A Fast XPATH Evaluation Technique with the Facility of Updates

Ashish Virmani
IIIT Allahabad
+919335131912

Suchit Agarwal
IIIT Allahabad
+919839412837

Rahul Thathoo
IIIT Allahabad
+919935256717

Shekhar Suman
IIIT Allahabad
+915322552376(8161)

Sudip Sanyal
IIIT Allahabad
919415235180

{virmani,suchit,thathoo,shekhar,ssanyal}@iiita.ac.in

ABSTRACT

This paper addresses the problem of fast retrieval of data from XML documents by providing a labeling schema that can easily handle simple as well as complex XPATH queries and also provide for updates without the need for the entire document being re-indexed in the RDBMS. We introduce a new labeling schema called the "Z-Label" for efficiently processing XPATH queries involving child and descendant axes.The use of "Z-Label" coupled with the indexing schema provides for smooth updates in the XML document.

Categories and Subject Descriptors

H.3.1 Content Analysis and Indexing – Indexing Methods.

General Terms: Algorithms, Performance.

Keywords: XML, XPath Query Optimization, Updates; Dewey Indexing; biaxes path expression

1. INTRODUCTION

In this paper, we have addressed the challenges in the use of relational engines for storing XML documents and providing efficient query answering.

2. DEFINITIONS

Definition 2.1: A *source path* [2], of a node n in an XML tree T, denoted as S(n) is the unique simple path, P (along the tree) from the root to itself.

Definition 2.2: A *biaxes path expression* is one that contains the tag names separated only by child axis steps (/) and descendant axis steps (//) e.g. /Season//League/Player

Definition 2.3: The set of nodes satisfying a query Q is represented as [[Q]] [2].

Definition 2.4: The Z-Label, S_n, of node n of the XML tree is the Z-Label of the biaxes path expression S(n) where S(n) is the source path of node n.

The Z-Label for any two nodes with same source path S(n) is the same. By definition 2.3, it implies that for a query Q with Z-Label S_Z,

$[[Q]] = \{ n \mid S_n \; LIKE \; S_Z \}$

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

3. HOW TO ASSIGN Z-Label

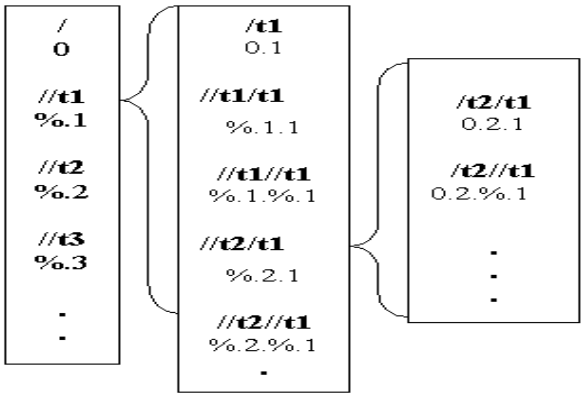

Figure 1

Step 1: We assign "//" the Z-Label "%"
Step 2: "/" is assigned the Z-Label "*0*"
Step 3: Each distinct tag of the XML tree is assigned a unique tagID which is an integer
Step 4: For each $//t_i$ we assign Z-Label "%.*i*".
Step 5: Each $//t_i$ $/t_j$, we assign Z-Label "%.i.j"
Step 6: Each $//t_i$ $//t_j$, we assign Z-Label "%.i.%.j"

The procedure is the same for any biaxes path expression that we might encounter.

Let us look at the way the containment between two biaxes path expressions P and Q can be determined with the help of z-label. The following cases arise:

- In the case when there is no "**.**" at the end of the z-label and there is no "**%**" in z-label of Q, then Q is contained in P if and only if $Z_P = Z_Q$.
- In the case when there is no "**.**" at the end of the z-label and there is no "**.%.**" in z-label of Q, then Q is contained in P if and only if $Z_P \; LIKE \; Z_Q$.
- In the case when there are *i* occurrences of "**.%.**" in the z-label of Q, then for each occurrence of "**.%.**" in the z-label, we *(a)* replace it with **null** and *(b)* keep it as it, generating a total of 2^i different possible z-labels for the biaxes path expression Q. Now, the biaxes path expression Q is contained in P if and only if: $\Omega_{1<j<2}i$ ($\mathbf{Z_P}$ **LIKE** $\mathbf{Z_{Qj}}$) holds where Ω concatenates all such expressions of the form "Z_P LIKE Z_{Qj}" with an "**OR**" between them.

- In the case when there is a "." at the end of the z-label of Q, P is contained in Q, if and only if **Z_P LIKE CONCAT(Z_Q, %) AND Z_P NOT LIKE CONCAT(Z_Q, %.%).**

where **LIKE, NOT LIKE** and **CONCAT** operators perform the same function as in SQL.

The advantage of Z-Label while handling updates is twofold:

1. Whenever a new tag is added to the tree, the tag is assigned a unique tagID and the Z-Label of the new node with the given tag is assigned on the basis of the source path of the node.
2. Whenever a new node of an existing tag is added to the tree, the Z-Label is assigned on the basis of its source path.

4. THE INDEXING SCHEMA

The dynamic indexing schema borrows its basic idea from Bohme et. al.[1]. The indexing schema is able to handle insertion and deletion of nodes as well as supports efficient querying.

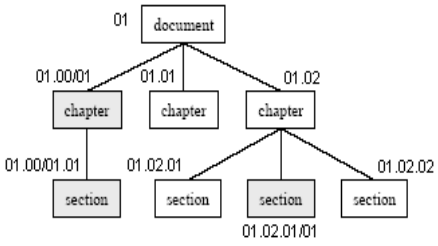

Figure 2

In Figure 2, nodes *01.00/01, 01.00/01.01* and *01.02.01/01* could be inserted without renumbering the existing nodes. As the character '/' has a greater ASCII value that the character '.', it is always ensured that the index given to a new node is greater than that of the node on its left and smaller than that of the node on its right. Also the index of the new node is greater than each node of the sub-tree whose root is the left node. So, this indexing always preserves document order of an XML document.

5. QUERY BREAKING ALGORITHM

The algorithm consists of predicate elimination from an XPath query. The basic steps of predicate elimination are to take a XPath query as an input string and traverse the query. Whenever a predicate occurs, the query is broken into parts. For e.g. a query of the form /a[b]/c is broken into three queries /a, /a/b, /a/c as shown in figure 6. Taking joins between these biaxes-path queries can generate the final SQL query.

The query breaking algorithm could not be shown due to space constraints.

6. DATA SETS & EXPERIMENTAL RESULTS

Table 1: XML Data Sets

	Shakespeare	Protein	Auction
Size	1.3MB	3.5MB	3.4MB
Nodes	31975	113831	61890
Tags	19	66	77
Depth	7	7	12

Protein

Q1 /proteindatabase/proteinentry/protein/name

Q2 /proteindatabase/proteinentry//authors/author

Q3 /proteindatabase/proteinentry[reference/refinfo[citation = "j. biol. chem." and year = "1977"]]/protein/name

Shakespeare

Q1 /plays/play/act/scene/speech/line

Q2 /plays/play/epilogue//line/stagedir

Q3 /plays/play/act/scene[title ="scene iv. the platform."]//line

Auction

Q1 //category/description/parlist/listitem

Q2 /site/regions//item/description

Q3 /site/people/person[profile/income < 10000]

Figure 3

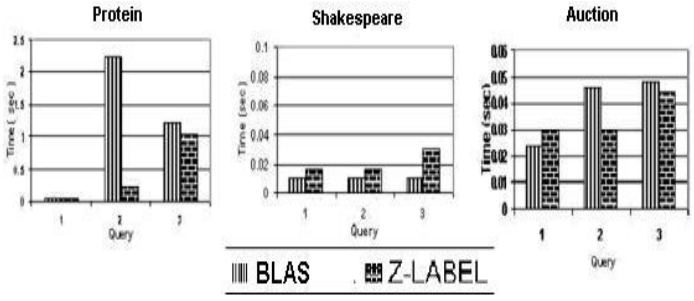

Figure 4

7. CONCLUSION

We have presented the *"Z-Label",* a labeling schema that is used to facilitate the fast retrieval of information from an XML document through XPath queries. The indexing schema that has been proposed also provides for updates in the XML document and makes it possible to insert or delete a node from the document without the need for all the nodes to be re-indexed. This offers a major advantage over previous indexing schemas that re-compute the indexes for all the nodes whenever a new node is inserted in the XML document.

8. REFERENCES

[1] Böhme, T.; Rahm, E.: Supporting Efficient Streaming and Insertion of XML Data in RDBMS. Proc. 3rd Int. Workshop Data Integration over the Web (DIWeb), 2004

[2] Yi Chen, Susan Davidson, and Yifeng Zheng. BLAS: An Efficient XPath Processing System,Proceedings of SIGMOD 2004.