exists) then
              theFTab.RemoveFields({theFTab.FindField("EndY-coord")})
            end
          else
            return nil
          end
        end
        x1 = Field.Make ("StartX-coord",#FIELD_LONG,18,0)
        y1 = Field.Make ("StartY-coord",#FIELD_LONG,18,0)
        x2 = Field.Make ("EndX-coord",#FIELD_LONG,18,0)
        y2 = Field.Make ("EndY-coord",#FIELD_LONG,18,0)
        theFTab.AddFields({x1,y1, x2, y2})
    end
```

Setelah field *intersect* diisi, selanjutnya dilakukan konversi. Dalam melakukan konversi, terlebih dahulu harus dibuat data input untuk konversi dan data output untuk hasil konversi, kemudian objek untuk konversi diciptakan. Dalam proses penciptaan objek diperlukan suatu objek dari kelas TDataAttribut, objek dari data input dipindahkan ke objek dari kelas TDataAttribut, setelah pemindahan data selesai, maka dilakukan konversi data, dalam pengkonversian membutuhkan objek dari TDataSRF, setelah melakukan konversi data dipindah ke dalam data output. Untuk lebih jelasnya, dapat digambarkan menggunakan sequence diagram sebagai berikut :



Gambar IV-3 Sequence Diagram Konversi Data

4.4.2 Proses Pencarian Rute

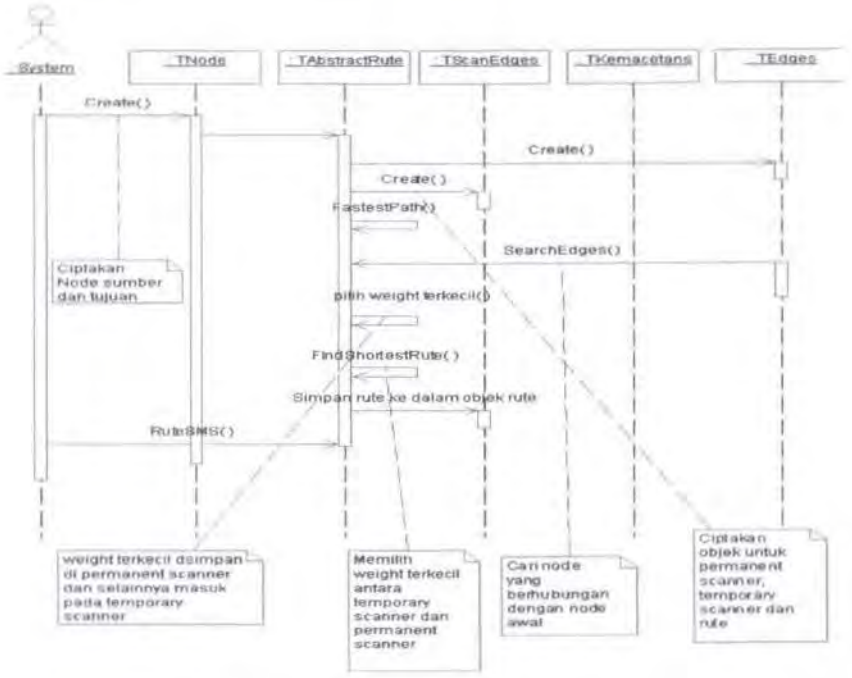


Gambar IV-4 Sequence Diagram Fastest Path

Sebelum melakukan pencarian rute tercepat, *node* sumber dan *node* tujuan harus diciptakan terlebih dahulu, begitu juga objek *edges* yang menentukan jalan-jalan yang ada. Setelah ketiga objek diciptakan, maka pencarian rute dimulai,

dalam pencarian rute langkah awal adalah mencari rute bobot paling kecil yaitu dengan mnecari terlebih dulu tingkat kemacetan pada waktu tertentu, dari hasil ratio kemacetan dan kecepatan dengan jarak untuk menentukan waktu rata-rata tempuh pada jalan tersebut. proses dilakukan terus menerus sampai rute ditemukan atau isi dari objek pemindai permanen kosong.

Untuk mencari rute terpendek, sequence diagramnya hampir sama, hanya pada rute terpendek penentuan tingkat kemacetan tidak dilakukan, *sequence diagram* untuk rute terpendek adalah sebagai berikut:

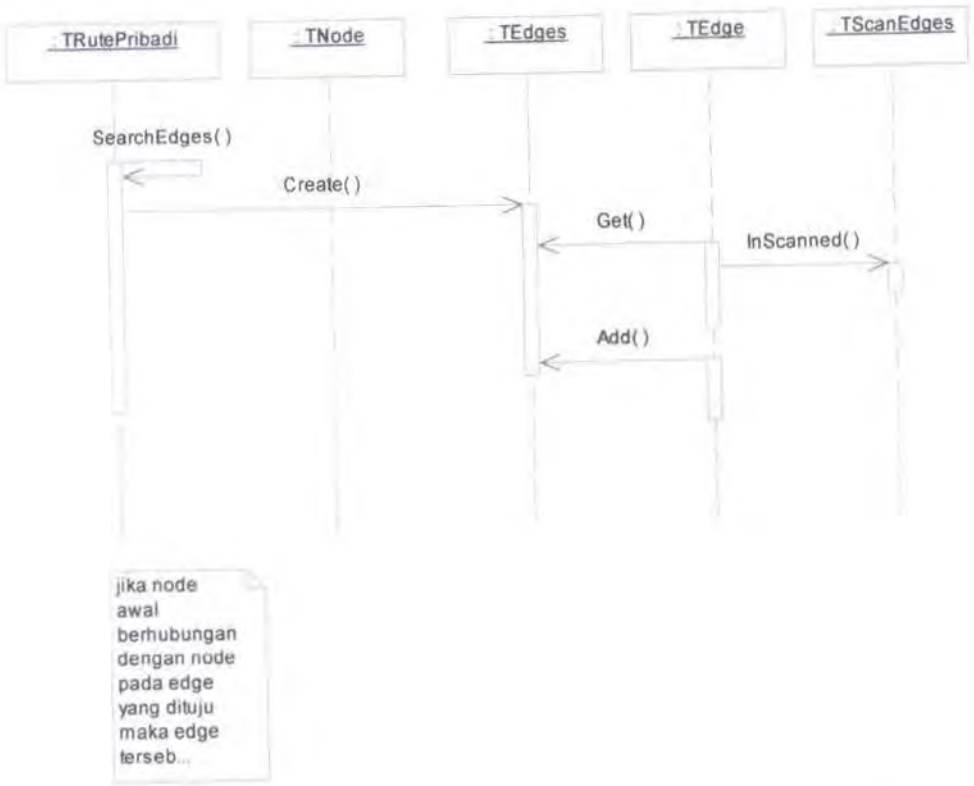


Gambar IV-5 Sequence Diagram Rute Terpendek

4.4.2.1 Pencarian Rute Menggunakan Kendaraan Pribadi

Dalam pencarian menggunakan kendaraan pribadi kelas yang digunakan adalah kelas TRutePribadi yang diturunkan dari kelas *abstract*nya yaitu kelas TAbstractRute, karena metode dari fungsi searchedges masih bersifat abstract,

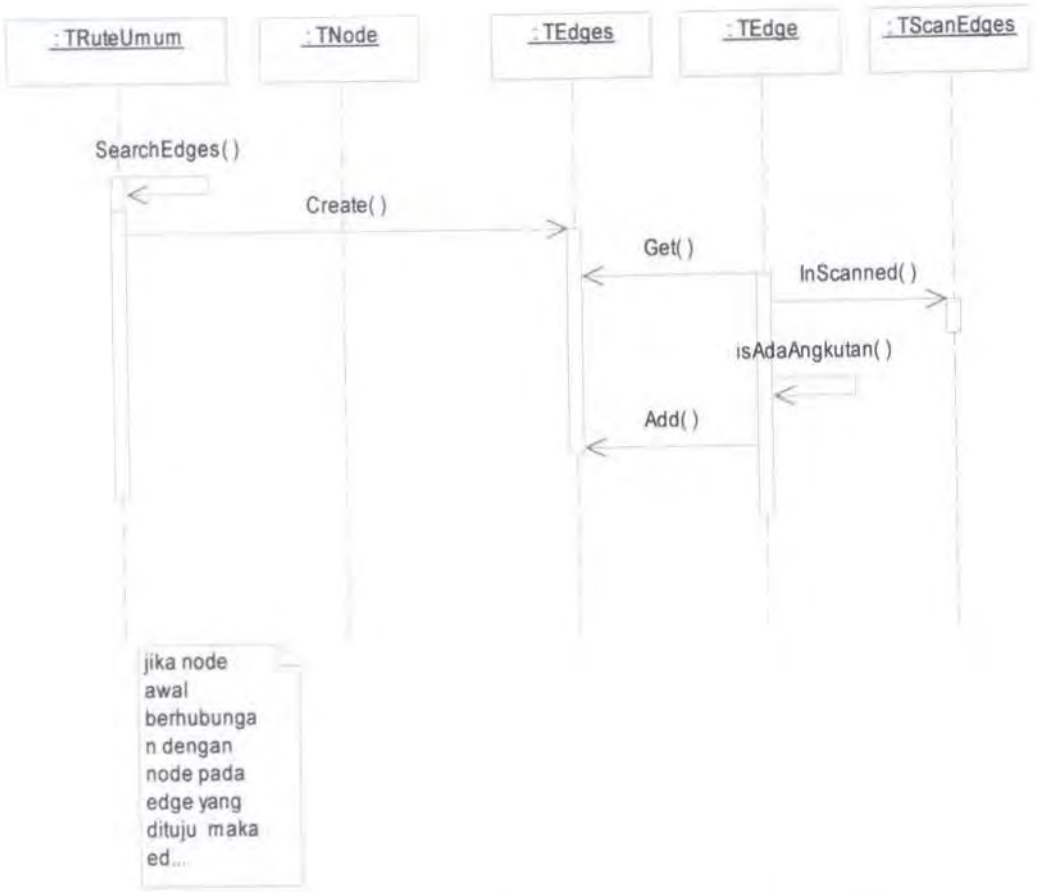
maka fungsi searchedges di*overrride* dengan metode yang baru, *sequence diagram* untuk fungsi searchedges pada kelas TRutePribadi adalah sebagai berikut :



Gambar IV-6 Sequence Diagram Fungsi SearchEdges pada Kelas TRutePribadi

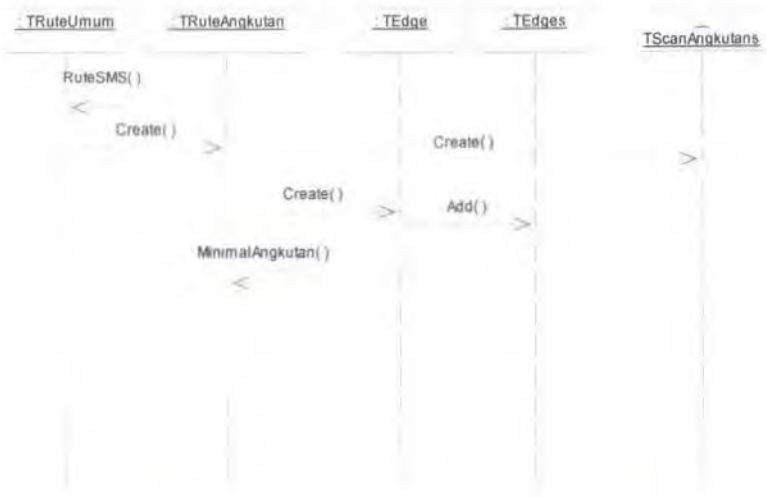
4.4.2.2 Pencarian Rute Menggunakan Angkutan Umum

Untuk mencari rute menggunakan kendaraan umum kelas yang digunakan adalah kelas TRuteUmum, kelas tersebut juga turunan dari kelas TAbstractRute dan harus meng*override* fungsi searchedges pada kelas *abstract*nya perbedaan metode pada fungsi searchedges pada kelas TRuteUmum adalah pada kelas TRuteUmum pada fungsi searchedges dilakukan pengecekan apakah terdapat angkutan umum pada *edge* tersebut.



Gambar IV-7 Sequence Diagram Fungsi SearchEdges pada Kelas TRuteUmum

Dari fungsi searchedges di atas, jika hasil pencarian rute berhasil ditemukan, maka langkah selanjutnya yaitu mencari angkutan minimal yang digunakan. kelas untuk mencari angkutan minimal adalah kelas TRuteAngkutan turunan dari kelas TAbstractRute, *sequence diagram* untuk pencarian angkutan minimal adalah sebagai berikut :

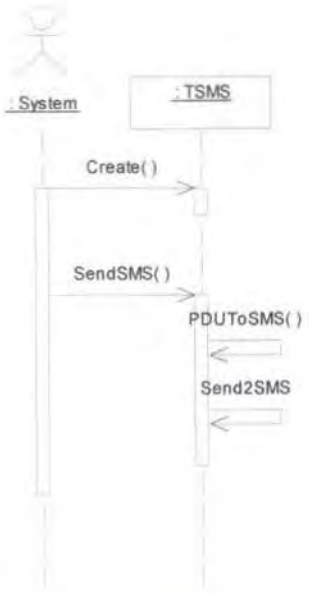


Gambar IV-8 Sequence Diagram Pencarian Angkutan Minimal

Pada gambar di atas, kelas TRute umum menjalankan fungsi RuteSMS untuk mencari angkutan minimal, di dalam fungsi tersebut menciptakan objek dari kelas TRuteAngkutan dalam penciptaan objek tersebut, dibuat edge-edge terlebih dahulu, untuk edge dari angkutan yang sejenis, bobotnya diberi nilai 1 sedangkan untuk yang tidak sejenis diberi nilai 2.

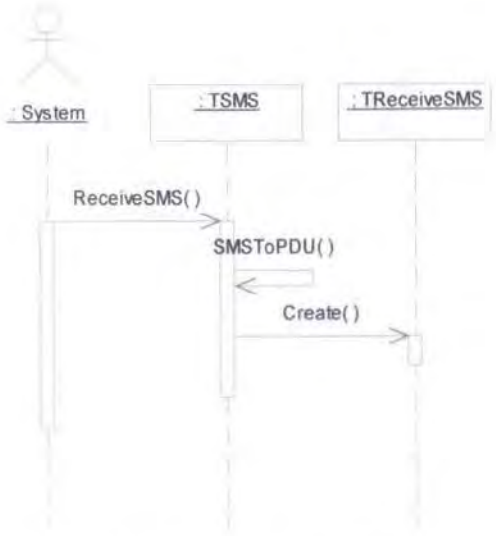
4.4.3 Proses Pengiriman dan Penerimaan SMS

Proses pengiriman SMS menggunakan kelas dari TSMS, pengiriman sms dengan menjalankan fungsi SendSMS, dalam fungsi tersebut pesan SMS di*encode* terlebih dahulu menjadi PDU kemudian dikirimkan melalui *serial port.*



Gambar IV-9 Pengiriman SMS

Proses penerimaan SMS data ditampung kedalam kelas TReceiveSMS, kelas TReceiveSMS dapat menampung banyak SMS



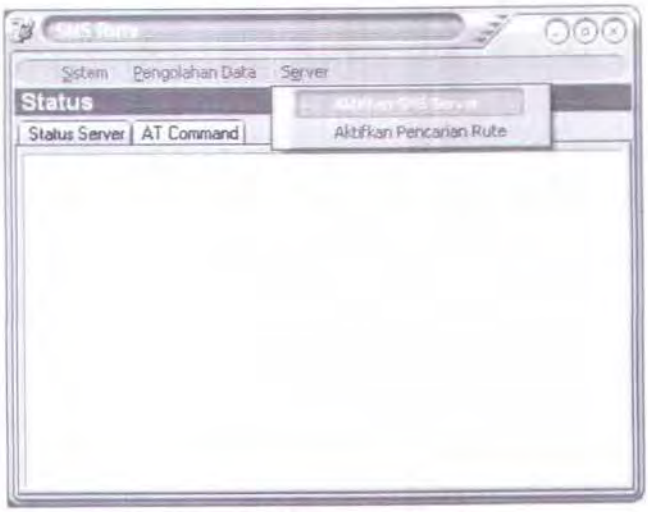
Gambar IV-10 Penerimaan SMS

4.5 IMPLEMENTASI ANTAR MUKA

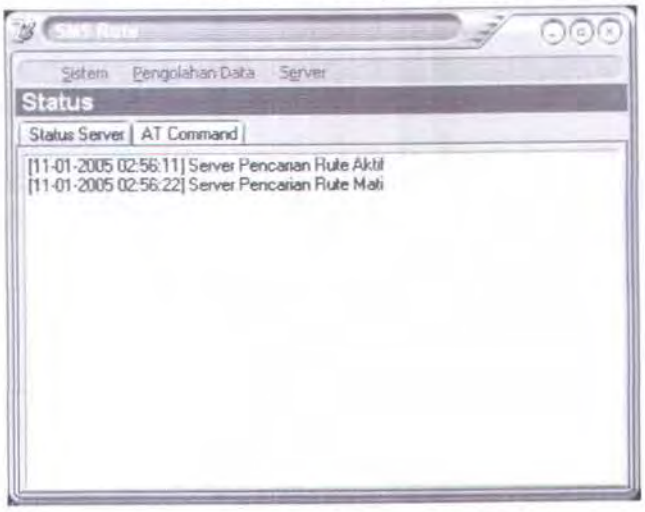
Berikut ini adalah implementasi dari rancangan antar-muka yang telah dibuat pada bab III .

Gambar IV-11 Menu Dropdown Sistem

Gambar IV-12 Menu Dropdown Pengolahan Data



Gambar IV-13 Menu Dropdown Server



Gambar IV-14 Contoh Status Server Aktif

Gambar IV-15 Contoh Pengiriman SMS oleh pengguna

Gambar IV-16 Contoh Penerimaan SMS oleh pengguna

BAB V
UJI COBA DAN EVALUASI

Pada bab ini dibahas mengenai uji coba terhadap aplikasi yang telah dibangun. Uji coba ini, uji coba yang dilakukan yaitu uji coba konversi data, pencarian rute menggunakan kendaraan pribadi maupun angkutan umum, uji coba pencarian rute tercepat dan uji coba pengiriman dan penerimaan SMS. Dan evaluasi dilakukan untuk mendapatkan kebenaran dalam melaksanakan uji coba.

5.1 Lingkungan Pelaksanaan Uji Coba

Pada sub bab ini akan dijelaskan mengenai lingkungan uji coba yang meliputi perangkat keras dan perangkat lunak yang digunakan. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pengujian ini dapat dilihat pada tabel berikut:

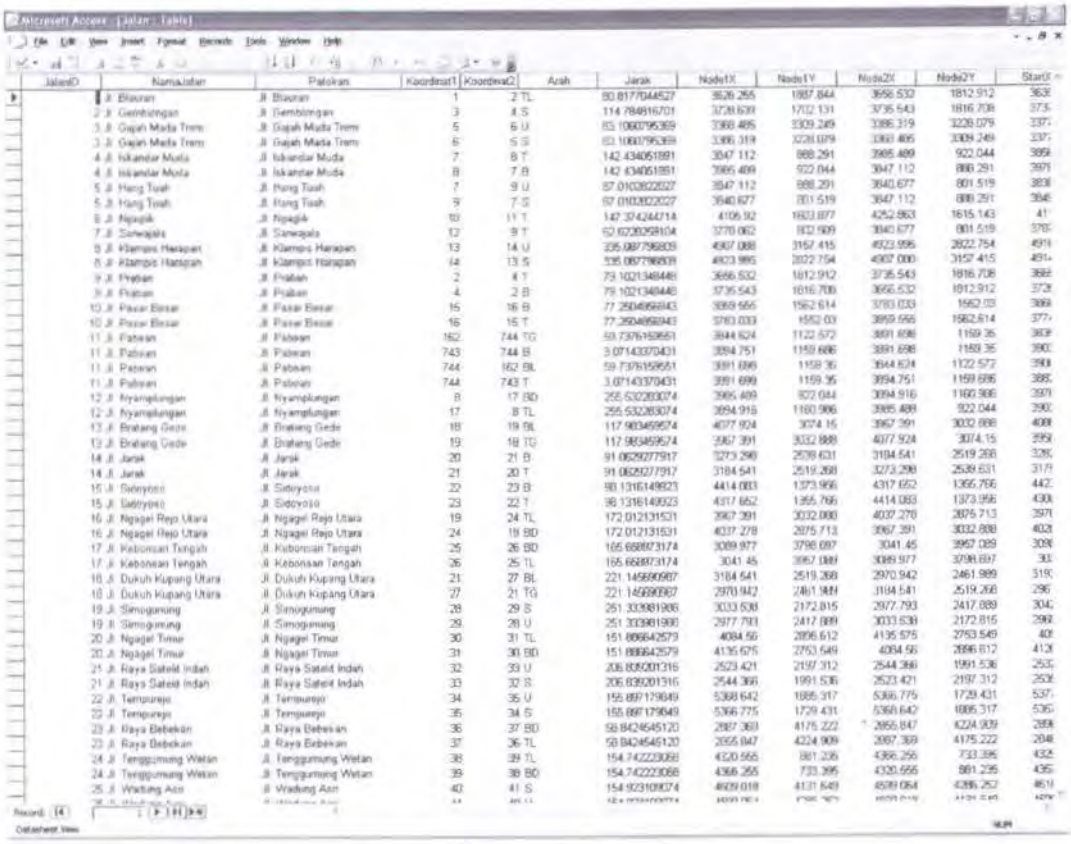
Perangkat Keras	Prosesor	: Intel Pentium IV 2400 MHz
	Memory	: 512 MB
	GSM Modem	: Handphone Siemens MC60
Perangkat Lunak	Sistem Operasi	: Microsoft Windows XP Pro.

Tabel V-1 Lingkungan Pengujian Aplikasi

5.2 PELAKSANAAN UJI COBA

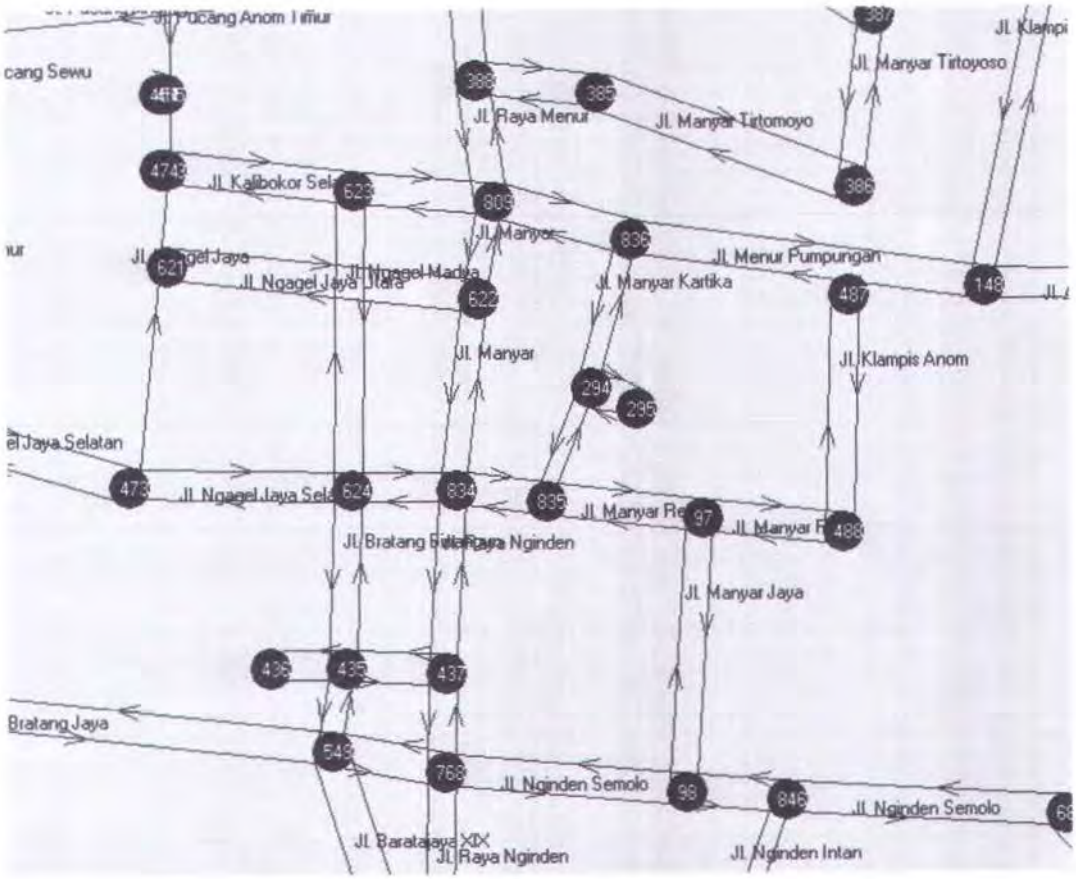
5.2.1 Uji Coba Konversi Data

Hasil konversi data yang dihasilkan berupa data graph dengan node-node yang saling berhubungan sehingga membentuk edge, gambar di bawah ini adalah hasil dari uji coba pengkonversian data.



JalanID	NamaJalan	Palokan	Koordinat1	Koordinat2	Arah	Jarak	Node1X	Node1Y	Node2X	Node2Y	StartX
	Jl. Blaran	Jl. Blaran	1	2 TL		80.8177044527	3626.255	1887.844	3656.532	1812.912	363
	Jl. Gembongan	Jl. Gembongan	3	4 S		114.784816701	3738.639	1702.131	3736.543	1816.708	373
	Jl. Gajah Mada Trem	Jl. Gajah Mada Trem	5	6 U		63.1060795369	3368.485	3309.249	3386.319	3228.079	337
	Jl. Gajah Mada Trem	Jl. Gajah Mada Trem	6	5 S		63.1060795369	3386.319	3228.079	3368.485	3309.249	337
	Jl. Iskandar Muda	Jl. Iskandar Muda	7	8 T		142.434051891	3847.112	888.291	3985.489	922.044	385
	Jl. Iskandar Muda	Jl. Iskandar Muda	8	7 B		142.434051891	3985.489	922.044	3847.112	888.291	397
	Hang Tuah	Jl. Hang Tuah	7	9 U		67.0102822027	3847.112	888.291	3840.677	801.519	383
	Jl. Hang Tuah	Jl. Hang Tuah	9	7 S		67.0102822027	3840.677	801.519	3847.112	888.291	384
	Jl. Ngagik	Jl. Ngagik	10	11 T		147.374244714	4106.92	1603.877	4252.863	1615.143	41
	Jl. Sarwajala	Jl. Sarwajala	12	9 T		62.6238298104	3770.662	802.909	3840.677	801.519	376
	Jl. Kampis Harapan	Jl. Kampis Harapan	13	14 U		335.087796809	4907.088	3157.415	4923.996	2822.754	491
	Jl. Kampis Harapan	Jl. Kampis Harapan	14	13 S		335.087796809	4923.996	2822.754	4907.088	3157.415	491
	Jl. Prabon	Jl. Prabon	2	4 T		79.1021348448	3656.532	1812.912	3736.543	1816.708	366
	Jl. Prabon	Jl. Prabon	4	2 B		79.1021348448	3736.543	1816.708	3656.532	1812.912	372
	Jl. Pasar Besar	Jl. Pasar Besar	15	16 B		77.2604956943	3869.566	1562.614	3783.033	1562.03	386
	Jl. Pasar Besar	Jl. Pasar Besar	16	15 T		77.2604956943	3783.033	1562.03	3869.566	1562.614	377
	Jl. Pabean	Jl. Pabean	162	744 TG		59.7376159651	3844.624	1122.572	3891.696	1159.36	387
	Jl. Pabean	Jl. Pabean	743	744 B		3.07143370431	3894.751	1159.696	3891.696	1159.36	390
	Jl. Pabean	Jl. Pabean	744	162 BL		59.7376159651	3891.696	1159.36	3844.624	1122.572	390
	Jl. Pabean	Jl. Pabean	744	743 T		3.07143370431	3891.696	1159.36	3894.751	1159.696	388
	Jl. Nyamplungan	Jl. Nyamplungan	8	17 BD		255.532283074	3985.489	922.044	3894.916	1160.986	397
	Jl. Nyamplungan	Jl. Nyamplungan	17	8 TL		255.532283074	3894.916	1160.986	3985.489	922.044	390
	Jl. Bratang Gede	Jl. Bratang Gede	18	19 BL		117.983459574	4077.924	3074.15	3967.391	3032.888	408
	Jl. Bratang Gede	Jl. Bratang Gede	19	18 TG		117.983459574	3967.391	3032.888	4077.924	3074.15	396
	Jl. Jarak	Jl. Jarak	20	21 B		91.0629277917	3273.298	2539.631	3184.541	2519.268	326
	Jl. Jarak	Jl. Jarak	21	20 T		91.0629277917	3184.541	2519.268	3273.298	2539.631	317
	Jl. Sidoyoso	Jl. Sidoyoso	22	23 B		98.1316149823	4414.083	1373.956	4317.652	1355.766	442
	Jl. Sidoyoso	Jl. Sidoyoso	23	22 T		98.1316149823	4317.652	1355.766	4414.083	1373.956	430
	Ngagel Rejo Utara	Jl. Ngagel Rejo Utara	19	24 TL		172.012131531	3967.391	3032.888	4037.278	2875.713	397
	Jl. Ngagel Rejo Utara	Jl. Ngagel Rejo Utara	24	19 BD		172.012131531	4037.278	2875.713	3967.391	3032.888	402
	Jl. Kebonsari Tengah	Jl. Kebonsari Tengah	25	26 BD		165.688873174	3089.977	3798.697	3041.45	3967.089	309
	Jl. Kebonsari Tengah	Jl. Kebonsari Tengah	26	25 TL		165.688873174	3041.45	3967.089	3089.977	3798.697	30
	Jl. Dukuh Kupang Utara	Jl. Dukuh Kupang Utara	21	27 BL		221.146866987	3184.541	2519.268	2970.942	2461.989	319
	Jl. Dukuh Kupang Utara	Jl. Dukuh Kupang Utara	27	21 TG		221.146866987	2970.942	2461.989	3184.541	2519.268	296
	Jl. Simogunung	Jl. Simogunung	28	29 S		251.330981986	3033.538	2172.815	2977.793	2417.089	304
	Jl. Simogunung	Jl. Simogunung	29	28 U		251.330981986	2977.793	2417.089	3033.538	2172.815	298
	Ngagel Timur	Jl. Ngagel Timur	30	31 TL		151.886642579	4084.56	2896.612	4135.575	2753.549	409
	Ngagel Timur	Jl. Ngagel Timur	31	30 BD		151.886642579	4135.575	2753.549	4084.56	2896.612	412
	Jl. Raya Satelit Indah	Jl. Raya Satelit Indah	32	33 U		208.895001316	2523.421	2197.312	2544.366	1991.536	253
	Jl. Raya Satelit Indah	Jl. Raya Satelit Indah	33	32 S		208.895001316	2544.366	1991.536	2523.421	2197.312	252
	Jl. Tembureji	Jl. Tembureji	34	35 U		155.897179849	5368.642	1885.317	5366.775	1729.431	537
	Jl. Tembureji	Jl. Tembureji	35	34 S		155.897179849	5366.775	1729.431	5368.642	1885.317	536
	Jl. Raya Bebekan	Jl. Raya Bebekan	36	37 BD		59.8424545120	2987.369	4175.222	2955.847	4224.909	289
	Jl. Raya Bebekan	Jl. Raya Bebekan	37	36 TL		59.8424545120	2955.847	4224.909	2987.369	4175.222	284
	Jl. Tenggumung Wetan	Jl. Tenggumung Wetan	38	39 TL		154.742223068	4320.666	881.236	4366.256	733.396	432
	Jl. Tenggumung Wetan	Jl. Tenggumung Wetan	39	38 BD		154.742223068	4366.256	733.396	4320.666	881.236	436
	Jl. Wadung Asri	Jl. Wadung Asri	40	41 S		154.923109074	4629.018	4131.649	4589.064	4286.252	461

Gambar V-1 Tabel Hasil Konversi Data

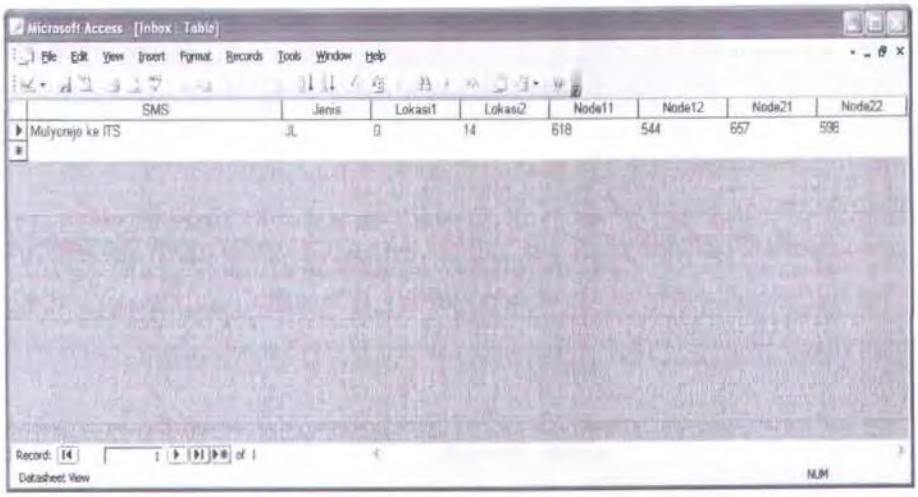


Gambar V-2 Hasil Konversi

5.2.2 Uji Coba Pencarian Rute

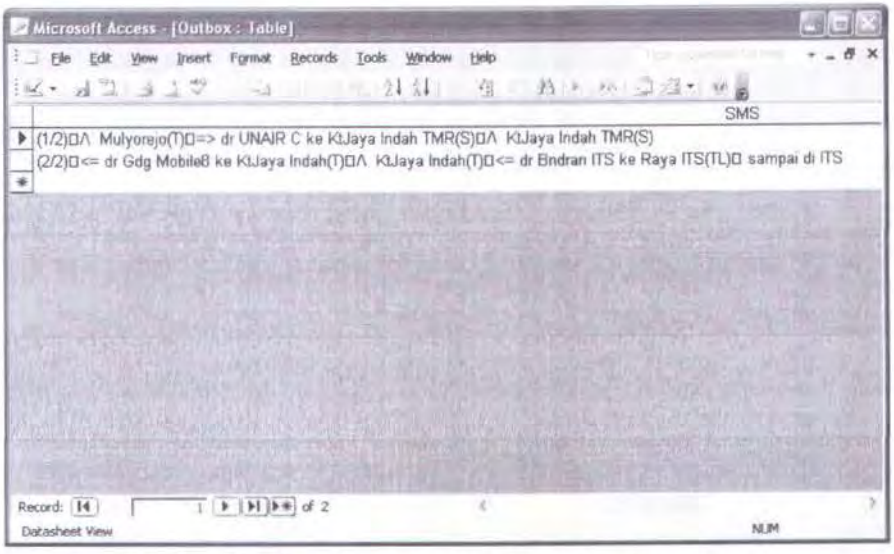
5.2.2.1 Uji Coba Skenario I

Uji coba pada skenario I yaitu uji coba dengan mencari rute dengan kendaraan pribadi pada waktu tertentu, adapun hasil SMS yang dikirimkan oleh pengguna dan diterima oleh aplikasi terlihat pada gambar di bawah ini :



Gambar V-3 SMS Pencarian Rute untuk Kendaraan Pribadi yang Diterima oleh Aplikasi

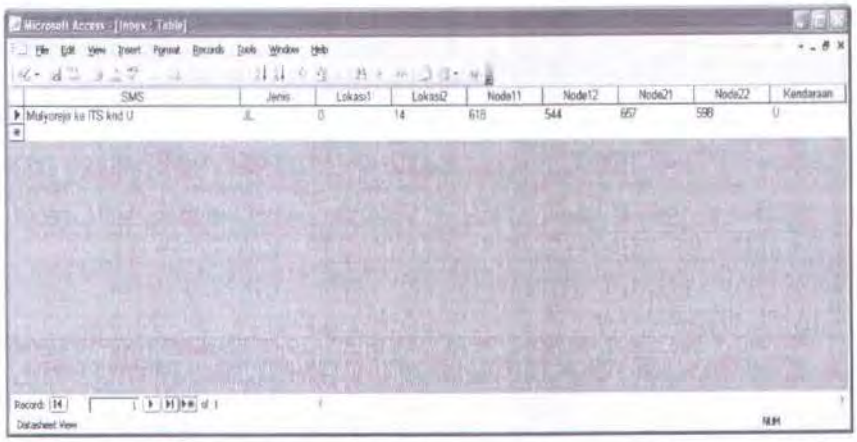
Pada gambar di atas, pengguna mencari rute dari mulyorejo menuju ke ITS. Output dari pencarian rute tersebut terdapat dua SMS, hasil dari pengiriman SMS di atas akan diproses oleh aplikasi, hasil pencarian SMS tersebut akan dikirimkan kembali ke pengguna yang meminta, adapun hasil SMS yang akan dikirimkan adalah sebagai berikut :



Gambar V-4 Hasil Pencarian Rute Untuk Kendaran Pribadi

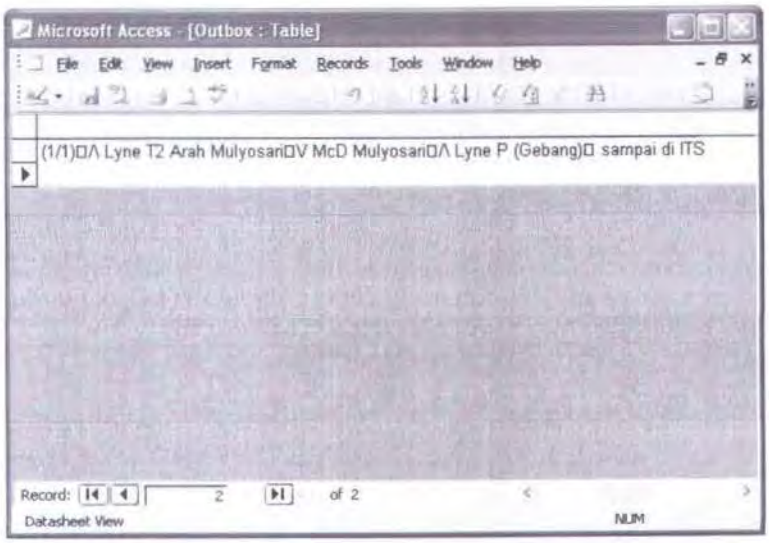
5.2.2.2 Uji Coba Skenario II

Uji coba pada skenario II yaitu uji coba untuk mencari rute dengan menggunakan angkutan umum. Adapun hasil pengiriman SMS yang diterima aplikasi untuk pencarian rutenya yaitu :



Gambar V-5 SMS Pencarian Rute untuk Angkutan Umum yang Diterima oleh Aplikasi

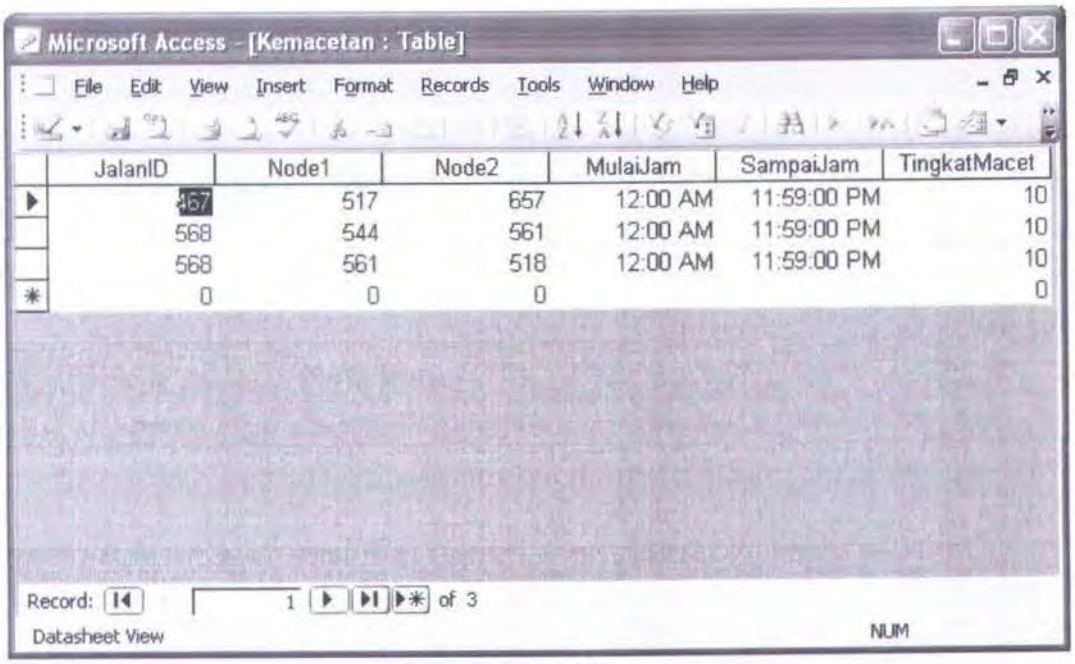
Hasil dari pengiriman SMS di atas, pada kolom kendaraan bernilai U, hal ini menunjukkan bahwa pencarian rute tersebut untuk angkutan umum, hasil pencarian SMS tersebut akan dikirimkan kembali ke pengguna, adapun hasil pencarian rutenya adalah sebagai berikut :



Gambar V-6 SMS Pencarian Rute untuk Angkutan Umum yang Diterima oleh Aplikasi

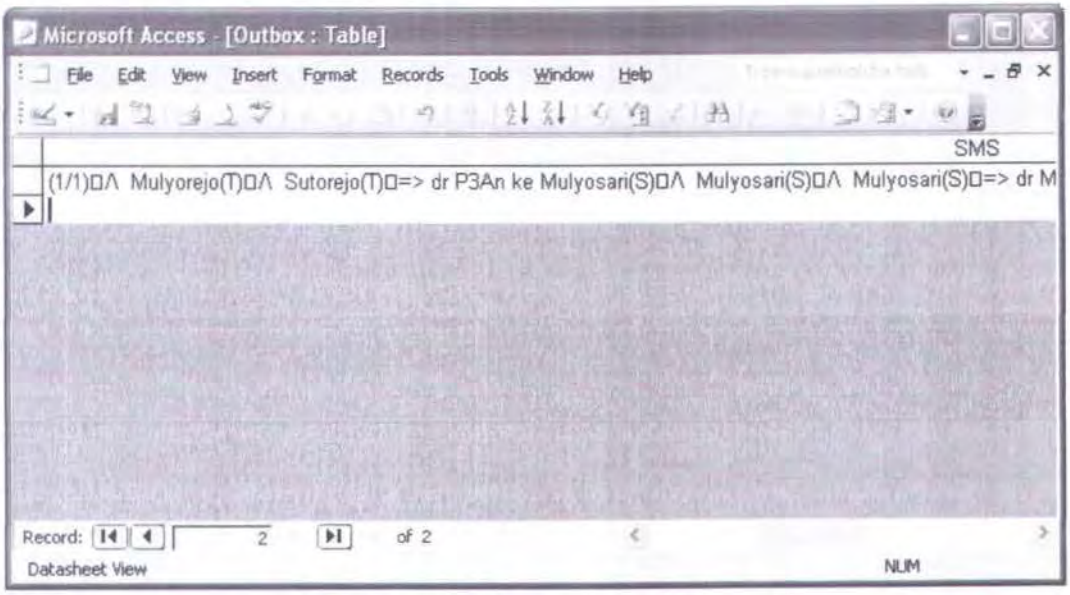
5.2.2.3 Uji Coba Skenario III

Hasil uji coba skenario III adalah membandingkan apakah hasil rute tersebut adalah rute tercepat, dengan mengisi tabel kemacetan pada waktu tertentu. Isi dari tabel kemacetan tersebut adalah sebagai berikut



Gambar V-7 Tabel Kemacetan

Pada pencarian untuk skenario I, dengan mengisi tabel kemacetan tertentu memberikan hasil yang berbeda dengan skenario I pada sumber dan tujuan pencarian yang sama, hasil pencarian rutenya adalah sebagai berikut :



Gambar V-8 Hasil Pencarian Rute Skenario III

5.2.3 Uji Coba Pengiriman dan Penerimaan SMS

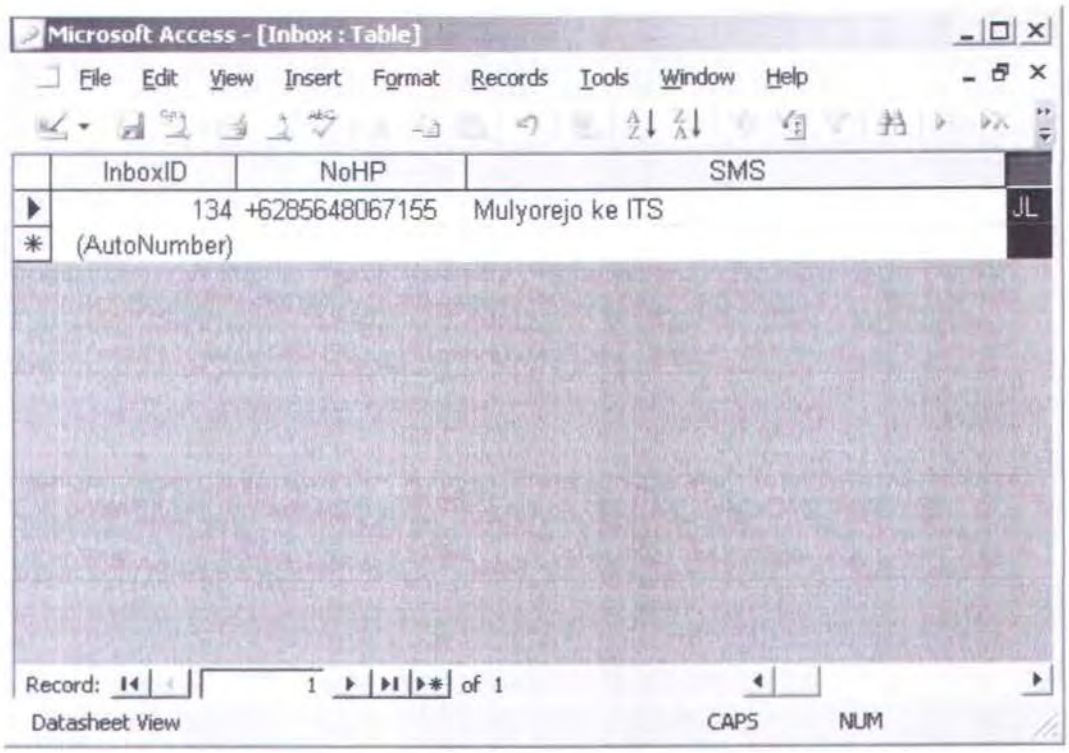
Setelah melakukan pencarian rute, hasilnya akan dikirimkan oleh aplikasi, berikut ini hasil pengiriman SMS dilakukan oleh aplikasi:

Gambar V-9 Hasil Pengiriman oleh Aplikasi

Gambar V-10 Lanjutan 1 Hasil Pengiriman oleh Aplikasi

Gambar V-11 Lanjutan 2 Hasil Pengiriman oleh Aplikasi

Adapun hasil pengiriman pengguna akan dikirimkan ke dalam tabel inbox, isi dari SMS yang dikirimkan oleh pengguna dan diterima oleh aplikasi adalah sebagai berikut :

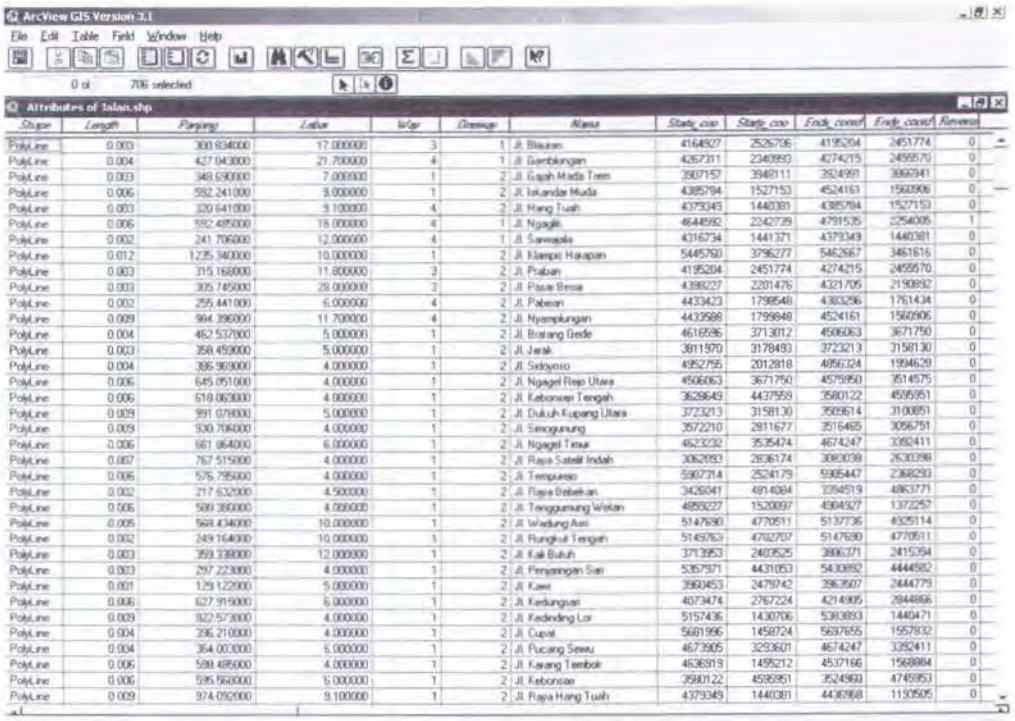


Gambar V-12 SMS yang Diterima Aplikasi

5.3 PELAKSANAAN EVALUASI

5.3.1 Evaluasi Konversi Data

Pada proses konversi data, data sebelumnya adalah data atribut untuk SHP, gambar di bawah ini adalah data atribut dari SHP :



ArcView GIS Version 3.1

File Edit Table Field Window Help

0 of 706 selected

Attributes of Jalan.shp

Shape	Length	Panjang	Lebar	Wgr	Damage	Nama	Start_x_coo	Start_y_coo	End_x_coord	End_y_coord	Reverse
PolyLine	0.003	300.934000	17.000000	3	1	Jl. Blauran	4164927	2526706	4195204	2451774	0
PolyLine	0.004	427.943000	21.700000	4	1	Jl. Genblongan	4267311	2340950	4274215	2459570	0
PolyLine	0.003	348.690000	7.000000	1	2	Jl. Gajah Mada Tmn	3907157	3948111	3924951	3869941	0
PolyLine	0.006	552.241000	9.000000	1	2	Jl. Iskandar Muda	4385784	1527153	4524161	1560906	0
PolyLine	0.003	320.641000	9.100000	4	2	Jl. Hang Tuah	4379345	1440381	4385784	1527153	0
PolyLine	0.006	592.485000	18.000000	4	1	Jl. Ngagel	4644582	2042739	4791535	2254005	1
PolyLine	0.002	241.706000	12.000000	4	1	Jl. Sanwajala	4316734	1441371	4379349	1440381	0
PolyLine	0.012	1235.340000	10.000000	1	2	Jl. Kampc Harapan	5445760	3796277	5462667	3461616	0
PolyLine	0.003	315.168000	11.800000	3	2	Jl. Prabon	4195204	2451774	4274215	2459570	0
PolyLine	0.003	305.745000	29.000000	3	2	Jl. Pasar Besar	4398227	2201476	4321705	2190892	0
PolyLine	0.002	255.441000	6.000000	4	2	Jl. Pabean	4433423	1798548	4383256	1761434	0
PolyLine	0.009	984.396000	11.700000	4	2	Jl. Nyamplungan	4433588	1799848	4524161	1560906	0
PolyLine	0.004	462.537000	5.000000	1	2	Jl. Bratang Gede	4616586	3713012	4586063	3671750	0
PolyLine	0.003	358.459000	5.000000	1	2	Jl. Jarak	3811970	3178493	3723213	3158130	0
PolyLine	0.004	386.969000	4.000000	1	2	Jl. Sidoyoso	4952755	2012818	4856324	1934629	0
PolyLine	0.006	645.051000	4.000000	1	2	Jl. Ngagel Rejo Utara	4586063	3671750	4575950	3514575	0
PolyLine	0.006	618.063000	4.000000	1	2	Jl. Kebonsari Tengah	3629649	4437559	3580122	4595951	0
PolyLine	0.009	991.078000	5.000000	1	2	Jl. Dukuh Kupang Utara	3723213	3158130	3569614	3100851	0
PolyLine	0.009	930.706000	4.000000	1	2	Jl. Simogunung	3572210	2811677	3516465	3056751	0
PolyLine	0.006	661.864000	6.000000	1	2	Jl. Ngagel Timur	4623232	3535474	4674247	3352411	0
PolyLine	0.007	767.515000	4.000000	1	2	Jl. Raya Satelit Indah	3062093	2836174	3080038	2630098	0
PolyLine	0.006	575.795000	4.000000	1	2	Jl. Tempureso	5907314	2524179	5905447	2368293	0
PolyLine	0.002	217.632000	4.500000	1	2	Jl. Raya Bebekan	3426041	4814084	3394519	4863771	0
PolyLine	0.005	500.380000	4.000000	1	2	Jl. Tenggumung Wetan	4859227	1520097	4804527	1372257	0
PolyLine	0.005	568.434000	10.000000	1	2	Jl. Wadung Asri	5147680	4770511	5137736	4925114	0
PolyLine	0.002	249.164000	10.000000	1	2	Jl. Rungkut Tengah	5149763	4702707	5147680	4770511	0
PolyLine	0.003	359.338000	12.000000	1	2	Jl. Kali Bukuh	3713953	2403525	3886371	2415294	0
PolyLine	0.003	297.223000	4.000000	1	2	Jl. Penyerangan Sari	5357971	4431053	5430852	4444582	0
PolyLine	0.001	129.122000	5.000000	1	2	Jl. Kawi	3969453	2479742	3963507	2444779	0
PolyLine	0.006	627.919000	6.000000	1	2	Jl. Kedungsari	4073474	2767224	4214905	2844866	0
PolyLine	0.009	922.573000	4.000000	1	2	Jl. Kedinding Lor	5157436	1430706	5383853	1440471	0
PolyLine	0.004	396.210000	4.000000	1	2	Jl. Cupat	5681996	1458724	5697655	1557832	0
PolyLine	0.004	364.003000	6.000000	1	2	Jl. Pucang Sewu	4673905	3293601	4674247	3352411	0
PolyLine	0.006	598.485000	4.000000	1	2	Jl. Kayang Tembok	4636919	1455212	4537166	1568884	0
PolyLine	0.006	595.560000	6.000000	1	2	Jl. Kebonsari	3580122	4595951	3524960	4745953	0
PolyLine	0.009	974.092000	9.100000	1	2	Jl. Raya Hang Tuah	4379345	1440381	4436968	1193505	0

Gambar V-13 Data Atribut pada SHP

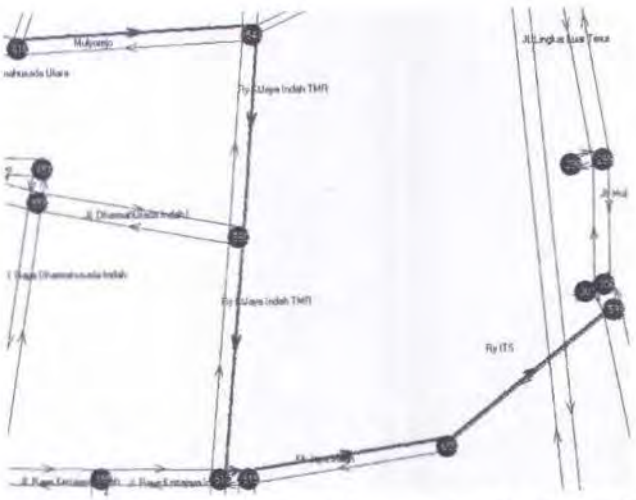
Gambar di bawah ini adalah gambar peta digital dari SHP dengan mengoperasikan *software* ArcView 3.1, gambar tersebut di*capture* pada posisi sesuai dengan gambar pada hasil evaluasi.

Gambar V-14 Peta Digital Dari ArcView

Hasil peta dari *ArcView* peta digambarkan dengan garis-garis sesuai dengan posisi jalan, pada uji coba gambar hasil konversi berupa node-node dan edge-edge, namun data hasil konversi mempunyai *edge* dan *node* untuk menentukan arah dan belokan jalan.

5.3.2 Evaluasi Pencarian Rute Menggunakan Kendaraan Pribadi

Hasil pencarian rute untuk kendaraan pribadi dari Mulyorejo ke ITS ditunjukkan pada garis tebal

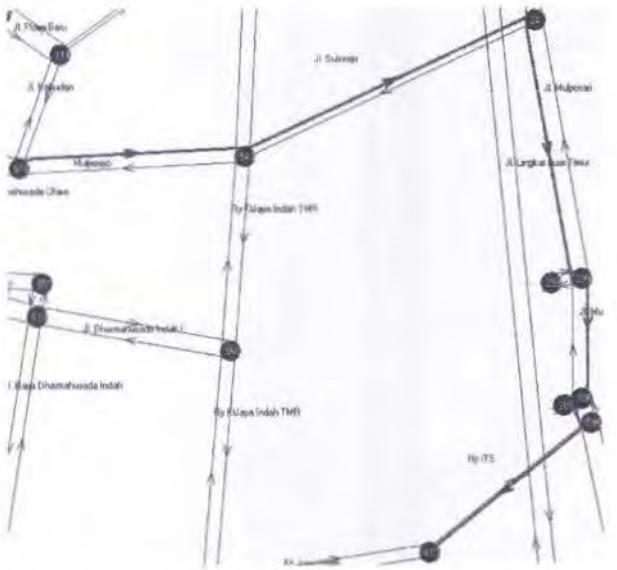


Gambar V-15 Hasil Pencarian Rute Menggunakan Kendaraan Pribadi

Dari gambar di atas, sesuai dengan rute yang dihasilkan oleh SMS pada uji coba, keterangan arah pada SMS yang diterima pengguna sesuai dengan gambar di atas.

5.3.3 Evaluasi Pencarian Rute Menggunakan Angkutan Umum

Hasil pencarian rute untuk angkutan umum dari Mulyorejo ke ITS ditunjukkan pada garis tebal



Gambar V-16 Hasil Pencarian Rute Menggunakan Angkutan Umum

Dari gambar di atas, sesuai dengan rute yang dihasilkan oleh SMS pada uji coba, bahwa rute dari Mulyorejo ke ITS menggunakan angkutan umum ternyata

mempunyai hasil yang berbeda, karena untuk sampai ke ITS angkutan yang ada berada pada jalan yang bergaris tebal.

5.3.4 Evaluasi Rute Tercepat

Hasil pencarian rute sebelum tabel kemacetan diisi data dan setelah diisi menunjukkan hasil yang berbeda, hal ini menunjukkan bahwa kemacetan mempengaruhi pencarian rute.

5.3.5 Evaluasi Pengiriman dan Penerimaan SMS

Dari hasil uji coba pembacaan data SMS dan penyimpanan data SMS ke dalam database diperoleh data-data sebagai berikut:

1. Pembacaan isi SMS dari *handphone* dapat berjalan dengan baik. Hal ini dapat dilihat pada bahwa data dapat ditampung pada *buffer* sebelum data dimasukkan ke dalam *database*.
2. Penyimpanan data SMS ke dalam database dapat berjalan dengan baik . Hal ini dapat dilihat dengan masuknya data di dalam *buffer* ke dalam tabel *inbox* serta dapat ditampilkan pada *handphone*.

BAB VI
PENUTUP

Bab ini berisi kesimpulan yang diperoleh berdasarkan uji coba yang telah dilakukan. Selanjutnya diberikan beberapa saran yang mungkin dapat digunakan untuk mengembangkan hasil yang diperoleh pada tugas akhir ini.

6.1 KESIMPULAN

Setelah dilakukan serangkaian uji coba dan analisa terhadap aplikasi, maka dapat diambil kesimpulan sebagai berikut:

1. Format data *graph* hasil konversi beserta informasi-informasi yang disediakan dapat menunjukkan arah serta pedoman-pedoman yang dapat membuat informasi rute lebih jelas dan informatif daripada rute yang hanya menunjukkan nama jalan saja.
2. Algoritma Dijkstra dapat diimplementasikan untuk pencarian rute, dalam pencarian rute tersebut mempunyai banyak bobot di antaranya bobot jarak, kemacetan, waktu dan angkutan umum.
3. Hasil dari pencarian rute baik dengan menggunakan kendaraan pribadi ataupun angkutan umum dengan memanfaatkan SMS dengan menunjukkan arah-arah dan pedoman sesuai dengan hasil pencarian rute yang ada pada data.

4. SMS yang dikirimkan oleh pengguna dapat diterima sesuai dengan yang direncanakan seperti terlihat pada gambar V-8 sampai V-11 dan hasil pencarian rutenya dikirimkan kembali ke pengirim.

6.2 SARAN

Aplikasi Pencarian rute menggunakan SMS ini dirancang dengan hasil pengiriman format SMS yang singkat dan jelas, sehingga seorang pengirim dapat memahami maksud dari hasil rute yang dicari, namun aplikasi ini masih dapat dikembangkan, adapun pengembangan aplikasi yang disarankan adalah sebagai berikut :

1. Pengkonversian data yang diambil adalah data koordinat awal dan akhir, jika terdapat jalan yang berbelok setengah lingkaran dan data atribut dari jalan tersebut hanya satu record saja, maka jalan tersebut tidak dianggap belok setengah lingkaran tetapi ditarik garis lurus antara koordinat awal dan koordinat akhir, jika dengan pengkonversian data yang lebih baik maka akan menghasilkan pencarian yang lebih baik pula.
2. Manajemen data lokasi, kemacetan, rute angkutan dan pedoman-pedoman dilakukan setelah melakukan konversi, jika pengkonversian data lagi maka data-data tersebut di atas akan tidak valid, oleh karena itu untuk data-data di atas, sebaiknya data tersebut sudah berbentuk data geografis, sehingga jika melakukan konversi data, maka data tersebut di atas juga dilakukan konversi.
3. Untuk pencarian rute tercepat parameter yang digunakan adalah kemacetan yang terjadi, padahal parameter-parameter yang menentukan

suatu rute tercepat tidak hanya kemacetan, misalnya jalan rusak, ada galian pipa PDAM, banjir dan lain-lain, dalam aplikasi ini parameter-parameter di atas tidak dimasukkan.

DAFTAR PUSTAKA

[1] Basori, Ahmad Hoirul. 2004. "Perancangan dan Pembuatan Perangkat Lunak untuk Ektraksi Isi Dokumen Multiformat dengan Menggunakan SMS dan Faximile". Surabaya. Teknik Informatika Institut Teknologi Sepuluh Nopember.

[2] Brown, David William. *An Introduction to Object-Oriented Analysis (Objects and UML in Plain English)*. 2rd Edition. New York. John Willey & Sons, Inc.

[3] Cormen, Thomas H, dkk. 1990. *Introduction to Algorithms*. 2rd Edition. Massachusetts. Massachusetts Institute of Technology Press.

[4] ESRI. 1997. "ESRI Shapefile: A Technical Description". California. Environmental Systems Research Institute, Inc.

[5] Langsam, Yedidyah. 1996. *Data Structures Using C and C++*. 2rd Edition. New Jersey. Prentice-Hall, Inc.

[6] Wahyudi, Edi. 2003. "Penentuan Lin atau Rute Angkutan Umum di Surabaya dengan Pendekatan Sistem Informasi Geografis". Surabaya. Teknik Informatika Institut Teknologi Sepuluh Nopember.