A New Indexing Strategy for XML Keyword Search

Yongqing Xiang, Zhihong Deng, Hang Yu, Sijing Wang, Ning Gao
Key Laboratory of Machine Perception (Ministry of Education), School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
xiangyq@cis.pku.edu.cn

Abstract

With the rapid increase of XML documents on the web, how to index, store and retrieve these documents has become a very popular and valuable problem. At present, there are two normal ways of retrieving XML documents. One is structure-based retrieval; the other is keyword-based retrieval. However, XML keyword search is becoming more and more popular because it is easy to master and manipulate. In XML keyword search system, a key problem is how to store the structure information into XML indices efficiently. At present, Dewey numbers are often used to label XML nodes in XML indices. However, Dewey numbers may lead to redundancy in XML indices. In this paper, we propose a new labeling method called LAF numbers for XML indices and we device a new indexing structure called Two-Layer index for XML keyword retrieval systems. At last, we have conducted an extensive experimental study and the experimental results show that our indexing method achieves better space efficiency than prevailing Dewey-number-based indexing method

Keywords-LAF numbers; Two-Layer; Indexing; XML keyword Search; Dewey numbers

I. INTRODUCTION

XML is a new kind of markup language emerging from 1998. Because XML provides a basic syntax that can be used to share information between different kinds of computers, different applications, and different organizations without having to pass through many layers of conversion, XML has become the standard of data representation, exchanging and sharing on the web. How to store and retrieve XML documents has become a widely discussed issue in the field of data management [1]. XML keyword search is a proven user-friendly way of querying XML documents. It allows users to find the information they are interested in without having to learn a complex query language or needing prior knowledge of the structure of the underlying data [2, 3, 4, 5, 6, 7, 8].

Traditional query processing approaches on relational and XML databases are constrained by the query constructs imposed by the language such as XQL and XQuery. These constraints bring about several critical issues. Firstly, the query language themselves are hard to comprehend and master for common users who are not professional database users. For example, the XQuery language is fairly complicated for common users to grasp. Secondly, these query languages require the queries to be formed based on the complex underlying XML data schemes. These traditional querying methods are powerful but unfriendly. Keyword based XML search is a proven user-friendly way of querying XML documents. It allows users to find the information they are interested in without having to learn a complex query language or needing prior knowledge of the structure of the underlying data. The only thing for the internet users to do is to input some keywords to the search engine. Due to the intrinsic property that XML data (or XML documents) contain content and structure at the same time [2, 3, 4, 5, 6, 7], XML keyword search brings many new challenges. One of the challenges is how to store the hierarchical structure information into XML indices efficiently. To address this problem, we first put forward a new encoding method called LAF number to store the hierarchical structure information with high space efficiency. And then we build a new index structure called Two-Layer LAF inverted index based on LAF number, which considers the XML documents as plain text documents and semi-structured documents respectively.

To summarize, we make the following technical contributions in this paper:

- We put forward a new encoding method called LAF numbers for XML documents. LAF number has two obvious advantages to compare with Dewey encoding, firstly, the length of LAF number for every XML node is constantly 3. Secondly, if we want to compare the size of two LAF numbers, we only compare the size of their level order sequence number, the complexity of which is O(1).
- We build a new index structure called Two-Layer LAF inverted index. We save the attributes of describing XML documents as plain text documents such as the length and size of XML documents, the URL of XML documents and so on into the first layer index and then we save the hierarchical structure information and attributes of XML elements into the second layer index. Our experiments show that this new index structure can decrease the redundancy of the XML indices and improve the space performance of XML retrieval systems.

The remainder of the paper is organized as follows. Section 2 introduces background of this paper, our motivation and related work. Section 3 introduces LAF number, a new kind of encoding scheme. Section 4 presents a new indexing structure called Two-Layer LAF inverted index, which is based on LAF number. Experimental results are presented in Section 5. Section 6 summarizes our study and points out some future research issues.

978-1-4244-5934-6/10/$26.00 ©2010 IEEE

```xml
<?xml version="1.0" encoding="utf-8"?>
<proceedings>
    <paper id='1'>
        <title>Efficient LCA based Keyword Search in XML Data</title>
        <institution>
            University of California, San Diego
            <authors>
                <name>Yu Xu</name>
                <name>Yannis Papakonstantinou</name>
            </authors>
        </institution>
        <introduction>
            This paper by Xu and Yannis propose an efficient algorithm
            called Indexed Stackto find answers to keyword queries based
            on XRank semantics.
        </introduction>
    </paper>
    <paper id ='2'>
        <title>
            Efficient Keyword Search for Smallest LCAs in XML Databases
        </title>
        <authors>
            <name>Yu Xu</name>
            <name>Yannis Papakonstantinou</name>
        </authors>
        <introduction>
            The core contribution of this paper by Yu and Papakonstantinou,
            the Indexed Lookup Eager algorithm,exploits key properties of
            smallest trees in order to outperform prior algorithms by
            orders of magnitude when the query contains keywords with
            significantly different frequencies
        </introduction>
    </paper>
</proceedings>
```

Figure1. XML document for proceedings

II. BACKGROUND AND MOTIVATION

A very important difference between XML documents and traditional plain text documents is that XML documents are semi-structured and can describe information more precisely. So how to save the hierarchical structure information in XML documents into indices and make the best use of this structure information to improve the quality of XML keyword search are meaningful and challenging problems. One effective way to save structure information is to label XML elements with numbers. Dewey number is a very popular XML encoding method at present. However, Dewey number has two obvious disadvantages: firstly, the length of Dewey number for an XML element increases with the depth of this element in the XML tree, which may cause indexing redundancy[2, 3, 12, 13, 15], for example, in Figure 2, the depth of the root is 1, and the length of its Dewey number is 1, but the node "0.0.1.1.1.0" whose depth is 6, the length of its Dewey number is 6; secondly, many query algorithms based on Dewey numbers need to sort elements according to the big and small of Dewey number[2, 3, 4, 5, 6, 7, 12], the complexity for comparing two Dewey numbers is O(n)(here n is the average length of the two Dewey numbers), which will be unacceptable in processing of large scale XML documents set, for instance, if we want to compare the big and small of node "0.0.1.1.0.0" and "0.0.1.1.1.0", the min number of comparing times is 6. In order to address these two problems, we introduce a new encoding scheme called LAF number in section 3. Our experiments show that our method has good space performance.

As is known to all, inverted files are the most popular index structure for plain text search. Besides, signature files are also very prevalent. In XML keyword search, Dewey number based inverted files (simple for Dewey Inverted files) are the most popular index structures. [2, 3, 4, 5, 6, 7, 8, 12] are all based on

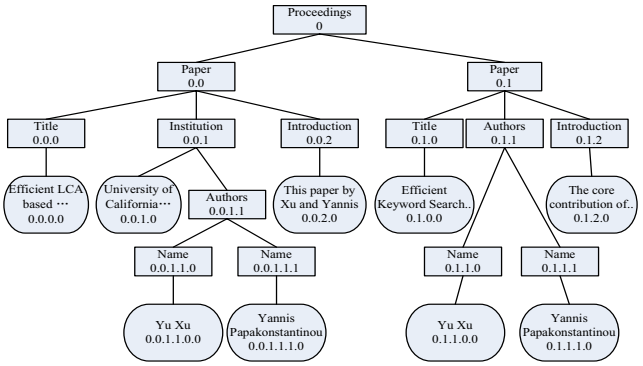

Figure2. An XML tree Labeled with Dewey numbers

Dewey inverted files. However, we know XML document is a special kind of text documents and XML keyword search mostly focused on XML elements but not full documents. Therefore, for each XML document, there are two kinds of attributes for it: the attributes about the XML documents as plain text documents, such as the length of the document and the URL of the document; and the attributes about XML elements and inner structure, such as Dewy numbers and the length of element. If we store these two kinds of attributes in just one layer inverted files, there must be much redundancy and this kind of inverted files are also adverse to the query processing algorithms. To address this problem, We devised a new index structure called Two-Layer LAF inverted index, we save the attributes to describe XML documents as plain text documents such as the length and size of XML documents, the URL of XML documents and so on into the first layer index and we save the hierarchical structure information into the second layer index. Our experiments show that this new index structure can decrease the redundancy in the XML indices.

III. LAF NUMBER

This section will introduce a new labeling strategy call LAF number for XML documents. At present, the most popular labeling method is Dewey number. In this section, we firstly introduce the Dewey number. Then we introduce our new encoding strategy called LAF number.

A. Dewey Encoding

Dewey number is a kind of very popular labeling method for XML documents. [1] has introduced Dewey order numbers to store and query XML documents for the first time. With Dewey order, each node is assigned a vector that represents the path from the document's root to the node. Each component of the path represents the local order (With local order, each node is assigned a number that represents its relative position among its siblings) among the sons of its parent node, Figure 2 is an XML document tree labeled with Dewey number for the XML document showed in Figure 1. It is easy to find that Dewey number has an important feature: It is easy to get the common ancestor node and judge the relationship of arbitrary two nodes in XML trees. In fact, the common ancestor node's Dewey number is the longest common prefix of these nodes' Dewey numbers. To give an example, Node 0.0.1.1.0.0 and Node

0.0.1.1.1.0 and Node 0.0.1.0 are three nodes in Figure 2, the longest common prefix of them is 0.0.1, so Node 0.0.1 is their common ancestor.

Nevertheless, Dewey number has two obvious disadvantages, because every sub-node should keep its parent node's Dewey number as its prefix, which may cause redundancy and increase the space complexity of XML indices; Moreover, the query processing algorithm based on Dewey encoding may cause low efficiency.

B. LAF numbers

In order to overcome the disadvantages of Dewey numbers, we put forward a new labeling method for XML documents called LAF number, which is short for Level order and Father number. LAF number is built on level-order tree traversal. Level-order traversal is a kind of global traversal strategy. When traversing an XML tree by level-order, we first visit the root of the tree, then, we traverse the tree level by level until all of the nodes in the tree are visited. We can label the XML tree by its sequence number of level-order traversal as Figure 3. So every node in XML tree has an only sequence number as its level order number. The level order sequence of XML tree in figure 3 is A, B, C, D, E, F, G, H, I, J, so the level order encoding of these ten nodes is 0, 1, 2, 3,4,5,6,7,8,9.

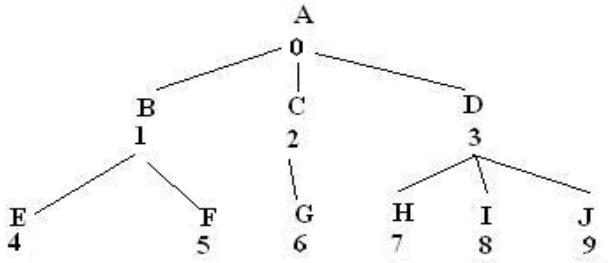

Figure3: Level order Encoding on XML tree.

LAF number is a new labeling scheme for XML tree. Every node in the XML tree has an unique LAF number. Every LAF number is made up of three components, which are the level order sequence number of current node, the level order sequence number of current node's father node and the level number(or depth) of current node. Especially, if current node is the root node, its father node's level order sequence number is set as -1 because the root node has no father node. The level number begins from 0, and it increases level by level. The structure of LAF encoding is illustrated as Figure 4.

Like Dewey numbers, LAF numbers can also be stored in a vector with only three dimensions. In Figure 3, It is easy to know that Node A's LAF number is 0.-1.0, because node A's level order sequence number is 0, so the first dimension of LAF vector is 0; and node A is the root of XML tree, so its father node's level order sequence number is set as -1; and node A is at the first level of XML tree, so its level number is 0. By parity of reasoning, we can label the tree in Figure 2 as Figure 5 using LAF numbers.

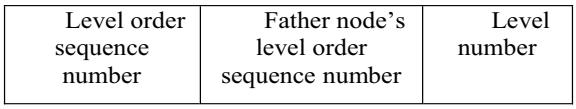

Figure4: the structure of LAF encoding

We can know that the LAF numbers for nodes in XML tree are different because their level order sequence numbers are

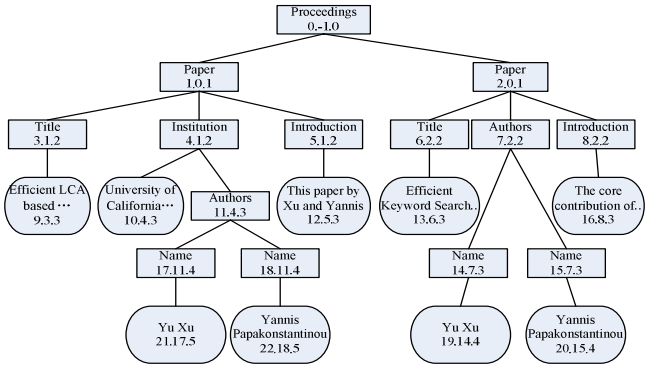

Figure5: LAF encoding example

different. On the other hand, given the LAF numbers of nodes in an XML tree, it is easy to build an only XML tree. That is to say, LAF number is stable and reversible.

IV. TWO LAYER INDEX

Inverted index (also referred to as postings file or inverted file), is a kind of indexing structure storing a mapping from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents. Now it is the most popular data structure used in document retrieval systems. This section will introduce Dewey encoding based inverted index firstly, then we propose a new index structure based on LAF encoding.

A. Dewey Inverted Index

Dewey inverted index, also referred to as Dewey inverted list, is now the most popular index structure used in XML document retrieval systems, such as XRANK [2] and XKSearch [3]. The inverted list for a keyword K contains the Dewey numbers of all the XML elements that directly contain keyword K. To handle multiple documents, the first dimension of each Dewey number can be set as the document ID..

Dewey number based inverted index is a kind of effective index for XML documents, however, because every node restore its father's Dewey number as its prefix, which can cause redundancy and increase the space complexity of XML indices. Can we design a new index structure, which has better time and space performance, for XML document? The answer is yes, following we will introduce the new index structure.

B. Two-Layer LAF Inverted Index

As is know to everyone, the biggest difference between retrieval on plain text and XML document is that XML

documents contain structure information except content and the granularity of XML retrieval is XML element but not an entire document. In fact, every XML document can be seen as a plain document, and can be also seen as a semi-structure document. That is to say, XML documents have two kinds of attributes, the attributes about it as a plain document and the attributes about the inner elements and structure. Can we divide these two different kinds of attributes in XML indices to speed up the efficiency of XML retrieval? Yes, we can build a two-layer inverted index to support these two kinds of attributes at the same time.

Two-layer inverted index includes two parts. The first part is the first layer index, which is similar to common inverted index. The second part is the LAF number table (simple for LAF table). The LAF table can be seen as the second Layer index.

LAF table is a table to store all the LAF numbers in an XML tree. If an XML tree has n nodes, its relevant LAF number table has n entries. LAF numbers in LAF table are sorted according to their level order sequence number. LAF table is very important here, because it stores the structure information of the XML tree. Table 1 is an example of LAF table for the XML tree in Figure 3.

The first layer index is an inverted index built on XML document. The first layer index is similar to common inverted index except that the first layer index should contain a list of level order sequence numbers of nodes containing current keyword occurs in current document. The sequence number list is related to the LAF table of current XML document, the level order sequence number in the list is one-to-one related to that in the LAF table. Figure 7 shows the structure of the first layer index.

Table1: LAF encoding table

Level order number	Father node's level order number	Level number
0	-1	0
1	0	1
2	0	1
3	0	1
4	1	2
5	1	2
6	2	2
7	3	2
8	3	2
9	3	2

document number	Term frequency in document	Level order sequence number list

Figure6: Structure of First Layer Index

Figure 8 is an example of two-layer LAF inverted index to store keywords "Yu" and "Yannis" in Figure 1, whose document number is set as 1000. The level order sequence number of nodes containing these two keywords can be respectively gotten from the XML tree labeled with LAF numbers in Figure 5.

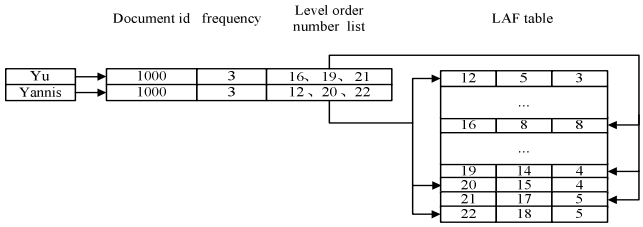

Figure7: Two-Layer LAF inverted Index

V. EXPERIMENTAL STUDY

We now experimentally evaluate the techniques presented in this paper. Firstly, we introduce the experimental environment and the XML data set used in this paper. Secondly, we compare Two-Layer LAF inverted index and traditional Dewey encoding based inverted index in view of space efficiency.

A. Experimental Setup

We run our experiments on XML data set from Wikipedia, which is the standard data set used by INEX2009 (The Initiative for the Evaluation of XML Retrieval), which is an international campaign involving more than fifty organizations worldwide. We do our experiments on four different data sets,

Table2: Details of the Data Sets

Number of documents	Data set size	Total elements	Total keywords
400	7524KB	210180	399386
800	15107KB	411243	537881
1200	22485KB	618147	797175
1600	28889KB	796850	1016924
2000	36112KB	994408	1283765

which include 400, 800, 1200, 1600 and 2000 XML documents respectively and are about 9505KB, 18775KB, 27775KB and 37001KB respectively. Table 2 shows details of the data sets we used. We choose XML data from Wikipedia because these data sets are made up by many small pieces of XML documents. Average size of each document is about 19KB.

We build both traditional Dewey number based index and Two-Layer LAF inverted index, which are both stored in Berkeley DB . We used C++ for our implementation, and used a 1.8GHz Pentium Dual processor with 2GB of main memory and 160GB of disk space.

B. *Space Performance*

From Table 3 we can see that the total size of Two-Layer LAF inverted index for each data set is a little lager than the original data set. Figure 9 shows the space requirements for Two-Layer LAF index and Dewey inverted index. We can see that the space complexity of Dewey inverted index is about triple the complexity of Two-Layer LAF index. There are two reasons about this difference. The first reason is that the length of Dewey number increases as the increase of the depth of element while the length of LAF number for each element is constantly 3. In Figure 2 element '0.0' whose depth is 2 needs two integers to record its Dewey number, but the element '0.0.1.1.0.0' whose depth is 6 needs six integers to record its Dewey number. The second reason is that Dewey inverted index is only one layer index which is built on elements but not on documents, so many document features (such as document id, document length and document URL and so on) are stored repeatedly many times, however, these features are only stored once in Two-Layer LAF inverted index.

Table 3: Compare the space performance

Original Data set size	Size of First layer index	Size of second layer index	Total size of Two-Layer inverted index	Size of Dewey Inverted Index
7524KB	7298KB	2385KB	9683KB	37670KB
15107KB	14980KB	4641KB	19621KB	49593KB
22485KB	21907KB	6991KB	28898KB	73421KB
28889KB	28168KB	8982KB	37150KB	93786KB
36112KB	35676KB	11182KB	47858KB	117993KB

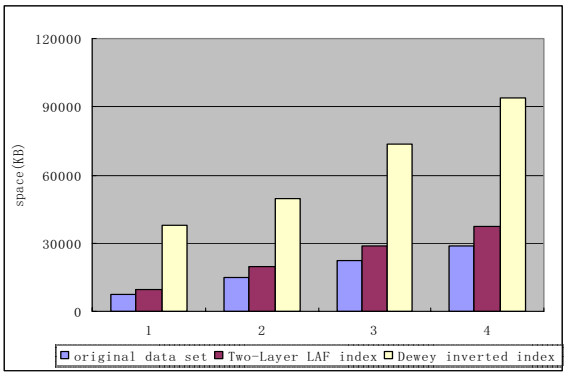

Figure 8: Space Performance Comparison

VI. CONCLUSIONS

In this paper, we have investigated the problems of keyword search based on Dewey encoding over XML documents. We found that Dewey number based keyword search system has two disadvantages: (1) It is low efficiency to store Dewey numbers because the length of Dewey number for an element in XML document is proportional to its depth in the XML tree; (2) It is low efficiency to compare two Dewey numbers, if we want to get the lager Dewey number, we need to compare each dimension of their Dewey number vectors. In order to solve these problems, we proposed a new encoding schema called LAF for XML elements. Then, we designed a new index structure called Two-Layer LAF inverted index. We have implemented the proposed method and the expensive experiment results showed that our new method has better space performance.

In the future, we will build an XML keyword retrieval system base on the method proposed in this paper.

ACKNOWLEDGMENT.

This work is partially supported by Supported by the National High Technology Research and Development Program of China (863 Program) under Grant No. 2009AA01Z136 and the National Natural Science Foundation of China under Grant No.90812001.

REFERENCES

[1] Tatarinov I, Viglas S, Beyer K, et al. Storing and querying ordered XML using a relational database system. In Proceedings of SIGMOD'02. Madison, Wisconsin: ACM, 2002

[2] Guo L, Shao F, Botev C, et al. BrickNet: XRANK: Ranked keyword search over XML documents [C]. In Proceedings of ACM SIGMOD'03. San Diego, CA: ACM, 2003: 16-27

[3] Yu Xu, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases .In Proceedings of SIGMOD'05. Baltimore, Maryland: ACM, 2005

[4] Yu Xu, Papakonstantinou Y. Efficient LCA Based Keyword Search in XML Data. In Proceedings of CIKM'07. Lisboa: ACM, 2007

[5] Yu Xu, Yannis Papakonstantinou. Efficient LCA based Keyword Search in XML Data.Nantes, France. EDBT'08, March 25–30, 2008

[6] Feng Shao, Lin Guo, Chavdar Botev, et al. Efficient Keyword Search over Virtual XML Views. In Proceedings of VLDB'07

[7] Li Guoliang, Feng Jianhua, Wang Jianyong. Effective Keyword Search for Valuable LCAs over XML Documents. In Proceedings of CIKM'07. Lisboa: ACM, 2007

[8] Chong sun, Chee-Yong Chan, Amit K. Goenka. Multiway SLCA-based Keyword Search in XML Data. In Proceedings of WWW'07

[9] Yunyao Li, Cong Yu, H. V. Jagadish. Schema-Free XQuery. Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

[10] Ziyang Liu, Jeffrey Walker, Yi Chen. XSeek: A Semantic XML Search Engine Using Keywords. In Proceedings of VLDB'07

[11] S. Soltan, A. Zarnani, R. AliMohammadzadeh, and M. Rahgozar. IFDewey: A New Insert-Friendly Labeling Schema for XML Data. PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 13 MAY 2006 ISSN 1307-6884

[12] Kong Lingbo, Tang Shiwei, Yang Dongqing. Layered Solution for SLCA Problem in XML Information Retrieval. Journal of Software, 2007, 18(4): 919-932

[13] William M. Shui, Franky Lam, Damien K. Fisher, Raymond K.Wong. Querying and Maintaining Ordered XML Data using Relational Databases. Conferences in Research and Practice in Information Technology, Vol. 39. 2005

[14] TORSTEN GRUST, MAURICE VAN KEULEN and JENS TEUBNER. Accelerating XPath Evaluation in Any RDBMS. ACM Transactions on Database Systems, Vol. V, No. N, Month 20YY, Pages 1-40

[15] Patrick O'Neil, Elizabeth O'Neil, Shankar Pa, Istvan Cseri, Gideon Schaller. ORDPATHs: Insert-Friendly XML Node Labels. SIGMOD 2004, June 13–18, 2004, Paris, France.