OLAP Query Processing for XML Data in RDBMS

Chantola KIT1, Toshiyuki AMAGASA1,2, and Hiroyuki KITAGAWA1,2
1Department of Computer Science, Graduate School of Systems and Information Engineering
2Center for Computational Sciences
University of Tsukuba
I-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
kchantola@kde.cs.tsukuba.ac.jp, {amagasa, kitagawa}@cs.tsukuba.ac.jp

Abstract

Extensible Markup Language (XML) has become an important format for data exchange and representation on the web. In addition to conventional query processing, more complex analysis on XML data is considered to become important in order to discover valuable information. In this research, we attempt to investigate an XML-OLAP, by which we can perform multidimensional analysis on XML data taking XML’s features into account. Users are allowed to specify XML data-cube by XPath, and perform analytical processing by XQuery with OLAP extension. The system is implemented on top of relational databases, and the given requests for data-cube specification and analysis are translated into SQL so that they can be processed using the underlying system. We show the feasibility of the proposed scheme by experimental evaluations.

1. Introduction

Since its emergence in 1998, Extensible Markup Language (XML) [7] has become a de facto standard for data exchange and representation on the web. Now, XML has been used in a wide spectrum of application domains, such as web documents, business documents, and log data. For this reason, in addition to conventional simple query retrieval, more complex ways to make analysis of XML data are considered to be more and more important in order to extract useful information from massive XML data.

In our previous research [2], we proposed a model for OLAP analysis on XML data using relational databases. Specifically, for allowing users to specify facts and dimensions about XML data, we employ slightly extended XPath expressions. The system extracts corresponding XML fragments from the underlying XML database based on the fact and dimension specifications, and constructs multidimensional XML data-cube. The users then make analysis on the data-cube by issuing multidimensional queries. One notable feature of the work is that we take account of structure-based concept hierarchy, as well as value-based concept hierarchy, which is an important characteristic of XML data.

In this paper, we will discuss an approach to XML-OLAP system using relational database systems based on our previous work. Our contributions in this paper are as follows:

- We discuss roll-up operation for XML data-cube. It is an extension in SQL2003 for supporting OLAP operations for relational data-cube. We employ the syntax, and adapt it for XML data-cube. We then discuss its implementation using the functionality of relational database systems.

- We evaluate the performance of the proposed scheme by a series of experiments. The experimental results show that the proposed scheme can deal with 100MB XML data with reasonable processing time.

The rest of this paper is organized as follows: in Section 2, we introduce preliminaries which we describe about OLAP and XML. Then, in Section 3, we discuss related works. In Section 4, we show an overview of our proposed system and the definitions of fact and dimension of XML data, XML hierarchies, and data-cube on XML data. We also discussed OLAP extensions to XQuery in this section. In Section 5 we describe our implementation issue which we will discuss relational XML storage, data cube construction, and query processing with both structure- and value-based grouping, and “ROLLUP” operation. In Section 6, we give experimental evaluation. Finally, in Section 7, we conclude this paper.

2. Preliminaries

In this section, we briefly overview OLAP, XML and its query languages, XPath and XQuery.
2.1. Online Analytical Processing (OLAP)

Online Analytical Processing (OLAP) is a category of software technology that enables analysts, managers, and executives to obtain insight into data through fast, consistent, interactive access to a wide variety of possible views of information. The information has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by users.

When considering OLAP, star schema, cube, and aggregation operations are the most important concepts. To represent the multidimensional data model, star schema, that consists of single fact table and some dimension tables, is used. Each dimension table contains columns corresponding to attributes of the dimension.

An OLAP system models the input data as a logical multidimensional cube with multiple dimensions which provides the context for analyzing measures of interest. To analyze the data with the cube structure, various aggregation operations, namely, drilling, pivoting (or rotating), and slicing-and-dicing, are used to change the number of dimensions and the resolutions of dimensions of interest.

2.2. XML, XPath, and XQuery

XML has become the language of choice for data representation across a wide range of application. It has been designed to represent both structured and semi-structured data. An XML data is basically modeled as a labeled tree: elements and attributes are mapped into nodes; directed nesting relationships are mapped into edges in the tree.

XML data can be queried by XML query languages such as XPath and XQuery. XPath [5] is a language for addressing portion of an XML data. We can specify an XML subtree in term of a navigational path over XML tree by conditions on the element’s label, value, and relationship among nodes along the path.

XQuery [6] is a query language designed to query collection of XML data. XQuery uses XPath as a sub-language to address specific parts of an XML document. It employs SQL-like FLWOR (FOR, LET, WHERE, ORDER BY, RETURN) expression for performing joins.

3. Related Works

Bordawakar et al. [1] investigated various issues related to XML data analysis, and proposed a logical model for XML analysis based on the abstract tree-structured XML representation. In particular, they proposed a categorization of XML data analysis system: 1) XML is used simply for external representation for OLAP results, 2) Relational data is extracted from XML data, and then processed with existing OLAP systems, 3) XML is used for both data representation and analysis. In order to support complex analytical operations, they also proposed new syntactical extensions to XQuery, such as “GROUP BY”, “ROLLUP”, “TOPOLOGICAL ROLLUP”, “CUBE”, and “TOPOLOGICAL CUBE”. In our research, we employ the syntax of “GROUP BY ROLLUP” and “GROUP BY TOPOLOGICAL ROLLUP” to allow users to specify OLAP operation in XQuery.

Jensen et al. [4] proposed a scheme for specifying OLAP cubes on XML data. They integrated XML and relational data at the conceptual level based on UML, which is easy to understand by system designers and users. In their scheme, a UML model is built from XML data and relational data, and the corresponding UML snowflake diagram is then created from the UML model. In particular, they considered how to handle dimensions with hierarchies and ensuring correct aggregation.

4. An Overview of the Proposed XML-OLAP System

4.1. System Overview

The left side of Figure 1 shows an overview of our proposed scheme. According to the content of XML data, a user at first gives a fact path and some dimension paths in XPath expression to denote his/her interest. Referring to the given fact and dimension paths, the system produces an XML cube. After getting the cube, the user can make analysis of the XML data-cube using XQuery with OLAP extensions.

The following discusses how XML cube can be constructed in our system.
4.2. Formal Definitions

To construct an XML data-cube, we first need to specify fact and dimensions. Let us look at the definitions of fact and dimensions.

Facts about an XML Data A fact-table in a traditional OLAP system stores data items being analyzed. We attempt to define the facts in an XML data after the traditional OLAP way. In order to identify the facts, we use XPath as the query language. For example, when a user wants to get information of book sales from sales XML data as in the upper left side of Figure 2, the related data items can be obtained by the fact path 

\[ p_f = \text{doc("sales.xml")//b}. \]

**Definition 1 (Fact path)** A fact path \( (p_f) \) is an absolute XPath expression that identifies data items of interest.

**Dimensions** Having fixed the fact data, we might additionally need some dimensions whose values are used to group the facts together for the subsequent aggregation operations. In traditional OLAP systems, dimensions are given as independent tables associated with the fact table. In this work we try to define a dimension as an XPath query, but we need to care about the relationship between the fact data and dimensions. In order to ensure this, a dimension path is in either of the two cases: relative path from the fact path and absolute path with referential constraints.

**Definition 2 (Dimension path)** A dimension path \( (p_d) \) in either of the two forms:

1. \( p_d \) is a relative path expression originated from the fact path \( p_f \), or
2. \( p_d \) is an absolute path expression contains at least one condition with the fact path \( p_f \).

Figure 2 shows an example of fact and dimension paths. The circles on the top left document represent the facts corresponding to \( p_f \). When we want to use the book title as a dimension for the subsequent analysis, a dimension path can be given as \( p_{d1} = t \), which is a relative path from \( p_f \). If we are interested in grouping the books according to price ranges represented in another XML data (the upper right document of Figure 2), we need to specify absolute path expression with referential constraints like

\[ p_{d2} = \text{doc("bookinfo.xml")//b[t = p_f/t]/p}. \]

As can be seen from the example, for a given book, we can obtain corresponding price in another XML data by using title as the clue.

Concept Hierarchy The concept hierarchy is a notable feature of traditional OLAP systems by which we can carry out flexible grouping operations over the data items stored in the fact table. As with the traditional OLAP systems, we assume that value-based concept hierarchies are given beforehand. We do not go into the detail of how to represent such a hierarchy, due to the page limitation. When dealing with XML data in the same context, we need a special consideration on the semistructured nature. Specifically, we have to take into account structure-based concept hierarchy which is naturally represented as the hierarchical structure of XML data.

Taking Figure 2 for example, all books (b) are categorized by the XML hierarchies according to the area or book category. The structure-based concept hierarchy allows us to aggregate facts using such XML data structure. We will discuss the detail later.

Data Cube on XML Data We are now ready to define data cube on XML data using the concepts of the fact and dimension paths. Before going into the definition, we introduce some notations as helpers. For a given XPath expression \( p \), \([p]\) denotes an evaluation of \( p \), and the result would be XML nodes, string-values, or a boolean. Let \([p]\) denotes an evaluation of \( p \) where \( p \) represents an XPath expression.

**Definition 3 (XML data-cube)** An XML data-cube is defined as \( (p_f, D) \) where \( p_f \) is a fact path and \( D = \{ p_{d1}, p_{d2}, \ldots, p_{dn} \} \) is a set of dimension paths. A fact \( f \) in the cube is an \( n + 1 \)-tuple \( (f, d_1, \ldots, d_n) \) where \( f \in [p_f] \) and each \( d_i \) is obtained by evaluating \( p_{d_i} \): \([p_{d_i}]f \) if \( p_{d_i} \) is in a relative form or \([p'_{d_i}] \) where \( p'_{d_i} \) can be obtained by replacing each occurrence of \( p_f/p_r \) in \( p_{d_i} \) with \( [p_r]f \), \( n \) is the rank of the XML-cube.

Let us consider an XML data-cube as an example (Figure 2). It is defined as \( (p_f, p_d) \), where

\[ p_f = \text{doc("sales.xml")//b} \quad \text{and} \quad p_d = \text{doc("bookinfo.xml")//b[t = p_f/t]/p}. \]

A tuple can be extracted as follows. Firstly, fact data can be extracted by evaluating fact path like \([p_f] = \{ b_1, b_2, \ldots, b_6 \} \). For each fact data \( b_i \), we can identify corresponding dimension data in another XML data as specified by \( p_d \). When evaluating \( p_d \), we need to rewrite the path according to the fact data. For example, for the fact \( b_i, p_f/t, \) which is a part of \( p_d \), is rewritten as \([p_f/t]_i = \{ "A" \} \), then turns out to be \( \text{doc("bookinfo.xml")//b[t = "A"]/p}. \)

In this way, we can extract all tuples from the data cube, that are set of 2-tuple: \( \{(b_1, p_1), (b_2, p_4), (b_3, p_3), (b_4, p_3), (b_5, p_2), (b_6, p_5)\} \).

In contrast to the existing OLAP, and XML-cube may contain much information more than the dimensionality (what we call “rank”). That is, each XML fragment may...
contain more information than a numerical value, such as elements, texts, attributes, and hierarchical information. In order to form a cube-like structure, we need to specify some of them as dimensions of the cube structure.

For instance, each tuple of the rank 1 XML data-cube in Figure 2 (lower side) contains two XML fragments of books coming from “sales.xml” and prices from “bookinfo.xml”. According to the fragments, this XML data-cube potentially has five attribute values: title, quantity, area, price, and category. Assume that we are interested in getting the information related to the book sales area and price, we can create a cube by specifying the area and price as the dimension.

4.3. OLAP Extensions to XQuery

Once the data cube is constructed, we perform multidimensional analysis using the dimensions and related information such as XML hierarchies. In our system, we attempt to use XQuery as the user query language. However, the current version of XQuery does not support aggregation function. So we employ the syntax of OLAP extension for XQuery [1], “GROUP BY ROLLUP” and “GROUP BY TOPOLOGICAL ROLLUP”. The same as the roll-up operation in ordinary OLAP systems, “ROLLUP” enables a “SELECT” statement to calculate multiple levels of sub-totals across a specified group of dimensions. It also calculates a grand total. “ROLLUP” is an extension to the “GROUP BY” clause so its syntax is extremely easy to use. The latter, “TOPOLOGICAL ROLLUP”, is similar to “ROLLUP” but for computing structure-based grouping over XML data.

5. Implementation Using Relational Database Systems

This section discusses an implementation of the proposed model and grouping operations (Figure 1, right). We try to make the best use of relational databases as the underlying data storage. The reasons are: 1) there are many commercial and open source products, 2) enormous amount of information resources are stored in relational systems, and 3) we can leverage established relational XML storage techniques. In addition, we can utilize grouping functionalities which are supported in most relational database systems, to implement value- and structure-based grouping of XML data.

5.1. Relational XML Storage

We employ the path-approach [8] for mapping XML data to relational tables, because we can manage any well-formed XML documents with fixed relational schema and realize practical subset of XPath solely by the use of SQL functionalities. Due to the limitation of pages, we just show a brief overview. In the path-approach, an XML node is basically mapped to a relational tuple of two tables, path table containing all absolute path expression of all XML nodes, and node table containing all XML node information. Table 1 (left) shows the path table extracted from “sales.xml” and “bookinfo.xml”. In the node table (Table 1, right), there are document id (did), pid (path id), nid (node id), nnum (node number), tname (tag name), and value.

5.2. Extracting Fact and Dimensions

The first step is to extract fact and dimensions. As discussed in Section 4.2, a fact and its dimensions are XML sub-trees specified by XPath queries. Hence, we can represent the fact (or a dimension) as a part of node table. This can be achieved by evaluating the fact (dimension) path, and storing the result as a new table. Those tables can be defined as either views or materialized views.
5.3. Data Cube Construction

In the next we create an XML data-cube. For this purpose, we need to establish the relationships between the fact and the dimension as described in Section 4.2. We join the base relations by giving the referential constraints as the join key. XML data-cube table containing all attributes from the fact and dimension, and each record consists of data from the fact and dimension which have the same book title.

5.4. Query Processing

As discussed in Section 4.3, we use XQuery with OLAP extensions as the user query language. In order to process a query, we need to translate the query into SQL, because we make use of relational database systems as the query processing engine. In fact, there have been several works on XQuery to SQL query translation [3], and we can borrow those ideas. So, in this paper, we focus on how to implement OLAP operations using SQL. Specifically, we discuss how to realize structure-based grouping and roll-up operations.

Structure-based Grouping Our basic strategy is to utilize path expressions as the clue to perform grouping. Specifically, for a given data item, we need to compute the prefix of each data, and then perform grouping on the prefixes. The level of grouping can be controlled by the length of the path prefixes.

Now, we discuss how to perform grouping operations using path expressions. Let us introduce some notations. Given an XML node \( n \), let \( pexp(n) \) denote \( n \)'s absolute path expression, and let \( prefix(exp, i) \) denote path expression \( exp \)'s \( i \)-th prefix, e.g., \( prefix("/a/b/c", 1) = "/a" \), and \( prefix("/a/b/c", 2) = "/a/b." \) Then, the grouping can be performed in the following way:

1. Let the depth, which is the distance from the root, of the dimension be \( d \), e.g., the depth of a path "sales/area/kanto/tsukuba/b" is \( d = 5 \).

2. Find the common prefix of all path expressions and let the depth be \( i \), e.g., the three path expressions, "sales/area/kanto/tsukuba/b", "sales/area/kansai/osaka/b", and "sales/area/kansai/kyoto/b" have the same prefix path "sales/area". As a consequence, we get \( i = 2 \).

3. The level-\( j \) (\( i \leq j \leq d \)) grouping can be computed by calculating \( prefix(pexp(n), j) \) for each dimension value \( n \), e.g., Referring to the previous three path expressions, let \( j \) be 3. The \( prefix(pexp(n), 3) \) computes two level-3 groupings of the paths which have the same 3-depth prefixes, "sales/area/kanto" and "sales/area/kansai".

In fact, the proposed grouping operation can be implemented in many ways, but an important remark is that it can be realized solely by the functionality of SQL. One possible way is to leverage the string match functionality provided by the database system. More precisely, we can make use of regular expressions to extract substrings, and use them with the "GROUP BY" clause. Assume that we would like to use the first two tags to group the facts, e.g., use "sales/area" out of "sales/area/kanto/tsukuba/b", we can achieve this by:

\[
\text{SELECT ... FROM ...}
\]
\[
\text{WHERE ... GROUP BY regexp_replace(dim.pexp, \"/(\[/]+\[/]+)/.+\", \"\1\")}
\]

Another possibility is to introduce dedicated indexes based on Dewey encodings or prime numbers. They might be good for speeding up the grouping operations compared to the above approach. The comparison might be an interesting topic to research.

5.5. ROLLUP Operations

One possible way to implement roll-up operations ("GROUP BY ROLLUP" and "TOPOLOGICAL ROLLUP") in the extended XQuery is to directly translate them into the counterpart in SQL2003, in which OLAP operations are supported. However, SQL2003 is not supported in many database systems. For this reason, we try to realize the roll-up operations using the functionality of SQL-92, which is supported in most systems. Here we show how roll-up operations are applied to XML data-cube. As mentioned in Section 4.3, "ROLLUP" and "TOPOLOGICAL ROLLUP" create subtotals that roll up from the most detailed level to a grand total, we use "UNION ALL", which enable us to compute set union over different grouping levels, to implement the operations.

6. Experimental Evaluation

6.1. Experimental Setup

All experiments were performed in Sun Microsystems Sun Fire X4200 server whose CPU is a 2-way Dual Core AMD Opteron(tm) processor (2.4GHz). This machine has 16GB memory and runs Sun OS 5.10. We used Java version 1.5.0_09 to parse XML data to relational tables, and PostgreSQL 8.1.4 to perform query processing.

For the experimental data, we used XMark data which is a comprehensive distributed system benchmarking and optimization suite. We tested the following sizes of XML data: 10MB, 100MB, 200MB, 300MB, 400MB, and 500MB.
6.2. Benchmark Queries

For the benchmark query, we give a fact path, \( p_f = \text{doc("xmark.xml")//item} \), and two dimension paths, \( p_{d1} = \text{quantity} \) and \( p_{d2} = \text{payment} \).

We ran three queries to show the performance of roll-up functions which we can calculate the total quantity of item grouped by value-based (payment), structure-based (region), and the combination (regionpay).

6.3. Experimental Results

Figure 3 shows the elapse times for data-cube construction and query processing. At first, “item”, “quantity”, and “payment” are created. After extracting those tables, the data-cube “payqty” is constructed. For the query processing, we had done three roll-up operations, value-based (payment), structure-based (region), and their combination (regionpay). The results show that the processing time for data-cube construction is quite time consuming even for 100MB data. However, the important remark here is that once the data-cube is constructed, analytical query processing can be processed in reasonable time. In real systems, in many cases, data-cube construction is performed once in the midnight, and analytical processing are applied repeatedly in business hours. From the observation, we think that the performance of the proposed scheme is acceptable.

7. Conclusions

In this paper, we proposed a system for XML-OLAP which is constructed on top of relational databases. Our system supports both value- and structure-based concept hierarchy, and XML data-cube. We then discussed OLAP extension to XQuery. For the implementation issues, we use the path approach for mapping XML data to relations, and we utilize “UNION ALL” to perform “GROUP BY ROLLUP” operation for both structure- and value-based groupings. Our experiments with large collections of XML data show that the “GROUP BY ROLLUP” queries perform less than 10 sec. for 500MB XML data. The results show the effectiveness of our proposed technique.

For the future research, we try to improve the performance of data-cube construction. We also plan to investigate how to incorporate textual features such as word vectors of XML data into the analytical processing.

8. Acknowledgments

This research is partly supported by the Grant-in-Aid for Scientific Research (17700110) from Japan Society for the Promotion of Science (JSPS), Japan, and the Grant-in-Aid for Scientific Research on Priority Areas (18049005) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.

References