ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

XML Path Algorithm for Matching XML Elements with Different Queries

K.Karthiban, M.Arul Jothi, G S Magthalin Nisha, V.Sindhu

Asst.Professor, Department of Information Technology, Karpagam College of Engineering, Coimbatore, Tamil Nadu, India.

ABSTRACT: The vital operation in xml query is searching for the elements of a tree pattern query in XML database. Earlier work shows that holistic rooted labeled tree pattern matching algorithm is an efficient technique to answer an XML tree pattern with parent-child (P-C) and ancestor-descendant (A-D) relationships, as it can effectively control the size of midway results during query processing. On the other side, XML query languages (e.g., XPath and Xquery) define more axes and functions such as negation function, order-based axis, and wildcards. The previous research a large set of XML tree pattern, that contains P-C, A-D relations, negation functions, wildcards, and/or expressions, order restrictions and all other axes that are defined in the xpath and xquery languages. XML Path algorithm is the algorithm proposed theoretically. This method is predicted to increase the algorithmic efficiency and to speed up the query. This method avoids generating large midway results which do not contribute to the final outcome. This technique reduces both input I/O cost and redundant midway result sizes and achieve good performance.

KEYWORDS: Query processing, XML/XSL/RDF, XPATH, TREE

I. INTRODUCTION

In today's world the transfer of data in web is increasing due to the mushrooming of industrial and business enterprises. These industrial and business enterprises generate data in XML format or can say like they transfer XML data more and more as their business need increases. Therefore it is inevitable to process the queries on the XML data. It is to be noted that the processing of the XML queries must be effective and efficient. The most indispensable and important process involved in XML document is searching for the occurrence of twig pattern in an XML document as quick as possible. In order to facilitate the sharing of data across different information systems over the internet, a new open standard for web data representation and exchange, XML (extensible Markup Language), has been suggested and adopted as the new generation web data format. XML started strong and has grown quite rapidly. It has proven itself a very valuable technology and allows data integration on the web. In fact, most data exchanged among a variety of web applications are already in XML format, such as web services that use XML-based descriptions in WSDL and exchange XML messages based on the SOAP protocol, e-commerce and e-business, collaborative authoring of large electronic documents and management of large-scale network directories. An XML query pattern commonly can be represented as a rooted, labeled tree (or called twig). XML stands for extensible Markup Language, which is a markup language for documents containing structured information. It is designed to meet the challenges of large scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. The increasing popularity of XML is partly due to the limitations of the other two technologies: Hypertext Markup Language (HTML) and Standard Generalized Markup Language (SGML, ISO 8879) for representing structured and semi-structured documents. HTML provides a fixed set of tags; these tags are mainly for presentation purposes and do not bear useful semantics while SGML is too difficult to implement for most applications because of its complex specifications. XML lies somewhere between HTML and SGML and is a simple yet flexible format derived from SGML. An XML document always starts with a prolog markup. The minimal prolog contains a declaration that identifies the document as an XML document. XML identifies data using tags, which are identifiers enclosed in angle brackets. Collectively, the tags are known as "markup". For example, Figure 1.1 shows a simple example of XML document. This document starts with a prolog markup that identifies the document as an XML document that conforms to version 1.0 of the XML specification and uses the 8-bit Unicode character encoding scheme (line 1). The root element (line 2-14) of the document follows the declaration, which is named as company element. Each XML document has a single root element. Next, there is an element (line 3-13) which describes the information employee details with age, sex, designation and the turnover of the company. In line 9, the element turnover contains a sub-element year id and numeric data.

<? xml version="1.0"?>
<company>

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

```
<employee id="001" sex="M" age="26" nam="Sreejith Pillai">MANAGER
</employee>
<employee id="002" sex="M" age="32" nam="Robin Singh">ACCNT
</employee>
<employee id="003" sex="M" age="31" nam="Shekar Kapoor">JER
</employee>
<turnover>
<year id="2000">200,000</year>
<year id="2001">240,000</year>
<year id="2002">400,000</year>
</turnover>
</company>
```

Fig: 1.1 Example XML Document

II. RELATED WORK

Many authors have worked on the xml based data engineering domain and found the holistic ways of processing the xml query languages like xquery and xpath. Some of the works are as follows.

2.1 COMPREHENSIVE XML TREE PATTERN

Previous algorithms proposed by different authors focus on XML tree pattern queries with only P-C and A-D relationships. Slight work has been done on XML tree queries which may contain wildcards, negation function, and order restriction, all of which are frequently used in XML query languages such as XPath and Xquery in extended xml tree pattern matching[1]. In this paper, we play with an XML tree pattern with negation function, wildcards, and / or order restriction like extended XML tree pattern. Fig. 2, for example, shows four extended XML tree patterns proposed in [1]. Query (a) includes a wildcard node "*",match any single node in an XML database. Query (b) includes a negative edge, denoted by " ¬ ". This query finds A that has a child B, but has no child C. In XPath language [2], the meaning of negative edge can obtainable with "not" Boolean function. Query (c) has the order restriction, which is equivalent to an XPath "//A/B[following-sibling::C]." The "< " in a box shows that all children under A are ordered. The meaning of order-based tree pattern is captured by a mapping from the pattern nodes to nodes in an XML database such that the structural and ordered relationships are satisfied. Finally, Query (d) is more intricate and knotty, which contains wildcards, negation function, and order restriction and (A-D), (P-C) relationship.

2.2 MATCH ELEMENTS IN XML TREE MODEL

Matching xml query in an xml db is an important function in xml processing. Given an extended tree query Q with n selected return nodes and an XML database D, a match of Q in D is identified by a mapping from nodes in Q to the elements in D, such that:

1. Query node types (i.e., tag name) are satisfied by the corresponding database elements and wildcards "*" can match any single database element;
2. The positive edge relationships (including positive parent-child and positives ancestor-descendant edges) between query nodes are satisfied by the corresponding database elements;
3. The negative edge relationships (including negative parent-child and negative ancestor-descendant edges) are satisfied, that is, no corresponding database element pairs exist; and
4. The order relationship of children of each ordered node is satisfied by the corresponding database elements.

Some of the other algorithms which was devised before by several authors are TwigStackList, orderedTJ, iTwigJoin, TJFast, Twig2Stack

Some of the basic properties of the algorithm [1] are as follows. The elements in the label list are ordered as per the order of the elements in the xml document. Second property is the memory requirement. That is the memory requirement is linear to the number of nodes.

2.3 TWIG

XML data is modeled as labeled and ordered tree or graph. A twig is a rooted, labeled tree. Naturally twig (a small tree) pattern becomes an essential part of many XML queries. A twig pattern can be represented as a node-labeled tree whose edges are either Parent-Child (P-C) or Ancestor-Descendant (A-D) relationship. For example, the following twig pattern written in XPath [6] format:

section [/title]/paragraph//figure..... (Q1)

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

The above query selects the figure element. First it selects the title under the section. That is the query selects the title element which should be the child of section and then select the paragraph node which is again child element of title. Finally it selects the descendant element figure which is not necessarily a child but has ancestor descendant relationship.

2.4 TWIG STACK

Twig Stack is i/o and CPU optimal among all sequential algorithms that read the entire input: it is linear in the sum of the sizes of the input list and the final result list but independent of the sizes of the midway results. The holistic TwigStack proposed by Bruno etal is CPU and I/O optimal for all path patterns and A-D only twig patterns. It associates each node q in the twig patterns with a stack Sq and stream Tq containing all elements of tag q. Each stream has an imaginary cursor which can either move to the next element or read the element under it. The algorithm operates in two main phases: Twig join and Merge

The advantages in this holistic algorithm are as follows:

- Scan only once
- No redundant output
- Bounded space complexity

The space complexity of the algorithm is bounded by its longest path in the source xml document. The major reason for the twigstack algorithm's inefficiency is that does not concentrate on the level information. Only focus on assuming all the edges as A-D relationship in the initial step.

2.5 TWIG STACK LIST

This algorithm just enlarges the optimal query of the algorithm Twig Stack by including P-C relationships in non branching edges. Hence, the current open problems include 1) how to identify a larger query class which can be processed optimally and 2) how to efficiently answer a query which cannot be guaranteed to process optimally.

Bruno et al. [2] proposes a novel holistic XML path and twig pattern matching method TwigStack which avoids storing midway results unless they contribute to the final results. The method, unlike the decomposition based method, avoids computing large redundant midway results. The method is CPU and I/O optimal for all paths (with no branch) patterns and twig patterns whose edges are entirely ancestor-descendant edges. Meanwhile the space complexity of their algorithm is bounded by the longest path in the source XML document. However the approach is found to be suboptimal if there are parent-child relationships in twig patterns. That is, the method may still generate redundant midway results in the presence of P-C relationships in twig patterns. Bruno et al. [2] proposes a novel holistic XML path and twig pattern matching method TwigStack which avoids storing midway results unless they contribute to the final results. The method, unlike the decomposition based method, avoids computing large redundant midway results. The method is CPU and I/O optimal for all path (with no branch) patterns and twig patterns whose edges are entirely ancestor-descendant edges. Meanwhile the space complexity of their algorithm is bounded by the longest path in the source XML document. However the approach is found to be suboptimal if there are parent-child relationships in twig patterns. That is, the method may still generate redundant midway results in the presence of P-C relationships in twig patterns.

III. PROPOSED TECHNIQUE

3.1 XML PATH

The proposed approach is based on theoretical analysis only. This theoretical analysis leads to propose a technique called xml path algorithm by modifying the existing algorithm treematch. This technique's work is theoretically predicted to increase the query speed and to support for more xquery patterns and axes while traversing xml document. The core function is to use xpath index technique which partition xml document. Xpath index technique gives the optimal processing of other classes of twig pattern (other than the twig patterns with A-D relationships). In observation of the terminology used in the original holistic pattern, the combination of XML indexing methods with containment labels as XML streaming schemes is used. This method reduces the amount of input I/O cost and reduces the size of redundant midway results. Mainly this increases the speed of the searching technique. This method increases the amount of holism.

3.2 ALGORITHM:

Findmatchlabel (L)
loop:
while not end of the root
a=fetch next node;
if a contains branching node

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

fetch next branching node
if (node==leaf)
index up to this traverse level
proceed to next branching node
add node to update set
else
return the node
Findmatchlabel(L)

The above is the modification of the algorithm tree match. Indexing is been added to this algorithm. This can be enhanced to support the various patterns. Indexing technique is been done to speed up the search. This algorithm is a theoretical proposal for the better utilization of the memory. This efficiently fetches the node that are given in the xpath or xquery. The loop begins by checking whether the cursor is on root or end of the root. If it is on the end of the root then it will not process. Else it will process ad fetch the next branching node and process till it is a leaf node. Then indexing is done with respect to that particular level or xml.

3.3 SYSTEM ARCHITECTURE

The diagram below shows the system architecture of the system. This gives an outline view of fetching nodes from xml file or xml database. The nodes are fetched using xpath that process using index algorithm which supports all xpath patterns and axis. This algorithm is very effective in processing xpath and xquery. This avoids midway results which are useless in finding the nodes that are searched. This indexing method partitions the xml documents that are necessary in tracing the elements and nodes.

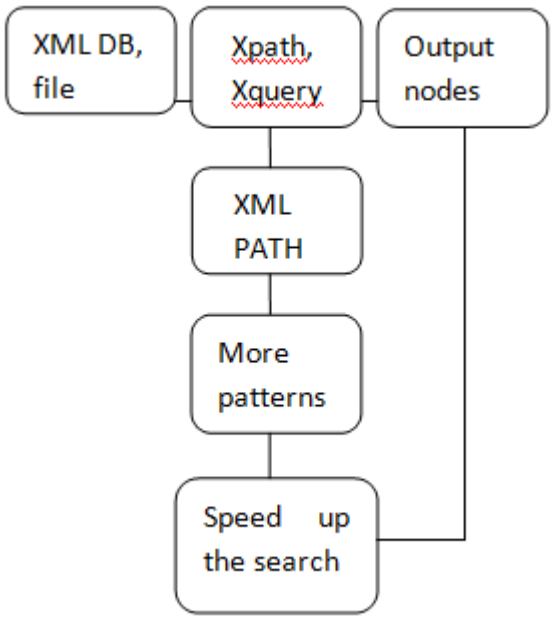

Fig.1 SYSTEM ARCHITECTURE

IV. BACKGROUND WORKS

4.1 GRAPH MODEL

The below is a graphical model that shows the plotting of query versus time. The index method speeds up the query and that is why the model is analysed and the graphical diagram is proposed here. Queries 1,2,3,4 are given in x axis. Query 1 and 2 are a normal xpath query that use usual coding technique whereas the query 3 and 4 use partial index and complete xpath index algorithm. Incase if it is implemented the output graph obtain must be like the above.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

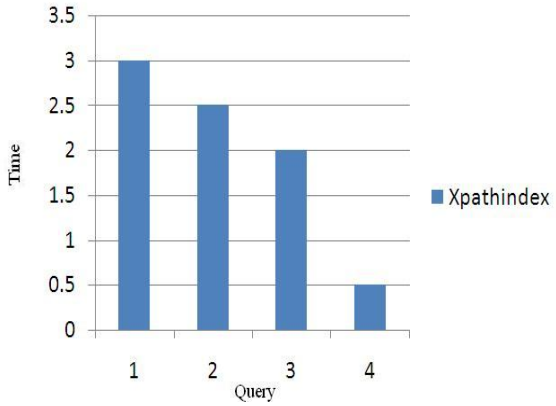

Fig.2 Graph - Time Vs Query

4.2 IMPLEMENTING TECHNIQUE

The nodes and elements are fetched using xpath. A sample xml is taken as xml db and it is traversed using the various algorithms. Some basic queries are tested. This is done using Microsoft Visual studio framework in C# language using ASP.NET

V. CONCLUSION

The proposed algorithm will work for all types of patterns that are in Xpath and Xquery. The theoretical proposal of the index path algorithm will speed up the queries in fetching nodes in the large xml database or file. In future this algorithm can be worked, implemented and enhanced for better tuning of processing queries related to xml.

REFERENCES

[1] J. Lu, T.W. Ling, Z. Bao, and C. Wang, "Extended XML Tree Pattern Matching: Theories and Algorithms," IEEE Transactions on Knowledge and Data Engineering, March 2011.
[2] A. Berglund, S. Boag, and D. Chamberlin, XML Path Language (XPath) 2.0, W3C Recommendation, http://www.w3.org/TR/xpath20/, Jan. 2007.
[3] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: optimal XML pattern matching. In Proceedings of SIGMOD 2002, pages 310–321, 2002.
[4] H. Jiang et al., "Holistic Twig Joins on Indexed XML Documents,"Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 273-284, 2003.
[5] T. Chen, J. Lu, and T.W. Ling, "On Boosting Holism in XML Twig Pattern Matching Using Structural Indexing Techniques," Proc.ACM SIGMOD [6] XPath. http://www.w3.org/TR/xpath.