Penggunaan Struktur Data Pohon Berakar dalam XML

Luthfi Chandra Fibrian - 13510047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Luthfi.chandra@s.itb.ac.id

Abstrak

Dalam dunia pemrograman diperlukan struktur data yang sesusai supaya infomasi yang tersimpan dalam struktur data tersebut mudah dibaca oleh komputer ataupun oleh manusia. Salah satu struktur data yang umum dalam dunia pemrograman adalah struktur data pohon. Salah satu aplikasi struktur data pohon adalah XML (Extended Markup Language). XML sering digunakan dalam pemrograman web, dan mobile. Data-data dalam XML disusun dengan menggunakan struktur data pohon supaya memudahkan pengaksesan data oleh komputer dan manusia.

***Kata kunci*—XML, pohon, struktur data**

I. PENDAHULUAN

XML merupakan *meta-language* yang sering digunakan dalam pemrograman web. XML pada umumnya dimanfaatkan untuk menyimpan data. Penyimpanan data dalam XML lebih mudah dibaca. Selain digunakan dalam web, XML juga digunakan dalam pemrograman JAVA.

XML memanfaatkan struktur data pohon berakar dalam penyimpanan datanya. Penyimpanan dengan cara ini dapat mempermudah pembacaan oleh komputer. Selain itu, data atau informasi di XML dapat dibaca oleh manusia karena menggunakan teks yang umum digunakan.

II. APA ITU XML?

XML (Extended Markup Language) adalah bahasa markup yang digunakan untuk menyimpan data dalam bentuk teks. Data-data disimpan dengan format tertentu. Sebenarnya XML bukanlah sebuah bahasa pemrograman. XML merupakan sintaks yang digunakan untuk menjalankan bahasa markup lain sehingga disebut sebagai *meta-language*.

XML berbeda dengan bahasa markup lain seperti HTML. HTML lebih memfokuskan kepada penampilan data, sementara XML bertujuan untuk menyimpan data dalam format tertentu dan siap diproses oleh bahasa pemrograman yang lain.

Aplikasi XML dalam dunia pemrograman bermacam-macam, seperti:

- XML digunakan untuk menyimpan data-data yang akan ditampilkan di web melalui bahasa HTML.
- Dapat digunakan untuk pengiriman data dari DBMS (*Database Management System*) yang berbeda, misalnya dari Oracle ke MySQL. Data-data diekspor ke bentuk XML sebelum dikirimkan ke DBMS yang berbeda.
- XML dapat digunakan sebagai file konfigurasi. Misalnya pada aplikasi JAVA terdapat beberapa file XML yang menyimpan konfigurasi dari aplikasi tersebut.

III. STRUKTUR XML

Bentuk XML hampir sama dengan HTML. XML terdiri dari *tag* yang di dalamnya dapat diisi data. Berbeda dengan HTML, tag dalam XML bebas dinamakan apa saja. Tidak ada ketentuan untuk menamakan *tag* dalam XML. *Tag* dalam XML berfungsi untuk penamaan tipe data tertentu sedangkan pada HTML berfungsi untuk menampilkan data dalam bentuk tertentu sesuai dengan *tag*-nya.

Tag XML dapat dinamakan secara bebas namun ada beberapa karakter yang tidak dapat digunakan untuk penamaan *tag* dan harus digantikan oleh karakter yang lain. Karakter-karakter tersebut tidak dapat digunakan karena akan menyebabkan konflik. Karakter-karakter tersebut antara lain:

Karakter yang dilarang	Pengganti
<	&lt
>	&gt
&	&amp
“ (tanda kutip)	&quot
‘ (apostrop)	&apos

Sebuah dokumen XML terdiri dari bagian bagian yang disebut dengan node. Node-node itu adalah:

1. **Root node** yaitu node yang melingkupi keseluruhan dokumen. Dalam satu dokumen XML hanya ada satu root node. Node-node yang lainnya berada di dalam root node. Pada contoh XML di bawah <biodata> merupakan Root Node.
2. **Element node** yaitu bagian dari dokumen XML yang ditandai dengan tag pembuka dan tag

penutup seperti <nama>Budi</nama>, atau bisa juga sebuah tag tunggal elemen kosong seperti <nama=”Budi”/>.Root node biasa juga disebut root element.

3. **Attribute note** termasuk nama dan nilai atribut ditulis pada tag awal sebuah elemen atau pada tag tunggal.
4. **Text node,** adalah text yang merupakan isi dari sebuah elemen, ditulis diantara tag pembuka dan tag penutup. Misalnya <Kota>Yogyakarta</Kota>. “Yogyakarta” merupakan sebuah text node.

Contoh:

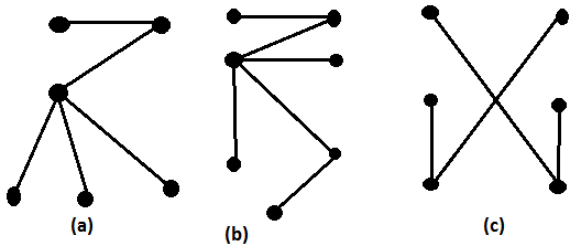
```
<?xml version="1.0"
encoding="UTF-8"?>

<biodata>
        <nama>Budi</nama>
        <TTL>
                <tempat>Solo</tempat>
                <tanggal> 12 </tanggal>
                <bulan>2</bulan>
                <tahun>1992</tahun>
        </TTL>
        <Kota> Yogyakarta </Kota>
        <universitas> UGM </universitas>
        <SMA> SMAN 1 Solo </SMA>
</biodata>
```

5. **Comment node,** adalah baris yang tidak dieksekusi oleh parser. Sintaks yang digunakan adalah <!— ini adalah komentar -->
6. **Processing Instruction node** adalah perintah pengolahan dalam dokumen XML. Node ini ditandai awali dengan karakter <? Dan diakhiri dengan ?>. Tapi perlu diingat bahwa header standard XML <?xml version=”1.0” encoding=”iso-8859-1”?> bukanlah processing instruction node. Header standard bukanlah bagian dari hirarki pohon dokumen XML.contoh processing instruction node: <?xml-stylesheet href="style.css" type="text/css">
7. **NameSpace Node**, digunakan untuk penamaan elemen dan atribut secara unik.

Selain itu, di XML terdapat DTD (Document Type Definition) yang berfungsi untuk mendefinisikan tipe dokumen XML. DTD hampir sama dengan pendeklarasian variabel, fungsi, dan prosedur pada bahasa pemrograman yang lain. Pada XML, struktur dokumen dan daftar-daftar elemen yang digunakan didefinisikan oleh DTD.

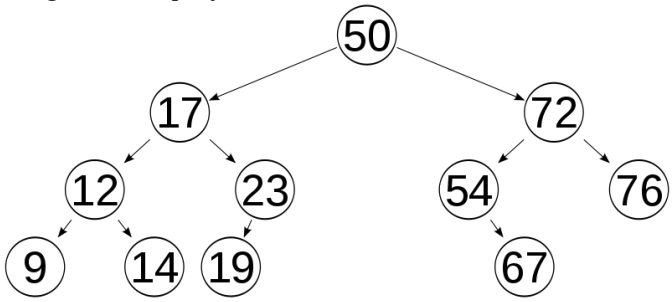
IV. STRUKTUR POHON DALAM XML

Pohon dalam struktur diskrit didefinisikan sebagai graf tak-berarah terhubung yang tidak mengandung sirkuit. Pohon dibedakan menjadi 2 macam, yaitu pohon bebas (*free tree*) dan pohon berakar (*rooted tree*). Pada pohon bebas, tidak ditentukan darimana elemen dalam pohon tersebut diacu.



Gambar 4.1 (a) dan (b) merupakan pohon sedangkan (c) bukan pohon

Jenis pohon yang diaplikasikan pada XML adalah pohon berakar (*rooted tree*). Definisi dari pohon berakar adalah pohon yang sebuah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah. Arah dalam pohon berakar dapat dihilangkan untuk penyederhanaan.

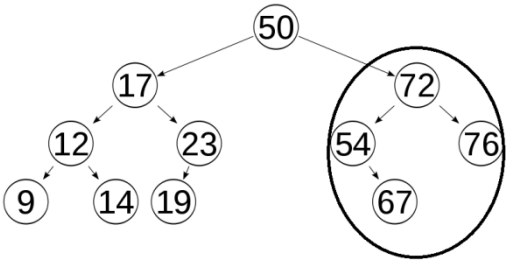


Gambar 4.2 contoh pohon berakar

Bagian-bagian pohon berarah:

- **Akar,** adalah simpul teratas dari pohon berakar. Pada contoh di atas 50 adalah akar.
- **Anak,** adalah simpul yang berada di bawah node lainnya. Pada contoh di atas, 17 dan 72 adalah anak dari 50; 12 dan 13 adalah anak dari 17; 54 dan 766 adalah anak dari 12. Simpul yang berada di atas anak disebut **orangtua** (*parent*).
- **Lintasan (*path*),** adalah runtutan simpul-simpul yang dilalui untuk menghubungkan simpul asal dengan simpul tujuan. Pada contoh di atas, lintasan dari 50 ke 9 adalah 50, 17, 12, 9. panjang lintasan adalah jumlah sisi yang dilalui dalam suatu lintasan, yaitu k – 1. Pada contoh, panjang lintasannya adalah 4 – 1 = 3.
- **Upapohon (*subtree*)**, adalah suatu bagian dari pohon struktur data yang dapat dilihat sebagai sebuah pohon lain yang berdiri sendiri. Simpul apapun dalam pohon *P*, bersama dengan

seluruh simpul dibawahnya, membentuk sebuah sub pohon dari *P*. Sub pohon yang terhubung dengan akar merupakan keseluruhan pohon tersebut.



Gambar 4.3 bagian yang dilingkari adalah upapohon.

- **Daun (*leaf*),** merupakan simpul paling rendah di dalam pohon dan tidak mempunyai anak. Pada contoh, anak dari pohon tersebut adalah 9, 14, 19, 67, dan 76.

Jika diamati, struktur dari XML sangat mirip dengan struktur pohon berakar. Penyimpanan data menggunakan struktur pohon dapat mempermudah pengaksesan informasi yang disimpan di dalam pohon tersebut.

Jika XML digambarkan sebagai pohon maka root node pada XML merupakan akar dari pohon. Anak dari akar pohon tersebut adalah element node dari XML. Di dalam XML juga dimungkinkan sebuah element node berisi element node lagi sehingga dapat membentuk upapohon.

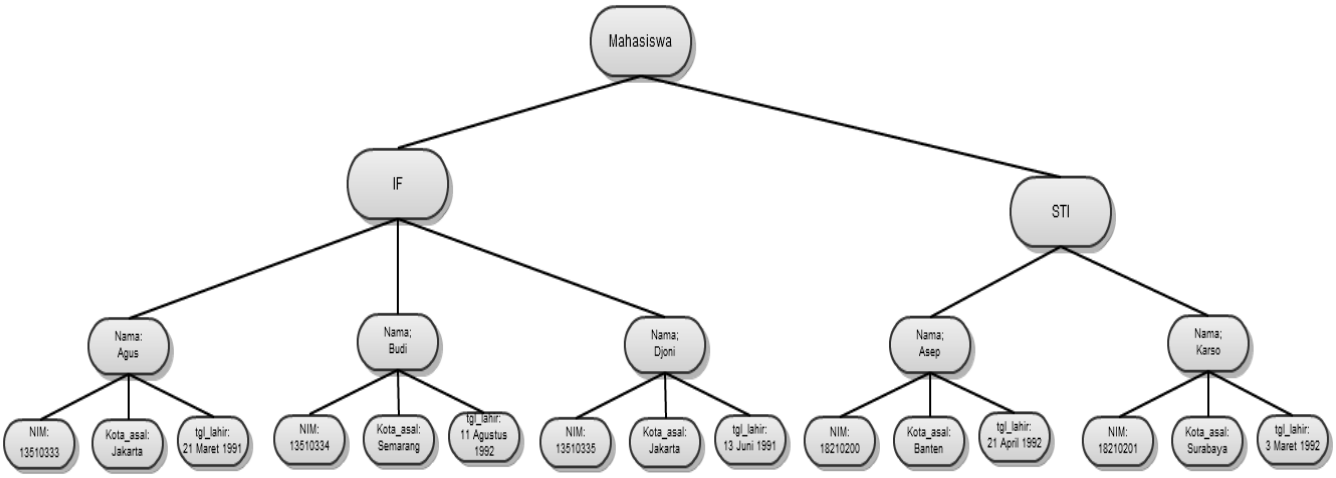
Pengaksesan data pada XML dimulai dari *root node* sama halnya dengan pengaksesan data pada pohon berakar yang dimulai dari akar. Setelah itu anak dari akar pohon diakses sampe ke bagian daun. Setiap simpul dari pohon merupakan himpunan bagian dari *parent*-nya. Penyimpanan data dengan cara itu memiliki keuntungan, yaitu data yang tersimpan dapat dikelompokan dengan rapi sesuai dengan himpunannya masing-masing.

Misalkan kita punya file XML sebagai berikut:

```
<?xml version="1.0" encoding="UTF-8"?>

<mahasiswa>
<prodi>IF</prodi>
        <biodata nama=”Agus”>
                <NIM>13510333</NIM>
                <kota_asal>Jakarta</kota_asal>
                <tgl_lahir>21 Maret 1991</tgl_lahir>
        </biodata>
        <biodata nama=”Budi”>
                <NIM>13510334</NIM>
                <kota_asal>Semarang</kota_asal>
                <tgl_lahir> 11 Agustus 1992</tgl_lahir>
        </biodata>
        <biodata nama=”Djoni”>
                <NIM>13510335</NIM>
                <kota_asal>Jakarta</kota_asal>
                <tgl_lahir> 13 Juni 1991 </tgl_lahir>
        </biodata>
<prodi>STI</prodi>
        <biodata nama=”Asep”>
                <NIM>18210200</NIM>
                <kota_asal>Banten</kota_asal>
                <tgl_lahir> 21 April 1992 </tgl_lahir>
        </biodata>
        <biodata nama=”Karso”>
                <NIM>18210201</NIM>
                <kota_asal>Surabaya</kota_asal>
                <tgl_lahir> 3 Maret 1992 </tgl_lahir>
        </biodata>
</mahasiswa>
```

File XML tersebut berisi data mahasiswa dari prodi IF dan STI. Setiap prodi memiliki data mahasiswa pada prodi tersebut.

File XML tersebut jika digambarkan sebagai pohon, akan menjadi sebagai berikut:



Pohon tersebut memiliki akar "Mahasiswa" dan pada XML merupakan root node. Setiap simpul di bawah "Mahasiswa" merupakan element node dari "Mahasiswa" karena berada di dalam tag <mahasiswa> .... </mahasiswa>

Pohon tersebut memiliki upapohon "IF" dan "STI". Dalam XML, IF dan STI merupakan himpunan bagian dari "Mahasiswa" dan merupakan "anak" dari "mahasiswa" karena berada di dalam tag "mahasiswa".

```
<mahasiswa>
    <IF>
    ...
    </IF>
    <STI>
    ...
    </STI>
</mahasiswa>
```

Daun dari pohon tersebut merupakan informasi dari masing-masing mahasiswa.

Jika diperhatikan, struktur XML dan pohon berakar terdapat banyak kesesuaian:

- Root node dalam XML merupakan akar dari pohon berakar.
- Element node dalam XML merupakan anak dari pohon berakar.
- Setiap element node dalam XML dapat berisi element node lainnya yang merupakan himpunan bagian dari *parent node* tersebut, dalam pohon berakar juga demikian. Setiap anak pohon dapat memiliki anak pohon lagi yang merupakan himpunan bagian dari *parent*-nya.
- Cara mengakses XML sama dengan cara mengakses infomasi dari pohon berakar. Pengaksesan dilakukan dari simpul paling atas, dalam kasus ini pengaksesan dimulai dari root node (XML) atau dari akar (pohon berakar).
- Upapohon dalam pohon berakar juga dapat terjadi pada XML. Kasus tersebut terjadi ketika anak pohon memiliki anak lagi. Pada contoh terdapat upapohon IF, STI, dan upapohon dari masing-masing simpul yang berisi informasi nama mahasiswa.

Cara mengakses data pada XML tersebut sama dengan cara mengakses data pada pohon berakar. Pengaksesan dilakukan dari atas dan menelusuri anak pohon sampai ke jenis data yang diinginkan.

Misalkan kita ingin mengakses data dari mahasiswa yang bernama Agus. Pertama-tama, simpul "mahasiswa" diakses lalu ditemukan 2 simpul lainnya yang terdiri dari simpul "IF" dan "STI". Kemudian dilakukan penelusuran *element node* dari masing-masing upapohon "IF" dan "STI".

Pada umumnya program untuk mem-*parsing* atau membaca XML melakukan iterasi dari *node* paling atas sampai *node* paling bawah. *Parser* akan mengecek akar lalu ke anak sampai ke bagian daun. Jika terdapat kekurangan atau kelebihan *tag*, *parser* akan menolak file XML tersebut dan akan terjadi *error*.

Penambahan elemen ke dalam file XML sama dengan penambahan elemen ke dalam pohon berakar. Pertama program menelusuri pohon XML sampai menemukan simpul yang dimaksud. Setelah itu dapat dilakukan penambahan daun baru atau anak baru sesuai dengan kebutuhan. Hal terpenting yang harus diperhatikan adalah, penambahan data harus memerhatikan *hierarki* dan jangan sampai melakukan perubahan *hierarki* yang dapat merusak keteraturan pohon XML.

V. KESIMPULAN

Struktur diskrit banyak diaplikasikan dalam kehidupan tanpa disadari. Dengan memanfaatkan struktur diskrit, penyelesaian suatu masalah dapat diatasi dengan mudah. Salahs satunya adalah masalah komunikasi data antar aplikasi atau bahasa program yang berbeda. Salah satunya adalah XML. XML dapat menjembatani pertukaran data antar aplikasi atau bahasa program.

XML adalah salah satu aplikasi dari pohon pada struktur diskrit. Penyimpanan dengan menggunakan stuktur pohon dapat mempermudah pengaksesan data. Selain itu, data yang disimpan dalam pohon dapat dengan mudah dikelompokkan. Setiap anak dari sebuah simpul merupakan himpunan bagian dari *parent*-nya.

REFERENCES

Rinaldi Munir, *Matematika Diskrit edisi keempat*, Informatika, 2008.

http://www.w3schools.com/xml/xml_tree.asp (waktu akses: 10 Desember 2011 pukul 19.00)

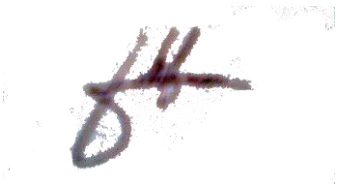
http://www.w3schools.com/dtd/default.asp (waktu akses: 10 Desember 2011 pukul 19.00)

http://en.wikipedia.org/wiki/XML (waktu akses: 10 Desember 2011 pukul 19.00)

PERNYATAAN

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

Bandung, 12 Desember 2011



Luthfi Chandra Fibrian
13510047