Novel Scheme for Labeling XML Trees based on Bits-masking and Logical Matching

Taher Ahmed Ghaleb, Salahadin Mohammed
Information and Computer Science
King Fahd University of Petroleum and Minerals Dhahran
Eastern province, KSA
{g201106210, adam}@kfupm.edu.sa

ABSTRACT: *The eXtensible Mark-up Language rapidly has become a very powerful standard for the data exchange. Labeling schemes have been introduced to optimize data retrieval and query processing on XML database documents. This is done by providing labels that hold information about XML tree nodes. In this paper we introduce a novel labeling scheme XDAS whose labeling technique is inspired by IP addressing and subnetting technique used in computer networks. This technique is used when dividing a network into several sub-networks. Each sub-network is assigned a subnet mask that helps in identifying the parent network. So, this labeling scheme treats XML documents as a network with sub-networks and assigns labels for XML tree nodes using the masking technique. Experimental results show that XDAS, when compared to Dewey and Range labeling schemes, provides an efficient label size, disk space required to store labels and matching time required to identify relationships between nodes.*

Keywords: XML, Labeling Scheme, XML Query Processing, XML Storage, XML Tree

Received: 19 May 2013, Revised 29 June 2013, Accepted 3 July 2013

1. Introduction

Nowadays, many researches have been introduced to manage XML documents, provide flexible data storage and facilitate and accelerate query processing. To efficiently do so, researchers have proposed many techniques to manage the tree-structure of XML document content in a way data can be retrieved faster. Providing a smart label which holds useful information with efficient size, storage space and query execution time is still a hot topic.

Labeling, is a method by which each node in the XML tree is given a label which holds information about that node, such as level, order or unique identifier in a way we can recognize its position and also its relationship with other nodes. Each node can be a parent, ancestor, child, descendant, or sibling of another node in the XML tree. The most important information which a labeling scheme should provide robustly is the Parent-Child (PC), Ancestor-Descendant (AD) and sibling relationships with minimum label size, storage space, and querying time.

The main contribution of this paper is proposing a novel labeling scheme, XDAS (clarified in Section III). XDAS uses a technique applied in computer networks called IP addressing and subnetting. This technique supports XDAS to do logical matching, using logical *AND* operator, in order to identify the A-D, P-C and sibling relationships between nodes. As it is known, logical operations are more efficient than range comparison operations and string matching operations.

The rest of the paper is organized as follows: Section II reviews some work done in the literature about labeling XML trees, and briefly introduces the idea of each labeling scheme. Section IV demonstrates and analyzes the experimental results provided by our scheme compared with the common labeling schemes, Dewey and Range, that many recent labeling schemes are based on. Finally, Section V concludes the paper and suggests the possible future work.

2. Related Work

Existing labeling schemes for XML trees usually categorized as Range based and Prefix based labeling schemes. Range based labeling schemes identify each node with a label that consist of start number, end number and level according to the pre-order traversal of the XML tree [9, 10, 11, 12]. Prefix based labeling schemes store information of ancestors labels in the labels of their descendants using a delimiter [1, 2, 5, 6, 7, 8]. Recently, it is common to see a hybrid labeling schemes which combine the advantages of two approaches [14].

Range labeling scheme, of the form (23, 44, 3), gives a given node a label of the form (*StartNo, EndNo, Level*). The strength of this kind of labeling scheme is the ability to determine the PC and AD relationships between two nodes using arithmetic range comparison operations. On the other hand, sibling relationship cannot be identified from labels themselves. This labeling scheme is not applicable for dynamic XML documents since all nodes must be relabeled in case of insertion of a new node or a subtree occurs.

J. H. Yun and C. W. Chung [11] proposed a range-based labeling scheme with the nested tree structure which eliminates the limitations and takes advantage of the previous interval-based node labeling schemes. Their new approach supports XML data updates with almost no node relabeling. Also, the integer comparison operation is changed to the integer list comparison operation.

In Prefix based labeling scheme, of the form (1.3.22.4), a given node *X* is a descendant of a node *Y* if the label of *Y* is the prefix of the label of *X*. Its advantage is that all structural information of relationships can be captured by looking at labels only. These structural information requires large storage space. On the other hand, it is efficient in case of identifying the *AD*, *PC*, and sibling relationships between tree nodes via string matching operations. Dewey ID [6] and Extended Dewey [7] are examples of prefix based labeling schemes that are not capable for dynamic XML documents since they requires relabeling of nodes if insertion occurs. Prefix based labeling schemes started with using only integers to represent labels, but afterwards, a combination of integers and alphabets have been used to represent node labels. In order to provide dynamic Dewey, new approaches was proposed called sibling labeling scheme[15] and DDE [16]. Sibling labeling scheme approaches requires relabeling of two nodes at most when new node is going to be inserted. Whereas, DDE avoids relabeling completely.

ONeil et al. [8] introduced OrdPath, which is a dynamic labeling scheme different from Dewey but of the same order. Node labels are assigned by the Dewey order except that it does not use even and negative integers in the initial labeling, of the form (1.5.7.9). It reserves even and negative integers for later insertions into an existing tree. Also it stores the label of each node as an encoded binary representation. But, the problem with OrdPath occurs when the size of the codes overflow, which means it must re-label all the existing nodes. The overflow problem [13] is also suffered by LSDX [3] and SCOOTER [5] labeling schemes. Thus, these labeling schemes are not preferred when XML documents have deep trees.

H. Ko and S. Lee [4], proposed IBSL "*Improved Binary String Labeling*" as a novel labeling scheme. Their labeling scheme uses Dewey order but using bit-strings of the form (101.1.100.111), with full support for update without recalculation or relabeling. On the other hand, this scheme does not use the characteristics of binary numbers to do bits-matching, however, it uses string matching in order to identify the relationships between nodes.

B. G. Assefa and B. Ergenc [1] proposed a dynamic OrderBased labeling scheme which optimizes the label size of every level. Their scheme provide efficient querying time when compared to Com-D [2]. Also, it has an efficient label sizes with efficient storage requirement when compared to LSDX [3].

3. Proposed Work

In computer networks, many methods and efficient algorithms have been developed and improved in order to provide best network performance. One approach used in computer networks is dividing a network into several sub-networks called IP addressing and subnetting. A subnet mask is assigned to each sub-network in order to preserve information about their addresses of parent networks [17]. Also, it is easy to add a new host to a specific sub-network and assign a new IP address to it without affecting other hosts. In addition, sub-networks can be joined using routers that make bridges between different sub-networks. Furthermore, removing a host from a sub-network does not affect the balancing of that sub-network.

In this paper, we propose using IP addressing and subnetting approach, with some adaptation, as a labeling scheme for XML tree nodes where instead of having an IP address, we have a node label. Our proposed labeling scheme is called XDAS, short of ***X**ML **D**ocuments **A**ddressing and **S**ubnetting*. XDAS label is formed as <*Level, Number*>. It provides an efficient time needed to identify the P-C, A-D and sibling relationships and efficient label size with less storage required to store labels. This is due to that it uses binary numbers with logical operation to identify the relationship between nodes. XDAS performance is slightly not affected by the depth of the XML tree.

3.1 XDAS labeling shceme

Labeling technique used in XDAS is inspired by IP addressing and subnetting used in computer network with some. To divide a network into several sub-networks, then the addresses of the sub-networks is going to be the number of parent network address followed by sequence numbers associated to them. Each sub-network also contains a mask that helps in identifying their parent network address.

In XDAS, we use the idea of masking technique used in subnetting but in different way. Figure 1 shows how to generate labels using masking technique. For example, if we have a node number equals to $(16)_{hex}$ which represents the binary number 10110, and we assume that the number of its children nodes is less than seven. Then the labels of its children is going to be generated as follows: **001**10110, **010**10110, **011**10110, **100**10110 and so on. If the number of children is more than seven, then we have to use more than 3-bits to represent their sequence numbers.

Algorithm 1 gives a detailed description of the operation used for labeling nodes using XDAS. In XDAS, *Breadth First Traversal* is implemented to visit all XML document nodes in level order. Each node is represented as structure of (*Node_Number, Node_Level, Parent_Number*) and is assigned a label of the form <*Level, Number*> where *Number* is a unique number generated using the masking technique and *Level* is the level where that node is exist. The labeling scheme starts generating labels for the nodes in the first level as follows : (1,001), (1,010), (1,011), (1,100), (1,101) and so on. When no nodes available in the first level, the labeling scheme goes for the second level. Nodes at that level will be assigned labels which contains sequence numbers starting from 1 concatenated with their parent numbers, i.e., the label of the first child of the first parent is **2,001**; and after concatenating it with its parent label it is going to be **2,001**001, the second child will have the number **2,010**001 and so on. Number of zeros that padded to the left of node label depends on the maximum bits used to represent nodes' numbers at a given level. All labels are stored in disk by representing the *Level* using only one byte, and the *Number* is represented in hexadecimal. For example, a node with the label 3,01010110 has the label 3,56 where 56 is the hexadecimal representation of 01010110.

Each level will have its own mask which is represented as by bits of 1s. Number of ones in a level mask is equal to the maximum number of bits used to represent labels at that level. For example, if the maximum *Number* at a given level is equal to 01011100101, then the maximum number of bits is eleven, so the mask must be 11111111111. The list of masks is stored in separated file in disk.

3.2 Identifying relationship between nodes using XDAS

Labels generated using Algorithm 1 are efficient labels because they carry useful information about the relationships between nodes. The relationship between two nodes is identified using logical *AND* operator. Algorithm 2 illustrates the bits-matching technique in which the relationship between two nodes can be identified. Comparison using logical operators is well-known to be hardware based operation which is faster that integer comparison and string matching mechanisms. For example, from the XML tree represented in Figure 2, if we take the label of the node *Book* 1, 2 as X_{node} and *FName* 3,26 as Y_{node}. Now by applying Algorithm 2, where the mask of each level is given in Figure 3. $Y \cap X_{level\_mask} = 100110 \cap 11 = 10$ which is equal to X_{number}. This means that X_{node} is either a parent or ancestor of Y_{node}. So, $X_{level} + 1$ equals to $2 \neq Y_{node}$, which means that X_{node} is an ancestor of Y_{node}. On the other hand, if we take the label of the node *Title* 2, 9 as X_{node} and *LName* 3,17 as Y_{node}. Then $Y \cap X_{level\_mask} = 10111$

$\cap$ 1111= 111 which is not equal to X_{number}. This means that X_{node} is neither a parent nor ancestor of Y_{node}. Hence, 10111 $\cap$ 1111 $\neq$ 1001 $\cap$ 1111 $\rightarrow$ 111 $\neq$ 1001, and also $X_{level} \neq Y_{level}$, which also means that X_{node} is not sibling of Y_{node}.

Algorithm 1. Generating labels

```
Input an XML document tree
Output computed labels for tree nodes
Begin
    01: lastLevel ← 0
    02: lastparent ← 0
    03: enqueue (XMLNode)
    04: while Queue.count > 0 do
    05:     currentnode ← dequeue( )
    06:     if lastLevel ≠ currentNodeLevel then
    07:         lastLevel ← lastLevel + 1
    08:     //max. no. of bits used at that level is stored as a Mask,
    09:     //e.g. 4 is stored as 1111
    10:     end if
    11:     if currentNodeParent ≠ lastparent then
    12:         number ← 1
    13:         lastparent ← currentNodeParent
    14:     end if
    15:     label ← currentNodeLevel number lastparent
    16:     // store label to disk
    17:     number ← number + 1
    18:     foreach ChildNode in currentnode.ChildNodes do
    19:         ChildNodeLevel ← currentLevel + 1
    20:         enqueue(ChildNode)
    21:     end for
    22: end while
End.
```

Algorithm 2. Identifying Relationship between nodes

```
Input two nodes X, Y
Output the relationship between X and Y
Begin
    01: Mask ← Mask_of_X_level
    02: if Y ∩ Mask = X then
    03:     if Level (Y) = Level (X) + 1 then
    04:         return Parent
    05:     else
    06:         return Ancestor
    07:     end if
    08:     else if (Y ∩ Mask = X ∩ Mask)
    09:             & (Level (Y) = Level (X) ) then
    10:         return Sibling
    11:     else
    12:         return other
    13:     end if
End.
```

Binary System :128 64 32 16 8 4 2 1

Child 1 No. : 0 0 1 **1 0 1 1 0** – –> 36_{hex}

Child 2 No. : 0 1 0 **1 0 1 1 0** – –> 56_{hex}

Child 3 No. : 0 1 1 **1 0 1 1 0** – –> 76_{hex}

Child 4 No. : 1 0 0 **1 0 1 1 0** – –> 96_{hex}

Parent Mask Length = 11111, Level Mask = 11111111

Figure 1. Masking technique of XDAS

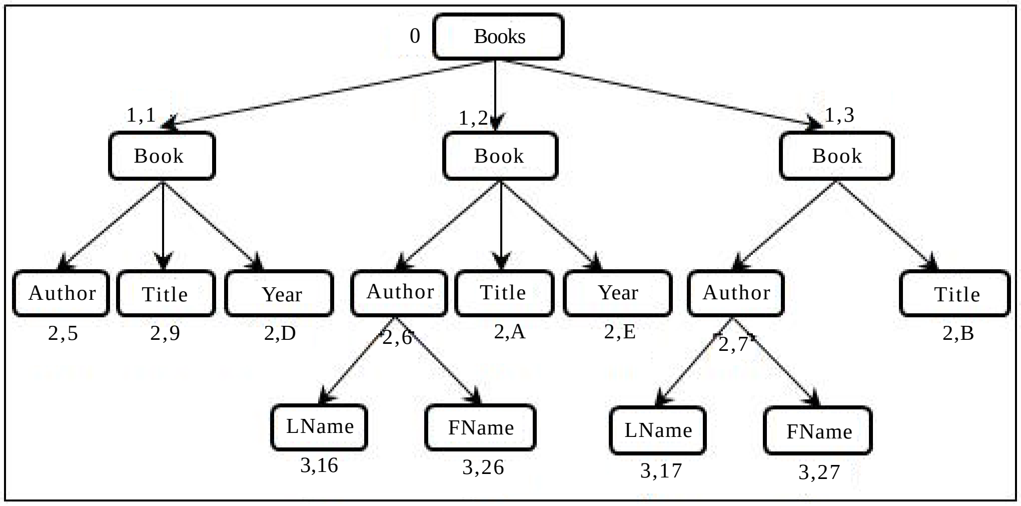

Figure 2. XDAS labels of an XML document

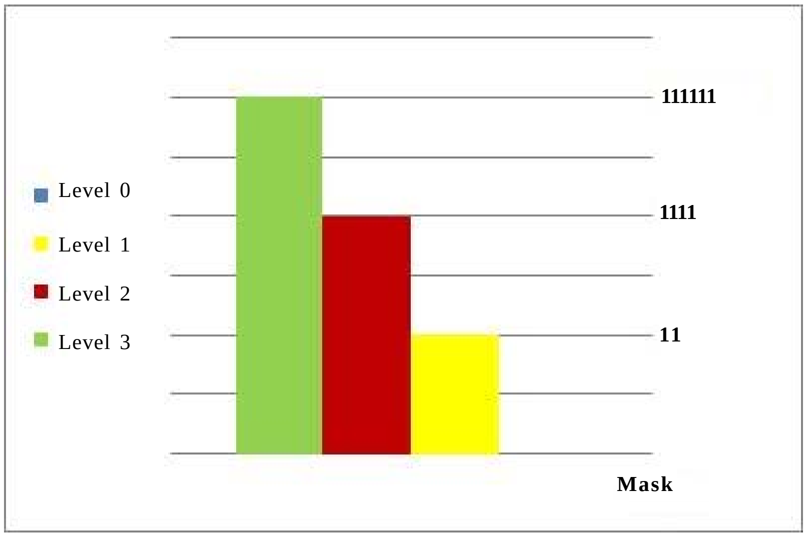

Figure 3. Mask of each level of the XML tree shown in Figure 2

4. Exprемental Results and Analysis

We experimentally compare the performance of XDAS with that of two common labeling schemes: Dewey and Range or Interval. The three labeling schemes were implemented in Visual C# 2010 and C language. Implementation in Visual C# includes generating

Dataset	Topic	Max/ average fan out	Max/average depth	# of nodes
D1	DBLP	328, 858/65, 930	6/3	3, 332, 130
D2	XMark	25, 500/3, 242	12/6	1, 666, 315
D3	Treebank	56384/1623	36/8	2, 437, 666

Table 1. Datasets Used For Testing

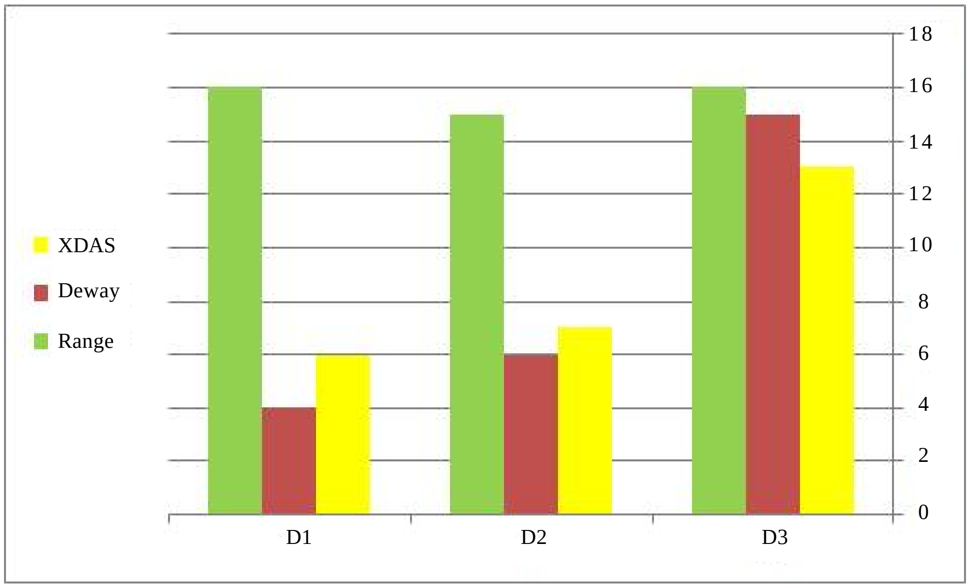

Figure 4. Average label size in each dataset using the three labeling schemes

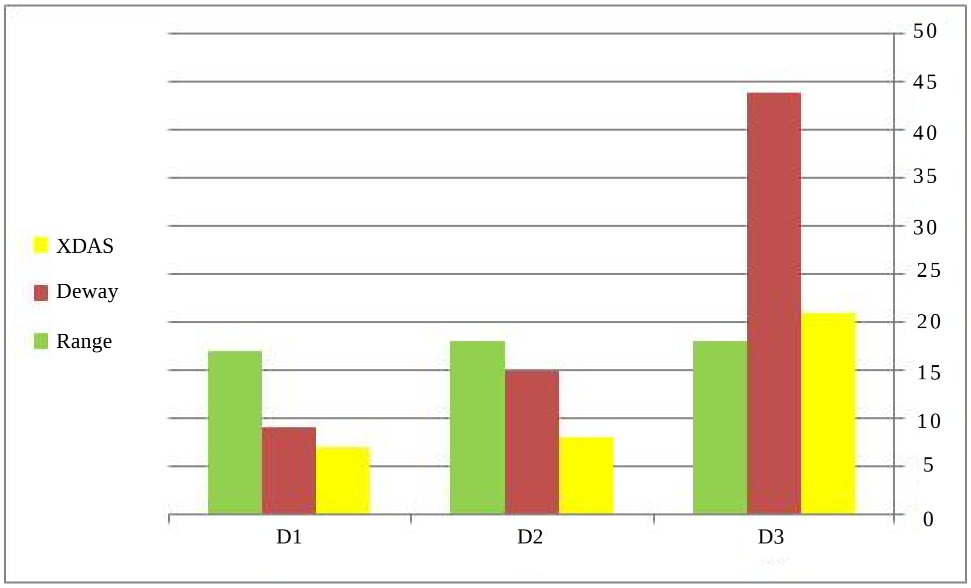

Figure 5. Max label size in each dataset using the three labeling

labels and calculating labels sizes using the three labeling schemes. Implementation in C language includes the binary matching which is used to identify the AD, PC and sibling relationships between nodes. We select three real-world XML documents listed in Table 1, and they are available for free online. These three different datasets have different depth, fan out and total number of nodes.

The experiments were applied on a 2.0 GHz Intel Core 2 Duo processor with 2 GB of RAM running 32-bit Windows 7 Ultimate.

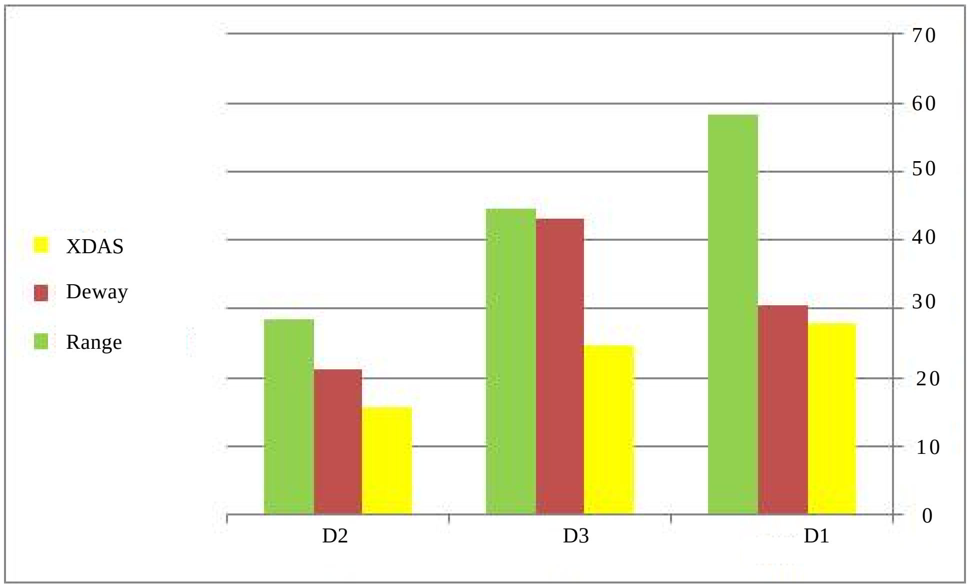

Figure 6. Space required to save labels using XDAS, Dewey and Range-based labeling schemes

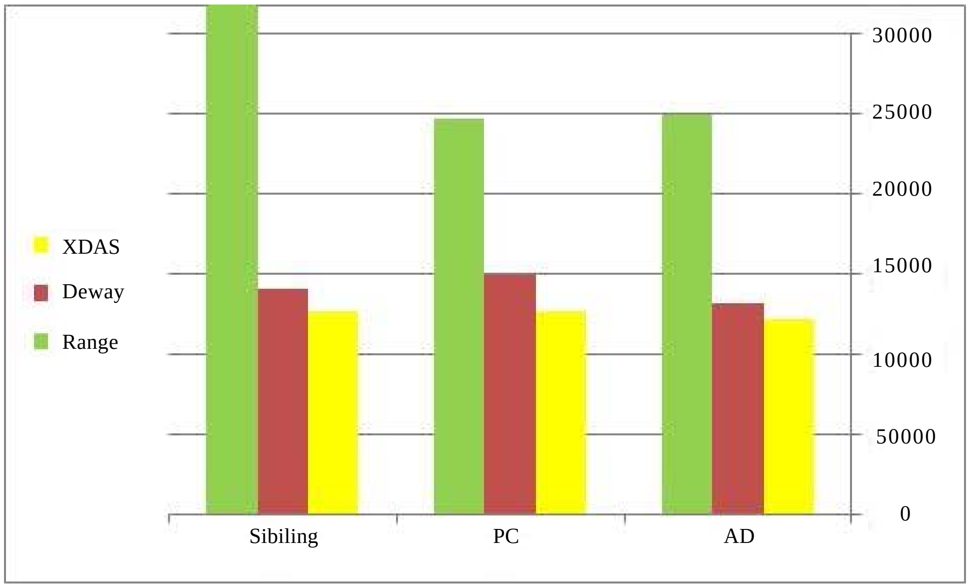

Figure 7. Time needed to determine the relationships between nodes' labels using XDAS, Dewey and Range-based labeling schemes

4.1 Label Size

Figure 4 shows the average label size obtained after implementing the three labeling schemes XDAS, Dewey and Range on the three datasets shown in Table 1 As it is shown, Range labeling schemes average label size is nearly the same among the three datasets, which means that it is almost not affected by the depth of XML tree. On the other hand, Dewey average label size depends on the depth of the XML tree.

Average label size of XDAS is slightly affected by the depth of XML tree and seems to be closer to the Dewey's. But Figure 5 demonstrates the XDAS max label size is more efficient than Dewey and Range.

4.2 Space Requirements

Figure 6 shows the space required to store labels of the three datasets using the labels generated by XDAS, Dewey and Range. As it is demonstrated, XDAS provides efficient disk space required to store labels.

4.3 Query processing performance

Randomly, from labels generated by each labeling scheme, two files containing labels of an XML document are selected, one file contains 100 labels and the other contains 300,000 labels. We ran a set of binary matching operations to identify the AD, PC and sibling relationships of those labels using XDAS, Dewey and Range-based labeling scheme and in each labeling schemes, we take the average running time. As shown in Fig.7, efficiently, XDAS needs less time than needed by Dewey and Range labeling schemes.

Many recent prefix-based labeling schemes which are based on Dewey structure, use compression and decompression techniques in order to minimize the label size and space requirement, but as a result query processing time suffers. On the other hand, some of them tried to play with Dewey structure and shrink it, but consequently, in order to process queries, they consume much time to do it recursively. Also, no labeling scheme till now used logical operation to do matching in which the relationships between nodes can be identified, even IBSL which uses binary numbers for nodes' labels.

5. Conclusion and Future Work

In this paper, we proposed XDAS, a novel labeling scheme for labeling XML documents. The proposed scheme uses IP addressing and subnetting technique used in computer networks for dividing a network into several sub-networks, adapts it, and then applies it for labeling XML documents. Experimental results done in this work is promising and show that the performance of XDAS, when compared to Dewey and Range labeling schemes, is efficient regarding label size, space requirements, and query processing. These results encourage us to improve this scheme in the future to make it applicable for dynamic XML documents in order to enable insertion, deletion of new nodes without relabeling.

6. Acknowledgment

This research is supported by KFUPM. Sincerely, thanks for all resources given to us by the university.

Referenes

[1] Assefa, B. G., Ergenc, B. (2012). Orderbased labeling scheme for dynamic XML query processing, CD-ARES 2012, LNCS 7465, p. 287–301, *International Federation for Information Processing*.

[2] Duong, M., Zhang, Y. (2008). Dynamic labeling scheme for XML data processing, Meers-man, R., Tari, Z. (eds.) OTM. LNCS, 5332, p. 1183–1199. Springer, Heidel-berg .

[3] Duong, M., Zhang, Y. (2005). LSDX: new labeling scheme for dynamically updating XML data, the 16th Australian Database Conference .

[4] Ko, H., Lee, S. (2010). A Binary String Approach for Updates in Dynamic Ordered XML Data, *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*.

[5] O'Connor, M. F., Roantree, M. (2012). SCOOTER: a compact and scalable dynamic labeling scheme for XML updates, Springer-Verlag Berlin Heidelberg.

[6] Tatarinov, I., Viglas, S., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, C. (2002). Storing and querying ordered XML using a relational database system, ACM SIGMOD.

[7] Lu, J., Ling, T., Chan, C., Chen, T. (2005). From region encoding to extended dewey: on efficient processing of XML twig pattern natching, VLDB.

[8] ONeil, P. E., et al. (2004). ORDPATHs: insert-friendly XML node labels, ACM SIGMOD.

[9] Diets, P. F. (1982). Maintaining order in a linked lists, *ACM Symposium on Theory of Computing*.

[10] Li, Q., Moon, B. (2001). Indexing and querying XML data for regular path expressions, *VLDB*.

[11] Yun, J. H., Chung, C.-W. (2008). Dynamic interval-based labeling scheme for efficient XML query and update processing, *Journal of Systems and Software* .

[12] Thonangi, R., "A concise labeling scheme for XML data. *In*: International Conference on Management of Data, COMAD 2006, Delhi, India (2006).

[13] Li, C., Ling, T. W. (2005). QED: A novel quaternary encoding to completely avoid re-labeling in XML updates, *CIKM*.

[14] Haw, S. C., Lee, C. S. (2009). Extending path summary and region eEncoding for efficient structural query processing in native XML databases, *Journal of Systems and Software*.

[15] Al-Jamimi, H. A., Barradah, A., Salahadin, M. (2012). Siblings labeling Scheme for updating XML trees dynamically, *International Conference on Computer Engineering and Technology*.

[16] Liang Xu, Tok Wang Ling, Huayu Wu, Zhifeng Bao. (2009). DDE: from dewey to a fully dynamic XML labeling scheme, *SIGMOD Conference*.

[17] Wegner, J. D., Robert Rockell. (2000). IP addressing and subnetting, including IPv6, *Syngress Media*.