IMPLEMENTATION OF INDEX SCHEMA FOR XML DOCUMENTS BASED ON STRUCTURE OF DATABASE

Youngrok Song, Kyonam Choo
Department of Information and Telecommunication Engineering, University of Incheon
177 Dowha-Dong, Nam-Gu, Incheon 402-749, Korea

Yoseop Woo, Hongki Min
Department of Information and Telecommunication Engineering, University of Incheon
177 Dowha-Dong, Nam-Gu, Incheon 402-749, Korea

Keywords: Index Schema, XML, Database.

Abstract: In this paper, structural information between tree nodes is presented without any structural changes in tree by converting number information added to tree into bit streams. It is also shown that other structural information can be retrieved and added to index schema, and that in query schema it is possible to restore host nodes by exploiting the given node information in case relative route query expressions as well as absolute query expressions are given. It has an advantage of making derived query expressions through a query. In addition, in the query-processing process, response time can be minimized by conducting bit operation between bit streams with index schema and query schema in use, and accurate results can be reached by searching only with information of record set by node in index files.

1 INTRODUCTION

Because of its simplicity and flexibility, XML is rapidly becoming the most popular format for information representation and data exchange on the web. It has become more difficult to locate information needed, so methods to search and manage XML document information more efficiently are necessary (Dao, 1998), (Milo and Suciu, 1999) and recently studies have been actively under way to store XML document information in storage media such as database(Zhang et al, 2001), (Chien et al, 2002), (Yoshikawa et al, 2001).

These studies aim to support efficient route search for a single large volume XML document or several XML documents with the same structure.

Thus, in case of using the aforementioned indexing techniques to find the wanted routes from several XML documents with different structures, we need to compose each index for each XML document and investigate every index. In addition, in case the route includes ancestor-descendant relationship, search performance becomes bad as the wanted route is located only by visiting every node of the established index. Though indexing methods have been suggested to complement the defects, there is still the drawback of lower performance in case the ancestor-descendant relationship in case an ancestor-descendant relationship appears in the middle of the rout, not in root (Chung et al, 2002).

At the same time, an integrated indexing method (Zhang et al, 2001), (Chien et al, 2002), (Yoshikawa and Amagasa, 2001) is also presented which enables us to search XML documents with different structures. It has adopted the way of reflecting and expanding structural characteristics of XML in the inverted index opposite to the term basis used in the information search field.

However, the index techniques suggested in Zhang et al (2001) and Chien et al (2002) face lower search functions as increased number of documents brings about index data to be compared and searched, Yoshikawa and Amagasa (2001) show the disadvantage of lower search performance with more documents. The characteristics serve as a factor that makes it difficult to apply them in the situation route search for XML documents is needed.

Song Y., Choo K., Woo Y. and Min H. (2007).
IMPLEMENTATION OF INDEX SCHEMA FOR XML DOCUMENTS BASED ON STRUCTURE OF DATABASE.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 402-405
DOI: 10.5220/0001278304020405

Therefore, query search is made possible through operations between bits by avoiding complicated structure search and using stream-based index algorithm. As well, in this article, query processing in diverse forms is given flexibility by adding structural information field needed for the structure of existing index files, if additional structural information is added to XML documents. As a result, the given query analysis time is reduced and thus response time to output return is reduced by using query schema based on index schema in order to improve query processing efficiency.

2 INDEX SCHEMA BASED ON DATABASE STRUCTURE

2.1 Index Schema

This paper has gone through the following processes to make index schema:

(1) Build up XML documents into trees by using DOM (document object model) trees.
(2) Give sequential numbers to the built-up trees by node of each level.
(3) Trees given numbers are rebuilt-up.
(4) Data to be stored in index files are acquired from the trees.
(5) Make bit streams by using the number given to each node.
(6) Store the data gained from [(4),(5)] in index files.

2.1.1 XML Tree Numbering Technique

Figure 1 is a DTD used for shipments between buyers and sellers.

```
<!ELEMENT deliveries (delivery*) >
<!ELEMENT delivery (sender, receiver) >
<!ELEMENT receiver (name*, item) >
<!ELEMENT name (#PCDATA)>
<!ELEMENT receiver (item*) >
<!ELEMENT item (item*, payment, price, raddress)>
<!ELEMENT raddress (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST item iname CDATA #REQUIRED
fee CDATA #IMPLIED
icode CDATA #IMPLIED
dn CDATA #REQUIRED
amount CDATA #IMPLIED>
<ellipsis>.....
```

Figure 1: DTD of shipping XML document.

Figure 2 the node of each level is sequentially given a number, and bits which can express each node are assigned in as the same number as that of the node of each level, among the numbers given to nodes of trees, those starting with bit 0 are excluded and the numbers should be given from bit 1. In this way, sequential numbers are given from the root node to the least significant nodes. After the operations are completed, the only bit stream for each route is generated when the assigned numbers are connected with the root node as the reference point. The value itself shows the super-sub relationship, the structural information of the entire node in the same tree such as parent-children, ancestor-descendant and sibling nodes from the root node up to the node where the route ends.

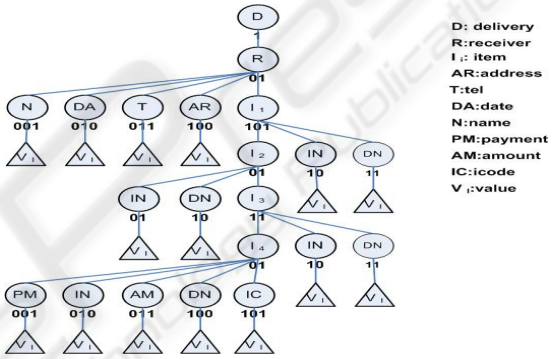

Figure 2: Tree of shipping XML document given bits.

2.1.2 Technique of Converting Bit Stream of XML Structural Information

Figure 2 shows that bit stream value given to each node is gained by successively visiting each from the root node, and with this bit stream, the bit value by level is stored from the least significant bit in the assigned fixed bit space. The bit values from L0 to L7 expressed in Figure 3 show the bit stream value concerning hierarchical relationship by node in Figure 2. The bit stream values correspond with all the nodes one to one, and they are the only and unique values. As well, as the value is the only value, the whole XML document can be restored if the bit value by each node name is known. The entire size of the bit stream is 64 bits, which means that the fixed space is assigned. Yet, if the level of an XML document goes up or the node number by level increases, the value will exceed the fixed bit of 64 bit. In that case, it can be solved by assigning a space bigger than 64 bits. Even so, the whole algorithm for the bit stream is not affected. It has the advantage of expanding the bit stream without changing the index schema when a number of

documents should be processed. The bit stream gained from the processes is again converted into hexadecimal forms, and it can be stored in each field of database index schema.

0_i = i times repetition of bit '0'

Level	L_7	L_6	L_5	L_4	L_3	L_2	L_1	L_0
Number of entire bit	← 32 bit →							
Bit stream per each node	0_{25}	0_3	0_2	0_2	0_2	0_3	0_2	1
							01	1
						001	01	1
						100	01	1
						101	01	1
					01	101	10	1
					10	101	01	1
				01	01	101	01	1
				11	01	101	01	1
			01	11	01	101	01	1
			10	11	01	101	01	1
		001	01	11	01	101	01	1
		010	01	11	01	101	01	1

Figure 3: Bit stream assigned to each node.

2.1.3 Building up Index Schema Based on Database

In index schema table, names of each node, unique bit stream values, bit stream values of parent node, and level values needed for query analysis when user's queries come up are stored. The roles of the schema structure of index files are defined as follows:

(1) N_name field: names of XML tree nodes
(2) B_value field: bit stream values of each node
(3) Tb_len field: certain node length of the whole bit stream
(4) B_len field: bit stream length assigned temporarily to the present node
(5) Level field: the level of the present node on XML tree
(6) P_value field: bit stream value of the parent node of the present node
(7) Ctype field: It shows if the present node is a primitive or an attribute.
(8) Position field: It shows the order of the present node as a child node to the parent node.
(9) D_num field: the document no. of XML document
(10) Data field: the value each node has

3 EXPERIMENTS AND RESULTS

The accuracy test was conducted about the general query types by using 1000 all different XML documents, and another test was carried out to compare performances between the experiments of existing XRel(Yoshikawa and Amagasa, 2001) and INRIA(Floresc and Kossman, 1999) and the methods suggested in this article.

3.1 Accuracy Test of Search Results in Addressing Queries

In this paper, the experiment data in Table 1 were built up to conduct an accuracy test for query performances in an XML index system.

The index algorithm suggested by Chien et al (2002) focuses on the index techniques centering on three points. That is, to process XML queries effectively, it takes I) quickly testing the structural relationship of ancestor-descendant (parent-child) between the two given primitives II) quickly finding out the candidate list that satisfies the structural relationship in I), and III) effectively drawing the pairs that satisfy the structural relationship from the candidate list, the result of II).

This paper also suggested the algorithm concerning the new structure search about I) and II) mentioned above. In stage III), Chien proved its efficiency with B+ trees. Yet, this paper presents a more efficient method of storing and searching the index algorithm that suits the processes of I) and II).

As the queries are already optimized after the processes of I) and II), the result of one to one corresponding from database in Stage III). However, in compared with Chien's method, this paper shows a slight difference. As Chien concentrates on structural relationship index, he does not consider the characteristics of storage media such as database. Rather, he simply stores and searches the structural information with B+ tree. However, there is a close relationship between index structure and storage structure in this paper, it is not appropriate to be compared with Chien's method which indexes only XML documents.

Table 1: Experiment data for accuracy evaluation.

MEAN	VALUE
Num. of entire documents	1000
Average size of documents	10KB
Average num. of elements per document	128
Average num. of attributes per documents	98
Average depth of element	5.24
Average num. of keywords per documents	226
Average num. of child nodes of element	4.29
Average value of K in index based on K-ary	10
Num. of entire elements	128,942

The index and query schema about XML documents suggested in this article was conducted 1000 times by repeating the queries with the XPath expression depth of 10 by using experiment data as in Table 1, and the results are in Figure 4. As shown in Figure 4, the accuracy was more than 94%, though the number of primitives, attributes and texts in expressions increases. The error rate was less than 6%, the reason is that with increased complexity of documents, the number of nodes by level in XML document trees increases. Thus, as the length of bit stream that can express structural information of each node exceeds 64 bits, query processing operations fail, otherwise inappropriate results were returned though query processing was successful.

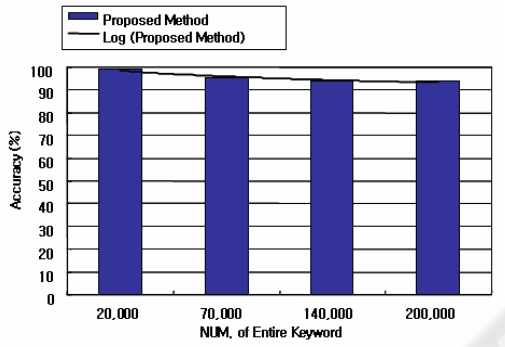

Figure 4: Accuracy of retrieval result.

3.2 Response Time Experiment of Query Processing

To experiment response time experiment of query processing, queries and Shakespeare's plays (Bosak Shakespeare Collection) used in INRIA and XRel's comparison experiment in Yoshikawa and Amagasa (2001) were adopted for experiment data and queries. The queries have different complexity level as it goes up from Q1 to Q6.

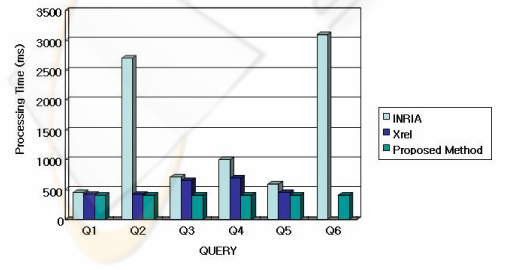

Figure 5: Response time of retrieval result.

As shown in Figure 5, the join operations were made to search the results of the given queries for the methods the existing INRIA and XRel suggested, but the queries were already converted into optimized forms before index file access in the query method suggested in this paper, there was a big difference from the operation results of INRIA and XRel since there were no join operations between tables that occur database access.

4 CONCLUSIONS

The index and query schema is the method of searching structural information and data of XML documents and storing it in the storage media such as database which can manage a lot of information.

However, when XML documents are changed, the numbers given to XML document trees should be changed too. It means that there are advantages: one is that bit stream values should be changed in that case and the other is that search is possible after bit streams are completed by visiting all the nodes of XML documents. Again, another disadvantage is that search time increases as index fine is bigger. Therefore, future studies are needed for the research about dynamic models of bit streams without any changes in bit streams even in case of changes in XML trees, and about the search techniques that makes it possible to search structures without visiting all the nodes of XML documents trees. In addition, another study about expanding index file structures is also necessary to cope with the problem that database table becomes big as structural information needed for XML document indexing increases.

REFERENCES

Tuong Dao, 1998. An Indexing Model for Structured Document to Support Queries on Content, Structure and Attributes. In *Processing of IEEE ADL*. pp. 88-97.

Tova Milo, Dan Suciu, 1999. Index structures for Path Expressions. In *ICDT*. pp. 277-295.

C.Zhang, J. Naughton, D. Dewitt, etc, 2001. On Supporting Containment Queries in Relational Database Management System. In *ACM SIGMOD*.

Shu-Yao Chien, Zografoula Vagena, Donghui Zhang, etc, 2002. Efficient Structural Joins on Indexed XML Documents. In *VLDB*. pp. 263-274.

Masatoshi Yoshikawa, Toshiyuki Amagasa, 2001. XRel : A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases. In *ACM TOIT*. pp. 110-141.

Chin-Wan Chung, Jun-ki Min, Kyu-seok Shim, 2002. APEX : An Adaptive Path Index for XML Data.
In *SIGMOD*. pp. 121-132.

D. Floresc, D. Kossman, 1999. A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database. In *Technical Report of INRIA*.