Single Number Path Encoding Tree of XML

Tao Cai, Shiguang Ju, Wenyi Zheng, and Dejiao Niu

Department of Computer Science, JiangSu University, JiangSu, P. R. China, 213013

Abstract:
Indexing of element's path is the important way to improve efficiency of XML querying. In this paper, we compare and analyze several existing methods of element's path indexing. Then we indicate some shortcomings of them such as inefficiency, poor flexibility and disorganizing the structure of XML. We present a new querying strategy for XML document. In this method we encode the element's path as single number and use B-Tree to index it. The calculation of element's path number is simple. And its flexibility is good. In order to decrease the blindness and reduce the range of XML querying, pretreatment the structure of XML (DTD or XML Schema) are needed. Finally, we analyze the querying performance and flexibility of single number path encoding tree. And we also discuss the possibility of using single number path encoding tree in querying semi-structured, structured and unstructured data.

Keywords:
XML, Index Structure, path encoding, Date Structure

Introduction

XML is a semi-structured and semantic language, which can be used to describe and exchange the information. How to accelerate the querying of XML is an important problem. The indeterminate element's path, element's name, element's value and element's type are needed in XML querying, which require several indexes to cooperate. Element's name indexed by XML-schema(DTD or XML Schema)[1]. Converting the element's value to string and indexing element's value, converting the element's type to string and indexing element's type. Element's path is complex and has indeterminate dimensional variable, so its index is very complex [2]. Now there are several indexing methods: indexing based on tree, indexing based on figure, indexing by element's path encode and indexing based on regular expression. One methods name as RE-tree comes from the RD-tree in object-oriented database management system [3].

In the lore project of Stanford University, there were three indexes relating to the node's path, which were Lindex(label, index), Bindex (edge, index) and Pindex (path, index)[4,5]. These indexes repeated with each other and cooperated in querying XML document. So it is inefficiency, its time-consuming and space-consuming were large.

Manuscript received October 5, 2025
Manuscript revised October 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.10.5

Yoon, Raghavan and Chakilam proposed a 3-D bitmap indexing method for XML document[6]. In their method, a collection of XML document was represented as a 3-D structure: XML document, element's path, terms and words. They defined a BitCube index scheme to partition document into clusters for efficient retrieval. But building up and renew BitCube index scheme was difficult, element querying was slow, time-consuming and space-consuming were large.

Flavio Rizzolo and Alberto Mendelzon of the University of Toronto proposed an indexing method that synthesizes ideas from object-oriented path indexing and extends to semi-structured realm of XML data[7]. Their method use an edge-labeled graph to model XML data. They signified element's path by object's path, and built element's path indexing by RD-Tree. There were some shortcomings of RD-Tree such as: inefficiency, large time-consuming, and large space-consuming. Brain Cooper et al. proposed an indexing method, called Index Fabric[8], that encoded element's path as simple strings and inserted those strings into a special index that is highly optimized for long and complex keys. The element's path querying converted to substring querying. But there were some difficulties in creating special substring index and matching substring.

Because element's path is a indeterminate dimensional information, there are some common disadvantages in above indexing methods such as: complex structure of indexing, inefficiency, large time-consuming, large space-consuming, and element querying by navigate indexing. The method of simple element path signifying is important to improve the path indexing performance. Researchers have proposed some element path encoding methods. In this paper, we introduce those encoding method firstly, analyze their performances, advantages and disadvantages. Then we present our novel encoding method of element's path, pretreatment method of the XML data by XML-schema, and implement a novel element path indexing approach. At last we analyze its performance.

2. Related Works

Dietz presented the numbering scheme of XML[11]. He used tree traversal order, preorder and post-order, to determine the ancestor-descendant relationship

between any pair of tree nodes. His proposition was the following:

For two give nodes x and y of a tree T, x is an ancestor of y and only if x occurs before y in the preorder traversal of T and after y in the post-order traversal.

The disadvantage is the limited flexibility. For each updating and inserting, the preorder and post-order numbers have to be recomputed for many tree nodes. At the same time, two dimensional indexing used to index preorder and post-order was inefficiency.

Li and Moon proposed a numbering scheme based on Dietz's method. They used an extended preorder and a range of descendants. The proposal numbering scheme associates each node with a pair of numbers ***<order, size>***, where order is an extended preorder and size is a range of descendants, as follows:

For a tree node y and its parent x, order(x) < order(y) and order(y) + size(y) = order(x) + size(x).

For two sibling nodes x and y, if x is a predecessor of y in preorder traversal, order(x) + size(x) < order(y).

Then, for a tree node **x**, size(**x**) = $\sum$ (size(**y**)) for all y's that are direct children of x. This method was to make the numbers far apart to make room for insertions. The proposition was following:

For two given nodes x and y, x is an ancestor of y if and only if order(x) < order(y) = order(x) + size(x).

Li and Moon's method solved the Dietz's numbering scheme limited flexibility partly. When the number's room was full, insertion and updating would cause many node's number to be recomputed. The two dimensional indexing have be used also and it was inefficiency.

To solve above problem Gongzhu Hu and Chunxia Tang proposed new numbering scheme[13]. It associate each node with a pair of numbers ***<K, C>***, K is the level of the node tree, and C is of the form ***c1:c2:…c3*** (***ci*** is the number of parent node at level ***ith***). Next is the algorithm of assigning the ***<K, C>*** pair to the nodes:

1. *The root node has the code (1, 1). That is, the root is at level 1 and c(root) = 1.*
2. *At level **k** (**k** > 1), the ith sibling node A is assigned the code <k, c(p) : i> where p is the parent of A.*
3. *At level k, assign the same serial number to all the sibling attributes so that they have the same code.*

This new numbering scheme allows inserting new node to the last of sibling nodes, without re-numbering existing nodes' number pair. But that disorganize the sequence of the sibling nodes in XML data. ***C*** composes by indeterminate number of variable and acts as string in its index, so its indexing approach has complex comparing method and is inefficiency.

In above numbering scheme, node's encode are indeterminate dimensional value, which are complex, and have poor flexibility. That makes indexing of node's encode has large time-consuming and space-consuming, it is inefficiency. If the indexed data is simpler and lower dimensional variable, its index has a higher performance. So we propose to reduce the complexity of element's path value, using single number encode in place of the complex and indeterminate element's path value. Using B-Tree to index single number encode of element path and improving the flexibility of encoding tree.

In above methods, researchers have not considered the indeterminacy of XML data. In XML document, nodes maybe appear in several different places or do not existing. Only after accessing all nodes, we can find whether the node exists in XML document. XML is not unstructured, using the XML-scheme to decrease the blindness and range of querying is an important method to improve the efficiency. We analyze the structure of element path, and present the single number encoding method and its indexing method.

3. Element Path's Signifying

XML document contains XML data and XML-scheme. We propose two definitions for element path's signifying.

Definition 1: We use triple of ***(sid, spath, sorder)*** to signify element path in XML-scheme. ***Sid*** is the identification of node in XML-scheme. ***Spath*** is the path's value from root node to destination node (***M(snode)***), it contains ***M*** numbers node's identification. ***Sorder*** is the node's order in sibling nodes.

Definition 2: We use triple ***(id, path order)*** to signify element path in XML data. ***Id*** is the identification of node and attribute. Path is path's value from root node to destination node (***M(node)***), it contains ***M*** numbers node's identification. ***Order*** is the node's order in sibling nodes.

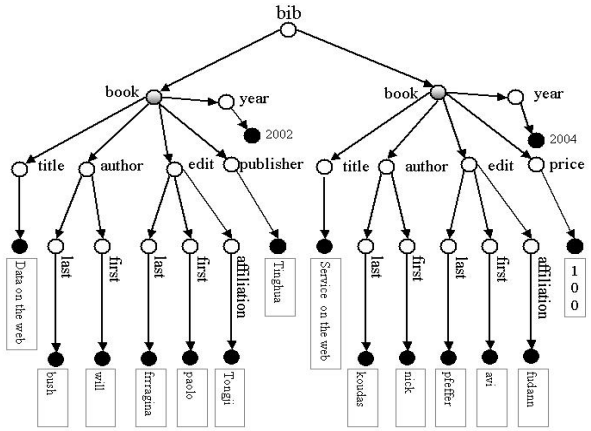
In above path variables, ***path*** and ***spath*** contain indeterminate ***M*** numbers node's identification. We can not use the popular indexing method to accelerate its querying. And node's path value contains node's order in sibling nodes, which makes indexing more difficultly.

We present a novel path encoding method, using single number to encode the indeterminate element's path value, improving the flexibility of single number encoded tree, and indexing it by B-Tree to improve the querying performance. Next we present the element path's single number encoding method.

4. Method of Element Path's Single Number Encoding

We use ○ as the node, use ● as node's value and use ◎ as the node's attribute. Figure 1 is the XML document tree using this method.

Fig.1 XML document tree



4.1 Simplification of XML document tree

In three element types of XML document tree, there are only node and attribute correlate to the element path. We make brief of XML document tree, delete all ● , retain ○ and ◎.

Element path contains all nodes' identification from the root node to itself. We can use node's ID or name to describe it. But relative position and layer of node are better to signify node path's value. We use single number to encode it. Numbering algorithm is presented as follow:

1) The root node has the code 1.

2) Assigning sequel number to sibling nodes from 1. Sibling nodes with different ancestor have not sequel code.

3) Node's attribute treat as node's son-node and is the latest one in all sibling nodes.

Then we exchange XML document tree in figure 1 to numbering tree in figure 2

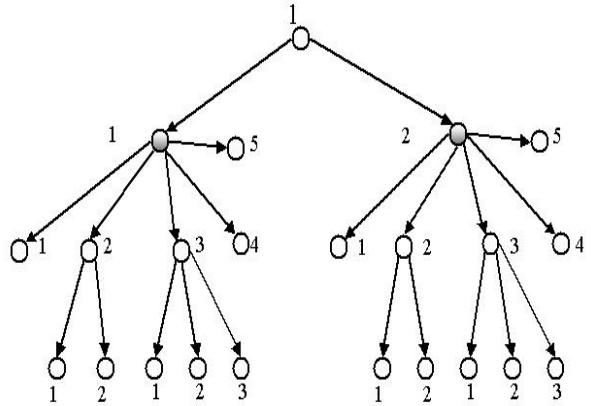


Fig.2 XML Numbering tree

4.2 Algorithm of single number path encoding tree

At first, we present the rule of translating numbering tree to single number path encoding tree. The algorithm is presented as follow:

1) Element path's single number encode is ***N*** numeral.

2) Root path's encode is 1.

3) Let parent's encode is ***FN*** and node's numbering is ***LO*** (***N*** numeral), then node path's encode is ***FN * N + LO***. Figure 3 is the single number path encoding tree converted from XML numbering tree.

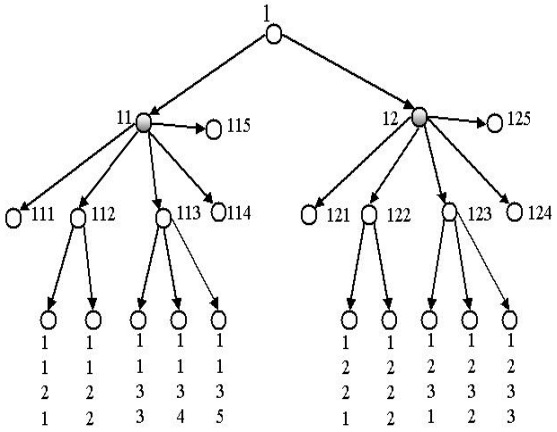


Fig.3 Single Number Path Encoding Tree (N = 3)

In the calculating of single number path's encoded, ***N*** is very important. The algorithm of calculating ***N*** is presented as follow.

```
FOR （All nodes with sub-node in XML document）
{
        Calculate the number of sub-node → NN;
NN -> NUM[i];
}
Calculate the max in the NUM[maxnode]  -> N1
If （N1 is odd）
        N1 = N1 +1;
```

(Let ***N*** bigger than ***N1*** which calculate form above method, then make room for modification of ***N*** to improve tree's flexibility. Latter we will discuss this tree's flexibility particularity.)

5. XML indexing method based on single number path encoding tree

Because of XML's indeterminacy, it is very difficult to query XML data. But XML data is not unstructured, we can reduce the range of querying and predict the querying result based on the XML-schema. Next we introduce the indexing method of single number

path encoding tree and pretreatment algorithm based on XML-schema.

5.1 Indexing of single number path's encode

We can use a two dimensional table to store the XML data, which contains four fields: node's identification or name, single number path's encode, pointer of stored place and deleted marking. Using B-Tree to index the single number path's encode and improving the efficiency of XML querying.

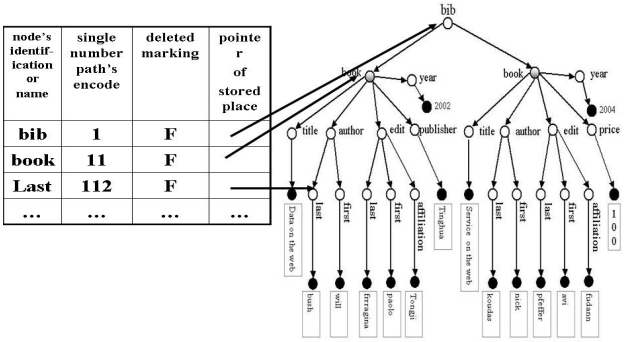


Fig.4 The single number path encoding index based on B-Tree

5.2 Algorithm of Node querying

There are two steps. In first step, we using XML scheme to predict querying result and reduce the range of querying.

```
FOR ( Every label of <!ELEMENT> in DTD )
{
        IF ( Destination node in sub-node of this
   <!ELEMENT> )
        {
                IF ( this sub-node is necessary ) AND ( the
   above sub-node are necessary )
                {
                        BEGIN=layer of parent node  * N
   + order in the sub-nodes;
                        END = ( layer of parent node + N –
   1 ) * N + order in the sub-nodes;
                }
                ELSE
                {
                        BEGIN = layer of parent nod * N
   + 1;
                        END = ( layer of parent nod + N -
   1 ) * N + ( N – 1 ) ;
                }
        }
        ELSE
        {
                Recurrence all sub-nodes
        }
}
```

As result, we use ***BEGIN*** and ***END*** to store the range of querying. If they are 0, we can not find destination node and prompt to user. Otherwise we do second step.

The second step: Using indexing of single number path encoding tree to compare node's name and value in those nodes which single number path's encode is in range from ***BEGIN*** to ***END***. The algorithm is presented as follow.

1) We find all nodes which single number encode in range from ***BEGIN*** to ***END***.
2) We compare node's name and value, and find out the suitable nodes.
3) Lastly we return all suitable nodes information and prompt to user.

5.3 Algorithm of Sub-node Querying

The algorithm is presented as follow:

1) Checking the XML-schema. If we can find node with the same name, do next steps. Otherwise return users that there is not the suitable node.
2) We assume the destination node is ***M1*** layer below the current node, its first possible position is ***L1*** in sibling node, and current node's path encode is ***NODEF***.
3) Querying in range from $NODEF*N^{M1}+L1$ to $NODEF*N^{M1}+N$ based on B-tree index, and finding out the suitable nodes.

5.4 Algorithm of Parent Node Querying

The algorithm is follow :

1) Checking the XML-schema. If we can find node with the same name, do next steps. Otherwise return user that there is not the suitable node.
2) We assume the destination node is ***M2*** layer below the current node, its first possible position is ***L2*** in sibling node, and current node's path encode is ***NODEC***.
3) Querying the suitable node in range from $NODEF*N^{M2}+L2$ to $NODEF*N^{M2}+N$ based on B-tree index, and finding out the suitable nodes.

5.5 Algorithm of Deleting Node

Do not update node's single number path encode immediately, node's deleted marking field and sub-nodes' deleted marking field of table in 5.1 are ***T***. Then we delete this node and its sub-nodes in logical.

When there are little missions in XML management system or XML management system shrink itself, the system updates nodes' single number path encode and modify its index. This contains three steps as follow ;

1) Deleting node and all its sub-nodes.

2) Recalculating some nodes' single number path encode: Those node are after current node in sibling nodes and all of their sub-nodes.

3) Using B-Tree to re-index single number path encoding tree.

5.6 Algorithm of Inserting Node

We must update some nodes' single number path encode immediately. We discuss it in two cases:

Case 1: The number of sibling nodes after inserting is not large than ***N***. We recalculate single number path's encode of nodes that are after current node in sibling nodes and their sub-nodes. And re-index new single number path encoding tree.

Case 2: The number of sibling nodes after inserting is large than ***N***. This is the worst situation of system. We must recalculate single number path encode of all nodes in XML document and re-index them. The steps as follow:

1) The new value of ***N*** is ***N+M***, ***M*** is the even number bigger than 2.
2) Using new ***N*** value to recalculate all nodes' single number path encodes in XML document.
3) Re-index single number path encoding tree using B-Tree.

6 Analysis of Performance

The performance contains three parts: flexibility, efficiency and generality.

6.1 Flexibility

The flexibility is important to the efficiency of XML querying. We discuss it in two cases:

1) Deleting and inserting without change of ***N*** value. Recalculate the nodes that its order after current node in sibling node and their sub-nodes. Those nodes are little part of all nodes in XML document, so its consuming is not large.

2) Deleting and inserting with the change of ***N*** value. It is the worst situation of system, we must recalculate all node's single number path encode in XML document. If making room for ***N***, we can reduce the incidence of this situation.

In summary, single number path encoding tree has good flexibility. The situation of recalculating frequently and recalculating large number of nodes path encode is few. And it may not disorganize the structure of XML document which occurred in Gongzhu Hu and Chunxia Tang approach.

6.2 Efficiency of Querying

Let the number of node in XML document is ***NUM1***, and the number of node in XML-schema is ***NUM2***. Then we analyze the efficiency in two parts of time-consuming and space-consuming.

1) Space-consuming

Additional space-consuming of single number path encoding tree contains: one space-consuming for ***N***, ***NUM2*** spaces-consuming for XML-schema and ***NUM1*** spaces-consuming for node's path encode of XML document. So it needs ***NUM1+NUM2+1*** additional space-consuming.

2) Time-consuming

a. Querying XML scheme: it needs ***NUM2*** times-consuming at the most.

b. Querying XML document its path encode in particular range: The range is small. The minimum of it is ***1***, the maximum of it is ***NUM1***, the average value is ***(1 + NUM1) / 2***. Using B-Tree to index it, the average querying times-consuming is $\log_{m/2}((NUM1+3)/4)+1$ (***M*** is the exponent of B-Tree).

In summary, the space-consuming of single number path encoding tree is small, and its speed of querying is quickly.

6.3 Generality

Encoding algorithm should have nice generality and could been used in different types of data querying. We discuss how to use single number path encoding tree to accelerate semi-structured data querying, structured data querying and unstructured data querying.The difference between semi-structured data and XML data is that the sibling nodes are unordered. So we modify encode calculating method, without adding the node's order of sibling nodes to node's encode. Then we can use single number path encoding tree to accelerate semi-structured data querying.

If standardizing the XML data and converting to structured data, it querying is similar to attribute querying in object-oriented database. Using single number path encoding tree, we can reduce range of querying, and its B-Tree indexing is more efficient than RD-Tree indexing.

Unstructured data do not have any structure. If we know the layer of the destination node, we can use node's layer to reduce the range of querying also. So the single number path encoding tree has the nice generality. It can be used to accelerate semi-structured data querying, structured data querying and unstructured data querying.

7. Conclusion

In this paper we present the single number path encoding tree to reduce the range of querying and accelerate XML querying. It has good flexibility and do not disorganize structure of XML document. Compare with other approaches, we outline several advantages as follow;

1） Using single number to signify indeterminate dimensional node's path.
2） The algorithm of calculating path's encode is simple and the single number node path encoding tree has good flexibility.
3） Using B-Tree to index path's encode is more efficient than other multidimensional indexing method.

We analyze and prove that single number path encoding tree has good flexibility and efficiency. And we discuss how to use single number path encoding tree to accelerate semi-structured data querying, structured data querying and unstructured data querying. And we prove that single number path encoding tree has nice generality.

Acknowledgments

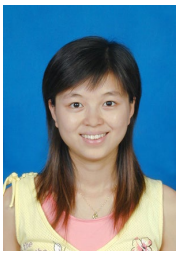
This work was supported by National Natural Science Foundation of China (No.0373069) and JiangSu Nature Science Foundation (No.BK200204).

References

[1] Tae-Sun Chung, Hyoung-Joo Kim, XML query processing using document type definitions,The Journal of Systems and Software 2002 (64)195-205.
[2] Chiyoung Seo, Sang-Won lee, Hyoung-Joo Kim, An efficient inverted index technique for XML documents using RDBMS, Information and Software Technology 2003(45) 11-22.
[3] Chee-Yong Chan,Minos garofalakis,Rajeev Rastogi, RE-tree:an efficient index structure for regular expressions, The VLDB Journal 2003 12: 102-119.
[4] Manolescu L,Florescu D,Kossmann D, et al. Agora:Living with XML and Relational. Http://citeseer.nj.nec.com/manolescu00agora.html
[5] J. McHugh, J. Wisdom, S. Abiteboul, Q. Luo, and A. Rajaraman: Indexing Semistructured Data. Technical Report, Standford University 1998
[6] J.Yoon,V. Raghavan,V. Chakilam, and L.Kerschberg: BitCube:AThree-Dimensional Bitmap Indexing for XML Documents. Journal of Intelligent Information Systems, 17, 2001
[7] F. Rizzolo, A. Mendelzon: Indexing XML Data with To Xin. Fourth International Workshop on the Web and Databases (in conjunction with ACMSIGMOD 2001). Santa Barbara, California 2001
[8] B. Cooper, N. Sample, M. Franklin, G. Hjaltason, and M. Shadmon. A fast index for semistructured data. In Proc. VLDB, 2001.
[9] R. Goldman and J. Widom. DataGuides: enabling query formulation and optimization in semistructured databases. VLDB, 1997.
[10] B. Cooper, N. Sample, and M. Shadmon. A parallel index for semistructured data. ACMSAC, 2002.
[11] P. F. Dietz: Maintaining order in a Linked List. Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, San Francisco, California 1982
[12] Q. Li and B. Moon: Indexing and Querying XML Data for Regular Path Expressions. Proceedings of the 27th International Conference on Very Large Databases, Rome, Italy 2001
[13] Gongzhu Hu, Chunxia Tang: Indexing XML Data for Path Expression Queries, SERA 2003, LNCS 3026, pp. 332–348.

Cai Tao received the B.S. and M.S. degrees from JiangSu University in 1998 and 2001, he is Ph.D candidate of computer science, Jiangsu University, Zhenjiang, China. And he is a teacher of Jiangsu University, Zhenjiang, China. His research activity mainly focuses on XML management system, storage network system and security of information.

Shiguang Ju received his M.S. degree in 1988, from Nanjing University of Science and Technology (China) and a Ph.d degree in 1996 from National Polytechnic Institute (Mexico). From September 2003 to April 2004, he was a visiting professor of Texas Tech University (U.S.A). He is now a professor of Jiangsu University, China. His current research interests include information security and spatial database.



WenYi Zheng received the B.E. from JiangSu University, Zhenjiang, China.in 2002 and M.E. degrees from JiangSu University, Zhenjiang, China in 2005. After working as a assistant (from 2002) in the Dept. of Computer Science, JiangSu Univ. Her research interest includes security of database management system, XML management system, and their application.

JiaoNiu De received the B.E. from JiangSu University Zhenjiang, China.in 2000 and M.E. degrees from JiangSu University, Zhenjiang, China in 2003. After working as a teacher (from 2000) in the Dept. of Computer Science, JiangSu Univ. Her research interest includes XML management system, digital signature and their application.