International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:1, 2008

IFDewey: A New Insert-Friendly Labeling Schema for XML Data

S. Soltan, A. Zarnani, R. AliMohammadzadeh, and M. Rahgozar

***Abstract*—**XML has become a popular standard for information exchange via web. Each XML document can be presented as a rooted, ordered, labeled tree. The Node label shows the exact position of a node in the original document. Region and Dewey encoding are two famous methods of labeling trees. In this paper, we propose a new insert friendly labeling method named IFDewey based on recently proposed scheme, called Extended Dewey. In Extended Dewey many labels must be modified when a new node is inserted into the XML tree. Our method eliminates this problem by reserving even numbers for future insertion. Numbers generated by Extended Dewey may be even or odd. IFDewey modifies Extended Dewey so that only odd numbers are generated and even numbers can then be used for a much easier insertion of nodes.

***Keywords*—**XML, Tree Labeling, Query Processing.

I. Introduction

TREE Labeling plays a key role in XML query processing. Every query processing algorithm depends deeply on the way in which XML tree is labeled. Various tree labeling schemas have been proposed [1, 2, 3, 7, 8] and some of them are widely used by XML query processors. Two major labeling methods are Region [4] and Dewey [3] encoding. Many query processing algorithms [4, 5, 6, 9] have employed Region encoding for labeling tree nodes. In [1], Extended Dewey was proposed and applied for XML query processing algorithm called TJFast. The advantage of this method is that, the path to reach any node from the root can be derived from its label using a Finite State Transducer (FST). The main problem of this method is the lack of flexibility for inserting new nodes. In the Ordpaths method [2], another extension for original Dewey encoding method, the authors apply the idea of using even numbers for insertion of the new nodes. In this paper we propose a new insert-friendly labeling method named IFDewey. We took advantages of both methods (Ordpaths and Extended Dewey) and introduced a new labeling schema named IFDewey.

We modified Extended Dewey so that like the method used in Ordpaths, only odd numbers are generated. With the help of even numbers new nodes can be easily added to XML tree. TJFast algorithm needs no extra changes for working with our new labeling method. Section 2 briefly reviews Original Dewey and demonstrates how Extended Dewey labels XML tree nodes and also describes the way that these labels are converted into XML tag names with the help of FST. In section 3, we explain IFDewey and show how our method has modified Extended Dewey to make it Insert-Friendly. We conclude the paper and discuss the future works in Section 4.

S. Soltan, A. Zarnani and R. AliMohammadzadeh are with Database Research Group, Faculty of ECE, School of Engineering, University of Tehran, Iran (e-mail: s.soltan,r.mohammadzadeh,a.zarnani@ece.ut.ac.ir).

M. Rahgozar is with Control and Intelligent Processing Center of Excellence, Faculty of ECE, School of Engineering, University of Tehran,Iran (e-mail: rahgozar@ut.ac.ir).

II. Preliminiaries

A. Original Dewey and Ordpaths Encoding

Dewey encoding is proposed in [3] for labeling XML trees. In this method, each node label is a combination of its parent label and an integer number. If *u* is the *x-th* child of *s* in XML tree then label of *u*, label(*u*), is concatenation of label of *s* and *x* which is presented as label(*s*).*x*. For example if element label for *u* is 2.5.3 then its 5th child label will be 2.5.3.5. The advantage of this method is that for any element label, we can easily extract node labels of its ancestors. For instance if an element label is 5.2.3.1 then its parent label is 5.2.3, its first ancestor label is 5.2 and so on.

Ordpaths [2] modified original Dewey encoding method so that it became insert-friendly. In this method the even numbers are reserved for adding the new nodes. When an XML tree is labeled for the first time, only odd numbers are generated. Considering two sibling elements with the labels 1.3.5.3 and 1.3.5.5, when a new node is added between these two nodes, using reserved even number "4", its label will be 1.3.5.4.1. Original Dewey was further improved in [1] for increasing the query processing performance.

B. Extended Dewey Encoding and FST

In this section we demonstrate the Extended Dewey Encoding proposed in [1]. Figure 1 shows a simple tree which is built from an XML document. The code above each node represents Extended Dewey label. Extended Dewey needs some schema information for labeling. Schema information can be extracted from DTD or the original document. In an XML document all distinct child names of any tag make a set, named child clue. $CT(t) = \{t_0, t_1, \ldots, t_{n-1}\}$ denotes the child name clue of tag t. Considering the following DTD:

<! ELEMENT bib (book*)>
<! ELEMENT book (author+, title, chapter*)>,

{author, title, chapter} presents the child clue of book tag. With the help of the child clue, an element name can be derived from the node label. Like original Dewey, Extended Dewey code for each element is a combination of its parent label and a postfix integer number (x_i). For any element e_i with

the name t_i Extended Dewey assigns an integer number, x_i, to e_i such that $x_i \bmod n_i = i$. For instance "title" Extended Dewey label is 0.4 because 4 mod 3 is 1 (1 is "title" index in book clue).

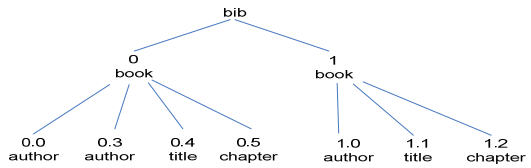

Fig. 1 XML tree with Extended Dewey labels

Suppose element name of *u* is *k-th* tag in $CT(t_s)$ ($k=0, 1, \ldots, n-1$) then **Extended Dewey** postfix code (x) for *u* (child of *s*) generates as follows:

1. if *u* is the first child of s ,then $x=k$
2. otherwise if we assume that *y* is the last component of the label of the left sibling u, then

$$x=\begin{cases}\left\lfloor \frac{y}{n} \right\rfloor \cdot n + k & if\ (y \bmod n) < k \\ \left\lceil \frac{y}{n} \right\rceil \cdot n + k & otherwise\end{cases}$$

For instance, label for "title"(*0.4*) tag is calculated as follows:

$y=3$, $n=3$ and $k=1$ so $x=\left\lfloor \frac{3}{3} \right\rfloor \cdot 3+1 = 4$, then the label of parent which is *"0"* will be attached to this number and final label *0.4* is generated.

The Next example shows how Extended Dewey labels are converted into Root-to-Node tags. Fig. 2 shows FST which is created from our sample DTD.

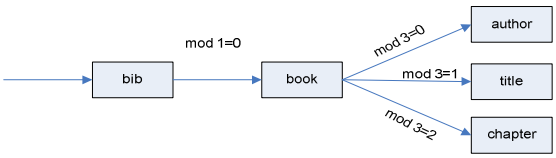

Fig. 2 Extended Dewey FST for sample DTD

Root-to-Node tags for "chapter"(0.5) are found as follows:

"*bib*" tag is common for all nodes so we go to "*bib*" state .In "*bib*" state We calculate 0 mod 1. It's 0 so we go to "book" state .In "book" state we calculate 5 mod 3. It's 2 and we go to "chapter" state. Finally Root-to-Node tags "*bib, book, chapter*" are generated.

III. IFDewey

Adding a new node into XML tree labeled by Extended Dewey leads to modification of several nodes. Supposing XML tree in Figure 1, if we want to add another "author" tag between "author"(0.3) and "title"(0.4), then labels for "title"(0.4) and "chapter"(0.5) should be recalculated. We will show that this problem can be eliminated by modifying some parts of Extended Dewey formula .Numbers generated by Extended Dewey formula are a mix of odd and even numbers. For instance numbers *0*, *3, 4* and *5* are assigned to children of "book"(*0*). We changed the formula so that only odd numbers are generated and even numbers will be reserved for future insertion. The code length can increasingly become so large. This problem is also seen in the methods proposed in [1, 2]. Fig. 3 shows the XML tree with IFDewey labels.

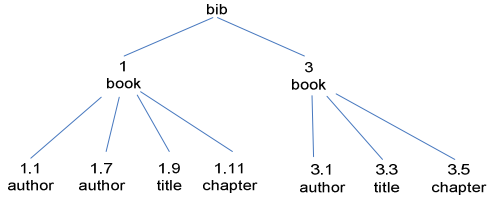

Fig. 3 XML tree with IFDewey labels

Suppose element name of u is *k-th* tag in *CT(ts)* ($k=0, 1, \ldots, n-1$) then **IFDewey** postfix code (x) for u (child of s) generates as follows :

1. if *u* is the first child of *s*, then $x=2k+1$
2. otherwise if we assume that *y* is the last component of the label of the left sibling u, then

$$x=\begin{cases}\left\lfloor \frac{y}{2n} \right\rfloor \cdot 2n + 2k + 1 & if\ (y \bmod 2n) < 2k+1 \\ \left\lceil \frac{y}{2n} \right\rceil \cdot 2n + 2k + 1 & otherwise\end{cases}$$

For example label for "title"(*1.9*) tag is calculated as follows:

$y=7$, n=*3* and $k=1$ so $x=\left\lfloor \frac{7}{6} \right\rfloor \cdot 6+2+1 = 9$, then node label of parent("*1*") will be attached to this number and finally label *1.9* is generated. Fig. 4 shows the FST which is built based on our method. To add a new tag, for example "author", between "author"(1.7) and "title"(1.9) we easily create an intermediate node with the even number between 7 and 9 (we may have more than one even number to choose in some situations) then we add the new node as a child of the intermediate node. This insertion couldn't be done on Extended Dewey because there is no integer number between 3 and 4. Fig. 5 illustrates changes after adding a new "author" node.

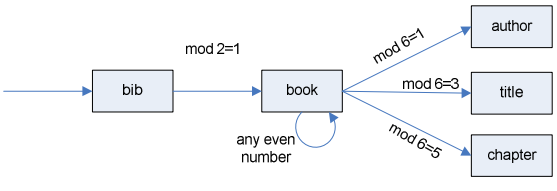

Fig. 4 IFDewey FST for Sample DTD

As it can be seen in Fig. 4, a new loop transition is added into "book" state which means if we reach an even number no new state transition is needed. Root-to-Node tags for "author"(1.8.1) are found as follows:
"*bib*" tag is common for all nodes, so we go to "*bib*" state .In "*bib*" state We calculate 1 mod 2 (2 instead of 1, because we use 2n instead of n). The result is 1 so we go to "book" state .in "book" state we omit even number 8 (even numbers are only used for intermediate nodes and will not be used in calculation) then we calculate 1 mod 6. It's 1 so we go to "author" state. Finally, Root-to-Node tags "*bib, book, author*" are generated.

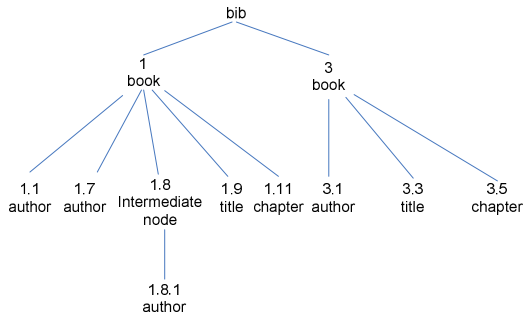

Fig. 5 XML tree after inserting new "author" node

IV. Conclusion

We proposed a new labeling method to resolve the insertion problem in Extended Dewey encoding. In our approach new nodes can be easily added to the XML tree and there is no need for modification of other node labels. A new finite transducer is also designed to translate insert friendly labels into Root-to-Node tags. TJfast algorithm can also use this new labeling method without any modification. IFDewey has the same code length problem as the Ordpaths and Extended Dewey methods. We plan to tackle this problem and investigate the efficiency of our method in our future work.

References

[1] J. Lu, T. Wang Ling, C. Chan, T. Chen : From Region Encoding To Extended Dewey: On Efficient Processing of XML Twig Pattern Matching. VLDB (2005) 193-204.
[2] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, , N. Westbury: ORDPATHs: Insert-Friendly XML Node Labels. SIGMOD (2004) 903-908.
[3] S. Tatarinov, K.S. Viglas., J. Beyer, E. Shanmugasun-daram, J. Shekita, C. Zhang, : Storing and querying ordered XML using a relational database system. SIGMOD (2002) 204-215.
[4] N. Bruno, D. Srivastava, N. Koudas : Holistic twig joins: optimal XML pattern matching. SIGMOD (2002) 310-321.
[5] S. Al-Khalifa, H.V. Jagadish, N. Koudas, J.M. Patel: Structural Joins: A Primitive for Efficient XML Query Pattern Matching. ICDE (2002) 141-152
[6] H. Jiang, W. Wang, H. Lu, J.X. Yu: Holistic Twig Joins on Indexed XML Documents. VLDB (2003) 273-284.
[7] Q. Li, B. Moon: Indexing and querying XML data for regular path expressions. VLDB (2001) 361-370.
[8] X. Wu, M. Lee, W. Hsu: A prime number labeling scheme for dynamic ordered XML trees. ICDE (2004) 66-78.
[9] M. Emadi, M. Rahgozar, A. Ardalan, A. Kazerani, M.M .Arian: A Comparative Study of DTD-Independent XML Data Storage Approaches. 11th International CSI Computer Conference, Tehran, Iran, CSICC (2006) 624-628.