World Applied Sciences Journal 18 (7): 945-949, 2012
ISSN 1818-4952
© IDOSI Publications, 2012
DOI: 10.5829/idosi.wasj.2012.18.07.1395

Node Numbering and Classification Technique to Optimize XML Query

Mohammed Ibrahim Alowais, Khairil Imran Ghauth and Ng Kok Why

Faculty of Computing and Informatics, Multimedia University, Malaysia

Abstract: As the eXtensible Markup Language XML becomes a standard for data exchange and retrieval over the Internet, the topic of optimizing the search time and space is a matter of discussion among several research and educational communities. In this research we discuss the XML query optimization methods and techniques and highlight some of the important work done in the topic so far. At the end of this paper we propose a node numbering and classification technique (NNC) to restrict the search in XML document to small portion of the XML data. The node numbering and classification technique showed good performance in shortening the time taken to process queries.

Key words: XML · Node classification · Node numbering · Query optimization

INTRODUCTION

The extensible markup language XML is widely used in today's World Wide Web as a standard for information exchange and retrieval. XML allows for easy exchange of information and documents in a standardize environment regardless of the running platform. The main goal of XML is to encourage the interoperability and simplicity in the use and development of the web. However the speed of processing XML queries tend to slow down as larger number of XML documents involved in the data exchange. This is due to the procedure the query has to go through and it also depends on the strategy adopted. Many query languages are based on path expression which requires us to navigate the whole graph. XML is seen as a graph starts with the root then node and ends with leaves (Figure1).

Path expression query needs to navigate from the root till the leave or till the node being searched and this would consume time and make the process slower. Another issue is that most of the XML systems use relational databases which take time to translate and transfer the query into SQL query. According to Florescu and Kossmann [1] in order to map XML data into relational DBs table, we scan and parse documents and store all information into the relational tables and then the XML queries are translated into the SQL queries.

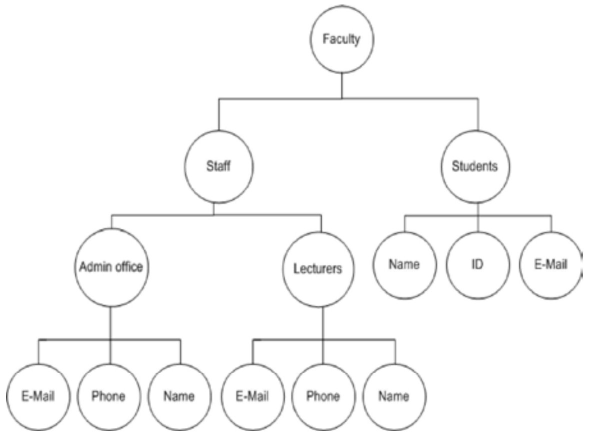

Fig. 1: Tree for Faculty staff and students

The challenge in the regular path expression is whether query contains single or multiple regular path expressions. Most optimization techniques perform well with single values but when multiple path expression is involved, the efficiency of the technique decreases. Also, ignoring the schema in some methods and techniques may cause the query optimization less efficient.

In this research, we will use Node Numbering and Classification (NNC) approach because the nodes contain valuable information about the data that would help in optimizing the search time and will number XML documents in order for the query to be processed faster.

Corresponding Author: Mohammed Alowais, Faculty of Computing and Informatics, Multimedia University, Malaysia.
Address: Riyadh 11646 Saudi Arabia. P.O.Box: 105441.
Tel: +966555467233.

Even if query is single of multiple path expression, the effect will be less with using the schema numbering. Similar approach has been used by Li and Moon [2].

The objectives of this research are to shorten the search time of XML query, determine the effect of the schema numbering on the performance of the querying process.

The rest of the paper is organized as follows: Section two describes literature review where we reviewed others work; Section three illustrates our proposed method in details; Section four describes the experiment evaluation; the result and discussion is discussed in section five; and this is followed by the conclusion section.

Literature Review: There are many query languages that have been proposed such as X-Query, XML_QL, XQL, Lorel and XPath. These query languages retrieve data from databases or documents. They are based on regular path expression. Many researchers discovered various methods and techniques for the evaluation of XML queries. Wang *et al*. [3] proposed two path expression optimization principles, which were the path shortening and path complementing. The two strategies utilized extent join algorithm. However, the first one reduced the number of join operations. The latter replaced a high cost path expression with a lower cost path expression. The path expression stored the full path of the XML document and whenever a query came in, it went along multiple join operations which slowed down the searching process.

Other researchers studied the path index. Yang *et al*. [4] proposed a path index based on Patricia-tries (PT) which helped to make the search faster, compressed the path indexes and stored the text and structure of XML data. So, it needs not to read the data from disk, which at the end enhances the searching process. The PT merged single child nodes with the parents to allow more free space and less searching. Li and Moon [2] proposed their XISS system for storing and indexing XML based on numbering scheme. The numbering scheme allows us to determine the ancestor-descent relationship which speeds up the process by determining the relationship between them faster. The last two papers are similar in using the numbering method in which the first one uses alphabetic and the second uses numbers to label each node and both of them decompose the paths into smaller size.

Cheng and Ng [5] suggested an efficient index lattice for XML query optimization. They discussed the inadequacy of the structural index to process the value-based queries where these indexes would examine the data value of each node in an equivalent class. Their proposed Structure Index Tree (SIT) considers different combinations of the root-to-leaf paths. In total, there are 2^n combination where n is the number of leaf nodes. The lattice elements exclude the inappropriate elements to speed up the query performance. Grimsmo [6] introduced two techniques for faster path indexes by combining inverted lists, selectivity estimation, hit expansion and brute force search for the first technique. The second technique used the suffix trees with additional statistics and multiple entry points into the query. These techniques are faster especially when path query uses descents axis and wildcards.

The use of DataGuides was proposed in [7]. It was intended to be a short and accurate summary of the database structure. The same path needed not to be repeated many times. DataGuides re-wrote the original database in a way that every path was repeatedly written only once. Nevertheless, the path index PT in [4] was much faster in processing the queries. Paparizos *et al*. [8] proposed the use of schema to optimize the query processing. They built a structure and named it as the schema information graph (SIG). This would store the metadata and process the graph to produce several paths that could be used, which had lower cost than the main ones.

In the PT index study, the objects are named using the alphabets for example; node 1 = a, node 2 = b, node 3 = c and etc. So, when the number of objects is more than the letters, a combination of two or more letters can be used. The document is compressed. For example, the document with size of 153M will be compressed to be 20M only. In the process of locating the elements, there are two mapping tables: one is the tag and alphabet table and the other is the element table. The elements are located using head node in the XML document.

Li and Moon [2] proposed their XISS system for storing and indexing XML. In their system, the schema numbering was used to address the problem of the insertion and deletion (update). The numbering schema allowed one to determine the relations between the different nodes. Three indexes structured had been proposed, which are element, attribute and structure index. As well as the proposed join algorithms that could process the regular path expression without going through the hierarchy of the XML document.

These methods are efficient in processing XML query. However, combining the numbering with other technique would result in a stronger query optimization technique. In the following section, we will explain our proposed method based on our findings in the literature review.

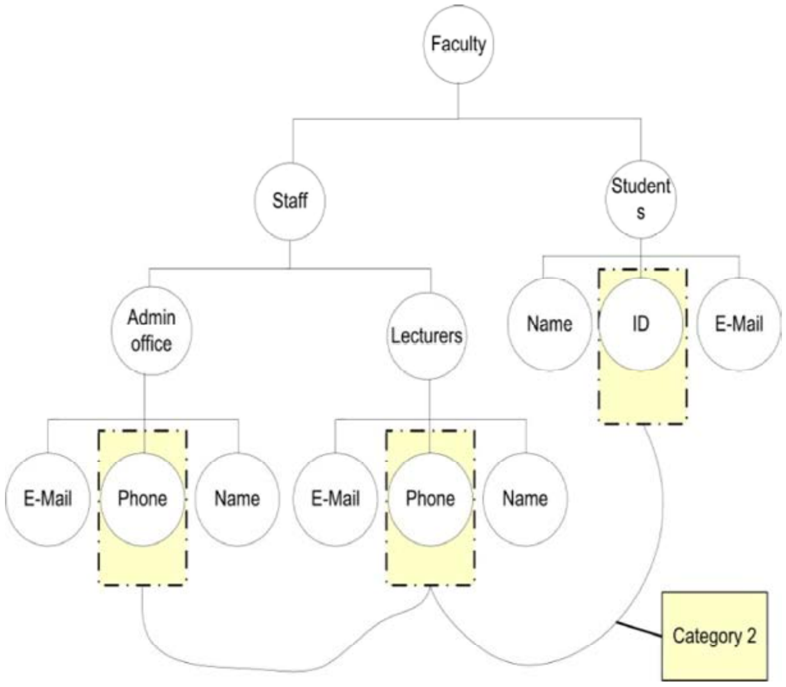

Fig. 2: Node Classification Example

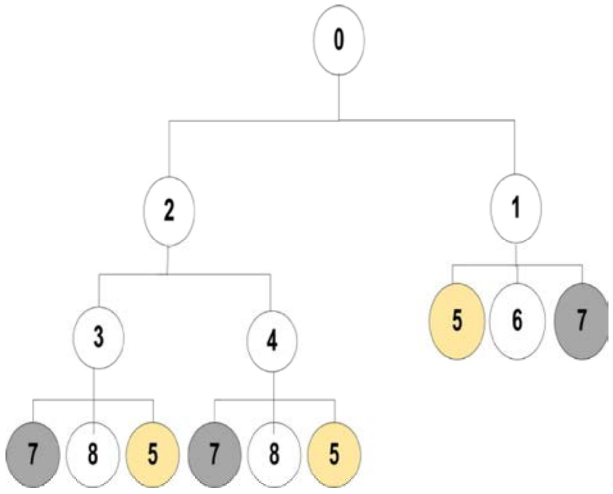

Fig. 3: Node Numbering Example

Proposed Method: To optimize the query of XML, we propose an approach similar to the ones proposed in [2,4]. Node numbering and classification technique (NNC) is used in our method to number the nodes of the XML documents and to classify or categorize the node based on the content of the nodes. For instance, if the query asks for objects that contain numbers like amount, price, phone and so on, the query does not need to search the other categories such as information about customers like name, address and so on. Consider the first XML tree example which we can explain our method as follow.

Figure 2 illustrates the classification method. The figure indicates how to classify the objects based on their types. For example, the phone and ID are of numerical types. So, it is categorized under category 2. The first category includes text types for the objects which are in this example the name. The third one is the email which is classified under category type number 3, because it contains mix of values numbers and alphabets. The node numbering is illustrated in the following Figure 3.

Figure 3 shows the numbering of the nodes. If we refer back to the Figure 2, we can see that the similar nodes' contents are numbered with the same number. For example, the node 7 represents the E-mail tag, node 5 represents the name and node 8 represents the phone tag and so on. Any query search for name will have node numbers that will minimize the search and restrict it for only the nodes number 5 which contains the names. Same thing goes for the other nodes Email, phone and ID.

The advantage of using the classification and numbering technique (NNC) is to limit the search for very small part of the XML documents instead of searching the complete tree.

- The numbering method is based on the following pseudo-code:

```
For each node (n) where n =! Root node
Find node tag(n) = node tag(ni) join tag n with tag(ni)
```

- The classification method looks for the similar data type or the content type.

```
V   =  node category number  find same v in each (ni)
       set N elements to V
```

Experimental Evaluation: In this experiment, we used the eBay dataset of size 36KB and Book Catalogue dataset of size 4KB to evaluate the effectiveness of the proposed method against the SAX method in terms of time to retrieve data. The experiment was carried on using a personal computer with Intel core i5 2.5 GHz with RAM of 4.0GB and disk storage of 300 Gigabytes. The measurement of the method performance is based on the time taken to complete the query.

For the performance measurement, we have created a program on C# to retrieve the data, process the query and to calculate the time taken to complete a query.

RESULT AND DISCUSSION

This section describes the performance result of our proposed method (NNC) and the result of the comparison with the SAX method.

Table 1 shows the comparison between the SAX way of retrieving data from XML and the node numbering and classification method (NNC). The NNC makes the search time faster by 54.54% for the eBay dataset and 66.67 % for the Book Catalogue, which is obviously faster than the SAX in retrieving document.

Table 1: Comparison between SAX and NNC

Dataset	Method	Time(s)	Method	Time(s)
eBay	SAX	0.011	NNC	0.006
Book Catalogue	SAX	0.004	NNC	0.002

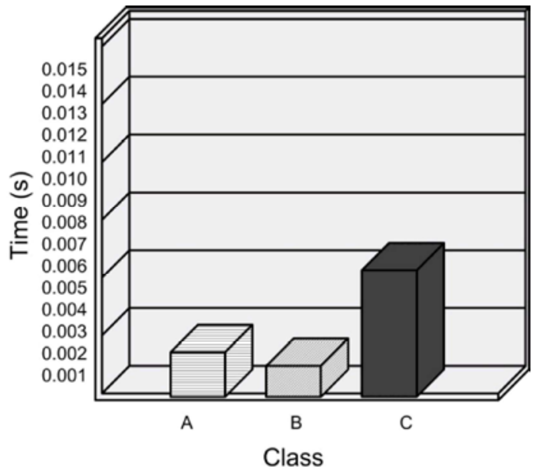

Fig. 4: Retrieve time on eBay dataset using NNC

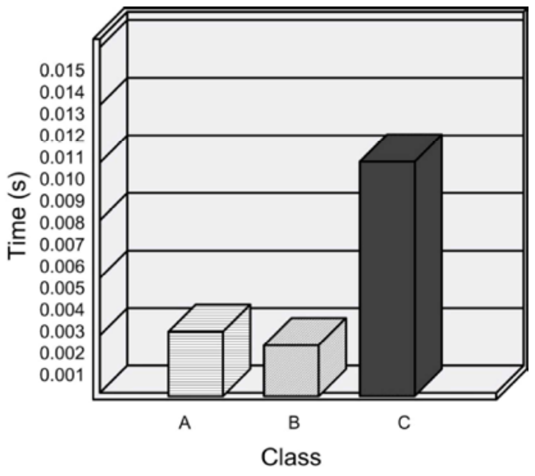

Fig. 5: Retrieve time on eBay dataset using SAX

More details on the time measurement is illustrated in the next figures. The eBay dataset contains three classes; numbers, letters and mix values. The Book Catalogue contains only two classes; letters and numbers. The figures show the retrieval time for each class using both SAX and NNC. The unit used is second.

The graph in Figure 4 above indicates the performance of the proposed NNC on the eBay dataset. The figure shows the retrieve time for each of the classes in the dataset. Figure5 indicates the retrieval time using the SAX on the eBay dataset. We can see that our method is faster in retrieving all the classifications of A, B and C.

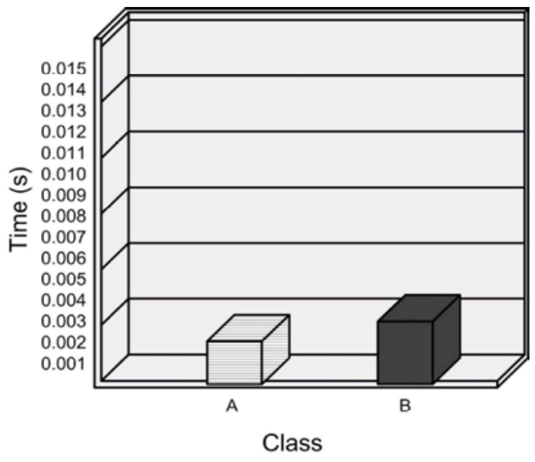

Fig. 6: Retrieve time on Book Catalogue dataset using NNC

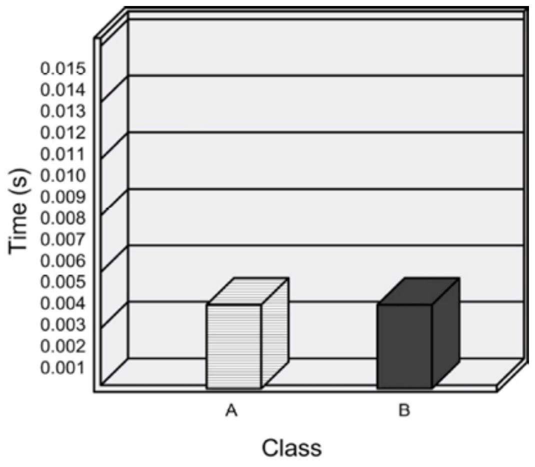

Fig. 7: Retrieve time on Book Catalogue dataset using SAX

Figure 6 shows the retrieve time on the Book Catalogue dataset using the NNC. This dataset has only two types of data numeric and characters. So, we have only two classifications of A and B. Figure 7 show the retrieve time on the Book Catalogue using the SAX. The two classes take similar amount of time. We can see that the NNC is faster by almost 50%.

CONCLUSION AND FUTURE WORK

In this research work, we have proposed a method that is able to optimize the speed of querying an XML data. We use the node numbering to limit the search for certain objects or nodes. In the result, our method showed an acceptable performance as compared to the others.

However, the matter of optimizing the search time and space is still open for discussion from many perspectives. The suggested works to be done in future are improvement of the schema naming or numbering techniques to locate the object being searched faster and adapting the tags classification method and expand it to function on the different XML document contents.

REFERENCES

1. Florescu, D. and D. Kossmann, 1999. Storing and Querying XML Data using an RDMBS. IEEE Database Engineering Bulletin DEBU, 1-8. Citeseer.
2. Li, Q. and B. Moon, 2001. Indexing and Querying XML Data for Regular Path Expressions. Data Base, 361-370. Morgan Kaufmann Publishers Inc.
3. Wang, G., M. Liu, J.X. Yu, B. Sun, G. Yu, J. Lv and H. Lu, 2003. Effective schema-based XML query optimization techniques. Symposium A Quarterly Journal In Modern Foreign Literatures, pp: 230-235. IEEE Comput. Soc.
4. Yang L., Y. Ping and L. Qiyan, "Optimizing Path Expression Queries of XML Data," icebe, pp: 497-504, IEEE International Conference on e-Business Engineering (ICEBE'05), 2005.
5. Cheng, J. and W. Ng, 2007. "An Efficient Index Lattice for XML Query Optimization", Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong.
6. Grimsmo, N., 2008. Faster Path Indexes for Search in XML Data. Proceedings of the nineteenth conference on Australasian Database, pp: 127-135
7. Goldman, R. and J. Widom, 1997. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. (M. Jarke, M.J. Carey, K.R. Dittrich, F.H. Lochovsky, P. Loucopoulos and M.A. Jeusfeld, Eds.)proceedings of the international conference on Very Large Data Bases, pp: 436-445.
8. Paparizos, S., J.M. Patel and H.V. Jagadish, 2007. SIGOPT: Using Schema to Optimize XML Query Processing. Data Engineering 2007 ICDE 2007 IEEE 23rd International Conference on. pp: 1456-1460. Ieee.
9. Mchugh, J. and J. Widom, 1999. Query Optimization for XML. (M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik and M. L. Brodie, Eds.) VLDB, pp: 315-326. Morgan Kaufmann..
10. Chung, T., 2003. Techniques for the evaluation of XML queries: a survey. Data and Knowledge Engineering, 46(2): 225-246.
11. Buneman, P., A. Deutsch, W. Fan, H. Liefke, A. Sahuguet and W.C. Tan, 1998. Beyond XML Query Languages. Query Language Workshop QL98.