Volume 3, No. 6, June 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

INFORMATION RETRIEVAL USING INDEXING SCHEME FOR TREE PATTERN FRAMEWORK

Muthukumar. R [*1] C. Chandrasekar [2]
Research scholar[*1], Associate Professor[2]
[*1,2] Dept. of Computer Science, Periyar University, Salem, TamilNadu –India
rammuthukumar@gmail.com

Abstract: Indexing an XML database in data warehouse is a complex problem. The major rationale for indexing XML database in data warehouse is owing to the heterogeneous and structural environment of XML data that can construct query pattern tedious. Existing techniques focused on clustering methods based on integrating the data warehouse with web data for Online Analytical Processing (OLAP) techniques. Through clustering process, fast retrieval of information is impossible because clustering technique exactly used for tree pattern building framework. Most XML indexing strategies split it into several sub-queries, and subsequently connect their results to present the response to the unique query. Join operations have been determined as the mainly time-consuming component in XML query processing for information retrieval. To enhance the search criteria in XML database present in the data warehouse, in this paper, an indexing scheme is used which separates the data based on the objective. An indexing technique XSeq is presented based on the tree structure pattern framework. XSeq constructs its indexing infrastructure framework on a much simpler and symbolize both XML data and XML queries as formation encoded sequences. Furthermore, the XSeq infrastructure unites both the content and the construction of XML documents, thus it attains a further presentation over indexing both just content and construction, or indexing them individually. A reliable performance improvement is achieved with the proposed IRIS (Information Retrieval using Indexing Scheme) in XML database to data warehouse, compared to an existing SDC technique for OLAP, in terms of search path length, search cost, Maintenance.

Key words: OLAP, Information retrieval, Indexing techniques, Xseq, Search criteria

INTRODUCTION

As trade and endeavors produce and replace XML data more frequently, there is a rising requirement for well-organized dealing out of queries on XML data. An XML query outline normally can be symbolized as an entrenched, tagged tree (or called twig). Organization of XML data, particularly the dealing out of XPath queries, has been the center of significant research and improvement action over the past few years.

Indexing techniques are critical for recently reacting twig queries in a huge database comprising of collections of XML documents. Certain a twig query given by a path expression, a query mainframe needs to identify the illustration of the documents or substructures thereof that assure the charge and structural limitation given in the twig query. A suitable indexing technique can moderately progress the presentation of the matching operation. Twig queries are typically termed as path expressions that comprise no value restrictions. In common, a XML tree pattern query might contain relationships, order restraint, exclusion function, and wildcards. The fig 1 below describes the XPath expressions.

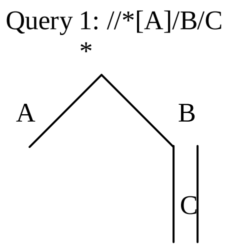

Figure 1 XML query tree pattern framework

A broad mixture of join-based, navigational, and mixture XPath processing strategies are used for processing the queries. These can be integrated with indexing techniques to increase their performance. Another process of XML indexing research centered on clustering strategies of the XML tree nodes. The indexes are constructed on the concept of similarity relative to cluster basically analogous XML elements into similarity classes. Each equivalence class is a directory node and these nodes are linked into a tree or graph supported on their structural relationships. Establishing these index nodes is regularly achieved using a navigational strategy. Clustering is applied for data sets that match to a standard scheme (e.g., an order constantly has an order id and ship date), but the index can produce very big for construction-rich data sets.

In this paper, we are going to implement an indexing technique XSeq for tree pattern framework which is derived from the XML database of data warehouse. Using XSeq, information retrieval process from data warehouse is efficiently done since the tree patterns are classified under the objectives.

LITERATURE REVIEW

A well-organized matching of XML tree patterns has been extensively measured as an interior operation in XML query processing. To our existing knowledge, no trustworthy indexing scheme was established for fractional match full-text queries. Nevertheless, regarding the superior domain of comparison search, there are some remarkable works. Partial-match queries [1] revisit data items that enclose a separation of the query keywords and organize the results supported on the arithmetic properties of the matched keywords. They are important for information repossession on huge document repositories.

A competent dispensation of queries on XML data and penetrating for the incidences of a tree pattern query using different techniques like XSeq [2] in an XML database is a middle procedure in XML query processing. A holistic twig pattern matching algorithm [3] is presented to figure an extended tree pattern matching concepts. Earlier algorithms centered on XML tree pattern queries with only data relationships. A slight work has been done on XML tree queries [4] which may enclose wildcards, order restriction, and negation function, all of which are normally utilized in XML query languages [8] such as XPath and XQuery. Many other current works then observe how to expand the most favorable query class is used to pace up performance using indexes [5].

An embracing investigational study of Tree Match on real-life and synthetic data sets using twig stack [7] by investigating the space complexity of dispensation XML twig queries [6]. A number of strategies are increasing around XML. These technologies encompass of XML Schemas [9], a replacement to DTDs that increases data typing and constrict capacities. In [10], proposed three novel tree structures to competently achieve incremental and interactive HUP mining. A novel genetic programming (GP) approach [11] removing the numerous tree prearranged patterns from tree planned data using soft clustering with a visualization representation [12]. In this work, we are implementing an indexing scheme to tree pattern framework for information retrieval process.

INFORMATION RETRIEVAL USING INDEXING SCHEME FOR TREE PATTERN FRAMEWORK

The proposed indexing scheme is efficiently designed for information retrieval process from data warehouse based on tree pattern framework. The proposed information retrieval indexing scheme operates under two phases. The first phase is to identify the XML data present in the XML database present in the active data warehouse. The second phase is to use an indexing scheme XSeq to the constructed tree pattern framework. The architecture diagram of the proposed information retrieval indexing scheme for tree pattern framework is shown in fig 3.1.

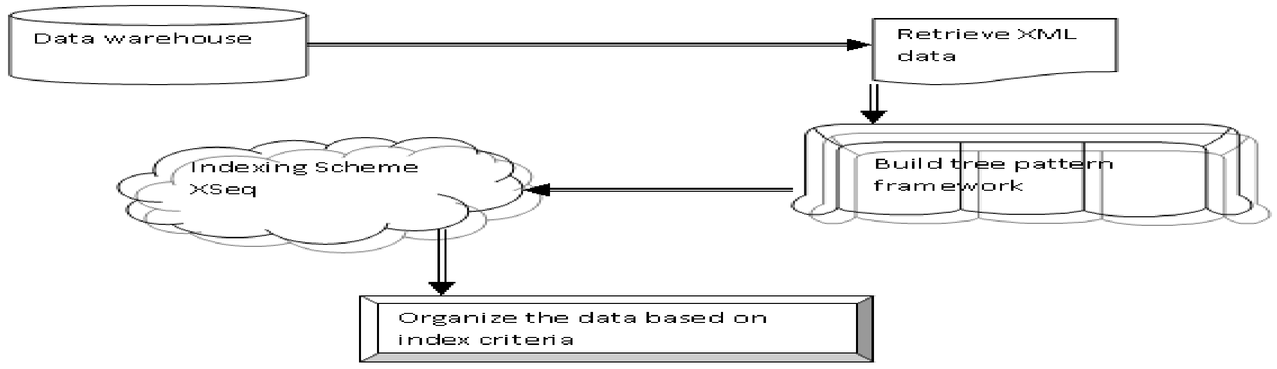
Data warehouse
Retrieve XML data
Build tree pattern framework
Indexing Scheme XSeq
Organize the data based on index criteria

Figure 3.1 Architecture diagram of the proposed IRIS using XSeq

The first operation is identifying the XML data retrieved from the data warehouse. Information retrieval systems are regularly differentiated with relational databases. Conventionally, IR systems have recovered information since formless text - by which mean ``raw" text with no markup. Databases are considered for querying collection of records that include values for predefined attributes such as employee number, title and salary.

The second operation is to apply indexing scheme for constructing tree structure pattern framework based on XSeq indexing scheme. XSeq constructs its indexing communications framework based on simpler data model i.e., sequences. Signify XML data and queries equally as formation encoded sequences. This novel data representation conserves query equality, and more significantly, structured queries can be countered openly without resorting to join operations. Furthermore, XSeq infrastructure framework unites indices on mutually the content and the structure of XML documents for indexing them individually.

Process of Xseq:

Querying XML data is secure to identifying the sub structures of the data that equal the query construction. The query structure comprises values, tree patterns, wild-cards (‘*’ and ‘//’), etc. The principle of XML indexing is to present effective support for structured queries. Nevertheless, in state of-the-art indexing resolution tree pattern is the most normally sustained query interface and represented as:

$$Simplepath \Rightarrow p(NodeIds)$$

That is, for a given path, the index construction precedes a collection of nodes that symbolize such a path. Some index strategies expand the above interface to sustain paths that begin with a ‘*’ or ‘//’ with a much bigger index. Still, for tree-pattern queries or queries with ‘*’ or ‘//’within, it is necessary to molder them into a collection of simple path queries. Then join operations are integrated with it to response the original query. To avoid expensive join operations, in this work, a sequence-based XML indexing, is presented which represents a chief disappearance from preceding XML indexing approaches. The new method gives a more common query interface as

$$Treepattern \Rightarrow p(DocIds)$$

That is, for a given tree pattern, the index proceeds a collection of XML documents that enclose such a pattern.

XSeq for building tree pattern framework:

Instead of openly controlling tree structures, XSeq assumes a data model: sequences. There are numerous ways to change a tree structure to a sequence. For illustration, we can predetermine each node by the path primary from the root to the node, and symbolize an XML tree by its preorder series. In the similar way, transfer XML queries to preorder sequences.

The intention is to achieve XML queries by subsequence matching so that prearranged queries can be practiced as a whole as a substitute of being busted out into slighter query units, since merging the outcomes of these sub queries during join operations is exclusive. The most significant step is to ascertain query similarity among a structure match and a subsequence match. The process of building the tree pattern framework is shown in fig 3.2.

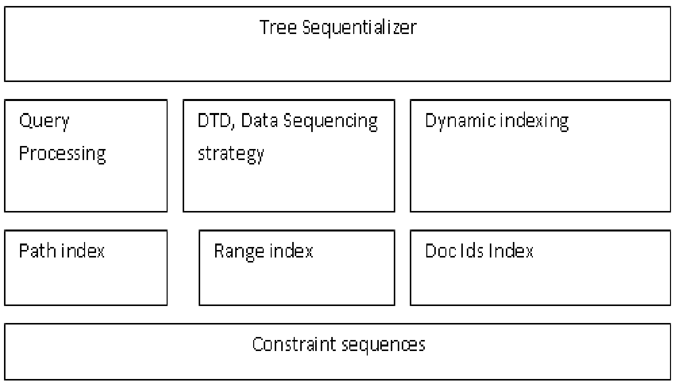

Figure 3.2 Process of building tree pattern framework using XSeq

The partition of XML document is done based on the XSeq indexing scheme. XSeq recognizes a collection of chronological representations for a tree structure, each of which conserves query equivalence. Then XSeq formulates the decision based on the DTD schema or the delivery of the dataset, which guides us to the 'best' sequencing technique in terms of index and query performance. The process of building tree pattern framework is defined with an illustration made over with the books and can be partitioned based on the indexing scheme by XSeq. The partitioned XML document based on indexing units using XSeq is shown in fig 3.3.

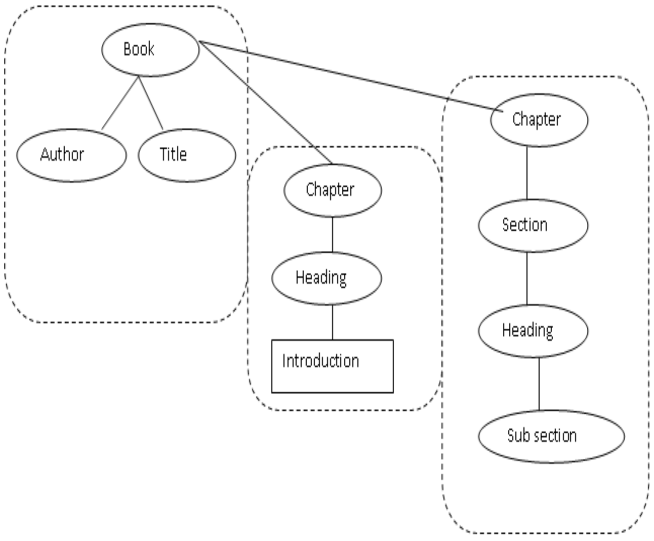

Figure 3.3 Partition of XML document into tree pattern framework based on indexing scheme

From the above figure, book is an XML document which has been partitioned based on indexing scheme XSeq. The Xml document is partitioned based on title, author, Chapter and sections. So, it will be easy for the user to search the necessary documents and retrieve it from the data warehouse. The algorithm below describes the process of indexing scheme using XSeq for information retrieval.

Step 1: Data Warehouse (large collection of data)
Step 2: Identify the XML data (Entities and its relationships)
Step 3: Retrieve the semantic data
Step 4: For an illustration
Consider a sample set of XML schema as:
<xs: schema>
<xs: element name="book">
<xs: element name="title" type="xs: string“/>
<xs: element name="author" type="xs: string" minOccurs="0" maxOccurs="“2"/>
</xs: element>
</xs: schema>
Step 5: Use XSeq indexing scheme
Step 6: Partition the XML document as several indexes based on its types and
objectives.

Index 1
```
<Book>
            <Title> …..
            </Title>
            <Author>
               List of authors
            </Author>
</Book>
```
Index 2
```
<Book>
            <Chapter>
              <Heading>
                <Sub-heading>
                  <Introduction>
                </ Sub-heading>
             </Heading>
            </Chapter>
</Book>
```
Index 3
```
<Book>
            <Chapter>
              <Section>
                <Sub-section>
                  </ Sub-section>
               </Section>
            </Chapter>
</Book>
```
Step 7: Build the tree pattern framework based in index (As like Fig 3.3)
Step 8: Structured tree formed based on indexing scheme

The above algorithm describes the entire process of indexing scheme using XSeq for tree pattern building framework mainly for information retrieval process. Step 1 to 4 describes the process of retrieval of XML data from the data warehouse. Then, XSeq indexing scheme is applied to those retrieved XML data to form a tree structure. A sample XML schema is used and the schema is built as tree structure based on index criteria depends on its types. So, it will be easy for the user to search the required XML data from the tree by choosing the efficient shortest path to reach the destination.

EXPERIEMNTAL EVALUATION

Widespread experimental studies have been examined to estimate the performance of the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework. We have employed the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework in Java, and permitted out a series of performance research in order to monitor the effectiveness of the proposed information retrieval process using XSeq indexing scheme for tree pattern framework. We ran our experimentation using diverse sets of XML data and queries. The proposed indexing scheme approach efficiently built the tree pattern framework for the XML data from DW using XSeq and tree pattern framework is done based on index criteria. The performance of the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework is measured in terms of

i) search path length,
ii) search cost,
iii) Maintenance.

Search Path Length is defined as the average number of index path traversed by a query message prior to it achieves the destination.

$$Searchpathlength = \sum_{i=1}^{n} Q_1 \quad \text{......... (eqn 1)}$$

Where i=1 to n be the path taken by the query Q_1
Search cost is defined as the average number of XML data earns in the search process.

$$search\,\text{cost} = retrive(d, PL, t) \quad \text{.......... (eqn 2)}$$

Where d- XML data
PL –Path Length
t- time interval
i.e., average number of XML data being retrieved from the XML database at a particular interval of time t.
Maintenance is defined as the average number of index criteria are maintained under the indexing scheme.

RESULTS AND DISCUSSION

In this work, we have observed how the tree pattern framework is competently built with the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework written in mainstream languages such as Java. We used different sets of XML data for comparing the results of the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework with an existing SDC technique for integrating the DW and web data using semantic data. The below table and graph describes the performance obtained through experimental evaluation.

Table 5.1 No. of queries vs. Search path length

No. of queries	Search path length	
	Proposed IRIS	Existing SDC
5	2	8
10	6	12
15	9	16
20	12	19
25	15	22

The above table (table 5.1) describes the process of queries choosing the index path to reach the destination in the tree pattern framework. The outcomes of the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework is compared with an existing SDC technique for integrating the DW and web data using semantic data.

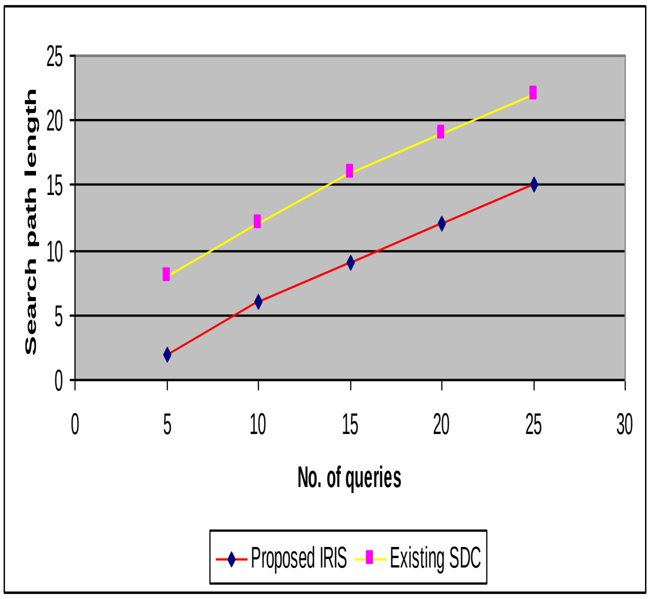

Figure 5.1 No. of queries vs. Search path length

Fig 5.1 describes the process of queries choosing the index path to reach the destination in the tree pattern framework. Since the proposed IRIS used XSeq for build the tree, the tree pattern framework is done based on sequences. The trees were built with an appropriate index criterion and has been grouped based on the objective. So, the process of analyzing the required data by the user consumes less time. The path chosen for searching the required data had also been small. Even the number of queries increases in the proposed IRIS, the queries will reach the destination in a less interval of time by choosing the smart path to reach the destination. Compared to an existing SDC technique, the search path length of the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework is less and variance is 25-30% less in the proposed IRIS.

Table 5.2 No. of queries vs. Search cost

No. of queries	Search cost	
	Proposed IRIS	Existing SDC
5	8	4
10	13	7
15	17	11
20	19	14
25	24	16

The above table (table 5.2) describes the amount of retrieval of XML data in a particular interval of time from the tree pattern framework. The outcomes of the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework is compared with an existing SDC technique for integrating the DW and web data using semantic data.

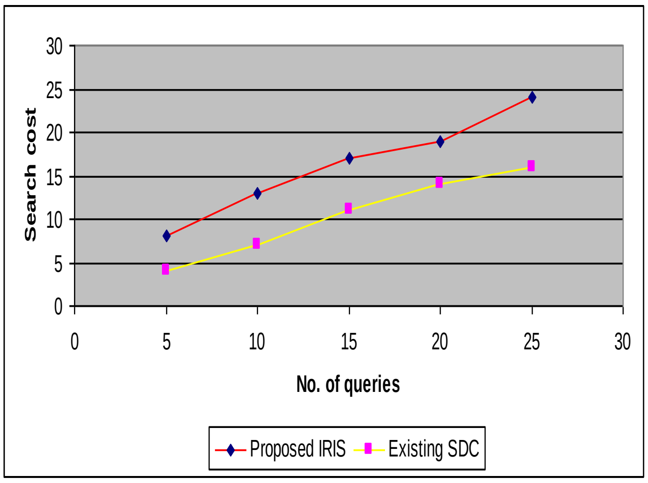

Figure 5.2 No. of queries vs. Search cost

Fig 5.2 describes the amount of retrieval of XML data in a particular interval of time from the tree pattern framework. Since the proposed IRIS used less path length, the amount of XNL data retrieved by processing the query is high. The tree pattern framework is done based on index criteria, the amount of retrieval of information is high even the number of queries in the queue length increases. The proposed IRIS accomplished shortest path for searching the index value, the amount of retrieval of information at a particular interval of time is large enough compared to an existing SDC technique for integrating the DW and web data using semantic data and the variance is 30-40% high in the proposed IRIS.

Table 5.3 No. of data vs. Maintenance

No. of data	Maintenance of tree (index criteria (%))	
	Proposed IRIS	Existing SDC
10	14	5
20	20	12
30	28	17
40	35	20
50	50	22

The above table (table 5.3) describes the maintenance of tree pattern framework when more number of data contents is added into the database. The outcomes of the proposed information retrieval indexing scheme using XSeq indexing scheme for tree pattern framework is compared with an existing SDC technique for integrating the DW and web data using semantic data.

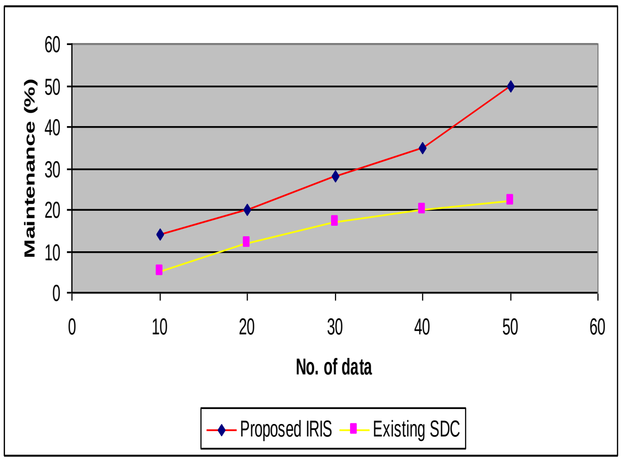

Figure 5.3 No. of data vs. Maintenance

Fig 5.3 describes the maintenance of tree pattern framework when more number of data contents is added into the database. The existing SDC technique built the tree pattern framework by clustering technique, so the process of maintaining the XML data in the framework is not as good as the proposed IRIS. Compared to an existing SDC technique for integrating the DW and web data using semantic data, the proposed IRIS maintained the tree pattern framework efficiently by using the sequences and grouped under the on objective of the XML data.

At last, it is being noted that the proposed IRIS efficiently built the tree pattern framework based on indexing schemes and allowed the search path criterion framework as easy as possible even the number of data contents in the XML database increases.

CONCLUSION

In this paper, we proposed an indexing scheme XSeq that supports for building the tree pattern framework based on index criteria. We developed the tree based on sequences using XSeq by minimizing the search path length for retrieval of information from XML database present in the data warehouse. Search path length is a significant performance evaluation since it concerns the query response time. The proposed IRIS shows that the search path is shortest. To recognize the purpose for a query, the proposed IRIS system consumes almost only half of the number of index path compared to an existing SDC technique. Experimental results showed that the proposed IRIS using XSeq

Efficiently builds the tree pattern based on indexing scheme and it allowed to process the contents in the database in a shortest path in a less interval of time compared to an existing SDC technique. The performance evaluation brings out the proposed IRIS provides fast retrieval of information and the performance is 80-90% high contrast to an existing SDC technique which builds the tree pattern based on clustering technique.

REFERENCES

[1]. Dyce Jing Zhao, Dik Lun Lee, ET. AL., “DPTree: A Distributed Pattern Tree Index for Partial-Match Queries in Peer-to-Peer Networks”, EDBT'06 Proceedings of the 10th international conference on Advances in Database Technology, Pages 515-532, 2006

[2]. Xiaofeng Meng , Yu Jiang et. Al., “XSeq: An Indexing Infrastructure for Tree Pattern Queries”, Proceedings of the ACM SIGMOD international conference on management of data, ACM Press (2004)

[3]. Jiaheng Lu, Tok Wang Ling, et. Al., “Extended XML Tree Pattern Matching: Theories and Algorithms”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, 2011

[4]. A. Berglund, S. Boag, and D. Chamberlin, XML Path Language (XPath) 2.0, W3C Recommendation, http://www.w3.org/TR/xpath20/, Jan. 2007.

[5]. 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 Large Data Bases (VLDB), pp. 19-30, 2006.

[6]. 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.

[7]. T. Yu, T.W. Ling, and J. Lu, "Twigstacklistnot: A Holistic Twig Join Algorithm for Twig Query with NOT-Predicates on XMLData," Proc. Database Systems for Advanced Applications (DASFAA),pp. 249-263, 2006.

[8]. 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.

[9]. Zhifeng Bao, Jiaheng Lu et. Al., 'Towards an Effective XML Keyword Search', IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 8, AUGUST 2010

[10]. Ahmed,C.F. Tanbeer, S.K. et. Al., "Efficient Tree Structures for High Utility Pattern MininginIncrementalDatabases",IEEE Transactions on Knowledge and Data Engineering, Volume**:** 21 , Issue: 12 , Dec 2009

[11]. Kengo Yoshida et. Al., "Evolution of multiple tree structured patterns using soft clustering", 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE).

[12]. Liang Gou et. Al., "TreeNetViz: Revealing Patterns of Networks over Tree Structures", IEEE Transactions on Visualization and Computer Graphics, Volume**:** 17 , Issue: 12, dec 2011

Short Bio Data Authors Profile

Mr. R.Muthukumar obtained M.Sc., Computer Science from P.S.G College, Bharathiyar University, Coimbatore, Tamil Nadu, India, in 2000, and M.Phil., Computer Science from Manonmaniam Sundaranar University, Thirunelveli, Tamil Nadu, India in 2002. He was working as IT analyst , in Department of Software development, **TATA Consultancy Services** ,Bangalore, Karnadaka, India. Currently he is working as Tech Lead in Department of Software development, **Infosys**, Bangalore, Karnadaka, India. He is pursing Ph.D in Data warehouse.

Dr. C. Chandrasekar completed his Ph.D in Periyar University, Salem at 2006. He worked as Head, Department of Computer Applications at K.S.R. College of Engineering, Tiruchengode, Tamil Nadu, India. Currently he is working as Associate Professor in the Department of Computer Science at Periyar University, Salem, Tamil Nadu. His research interest includes Mobile computing, Networks, Image processing and Data mining. He is a senior member of ISTE and CSI.