RC23734 (W0510-012) October 4, 2005
Computer Science

IBM Research Report

Analytical Processing of XML Documents: Opportunities and Challenges

Rajesh R. Bordawekar, Christian A. Lang
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Analytical Processing of XML Documents: Opportunities and Challenges

Rajesh R. Bordawekar Christian A. Lang
IBM T. J. Watson Research Center, Hawthorne, NY 10532
{bordaw,langc}@us.ibm.com

ABSTRACT

Online Analytical Processing (OLAP) has been a valuable tool for analyzing trends in business information. While the multi-dimensional cube model used by OLAP is ideal for analyzing structured business data, it is not suitable for representing and analyzing complex semi-structured data, such as XML documents. Need for analyzing XML documents is gaining urgency as XML has become the language of choice for data representation across a wide range of application domains. This paper describes a proposal for analyzing XML documents using the abstract XML tree model. We argue that OLAP's multi-dimensional aggregation operators can not express structurally complex analytical operations on XML documents. Hence, we outline new extensions to XQuery for supporting such complex analytical operations. Finally, we discuss various challenges in implementing XML analysis in a real system.

1. INTRODUCTION

Since its inception as a language for large-scale electronic publishing, Extensible Markup Language (XML) has emerged as the lingua franca for portable data representation. As a derivative of SGML, XML has been designed to represent both structured and semi-structured data. XML's ability to succinctly describe complex information can also be used for specifying application meta-data. XML's popularity is evident from its use in a wide spectrum of application domains: from document publication, to computational chemistry, health care and life sciences, multimedia encoding, geology, and e-commerce. Increasing popularity of web-based business processes and the emergence of web services has led to further acceptance of XML.

In spite of XML's wide-spread use, currently there are very few tools for analyzing XML data. XML data can be analyzed in two ways: (1) as semantically-rich text documents, and (2) as domain-specific data formulated using XML's semi-structured data model. Current efforts in XML analysis belong to the first category and use information retrieval techniques (e.g., keyword text searching) for knowledge discovery from XML documents [3, 18, 10]. To the best of our knowledge, there is no known work that analyzes XML data using domain-specific information. This is the focus of our current work.

An example of domain-specific analysis is Online Analytical Processing (OLAP) [8, 5], which has been extensively used by decision support systems. Such analysis is used to detect and predict trends in non-volatile time-varying business data. An OLAP system models the input data as a logical multidimensional *cube* with multiple *dimensions* which provide the context for analyzing *measures* of interest. Traditionally, measures are numeric values (e.g., units of sales or total sale amount) associated with the business data. Data analysis usually involves dimensional reduction of the input data using various aggregation functions, e.g., statistical (median, variance, etc.), physical (center of mass), and financial (volatility). Most database vendors support similar aggregation functions along with dimensional operators such as, `ROLLUP`, `GROUPBY`, and `CUBE`.

While OLAP is an effective tool for evaluating hierarchical relationships in structured data, its applicability is currently restricted to well-formulated business data that can be mapped to the multi-dimensional OLAP model. This prevents application of several useful OLAP features, e.g., grouping based on common data properties, structured aggregation, and trend analysis to XML data. There are three possible ways of using XML data in a data analysis system:

1. In the first approach, XML is used simply for external presentation of the OLAP results [20, 24]. The raw data is stored using either the relational (ROLAP) or the multi-dimensional (MOLAP) storage. Various data analysis operations (e.g., `CUBE` queries) are executed using the traditional multi-dimensional OLAP model.

2. In the second approach, input data is stored as XML documents. Relevant data is first extracted from the input XML documents using a XML processing language (e.g., XSLT, XQuery, or SQL/XML) and exported to the OLAP engine. The data analysis is still implemented using the multi-dimensional model. The results from the OLAP analysis may also be exported as XML documents.

3. The third approach uses XML both for data representation and processing. The data analysis engine represents the XML documents as trees using the tree-based (hierarchical) XML model and analyzes both the structure and the data values using an XML processing language.

In this paper, we examine analysis of an XML document using the tree-based XML model. In Section 2, we first review the XML data model and discuss why the multi-dimensional OLAP model is not suitable for analyzing XML documents. We then propose a new approach for analyzing XML documents using the XML data model and describe new operators for supporting structured aggregation over

XML data. Section 3.2 discusses various challenges in implementing the proposed XML analysis model in a real system. Section 4 discusses the related work. We present our conclusions and outline future work in Section 5.

2. ISSUES IN XML ANALYSIS

Traditional OLAP uses a regular multi-dimensional model where multiple *independent attributes* called dimensions *jointly* define the context for the corresponding numeric measures. Measures are those attributes of the data model that are used as input to the aggregation operations. Dimensions can have sub-attributes called, *members*, that exhibit *hierarchical non-recursive containment* relationships (e.g., the time dimension can have the following hierarchy[1] with members: year, quarter, month, days, and hours). Multi-dimensional OLAP is characterized by the following key features: (1) Input data organized into independent dimensions and numerical measures (e.g., using the star or snowflake schema on relational base tables), (2) Multi-dimensional array-like addressing of numeric measures and (3) Computations dominated by *structured* aggregation operations over numerical measures: (a) across levels of individual dimensions and (b) across dimensions.

Online analytical processing of XML documents raises issues that are substantially different from the traditional multi-dimensional OLAP. XML analysis differs both in the underlying data model and the prospective query patterns. We discuss these differences below:

Differences in the Data Model:

1. **Semantically rich contents:** XML is a flexible text format derived from SGML. An XML document is a text document whose textual entities are scoped in a hierarchy of self-descriptive markup tags. XML can be used to develop different domain-specific vocabularies that can encode the domain content via semantic markups and encode inherent relationships among the content entities via markup hierarchies. The XML data model views an XML document as a tree in which the internal nodes correspond to elements (denoting the markup), the leaves correspond to the textual content, and the tree edges correspond to the relationships among the content entities. Different axes in XML data can represent various relationships, e.g., *containment* (HAS-A) and *subclass* (IS-A) relationships.

   For analytical purposes, internal nodes of an XML tree (i.e., elements) can be viewed as members of scoped **dimensions**, where the dimension scope is determined by their parent elements, and values of the leaves can be viewed as the corresponding **measures**. In this model, dimension members are related to each other via XML's hierarchical structure. However, not all dimensions are mutually dependent, e.g., dimensions defined by **unique** siblings (and their subtrees) are independent within the scope of their parent dimension. Further, unlike traditional OLAP, classification between dimensions and measures is not rigid. Any XML `element` can be associated with a set of attributes that provide additional information on that element. Such information could also be used for analysis purposes. In other words, some *dimensions* could also be analyzed as *measures*. The push and pull operators [22] provide similar functionalities for relational OLAP systems.

2. **Semi-structured Data Model:** Unlike relational data, XML data (or documents) does not adhere to a rigid schema and can exhibit irregular structure. At the same time, all *well-formed* XML documents conform to an abstract XML tree whose nodes are ordered in an in-order, depth-first manner (called the document order). XML documents can have recursive hierarchies or hierarchies with different members. Thus, XML is an ideal representation of *semi-structured* data. The flexible structure of an XML document can be specified using a strongly-typed XML schema [7]. Potentially, more than one XML *instance* document can map to an XML schema. Unlike the multi-dimensional OLAP, the context of a measure is defined by the hierarchy in which it is scoped. In an XML document, a measure attribute can appear in more than one contexts (or hierarchies). Therefore, an analytical operation over a measure in one context may not be applicable for the same measure in another context. Finally, since XML nodes are ordered in the document order, *measures* themselves could be *semantically related* by the order relationship.

3. **Navigational addressing:** The abstract tree to represent the XML document is addressed using the XPath navigational language [7]. XPath navigates the abstract XML tree via five distinct axes. These axes support navigation on the tree over explicit parent-child edges and implicit edges such as sibling edges. Hence, any node of an XML tree can be addressed in a multitude of ways. This is in contrast to the rigid array-based addressing in the OLAP data model.

4. **Non-numeric measures:** Traditional OLAP involves analyzing only numeric measures (e.g., sales) of business data using aggregation functions. Since XML is also used for specifying non-business data (e.g., genome databases), it can have both numeric and non-numeric data (e.g., ATCG strings representing amino acid sequences) that need to be analyzed.

Differences in the Query Patterns:

1. **Order-dependent queries:** The XML data model enforces a strict document ordering of the XML nodes. The XML node ordering is exploited by XPath to support position-based queries on the XML tree, e.g., identify the first child of a node. Similar position-based queries could be used for analyzing ordered data sets whose ordering carries certain semantics. For example, consider an XML document that stores effects of a drug on a bio-metric parameter (e.g., white blood cell count) in a clinical drug study [13]. Figure 1 represents the corresponding abstract XML tree. Typical order-dependent analytical queries on this document can include: (1) For each asthma drug, compare the blood cell count after every usage with the corresponding count for the healthy case, (2) Determine those drugs whose second usage results in the maximum change in the white blood cell count, or (3) For

[1] A dimension can have more than one hierarchy.

all asthma drugs, find the maximum variation in the white blood cell count after the second usage.

2. **Generalized grouping:** Typical relational OLAP operations such as GROUPBY, ROLLUP or CUBE group tuples of a relation based on *values* of its column attributes. In XML analysis, one can also group XML entities based on their *path attributes* that encode entity relationships. Path attributes can be specified via XPath expressions or can use generalized tree patterns specified using regular path expressions. Section 3 presents a proposal of incorporating structural grouping features into XQuery.

3. **Queries on non-numeric measures:** Non-numeric (textual) measures could be used in two types of queries: (1) Structured queries which involve aggregation operations over strings, e.g., find the maximum or average length of the string measures, and (2) approximate queries which involve substring or string pattern matching. An example application is searching for similar images in MPEG-7 [9]. The MPEG-7 standard is based on XML and allows the storage of image and video features as strings. Similarity searching on images and videos is thereby transformed into similarity searching on strings.

4. **Approximate hierarchical queries:** Due to the semi-structured nature of XML documents, analyses based on tag names and path specifications are affected by structure changes, misspellings, or renamings. Consider, for example, the use of `surname` rather than `lastName` in an XPath expression. Similarly, the tag order in a path specification may be inverted, as in `//order/customer/address` versus `//customer/order/address`. In such cases, approximate tag and path matching based on semantic similarity [23] and sequence similarity [11] become a necessity.

5. **Hierarchical slice&dice:** In a traditional OLAP system, slicing involves reducing dimensions of a data cube and then projecting the data cube using the reduced dimension. Equivalently, an XML tree could be sliced over its independent dimensions by selectively eliminating the subtrees in those dimensions. Similarly, the dicing operation identifies and removes subtrees based on values derived from structural properties (e.g., depth of an XML node) or node values.

6. **Structural analytics:** In the traditional OLAP system, *what-next* analysis has been extensively used to predict future trends. The what-next analysis involves modifying values of certain measures and studying its impact on the overall data trends by using different aggregation functions. In XML analysis, one can evaluate the impact of relationships by modifying structures of the XML data. For example, consider an XML document describing the structure of an organization where the organization has many divisions, each division has many departments, each department has many groups, and each group consists of several employees. Each division has a fixed budget which gets percolated down the organization hierarchy according to a certain formula. Consider an analyst who wants to find out the impact of the organization hierarchy on a group's budget. She can rerun the budget computation by moving the group to another departmental hierarchy. Existing OLAP systems can not support such structural analytics.

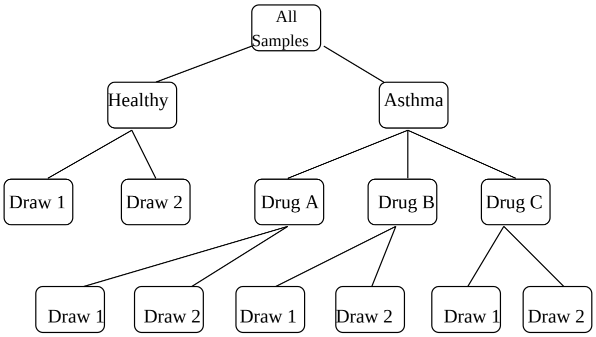

Figure 1: An Example of Order-dependent OLAP Query from a clinical-study application.

3. A MODEL FOR XML ANALYSIS

As described in Section 2, key requirements of an XML analysis system are: (1) ability to represent semi-structured data, (2) ability to support order-oriented queries, and (3) ability to support value and structural queries over numeric and non-numeric measures. The abstract tree model proposed by the XML standard [7] satisfies the first two requirements easily. Further, any XML document can be mapped to the abstract model without any additional formulations. Hence, we base our *logical* analytical model on XML's abstract tree model. Any XML document that is being analyzed is first parsed and translated into a logical tree-based representation (note that the actual physical representation of the document can vary). This logical tree can then be traversed using existing XML languages, e.g., XPath, XSLT, and XQuery. It is interesting to note that many XML analysis queries can be specified using these languages. For example, hierarchical slice & dice can be easily implemented using an XQuery application. XQuery could also be used for implementing structural analytics. Useful analytical features missing from XQuery are structured grouping and aggregation. We now propose simple yet powerful extensions to XQuery to support these features. We then discuss various challenges in implementing our analysis model in a real system.

3.1 Structured Aggregation Operators

As in traditional OLAP, we refer to the components of XML analysis as *analysis dimensions*. Our model comprises two types of analysis dimensions: value-based and structure-based dimensions. The former specify text contents in an XML document (which can be values of XML `attribute` or `text` nodes), while the latter specify tree patterns in an XML document (and therefore represent a set of valid pattern matches).

As an example, consider a human resources XML document storing the employee data (Figure 2). The organization consists of divisions (highest level), departments, and groups (lowest level). Each group again consists of employees that are members of this group. As indicated, each em-

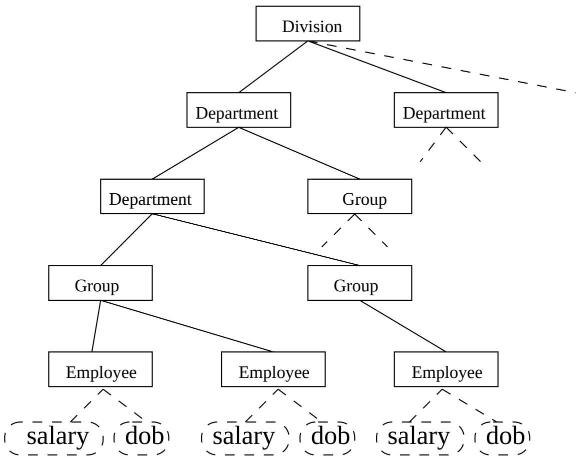

Figure 2: Human resources XML Document

ployee has attributes `salary` and `dob` (date of birth). For this example, a value-based analysis dimension would be `//employee/dob` which represents the set of all dates of birth of all employees. Similarly, a structure-based analysis dimension could be specified as `//division/{$x | $x/$y}/group`. This pattern (among others) matches to the following tree patterns:
`/division/department/group` and
`/division/department/department/group`. The first match binds the variable `$x` to the `department` nodes and the second match binds both variables `$x` and `$y` to `department`. Note that the pattern specification syntax is presented for illustrative purposes only. We envision the use of a notation that allows the specification of more general tree patterns as discussed by Chen et al. [6].

3.1.1 Structured XML Group-By

Consider an analytical query, *What is the average income per dob range?* over the human resources XML document. Consider that `dob` is already reported as ranges in the XML document. Then an XML query to answer this question could look like this (for simplicity, we use a slightly modified FLWOR [7] syntax.):

```
for $e in //employee
GROUP BY(//employee/dob)
let $s := avg( $e/salary )
return
         <ageGroup>
            <dob_range> $e/dob </dob_range>
            <avgSalary> $s </avgSalary>
         </ageGroup>
```

In this example, `//employee/dob` is a value-based analysis dimension. Note that one could also answer the question, *What is the average dob per salary range?*, by using `//employee/salary` as the analysis dimension. This is in contrast with the traditional OLAP where usually the set of measure attributes and dimensions is predefined and cannot be changed easily.

Consider another question, *What is the average salary per department level?*. This question requires the use of a structure-based analysis dimension:

```
var d,d1,d2=department
for $e in //employee
GROUP BY (//division/{$d | $d1/$d2}/employee )
let $s := avg( $e/salary ),
    $l := ( $d1 == NULL ? 1 : 2 )
return
         <levelGroup>
            <level> $l </level>
            <avgSalary> $s </avgSalary>
         </levelGroup>
```

In this case, the averaging in the `let`-clause is done over the `employee` nodes with the same number of `department` nodes on the path. Similarly, the following `group-by`

```
GROUP BY ( //division/{$d | $d1/$d2}/employee,
           //employee/dob ),
```

averages salaries over different age ranges in different department levels. Note that one can also use a Dewey-ID based numbering scheme [10] to encode the element level.

While simpler versions of the value-based grouping operations can be implemented using a nested XQuery (see Appendix G.2, XQuery Working Draft, 12 November 2003 [7] and [19]), current XQuery proposal can not express structural grouping operations.

3.1.2 Structured XML Roll-Up and Cube

We now describe structured roll-up and cube operations in the XML analysis model. As an example, assume that each group node in our example document (Figure 2), stores the corresponding budget information. An analyst may be interested in viewing the budgets per group but also at higher levels. This is similar to a traditional OLAP roll-up operation and could be written as follows:

```
var d, d1=department
for $b in //budget
GROUP BY ROLLUP( //division//group,
//division/$d//group, //division/$d/$d1/group )
let $s := sum( $b/value() ),
    $n := $b/../name()
return
     <levelGroup>
       <name> $n </name>
       <overallBudget> $s </overallBudget>
      </levelGroup>
```

As in the traditional OLAP, the `ROLLUP` clause specifies the order in which the roll-up occurs. In this example, the roll-up sequence is

```
( //division//group, //division/$d//group,
  //division/$d/$d2/group)
( //division//group, //division/$d//group )
( //division//group )
```

The result of this `ROLLUP` would therefore be the sum of budgets of the groups on the lowest level, followed by the sum of budgets of the groups on the second-highest level, followed by highest level groups. Note that the `ROLLUP` clause in the example uses the same variable `$d` in the last two dimensions. This ensures that the roll-up operation correctly uses the department hierarchy for aggregation.

We note here that the `ROLLUP` operator does not require the dimensions to be dependent as in this example. One can equally well imagine a roll-up such as

```
GROUP BY ROLLUP({//division/{$d | $d1/$d2}/employee},
//employee/dob)
```

which will compute aggregates first over employees in the same age-group within the same level and then over employees in the same group level.

Drill-down and cube operators can be defined similarly and will therefore be skipped for brevity. The interested reader is referred to Gray et al. [8] who discuss the generalization of group-by to roll-up and cube in more detail for the traditional OLAP setting.

Finally, we propose a special operator, called *topological roll-up* for performing roll-up aggregation along an XML hierarchy. Instead of specifying

```
GROUP BY ROLLUP(//division//group,
//division/$d//group, //division/$d/$d2/group)
```

one can use the `topological` specifier as follows:

```
GROUP BY TOPOLOGICAL ROLLUP(//division/$d/$d2/group)
```

The complete roll-up as discussed above could then be generated automatically by topological sorting of the XML structures. Similarly, drill-down and cube operators could be extended with the "`topological`" keyword to indicate automatic expansion or contraction of the XML structure. Thus the precise schema of the XML documents need not be known to the user. The simple expression

```
GROUP BY TOPOLOGICAL CUBE(//division/$d1/$d2/employee,
 //division/$d1/$d2/budget)
```

would then correspond to the much longer expression

```
GROUP BY CUBE(//division/employee,
//division/$d1/employee, //division/$d1/$d2/employee,
//division/budget, //division/$d1/budget,
//division/$d1/$d2/budget)
```

3.2 Implementation Challenges

We now discuss various challenges in implementing a real-life analysis system based on the proposed hierarchical XML analysis model.

1. **Supporting new XML analytics operators:** The XML analysis model proposed in Section 3 introduces several new aggregation operators such as the structural group by, topological roll-ups, and cubes. Efficient implementation of these operators in the XQuery framework would be challenging. Another area that needs further investigation is support for approximate structural and value queries.

2. **Supporting large XML documents:** Decision support systems traditionally deal with large datasets stored in data warehouses. In practice, terabyte-sized OLAP cubes are not uncommon. Similarly, data sources such as the Protein Data Bank (PDB) can generate XML documents with sizes of 100 GByte and more. Therefore, it is imperative that any XML analysis system is able to load and process large complex XML documents efficiently. Current XML processing systems tend to materialize an entire document in memory, which is clearly impractical. Techniques such as XML projection [17] are needed to materialize XML documents with smaller memory footprints.

3. **Creating complex XML views:** Until now, we have considered analysis of a single large XML document. In practice, there might be a large number of XML documents, potentially with different schemas. In such cases, one may need to compute XML views using multiple *base* XML documents. In theory, XQuery can be used for generating XML views from multiple base documents, but in reality, most current XQuery implementations can work on only one or two XML documents.

4. **Support for visualizing complex hierarchical data:** Visualization forms an important aspect of decision support systems. Visualizing large hierarchical datasets poses new challenges, e.g., how to view a large tree in real time with limited memory resources?, how to support zoom-in and zoom-out on the data tree?, how to support context-directed navigation?, or how to support XPath-based slice-and-dice?

5. **Integrating with a traditional OLAP system:** It is imperative that any XML analysis system should be integrated with a traditional OLAP system. The base XML data can be stored either in a relational database or as text documents. The former approach (ROLAP) allows easier integration with a traditional OLAP processor, while the latter, which can be termed XOLAP, allows flexibility of using existing XML processing languages (e.g., XQuery or XSLT) directly.

4. RELATED WORK

Over the years, Online Analytical Processing (OLAP) has been studied extensively. We report here a few related efforts. Vassiliadis and Sellis [25] present a survey of logical models for OLAP databases. Gray et al. [8] first introduced the OLAP `CUBE` operator. Chaudhuri and Dayal [5] present an overview of relevant concepts in data warehousing and decision support systems. Most database vendors support OLAP in their database systems. The OLAP Report [21] presents a detailed study of current industrial OLAP offerings. Current work in using XML for OLAP applications involves using XML for representing external data. To the best of our knowledge, no one has investigated exploiting XML's tree model for analytical purposes. Recently, Pedersen et al. have been exploring the integration of XML data with the traditional OLAP processing [20]. Jensen et al. describe how to specify multi-dimensional OLAP cubes over source XML data [15].

Recently, several researchers have proposed extensions to relational databases for supporting complex OLAP functionalities. Hurtado and Mendelzon [12] and Jagadish et al. [14] have investigated OLAP processing over heterogeneous hierarchies defined over relational data. Chaudhuri et al. [4] and Babcock et al. [1] have studied approximate query processing in the context of aggregation queries. Barbara and Sullivan have proposed Quasi-Cubes, for computing approximate answers in multidimensional cubes [2]. These approaches use approximation to reduce computation time over *precise* data. In our case, the source XML data is inherently imprecise. Lerner and Shasha recently proposed extensions to SQL for supporting order-dependent queries (AQuery) [16]. Carmel et al. have investigated approximate searching of XML documents using structural templates (called XML fragments) [3]. Navarro and Baeza-Yates have proposed a model to query documents by their content and structure [18].

5. CONCLUSIONS AND FUTURE WORK

In this paper, we investigated various issues in analyzing XML documents. We showed that the traditional multidimensional OLAP model is unsuitable for XML analysis due to its weakness in representing semantically-rich, semistructured XML data. We proposed a new logical model for XML analysis based on the abstract tree-structured XML representation. We demonstrated the practicality of the new model model by proposing new structural aggregation extensions for XQuery.

We are developing a prototype XML analysis engine to investigate various tradeoffs in data mapping and query translation schemes. In particular, we are evaluating relational-based and XML-based backend engines for functionality, space usage, and execution costs.

6. REFERENCES

[1] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*, pages 539–550. ACM Press, 2003.

[2] D. Barbara and M. Sullivan. Quasi-Cubes: Exploiting Approximations in Multidimensional Databases. *ACM SIGMOD Record*, 26(3):12–17, 1997.

[3] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML documents via XML fragments. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 151–158. ACM Press, 2003.

[4] S. Chaudhuri, G. Das, and V. Narasayya. A robust, optimization-based approach for approximate answering of aggregate queries. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 295–306. ACM Press, 2001.

[5] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *Data Mining and Knowledge Discovery*, 26(1):65–74, 1997.

[6] Z. Chen, H. V. Jagadish, L. V. S. Lakshmanan, and S. Paparizos. From Tree Patterns to Generalized Tree Patterns: On Efficient Evaluation of XQuery. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 237–248, September 2003.

[7] World Wide Web Consortium. W3C Architecture Domain: XML. `www.w3c.org/xml`. Online Documents.

[8] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, March 1997.

[9] Moving Pictures Experts Group. MPEG Standards. www.chiariglione.org/mpeg.

[10] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*, pages 16–27. ACM Press, 2003.

[11] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, January 1997.

[12] C. A. Hurtado and A. O. Mendelzon. Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. In *Proceedings of the International Conference on Database Theory*, pages 375–389, 2001.

[13] N. Huyn. Data Analysis and Mining in the Life Sciences. *ACM SIGMOD Record*, 30(3):76–85, 2001.

[14] H. V. Jagadish, L. V. S. Lakshmanan, and D. Srivastava. What can Hierarchies do for Data Warehouses? In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 530–541, September 1999.

[15] M. R. Jensen, T. H. Moller, and T. B. Pedersen. Specifying OLAP Cubes on XML Data. In *Proceedings of the 13th International Conference on Scientific and Statistical Database Management*, pages 18–20, July 2001.

[16] A. Lerner and D. Shasha. Aquery: Query language for ordered data, optimization techniques, and experiments. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 345–356, September 2003.

[17] A. Marian and J. Simeon. Projecting XML Documents. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 213–224, September 2003.

[18] G. Navarro and R. Baeza-Yates. Proximal nodes: a model to query document databases by content and structure. *ACM Trans. Inf. Syst.*, 15(4):400–435, 1997.

[19] S. Paparizos, S. Al-Khalifa, H. V. Jagadish, L. V. S. Lakshmanan, A. Nierman, D. Srivastava, and Y. Wu. Grouping in XML. In *EDBT Workshops 2002*, pages 128–147, 2002.

[20] D. Pedersen, K. Riis, and T. B. Pedersen. Query Optimization for OLAP-XML Federations. In *Proceedings of DOLAP 2002, ACM Fifth International Workshop on Data Warehousing and OLAP*, pages 57–64, November 2002.

[21] N. Pendse. The OLAP Report. Online Document. www.olapreport.com.

[22] E. Pourabbas and M. Rafanelli. Hierarchies and Relative Operators in the OLAP Environment. *ACM SIGMOD Record*, 29(1):33–37, 2000.

[23] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of IJCAI*, pages 448–453, 1995.

[24] J. Trujillo, S. Lujan-Mora, and I. Song. Applying UML and XML for designing and interchanging information for data warehouses and OLAP. *Journal of Database Management*, 15(1):41–72, 2004.

[25] P. Vassiliadis and T. Sellis. A Survey of Logical Models for OLAP Databases. *ACM SIGMOD Record*, 28(4):64–49, 1999.