Development of Embedded System for storing and retrieving XML data *

Joonho Kwon[1], Dongseop Kwon[1], Hyoseop Shin[2], Sukho Lee[1]

[1] School of Electrical Engineering and Computer Science
Seoul National University, Seoul, Korea
{bluerain, subby}@db.snu.ac.kr, shlee@cse.snu.ac.kr
[2] Software Center, Samsung Electronics, Co., Ltd., Seoul, Korea
hyoseop@samsung.com

Abstract. This paper describes the design and implementation of an embedded-type XML storage and retrieval system which is built on top of relational databases. The proposed system stores each node of input XML documents as a relational tuple by a node numbering scheme. User queries written in XQuery are translated into a set of simple SQL queries before execution so that an embedded-type relational engine with limited resources can handle those queries more efficiently. XML type handling mechanism of the system enables it to cover various kinds of XML applications. Futhermore, the system provides an efficient XML indexing method for processing twig queries.

1 Introduction

In this paper, we design and implement an embedded system, the eXDM system, based on RDBMS and XQuery in order to store and retrieve XML data representing TV programs. Since XML becomes the de facto standard for representing, exchanging and accessing data over the Internet, several techniques have been proposed for storing XML documents into tables and for processing an XML query after translating it into SQL queries over these tables. The edge approach[1] and the numbering scheme approach[2][3] have been proposed for storing XML documents in tables. To retrieve XML documents, several query languages such as XML-QL, XPath, and XQuery[4] have been proposed. Among them, XQuery is the first public working draft of XML query language from the World Wide Web Consortium(W3C). XML document repositories that support XQuery such as eXceleon and Tamino store XML documents in native XML database.

2 eXDM system

2.1 Overview of Architecture

eXDM consists of four components : type handler, index handler, data loader, and XQuery processor.

* This work was supported by the Brain Korea 21 Project.

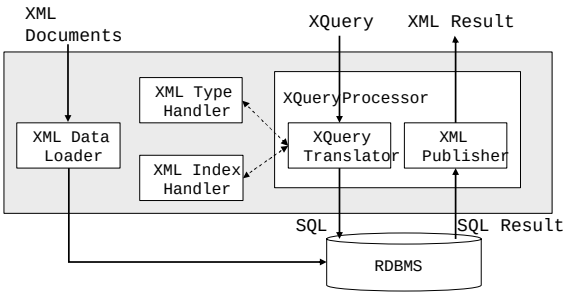

Fig. 1. eXDM Architecture Overview

The XML data loader parses XML documents and stores the extracted elements, attributes or values into corresponding tables. The XML type handler provides an ability to specify data types for element and attributes. The XML index handler manages index information of elements and/or attributes which is based on regular path expression. The XQuery processor has two roles. The first role of the XQuery processor is to translate an user query in the XQuery grammar to several SQL statements. The translated SQL statement is passed to SQL engine. Then SQL engine executes the SQL statement and sends the results to XQuery processor. The second role of XQuery processor is to publish the relational results as XML data. In eXDM, a knowledge administrator familiar with the XML documents can provide type information and create indexes. Fig.1 shows the overall architecture of eXDM.

2.2 The Scope of the Query Language

In the eXDM system, the simplified syntax of XQuery is used. The eXDM system supports only four clauses FWRS(FOR, WHERE, RETURN, SORTBY). But it is enough to construct most of queries which users would want to use. And the XQuery of eXDM also supports logical operators, arithmetic expressions, regular path expressions and constructor.

```
FOR $pi in //ProgramInformation,
    $be in //BroadcastEvent
WHERE $be//PublishedTime >= '2002-08-21 13:00:00'
  AND $pi/@programId = $be/@crid
RETURN $pi//Title/text(), $be//PublishedTime/text()
SOBTBY $pi//Title/text()
```

Fig. 2. XQuery example

Fig.2 shows a typical XQuery example to request the title and time of programs that are broadcasted after 1 P.M. on 2002-08-21.

2.3 The Schema of Tables

In order to store XML documents in relational tables, eXDM uses an edge approach[1] and a numbering scheme[2][3] together. It is possible to store XML documents in a system without DTD or XML schema as well as to store various structures in a system. The proposed numbering scheme quickly determines the ancestor-descendant relationship between elements in the hierarchy of XML data.

XML documents can be mapped into the following tables:

$NODE$ (n_id, d_id, name, v_type, v_id, start_pos, end_pos, level)
$DOCUMENT$ (d_id, contents)
$VALUE_{type}$ (v_id, value)

The $NODE$ table stores elements and attributes of XML documents, while the $DOCUMENT$ table stores original XML documents. The actual values of elements and attributes are stored in the corresponding $VALUE_{type}$ table.

2.4 Query Splitting

As is mentioned above, the XQuery processor translates XQuery into SQL. A simple approach is to translate an XQuery statement into one SQL statement. But in this case, the translated SQL has many self-joins. A join is frequently the most expensive physical operation. In general, the embedded SQL engine has a limited memory and a computing power. Thus it takes a long time to execute the SQL statement which has many joins in embedded SQL engine.

So XQuery processor uses the splitting method. XQuery processor translates a given XQuery statement into several simple SQL statements. A simple SQL statement means that it has only one join operation. The results of a simple SQL statement are stored in a temporary table. Thus the intermediate results are reused by another simple SQL statement. After the final simple SQL statement is executed, all temporary tables are deleted.

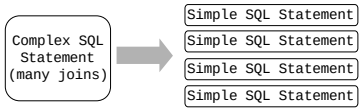

Fig. 3. Concept of Splitting Method

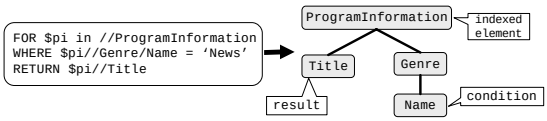

Fig. 4. Twig Query

2.5 Index

Users can create indexes on elements and/or attributes frequently used in XQuery. The eXDM system can use this index instead of generating several temporary tables during the query splitting. For example, the regular path expression in where clause in Fig 4. is $pi//Genre/Name = 'News'. If there is no index, this path expression is translated into four simple SQL statements. Three statements express ancestor-descendant relationship between elements, and one statement expresses a condition to be satisfied(Name = 'News'). If there exists an index on $pi//Genre/Name that satisfies the condition, only one simple SQL statement is generated.

Especially, the usual form of the XQuery is a twig shape like Fig 4. The index structure of the eXDM is also efficient to process these queries.

2.6 Type Handling

In the eXDM system, users provide type information about XML documents. For example, users can define a 'PublishedTime' element as datetime. This allows comparisons of datetime data and corresponding retrieval options such as "after 1 PM. on 2002-08-21". With no schema information, elements and attributes are treated as string.

3 Conclusion and Future Work

We have developed an embedded XML storage and retrieval system(eXDM). The eXDM system stores XML documents on relational tables using the edge approache and the numbering scheme. It is possible to process regular path expression queries because users can use XQuery as a standard query interface. For efficient execution of SQL translated from XQuery, eXDM uses splitting methods and provides index mechanism.

In the future work, we can adopt an XML schema as type information and extend our system to support full specifications of XQuery.

References

1. Daniela Florescu and Donald Kossman, Storing and Querying XML Data using an RDMBS. IEEE Data Engineering Bulletin, 22(3):27-34, 1999.
2. C. Zhang, J. Naughton, D. Dewitt, Q. Luo, G. Lohman, On Supporting Containment Queries in Relational Database Management Systems, Proceedings of SIGMOD, May 2001.
3. Q. Li and Bongki Moon, Indexing and Querying XML Data for Regular Path Expressions, Proceedings of VLDB, 361-370, September 2001.
4. S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. XQuery 1.0: An XML Query Language. W3C Working Draft. Available from http://www.w3.org/TR/xquery.