A Survey Of Algorithms Related To Xml Based Pattern Matching

Dr.R.Sivarama Prasad [1], D.Bujji Babu[2], Sk.Habeeb[3], Sd.Jasmin[4]

[1] *Coordinator ,International Business Studies , Acharya Nagarjuna University,Guntur,A.P,India,*
[2] *Associate Professor,Prakasam Engineering College, Kandukur,A.P, India ,*
[3] *Assistant Professor, ,Nimra Institute of Engineering and Technology, Ongole,A.P, India ,*
[4] *CSE Dept ,Prakasam Engineering College,Kandukur, A.P, India,*

Abstract

The New era practitioners are highly depending on the flexible open standard data structures to store and transmit the data in the B2B process, as a part of that, eXtended Markup Language(XML) Patterns are needed for efficient processing of user queries on different XML-enabled (MS-SQL Server, Oracle) Databases. The XML Document can be converted into XML Tree by using different tools like XML DOM. There are four central problems in data management: capture, storage, retrieval, and exchange of data. XML is a tool used for data exchange. Data exchange has been a long issue in information technology, but the Internet has elevated its importance. Electronic Data Interchange (EDI), the traditional data exchange standard for large organizations, is giving way to XML, which is likely to become the data exchange standard for all organizations, irrespective of size. Most of the business and enterprises generate and exchange XML data more often, so there is need for efficient processing of queries on XML data. An XML query pattern can be represented as rooted labeled tree called as twig pattern. The core operation in XML query processing is efficient matching of XML tree patterns. The main reason why this one is introduced here is , it process large XML queries. The existing algorithms process only small queries it includes only P-C and A-D relations. It process large queries with P-C, A-D, wildcards, negation function and order restriction called extended XML tree pattern.

In this paper, we present the survey of the recent works and algorithms related to eXtended Markup Language based Small Patterns Matching. We present an analysis of the Small Patterns Matching algorithms likes Twigstack, Twigstack List, ordered TJ and TJFast. Tree Match Twig pattern Matching algorithm can solve complicated queries and works with good performance.

Keywords: *B2B, XML Tree Pattern, XML, database.*

---***---

1. INTRODUCTION

As the Trend change, the conventional business is transmitting into the form of e-business through the Internet, which is known as B2B (Business-to-Business). It is very clear for us the B2B problem are all website based, same of the B2B websites are company websites, product supply and procurement exchanges, specialized or vertical industry portal, Information sites etc. For any type of e-business the websites is essential in this context the web information is presented in xml format and XML Document contain all the XML components of websites and the determination of XML Patterns are needed to solve the problems in B2B sites and to get more performance.

In this paper we present the core concepts of Query languages. XML Trees can be Ordered and Unordered Trees. The present XML Trees can understand with a good labeling schemes[5].XPath[1], XQuery[2] are different query languages for XML. We also present the experimental results for pattern search with using the XMLBuilder and SAX parser in Java.

2. OBJECTIVE

The basic idea is searching. It means searching for the occurrence of a tree pattern query in an XML database is a core operation in XML query processing. Previous algorithms search only the small tree patterns in the XML databases. Here the idea for this one is to search large tree patterns and to get optimal answers. It uses matching cross that's a theoretical framework that demonstrates the intrinsic reason in the proof of the optimality of holistic algorithms. The reason behind why we introduce matching cross is the previous algorithm's performs their work in two phases means matching and merging, at matching phase there is a chance to occur many useless intermediate results (path solutions).

.

3. ALGORITHMS

An algorithm is a step by step procedure to solve a particular problem. The following are the algorithms for pattern matching.

3.1 Algorithm: TreeMatch

Input: The query Q

Output: The tree pattern called as twig pattern

```
1. locateMatchLabel (Q);
2. while(notEnd (root)) do
3. $f_{act}$= getNext (topBranchingNode);
4. if ($f_{act}$ is a rturn node) then
5. addToOutputList (NearestAncestorBraching ($f_{act}$),cur ($T_{fact}$));
6. advance ($T_{fact}$); // read the next element in $T_{fact}$
7. updateSet ($f_{act}$ ); // update set encoding
8. locateMatchLabel (Q); // locate next element with matching path
9. emptyAllSets (root);
```

The following are the procedures and functions are used in the TreeMatch algorithm:

Procedure locateMatchLabel(Q)

```
1. for each leaf q∈Q, locate the extended Dewey label eq in list Tq such that eq matches the individual rootleaf path
```

Procedure addToOutputList(q,e_{qi})

```
1.for each $e_q$ε$S_q$ do
2.if(satisfyTreePattern($e_{qi,}$ $e_q$)) then
3.outputList($e_q$); add($e_{qi}$);
```

Function satisfyTreePattern(e_{qi} , e_q)

```
1.if(bitVector($e_q$ , $q_i$ )==’1’) then return true;
2.else return false;
```

Procedure updateSet(q,e)

```
1.cleanSet(q,e)
2.add e to set $S_q$; //set the proper bitVector(e)
3.if(isNotRoot(q) and bitVector(e)==”1…1”)) then updateAncestorSet(q);
```

Procedure cleanSet(q,e)

```
1.for each element $e_q$∈$S_q$ do
2.if(satisfyTreePattern($e_q$,e)) then
3.if(q is a return node) then
4.addToOutputList(NearestAncestorBranching(q),e);
5.if(isTopBranching(q)) then
6.if(there is only one element in $S_q$) then
7. output all elements in outputList ($e_q$);
8. else merge all elements in outputList ($e_q$) to
   outputList ($e_a$),where $e_a$=NearestAncestorBranching(e);
9. delete $e_q$ from set $S_q$ ;
```

Procedure updateAncestorSet(q)

```
1. /*assume that q' = NearestAncestorBranching (q)*/
2. for each e∈ Sq' do
3. if (bitVector (e, q) = 0) then
4. bitVector (e, q) = 1;
5. if (isNotRoot (q ) and (bitVector (e )=“ 1.. .1” )) then
6. updateAncestorSet (q');
```

Procedure emptyAllSets(q)

```
1. if (q is non-leaf node) then
2. for each child c of q do emptyAllSets (c);
3. for each element e ∈ Sq do cleanSet (q, e);
```

3.2 Algorithm for getNext(n):-

```
1. if (isLeaf (n )) then
2. return n
3.else
4. for each ni ∈ NearestDescendantBranching(n) do
5. $f_i$ = getNext (n)
6. if (isBranching($n_i$) and notEmpty ($S_{ni}$ ) ) then
7. return $f_i$
8. else ei = max {p | p∈ MB ($n_i$, n)}
9.end for
10. max = $maxarg_i\{e_i\}$
11. for each n ∈NearestDescendantBranching (n) do
12. if (ForAll (e) ∈ MB ($n_i$, n) : e is not belongs to Ancestors ($e_{max}$)) then
13. return $f_i$;
14. end if
15.end for
16. min = $minarg_i$ { $f_i$ | $f_i$ is not a return node}
```

```
17. for each e $\in$ MB ($n_{min}$, n)
18. if (e$\in$ ancestors ($e_{max}$)) then updateSet ($S_n$,e)
19.end for
20. return $f_{min}$
21.end if
```

Function MB(n,b)

```
1. if (isBranching(n)) then
2. Let e be the maximal element in set $S_n$
3.else
4. Let e = cur ($T_n$)
5.end if
6.Return a set of elements **a** that is an ancestor of **e** such that **a** can match node **b** in the path solution of **e** to path pattern $p_n$.
```

4. EXPERIMENTAL RESULTS

Our experiment uses a concise encoding to present matching results, which leads to the reduction of useless intermediate results. We have an extensive set of experiment on synthetic and real data set for performance comparison. We compared TreeMatch with previous holistic XML tree pattern algorithms. The experimental results show that our algorithm can correctly process XML tree patterns called twigs. It achieves performance speedup. This is because of reduction of useless intermediate results during XML query processing.

Table 4.1: Number of output elements(O) and the percentage(P) of useful elements for TreeMatch algorithm

Query	D1		D2		D3	
	O	P	O	P	O	P
Q1	660	100%	3276	100%	6702	100%
Q2	1775	100%	8757	100%	17325	100%
Q3	4575	98.8%	47290	99.9%	80145	94.5%
Q4	3335	100%	16054	100%	32012	100%
Q5	148	100%	606	100%	1302	100%
Q6	3275	100%	47100	100%	67478	99.9%

Table 4.2: Number of required buffered elements

Query	D1	D2	D3
Q1	5	6	6
Q2	9	10	11
Q3	260	13030	45001
Q4	6	7	8
Q5	7	8	10
Q6	260	13304	44989

The purpose of this is to simulate the application where the available main memory is large so that a big portion of documents can fit in the main memory. Table 4.2 shows the maximal number of elements buffered in order to avoid outputting any useless intermediate results. An obvious observation is that Q3 and Q6 (Queries) need to buffer many elements, but all other queries only need to buffer very small number of elements. This can also be explained that all queries except Q3, Q6 belong to the optimal query class. We compared the performance of three algorithms (D1,D2 and D3) .From this one TreeMatch is better than the other algorithms because it reaching 20%-90% improvement in execution time for all queries.

4.1 SCREENS

The following screen allows to select XML file for viewing XML Tree

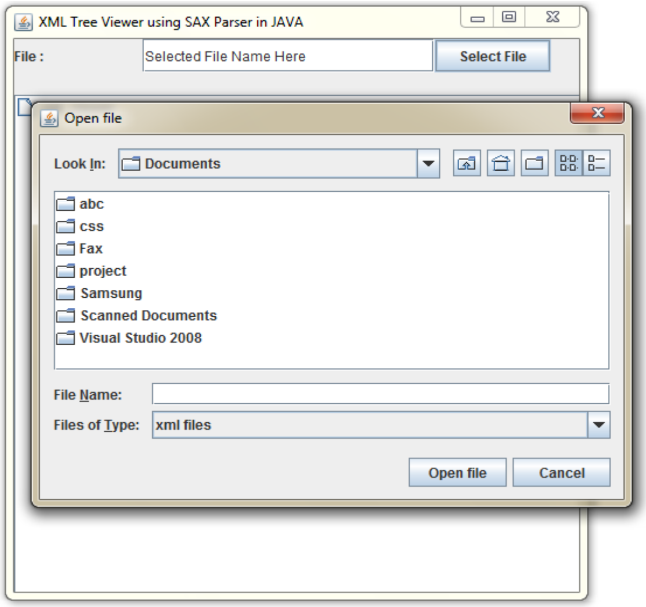
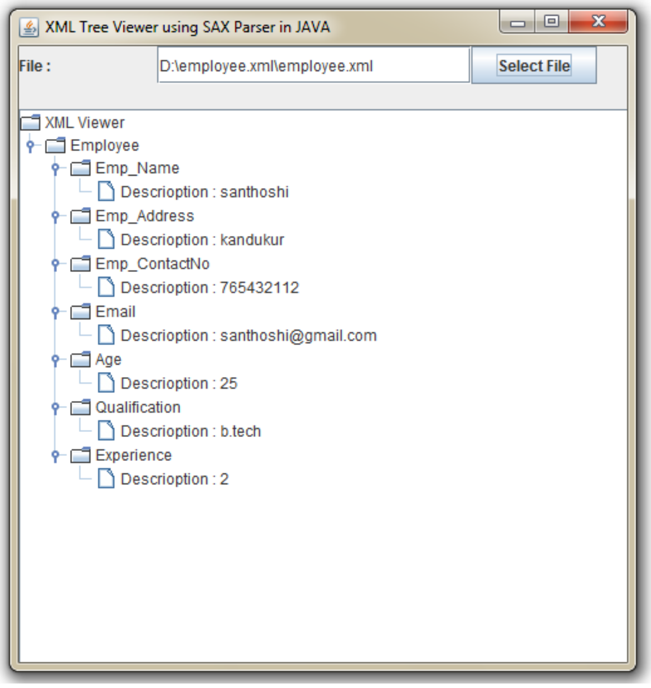


Fig:1:XML file selection

After selecting the location of an XML file the XML Tree will appears as follows



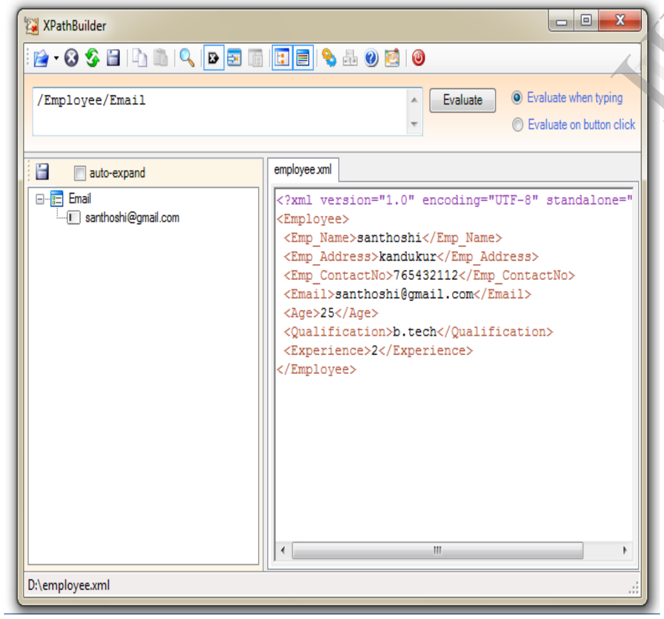
XML Tree Viewer using SAX Parser in JAVA

File : D:\employee.xml\employee.xml [Select File]

```
XML Viewer
  Employee
    Emp_Name
      Descrioption : santhoshi
    Emp_Address
      Descrioption : kandukur
    Emp_ContactNo
      Descrioption : 765432112
    Email
      Descrioption : santhoshi@gmail.com
    Age
      Descrioption : 25
    Qualification
      Descrioption : b.tech
    Experience
      Descrioption : 2
```

Fig:2:XML Tree

Small XML query patterns can be search in the XML as follows



```
/Employee/Email
```

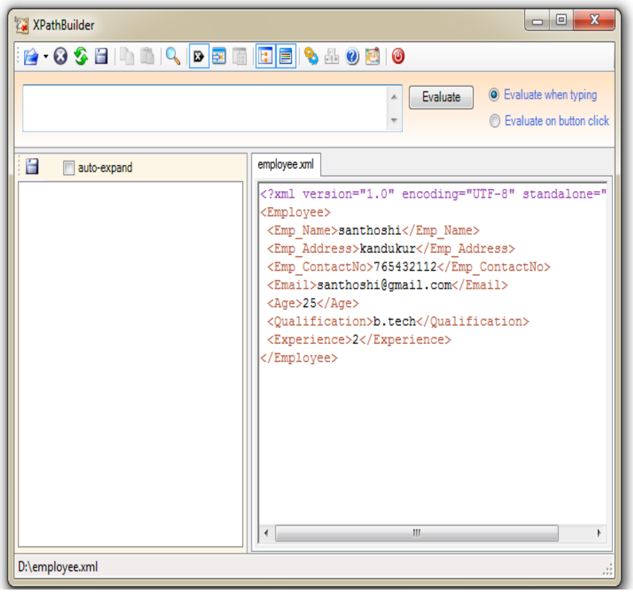
```
Email
  santhoshi@gmail.com
```

```
<?xml version="1.0" encoding="UTF-8" standalone="
<Employee>
  <Emp_Name>santhoshi</Emp_Name>
  <Emp_Address>kandukur</Emp_Address>
  <Emp_ContactNo>765432112</Emp_ContactNo>
  <Email>santhoshi@gmail.com</Email>
  <Age>25</Age>
  <Qualification>b.tech</Qualification>
  <Experience>2</Experience>
</Employee>
```

D:\employee.xml

Fig:3:Searching small patterns in XML

The following is one of the application to search the XML queries for XML tree pattern.



```
<?xml version="1.0" encoding="UTF-8" standalone="
<Employee>
  <Emp_Name>santhoshi</Emp_Name>
  <Emp_Address>kandukur</Emp_Address>
  <Emp_ContactNo>765432112</Emp_ContactNo>
  <Email>santhoshi@gmail.com</Email>
  <Age>25</Age>
  <Qualification>b.tech</Qualification>
  <Experience>2</Experience>
</Employee>
```

D:\employee.xml

Fig:4:XPath Builder for searching

5. CONCLUSION

In this paper, we presented the survey results of XML based pattern matching algorithms . The TreeMatch algorithms with different query classes are introduced. TreeMatch has an overall good performance in terms of running time and the ability to process extended XML tree patterns (twigs). The previous twig pattern matching algorithms (*TwigStack*, *TwigStackList*, *OrderedTJ*, *and TJFast*) requires bounded main memory for small queries. But, the *TreeMatch* works on one-phase query evaluation and it requires bounded main memory for larger queries.

REFERENCES

[1]. C. Zhang, J. F. Naughton, D. J. DeWitt, Q. Luo, and G. M. Lohman.On supporting containment queries in relational database management systems. In Proc. of SIGMOD Conference, pages 425–436, 2001.
[2]. I. Tatarinov, S. Viglas, K. S. Beyer, J. Shanmugasundaram, E. J. Shekita,and C. Zhang:. Storing and querying ordered XML using a relational database system. In Proc. of SIGMOD, pages 204–215, 2002.
[3]. J. Lu, T. W. Ling, T. Yu, C. Li, and W. Ni. Efficient processing of ordered XML twig pattern matching. In DEXA, pages 300–309, 2005.
[4]. J. Lu, T. W. Ling, C. Chan, and T. Chen. From region encoding to extended dewey: On efficient processing of xml twig pattern matching. In VLDB, pages 193–204, 2005.

[5]. S. Al-Khalifa, H. V. Jagadish, J. M. Patel, Y. Wu, N. Koudas, and D. Srivastava. Structural joins: A primitive for efficient XML query pattern matching. In Proc. of ICDE Conference, pages 141–152, 2002.

[6]. N. Bruno, D. Srivastava, and N. Koudas. Holistic twig joins: optimal XML pattern matching. In Proc. of SIGMOD Conference, pages 310–321, 2002.

[7].S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, and K. S.Candan. Twig2stack: Bottom-up processing of generalized-tree-pattern queries over xml document. In Proc. of VLDB Conference, pages 19–30, 2006.

[8]. B. Choi, M. Mahoui, and D. Wood. On the optimality of the holistic twig join algorithms. In Proceeding of DEXA, pages 28–37, 2003.

[9]. H. Jiang et al. Holistic twig joins on indexed XML documents. In Proc.of VLDB, pages 273–284, 2003.

[10]. H. Jiang, H. Lu, and W. Wang. Efficient processing of XML twig queries with OR-predicates. In Proc. of SIGMOD Conference, pages 274–285, 2004.

[11]. J. Lu, T. Chen, and T. W. Ling. Efficient processing of xml twig patterns with parent child edges: a look-ahead approach. In CIKM, pages 533–542, 2004.

BIOGRAPHIES

Dr.R.Sivarama Prasad is currently working as a coordinator for International Business Studies at Acharya Nagarjuna University,Guntur,A.P.India.He is a interdisciplinary researcher in computer science and management Sciences.He Published a dozens of research papers. His interested area of research is software Engineering, Data Warehouse and Data Mining, e-commerce,Business Intelligence, Systems Architectures. Customer Relationship Management. He Authored seven books.

D.Bujji Babu is an Associate Professor in the department of Computer Science and Engineering at Prakasam Engineering College, Kandukur, A.P. India. He published many research papers in different referred journals. His interested area of research is software Engineering, Data Warehouse and Data Mining, Business Intelligence, Systems Architectures.

Mrs.Sk.Habeeb is currently working as an assistant professor in the department of Computer Science and Engineering at Nimra Institute of Engineering and Technology, Ongole, A.P, India.

Miss. Sd.Jasmin is currently working as an assistant professor in the department of Computer Science and Engineering at Prakasam Engineering College, Kandukur , A.P, India.