Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp475-480)

An XML Index using Prime Number Coordinate for Efficient Query Processing

Xiaofan Hong, Jeonghoon Han,
Tackgon Kim, Woosaeng Kim
Dept. of Computer Science,Kwangwoon University
Wolgye-dong, Nowon-gu, Seoul, Korea

Abstract: - Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML and is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. Recently, a lot of index techniques for storing and searching XML documents have been studied so far and many researches of them used coordinate-based methods. In this paper, we propose an efficient index technique with Prime Number Coordinate (PNC) for retrieving the position and relationships of nodes using simple calculation on XML trees. It can be efficient index to search query as containment query with relational database and carry out update operation, insert operation and delete operation on XML documents easily.

Key-Words: - XML Index, Query Processing, Prime Number Coordinate, PNC

1 Introduction

XML [1], which is derived from SGML [2] through recommendations of W3C (World Wide Web Consortium), comes up as a standard language, that can exchange the data through Internet and still can keep the interoperability, in recent years. There are many applications with abundant representation ability of XML language. It is the trend to use XML for expression of all the data in business application program. Therefore, there are steady efforts for the study to store and retrieve the XML documents in the database efficiently and it needs to provide any methods to preserve not only the structural contents also positional information in the XML documents.

As the popularity of XML increase, many XML query languages have been proposed. The people spend efforts for the study to store and search the XML documents in the database efficiently. It needs to provide any methods to preserve not only the contents also structural and positional information on the XML documents.

In this paper, we propose an index technique that expresses relative structural information with PNC using RDBMS. This technique utilizes prime number to label the position of nodes in Prime Number Coordinate so that we can easily retrieve the position and relationships of nodes with simple calculation. In addition, we consider update operations to manage XML documents easily, and these operations which are insert, update and delete operations, show good performances.

This paper is organized as follows. In chapter 2, we review related works about index techniques of XML documents. In chapter 3, we introduce an efficient index technique with Prime Number Coordinate. In chapter 4, we present operations for proposed index technique, and in chapter 5, we show the performance evaluation between the proposed technique and existing coordinate-based techniques. Finally we make a conclusion in chapter 6.

2 Related Works

There are many works on indexing techniques available in the field of database. Index techniques for XML documents are described in a lot of researches. In [3] explains position-based indexing and path-based indexing to access XML document by content, structure, or attributes.

In path-based indexing, the location of words is expressed as structural elements and the paths in tree structures are used for the processing of query. In order to determine the position of a word within a document, it is necessary to construct an encoding of the path of the element names from the root of the document to the leaf node containing the word. And then, for each word occurrence, the inverted list includes a representation of the path to that word.

Absolute Region Coordinate (ARC) is a way to locate the content data in structured documents [4]. We can find the content of an element in an XML document based on a pair of numbers, which point to the start and the end of the piece of text corresponding to the element. In case there are many XML documents, another number may be added to the pair to indicate which document the element belongs to.

In [4], data structure for indexing XML documents based on relative region coordinates (RRC) is used. Region coordinates describe the location of content data on XML documents. They refer to start and end points of text sequences on XML documents. Region coordinates are adjusted by offsets relative to the corresponding region coordinate of the parent node in the index structure.

In [5][6], new technique based on bitmap indexing was introduced. XML documents are represented and indexed using a bitmap indexing technique. They define the similarity and popularity of the available operations in bitmap indices and propose a method for partitioning a XML document set. 2-dimensional bitmap index is extended to a 3-dimensional bitmap index, called BitCube. They define statistical measurements and correlation coefficient. BitCube proved eminent performance in performance assessment with systems such as existent XQEngine, XYZFind already through the fast search speed.

Recently, there is a trend towards dynamic labeling schemes where the nodes inherit their parents' labels as the prefix to their own labels [7]. This allows one to determine the existence of an ancestor-descendant relationship by simply examining whether the prefix relationship exists in the labels of the two nodes.

3 Prime Number Coordinate

In this section, we introduce our proposed index technique, and how to process prime number coordinate for index.

3.1 Description of index technique with Prime Number Coordinate (PNC)

To define proposed index technique, we consider prime number. If an integer *m* has a prime factor which is not a prime factor of another integer *n*, then *n* is not divisible by *m*. Prime number Coordinate (PNC) method expresses a relative relationship of nodes on XML document using prime factor like Figure 1. PNC value of the node is a number multiplied unique prime number by 2^n ($n \geq 0$). Relationship shows in (1).

$$PNC_{des} = \gamma PNC_{anc} (\gamma = 2^n)$$
$$PNC_{des} \bmod PNC_{anc} = 0 \quad (1)$$

If relationship between a node A with small PNC value and another node B with big PNC value is ancestor - descendant relation, the result of B mod A is 0 because of prime factor. And if relationship between A and B is parent – child relation, the value of n is 1. So n means depth between two nodes. For keeping the condition in (1), prime number starts 3 not 2.

Example 1. In case we assume a simple XML file using A, B, C, D, E, F and G. In Figure 1, Prime Number Coordinate X axis stands for prime number beginning from 3 and Y axis stands for 2^n ($n \geq 0$). We can get PNC value by this way that X axis value is multiplied by Y axis value. Like this, we build PNC matrix. The positions of elements on node mapping are the same to the position of element in PNC matrix.

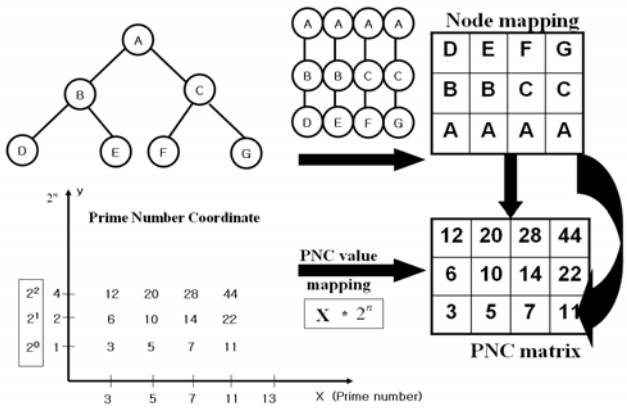

Fig. 1 PNC algorithm

3.2 PNC Features

The PNC method has prominent feature because of prime number and 2^n($n \geq 0$). Because the prime number is a special number, we create two calculations according to component structure of PNC. They are relationship calculation and depth calculation.

In PNC matrix, every column value has ancestor relation and every row has no relation because of prime factor. To avoid modular calculation, we use Enumber that is γ in (1). So it is obvious to see their

relation as (2).

Relationship Calculation

$$PNC_{des} / Enumber_{des} = PNC_{anc} / Enumber_{anc} \quad (2)$$

In PNC value calculation we use γ in order to recognize the depth with *n* between two elements. So it is easy to calculate depth between two nodes that have ancestor – descendant relation as (3).

Depth Calculation

$$\begin{aligned} PNC_{des} / PNC_{anc} &= \gamma \\ depth(n) &= \log_2 \gamma \end{aligned} \quad (3)$$

3.3 Table schemas for index

We use the table schemas in figure 2 in order to store positional and relationship of nodes. Schemas for relative structural information consist of three tables: ElementID, XMLIndex and Text

ElementID {EDocID, EID, EName}
XMLIndex {DocID, EID, Enumber, PNC}
Text {TDocID, EID, Content}

Fig. 2 Table schemas

EDocID, DocID and TDocID representing each XML document's unique ID are the same in three tables. The value of EID representing each Element's unique ID in tables is the same, too. And then in the ElementID table EName field represents Element name of node. In XMLIndex table Enumber field is used in the depth calculation and PNC field knows each Element's relation. In Text table content field contains Text information of node.

Example 2. We use a telephone XML file to express PNC. Figure 3 shows this sample and figure 4 shows the result of those tables using PNC in database.

```
<handphone>
  <Anycall>
            <style> A6 </style>
            <color> red </color>
  </Anycall>
  <LG>
            <style> L5 </style>
            <color>black</color>
  </LG>
</handphone>
```

Fig. 3 Sample

ElementID table

EDocID	EID	EName
1	1	handphone
1	2	Anycall
1	3	style(Any)
1	4	color(Any)
1	5	LG
1	6	style(LG)
1	7	color(LG)

Text

TDocID	EID	Content
1	3	A6
1	4	red
1	6	L5
1	7	Black

XMLIndex table

DocID	EID	Enumber	PNC
1	1	1	3
1	2	2	6
1	3	4	12
1	1	1	5
1	2	2	10
1	4	4	20
1	1	1	7
1	5	2	14
1	6	4	28
1	1	1	11
1	5	2	22
1	7	4	44

PNC matrix

12	20	28	44
6	10	14	22
3	5	7	11

database

Fig. 4 Tables in database

4 Operations for Index Technique with Prime Number Coordinate

In this section, we focus on different operations using PNC. They are search, update, insert and delete operations.

4.1 Search operation

In this paper, we aim to build to an index for relationship and position searching on XML tree. We make use of PNC feature that is consisted of relationship calculation and depth calculation during index.

We use examples to illustrate these queries and then show how to use the proposed index technique and the relational database schemas to process these containment queries. There are two type of containment query.

1. ***Direct Containment Query***: A query consisting of direct containment relationships (parent–child relationships) among elements, attributes, and their contents.
2. ***Indirect Containment Query:*** A query consisting of indirect containment relationships (ancestor –descendant relationships) among elements, attributes, and their contents.

Example 3. Figure 5 describes the sample SQL statement for expressing direct containment relationship query where 'handphone' for parent node, and 'LG' for child node.

Query: *handphone/LG*

```
select  a.docid, a.eid, b.eid
from    XMLIndex a, XMLIndex b,
        ElementID c, ElementID d
where   c.EName = 'handphone'
        and d.EName = 'LG'
        and a.EID = c.EID
        and b.EID = d.EID
        and a.DocID = b.DocID
        and c.eDocID = d.eDocID
        and a.docid = c.edocid
        and b.PNC = a.PNC*2
```

Fig. 5 Direct containment query

Example 4. Figure 6 describes the sample SQL statement for expressing indirect containment relationship query where 'handphone' for ancestor node, and 'style' for descendent node.

Query: *handphone//style*

```
Select  a.DocID, a.EID, b.EID
from    XMLIndex a, XMLIndex b,
        ElementID c, ElementID d
where   c.EName = 'handphone'
        and d.EName = 'style'
        and a.EID = c.EID
        and b.EID = d.EID
        and a.DocID = b.DocID
        and c.EDocID = d.EDocID
        and a.DocID = c.EDocID
        and b.PNC/b.Enum = a.PNC/a.Enum
```

Fig. 6 Indirect containment query

4.2 Insert operation

For node insertion, ancestor nodes need be added to new prime numbers. Figure 7 shows pseudo algorithm for changing index structure in insert operation.

```
■ Pseudo algorithm for insert operation
1. Get EID of node to be inserted
2. Put EID node to be inserted into stack
3. Get the PNC value of node to be inserted
4. Find parent nodes
5. If ( find the EID having PNC/2) goto step 2
       then goto step 6 (found node is root node)
6. Create the variable named newPrime and initialized to
   new prime number
7. Give the newPrime*2 to every pulled node from stack
```

Fig. 7 Pseudo algorithm for insert operation

Example 5. Figure 8 shows we insert an element ' H' below element 'C'. And we needn't change the others in PNC matrix.

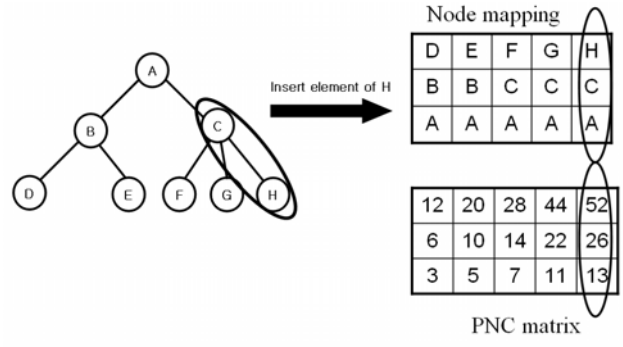

Fig. 8 Insert operation

4.3 Delete operation

We suppose that delete operation is occurred in only leaf node. If delete operation in leaf node, we must know ancestor nodes using the PNC value. The delete operation of nodes processes using pseudo algorithm in figure 9.

```
■ Pseudo algorithm for delete operation
1. Get removing node's PNC
2. Find parent nodes
3. If (find the EID having PNC/2)
       delete the PNC value of node on XMLIndex and
       goto step 2
   else end. (until root node)
```

Fig. 9 Pseudo algorithm for delete operation

Example 6. Figure 10 shows we delete an element ' E' below element 'B'. And this operation doesn't affect on the others in PNC matrix.

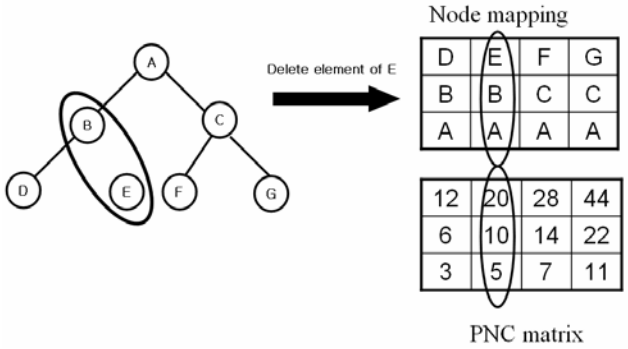

Fig. 10 Delete operation

4.4 Update operation

The update operation is used to modify the content of a leaf node on XML tree. When an update occurring in a leaf node, we change the value of text table and needn't change the others in PNC matrix and XMLIndex table. The update operation processes with pseudo algorithm in figure 11.

■ **Pseudo algorithm for update operation**
1. Get the changed node's EID
2. Change the content of the node in text table

Fig. 11 Pseudo algorithm for update operation

5 Performance Evaluations

In this section, we compare ARC with PNC in order to prove PNC algorithm is efficient in query processing. Proposed algorithm reduces the workload to perform the comparative and update operation when it retrieves and indicates data in the relational tables.

5.1 Theoretical Evaluation

5.1.1 Comparative evaluation for containment query

In ARC it uses two position values, which are relevant to start and end position, to represent the positional region of the node. In the method, it should compare the pairs of start point and end point of the two nodes to check the containment relationship between the nodes. In the proposed index technique, it can perform containment relationship query along the little amount of comparative operation because it compares just containment relationship of PNC between ancestor node and descendent node. Figure 12 shows the difference of comparison times for verifying the containment relationship between conventional coordinate-based index technique and proposed index technique.

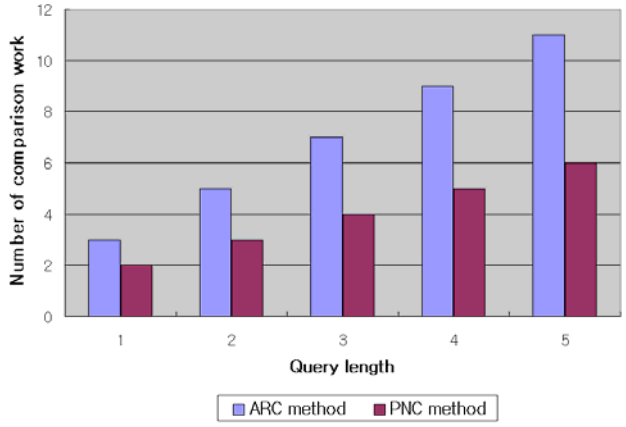

Fig. 12 Comparative evaluation for containment query

5.1.2 Comparative evaluation for updating

On the XML tree, according to growing the depth and the width, the number of nodes is to be increased in a geometrical progression. In the situation that update of an offset data affects other nodes by the insertion of a node. In the worst case, all of the position data should be reconstructed and it is caused lots of cost for the operation.

Let XML tree is balanced tree, depth is from 0 to k, and number of sibling nodes is s, in case using coordinate-based index and proposed index, the accumulative number of the node of which offset data is changed from the insertion and update operations are as follows. Figure 13 shows the comparison of the number of update operation per node in accordance with increasing the number of node among the different index techniques. In case of RRC, ARC insert operations are too heavy. But PNC method only changes a text field value.

$$\text{ARC method} = \frac{1}{2}\prod_{j=1}^{k} s_j \left(\sum_{m=1}^{k}\prod_{j=1}^{m} s_j + k + 2\right)$$

$$\text{RRC method} = \frac{1}{2}\prod_{j=1}^{k} s_j \left(\sum_{j=1}^{k} s_j + k + 2\right)$$

PNC method = constant

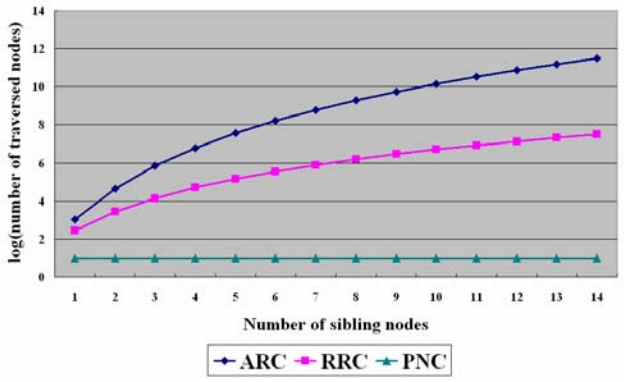

Fig. 13 Update evaluation on leaf nodes according to increase the number of sibling nodes with depth is 5

5.2 Experimental Evaluation

For experimental evaluation, we use computer having Intel Pentium 4 3.0GHz CPU and 1GB DDR memory**.** XML documents for experiment consist of 12 XML files that have various length, number of child nodes, depth of XML tree, and form balanced tree.

5.2.1 Comparative evaluation for running time

We use a program to compare their running time. In this experimentation, we prepare 10 queries showed in the figure 14 and figure 15 shows their results. The experimentation shows running time of PNC algorithm is shorter than that of ARC algorithm because of less comparison explained in chapter 5.1.1.

No	Query statement	No	Query statement
1	book // author	6	Author // last name
2	Info // content	7	Books // paragraph
3	Books //title	8	Handphone // lg
4	Elements // info	9	Doc // author
5	Element // author	10	Element // title

Fig. 14 Queries

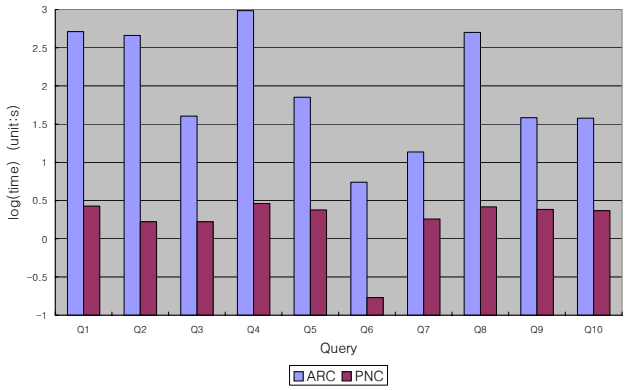

Fig. 15 Results of running time

5.2.2 Comparative evaluation for space size

In database because of relationship, the used space of PNC tables is bigger than the space of ARC tables on the same XML file owing to the minimum redundancy for representing XML tree structure. Figure 16 shows their size of used space in database. However PNC's size does not effect efficient query processing.

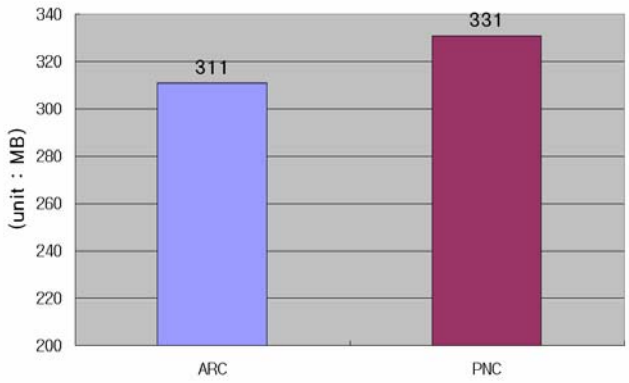

Fig. 16 Size of space in database using ARC and PNC

6 Conclusion

We proposed an index technique with prime number coordinate to express relative relationship among the nodes in a XML document. It is an efficient index technique that simplifies the comparative objects applied to search query as containment query with relational database.

Compared to traditional index method, PNC is an efficient index technique that simplifies and comparavtive evaluation applied to a query and minimizes the reconstruction of index structure by updata operation. The running time in query is reduced. And The performance of insert, update and delete opertation is better.

References:

[1] T. Bray, et al, "Extensible Markup Language (XML) 1.0", http://www.w3.org/TR/2000/REC-xml-20001006

[2] ISO, "Information Processing – Text and Office System – Standard Generalized Markup Language (SGML)", *ISO/IEC 8879*, Oct. 15, 1986

[3] R. Davis, T. Dao, J. Thom, J. Zobel, "Indexing documents for queries on structure, content and attributes", *International Symposisium on Digital Media Information Base (DMIB '97)*, Nov. 1997.

[4] D. Kha, M. Yoshikawa, S. Uemura, "An XML indexing structure with relative region coordinate," *ICDE'2001*, April 2001.

[5] J. Yoon, V. Rahgavan, and V. Chakilam, "BitCube: Clustering and Statistical Analysis for XML Documents", *13th International Conference on Scientific and Statistical Database Management*, Virginia, July 2001.

[6] J. Yoon, V. Rahgavan, and V. Chakilam, "BitCube: A Three-Dimensional Bitmap Indexing for XML Documents", *Journal of Intelligent Information System*, Vol.17, 2001, pp.241-254

[7] Tatarinov, S.D. Viglas, K.Beyer, J.Shanmugasundaram, E.Shekita and C. Zhang, Storing and Querying Ordered XMl using a Relation Database System, *ACM SIGMOD*, 2002