# XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses

Byung-Kwon Park[1], Hyoil Han[2], and Il-Yeol Song[2]

[1] Dong-A University, Busan, Korea
bpark@dau.ac.kr
[2] Drexel University, Philadelphia, PA 19104, USA
hyoil.han@cis.drexel.edu, songiy@drexel.edu

**Abstract.** Recently, a large number of XML documents are available on the Internet. This trend motivated many researchers to analyze them multi-dimensionally in the same way as relational data. In this paper, we propose a new framework for multidimensional analysis of XML documents, which we call *XML-OLAP*. We base XML-OLAP on XML warehouses where every fact data as well as dimension data are stored as XML documents. We build XML cubes from XML warehouses. We propose a new multidimensional expression language for XML cubes, which we call *XML-MDX*. XML-MDX statements target XML cubes and use XQuery expressions to designate the measure data. They specify text mining operators for aggregating text constituting the measure data. We evaluate XML-OLAP by applying it to a U.S. patent XML warehouse. We use XML-MDX queries, which demonstrate that XML-OLAP is effective for multi-dimensionally analyzing the U.S. patents.

## 1 Introduction

An online analytical processing (OLAP) system is a powerful data analysis tool for decision-making [11]. It provides an analysis from multiple perspectives or dimensions for a large amount of data residing in a data warehouse. Data warehouses are commonly organized with one large fact table and multiple small dimension tables. The fact and dimension tables are typically the structured data stored in a relational database.

Recently, a large number of XML documents are available on the Internet. Thus, we need to analyze them multi-dimensionally in the same way as relational data. However, the data model of XML documents is not flat like relational data, but a tree structure. In addition, XML documents can contain unstructured data such as text. Thus, we need to develop a new framework of multidimensional analysis for XML documents.

In this paper, we propose an OLAP framework for XML documents, which we call *XML-OLAP*. We base XML-OLAP on XML warehouses where every fact data as well as dimension data is stored as an XML document. XML cubes are built from XML warehouses. While conventional data cubes have numeric data as measure values, XML cubes have either numeric or text data. We propose a

A Min Tjoa and J. Trujillo (Eds.): DaWaK 2005, LNCS 3589, pp. 32–42, 2005.

new multidimensional expression language over XML cubes, which we call *XML-MDX*. XML-MDX statements target XML cubes and use XQuery expressions to specify the measure data, axis dimensions, and slicer. They also specify text mining operators for aggregating text constituting the measure data such as summarization, classification, and top keyword extraction.

We show an XML-OLAP example for the U.S. patent warehouse to evaluate its effectiveness. The U.S. patent warehouse is built by extracting information from the U.S. Patent Web Site [12], converting it into XML documents, and storing them in a native XML database. Dimension tables are also built in the form of XML documents and stored in the native XML database. We demonstrate XML-MDX queries to show that XML-OLAP is effective for multi-dimensionally analyzing the U.S. patents.

This paper makes the following contributions: (1) We propose a new framework, XML-OLAP, for multi-dimensional analysis of XML documents. We believe XML-OLAP is the first framework for online analysis of an XML document set. (2) We propose a new multidimensional expression language, XML-MDX. We are inspired by the Microsoft MDX language [11] which is widely accepted as an OLAP query language. XML-MDX can accommodate the hierarchical tree structures of XML documents. (3) We propose a mechanism to enable the aggregation of text data contained in XML documents using text mining operations. It can give the text mining community a vehicle to make their technology accessible to a broad user base.

This paper is organized as follows: Section 2 describes the related work. Section 3 describes building an XML warehouse. Section 4 describes building XML cubes and querying them using XML-MDX. Section 5 describes the XML-OLAP application to the U.S. patent XML data. Section 6 concludes the paper.

## 2 Related Work

Pokorny [9] applied a star schema to XML data. A dimension hierarchy is defined as a set of logically connected collections of XML data. Facts may also be conceived as elements of XML data. Pokorny proposed a formal model for dimension hierarchies and referential integrity constraints in an XML environment. We also assume that both dimension and fact information are represented as XML documents in the same way as Pokorny does.

Nassis et al. [7] also worked on XML document warehousing. They focused on the conceptual design of XML document warehouses and the concept of virtual dimensions using XML views. They utilized object-oriented concepts in UML to develop a conceptual model for XML document warehouses from user requirements.

Golfarelli et al. [2] dealt with the problem of automatically deriving the conceptual schema from an XML source. They assumed that the XML data have all the information for the schema. They proposed a semi-automatic approach for building a schema from an XML DTD or schema.

Hümmer et al. [3] proposed a family of XML document templates, called XCube, to describe a multidimensional structure, dimensions and fact data for integrating several data warehouses into a virtual or federated data warehouse. The XML templates are not directly related to XML warehousing, but they can be used for representing hierarchical dimension data in our framework.

There are a lot of work on constructing OLAP cubes from distributed XML data. Jensen et al. [4,5] transformed XML data on the web into relational data in order to be used by conventional OLAP tools. Niemi et al. [8] proposed a system which can construct an OLAP cube based on an user's MDX query. They all construct a relational OLAP cube by transforming XML data collected from distributed XML sources, whereas we construct an XML cube from XML documents.

## 3 XML Warehouses

In Section 3.1, we present the multidimensional model of an XML warehouse and how to derive it. In Section 3.2, we present how to build an XML warehouse from the given XML document set.

### 3.1 Multidimensional Modeling of XML Warehouses

We assume that an XML warehouse has a multidimensional model as in Figure 1. The model has a single repository of XML documents, which forms fact data, and multiple repositories of XML documents, in which each forms one dimension data. In Figure 1, there are $n$ dimensions and thus, $n$ repositories of XML documents.

The fact repository is the same as assumed by Nassis et al. [7]. Each fact is described in a single XML document. Thus, the fact data is not as simple as in a conventional data warehouse. It has a hierarchical tree structure containing structured data and unstructured data.

Dimension data are described in XML documents, and each dimension data is grouped into a repository of XML documents. Since each dimension has a hierarchy, a single XML document in a repository contains an instance of a dimension hierarchy rooted at a top level member in the hierarchy. Some auxiliary data structures like indexes are used to link dimension data with fact data.

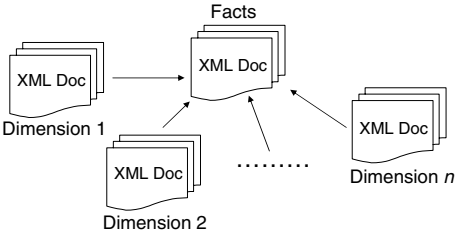


**Fig. 1.** Multidimensional Model of An XML Warehouse

The XML multidimensional model described in this section has the following advantages: (1) Since all the fact and dimension data are described in XML documents, we can easily collect them. (2) We can store all the fact and dimension data in a native XML database and can easily manage and query them through the native XML database management system. (3) Since each dimension data is described in an XML document, we can easily represent the dimension hierarchy in a single document using the tree structure of an XML document. Thus, we need not join multiple tables as required in the relational snowflake model.

### 3.2 Building an XML Warehouse

Building an XML warehouse consists of two steps: building a single XML repository for fact data and building a number of XML repositories for dimension data. Rusu et al. [10] dealt with the problem of processing raw XML documents into a data warehouse repository, and proposed some rules and techniques for data cleaning, integration, summarization, and updating/linking existing documents. We assume that the XML repository for fact data is provided after cleaning. We focus on building the XML repositories for dimension data. In order to decide the required dimensions to analyze the given XML document repository, we need to build the conceptual model of the XML documents.

There are several works on the conceptual modeling of XML data using UML. Jensen et al. [5] proposed an algorithm for automatically constructing UML diagrams from XML data based on their DTD's. Lujan-Mora et al. [6] extended UML for multidimensional modeling including multistar model, shared hierarchy levels, and heterogeneous dimensions. We adopt their methods for conceptual modeling of XML data in UML class diagrams.

From the conceptual model of fact data, we can decide dimensions for analyzing them. We assume that it is done manually because the conceptual model is object-oriented and expressed in UML class diagrams for the purpose of helping people to understand the logical structure of fact data. Nassis et al. [7] proposed to select dimensions based on user requirements and to represent the dimensions virtually using XML views since they assumed that all the dimension data are part of fact data. However, we assume that some dimension data are out of fact data. Thus, for simplicity, we materialize an XML repository for each dimension selected.

For multidimensional analysis, we need some mechanism to join dimension and fact data. In a conventional data warehouse, we insert a foreign key in the fact data to match with each dimension data. In this paper, we rely on an index structure that matches each dimension data with the corresponding fact data. We build the index together with the XML repositories for dimension data.

## 4 The Multidimensional Analysis Framework for XML Warehouses

In this section, we describe XML-OLAP, which is about generating XML cubes and expressing multidimensional queries. In Section 4.1, we present a new notion

of XML cube (called *XQ-Cube*). In Section 4.2, we present a new multidimensional expression language (called *XML-MDX*) for XQ-Cubes.

### 4.1 XML Cubes

Since our XML warehouse has XML documents as fact data, the cube constructed from the XML warehouse should have the cells whose values are an aggregation of XML documents. Defining an aggregation over multiple XML documents is difficult because an XML document is a hierarchically structured composite data object. However, defining an aggregation over such data segments as numeric or text data segments is possible.

We propose to use an XQuery expression for measure specification. We call the cube constructed from the XML warehouse with measure values described by an XQuery expression *XQ-Cube*. The measure values, the evaluation result of the XQuery expression, are numeric or text data. If they are numeric, the aggregation will be the same as in relational cubes such as addition and average; otherwise, the aggregation will be a kind of text operation. We introduce text mining operations for text aggregation (see Section 4.2).

An XQ-Cube has the following advantages: (1) A variety of cubes can be generated. Since the measure data is specified using an XQuery expression, any kind of cube can be defined that XQuery can generate. (2) It provides an aggregation mechanism over XML documents. Since the measure data is a fragment of an XML document, we can apply various aggregation operators according to the data type. (3) An XQ-Cube is a generalization of a relational cube. It becomes a relational cube when the XQuery expression is evaluated to numeric data, while it becomes a text cube when evaluated to text data.

### 4.2 Multidimensional Expression Language

For querying a cube, we need a query language for cubes. Microsoft designed a multidimensional expression language called *MDX* [11] for relational cubes. We are inspired by Microsoft MDX to design a new multidimensional expression language called *XML-MDX* for XQ-Cubes. XML-MDX has two statements: CREATE XQ-CUBE and SELECT. The former is for creating a new XQ-Cube, and the latter for querying. In general, in order to enhance the query processing performance of XML-MDX or to use an XQ-Cube multiple times, we first create an XQ-Cubes and then refer to it in queries.

*CREATE XQ-CUBE:* Figure 2 shows the basic syntax of the CREATE XQ-CUBE statement. The <XQ-Cube_name> value specifies the name of the XQ-Cube to create. A CREATE XQ-CUBE statement is composed of two clauses: FROM and WHERE. The created XQ-Cube is stored for use by XML-MDX queries.

The FROM clause specifies the measure data from which the XQ-Cube will be created. Figure 3 shows the definition of the FROM clause expressed in BNF notation. The <XQ-Cube_specification> value specifies the measure data using

```
CREATE XQ-CUBE  <XQ-cube name>
FROM  <XQ-cube specification>
[ WHERE  <slicer specification> ]
```

**Fig. 2.** Definition of CREATE XQ-CUBE statement in BNF

```
<FROM_clause> ::= FROM <XQ-cube_specification>
<XQ-cube_specification> ::= <XQuery_expression> : <aggregation_operator> ]
<aggregation_operator> ::= ADD | LIST | COUNT | SUMMARY | TOPIC |
                           TOP KEYWORDS | CLUSTER
```

**Fig. 3.** Definition of FROM Clause in BNF

```
<WHERE_clause> ::= WHERE <slicer_specification>
<slicer_specification> ::= "(" <XQuery_expression> { "," <XQuery_expression> } ")"
```

**Fig. 4.** Definition of WHERE Clause in BNF

```
SELECT  <axis 0 specification>,
        <axis 1 specification>,
          ...
FROM  <XQ-Cube name>
[ WHERE  <slicer specification> ]
```

**Fig. 5.** Basic Syntax of XML-MDX

an XQuery expression. We should specify an aggregation operator according to the measure data, the evaluation result of the XQuery expression.

In this paper, we define the following seven aggregation operators: ADD, LIST, COUNT, SUMMARY, TOPIC, TOP KEYWORDS, and CLUSTER. ADD is for numeric data as it is in relational OLAP. The other operators are all for non-additive data including text. LIST displays the data consecutively in a sequence. COUNT displays the number of measure data. The others are all from text mining techniques: SUMMARY, TOPIC, and TOP KEYWORDS display a total summary, a topic, and top keywords respectively from all the text data to aggregate. CLUSTER builds a cluster over all the text data to aggregate. We can expand the aggregation operators as the text mining techniques progress.

The WHERE clause is optional. It determines which dimension members to use for the slicer which restricts the extractions of data to the determined dimension members. Figure 4 shows the definition of the WHERE clause expressed in BNF notation. The <slicer_specification> value specifies a slicer which is a tuple of XQuery expressions. Each XQuery expression specifies a dimension member which filters off the other members. A special XQuery expression is used for specifying the 'All' member (see Figure 8). The dimensions that are not specified in the <slicer_specification> form the axis dimensions of the XQ-Cube created.

```
<SELECT_clause> ::= SELECT <axis_specification> { "," <axis_specification> }
<axis_specification> ::= <XQuery_expression_set> ON <axis_name>
<XQuery_expression_set> ::= "{" <XQuery_expression> { "," <XQuery_expression> } "}"
<axis_name> ::= COLUMNS | ROWS | PAGES | SECTIONS | CHAPTERS |
                AXIS(<index>)
```

**Fig. 6.** Definition of SELECT Clause in BNF

*SELECT:* Figure 5 shows the basic syntax of the SELECT statement. A SELECT statement has the same structure as that of Microsoft MDX, which is composed of three clauses: SELECT, FROM, and WHERE. The FROM clause designates an XQ-Cube name previously created by a CREATE XQ-CUBE statement. We populate the result set of the SELECT statement from the designated XQ-Cube.

The SELECT clause specifies axis dimensions. Each axis dimension determines an edge of a multidimensional result set. Figure 6 shows the definition of the SELECT clause expressed in BNF notation. Each <axis_specification> value defines one axis dimension. The number of dimensions in an XML warehouse is the maximum number of axis dimensions. An <axis_specification> value is broken down into a set of XQuery expressions and an axis name.

The result set of the XQuery expressions constitute the members of an axis dimension. Since each dimension having a hierarchical structure is represented in a single XML document, an XQuery expression specifies a member of a dimension level. We need an XQuery expression for each member of an axis dimension.

We assign axis names in the same way as Microsoft MDX does [11]. Each axis dimension is associated with a number: 0 for the X-axis, 1 for the Y-axis, 2 for the Z-axis, and so on. The <index> value is the axis number. For the first 5 axes, the aliases COLUMNS, ROWS, PAGES, SECTIONS, and CHAPTERS can be used in place of AXIS(0), AXIS(1), AXIS(2), AXIS(3), and AXIS(4), respectively. An XML-MDX query cannot skip axes. That is, a query that includes one or more axes must not exclude lower-numbered or intermediate axes.

The definition of the WHERE clause of the SELECT statement is the same as that of CREATE XQ-CUBE statement. The <slicer_specification> value filters the XQ-Cube specified in the FROM clause. Note that, as in Microsoft MDX, the dimensions that are not explicitly assigned to axes in the SELECT clause are assumed to be slicer dimensions. They filter the XQ-Cube with their default members. A default member is the All member if an 'All' level exists, or an arbitrary member of the highest level.

XML-MDX has the following advantages over Microsoft MDX: (1) It can have all the features of Microsoft MDX since XML-MDX is designed based on Microsoft MDX. (2) Composing and processing XML-MDX queries are easy since all the specifications for the measure values and the dimension members are expressed in XQuery. We use an existing XQuery engine to process XML-MDX queries since no special syntax is required for XML-MDX. (3) When specifying slicer or axes, selecting the dimension members satisfying a condition is possible since we are using XQuery to specify them. Microsoft MDX has no such facility and use only a path in the dimension hierarchy.

## 5 Application to US Patent XML Warehouse

We assume that we are given a huge collection of XML documents about U.S. patents. They form the XML repository representing fact data of the U.S. patent XML warehouse. Figure 7 shows an example of such documents. After reviewing the fact repository, we build a conceptual model using a UML class diagram. From the conceptual model, we decide the dimensions to use for multidimensional analysis.

Figure 8 shows the hierarchies of the four dimensions selected for the U.S. patent XML warehouse. Each dimension has the 'All' level. The two dimensions, 'Appl.Time' and 'Reg.Time', represent when a patent was applied and registered respectively. The dimension, 'Inventor', represents a patent's inventors. The dimension, 'Topic', represents a patent's classification.

Figure 9 shows an XML document in the XML repository representing the dimension, 'Appl.Time'. The document is about an application year, 1998. The

```
<uspatent>
  <title>
    <text> Rule based database security system and method </text>
  </title>
  <abstract>
    <text> A rule-based database security system and method are disclosed. </text>
  </abstract>
  <inventor>
    <name> Cook; William R. </name>
    <addr> Redwood City, CA </addr>
  </inventor>
  <patent>
    <no> 6,820,082 </no>
    <applNo> 541227 </applNo>
  </patent>
  <registeredOn> <date> November 16, 2004 </date> </RegisteredOn>
  <filedOn> <date> April 3, 2000 </date> </FiledOn>
  <claim>
    <number> 1 </number>
    <text> A method for processing requests from a user to perform an act ...</text>
  </claim>
</uspatent>
```

**Fig. 7.** Fact Data about U.S. Patents

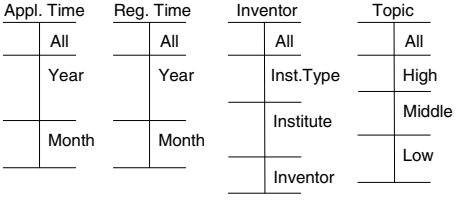


**Fig. 8.** Dimension Hierarchies of U.S. Patent XML Warehouse

```
<year num = "1998">
   <month num = "3" name = "Mar." />
   <month num = "9" name = "Sep." />
</year>
```

**Fig. 9.** An XML Document for Dimension *ApplTime*

```
<instType name = "university" code = "001">
    <institute name = "Drexel" addr = "Philadelphia, PA">
        <inventor name = Il-Yeol Song" addr = "Philadelphia, PA" />
    </institute>
</instType>
```

**Fig. 10.** An XML Document for Dimension *Inventor*

```
CREATE XQ-CUBE  XQ-Cube-1
FROM   col('/db/uspatent')//patent/no : COUNT
WHERE  ( col('/db/applTime')/ALL,
              col('/db/regTime')//year[@num>2000] )
```

**Fig. 11.** An Example of CREATE XQ-Cube Statement

level, 'year', has an attribute, 'num', It has the lower level, 'month', having two attributes: 'num' and 'name'. The level, 'month', has two members whose values of the attribute, 'num' are 3 and 9 respectively.

Figure 10 shows an XML document in the XML repository representing the dimension, 'Inventor'. The document is about an institution type, 'university', which is a member of the level, 'instType' of the dimension. The level, 'instType', has two attributes: 'name' and 'code'. It has the lower level, 'institute', having two attributes: 'name' and 'addr'. The level, 'institute', has the lower level, 'inventor', having two attributes: 'name' and 'addr'.

Figure 11 shows an example to create an XQ-Cube named XQ-Cube-1. The XQuery expression, "col('/db/uspatent')//patent/no", specifies the measure of XQ-Cube-1. The collection, "/db/uspatent" contains the fact data from which we collect "//patent/no" for the measure. The aggregation operator, COUNT, counts the number of patent no's. The WHERE clause has two XQuery expressions. The collection, "/db/applTime" contains the XML documents for 'Appl.Time'. The special XQuery expression, "/All" means that the 'All' level is selected for slicing, which results in the aggregation along all the members of the dimension. The collection, "/db/regTime" contains the XML documents for 'Reg.Time'. The XQuery expression, "//year[@num>2000]" results in slicing off all 'year' less than or equal to 2000. As a result, XQ-Cube-1 has three axis dimensions: 'Inventor', 'Topic', and 'Reg.Time' with 'year' greater than 2000.

Figure 12 shows an XML-MDX query for XQ-Cube-1. The slicer specification in the WHERE clause slices off the registration years less than or equal to 2002.

```
SELECT  { col('/db/topic')//high[@topic='XML'],
                col('/db/topic')//high[@topic='OLAP'] } ON COLUMNS
          { col('/db/inventor')//instType[@name='university'],
            col('/db/inventor')//instType[@name='industry'] } ON ROWS
FROM    XQ-Cube-1
WHERE  ( col('/db/regTime')//year[@num > 2002] )
```

**Fig. 12.** An Example of XML-MDX Query

Then, a new XQ-Cube is returned as a result, which has the axis dimensions specified in the SELECT clause. The axis COLUMNS has two members of the dimension 'Topic': 'XML' and 'OLAP'. The axis ROWS has two members of the dimension 'Inventor': 'university' and 'industry'.

## 6 Conclusions

In this paper, we proposed XML-OLAP as a new framework for multidimensional analysis of XML warehouses. We assumed that both fact and dimension data are all represented as XML documents in XML warehouses. We proposed to construct a new type of cube named XQ-Cube from XML warehouses. An XQ-Cube is constructed from the measure data specified by an XQuery expression. We used text mining operations for the aggregation of text measure data. We proposed XML-MDX as a new multidimensional expression language for XQ-Cubes. We demonstrated its effectiveness through the example of U.S. Patent XML Warehouse. We believe our framework will contribute to the effective analysis of the vast amount of XML documents on the Web.

## Acknowledgement

This work was supported by the Post-doctoral Fellowship Program of Korea Science & Engineering Foundation (KOSEF).

## References

1. A. Abello, J. Samos and F. Saltor "Understanding Facts in a Multidimensional Object-Oriented Model," In *Proc. The 4th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP01)*, pp. 32–39, Atlanta, 2001.
2. M. Gofarelli, S. Rizzi, and B. Vrdoljak, "Data Warehouse Design from XML Sources," In *Proc. The 4th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP01)*, pp. 40–47, Atlanta, 2001.
3. W. Hümmer, A. Bauer, and G. Harde, "XCube – XML For Data Warehouses," In *Proc. The 6th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP03)*, pp. 33–40, New Orleans, Louisiana, 2003.
4. M. R. Jensen, T. H. M$\phi$ller and T. B. Pedersen, "Specifying OLAP Cubes on XML Data," *Journal of Intelligent Information Systems*, Vol. 17, No. 2/3, pp. 255–280, 2001.
5. M. R. Jensen, T. H. M$\phi$ller and T. B. Pedersen, "Converting XML Data To UML Diagrams For Conceptual Data Integration," In *Proc. The 1st Intl Workshop on Data Integration Over The Web*, pp. 17–31, 2001.
6. S. Lujan-Mora, J. Trujillo and P. Vassiliadis, "Advantages of UML for Multidimensional Modeling," In *Proc. the 6th Intl Conf. on Enterprise Information Systems (ICEIS 2004)*, pp. 298–305, ICEIS Press, Porto (Portugal), 2004.
7. V. Nassis, R. Rajugan, T. S. Dillon and W. Rahayu, "Conceptual Design of XML Document Warehouses," In *Proc. Data Warehousing and Knowledge Discovery, 6th International Conference, DaWaK 2004*, pp. 1–14, Zaragoza, Spain, 2004.

8. T. Niemi, M. Niinimaki, J. Nummenmaa and P. Thanisch, "Constructing an OLAP Cube from Distributed XML Data," In *Proc. The 5th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP02)*, pp. 22–27, McLean, 2002.
9. J. Pokorny, "Modelling Stars Using XML," In *Proc. The 4th ACM Intl Workshop on Dara Warehousing and OLAP (DOLAP01)*, pp. 24–31, Atlanta, 2001.
10. L. I. Rusu, W. Rahayu and D. Taniar, "On Building XML Data Warehouses," In *Proc. Intelligent Data Engineering and Automated Learning - IDEAL 2004, 5th International Conference*, pp. 293–299, Exeter, UK, 2004.
11. G. Spofford, *MDX Solutions with Microsoft SQL Server Analysis Services*, John Wiley & Sons, 2001.
12. United States Patent and Trademark Office, http://www.uspto.gov/