Path-based XML Relational Storage Approach

Qi Wang¹, Zhongwei Ren¹, Liang Dong² and Zhongqi Sheng²

¹School of Mechanical Engineering
Shenyang University of Technology
Shenyang 110178, China

²School of Mechanical Engineering and Automation
Northeastern University
Shenyang 110004, China

angelwangqi@sina.com and rzw9811210@163.com
dongliang902@126.com and zhqsheng@mail.neu.edu.cn

Abstract

As an important medium for the description of product information and data exchange, XML (eXtensible Markup Language) is widely used in the network supported collaborative design. How to store and query XML data will directly affect the performance of collaborative design system. Compared the advantages and disadvantages of existing XML data storage technology, a new path-based XML data storage method using relational database is proposed. This paper gives the specific storage mapping algorithm. The different XML documents are stored using a fixed relationship model and not limited by the document DTD (Document type definition). The XML data stored using the method have unique advantages such as simple structure, small volume and easy to query processing. Good data processing is helpful for improving the efficiency of collaborative design system.

1. INTRODUCTION

This work is supported by the Fundamental Research Funds for the Central Universities (N090403005) and Funds for Engineering Technology Center of Liaoning Province (2009402017).
The network supported collaborative design opened up a new domain for engineering design and established a new work mode for people in the era of information, which was grouped, distributed, interactive and cooperative[1]. The information sharing and exchange among members of a distributed design team is an essential and fundamental part in network supported collaborative design. With the development of internet technology, data processing is becoming one of the most important research fields in network supported collaborative design.

Because of the characters of XML, which includes Platform independence, easy for expansion, better interaction, abundance semantic, well formatted, XML has been used to describe the product data and already become an emerging standard for information exchange in the network supported collaborative design[2]. But XML is a self-defined markup language, it needs a criterion to define the element, attribute and the data type of the XML document, so that XML document can be exchanged and shared among cooperation members.

The purpose of this paper is to realize the XML data storage more effectively in collaborative design.

2. PATH-BASED XML RELATIONAL STORAGE

There are three possible approaches to store semi-structured data (i.e. XML documents) and to execute query on that data [3]. The first, builds a special-purpose database system. Such a system is particularly tailored to store and retrieve XML data, using specially designed structures and indices and particular query optimization techniques. The second, uses an object-oriented database system. In this approach, the rich data modeling capabilities of OODBMSs are exploited. The third, uses a (standard) relational database system. In this approach, XML data is mapped into tables of a relational schema and queries posed in a semi-structured query language are translated into SQL queries.

It is still unclear which of above three approaches is going to find wide-spread acceptance [4]. In theory, special-purpose systems should work best, but it is going to take a long time before such systems are mature and scale well for large amounts of data. Likewise, the current generation of object-oriented database systems is not yet mature enough to evaluate queries on very large databases. At present, Because of the merits of relational-database, which includes mature technique, abroad application, easy for expansion, better interaction, abundance semantic, well formatted, well ability for control data, security [5], so XML data-processing which is based on relation, is obviously a feasible and prospective way among all ways of storing and managing XML data.

2.1 XML Relational Storage

Relational databases, however, have been built to support traditional (structured) data and the requirements of processing XML data are vastly different from the requirements to process such traditional data. To achieve storing of XML data in the relationship database, an entire XML document must be separated into multiple related tables.

Relations storage strategy on XML can be divided into two categories: Structure-mapping approach and Model-mapping approach [6]. Relationship models are built based on the structure (DTD or XML Schema) of the XML documents using Structure-mapping approach. Different XML document has a different relationship model. So, the size and number of the mapping relational table will vary with the XML schema. This will make it difficult for the XML data querying. In Model-mapping approach, the nodes and sides of the XML document tree are mapped to the relationship model. The relationship model obtained has nothing to do with the structure of the XML documents. That is to say, this method is to use a fixed relationship model to store all of the XML document structure. This method is suitable for the XML data management and queries.
The path-based XML relational storage approach is a model-mapping approach. Various XML
document with different structures are stored using a fixed relationship model. This relationship model
does not consider the documentation DTD information. Therefore the relationship model will not be
limited by documentation DTD information. The mapping relational tables obtained by path-based
relational storage approach have many features, such as syntactic frozenness, simple structure and small
in number. The important characters of the approach make it easier for XML querying that we do not
need to establish the index structure similar to B’-tree [7]and R-tree [8]. Next, we will describe the
internal mechanisms of this storage method. An elaboration of this storage approach is offered in the
following sections.

2.2 Path-based Storage Structure

PreOrder Traverse is the first step of the path-based XML relational storage approach. The result of
this step is to extract the whole useful information of every node in the XML tree and give the unique
identifier for each. The all information of each node, side, value and their nesting are recorded in two
tables: Value_Table and No_value_Table.

Value_Table is used to store the information that is relevant to the elements (or attributes) of the XML
document which have the value. In order to obtain the entire record of the relevant information of all of
the elements (or attributes), each item of Value_Table must to be set of fields with six columns that
identifies the individual records of every node.

The values stored in the six columns are:
- id: the sequence number of node, which reflect the order of their appearance during the PreOrder
  Traverse of document tree;
- name: the name of the element(or attribute);
- value1: the value of the element(or attribute);
- path: record the path from a root node to the present node, the path is composed of a sequence of
element linked by “/”;
- parentid: the id of the parent node of the element(or attribute);
- level1: the layer number which the node located.

In Value_Table , the elements and attributes are treated equally. The relevant information of all of the
elements (or attributes)which have value is stored entirely and without any loss.

No_value_Table is used to store the information that is relevant to the elements (or attributes) of the
XML document which have no value. That is to say, it records all relevant information of the non-value
intermediate nodes (or root node). No_value_Table must include four columns (i.e. four fields): id, name,
parented and level1. The content stored in these four columns is the same meaning as in Value_Table.
Non-value-table settings make it sure that the root element and the non-value intermediate elements of the
XML document have been documented.

In the two tables, the primary key is id whose content is not allowed to be empty.

To sum up, this storage mapping method takes account of the overall structure of an XML document
both and the information of the nodes and edges. It will be a combination of these three areas, which
makes XML information intact, and enables XML data more convenient query.

2.3 Mapping Algorithm of Path-based Storage

We will introduce the realization process of Path-based relational storage approach in detail. First of
all, we establish a database for user in SQL Server. The database has two tables, named Value_Table and
No_Value_Table. Each column of the tables has an attribute. Performing PreOrder Traverse to the XML
document tree, we can obtain the PreOrder number, name, father-node id and the layer number of each
element/attribute. All of this information and the path of elements/attributes which have value will be
stored in corresponding tables. The store conversion process comes to the end when the action of
PreOrder Traverse to the XML document tree is over.

This mapping algorithm of path-based storage is as follows:
Input: the parsed XML document tree
Output: the XML document tree is stored in relational tables named Value_Table and No_Value_Table

Steps:
    InitStack(&s1); /*initialize the stack s1*/
    p=head; /*set the root node of XML document tree to the current node*/
    int id=1, level1=1, pid=0;
        /*defines three integer variables recorded separately the current node id, number of layers, the
         parent node id */
    Push(&s1, p);
    CString sPath="/"; /* defines a string variable sPath to record the path of the current node */
    sPath+=p->name; /* save the intermediate nodes the path passing */
    while (if the stack is not empty)
        {while (the current node has children nodes)
            {p=p->firstchild; /* according to the order of preorder traversal, move the pointer;
                p for the current node */
                if (the current node has no children nodes)
                    */ the node is an element / attribute has value */
                    {
                        id, p->name, pid, level, p->value, path of the current node is stored in the Value_Table in all
                        appropriate fields;
                    }/*endif*/
                else
                    {
                        id, p->name, pid, level of the current node are stored in No_Value_Table in all appropriate
                        fields;
                    }/*endelse*/
            }/*endwhile*/
            Push(&s1, p);
        }/*endwhile*/
    Pop(&s1, p1);/* stack element out of the stack, and set to the current node */
    sPath=sPath.Left(sPath.ReverseFind('/'); /* back up the path layer */
        /*to determine the existence of sibling node of the current node */
    if (sibling node exists)
        {p=p1->nextsibling;/* then set its sibling node to be the current node*/
            if (the current node has no children nodes)
                {record all of the information of the current node in the corresponding fields of the Value_Table;
                    }/*endif*/
            Else (the current node has children nodes)
                {record all of the information of the current node in the corresponding fields of the
                    No_Value_Table;
                    }/*endelse*/
                Push(&s1, p);
        }/*endif*/
else sibling node does not exist
  {return to its upper node, continues to traverse;}
}/*endwhile*/

3. CONCLUSIONS

Through in-depth research and discussion for the XML data store, this paper presents a novel use of
relational databases to store the XML data. The nodes have text values and the non-text value nodes in the
XML documents tree are stored in two relational tables. This method does not concern documents DTD
schema information, nor does it require the creation of any index structure. Using the mapping algorithm
of path-based storage, the user may save lossless the XML documents to the fixed pattern relational table
which is very convenient to the XML data query.

References

27-34, 1999.
Chinese), Shanxi.
[7] Lu Yan, Zhang Liang, Duan Qiyang, “DTD-based XML indexing,” Journal of Computer Research and Development, vol. 01,
pp. 21-26, 2005.