IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 7, July 2015.
www.ijiset.com

ISSN 2348 – 7968

Efficient Query Optimization Of XML Tree Pattern Matching By Using Holistic Approach

Miss. Bharti D. Wanjari[1] **Professor Mr. Kapil N. Hande** [2]

[1]RTMNU University, PBCE Nagpur, Maharashtra, India.

[2]RTMNU University, Department of Computer Science and Engineering, PBCE Nagpur, Maharashtra, India.

Abstract

With XML becoming pervasive language for data interoperability purposes in various domains, efficiently querying XML data is a vital issue. XML has become a practice standard to exchange, share and store business data across similar and dissimilar platforms. As enterprises are generating vast amount of data in XML format, there is a require for processing XML tree pattern queries. This paper presents some developments in the field of XML tree pattern query processing, especially focusing on holistic approaches. The obtainable holistic algorithms for XML tree pattern matching queries display suboptimmality problem as they consider intermediate consequences before taking final results. This causes suboptimal performance. This suboptimality is overcome by using TreeMatch algorithm. This paper implements a model application that makes use of dewey labeling scheme to beat suboptimality with TreeMatch algorithm. The experimental results discovered that the proposed algorithm is better than the existing algorithms.

Keywords: *XML, XML tree pattern query, query optimization and evaluation, dewey labeling, holistic algorithm.*

1. Intoduction

With the rapidly increasing popularity of XML for data representation, there is a lot of interest in query processing over data that conform to the *labeled-tree* data model. Due to the business alliance and for the purpose of adjustability organizations are storing data in XML format. This has become a common practice as XML is easily transported and irrespective of platforms in which applications were developed, they can share data through XML file format. Such XML files are also approved using DTD or Schema. XML parsers are available in all languages that ease the usage of XML programmatically. Besides XML is tree based and it is convenient to handle easily using Document Object Model(DOM) API. XML tree pattern queries are to be processed efficiently as that is the main operation of XML data.

freshly many researchers developed various methods or algorithms [4], [5], [6], [7], [8], [9] for processing XML tree queries. A stack based algorithm [3] was proposed by Khalifa et al. that matches relationships such as A-D, and P-C. TwigStack is another algorithm proposed by Bruno et al. [10] for the reason of XML tree pattern queries. However, the drawback of these algorithms is that they take unnecessary intermediary nodes while processing the query thus causing more time to process. To overcome the drawbacks indexing concept is used by algorithms provided in [11] and [12]. Some other algorithms use labeling schemes [13]. In industrial and educational applications these algorithms have verified to be extremely promising [14]. From the study of review we observed the fact that there is less research on xml tree pattern query with wildcard, order restriction and negation.

To concentrate on all these problems, we implement a TreeMatch algorithm that avoids suboptimality of those algorithms. This algorithm is based on the dewey labeling. As per the labeling method, the root node, children, grand children etc. a number or label is related. For example 0 is assigned to root node. The children of root gets labeling such as 0.0, 0.1 etc. The grand children of first parent node start with 0.0.0 and continue like 0.0.1 etc.

1.1 Outline

The rest of the paper is organized as follows: Section 2 gives the reviews the literature that approaching into the research topic. Section 3 gives the preliminary about the existing system and proposed application. Section 4 presents an extended XML tree pattern matching algorithm called TreeMatch. Section 5 presents thorough experimental studies of the novel algorithms and the result of the prototype application. Finally, Section 6 concludes this paper.

2. Related work

As XML databases are growing in quantity and usage by enterprises, it is essential to have an efficient mechanism to answer such queries. In literature many researchers have proposed algorithms for XML tree pattern query processing. Many have handled such queries efficiently. However, most of them did not focus on wildcards, functions, order criteria and negation. In this paper we overcome this entire problem by implementing a TreeMatch algorithm [3]. XML query processing was considered by Timber [15] and Lore DBMS [11]. More focused research on XML storage, and relational databases were done in [16], [17], [18], [19]. These techniques can be leveraged by our holistic algorithm TreeMatch. Choi et al. [20] developed many theorems that are used to prove that the holistic algorithms for XML tree pattern matching can't solve the problem of suboptimality. XML twig queries and their space complexity has been studied in [21] by Shalem and BarYossef. They considered upper bound in parent–child and ancestor– descendent relationships in query processing. Many algorithms developed in existing works have focused on XML tree pattern matching queries in terms of P-C or A-D relationships or both. All these holistic algorithms have drawbacks in query processing. Their computational cost is more as they take some intermediary results in to consideration which is not necessary. That problem is considered matching cross and those results in suboptimiality of performance. Apart from holistic algorithms for XML tree pattern matching queries; there are many other approaches available. For instance ViST [22] and PRIX [23] perform processing by transforming XML tree pattern match into a sequence match. Those algorithms gave importance to only ordered queries and extending them to support unordered queries is non-trivial. In [16] a comprehensive research and experiments were made in order to compare various algorithms used to match XML tree pattern matching queries. Those algorithms are holistic and confirm that they are best available algorithms for XML tree pattern matching queries. They provide guarantee in performance and robust in nature. Nevertheless, they have drawback of using some unnecessary intermediary results that they can't prevent. The proposed algorithm TreeMatch thing in the lines of holistic and overcomes the problem of suboptimality in XML tree pattern matching queries. Moreover it supports 3 types of classes as described and the prototype application implemented demonstrates all three kinds of queries with negative predicates, wild cards, ordered and unordered restrictions. The tree match algorithm is based on the extended dewey labeling scheme.

3. Preliminaries

3.1 Existing system

All the previous algorithms focus on XML tree pattern queries with XML tree queries which may contain negation function, wildcards and order restriction, all of which are commonly used in XML query languages such as XQuery and XPath, for which the query processing gets bit complex.

3.2 Proposed system:

We develop abstract framework on optimal processing of XML tree pattern queries using TREEMATCH ALGORITHM for notations such as Negation functions, wildcard and ordered restrictions. The TreeMatch algorithm is proposed to get larger optimal query classes. It uses a concise encoding technique to match the results and also reduces the useless intermediate results. Most XML query processing algorithms on XML documents rely on certain labeling schemes, such as region encoding scheme, ORDPATH, prefix scheme, extended Dewey scheme. In this paper, we use the extended Dewey labeling scheme, to assign each node in XML documents a sequence of integers to capture the structure information of documents.

I. labeling scheme:

Mainly XML query processing algorithms on XML documents rely on certain labeling schemes, such as region encoding scheme [24], prefix scheme [25], ORDPATH [27], and extended Dewey scheme [26]. In this paper, we use the Dewey labeling scheme, to assign each node in XML documents a sequence of integers to capture the structure information of documents. Dewey labeling scheme is a variant scheme of the prefix labeling scheme. It is an enhanced labeling scheme that considers the root label as zero(0). The children of root are given 0.1, 0.2, etc. The grand children of root are given as 0.1.0, 0.1.1, etc. This labeling scheme can describe the tree easily. Once a node is found, its ancestors and descendents and other relationships can be found easily. For instance, 0.0.1.4 indicates that 0.0.1 is the parent of current node and grand parent is 0.0 and the root is 0. This labeling makes XML tree pattern matching query processing easy.

4. Proposed algorithm

Algorithm 1. holisticmatch for set A //algorithm for wild card

```
1. read book_data
2. while (//* end(root)) do//loop for exicute all nodes
3. fact =getNext(topnode);//search subnode
4. if ( fact is return node);
5. display allNode();
6.      display allSubnode();//display all subnode of parent node book
7. updateSet(fact);

8. empltyAllSets(root);
```

Algorithm 2. holisticmatch for set A //algorithm for negation

```
1. read book_data
2. fact =getNext(topnode);
3. while(//not subnode(node)) do
4. if ( fact is return node);
5.  display allNode();
6.      display allSubnode();
7. updateSet(fact);

8. empltyAllSets(root);
```

These algorithms raises functions provided in [3] to whole query processing. It makes use of locating matching dewey label for the given query and then completes processing. As discussed in the introduction, this algorithm is an enhanced holistic algorithm that makes use of labeling scheme and avoids taking useless intermediary results. Thus its processing time is less and resolves the problem of suboptimality.

5. Experimental Result:

In this section, we present a wide experimental study of TreeMatch on real-life and artificial data sets. Our results verify the effectiveness, in terms of precision and optimality, of the TreeMatch as holistic twig join algorithms for large XML data sets. These benefits become obvious in a comparison to previously four proposed algorithms TwigStack [3], TJFast [16], OrderedTJ [17], and TwigStack-ListNot [26]. The reason that we choose these algorithms for comparison is that 1.similar to TreeMatch, both TJFast and TwigStack are two holistic twig pattern matching algorithms. But they cannot process queries with order restriction or negative edges; and 2. OrderedTJ is a holistic twig algorithm which can handle order-based XML tree pattern, but is not suitable for queries with negative edges; and finally 3. TwigStacklistNot is proposed for queries with negative edges, but it cannot work for ordered queries. Only TreeMatch algorithm can process queries with order restriction, wildcards, and negative edge.

Result

For the XML tree pattern matching while making the query on XML database, we create the data into database. Here we take the example of book data which have different category. Fig.1 shows the main screen of the application search the data of book element. The application supports all kinds of XPATH queries. However, the queries are processed as per the algorithms presented in [1]. Especially dewey labeling made the query processing easy as it is simple to determine ancestor and descendent relationships with dewey labeling.

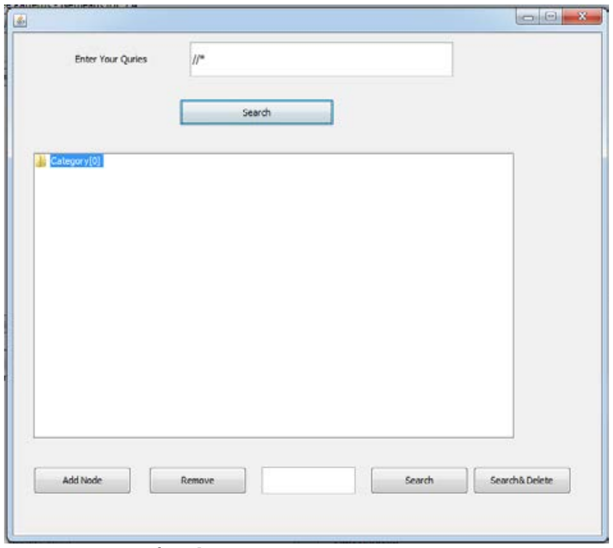
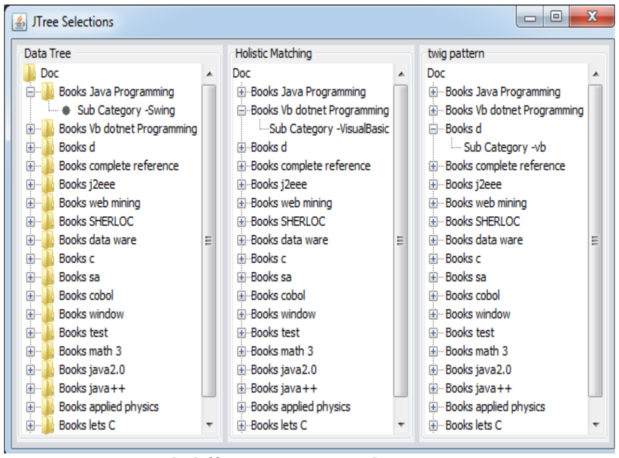


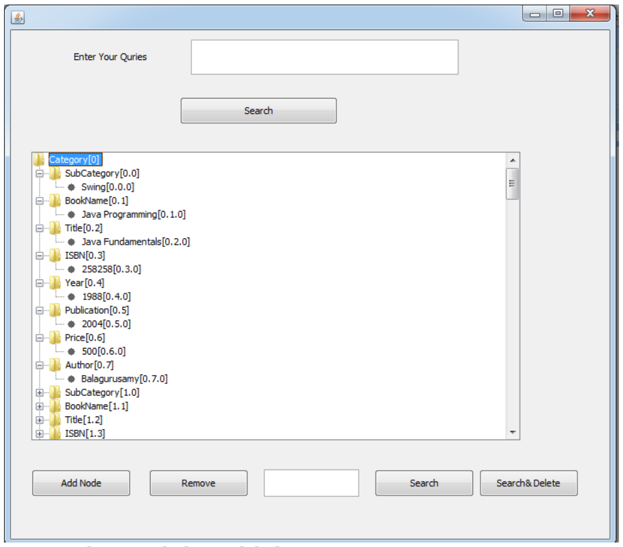
Fig1. Main GUI of application

In Fig. 2 books details are in XML tree format. The root element is category and it contains a collection of books. All queries presently are made on this XML only. However, the application has been tested with a variety of XML files containing variations such as empty elements, elements only elements, mixed elements, elements with body etc. The tested XML files have both complex and simple elements.


JTree Selections

Data Tree	Holistic Matching	twig pattern
Doc	Doc	Doc
Books Java Programming	Books Java Programming	Books Java Programming
Sub Category -Swing	Books Vb dotnet Programming	Books Vb dotnet Programming
Books Vb dotnet Programming	Sub Category -VisualBasic	Books d
Books d	Books d	Sub Category -vb
Books complete reference	Books complete reference	Books complete reference
Books j2eee	Books j2eee	Books j2eee
Books web mining	Books web mining	Books web mining
Books SHERLOC	Books SHERLOC	Books SHERLOC
Books data ware	Books data ware	Books data ware
Books c	Books c	Books c
Books sa	Books sa	Books sa
Books cobol	Books cobol	Books cobol
Books window	Books window	Books window
Books test	Books test	Books test
Books math 3	Books math 3	Books math 3
Books java2.0	Books java2.0	Books java2.0
Books java++	Books java++	Books java++
Books applied physics	Books applied physics	Books applied physics
Books lets C	Books lets C	Books lets C

Fig 2. XML tree with different pattern matching



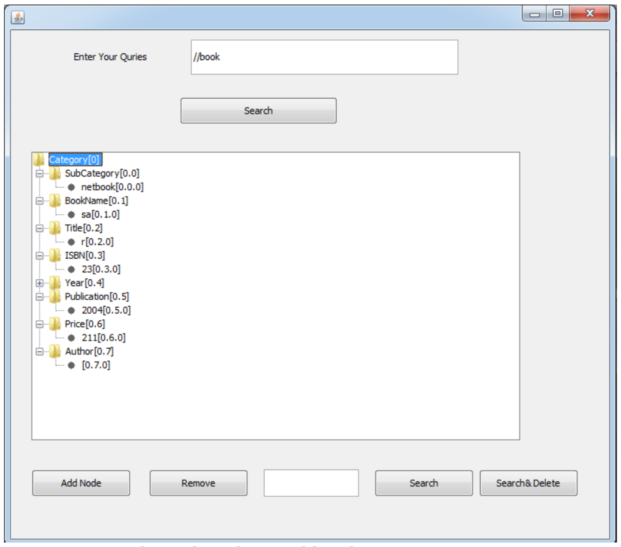
```
Category[0]
  SubCategory[0.0]
    Swing[0.0.0]
  BookName[0.1]
    Java Programming[0.1.0]
  Title[0.2]
    Java Fundamentals[0.2.0]
  ISBN[0.3]
    258258[0.3.0]
  Year[0.4]
    1988[0.4.0]
  Publication[0.5]
    2004[0.5.0]
  Price[0.6]
    500[0.6.0]
  Author[0.7]
    Balagurusamy[0.7.0]
  SubCategory[1.0]
  BookName[1.1]
  Title[1.2]
  ISBN[1.3]
```

Fig 3. Xml tree with dewey labeling

Dewey labeling for the selected XML file is visualized as shown in fig. 3. As can be seen in fig. 3, dewey labeling starts with root element with label zero. All children of root get labeling like 0.1, 0.2 and so on. The root element is “category ” which has label 0. The first child “book” element has label 0.1. In turn the children of first book element have the labels like 0.1.0, 0.1.1, 0.1.2 and so on. This way labeling is applied to the entire tree. When label of any element dewey is known, it is possible to find its siblings, ancestor and descendents easily. Wild cards are also supported in the query processing of the application.

Tree Pattern Matching Queries

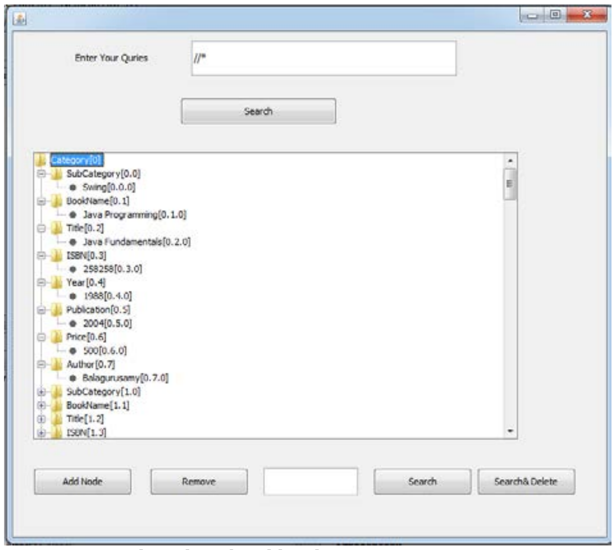
On clicking “Query Answering” button on the main screen of the application user is given a query window where tree pattern matching queries can be given. The proposed application supports the 3 types of queries as described in [1]. Fig. 8 shows both query and the corresponding answer. The query here is //book . It does mean that all book elements irrespective of their position in the XML root element are to be presented. This is evident in the results. The results window also has provisions to further manipulate the XML tree visualized. It supports operations like adding new node, deleting node, searching for a node and searching and deleting a node.



//book

```
Category[0]
  SubCategory[0.0]
    netbook[0.0.0]
  BookName[0.1]
    sa[0.1.0]
  Title[0.2]
    r[0.2.0]
  ISBN[0.3]
    23[0.3.0]
  Year[0.4]
  Publication[0.5]
    2004[0.5.0]
  Price[0.6]
    211[0.6.0]
  Author[0.7]
    [0.7.0]
```

Fig. 5 Query with result without wildcard

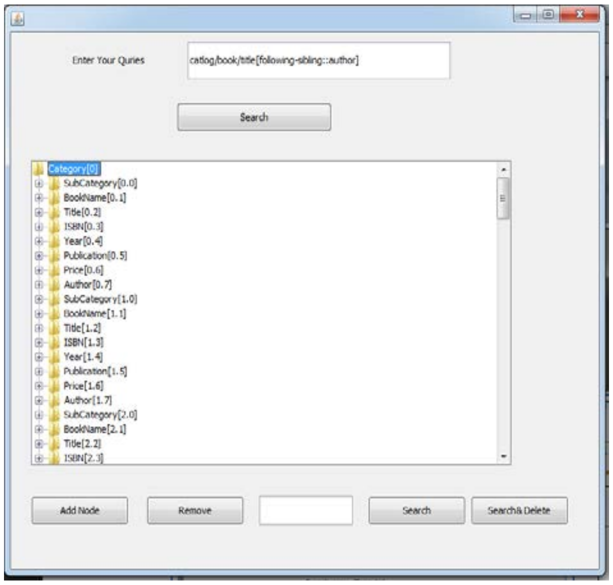


//*

```
Category[0]
  SubCategory[0.0]
    Swing[0.0.0]
  BookName[0.1]
    Java Programming[0.1.0]
  Title[0.2]
    Java Fundamentals[0.2.0]
  ISBN[0.3]
    258258[0.3.0]
  Year[0.4]
    1988[0.4.0]
  Publication[0.5]
    2004[0.5.0]
  Price[0.6]
    500[0.6.0]
  Author[0.7]
    Balagurusamy[0.7.0]
  SubCategory[1.0]
  BookName[1.1]
  Title[1.2]
  ISBN[1.3]
```

Fig 6. Query and result with wildcard

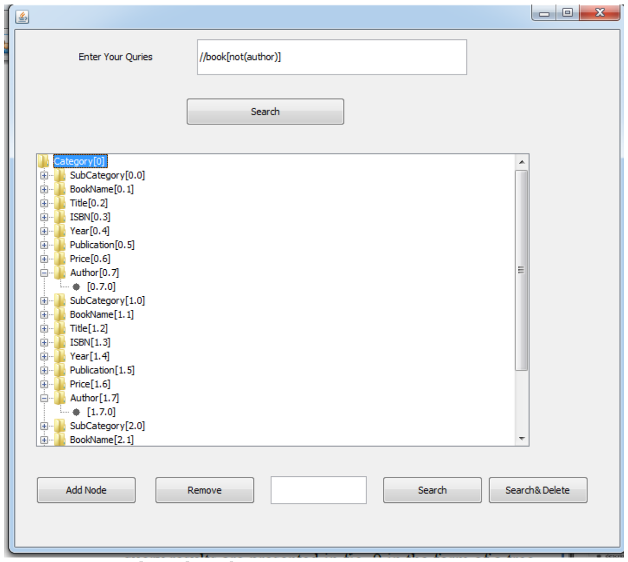
Fig 6 shows the query and result with wildcard. As can be seen in fig. 6, the query is given as “//*”. This does mean that any element present anywhere in the document. The results reveal the entire XML file. The query results are presented in fig. 9 in the form of a tree. JTree class of Java SWING API is used to visualize XML tree. This tree is flexible and can be navigated and manipulated easily. Fig 7. Shows the query with order restriction in terms of sibling. As can be seen in fig. 7 the query given is “category/book/title[following-sibling::author]”. It does mean that the root element must be category and its sub element subcategory.


Enter Your Quries: catlog/book/title[following-sibling::author]

```
Category[0]
  SubCategory[0.0]
  BookName[0.1]
  Title[0.2]
  ISBN[0.3]
  Year[0.4]
  Publication[0.5]
  Price[0.6]
  Author[0.7]
  SubCategory[1.0]
  BookName[1.1]
  Title[1.2]
  ISBN[1.3]
  Year[1.4]
  Publication[1.5]
  Price[1.6]
  Author[1.7]
  SubCategory[2.0]
  BookName[2.1]
  Title[2.2]
  ISBN[2.3]
```

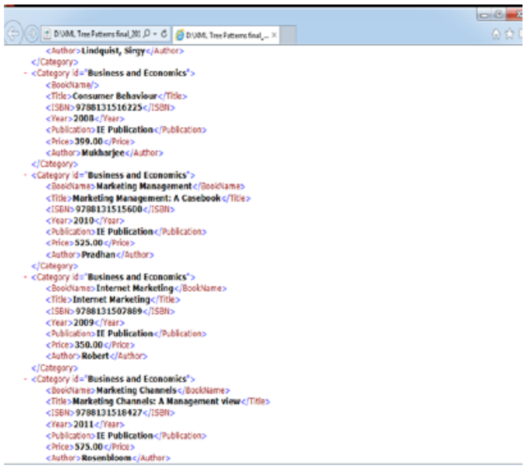
Fig 7. Query and result with Sibling(order Restriction)

As can be seen in fig. 8, the given query is "//book[not(author)]. It does mean that "book" element can be anywhere in the XML file without having "author" element in it. The query results revealed that two books are available without author.


Enter Your Quries: //book[not(author)]

```
Category[0]
  SubCategory[0.0]
  BookName[0.1]
  Title[0.2]
  ISBN[0.3]
  Year[0.4]
  Publication[0.5]
  Price[0.6]
  Author[0.7]
    [0.7.0]
  SubCategory[1.0]
  BookName[1.1]
  Title[1.2]
  ISBN[1.3]
  Year[1.4]
  Publication[1.5]
  Price[1.6]
  Author[1.7]
    [1.7.0]
  SubCategory[2.0]
  BookName[2.1]
```

Fig 8. Query and result with negation



```
    <Author>Lindquist, Sirgy</Author>
  </Category>
- <Category id="Business and Economics">
    <BookName/>
    <Title>Consumer Behaviour</Title>
    <ISBN>9788131516225</ISBN>
    <Year>2008</Year>
    <Publication>IE Publication</Publication>
    <Price>399.00</Price>
    <Author>Mukharjee</Author>
  </Category>
- <Category id="Business and Economics">
    <BookName>Marketing Management</BookName>
    <Title>Marketing Management: A Casebook</Title>
    <ISBN>9788131515600</ISBN>
    <Year>2010</Year>
    <Publication>IE Publication</Publication>
    <Price>525.00</Price>
    <Author>Pradhan</Author>
  </Category>
- <Category id="Business and Economics">
    <BookName>Internet Marketing</BookName>
    <Title>Internet Marketing</Title>
    <ISBN>9788131507889</ISBN>
    <Year>2009</Year>
    <Publication>IE Publication</Publication>
    <Price>350.00</Price>
    <Author>Robert</Author>
  </Category>
- <Category id="Business and Economics">
    <BookName>Marketing Channels</BookName>
    <Title>Marketing Channels: A Management view</Title>
    <ISBN>9788131518427</ISBN>
    <Year>2011</Year>
    <Publication>IE Publication</Publication>
    <Price>575.00</Price>
    <Author>Rosenbloom</Author>
```

Fig 9. XML Document(XML File)

As can be seen in fig 9. The data has inserted into the database is created into the XML file or XML Document.

The experimental results of TreeMatch algorithm is compared with other holistic algorithms such as TwigStack, and TJFast Combined.

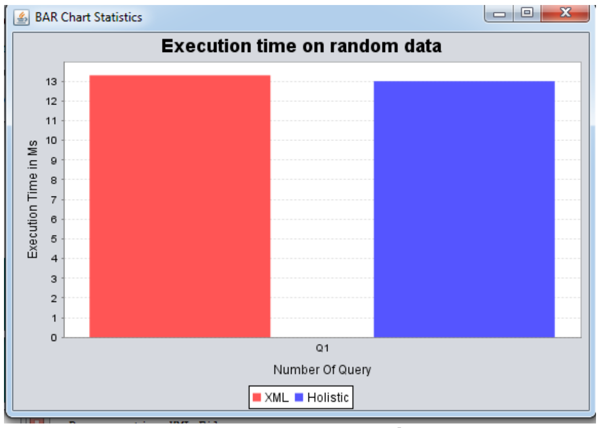


Fig. 10 – Execution time on data

As seen in fig. 10, with respect to large memory with TreeBank data TreeMatch performance is always best.

6. Conclusion:

This paper overcomes the suboptimality in holistic XML tree pattern matching algorithms. The TreeMatch algorithm as described in [3] is explored and the processing of all three types of XML tree pattern matching queries with the help of dewey labeling scheme. A sample application is built to demonstrate the efficiency of TreeMatch algorithm and tested with extensively with all three kinds of queries. The experimental results reveal that the algorithm is able of avoiding repossession of intermediary results before obtaining final results. It does mean that it overcomes the problem of suboptimamlity that existed in the preceding algorithms.

Reference:

[1] A. Berglund, S. Boag, and D. Chamberlin. XML path language (XPath) 2.0. W3C Recommendation 23 January 2007 http://www.w3.org/TR/xpath20/.
[2] S. Boag, D. Chamberlin, and M. F. Fernandez. Xquery 1.0: An XML query language. W3C Working Draft 22 August 2003.
[3] Jiaheng Lu, Tok Wang Ling, Senior Member, IEEE, ZhifengBao, and Chen Wang. Extended XML Tree Pattern Matching: Theories and Algorithms. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 3, MARCH 2011
[4] R. Goldman and J. Widom, “Dataguides: Enabling Query Formulation and Optimization in Semistructured Databases, ”Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 436-445, 1997.
[5] Q. Li and B. Moon, “Indexing and Querying XML Data for RegularPath Expressions,” Proc. Int’l Conf. Very Large Data Bases(VLDB),pp. 361-370, 2001.
[6] N. Bruno, D. Srivastava, and N. Koudas, “Holistic Twig Joins:Optimal XML Pattern Matching,” Proc. ACM SIGMOD, pp. 310-321, 2002.
[7] H. Jiang et al., “Holistic Twig Joins on Indexed XML Documents, ”Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 273-284, 2003.
[8] C.Y. Chan, W. Fan, and Y. Zeng, “Taming Xpath Queries by Minimizing Wildcard Steps,” Proc. Int’l Conf. Very Large Data Bases(VLDB), pp. 156-167, 2004.
[9] W. Wang, H. Wang, H. Lu, H. Jiang, X. Lin, and J. Li, “Efficient Processing of XML Path Queries Using the Disk-Based F&B Index,”Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 145-156, 2005.
[10] N. Bruno, D. Srivastava, and N. Koudas, “Holistic Twig Joins:Optimal XML Pattern Matching,” Proc. ACM SIGMOD, pp. 310-321, 2002.
[11] 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,” Proc. Int’l Conf. Very LargeData Bases (VLDB), pp. 19-30, 2006.
[12] H. Jiang et al., “Holistic Twig Joins on Indexed XML Documents” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 273-284, 2003.
[13] J. Lu, T.W. Ling, C. Chany, and T. Chen, “From Region Encoding to Extended Dewey: On Efficient Processing of XML Twig Pattern Matching,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 193-204, 2005.
[14] P. ONeil, E. O’Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury, “ORDPATHs: Insert Friendly XML Node Labels,” Proc. ACMSIGMOD, pp. 903-908, 2004.
[15] H.V. Jagadish and S. AL-Khalifa, “Timber: A Native XML Database,” technical report, Univ. of Michigan, 2002.
[16] M. Moro, Z. Vagena, and V.J. Tsotras, “Tree-Pattern Queries on a Lightweight XML Processor,” Proc. Int’l Conf. Very Large DataBases (VLDB), pp. 205-216, 2005.
[17] C. Zhang, J.F. Naughton, D.J. DeWitt, Q. Luo, and G.M. Lohman, “On Supporting Containment Queries in Relational Database Management Systems,” Proc. ACM SIGMOD, pp. 425-436, 2001.
[18] 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,” Proc. ACM SIGMOD, pp. 204-215, 2002.
[19] P. ONeil, E. O’Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury ,“ORDPATHs: Insert-Friendly XML Node Labels,” Proc. ACMSIGMOD, pp. 903-908, 2004.
[20] B. Choi, M. Mahoui, and D. Wood, “On the Optimality of the Holistic Twig Join Algorithms,” Proc. 21st Int’l Conf. Database and Expert Systems Applications (DEXA), pp. 28-37, 2003.
[21] M. Shalem and Z. Bar-Yossef, “The Space Complexity of Processing XML Twig Queries over Indexed Documents,” Proc.24th Int’l Conf. Data Eng. (ICDE), 2008.
[22] H. Wang and X. Meng, “On the Sequencing of Tree Structures for XML Indexing,” Proc. 21st Int’l Conf. Data Eng. (ICDE), pp. 372- 383, 2005.
[23] P. Rao and B. Moon, “PRIX: Indexing and Querying XML Using Prufer Sequences,” Proc. 20th Int’l Conf. Data Eng. (ICDE), pp. 288-300, 2004.
[24] C. Zhang, J.F. Naughton, D.J. DeWitt, Q. Luo, and G.M. Lohman, “On Supporting Containment Queries in Relational Database Management Systems,” Proc. ACM SIGMOD, pp. 425-436, 2001.
[25] Q. Li and B. Moon, “Indexing and Querying XML Data for Regular Path Expressions,” Proc. Int’l

Conf. Very Large Data Bases (VLDB), pp. 361-370, 2001.

[26] J. Lu, T.W. Ling, C. Chany, and T. Chen, "From Region Encoding to Extended Dewey: On Efficient Processing of XML Twig Pattern Matching," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 193-204, 2005.

[27] P. ONeil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury, "ORDPATHs: Insert-Friendly XML Node Labels," Proc. ACMSIGMOD, pp. 903-908, 2004.

[28] Marouane Hachicha and Je´ roˆme Darmont, Member, IEEE Computer Society *"A Survey of XML Tree Patterns"* IEEE Transactions On Knowledge And Data Engineering Vol:25 No:1 Year 2013.