A Development of an XML-OLAP System

Chantola Kit†, Toshiyuki Amagasa†† and Hiroyuki Kitagawa††
†Graduated School of Systems and Information Engineering
‡Center for Computational Sciences
University of Tsukuba, Japan
kchantola@kde.cs.tsukuba.ac.jp, {amagasa, kitagawa}@cs.tsukuba.ac.jp

1 Introduction
Since the Extensible Markup Language (XML) has become a de facto standard for data exchange and representation on the Web, the more complex ways to make analysis of XML data are considered to be extremely important in order to extract useful information from massive XML data. To answer to the need, we have been proposing an XML-OLAP system [1] which is based on relational databases for XML data analysis.

Recently, we have developed some algorithms for TOPOLOGICAL ROLLUP [2], a structure-based grouping in XML-OLAP, in the purpose of speeding up the computation. Based on Structural Join algorithm, our proposed algorithms are the straightforward technique, Top-Down Algorithm (TDA), Bottom-Up Algorithm (BUA), and BUA’s enhanced version, Bottom-Up Algorithm for Mixed Structure (BUA-Mix). In the succeeding attempt, we proposed more efficient Single-Scan by preorder/postorder algorithms, in which we could compute the structure-based grouping by a single scan over the stream of XML nodes.

In the followings, we are going to show the overview of XML-OLAP system, then show the proposed algorithms and their evaluation by performing several experiments with generated XML data varied by some features.

2 XML-OLAP System
This section describes our proposed XML-OLAP system based on relational databases. In Figure 1 (left side), according to the content of XML data, a user firstly gives a fact path and some dimension paths in XPath expression to denote his/her interest by which the system produces an XML cube. Then, the user can make analysis of XML data-cube using XQuery with OLAP extensions. We are based on relational databases because there are many off-the-shelf systems and they provide good performance.

In the right side of Figure 1, XML documents are translated into relational tables, node table and path table, which store path expression, preorder, postorder, value, etc. of every XML nodes. Then, the given fact and dimension paths are translated to SQL query to extract all fact and dimensions from relational tables and form XML data-cube. Finally, the given XQuery with OLAP extension, specifically TOPOLOGICAL ROLLUP, will also be converted into SQL query in order to extract the user query result from XML data-cube.

Since we had noticed that our TOPOLOGICAL ROLLUP computation by SQL consumed a lot of time, we proposed some efficient algorithms for TOPOLOGICAL ROLLUP which will be shown in the following section.

3 TOPOLOGICAL ROLLUP Operation
As we mentioned previously, TOPOLOGICAL ROLLUP is a specialized operation for XML data where a number of group-bys are computed according to the hierarchy of an XML data. Based on a well-known algorithm of Structural Join, we proposed several algorithms for TOPOLOGICAL ROLLUP.

Structural Join (STJ for short), also referred by [2], is an algorithm for finding the relationship between sets of XML nodes. This algorithm can help us to find nodes in the same group. STJ requires only two input lists: ancestor node list (AList) and descendant node list (DList).
and it outputs a list of ancestor-descendant pairs in the order of ancestor/descendant nodes.

Based on Stack Tree Join, one of Structural Join algorithms which uses stack for storing the intermediate ancestor-descendant pairs, we proposed several algorithms as the followings.

1. **Top-Down Algorithm (TDA)** requires multiple $ALists$ according to the number of hierarchies and single $DList$ of all values to be grouped. It starts grouping by STJ join $AList$ and $DList$ from the top to bottom level.

2. **Bottom-Up Algorithm (BUA)** is in the reversed process from TDA which starts from bottom to top level and $DList$ for higher level is from the joining result of lower level.

3. **Bottom-Up Algorithm for Mixed Structure (BUA-Mix)** is proposed to cope with the mixed structured of XML data which BUA did not consider. For each resulted $DList$ for higher level, BUA-Mix keeps descendant nodes which are out of ancestors’ scope. Doing so it will not miss the descendants that are at the same level to the aggregation nodes of $ALists$.

4. **Single-Scan by Preorder Number (SSC-Pre)** is proposed in addition to the straightforward algorithms, TDA and BUA. SSC-Pre requires only single $AList$ of all ancestor nodes in all hierarchies and single $DList$ of all descendant nodes where both lists are sorted in preorder number. By a single traversing through the tree of XML data, SSC-Pre uses stack to maintain the intermediate aggregation results and produced the complete aggregation value when it return to the top ancestor node.

5. **Single-Scan by Postorder Number (SSC-Post)** outputs the same result as in SSC-Pre, but both $AList$ and $DList$ are sorted in postorder number. SSC-Post starts scanning from the lowest level up to top level by reducing the cost of SSC-Pre traversing on XML tree circling from top to bottom and getting back to the top. It also enables us to partition XML data in case that the data size is very large.

4 Experimental Evaluation

Our experiments were performed in Sun Microsystems Sun Fire X4200 server whose CPU is in a 2-way Dual Core AMD Opteron(tm) processor (2.4GHz), with 16GB memory and runs Sun OS 5.10. We used Java version 1.5.0 to implement the algorithm, and PostgreSQL 8.2.6 to store XML data. Specifically, we used the path-approach which also mentioned in [1] to convert XML data to relational tables. Then, for the algorithms’ input, $ALists$ and $DList$, were extracted from the relational tables by SQL queries.

We used two kinds of synthetic XML data. One is XMark in which we computed aggregation according to the hierarchy and the sizes of XML data from 10MB to 1GB. Another data is the synthesized XML data by our data generator. It randomly generates XML data according to the predefined statistical values which produced the following sets of XML data: 1) XML data that do not have any text node being aggregated at the same level to aggregation nodes. This is to evaluate BUA, 2) XML data by changing the maximum depth, and 3) XML data by changing number of children for each node, thereby varying the width of XML data.

4.1 Experimental Results

The experimental results suggest that: 1) the proposed algorithms performs much better than the baseline implementation by SQL, 2) TDA and BUAs have similar performance trends, but BUAs performs slightly better than TDA due to its nature, and 3) the proposed algorithms SSC-Pre and SSC-Post perform better than TDA and BUAs for all cases, and SSC-Post has the best performance. As a consequence, we have a freedom to choose an appropriate algorithm out of SSC-Pre and SSC-Post considering the way how underlying data storage of the XML database is organized.

5 Conclusions

We described XML-OLAP system overview then proposed some efficient algorithms for TOPOLOGICAL ROLLUP based on Stack Tree Join algorithm. Our experiments with large collection of XMark and our generated XML data showed the effectiveness and proficiency of our proposed algorithms, especially SSC-Pre/Post. For the future works, we plan to propose a new algorithm for very large XML data aggregation by applying partitioning technique to our proposed algorithms. We also attempt to apply information retrieval on XML data.

参考文献
