Analisis Empiris Atas *ORDPATH Encoding* Untuk Kinerja *Insert Node* Pada *Ordered XML Tree*

Irvanizam Zamanhuri
Jurusan Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Syiah Kuala
Jl. Syech Abdurrauf No. 5, Darussalam Banda Aceh 23111, Propinsi Aceh
Email: irvanizam.zamanhuri@informatika.unsyiah.ac.id

Masuk: 22 Januari 2015; Direvisi: 20 Februari 2015; Diterima: 23 Februari 2015

Abstract. *The eXtensible Markup Language (XML) has quickly become the de facto standard for data exchange via web. An XML document can be viewed as an ordered tree that has at least one node. Each node must be labeled by using a scheme approach to describe the XML data structure. There are two famous existing encodings, namely Dewey and Inteval Encodings. In this paper, ORDPATH encoding based on Dewey together with the two other encodings are empirically demontrated on dblp, nasa, and treebank datasets. The results show that while a new node was inserted into the tree, Dewey and Interval have to relabel the inserted node's siblings and modify the interval number of the sibling nodes, respectively. Whereas, the ORDPATH eliminates this problem by adding an even number used as a caret for the new insertion node.*
Keywords: *Ordered Tree, XML, ORDPATH.*

Abstrak. *EXtensible Markup Language (XML) terus menjadi standard untuk penukaran data melalui web. Sebuah dokumen XML dapat ditinjau menjadi tree terurut yang berisikan sedikitnya satu node. Setiap node harus dilabelkan menggunakan sebuah algoritma pelabelan untuk mendeskripsikan struktur data XML tersebut. Ada dua algoritma encoding yang terkenal selama ini, Dewey dan Interval encoding. Pada tulisan ini, metode ORDPATH yang berbasiskan Dewey bersama-sama dengan Dewey dan Interval didemontrasikan secara empiris dengan menggunakan dataset dblp, nasa, dan treebank. Hasil menunjukkan bahwa ketika node baru dimasukkan ke dalam tree, Dewey dan Interval harus melakukan pelabelan kembali dan memodifikasi interval sibling node. Akan tetapi, ORDPATH dapat mengatasi masalah ini dengan memberikan angka genap yang digunakan sebagai penanda untuk node baru.*
Kata Kunci: *Ordered Tree, XML, ORDPATH.*

1. Pendahuluan

Dewasa ini XML sangat cepat popular dan menjadi standar untuk pertukaran informasi melalui internet (Tatarinov, dkk., 2002; Lu, dkk., 2005; O'Neil, dkk., 2004; Soltan, dkk., 2006). XML dapat menyajikan beragam informasi yang dihimpun melalui sebuah berkas dokumen. Salah satu *website* yang menyajikan informasi terkemuka dengan menggunakan dokumen XML sebagai tempat penyimpanan datanya adalah *Database Very Large Project (DBLP)*. Website ini menghimpun lebih dari 1,9 juta artikel, jurnal, dan proseding *Computer Science* serta *link website* ilmuan komputer secara *up-to-date*.

Namun, penyimpanan data melalui dokumen XML menjadi hambatan ketika sebuah informasi baru harus ditambahkan ke dalam dokumen XML. Hal ini disebabkan karena besarnya ukuran *file* sehingga sulit mencari posisi yang tepat untuk menambahkan informasi baru tersebut. Selain itu, perubahan struktur dan alur informasi akan mengakibatkan perubahan yang besar pada struktur data XML.

Untuk mengatasi hambatan tersebut, banyak peneliti telah mengusulkan penggunaan sistem basis data yang terelasi dengan mentransformasikan struktur data XML ke dalam sebuah sistem basis data (Tatarinov, dkk., 2002; Deutsch, dkk., 1999; Florescu & Kossmann, 1999; Shanmugasundaran, 2001; Shimura, dkk., 1999). Namun yang menjadi fokus pada pendekatan

ini adalah mencari model *schema* untuk pelabelan XML terurut *(Ordered XML documents)* yang bagus, efisien, dan mampu mendeskripsikan struktur data XML seperti aslinya.

Sebuah dokumen XML dapat dipresentasikan menjadi *tree* yang terurut (Zamanhuri, 2014). Setiap *tree* mempunyai setidaknya sebuah *node* yang merupakan *root* dari *tree* atau terdiri dari beberapa *node*. Urutan *node* di dalam *tree* menunjukkan urutan data atau informasi yang berada pada dokumen original XML. Karena ini merupakan sebuah *tree*, maka konsep *children*, *sibling*, dan *parent nodes* dapat diterapkan pada dokumen XML.

Banyak peneliti telah mengusulkan metode-metode pelabelan *node* pada sebuah *tree* (Zamanhuri, 2014). Dua yang paling terkenal adalah Dewey (Tatarinov, dkk., 2002) dan *Interval* (Dagys, 2008) *encoding*. Metode Dewey mendefinisikan *node* dengan menampilkan urutan bilangan *integer* sehingga label setiap *node* menjadi seperti katalog buku di perpustakaan atau urutan daftar isi pada buku. Sedangkan *interval encoding* menglabelkan setiap *node* di dalam *tree* dengan mendefinisikan nilai *interval* yang ditentukan melalui penambahan *(increment)* nilai *integer* ketika setiap *node* dijelajah secara *pre-order traversal*.

Selain itu, terdapat juga beberapa metode pelabelan *node* baru yang berbasiskan Dewey telah ditemukan. Salah satunya adalah yang diusulkan melalui jurnal (O'Neil, dkk., 2004) yaitu ORDPATH *encoding*. Metode ini menginisialisasikan setiap label *node* dengan bilangan ganjil *integer* dan menggunakan bilangan genap sebagai tanda sisipan *(caret)* ketika sebuah *node* baru ditambahkan ke dalam *tree*. Ketiga metode *encoding* yang disebutkan di atas mempunyai cara dan teknik penambahan *node* baru yang berbeda-beda.

Dalam penulisan ini, penulis akan menunjukkan dan membandingkan proses penambahan *node* dengan menggunakan ketiga *encoding* tersebut dan menampilkan kelebihan ORDPATH *encoding* dari kedua *encoding* yang lain terutama pada kasus penambahan *node* baru. Selain itu, penulis juga akan membahas kembali ide dan gambaran algoritma pelabelan *node* menggunakan metode ORDPATH.

2. Tinjauan Pustaka

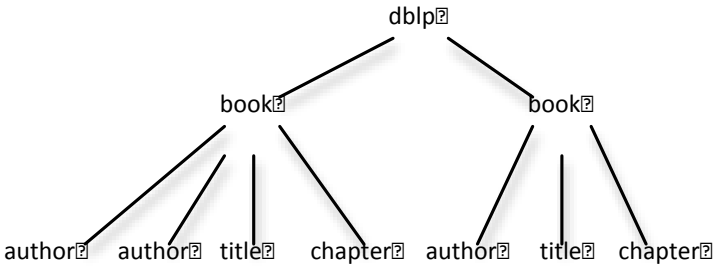
2.1. *Ordered XML Tree*

Sebuah dokumen XML yang berisikan elemen, attribut dan data dapat dilihat sebagai sebuah *tree* (Tatarinov, dkk., 2002). *Tree* mempunyai *root* dan *node-node* sebagai representasi dari elemen, atribut dan data dari dokumen original XML. Dokumen XML tersebut merupakan model data terurut sehingga posisi suatu elemen XML tidak dapat dipindahkan ke posisi lain. Perpindahan ini mengakibatkan perubahan urutan *node* di dalam XML *tree*. Kode 1 dan Gambar 1 memperlihatkan dokumen XML yang sangat sederhana dan hasil konversi XML ke dalam ordered XML *tree*.

Kode 1. Dokumen XML Sederhana

```
<dblp>
  <book>
     <author>Data Mining in Bio</author>
     <author>Jason Tsong</author>
     <title>Dennis Shasha</title>
     <chapter>Chapter 3</chapter>
  </book>
  <book>
     <author>ANSI / C</author>
     <title>Dennis Richie</title>
     <chapter>Chapter 1</chapter>
  </book>
</dblp>
```

Sebuah *tree* dibangun oleh sedikitnya satu *node* atau beberapa *node-node* yang saling berhubungan atau dikenal dengan *complete tree*. *Complete Tree* Sebuah *node* memiliki setidaknya *parent*, *sibling*, dan *children node*. Dari Gambar 1, *node* dblp merupakan *parent node* dari *node book*. *Node author*, *title*, dan *chapter* adalah *children node* dari *node book*. Sedangkan *node title* merupakan *sibling node* dari *node author*.



Gambar 1. Konversi Dokumen XML Menjadi XML *Tree*

2.2. Properti *Tree*

Menurut Gros & Yellen (2010), *directed tree* adalah sebuah graf yang mempunyai arah. Salah satu contoh graf ini adalah *rooted tree* yang mana setiap *edge*-nya secara implisit berarah dari akar *tree*-nya. Dalam *rooted tree*, *depth* atau *level* dari sebuah *vertex v* adalah jarak *vertex v* dengan *root* dan *height* dari *rooted tree* adalah jarak *path* yang terpanjang dari *root*. Selain itu, *rooted tree* juga mempunyai beberapa properti penting seperti *parent*, *child*, *siblings*, *descedant*, *ancestor*, *leaf*, dan *internal vertex*.

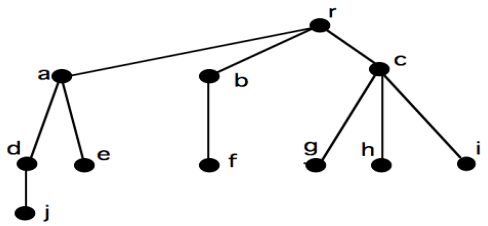
Definisi 1. Jika *vertex v* secara langsung terhubung dengan *vertex w* melalui sebuah path dari *root* ke *w*, maka *v* adalah ***parent*** dari *w*, dan *w* adalah ***child*** dari *v*.

Definisi 2. *Vertex-vertex* yang mempunyai *parent* yang sama dinamakan dengan ***siblings***.

Definisi 3. *Vertex w* dinamakan ***descendant*** dari *vertex v* (dan *v* disebut ***ancestor*** dari *w*), jika *v* berada pada *path* yg unik dari *root* ke *w*. Jika $w \neq v$, maka *w* adalah ***proper descendant*** dari *w* (dan *v* adalah ***proper ancestor*** dari *w*).

Definisi 4. ***Leaf*** adalah *vertex* yang tidak mempunyai children. Sedangkan ***internal vertex*** adalah *vertex* yang mempunyai paling sedikit satu *child*.

Gambar 2 memperlihatkan contoh-contoh dari ketujuh properti dari *rooted tree*. Gambar 2 dapat dijelaskan bahwa *height* dari *tree* tersebut adalah 3. *Vertex* r, a, b, c, dan d merupakan *Interval Vertex*. *Vertex* e, f, g, h, i, dan j adalah *leaf* dari *tree*. *Vertex* g, h, dan i adalah *siblings*. Kemudian, *vertex* a merupakan *ancestor* dari j, dan j merupakan *descendant* dari a.



Gambar 2. Contoh *Rooted Tree*

2.3. Konsep Dewey, *Interval*, dan *ORDPATH Encoding*

Metode Dewey *encoding* pertama sekali diperkenalkan oleh Tatarinov merupakan metode pelabelan XML *tree* seperti pengkode katalog buku. Dalam metode ini, setiap *node* dilabelkan dengan menggunakan kombinasi antara label *parent node*-nya dengan sebuah bilangan positif *integer*. Jika *u* merupakan anak ke-*x* dari *node s* pada XML *tree*, maka label untuk *node u*, label(*u*) adalah penggabungan label dari *node s* dan x yang dipisahkan oleh sebuah karakter '.', atau dipresentasikan dengan label(*u*) = label(*s*).x. Sebagai contoh, jika label *node u* adalah 1.5.3, maka label untuk *child node* yang ke-5 dari *node u* adalah 1.5.3.5.

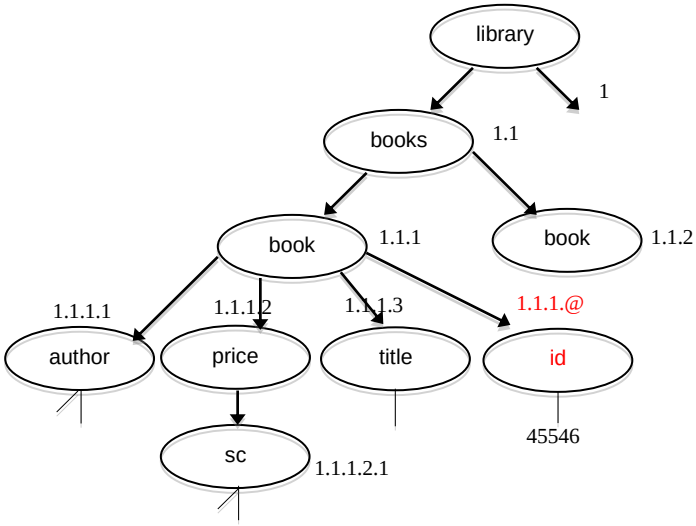
Interval encoding merupakan metode *encoding* untuk melabelkan *node* dengan

menggunakan nilai *interval (range)* dengan teknik penelusuran *pre-order traversal* (Augsten, 2009). Nilai *interval* dari sebuah *children* akan selalu berada di antara nilai *interval parent*nya.

Sedangkan ORDPATH *encoding* yang dipaparkan pada tulisan (O'Neil, dkk., 2004) merupakan metode *encoding* berbasiskan Dewey yang menggunakan bilangan ganjil. Namun, metode ini juga menggunakan bilangan genap sebagai tanda sisipan *(caret)* untuk menambahkan *node* baru ke dalam *tree*.

2.4. Penyimpanan Elemen XML ke dalam Sistem Manajemen Basis Data

Dokumen XML dapat disimpan dan dikelola ke dalam *Database Manajement System* (DBMS) menggunakan *object relational* DBMS (Ferraz, dkk., 2010). Menurut Tatarinov, dkk. (2002), dokumen tersebut dipecahkan menjadi dua relasi: *Path (id, path)* dan *Edge (dewey, path_id, value)*. Pada relasi *Edge*, atribut Dewey menyimpan kode *node* yang dibentuk melalui algoritma Dewey. Sedangkan relasi *Path* menyimpan informasi tentang *path* dari elemen XML yang disimpan. Nama *node* tidak pernah disimpan di dalam relasi *Edge* karena nama *node* dapat diperoleh melalui atribut *path* pada relasi *Path*. Contohnya, hirarki dokumen XML dari Gambar 3 yang telah dibentuk kode *node* menggunakan algoritma Dewey dapat disimpan ke dalam DBMS. Tabel 1 dan 2 adalah tabel relasi *Path* dan *Edge* dari Gambar 3. Sedangkan Tabel 3 adalah tabel relasi dokumen.



Gambar 3. Hirarki Dokumen XML dengan Algoritma *Dewey*

Ketika semua elemen dari dokumen XML sudah ada di dalam tabel relasi *Path* dan *Edge*, maka *query-query* seperti *parent*, *children*, *ancestor*, *descendant*, dan *sibling* dapat diperoleh dengan menggunakan ekspresi *Structure Query Language (SQL)*.

Tabel 1. Tabel Relasi *Path*

id	*Path*
1	*library*
2	*library/books*
3	*library/books/book*
4	*library/books/book/@id*

Tabel 2. Tabel Relasi *Edge*

value	*path_id*	*dewey*	*doc_id*
	1	1	1
	2	1.1	1
	3	1.1.1	1
	3	1.1.2	1
45546	4	1.1.1.@	1

Tabel 3. Tabel Relasi Dokumen

id	*name*
1	*Library*
2	*Cars*
3	*Videos*
4	*Universities*

3. Metodologi Penelitian

3.1. Bahan

Bahan yang digunakan dalam penelitian ini adalah delapan jenis *dataset* dalam bentuk dokumen XML yang berupa *file* XML dblp, nasa, dan *Treebank*. *Dataset* dblp berisikan lebih dari 1.9 juta artikel, jurnal, dan *proceeding Computer Science*. *Dataset* nasa berisikan tentang data asronomik. Sedangkan *dataset Treebank* berisikan data linguistik dalam Bahasa Inggris. Karakteristik dari *dataset* tersebut dapat dilampirkan pada Tabel 4.

Tabel 4. Dokumen XML *(dataset)*

Dataset	*Ukuran (MB)*	*Jumlah Node*	*Max/Avg depth*	*Max/Avg fanout*
Dblp-1	0.1	2K	2/1	199/1999
Dblp-2	1.5	30K	3/3	974/56
Dblp-3	4.1	79K	3/3	8662/107
Dblp-4	7.2	139K	3/3	15K/107
nasa	23.8	533K	10/7	2K/225
*Tree*bank	85.4	2M	36/8	56K/1.6K
Dblp-5	260	5M	3/3	79K/2K
Dblp-6	592	11M	3/3	92K/3K

3.2. Metode Penelitian

Untuk membandingkan antara ketiga metode *encoding*, proses penambahan *node* dengan algortima ketiga *encoding* tersebut diimplementasikan dengan menggunakan *Structure Query Language (SQL)*. Dalam penelitian ini, ada dua langkah yang dilakukan. Pertama, transformasi dokumen XML ke dalam sebuah sistem basis data. Sebelum melakukan proses perbandingan, tiga buah sistem basis data dibentuk secara terpisah. Masing-masing basis data tersebut mempunyai delapan tabel yang berisikan kedelapan *dataset* dari dokumen original XML dan masing-masing tabel mempunyai satu atribut *primary key* yang berisikan label *node* dari ketiga metode *encoding* itu.

Kedua, implementasi proses penambahan *node*. Untuk melihat perbedaan proses penambahan *node* baru, satu perintah SQL diimplementasikan untuk menambahkan *node* baru dan memodifikasikan label *sibling node* yang diakibatkan karena penambahan *node* baru untuk masing-masting *encoding*. Kemudian rata-rata total *node* yang harus dilabelkan kembali dibandingkan dan ditampilkan melalui grafik. Eksperimen ini dicoba pada dua jenis *dataset* yang berbeda, yaitu *dataset* yang dalam *(deep dataset)* dan yang dangkal *(wide dataset)*.

Penelitian ini menggunakan *deep dataset Treebank* yang berisikan lebih dari 2 juta *node*, 36 tingkat maksimum kedalaman *tree*, dan lebih dari 16 ribu rata-rata *fanout*nya. Sedangkan untuk *wide dataset*, penelitian ini menggunakan *dataset* dblp-5.

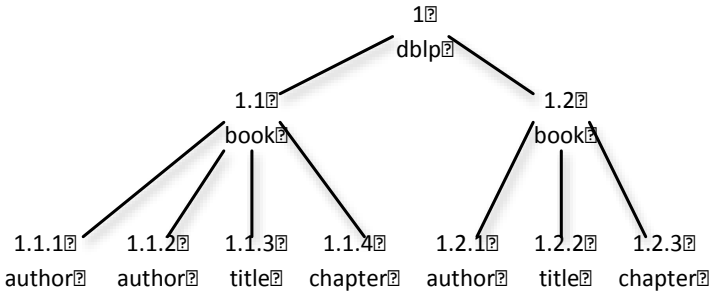
4. Pengujian dan Pembahasan

4.1. *Dewey Encoding*

Dewey encoding yang diusulkan melalui jurnal (Tatarinov, dkk., 2002) merupakan metode pelabelan XML *tree* dengan menggunakan kode katalog buku. Dalam metode ini, setiap *node* dilabelkan dengan menggunakan kombinasi antara label *parent node*-nya dengan sebuah bilangan positif *integer*. Jika *u* merupakan anak ke-*x* dari *node s* pada XML *tree*, maka label untuk *node u*, label(*u*) adalah penggabungan label dari *node s* dan x yang dipisahkan oleh sebuah karakter '.', atau dipresentasikan dengan label(*u*) = label(*s*).x. Sebagai contoh, jika label *node u* adalah 1.5.3, maka label untuk *child node* yang ke-5 dari *node u* adalah 1.5.3.5.

Metode ini sangat efektif untuk menentukan *parent node* atau *sibling* dari sebuah *node*. Jika label *node u* adalah 1.5.3.5, maka *parent node* dari *node u* adalah 1.5.3 yaitu dengan menghapus sebuah *integer* terakhir dari label *node u*. Sedangkan semua label *node* yang memiliki label *parent* yang sama dengan parent(*u*) merupakan *sibling* dari *node u*. Sebagai contoh, label *node* 1.5.3.1, 1.5.3.6, dan 1.5.3.2 adalah *sibling* dari *node u* (berlabel 1.5.3.5). Gambar 4 menampilkan dokumen XML (Kode 1) dipresentasikan menggunakan metode Dewey.

Metode Dewey sangat lemah dalam melakukan proses penambahan *node* baru (Tatarinov, dkk., 2002). Proses ini melibatkan pelabelan kembali terhadap sibling *node* dari *node* baru tersebut. Misalnya sebuah *node* baru dimasukkan menjadi *child node* ke-2 dari *node* yang berlabel 1.2 (*book*) dan dengan label 1.2.2 (baru). Kemudian Dewey akan melakukan proses label ulang terhadap label *node* 1.2.2 (*title*) menjadi label *node* 1.2.3, begitu juga label *node* 1.2.3 (*chapter*) menjadi label *node* 1.2.4. Hal ini sangat tidak efesien dilakukan ketika sebuah *node* baru ditambahkan ke dalam *tree* dan menjadi *child node* ke-1 dari *root*. Akibatnya semua *node* kecuali *root* harus dilakukan proses pelabelan ulang.

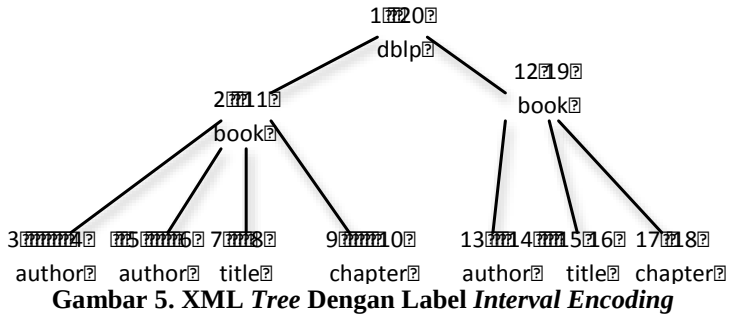


Gambar 4. Konversi Dokumen XML Menjadi XML *Tree*

4.2. *Interval Encoding*

Interval encoding merupakan metode *encoding* untuk melabelkan *node* dengan menggunakan nilai *interval (range)* dengan teknik penelusuran *pre-order traversal* (Augsten, 2009). Setiap *node* mempunyai nilai *interval* yang terdiri dari dua nilai, *left* dan *right*. Karena penelusuran diawali melalui *root*, maka nilai *left* untuk *root* adalah 1. Kemudian *children* sebelah kiri dari *root* ditelusuri dan diberikan nilai *left*nya dengan meng-*increment*-kan nilai *left* sebelumnya. Begitu juga ketika sebuah *node* telah selesai ditelusuri, maka nilai *right* dari *node* tersebut diinisialisasikan. Gambar 5 menunjukkan XML *tree* dengan menggunakan label *Interval encoding*.

Metode ini secara mudah dapat menentukan *children*/*descendant* dan *parent*/*ancestor* dari sebuah *node*. Jika *node u* mempunyai nilai *interval* yang berada diantara nilai *left* dan *right* dari *node s*, maka *node u* adalah *children* atau *descendant* dari *node s*. Sebagai contoh label *node* s_1 dengan nilai *interval* <2,11> dan *node* s_2 dengan nilai *interval* <12,19> merupakan *children* dari *node* s_3 bernilai *interval* <1,20>. Begitu juga sebaliknya, jika nilai *interval node s* berada di dalam nilai *interval node u* dan nilai right dari *node s* lebih kecil dari nilai *right* pada *node u*, maka *node u* merupakan *parent* atau bisa jadi *ancestor* dari *node s*. Sebagai contoh, karena nilai *interval node* s_1 <2,11> berada di dalam nilai *interval node* s_2 <1,20>, dan nilai *interval* right dari *node* s_1 lebih kecil dari nilai *interval* right dari *node* s_2, maka s_2 adalah *parent* atau *ancestor* dari *node* s_1.



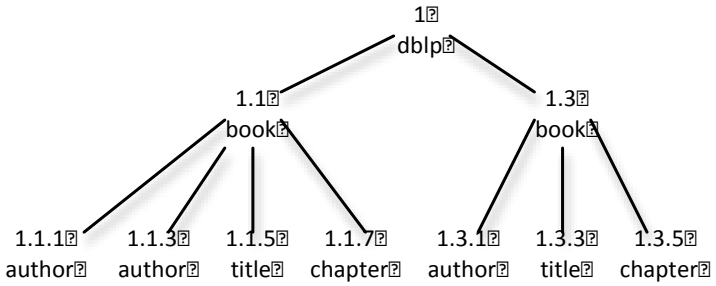
Gambar 5. XML *Tree* Dengan Label *Interval Encoding*

Sama halnya seperti dewey, *interval encoding* juga masih tidak mendukung proses penambahan *node* baru secara efisien. Karena jarak nilai *interval* dari *leaf node* adalah 1, maka proses pelabelan ulang terhadap *sibling node* tidak dapat dihindari. Kasus proses pelabelan *node* kembali yang terjadi paling lama adalah ketika sebuah *node* baru ditambahkan ke dalam *tree* dan menjadi *child node* ke-1 dari *root*. Pada kasus ini, *interval encoding* harus mendefinisikan kembali nilai *interval* dari semua *node* di dalam *tree*, termasuk juga *root* dari *tree* itu sendiri.

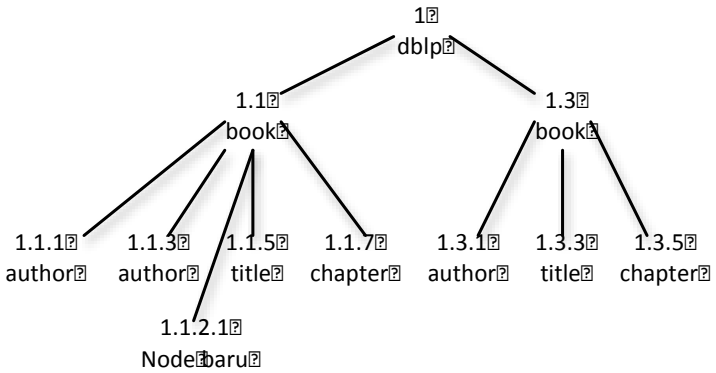
4.3. ORDPATH *Encoding*

ORDPATH *encoding* yang dipaparkan pada tulisan (O'Neil, dkk., 2004) merupakan metode *encoding* berbasiskan Dewey yang menggunakan bilangan ganjil (lihat Gambar 6). Namun, metode ini juga menggunakan bilangan genap sebagai tanda sisipan *(caret)* untuk menambahkan *node* baru ke dalam *tree*. Tanda sisipan dilokasikan diantara dua buah bilangan

ganjil. Sebagai contoh pada Gambar 6 ORDPATH memberikan label kepada *node* baru yang berada di antara ”*author*” (1.1.3) dan ”*title*” (1.1.5). Ketika *node* baru ditambahkan diantara kedua *node* tersebut, maka ORDPATH membentuk sebuah *caret*, bilangan genap antara 3 dan 5, yaitu 2 serta menggabungkan label *parent* mereka dengan *caret* tersebut dan bilangan ganjil yang lainnya (misalnya 1). Sehingga label untuk *node* baru tersebut menjadi 1.1.2.1 tanpa harus memberikan label ulang kepada *node* ”*author*”(1.1.3) dan ”*title*”(1.1.5). Gambar 7 menunjukkan proses penambahan *node* baru diantara dua buah *node* yang dilabelkan dengan menggunakan metode ORDPATH.



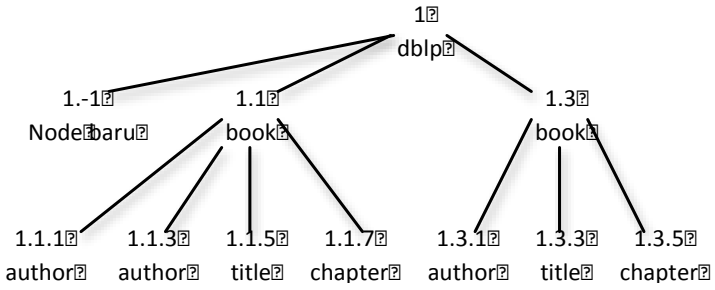
Gambar 6. XML *Tree* Dengan Label Dewey *Encoding*



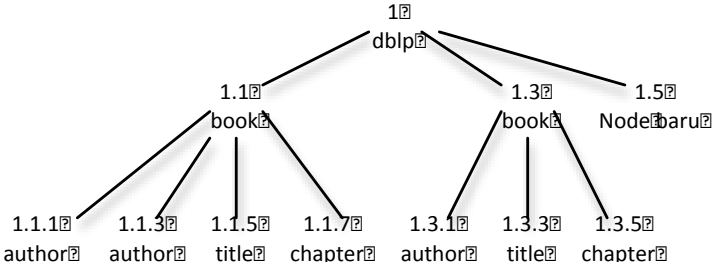
Gambar 7. Penambahan *Node* Baru Pada Posisi Diantara Dua Buah *Node* Menggunakan ORDPATH *Encoding*

Untuk menambahkan *node* baru sebagai *child* pertama dari sebuah *node*, ORDPATH memberikan label kepada *node* tersebut dengan menggabungkan label *parent* dari *node* baru dengan bilangan ganjil negatif. Sebagai contoh, jika *node* baru ditambahkan pada posisi sebelum *node* ”*book*”(1.1), maka label untuk *node* baru adalah 1.-1 tanpa harus memodifikasikan label untuk *node sibling*nya. Gambar 8 memperlihatkan sebuah *node* baru ditambahkan pada posisi sebagai *child* pertama dari *root*.

Sedangkan untuk menambahkan *node* baru sebagai child terakhir dari sebuah *node*, ORDPATH mendefinisikan label *node* tersebut dengan menambahkan bilangan ganjil dari *node* terakhir dengan bilangan 2. Sebagai contoh, jika *node* baru ditambahkan pada posisi sesudah *node* ”*book*”(1.3), maka label untuk *node* baru adalah 1.(3+2) = 1.5 tanpa harus memodifikasikan label *node* yang lain. Gambar 9 memperlihatkan sebuah *node* baru ditambahkan pada posisi sebagai *child* terakhir dari *root*.



Gambar 8. Penambahan *Node* Baru Pada Posisi *Child* Pertama Dari *Root Node* Menggunakan ORDPATH *encoding*

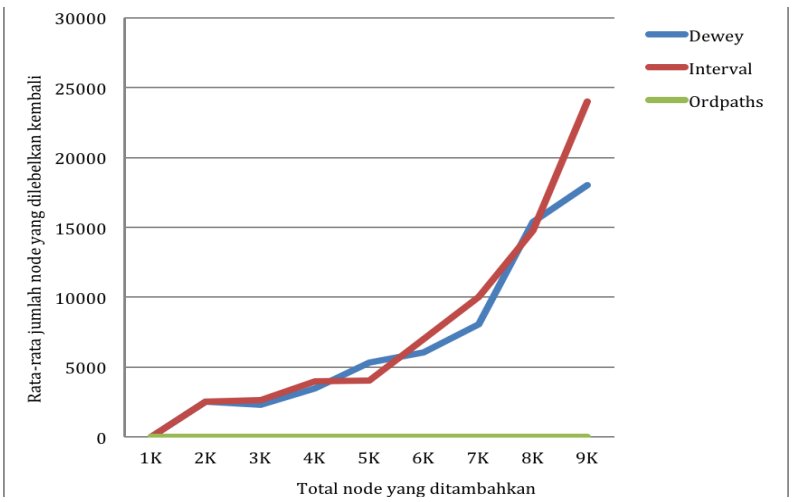


Gambar 9. Penambahan *Node* Baru Pada Posisi *Child* Terakhir Dari *Root Node* Menggunakan ORDPATH *Encoding*

3.4. Analisis Proses Penambahan *Node*

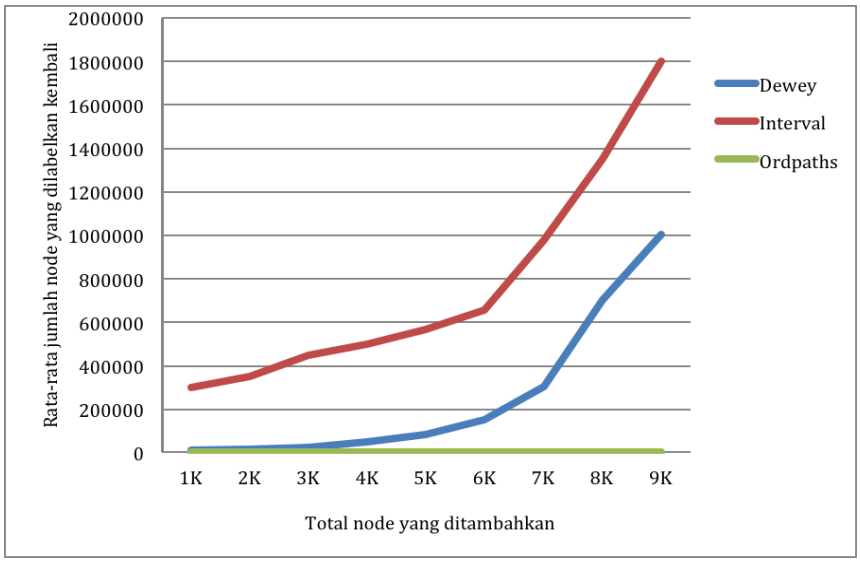
Dalam bagian ini, proses penambahan *node* baru dengan menggunakan ketiga metode tersebut ditampilkan dan membandingkan dalam bentuk grafik garis. Untuk eksperimen ini, penambahan *node* dilakukan dengan menambahkan beberapa *node* pada posisi acak di dalam dua jenis *tree* yang berbeda *(deep dan wide trees)*, dan diperlihatkan perbandingan melalui grafik. Gambar 10 menampilkan jumlah *node* yang harus dilabelkan kembali oleh ketiga *encoding* setelah proses penambahan beberapa *node* baru dilakukan pada posisi acak dengan menggunakan *wide dataset*.

Pada Gambar 10 sangat jelas ditampilkan bahwa ORDPATH *encoding* mampu mengeliminasikan proses pelabelan ulang saat *node* ditambahkan ke dalam XML *tree*. Sedangkan Dewey dan *interval encoding* cenderung memodifikasi label *node* dalam jumlah yang sangat besar ketika proses penambahan *node* dilakukan berulang kali.



Gambar 10. Grafik Perbandingan *Encoding* Dalam Proses Penambahan *Node* Pada *Wide Dataset*

Sedangkan pada Gambar 11, ORDPATH *encoding* masih menunjukkan kehandalannya dalam menambahkan *node* ke dalam *deep tree*. Penambahan *node* dengan metode ini tidak mempengaruhi bentuk, besar, dan kedalaman dari sebuah *tree*. Namun Dewey sedikit ada peningkatan pada *dataset* ini dan lebih baik dibandingkan dengan *deep dataset*. Hal ini disebabkan karena kedalaman *tree* dan jumlah *children* dari suatu *node* mempengaruhi proses *relabeling*. Dewey akan cepat melakukan proses penambahan *node* jika jumlah *sibling* dari *node* baru tersebut tidak begitu banyak. Beda halnya dengan *interval encoding*, dimanapun posisi *node* yang akan ditambahkan, maka seluruh *sibling* dan *descendant* dari *node* tersebut harus dilabelkan kembali.



Gambar 11. Grafik Perbandingan *Encoding* Dalam Proses Penambahan *Node* Pada *Deep Dataset*

4. Kesimpulan

Dari penelitian ini dapat ditarik kesimpulan: (1) ORDPATH *encoding* sangat efisien digunakan untuk proses penambahan *node* pada semua bentuk XML *tree*. (2) Dewey *encoding* cenderung baik digunakan jika XML *tree* berbentuk datar *(wide)* dengan jumlah *node* yang tidak begitu besar. (3) *Interval encoding* sangat tergantung dari jumlah *node* yang ada di dalam XML *tree* dan tidak terpengaruh dari kedalaman dan bentuk *tree* itu sendiri. (4) Penggunaan *caret* pada ORDPATH *encoding* sangat efektif diterapkan untuk mengeliminasi proses *relabeling node* dan tidak ada pengaruh terhadap urutan *node* di dalam XML *tree*.

Referensi

Augsten, N. 2009. *Ordered Labeled Trees in a Relational Database, Approximation: Theory and Algorithm Course*. Bozen-Bolzano: Free University of Bozen-Bolzano.

Dagys, E. 2008. *Storing XML using Interval Encoding with Sparse Numbering*. Thesis tidak diterbitkan. Bolzano-Italia: Free University of Bozen-Bolzano.

Deutsch, A., Fernandez, M. & Suciu, D. 1999. *Storing Semistructured Data with STORED*. Proc. Of SIGMOD Conference 1999.

Ferraz, C., A., Braganholo, V. & Mattoso, M. 2010. ARAXA: Storing and Managing Active XML Documents. *Web Semantics: Science, Services, and Agents on the World Wide Web 8*, hal. 209-224.

Florescu, D. & Kossmann, D. 1999. Storing and Querying XML Data using an RDBMS. *IEEE Data Engineering Buletin 22*(3).

Gros, J., & Yellen, J. 2010. *Graph Theory and its Appplication*. London: Chapman & Hall.

Lu, J., Wang Ling, T., Chan, C. & Chan, T. 2005. *From Region Encoding to Extended Dewey:*

On Efficient Processing of XML Twig Pattern Matching. VLDB 2005, pp. 193–204.

O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Shaller, G. & Westbury, N. 2004. *ORDPATH: Insert-Friendly XML Node Labels*. SIGMOD 2004, pp. 903-908.

Shanmugasundaran, J. 2001. *A General Technique for Querying XML Documents using a Relational Database System*. SIGMOD Record, September 2001.

Shimura, T., Yoshikawa, M. & Uemura S. 1999. *Storage and Retrieval of XML Documents using Object-Relational Database*. Proc. Of DEXA Conference.

Soltan, S., Zarnani, A., AliMohammadzadeh, R. & Rahgozar, M. 2006. *IFDewey: A New Insert-Friendly Labeling Schema for XML Data*. World Academic of Science, Engineering and Technology, 13, 2006, hal. 116-118.

Tatarinov, I., Viglas, S.D., Beyer, K., Shanmugasundaran, J., Shekita, E. & Zang C. 2002. *Storing and Querying Ordered XML Using a Relational Database System*. ACM SIGMOD 2002, June 4-6, Madison, Wisconsin, USA.

Zamanhuri, I. 2014. *FDewey Encoding: An approach Based on Dewey for Storing an XML Document into Database*. The 2nd International Conference on Natural and Enviromental Science (ICONES) 2014, hal. 207-212.