International Journal of Pure and Applied Mathematics
Volume 118 No. 24 2018
ISSN: 1314-3395 (on-line version)
url: http://www.acadpubl.eu/hub/
Special Issue

http://www.acadpubl.eu/hub/

AUTOMATED KEYWORD QUERY PROCESSING IN HIERARCHICAL STRUCTURE

Mrs. S. Sahunthala, S.Aishwarya Varu,
Janani.K, Mary Florence.S.A.
Department of Information Technology,
Anand Institute of Higher Technology,
Chennai.
sahukalai2011@gmail.com ,

May 27, 2018

Abstract

ABSTRACT XML is a de facto standard for Internet data exchange, XML data management has several new features in the cloud computing environment, and XML data has been widely used in configuration languages. XML keyword queries has been used in many applications, such as that in scientific, business and other domain for storing, publishing and exchanging data .We define XML data exchange settings in which source-to target dependencies refer to the hierarchical structure of the data .With the growing importance of XML in data exchange, much research has been done in providing flexible query mechanisms to extract data from XML documents with the approach such as top-down. In structure query language, XPath and XQuery is used by the users to understand the complex query language. The existing technique undergoes LList and Baseline algorithm degrades the performance when index size is large. To enhance the performance in the keyword query we propose an Automated Keyword Query Processing in Hierarchical Structure. The technique that we are going

to propose is fully based on automation. Here we perform query processing using automation tool.

***Key Words*:**XML data, XQuery, index, automation,LList

1 Introduction

XML Query Processing and Keyword Search is using the latest technique for searching XML data. An overview of the current query processing and keyword search techniques on XML data is presented which includes XML labeling schemes, indexing, processing XML tree patterns(order and un-order) , XML query optimization, it estimates the result, as well as searches the XML keyword. There are three well-known and widely implemented query languages in XML such as: XPATH, XSLT, and XQuery. XPATH is the simplest of the three, where XPATH provides a path-like syntax for refers to nodes in an already existing document. XSLT is an XML-based syntax with a recursive processing model, an new XML nodes from scratch is constructed from multiple input documents. XQuery is a mostly similar to XSLT and has SQL-inspired syntax with a (first-order) functional processing model and functionality

XML query language designers face several difficult tasks and choices, in addition to the usual challenges facing programming language designers in addition to XML query language designers face several difficult tasks and choices, and these choices that permeate the rest of the language. The data model that is being chosen should certainly be more expressive than XML itself .It must also handle both XML queries as well as non XML queries are evaluated.

XML Schema as the basis for type system, it already defines conversions from the textual representation to all simple types and from all simple types to a canonical textual representation it may or may not match string conversions but XML Schema does conversions between types is not defined. Derivation by list, derivation by extension, concepts of XML schema are substitution groups and that which pose difficulties are local types. If other type of chosen then XML Schema as the basis for the type system, then it must decidehow to handle XML that has a schema associated with it or that which contains meta-data attributes , and some level of interoperability with existing schema-based applications and formats are expected by the users.

XML has the accepted standard for data storage, exchange, and integration over the Internet. An interesting and challenging research topic in XML database research because of the rapidly emerging applications in XML query analysis and optimization is the standard for data storage is XML query processing. XML query processing includes the pattern matching.

In the remaining paper section 2 describes the literature survey to improve the performance of query generation, section 3 deals with the related work of our model, section 4 shows the experimental model of our technique and section 5 gives the conclusion and future work of our model.

1.1 LITERATURE REVIEW:

XML has been successfully designed to store and transport data. Keyword search is as popular as XPath and XQuery[8], keyword search is popular as it can put at one's ease and it gained lot of attention the basic semantics used in existing paper is Lowest Common Ancestor (LCA)[6].Based on LCA there are two other semantics ELCA(Executive LCA) [3] and SLCA(Small LCA)[1][2].An algorithm can support only one query semantic at a time. It is not possible to design an algorithm which supports all query semantics because it cannot work well all the time.

Structural index is also introduced in order to improve the query efficiency, sequential index is also used in order to come out with an efficient query. First the nodes where represented using the Dewey label[3] and then it was represented using Prime numbers, there were lot of techniques such as LCA,ELCA,SLCA[1][2][3] which made improvements on firstly visiting all the nodes and skipping null nodes. Still they recursively visit the null node, the two common problems in the existing techniques is CAR [1] and VUN [1] problem. The CAR is also known as Common Ancestor Repetition [4].Here each node 'x' is represented as Dewey label. The Dewey labeling consist of a set of components from the root to x. Here they wont notice that some component repeatedly appear in many different Dewey label. The VUN problem [1], here the number of visited components is more. In top-down [10] the components which occur in Dewey labels are visited in left to right order. The labeling scheme in this method is L list, which visits every node

only once and stores information and answers the given keyword. The L list is labeling scheme independent but this technique is not suitable if the size of the indexes becomes too large to be loaded into the memory. Now let us analyze the existing techniques (2.1, 2.2, with the following hierarchical data.

$< student >$
<student id>111</student id>
<student name>
$< Fname > aaa < /Fname >$
$< Mname > bbb < /Mname >$
$< Lname > ccc < /Lname >$
$< /studentname >$
<student class>3 </student class>
<student result>pass</student result>
<student marks> 85 </student mark>

1.2 Dewey Labeling scheme:

The Dewey labeling scheme [3] is used to label an XML document to facilitate XML query processing .We introduce a labeling scheme, called extended Dewey scheme to improve the efficiency of XML tree pattern matching, we introduce, it effectively extends the existing Dewey labeling scheme to combine the types and identifiers of elements in a label, and to avoid the scan of labels for internal query nodes to speed up query processing. All the occurrences of the tree pattern is found in a XML document is a main operation for efficient correction of XML queries. This scheme is used to label an XML document by recording information on the path of an element.

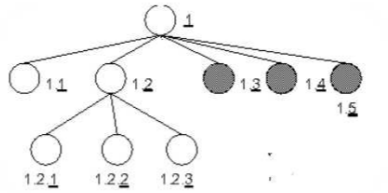

Figure .1 Dewey labeling for student xml data

Fig 1 shows the Dewey labeling for student xml database. Here when new nodes are inserted it suffers performance degradation. This is the limitation of this method. Here the root node is denoted as 1 and their corresponding leaf node is denoted as 1.1,1.2,1.3,1.4,1.5. Hence the hierarchical tree structure is constructed for the given XML document.

1.3 Region Labeling scheme:

Region extraction and labeling process in building computer systems that analyzes automatically and interpret hierarchical structure with sequences of a scene. It involves extracting meaningful regions and their attributes from the given hierarchical structure. The applicability of the scheme stems from its strengths in identifying the fundamental and common computational steps and it reduces the number of scan of the tree structure.

Hence the input is given as hierarchical tree structure. Then the region is selected as per our interest, and the texture information is added and the color is added as per our interest, then the region expands in multiple ways at last the resultant tree structure of the hierarchical XML data is obtained. This method consumes lot of time therefore the performance is degraded. As the process to select the required region takes lot of time and the time to construct the tree takes long time.

1.4 Lowest common ancestor:

LCA[6] of nodes in a rooted tree is defined as the node located farthest from the root that has both as descendants. For example, in figure 2, LCA of node 4 and node 9 is node 2. Here, the lowest common ancestor for node 9 and node 4 is node 2.

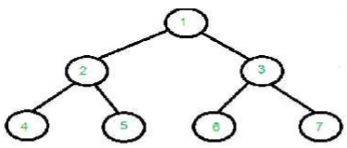

Figure 2 General LCA Labeling Semantic

Hence LCA(4,7)=1,LCA(5,6)=1,LCA(4,5)=2and LCA(6,7)=3. LCA is useful for finding shortest path between two nodes of Binary Tree but it does not suit for all kinds of tree structure. A variant of the problem is the dynamic LCA problem in which the data structure should be prepared to handle LCA queries intermixed with operations then the tree is changed. This is done by maintaining the forest using the dynamic trees data structure as the size is partitioned, this then maintains a decomposition of each tree, and allows LCA queries to be carried out in logarithmic time in the size of the tree.

1.5 Smallest Lowest Common Ancestor:

Cloud provides dynamically computing services for very large scales of data over the Internet. IaaS (Information as a Service) is one of most important utilities to provide information service in Cloud computing. XML data are produced in large scales continually in Internet. So we need efficient information filtering services. Nowadays many of them are using keywords to describe requirements. SLCA [6] (Smallest Lowest Common Ancestor)-based XML keyword search is one of the most important approach for retrieving information. Previous approaches focus on building centralized index for a large scale of XML document collection but they cant process continuous XML streams. SLCA computing scheme for continuous XML document. A SLCA computing scheme is designed, where SLCAs are obtained in just single scan of XML stream.

Hence, the figure 2 the tree is branched into many parts, and one part of the tree is considered as SLCA, this is efficient compared to LCA but here it suffers from updation. It cannot update automatically.

1.6 CAR problem:

In the CAR [1] (Common Ancestor Repetition) problem, each node is referred to as a u a Dewey Label based upon which two operations are used to generate efficient results.

i) Two nodes are tested for their document order &

ii) Computation of LCA for the two nodes.

Dewey Label u is a set of components, in which each component itself corresponds to a node from root to u in the XML tree and these set of components together represent the node u, such that both the operations are visiting all the Common Ancestors (CAs) of the two involved nodes at least once.

Thus, u could be the CA for several multiple nodes, performing both the operations frequently, that will make all CAs on the path from root to u be repeatedly visited, which is named as Common Ancestor Repetition.

1.7 VUN problem:

Let U be the set of nodes in a given document D and given a keyword query Q. The document D contains at least one query keyword in any of their sub-trees and thus nodes of U can be categorized as,

i) Common Ancestors (CAs) [1] [10], which contains all the query keywords in its sub trees.

ii)Useless Nodes (UNs) [1, 10], the node that will not be the child for any CA node.

iii)Auxiliary Nodes (ANs) [1, 10], the nodes that will definitely be the child for some random CA node.

The main problem is that not even a single existing method can avoid visiting the UNs, thus we call this as VUN problem. The main reason for visiting the UNs mainly depends on LCA or SLCA or ELCE semantics. They explain that the satisfiability of any node u is determined only by that nodes i.e. us descendant nodes including the UNs, instead of its auxiliary nodes.To overcome the above two problems some efficient methods and algorithms have been found in order to reduce the count of visited components.

To encounter the CAR Problem, a generic top-down XML keyword query processing method have been found which considers the Dewey Label components as the basic processing unit and makes them to visit all the CA nodes in depth-first left-to-right order. This method can also be used to find any one of the xLCA (i.e. LCA or SLCA or ECLA) results.

Likewise, to encounter the VUN Problem, child nodes have been used instead of descendant nodes to check the satisfiability of any node u depending upon xLCA semantics. To avoid visiting the

UNs, a top-down algorithm namely TDxLCA [1] has been proposed depending upon traditional inverted lists to obtain xLCA nodes.

To maintain every node in each level of the traditional inverted list just once and to store all required information for acknowledging a given keyword query without any damage to those informations, a labeling-scheme-independent inverted index, namely LList[8] has been proposed.

Both LList Based Algorithm and Hash Search Based Algorithm has its own disadvantages and in this paper we are going to propose some other alternate methods and algorithms to overcome the problems faced by the above two algorithms.

1.8 Hash search based algorithm:

ELCA based algorithm can work well for both CAR and VUN Problems, especially TDELCA, TDELCA-L, TDELCA-H, TDELCS-HO. The Hash Search Based Algorithm used in LList [8] is ELCA algorithm, which repeatedly gets all the CA nodes in a top-down manner and then checks the satisfiability of each CA node operates on traditional inverted list of labels w.r.t Dewey Label or any one of its variants.

To achieve this it has to overcome two main problems,

- To find the CA nodes (us) set of child CA nodes.
- To test the satisfiability of u w.r.t ELCA semantics.

For instance, given a CA node and its subtree and to obtain the set of child CA nodes of each CA node, it is not necessary to check whether each subtree consists of all the keyword queries, instead it is necessary to check whether each node contains least number of child nodes of u. LList Based algorithm requires two search operations to find the first and last IDDewey [5] labels and thus it performs Binary Search twice. The node u that is closer to the root node and in each inverted list, the most repeated node for u is the longer child list of u.

The factor that affect its performance is the length of the search interval since intersection operation has been performed on the set of child lists of u .Both TDELCA-H and TDELCA-HO [7] have similar time complexity and it is not perform efficient binary search when compared to TDELCA, TDELCA-L.

Time Complexity (TDELCA-H) = O (n — Ri (u) —)

Both TDELCA and TDELCA-L are similar and thus LList used TDELCA-L since the binary search can be done more efficiently when compared to TDELCA, TDELCA-H, TDELCA-HO. The results obtained in this algorithm are then extended to other semantics as well to check the satisfiability of each node of u.

Time Complexity (TDELCA-L) =

$$o(n \sum_{u \in CA(q)} |R_i(u)|. \log |R|)$$

where,

u contains at least one occurrence of each keywords in Q

CA Common Ancestor

q input query

R_i set of child nodes of u

R set that contains maximum number of child nodes for all the nodes

As a result of these extensions to other semantics, extra query semantics have also been proposed namely XSearch, MLCA and VLCA. The common characteristics of these extra semantics are that, the criteria that they use for validation must check whether the names of all the nodes from LCA nodes to its descendant nodes contain any keyword nodes directly.

Hash Search Based algorithm depends on the comparison operation (performed by binary search) to adjust the cursors of the inverted lists. An additional hash index has been used on the inverted lists to improve the overall performance due to which each comparison operation takes only O(1) time complexity without using binary operation.

The Baseline Hash Search Based Algorithm considers the shortest list as working list. It repeatedly processes all the CA nodes in a top-down manner. Then for each CA node u, it coherently checks whether each of the child node that are present is a CA node. If so, then it provides u as an ELCA result.

Both TDELCA and TDELCA-L performs tedious process and TDELCA-H is used only to process CA nodes and their child nodes in lists. It checks whether u is a CA node by hash comparison operation instead of performing an intersection operations on a set of child lists. Even though both the methods reduces time complexity in an efficient manner they failed to perform XML keyword query

processing for large sized indexes that can be completely loaded into the memory.

To overcome this problem, we are going to propose some other efficient algorithms to perform indexing even for very large index sizes. We are also including some traces of both LList based algorithms and Hash based search algorithms to avoid visiting the UNs and to find the satisfiability of each node u w.r.t. all the semantics to some lower extents

2 RELATED WORK:

2.1 PROBLEM DEFINITION:

To design a XML keyword query processing system by using automatic query generation approach so that it can be used for faster and efficient retrieval of the pertinent document.

2.2 SYSTEM ARCHITECTURE:

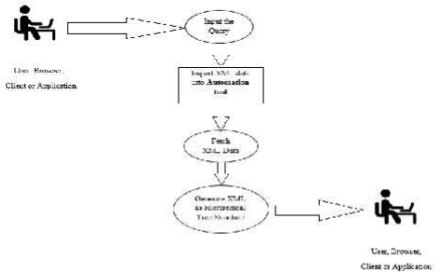

Figure 3 System Architecture of Proposed Model

In the proposed system, query processing of XML documents were done using indexing techniques such as LList based algorithm and Hash search based algorithm for relevant keyword generation. The proposed system used,

• Tree-based structure in which nodes were represented as an attribute or a value

• Hierarchical relationships represented XML elements as edges.

XML Query processing on XML documents [9] mainly depends on tree traversal techniques such as top-down or bottom-up techniques. However, these techniques produces large amount of XML

documents. To minimize the overhead of XML Query processing on elements, automatic query generation techniques are used dynamically and it includes processing the XML documents effectively by importing it into Oxygen XML Editor Tool.

Oxygen XML Editor is the best Editor and Developer Tool available and is used mainly for automatic XML query generation in an efficient manner. Oxygen XML Editor is mainly platform independent and can be used as a standalone application or in conjunction with Eclipse. Our method functions as follows,

User or browser inputs the query into the Oxygen XML Editor Tool. In the XML Oxygen editor tool, all the basic components (such as element, attributes and its types) are available to create a XML structure for which some options must be given and some values must be bind along with the top down automation algorithm. XML data is fetched and is evaluated using XML Parser. Two cases are generated in our proposed model.

• When the dataset is given as input, then the XML document will be generated as Hierarchical tree structure.

• When Document Type Definition (DTD) is given as input, then the XML document will be generated as Document Object Model (DOM) structure.

By parsing the XML documents into the Oxygen XML Editor Tool, a XML query can be efficiently processed and it is also beneficial for the users or browsers in several ways such as reduce the time, increase the index size than the existing techniques.

2.3 Algorithm and analysis:

2.3.1 Algorithm: Top-Down Automation algorithm.

Input: XML Document (doc), Query (q).

Output: Generated Query (Q).

Variables: r, i, j, n, m.

import doc

for i=1; i<=n; i++ // i ranges from 1 to n representing all parent nodes

for j=1; j<=m; j++ // j ranges from 1 to m representing all inner child nodes

if(q==j)

//display=i/j
Q=concat(i,j) //store the merged values of i and j in Q
display(Q)
//if the input query q does not match with Child node j, then it switches to next node in the doc.
end if
end for
end for

Table 1. Notation in Top-Down Automation Algorithm

NOTATION	DESCRIPTION
R	Total number of records
N	Number of parent element
M	Number of child element

2.3.2 Analysis:

An XML Document (doc) is imported and input query (q) is considered as the input and then imported, and generated query (Q) is expected as the output. The variables used here are r- Total number of records; n- Number of parent element, m- Number of child element, i- Child node, j- Parent node. Initially, this algorithm takes XML data set as an input. Algorithm needs to perform looping operation for both child i and parent j node. In this looping operation, each node is compared with input query q for both i & j. If it is satisfied with the above condition, then the generated query Q displays the concatenation of i & j. If it is not satisfied, then it is moved to the next node. The algorithm guarantees that for any node given as an input, it can be computed automatically. The time complexity of our algorithm is O(2mn+1).This is better than the existing techniques.

2.3.3 Advantages:

• It is a commercial tool and easy to develop our techniques.

• Though it is an editor tool, it also has the ability to edit very large amount of XML textual formats.

• Input datasets are quiet simple and this tool makes it easy to configure whereas output document schema conforms to input XML schema file.

• The tool has Graphical User Interface (GUI) and the Oxygen Generator tool is also available as command line tool.

• Oxygen XML Editor and Developer Tool mainly helps in hierarchical Document Transformation.

3 EXPERIMENTS:

Our proposed model runs on Windows system .System requires minimum 2GB RAM. The Java is used to implement the technique. We can execute our proposed model on book data set.

Figure 4: Graph structure of the hierarchical data

Here, the java input output package is imported and the dataset is got as input .The query is given using the xquery and the for clause is used to generate the result of the given expression is bound element wise to the variable and the individual query construct is executed for each element .There are l of semantics already apparent in the grammar. The for clauses here the grammar requires the use of the key word .The list and the nodes are represented using the $ symbol , and the entire nodes that are used in the graph has been represented. Figure 4 shows the Graph structure of the book dataset .Then the size of the list has also been represented specifically the right and the left list sizes are being, separately then the time duration is denoted at the end (i.e.)in how much time the query is being executed.

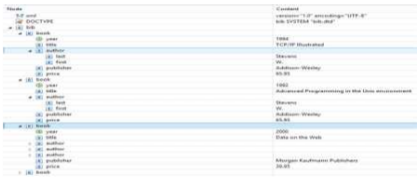

Figure 5 : Top down automatic structure of the book dataset

In the Figure 5 query the hierarchical tree structure is being represented in the top down manner. Here the book is a dataset which has the following properties year ,where the year is considered as the primary key then the title , author ,publisher and the price is listed if the book is clicked then the author consists o the first as well as the last name. For our techniques the time complexity is Time complexity= O (2mn+1). For instance , consider m=2,n=5then the time complexity is O(21)

4 CONCLUSION AND FUTURE WORK

The XML document is imported successfully in the Oxygen tool and the query is generated within expected duration with top down automation technique. We have successfully evaluated the XQuery and generated the hierarchical graph structure. Key factors which results in inefficiency for existing XML keyword search considering algorithms are LList and Baseline Hash Search problems. By using Automated Keyword Query Processing in Hierarchical Structure, the problems are solved and use of child nodes provides satisfiability. We proposed Top-Down Automation algorithm which reduces memory overload and improve the performance of the XML keyword search for the query processing. The future study focus on how to support large indexes as lot of memory is required. The automation tool will not work well if more than one keyword is given in XPath builder for Query processing .

References

[1] Junfeng Zhou, Wei Wang, Ziyang Chen and Jeffrey Xu Yu, Top-Down XML Keyword Query Processing, in IEEE Transactions on Knowledge and Data Engineering, Volume:28, Issue: 5, May 1 2016,pp. 1340-1353.

[2] Yi Chen, Wei Wang and Ziyang Liu, Keyword-based search and exploration on databases, in 2011 IEEE 27th International Conference on Data Engineering, DOI. 10.1109/ICDE.2011.5767958, 16 May 2011, pp. 1380-1383.

[3] W. Wang, X. Wang, and A. Zhou, Hash-search: An efficient SLCA-based keyword search algorithm on XML documents, in Proc. 14th Int. Conf. Database Syst. Adv. Appl., 2009, pp. 496510.

[4] V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava, Keyword proximity search in XML trees,IEEE Trans. Knowl. Data Eng., vol. 18, no. 4, pp. 525539, 2006.

[5] M. K. Agarwal and K. Ramamritham, Enabling generic keyword search over raw XML data, in Proc. 31st Int. Conf. Data Eng., 2015, pp. 14961499.

[6] J. Li, C. Liu, and J. X. Yu, Context-based diversification for keyword queries over XML data, IEEE Trans. Knowl. Data Eng., vol. 27, no. 3, pp. 660672, Mar. 2015.

[7] L. Kircher, M. Grossniklaus, C. Grun, and M. H. Scholl, Efficient structural bulk updates on the pre/dist/size XML encoding, in Proc. IEEE 31st Int. Conf. Data Eng., 2015, pp. 447458.

[8] J. Zhou, Z. Bao, W. Wang, J. Zhao, and X. Meng, Efficient query processing for XML keyword queries based on the idlist index, Int. J. Very Large Data Bases, vol. 23, no. 1, pp. 2550, 2014.

[9] Z. Liu and Y. Chen, XML Query Processing and Query Languages: A survey, J. World Wide Web, vol. 14, nos. 56, pp. 671707, 2011.

[10] Nikita R. Alai, A. S. Vaidya, XML Keyword Query Processing on Disk based Index using Top-Down Approach, Sixth Post Graduate Conference for Computer Engineering (cPGCON 2017) Procedia International Journal on Emerging Trends in Technology (IJETT)