IMPLEMENTASI ALGORITMA ADJACENCY LIST MODEL PADA HIERARCHICAL DATA STRUCTURE UNTUK DINAMISASI LAYANAN STIMATA M-ACADEMIK

Mahmud Yunus
Dosen STIMIK PPKIA Pradnya Paramita Malang

Abstraksi

STIMATA *Mobile-Academic* (*M-Academic*) merupakan suatu sistem layanan informasi akademik dan kemahasiswaan berbasis SMS (*Short Message Service*), bagi segenap civitas akademika STMIK Pradnya Paramita Malang (STIMATA). Dikembangkannya sistem informasi berbasis SMS ini, dimaksudkan untuk dapat memberikan layanan informasi akademik secara lebih *mobile*, praktis dan efisien.

Namun dalam implemetasinya, layanan SMS STIMATA *M-Academic* ini seringkali mengalami perombakan pada program aplikasi-nya. Hal ini seringkali terjadi jika ada penambahan atau pengurangan layanan informasi akademik yang disediakan. Sistem layanan yang demikian dirasa kurang praktis dan efisien. Selain itu sistem layanan ini juga belum mampu memberikan respon yang informatif, khususnya untuk struktur bahasa (*syntax*) SMS yang salah dan tidak sah. Oleh karena itu perlu dipikirkan cara-cara baru untuk meperbaiki kinerja dari sistem STIMATA *M-Academic*.

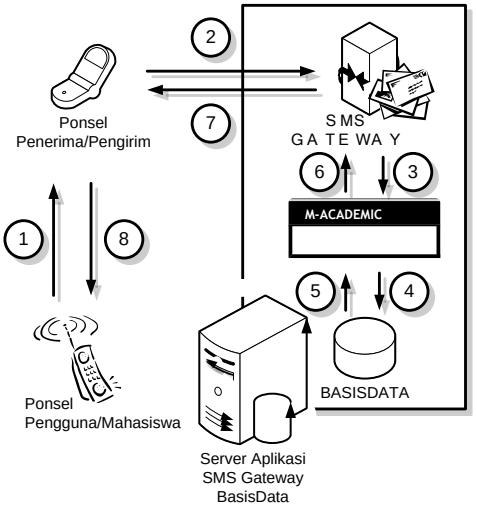
Hal yang dapat dilakukan adalah (1) mengimplementasikan suatu sistem *Respons by Request*, dimana sistem STIMATA *M-Academic* yang dikembangkan harus mampu memberi jawaban dengan baik, untuk semua permintaan layanan, baik SMS yang sah dan benar, maupun SMS yang *error* secara *syntax* yang telah ditentukan; dan (2) *syntax* layanan pada aplikasi *M-Academic* yang dikembangkan harus bersifat dinamis, dimana setiap kali ada penambahan atau pengurangan jenis layanan, program aplikasi tidak perlu dirombak dan dikompilasi ulang. Pengimplementasian algoritma *Adjacency List Model* pada *Hierarchical Data Structure*, dapat dijadikan jembatan dalam mendinamiskan layanan pada sistem *M-Academic* di STIMATA.

Kata-kata kunci : SMS, Adjacency List Model, Hierarchical Data Structure, Syntax

1. PENDAHULUAN

STIMATA *Mobile-Academic* (*M-Academic*) merupakan suatu sistem layanan informasi akademik dan kemahasiswaan bagi segenap civitas akademika STMIK Pradnya Paramita Malang (STIMATA), yang berbasis SMS (S*hort Message Service*). Dikembangkannya sistem informasi berbasis SMS ini, dimaksudkan untuk dapat memberikan layanan informasi akademik secara lebih *mobile*, praktis dan efisien. Harapan tersebut kiranya tidak berlebihan jika mengingat perangkat (*gadget*) telekomunikasi *mobile* (*handphone*), saat ini telah banyak beredar dan digunakan oleh masyarakat luas. Melalui layanan ini, mahasiswa dan pengguna lainnya dapat mengetahui informasi akademik dan kemahasiswaan seperti indeks prestasi, sisa angsuran biaya kuliah, dan jadual matakuliah, dengan hanya mengirim SMS ke nomer yang telah ditentukan dan dengan biaya normal (murah).

Secara ringkas, diagram blok proses penerimaan dan pengiriman SMS dari dan ke pengguna oleh sistem STIMATA *M-Academic* dapat dilihat pada gambar 1.



Gambar 1: Diagram Blok Sistem STIMATA *M-Academic*

Dari gambar 1 tersebut dapat dijelaskan proses yang terjadi pada sistem STIMATA *M-Academic*, sebagai berikut ;

1. Pengguna layanan mengirim SMS ke sebuah ponsel yang berfungsi sebagai terminal penerima dan pengirim
2. Secara berkala *SMS Gateway* memeriksa pesan SMS yang terdapat pada kotak *inbox* ponsel terminal tersebut
3. Sistem *M-Academic* mengambil pesan dan menyimpan data nomor ponsel pengirim layanan dengan bantuan *SMS Gateway*.
4. Sistem *M-Academic* menerjemahkan permintaan layanan dan melakukan *query* ke server basisdata (*database server*)
5. Melalui sebuah algoritma, pesan permintaan layanan dianalisa keabsahannya. Jika permintaan sah maka server basisdata mengirim jawaban ke sistem *M-Academic*.
6. Sistem *M-Academic* mentransfer jawaban *query* ke *SMS Gateway*
7. *SMS Gateway* mentransfer jawaban ke ponsel terminal.
8. Ponsel terminal mengirim jawaban ke pengguna layanan.

Untuk dapat mengakses informasi melalui layanan *M-Academic* tersebut, mahasiswa/pengguna harus menuliskan kalimat permintaan tertentu, kemudian mengirimkannya ke nomor *M-Academic* yang telah ditentukan. Misalnya untuk mengetahui indeks prestasi kumulatif, bentuk umum kalimat layanannya adalah sebagai berikut:

ID <spasi> PIN <spasi> IPK

ID merupakan kode identitas mahasiswa yang dapat juga berupa **NIM** (Nomor Induk Mahasiswa)-nya, sedangkan **PIN** merupakan kata kunci (*password*) untuk validasi, serta **IPK** adalah kode layanan untuk mengakses informasi indeks prestasi kumulatif mahasisswa yang bersangkutan. Contohnya, jika pengguna mengetikkan kalimat

07.52.0019 RAHASIA IPK

maka artinya pengguna meminta layanan indeks prestasi kumulatif (IPK) untuk mahasiswa dengan kode identitas/NIM **07.52.0019** dengan PIN **RAHASIA**.

Namun dalam penerapannya, pada layanan SMS-Akademik ini ditemukan banyak permasalahan, diantaranya adalah; (1) seringnya pengguna salah menuliskan kalimat permintaan, namun tidak mengetahui dimana letak kesalahan penulisannya; (2) bila ada tambahan jenis layanan baru, maka listing program aplikasinya harus dirubah dan dikompilasi ulang, hal ini dirasa kurang

praktis dan efisien, (3) adanya kemungkinan pengguna yang sekedar iseng mengirim SMS, yang mungkin dalam jumlah banyak dan intensif dapat mengacaukan sistem; dan (4) kemungkinan pengaksesan informasi oleh para *hacker* dengan maksud mengacaukan dan atau merusak sistem.

Untuk mengatasi beberapa kekurangan tersebut, perlu dipikirkan cara-cara baru untuk meperbaiki kinerja dari sistem STIMATA *M-Academic*. Beberapa hal yang dapat dilakukan adalah; (1) mengimplementasikan suatu sistem *Respons by Request*, dimana sistem STIMATA *M-Academic* yang dikembangkan harus mampu memberi jawaban dengan baik, untuk semua permintaan layanan, baik SMS yang sah dan benar, maupun SMS yang salah (*error*) secara struktur bahasa (*syntax*) yang telah ditentukan; dan (2) *syntax* (struktur bahasa) layanan pada aplikasi *M-Academic* yang dikembangkan harus bersifat dinamis, dimana setiap kali ada penambahan atau pengurangan jenis layanan, program aplikasi tidak perlu dirombak dan dikompilasi ulang.

Dengan pengimplementasian algoritma *Adjacency List Model* pada *Hierarchical Data Structure* dapat dijadikan jembatan dalam mendinamiskan layanan pada sistem *M-Academic* di STIMATA.

2. RUMUSAN MASALAH

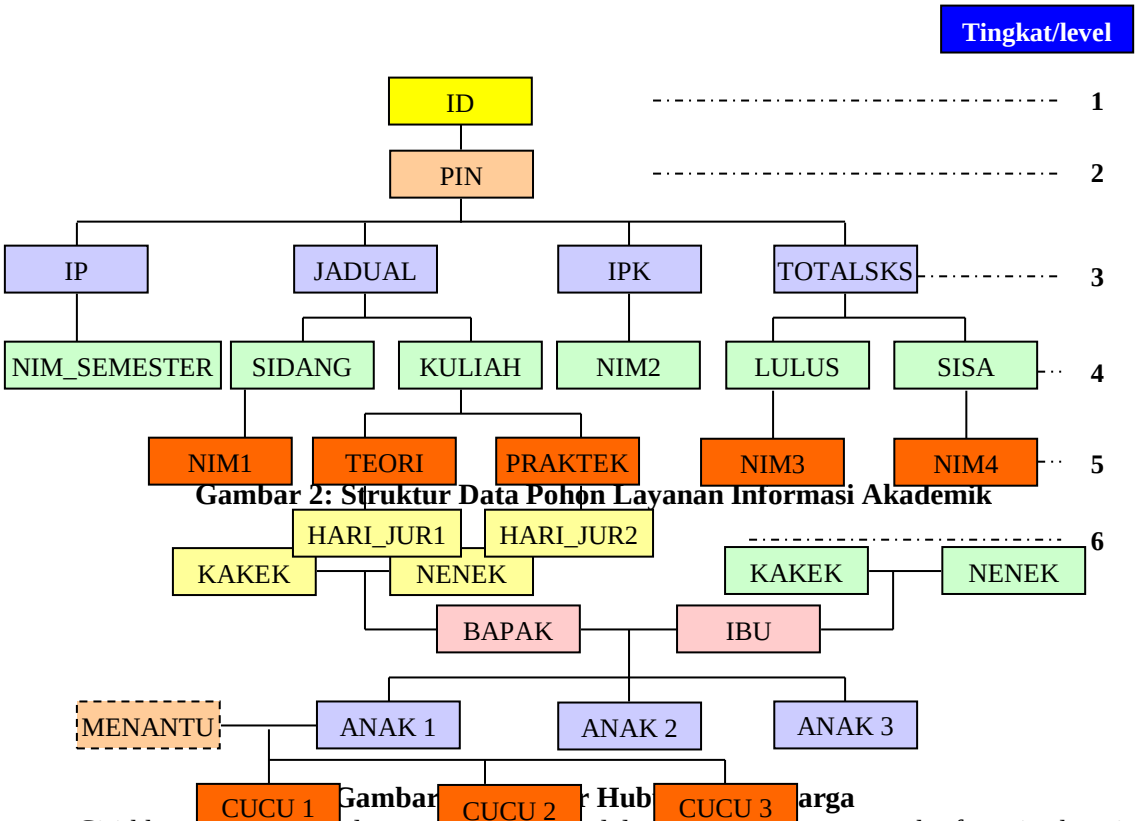
Bagaimana mengimplemtasikan algoritma *Adjacency List Model* pada *Hierarchical Data Structure*, untuk merancang algoritma pemrograman yang secara dinamis, dapat penambahan atau mengurangi layanan pada sistem STIMATA *M-Academic*, tanpa harus mendesain dan mengkompilasi ulang software aplikasinya ?

3. LANDASAN TEORI

3.1 Struktur Data Hirarki (*Hierarchichal Data Structure*)

Struktur data hierarki (*hierarchichical data structure*) dalam bidang ilmu struktur data, dikenal juga sebagai struktur data pohon (*trees data structure*). Struktur data pohon, merupakan struktur data yang memiliki hubungan antar elemen-elemennya secara hierarkis. Jika terdapat sekumpulan data yang memiliki hubungan secara hierarki, maka kita dapat menganalogikannya sebagai hubungan antara orang tua (*parent*) dan anak (*child*) dalam hubungan keluarga (*family relationship*). Setiap elemenya dapat memiliki orang tua (*parent*) yang secara struktur berada diatasnya, dan dapat pula memiliki anak (*child*) yang berada di bawahnya.

Perbedaan mendasar dari struktur data pohon dengan struktur hubungan keluarga pada kehidupan nyata, adalah setiap element pada struktur data pohon hanya boleh memiliki satu element *parent*, sedangkan pada struktur hubungan keluarga setiap individu dapat memiliki 2 (dua) *parent* yaitu ayah dan ibu. Namun baik pada struktur data pohon, maupun struktur hubungan keluarga pada umumnya, setiap elemen atau individu dapat memiliki lebih dari 1 *child*. Untuk lebih jelasnya dapat dilihat perbandingan kedua struktur tersebut pada gambar berikut;



Gambar 2: Struktur Data Pohon Layanan Informasi Akademik

Gambar Hub arga

Ciri khas dari struktur data trees "T", harus selalu ada satu elemen yang berfungsi sebagai akar (*roots*), dan akar dapat mempunyai elemen-elemen sebagai cabang. Cabang dapat mempunyai elemen-elemen sebagai cabang lagi, atau tidak mempunyai elemen-elemen lagi. Elemen yang tidak mempunyai elemen pengikut kita sebut daun (*leaf*) (Paulus Bambangwirawan, 2004:103).

Karakteristik dari struktur data pohon **T** adalah; (1) struktur data pohon **T** adalah kosong (*empty*), jika tidak satu elemenpun dalam struktur **T**; (2) hanya terdapat satu elemen/simpul (*node*) dalam struktur **T**, yang tidak memiliki pendahulu (*parent*) dan (3) semua elemen/simpul (*node*) lainnya pada struktur **T**, hanya memiliki satu pendahulu (*parent*). **N** (*node*) disebut *parent* dari *child*s $\mathbf{C_1, C_2, C_3 \ldots C_n}$, jika N berada tepat diatas dari $\mathbf{C_1, C_2, C_3 \ldots C_n}$ dalam struktur data pohon **T**. Sebaliknya $\mathbf{C_1, C_2, C_3 \ldots C_n}$ merupakan *child* dari **N**, jika $\mathbf{C_1, C_2, C_3 \ldots C_n}$ tepat berada dibawah **N** dalam struktur **T**.

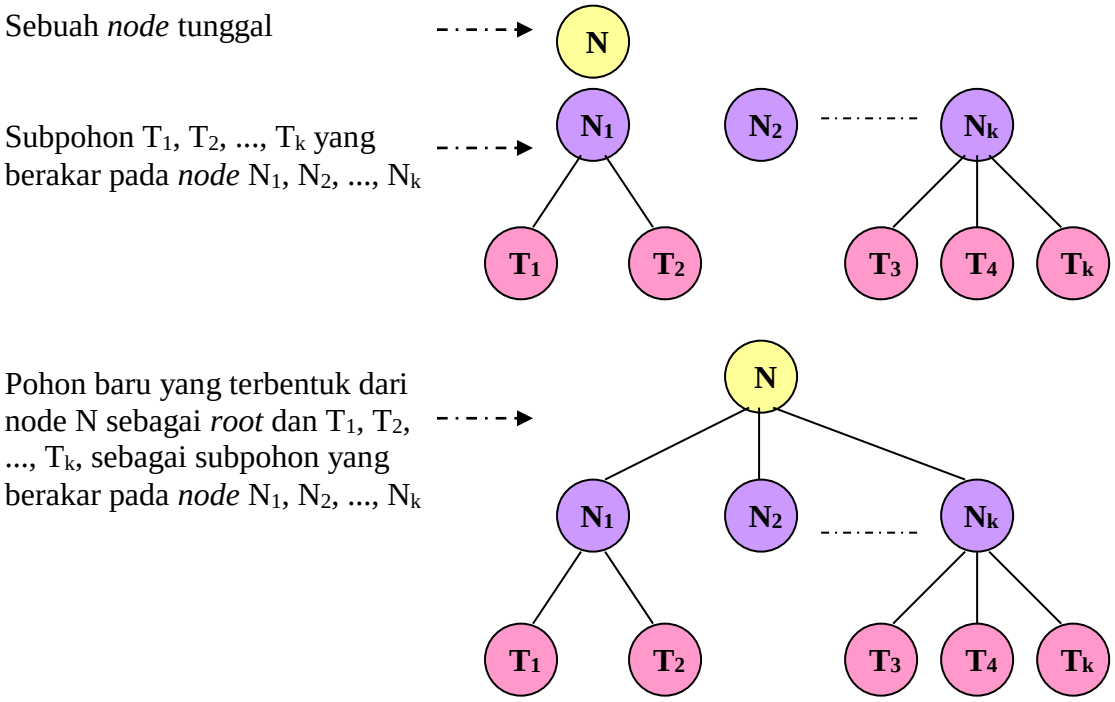
3.2 Istilah Dasar Dalam Struktur Data Hierarki

Secara sederhana, struktur data hierarki atau struktur data pohon, dapat didefinisikan sebagai kumpulan elemen yang salah satu elemennya disebut dengan **akar** (*root*), dan sisa elemen yang lain (yang disebut **simpul**) terpecah menjadi sejumlah himpunan yang saling tidak berhubungan satu sama lain, yang disebut dengan subpohon (*subtree*), atau juga disebut dengan cabang (Insap Santosa, 2001:248). Setiap elemen dalam struktur data hierarkis merupakan sebuah struktur data yang sejenis, baik dia bertindak sebagai *root* maupun sebagai anggota *subtree* dibawahnya. Sehingga elemen dalam sebuah struktur data hierarki/pohon disebut juga sebagai simpul (*node*).

Jika dilihat lagi pada gambar 2, maka terlihat beberapa *subtree* yang memliki akar dan yang tidak. Simpul atau *node* **IP**, **JADUAL**, **KULIAH** dan **TOTALSKS**, merupakan simpul-simpul yang memiliki *subtree*. Setiap *subtree* tersebut tidak memiliki hubungan satu dengan lainnya. Simpul-simpul **IP**, **JADUAL, IPK** dan **TOTALSKS** pada gambar 2 tersebut memiliki *root* yang sama yaitu simpul **PIN**, sedangkan simpul **PIN** sendiri memiliki *root* **ID**, yang merupakan

simpul/*root* tertinggi dalam struktur hierarkis pada gambar 2 tersebut. Berdasarkan contoh tersebut, maka struktur data hierarki secara rekursif dapat didefinisikan sebagai berikut;

- Sebuah simpul tunggal adalah pohon
- Jika terdapat sebuah simpul N dan beberapa subpohon T_1, T_2, ..., T_k yang saling berhubungan, dan berakar pada N_1, N_2, ..., N_k, maka dari simpul N dan subpohon-subpohon ini kita bisa membentuk sebuah pohon yang berakar pada simpul N (Insap Santosa, 2001:249).



Gambar 4: Pembentukan Sebuah Struktur Data Pohon

Secara hierarki, **tingkat** (*level*) suatu simpul (*node*) dapat ditentukan dengan menetapkan terlebih dahulu *root* sebagai *node level* 1. Kemudian secara berturut-turut, dapat ditentukan *level* untuk tiap-tiap *node* yang secara hierarkis berada dibawahnya. Jika suatu simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya dikatakan berada dalam tingkat N + 1 (Insap Santosa, 2001:250).

Istilah **derajat** (*degree*) dari suatu node dalam suatu struktur data pohon, dikenal sebagai banyaknya anak atau keterunan dari *node* tersebut. Untuk simpul-simpul yang tidak memiliki anak atau berderajat 0, disebut juga sebagai simpul daun (*leaf node*). Derajat masing-masing simpul pada contoh gambar 2, dapat kita jabarkan sebagai berikut;

- Simpul **ID**, **IP**, **TEORI** dan **PRAKTEK** berderajat **1**
- Simpul **TOTALSKS** dan **KULIAH** berderajat **2**
- Simpul **PIN** berderajat **4**
- Sedangkan simpul **SEMESTER**, **SIDANG**, **LULUS**, **SISA, HARI_JUR1** dan **HARI_JUR2** merupakan simpul daun (*leaf node*) yang berderajat **0**

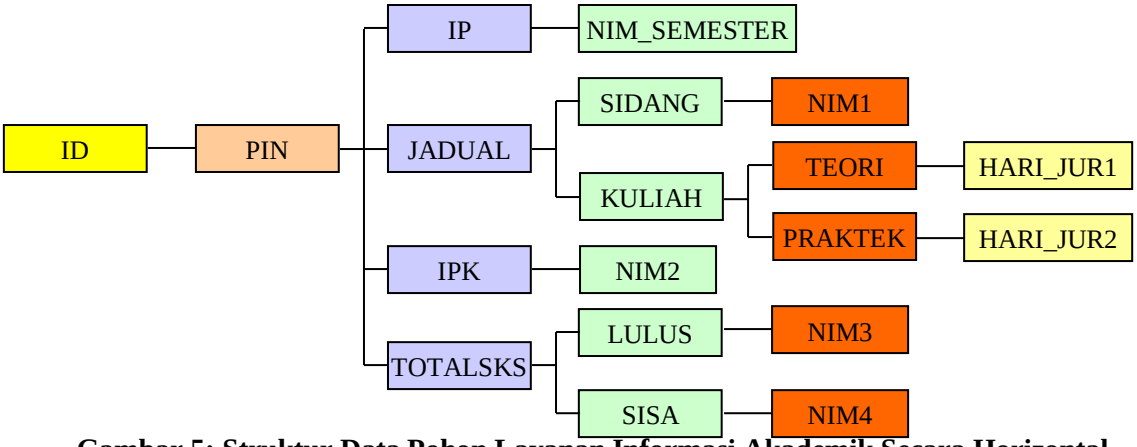
Simpul daun (*leaf node*) sering juga disebut sebagai simpul luar (*external node*), sehingga simpul-simpul lain yang secara hierarki berada diatasnya disebut pula sebagai simpul dalam (*internal node*).

Tinggi (*height*) atau **kedalaman** (*depth*) dari suatu struktur data pohon merupakan merupakan tingkat (*level*) maksimum dari simpul terluar dalam struktur pohon tersebut dikurangi dengan **1**. Sehingga pohon pada gambar 2 yang berakar pada simpul **PIN**, memiliki kedalaman **5**.

Istilah ***ancestor*** pada suatu simpul adalah semua simpul yang terletak dalam satu jalur dengan simpul tersebut dari akar sampai simpul yang ditinjau (Insap Santosa, 2001:251). *Ancestor* atau jalur lintasan untuk menuju simpul **SEMESTER** dari struktur data pohon pada gambar 2, adalah simpul-simpul **ID**, **PIN** dan simpul **IP**.

3.3 Penggambaran Struktur Data Hierarki Secara Horizontal

Suatu struktur data hierarkis dapat juga digambarkan secara horizontal. Hal ini mendorong terciptanya suatu sudut pandang baru, dalam melihat jalur lintasan (*ancestor*) yang harus dilalui guna memperoleh data/informasi yang dimaksud. Dari gambar 5 berikut yang merupakan penggambaran secara horizontal gambar 2, akan diperoleh beberapa *ancestor* untuk mendapat informasi dari simpul terluar.

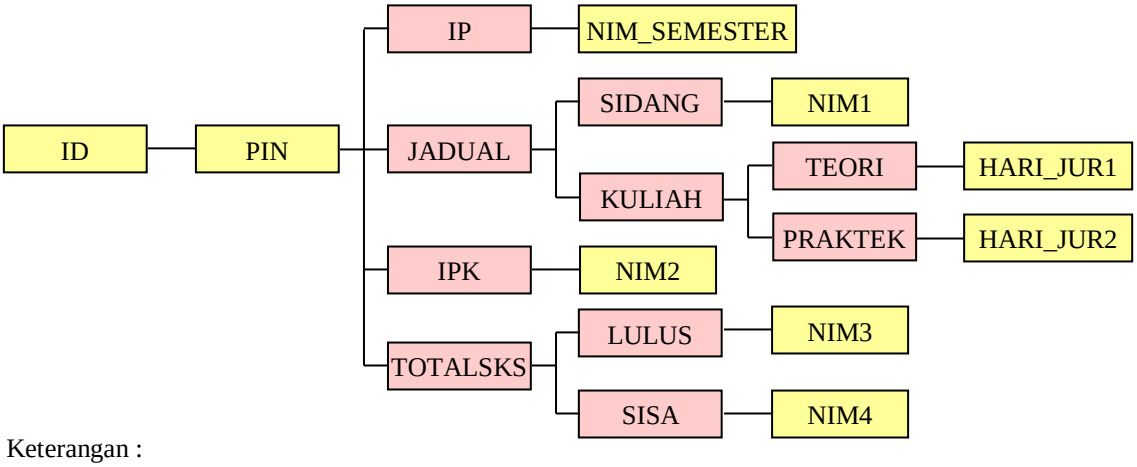


Gambar 5: Struktur Data Pohon Layanan Informasi Akademik Secara Horizontal

Contoh berikut adalah untuk memperoleh informasi yang diinginkan dengan menunjukkan lintasan yang harus terpenuhi;

- Menampilkan informasi **JADUAL** matakuliah **TEORI** pada **HARI** Senin untuk mahasiswa Jurusan TI (Teknik Informatika), akan diperoleh lintasan sebagai berikut;
  ID → PIN → JADUAL → KULIAH → TEORI → HARI_JUR1
- Menampilkan informasi **IP** seorang mahasiswa pada **SEMESTER** tertentu, maka lintasan yang harus terpenuhi adalah sebagi berikut;
  ID → PIN → IP → NIM_SEMESTER

Jika diberlakukan sebuah ketentuan, bahwa simpul ID dan PIN serta simpul terluar dan dalam struktur data hierarkis, adalah sebagai data isian (simpul data), dan simpul-simpul lainnya adalah simpul kata (*word*), maka lintasan yang harus dipenuhi untuk memperoleh informasi yang dimaksud akan berubah. Sebagai contoh dapat dilihat pada gambar berikut beserta uraiaanya;



Keterangan :

Simpul Data

Simpul Kata

Gambar 6: Struktur Data Pohon Layanan Informasi Akademik Hasil Modifikasi

Berdasarkan gambar tersebut, maka untuk memperoleh informasi yang diinginkan dan lintasan yang harus terpenuhi, adalah sebagai berikut;

- Menampilkan informasi **JADUAL** matakuliah **TEORI** pada **HARI Senin** untuk mahasiswa Jurusan **TI** (Teknik Informatika);
  Syntax : **ID → PIN → JADUAL → KULIAH → TEORI → HARI_JUR1**
  Ancestror : **07.52.0555 → 030101 → JADUAL → KULIAH → TEORI → SENIN_TI**
- Menampilkan informasi **IP** seorang mahasiswa pada **SEMESTER** tertentu;
  Syntax : **ID → PIN → IP → NIM_SEMESTER**
  Ancestor : **07.52.0555 → 030101 → IP → 07.52.0555_2**

3.4 Basisdata Relasional (*Rational Database*)

Basisdata (*database*) merupakan kumpulan data (elementer) yang secara logika berkaitan, dalam mempresentasikan fenomena/fakta yang ada secara terstruktur dalam domain tertentu, guna mendukung aplikasi pada sistem tertentu. Basisdata (*database*) dapat juga berarti sebagai kumpulan data yang saling berhubungan dalam merefleksikan fakta-fakta yang terdapat pada sebuah organisasi atau perusahaan. Sistem Manajemen Basisdata atau DBMS (*Database Manajemen System*) merupakan sistem yang dapat digunakan untuk merancang/mendefinisikan, menciptakan, mengelola dan mengendalikan pengaksesan basisdata. Pada prakteknya, konsep DBMS ini diterapkan pada sebuah perangkat lunak (*software*) DBMS. Beberapa *software* DBMS yang populer saat ini adalah; Oracle, MySQL, Microsoft SQL Server, PostgreSQL dan Sybase, dan sebagainya.

Kebanyakan sistem basisdata yang digunakan saat ini, berbasis model data relasional, dimana data disimpan dalam bentuk relasi-relasi (tabel-tabel) yang dapat diakses dengan menggunakan SQL (*Structure Query Language*) sederhana. Kenyataan ini menyebabkan seorang pemrogram komputer tidak lagi dipusingkan dengan kopleksitas nyata, tentang bagaimana data disimpan, direlasikan dan diakses/dimanipulasi. Disisi lain, basisdata dapat berupa objek-objek yang variatif (teks, suara, gambar, film dll.) dan berukuran besar, seperti katalog berbasis internet, ensiklopedia digital, video clip dan sebagainya. Namun selama tabel-tabel yang terdapat dalam sebuah *database* dapat direlasikan (dihubungkan) satu dengan yang lainnya menurut tata cara yang berlaku, maka dengan kesederhanaan bahasa SQL, semua kebutuhan informasi yang terbentuk dari data dalam tabel-tabel tersebut sangat mungkin diperoleh.

3.5 Structured Query Language (SQL)

Structure Query Language (SQL) merupakan bahasa terstruktur yang diaplikasikan untuk mengakses tabel basisdata relational. SQL sendiri dibangun dengan fondasi matematis antara lain himpunan, relasi dan fungsi. Bahasa SQL adalah sarana dasar yang dibutuhkan untuk mengakses data di dalam database relasional (Jose Ramalho, 2001:79). Deklarasi atau perintah SQL secara umum dibagi ke dalam dua kategori, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

Data Definition Language (DDL) merupakan bagian dari SQL yang digunakan untuk mendefinikan data dan objek database. Sedangkan *Data Manipulation Language* (DML) adalah bagian dari SQL yang digunakan untuk memulihkan dan memanipulasi data. Perintah-perintah ini bertanggung jawab untuk melakukan query dan perubahan yang dilakukan di dalam tabel (Jose

Ramalho, 2001:81). Berikut ini adalah beberapa contoh SQL yang digunakan untuk memanipulasi dan meng-query data dari beberapa tabel. Misalkan terdapat tabel-tabel basisdata sebagai berikut :

Tabel 1: Tabel Basisdata Mahasiswa

NIM	NAMA_MHS	PIN	IPK
07.52.0555	A. Zainal Abidin	030101	3,26
07.52.0574	Lifatun Choiriyah	25100C	2,73
07.53.0582	Ahmad Munawi	6BK17E	2,90

Keterangan : Digit 4 & 5 dari field NIM, merupakan kode Jurusan

Tabel 2: Tabel Basisdata Jurusan

KDJUR	NAMA_JUR
52	Teknik Informatika
53	Sistem Komputer

Tabel 3: Tabel Basisdata IP Per Semester

NIM	SEMESTER	IP
07.52.0555	1	3,30
07.52.0574	1	2,65
07.53.0582	1	3,01
07.52.0555	2	3,22
07.52.0574	2	2,88
07.53.0582	2	2,80

Beberapa contoh kalimat SQL dalam MySQL berikut, akan menghasilkan data;

- CREATE OR REPLACE VIEW V_GABUNGAN AS
  (SELECT M.NIM, M.NAMA_MHS, M.PIN, M.IPK,
  J.NAMA_JUR, I.SEMESTER, I.IP
  FROM IP AS I, MAHASISWA AS M, JURUSAN AS J
  WHERE (M.NIM = I.NIM) AND (J.KDJUR = SUBSTRING(M.NIM, 4, 2)));

  Akan menghasilkan sebuah meta tabel (*view*) baru bernama V_GABUNGAN, yang berisikan data sebagai berikut;

Tabel 4: Meta Tabel Basisdata V_GABUNGAN

NIM	NAMA_MHS	PIN	IPK	NAMA_JUR	SEMESTER	IP
07.52.0555	A. Zainal Abidin	030101	3,26	Teknik Informatika	1	3,30
07.52.0574	Lifatun Choiriyah	25100C	2,73	Teknik Informatika	1	2,65
07.53.0582	Ahmad Munawi	6BK17E	2,90	Sistem Komputer	1	3,01
07.52.0555	A. Zainal Abidin	030101	3,26	Teknik Informatika	2	3,22
07.52.0574	Lifatun Choiriyah	25100C	2,73	Teknik Informatika	2	2,88
07.53.0582	Ahmad Munawi	6BK17E	2,90	Sistem Komputer	2	2,80

- SELECT NIM, NAMA_MHS, PIN, SEMESTER, IP
  FROM V_GABUNGAN
  WHERE (**NIM** = "**07.52.0555**") AND (**PIN** = "**030101**")
  AND (**SEMESTER** = **2**);

  Akan menghasilkan data berikut;

NIM	NAMA_MHS	PIN	SEMESTER	IP
07.52.0555	A. Zainal Abidin	030101	2	3,22

- SELECT DISTINCT NIM, NAMA_MHS, PIN, IPK
  FROM V_GABUNGAN
  WHERE (**NIM = "07.52.0555"**) AND (**PIN = "030101"**);

data yang dihasilkan;

NIM	NAMA_MHS	PIN	IPK
07.52.0555	A. Zainal Abidin	030101	3,26

4. IMPLEMENTASI HIERARCHICAL DATA STRUCTURE PADA LAYANAN STIMATA M-ACADEMIC

4.1 Penyimpanan dan Pengaksesan Data Hierarkis Dalam Basisdata

Menyimpan data dari sebuah struktur data hierarkis/pohon dalam sebuah database, merupakan masalah umum yang sering ditemui. Inti masalahnya tidak hanya terletak pada bagaimana simpul-simpul (data) pada sebuah struktur data hierarkis itu disimpan. Namun juga harus dipikirkan, bagaimana simpul-simpul tersebut dapat ditampilkan (diakses) dengan cara yang mudah dan lintasan (*ancestor*) yang dilaluinya valid (benar).

Paling tidak telah terdapat dua metode/model yang dapat diaplikasikan dalam memecahkan permasalahan ini, yaitu; (1) *adjacency list (recursion) model* dan (2) *modified preorder tree traversal algorithm*. Metode yang diterapkan pada *adjacency list (recursion) model*, adalah dengan mulai ngunjungi simpul terluar (*external node*) pada awalnya, kemudian berturut-turut secara rekursif, mengunjungi simpul *Parent* dari setiap simpul yang ditemuinya, hingga simpul root (Gijs Van Tulder). Jika struktur data hierarki pada gambar 6 disimpan pada sebuah table dalam database, maka akan menghasilkan daftar simpul seperti pada table 5.

Tabel 5: Tabel Basisdata TREE

NODE	PARENT	LEVEL	TIPE
ID	- ***(NIL)***	1	1
PIN	ID	2	1
IP	PIN	3	0
JADUAL	PIN	3	0
IPK	PIN	3	0
TOTALSKS	PIN	3	0
NIM_SEMESTER	IP	4	1
SIDANG	JADUAL	4	0
KULIAH	JADUAL	4	0
NIM2	IPK	4	0
LULUS	TOTALSKS	4	0
SISA	TOTALSKS	4	0
NIM1	SIDANG	5	0

NODE	PARENT	LEVEL	TIPE
TEORI	KULIAH	5	0
PRAKTEK	KULIAH	5	0
NIM3	LULUS	5	0
NIM4	SISA	5	0
HARI_JUR1	TEORI	6	1
HARI_JUR2	PRAKTEK	6	1

Keterangan Tipe :

- 0 : Node/Simpul Kata
- 1 : Node/Simpul Data

Tabel 6: Struktur Tabel TREE

Nama Field	Tipe Data	Keterangan
NODE	Char(20) pk	Nama simpul & primary key
PARENT	Varchar(20)	Parent simpul (simpul sebelumnya)
LEVEL	TinyInt(1)	Level kedalaman/urutan simpul
TIPE	TinyInt(1)	Tipe simpul (simpul kata/data)

Selanjutnya untuk simpul-simpul ber-TIPE 1 (simpul data) pada table 5, akan dibuatkan tabel *database* tersendiri dengan struktur data seperti pada tabel 7. Table tersebut nantinya digunakan untuk penyimpanan data/informasi yang akan ditampilkan. Setiap *record* yang tersimpan pada tabel-tabel yang terbentuk dari simpul data, akan memiliki *primary key* yang uniq sebagai kunci pencarian. Khusus untuk simpul data ID dan PIN akan disimpan pada tabel MEMBER dengan struktur yang ditunjukkan pada tabel 8.

Tabel 7: Struktur Tabel Simpul Data

Nama Field	Tipe Data	Keterangan
KODE	Char(20) pk	Sebagai kata kunci (*primary key*)
DESKRIPSI	Varchar(45)	Keterangan dari *field* KODE
DATA	Varchar(255)	Data/informasi keluaran

Tabel 8: Struktur Tabel MEMBER

Nama Field	Tipe Data	Keterangan
ID	Char(10) pk	Identifier (*primary key*)
PIN	Varchar(6)	Personal Identifier Number
Keterangan	Varchar(255)	Keterangan tambahan

Berdasarkan tabel-tabel tersebut dapat ditelusuri kebenaran lintasan (*ancestor*) yang dimasukkan, apakah sesuai dengan *syntax* yang berlaku atau tidak. Misalkan untuk menampilkan informasi **JADUAL** matakuliah **TEORI** pada **HARI Senin** untuk mahasiswa Jurusan **TI** (Teknik Informatika); dimana s*yntax* yang berlaku adalah;

ID → PIN → JADUAL → KULIAH → TEORI → HARI_JUR1

ID → PIN → JADUAL → KULIAH → TEORI → HARI_JUR1

LEVEL	**1**	**2**	**3**	**4**	**5**	**6**
TIPE DATA	**1**	**1**	**0**	**0**	**0**	**1**

dan *ancestror* yang dimasukkan adalah;

07.52.0555 → 030101 → JADUAL → KULIAH → TEORI → SENIN_TI

4.2 Penerapan Algoritma Adjacency List Model Pada Basisdata Berstruktur Hirarki

Penerapan algoritma *adjacency list (recursion) model*, adalah dengan mulai ngunjungi simpul terluar (*external node*) pada awalnya, kemudian berturut-turut secara rekursif, mengunjungi simpul *Parent* dari setiap simpul yang ditemuinya, hingga simpul root.

Algoritma ini akan mengecek, apakah urutan data yang dimasukkan telah sesuai dengan lintasan yang dipersyaratkan dalam memperoleh informasi yang diinginkan. Pengecekan dilakukan mulai dari simpul terluar (paling kanan) hingga simpul root (paling kiri) atau proses pengecekan berhenti jika ditemui suatu kesalahan penulisan lintasan (*ancestor*) yang dimasukkan. Paling tidak terdapat tiga jenis kesalahan yang mungkin dapat terjadi dalam penulisan *ancestor*, yaitu (1) kesalahan peletakan urutan simpul sesuai lintasan yang dipersyaratkan; (2) kesalahan penulisan simpul (simpul tidak ada) dan (3) *ancestor* yang dimasukkan benar, namun data/informasinya kosong (belum dimasukkan).

Misalkan untuk menampilkan informasi **JADUAL** matakuliah **TEORI** pada **HARI Senin** untuk mahasiswa Jurusan **TI** (Teknik Informatika); dimana s*yntax* yang berlaku adalah sebagai berikut;

ID → PIN → JADUAL → KULIAH → TEORI → HARI_JUR1

	ID →	**PIN →**	**JADUAL →**	**KULIAH →**	**TEORI →**	**HARI_JUR1**
LEVEL	**1**	**2**	**3**	**4**	**5**	**6**
TIPE DATA	**1**	**1**	**0**	**0**	**0**	**1**

dan *ancestror* yang dimasukkan adalah;

07.52.0555 → 030101 → JADUAL → KULIAH → TEORI → SENIN_TI

Bentuk penerapan algoritma *Adjacency List Model* untuk penelusuran kebenaran lintasan tersebut, adalah sebagai berikut;

a. Definisikan variabel array string SIMPUL untuk menampung data lintasan yang dimasukkan;

```
SIMPUL[1] := ’07.52.0555’;
SIMPUL[2] := ‘030101’;
SIMPUL[3] := ‘JADUAL’;
SIMPUL[4] := ‘KULIAH’;
SIMPUL[5] := ‘TEORI’;
SIMPUL[6] := ‘SENIN_TI’;
```

b. Definisikan nilai awal dari beberapa variabel penanda

```
MAXNODE := 6        // Maksimal simpul yang dimasukkan
Error := False;     // Penanda kesalahan (nilai False → tidak ada kesalahan)
ErrorCode := 0;     // Kode kesalahan (nilai 0 → tidak ada kesalahan)
I := MAXNODE;       // Penelusuran dimulai dari simpul terluar (paling kanan)
```

c. Melakukan perulangan untuk pengunjungan/pengujian setiap simpul, mulai simpul terluar (paling kanan) hingga simpul terdalam (paling kiri/root), atau hingga ada kesalahan.

```
//Selain simpul ID & PIN
While ((I > 2) and Not Error) Do Begin
    // Uji simpul terluar
    If (I = MAXNODE) Then Begin
        Result := ExecSQL('Select * From SIMPUL[I-1] Where KODE Like SIMPUL[I]');
        If Empty(Result) Then Begin
            //Jika Result kosong (KODE tidak ditemukan di tabel simpul SIMPUL[I-1])
            ErrorCode := 2;
            Error := True;
        End
        // Jika Result tidak kosong, maka tampung data/informasi yang diperoleh
        // ke variabel Data_Message, yang selanjutnya akan dikirimkan ke pengguna
        // layanan STIMATA M-Academic
        Else Data_Message := Result.FieldByName('DATA').AsString;
    End
    Else Begin
        //Jika bukan simpul terluar, maka uji urutan simpul (SIMPUL KATA)
        Result := ExecSQL('Select * From TREE Where NODE Like SIMPUL[I]');
        //Uji urutan simpul kata sebelumnya dengan mengecek field PARENT setiap simpul
        //Untuk simpul urutan ke-3, uji apakah PARENT-nya sama dengan simpul 'PIN'
        If (Result.FieldByName('PARENT').AsString Not In [ SIMPUL[I-1], 'PIN' ]) Then Begin
            ErrorCode := 3;
            Error := True;
        End;
        If (Resul.FieldByName('LEVEL').AsInteger <> I) Then Begin
            //Jika level simpul tidak sama dengan urutan DATA yang diinputkan
            ErrorCode := ErrorCode + 4;
            Error := True;
        End;
    End;
    Dec(I);
End;
//Uji apakah simpul ID & PIN (simpul urutan ke-1 & ke-2)
// cocok seperti yang terdapat di table MEMBER
Result := ExecSQL('Select * From MEMBER Where ID Like SIMPUL[1]');
If Empty(Result) Or (Result.FieldByName('PIN').AsString <> SIMPUL[2]) Then Begin
    ErrorCode := 1;
    Error := True;
```

```
End;
```

d. Uji kode kesalahan (jika ada), untuk menentukan jenis kesalahannya

```
If Error Then Begin
    Case Result Of
        1 :  Data_Message := 'ID & PIN tidak cocok/ID belum terdaftar';
        2 :  Data _Message := 'Data/Informasi tidak ditemukan';
        3 :  Data _Message := 'Setelah '+SIMPUL[I]+' bukan '+SIMPUL[I+1];
        4 :  Data _Message := 'Urutan '+SIMPUL[I+1]+' salah';
        7 :  Data _Message := 'Urutan '+SIMPUL[I+1]+' salah, dan Setelah '+SIMPUL[I]+
                              ' bukan '+SIMPUL[I+1];
    End;
End;
```

e. Kirim atau tampilkan informasi yang dihasilkan, sesuai nilai yang terdapat pada variabel Data_Message

```
SendMessage(Data_Message);
```

Hasil akhir dari algoritma tersebut adalah diterima atau tidak *ancestor* yang dimasukkan, yang ditandai oleh ada tidaknya kesalahan. Jika ada kesalahan, maka variabel **Error** bernilai **True**, dan jika sebalikanya, variabel **Error** akan tetap bernilai **False**. Selanjutnya, algoritma tersebut akan mengirimkan data/informasi sesuai nilai yang ada pada variabel **Data_Message**, baik ada kesalahan (**Error** bernilai **True**) maupun tidak ada (**Error** bernilai **False**).

Walaupun algoritma tersebut sudah dapat digunakan untuk mendinamiskan layanan STIMATA M-Academic, namun terdapat satu kelemahan mendasar yang perlu dipikirkan lebih lanjut. Kelemahannya adalah seringnya melakukan permintaan (*request*) data ke server dengan mengirim SQL, untuk mengurai/mengecek keabsahan setiap simpul yang dimasukkan. Sehingga pada saat-saat tertentu karena beban jaringan yang berat, kinerja sistem yang mengimplementasikan algoritma tersebut, akan terasa lambat atau bahkan mengalami kegagalan sistem (sistem macet/hang).

5. KESIMPULAN

Penerapan algoritma *adjacency list (recursion) model* pada basisdata berstruktur hirarki (*hierarchical data structure*), dapat dijadikan suatu algoritma alternativ dalam membangun aplikasi komputer berbasis SMS (*short message service*) yang bersifat dinamis. Program aplikasi STIMATA *M-Academic* telah mengimplementasikan algoritma tersebut dalam memberikan layanan informasi akademik dan kemahasiswaan berbasis SMS (*Short Message Service*), bagi segenap civitas akademika STMIK Pradnya Paramita Malang (STIMATA).

Kedinamisan layanan yang disediakan oleh STIMATA *M-Academic* sangat praktis dan efisien, terutama dalam penambahan atau mengurangi layanan informasi. Hal tersebut dapat dilakukan tanpa harus mendesain dan mengkompilasi ulang software aplikasinya.s

6. DAFTAR PUSTAKA

Gijs Van Tulder. 2003. ***Storing Hierarchical Data in a Database,*** Diakses pada alamat website **http://www.sitepoint.com/hierachical-data-database/**

Insap Santosa. 2001. ***Struktur Data Dengan Delphi 6.0.*** ANDI. Yogyakarta

Jose Ramalho. 2001. ***SQL Server 7***. Diterjemahkan oleh Adi Kurniawan. Elex Media Komputindo. Jakarta

Paulus Bambangwirawan, Dipl. Inform., 2004. ***Struktur Data dengan C***. ANDI. Yogyakarta.