社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DBS－140 (I) (28)
2006／7／12

XMLデータの分析処理に向けて

キットチャントラ[†]　天笠俊之[†,††]　北川博之[†,††]

概要

Extensible Markup Language (XML) は，ネットワークにおけるデータ表現・交換フォーマットになった．XML はあらゆる分野で標準的に使われるようになってきているため，今後はこれまでの単純な問合せ処理に加えて，新たな情報を発見するための複雑な分析処理が必要になることが予想される．そこで本研究では，XML の階層構造などの特徴を考慮した，分析処理のためのモデルを提案する．また，関係データベースシステムを利用した実装方法についても述べる．

Towards Analytical Processing of XML Data

Chantola KIT[†]　Toshiyuki AMAGASA[†,††]　Hiroyuki KITAGAWA[†,††]

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 a model for XML data analysis in that features of XML data, such as grouping according to tree structure of XML, are taken into account. Finally, we discuss an implementation of the system for XML data analysis using relational database systems.

1 Introduction

Since its inception in 1998, the Extensible Markup Language (XML) [1] has become a de facto standard for data exchange and representation on the web. Now, XML is being used in a wide spectrum of application domains, such as electronic documents, business data, and log data.

Traditionally, when a user wants to extract necessary information from XML data, his/her information need is expressed in terms of a query written in a particular query language such as XPath [2] and XQuery [3], or transformation languages like XSLT [4], and the query is then processed by XML processor and databases. However, as application domains of XML are continuously growing, we are under the pressure of the necessity for not only querying but also discovering information from XML data.

Let us take a look at a simple example of sales data in a bookstore chain. The sales data is recorded in XML format (Figure 1). It contains sales information at the bookstores in the chain. The sales data is geographically categorized by the XML hierarchy. The top-level is consisting of Kanto and Kansai areas, each of which contains Tsukuba, and Kyoto and Osaka, respectively. A book sales (b) contains title (t) and quantity (q). Another XML tree represents the book category (Figure 2). The XML hierarchy represents the category-subcategory relationships, and each book information is stored with its title (t) and price (p). Note that the name of each category (c) is given as the name attribute. Note also that the book titles are used as the key to identify each book in the both documents.

In such an application domain, it is often the case that one would like to calculate the total number of book sales with differentiating categories and areas. Another example is computing the most selling book category in terms of total sales with some ranges. An important implication here is that the category and area hierarchies are expressed in terms of XML hierarchies, and we need a mechanism to group any XML data of interest by such XML structures. In order to implement such systems, these features should carefully be taken into account.

[†]筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[††]筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba

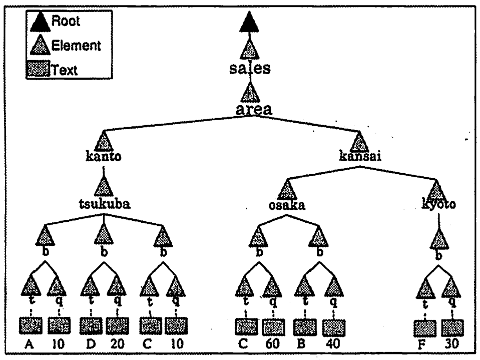

図 1: Sales data.

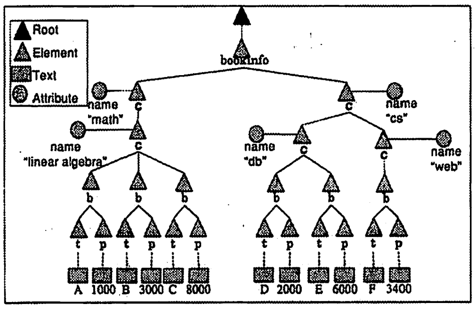

図 2: Book category.

In fact, in order to perform interactive analysis on relational data, *Online Analytical Processing (OLAP)* systems [5] have successfully been used for the past decade. An OLAP system enables us multidimensional viewing, analysis, and querying of large amount of data, and has been used for massive database decision support. However, when considering XML data, it is not sufficient in the sense that its inability to deal with non-numeric measures and XML hierarchies. As a consequence, we can conclude that using OLAP systems for XML data analysis is not sufficient to fulfill our requirements.

One might think that XML processors or databases that support XQuery could be used to analyze XML data. However, this does not work due to the fact that XQuery does not support grouping functionality, that is the key mechanism to perform multidimensional data analysis.

From the above reasons, in this research, we attempt to propose a model for XML data analysis. We also discuss how to integrate structure- and value-based hierarchies of XML data into the model and discuss a relational implementation of the proposed model.

The rest of this paper is organized as follows: in Section 2, we introduce preliminaries. We will introduce the concepts of fact path, dimension path, and XML data cube in Section 3, and discuss the concept hierarchy in Section 4. Section 5 discusses implementation issues. We discuss related works in Section 6 and conclude this paper in Section 7.

2 Preliminaries

2.1 Online Analytical Processing (OLAP)

Online Analytical Processing (OLAP) is a category of software technology that enables analysts, managers, and executives to gain 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 we talk about 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. We give an example of a book sales star schema (Figure 3), which contains a sales fact table and two dimension tables, bookinfo and store.

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 dimensions and the resolutions of dimensions of interest.

Referring to the same example, the cube, in Figure 4, shows a sales cube of three dimensions, area, category, and price. Starting from the cube at the upper left, we can create a new cube with coarser granularity on the area axis by applying roll-up operation. We can go back to the finer granularity by drill-down operation. By slicing on the price dimension, we can get only the common

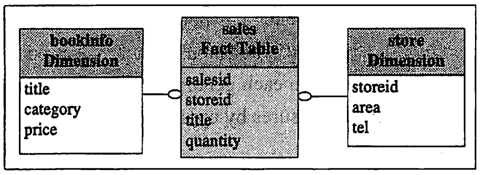

図 3: Star schema.

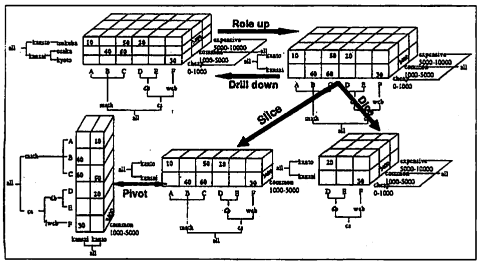

図 4: OLAP example.

price cube. Dice operation enables us to change order of the dimensions.

3 Data Cube on XML Data

This section discusses our proposed data cube model constructed on XML data. We first give the definitions of facts and dimensions, and then discuss how a data cube is constructed.

3.1 Facts about an XML Data

A fact-table in a traditional OLAP system stores data items of interest. We attempt to define the facts in an XML data after the traditional OLAP way. But the situation is different in the sense that we do not have dedicated XML data containing the facts of interest, whereas the fact-table is assumed to be given beforehand in OLAP systems. In order to identify the facts, we can use XML query languages such as XPath and XQuery. In this paper, we focus on XPath for the sake of simplicity.

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

For example, when a user wants to get information of book sales. The related data items can be obtained by the fact path p_f = `doc("sales.xml")//b`.

3.2 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 is a XPath expression (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 5 shows an example of fact and dimension paths. The circles on the left document represent the facts corresponding to p_f. When we would like 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 which is represented in another XML data, we need to specify absolute path expression with referential constraints like
p_{d2} =`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.

3.3 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. We additionally introduce evaluation with context. Let $[[p]]_c$ denotes an evaluation

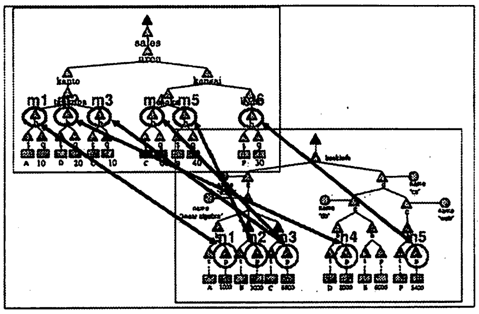

図 5: Fact path, dimension path, and their correspondence.

of p where p and c represent an XPath expression and a context node, respectively. If context nodes are given as a set like $[[p]]_C$, it can be defined as a natural extension: $\bigcup_{c \in C}[[p]]_c$.

Definition 3 (n-dimensional data cube) *A nD-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 nD-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_{di}: $[[p_{di}]]_f$ if p_{di} is in a relative form or $[[p'_{di}]]$ where p'_{di} can be obtained by replacing each occurrence of p_f/p_r in p_{di} with $[[p_r]]_f$.*

Let us consider an $1D$-cube as an example. It is specified by $(p_f, \{p_d\})$ where p_f =`doc("sales.xml")//b` and p_d =`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 $[[p_f]] = \{m_1, m_2, \ldots, m_6\}$. For each fact data m_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 m_1, $p_f/$`t`, which is a part of p_d, is rewritten as $[[p_f/\texttt{t}]]_{m1} = \{"A"\}$, that turns out to be `doc("bookinfo.xml")//b[t = "A"]/p`. In this way, we can extract all tuples from the data cube, that are set of 2-tuples: $\{(m1, n1), m2, n4), (m3, n3), (m4, n3), (m5, n2), (m6, n5)\}$.

Once the data cube is constructed, we can make analysis using 1) the dimensions and 2) related information such as XML hierarchies as clues to group the facts. An important remark is that unlike traditional OLAP systems, an XML data cube has potential axes, in addition to explicitly specified dimensions, that are derived from XML hierarchies. To each generated group, we can calculate specified measures by applying aggregation functions.

4 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. When dealing with XML data in the same context, we need special consideration on its semistructured nature. In addition to value-based concept hierarchy that has been used in traditional systems, we have to take into account structure-based concept hierarchy represented as XML tree.

4.1 Value-based Concept Hierarchy

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, because it is beyond the scope of this paper.

4.2 Structure-based Concept Hierarchy

In addition to the value-based concept hierarchy, there is a need to group fact data according to the hierarchical structure of XML data. This kind of concept hierarchy is called "structure-based." More precisely, there are two cases in this category according to the way how category information is represented. The first case is called "direct" where the concepts are directly represented by tag names. For example, in Figure 1, the geographical information of each book can be observed by looking at the path expression like `/area/kanto/tsukuba`. The other case is called "indirect." This is different from the direct case in the sense that the name of each category is not presented explicitly by XML tags, although the categorization is expressed in terms of inclusion relationship of XML elements. For example, in Figure 2, it is possible to categorize each book according to the XML hierarchy, but the name of each category is not explicitly represented by the element names (/bookinfo/c/c).

4.3 How to Realize?

We discuss here how to realize grouping operation based on structure-based concept hierarchy. Our basic strategy is to utilize path expressions. We will divide the subsequent discussions into two cases.

For the case of direct structure-based hierarchy, we can just use path expressions as the key to perform grouping, because the tags directly represent category information. For a given data items, 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.

For the indirect cases, the problem becomes a bit complicated. As discussed above, using path expressions is not enough, because category information is embedded in attributes or elements. So, we need to extract the necessary information from the XML data in a preprocess, that is, a user has to specify his/her information need using an XML query language or a dedicated language, and the system can extract the information. Such information can also be represented as path-like expressions. For example, in the example of Figure 2, each book can be categorized one of /math/linear algebra, /cs/db, or /cs/web. We call such expression "context expression." When dealing with indirect structure-based hierarchy, we can use context expressions instead of path expressions.

Now, we discuss how to perform grouping operations using path (context) expressions. Let us introduce some notations. Given an XML node n, let $pexp(n)$ ($cexp(n)$) denote n's absolute path (context) expression, and let $prefix(exp, i)$ denote path (context) expression exp's i-th prefix, e.g., $prefix(\text{"/a/b/c"}, 1) = \text{"/a"}$, and $prefix(\text{"/a/b/c"}, 2) = \text{"/a/b."}$ Then, the grouping

1. Let the depth of the dimension be d.

2. Find the common prefix of the path (or context) expressions and let the depth be i.

3. The level-j ($i \leq j \leq d$) grouping can be computed by calculating $prefix(pexp(n), j)$ ($prefix(cexp(n), j)$) for each dimension value n.

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, which will be discussed in the next section.

5 Implementation using Relational Database Systems

This section discusses an implementation of the proposed model and grouping operations. 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 functionality, that is 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 [6] 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 the brief overview. In the path-approach, an XML node is basically mapped to a relational tuple with the document ID, the path expression from the root node, the node label by which we can preserve ancestor-descendant relationship, and the node values. Table 1 shows a storage example. The attributes did, pexp, cexp, pre, post, type, and value denote document id, path expression, context expression, pre-order, post-order, node type (element, text, or attribute), and node value, respectively. Note that we assume that the way how to extract context expressions is given by the user, and how to express such an request will be discussed in another article.

5.2 Data Cube Formulation

For the purpose of data cube formulation, we also would like to make the best use of the functionality of the un-

表 1: Node table.

did	pexp	cexp	pre	post	type	value
1	/sales		1	74	E	
1	/sales/area/kanto/tsukuba/b		5	14	E	
1	/sales/area/kanto/tsukuba/b		15	24	E	
1	/sales/area/kanto/tsukuba/b		25	34	E	
1	/sales/area/kansai/osaka/b		39	48	E	
1	/sales/area/kansai/kyoto/b		49	58	E	
1	/sales/area/kansai/kyoto/b		61	70	E	
1	/sales/area/kansai/kyoto/b/t		63	64	T	F
...	...	...	...	...	...	...
2	/bookinfo		1	72	E	
2	/bookinfo/c		2	35	E	
2	/bookinfo/c/c/b	/math/linear algebra	4	13	E	
2	/bookinfo/c/c/b	/math/linear algebra	14	23	E	
2	/bookinfo/c/c/b	/math/linear algebra	24	33	E	
2	/bookinfo/c/c/b	/cs/db	38	47	E	
2	/bookinfo/c/c/b	/cs/web	60	69	E	
2	/bookinfo/c/c/b/t	/cs/web	62	63	T	F
...	...	...	...	...	...	...

derlying database systems. Actually, we can use SQL to formulate the data cube on XML data discussed in Section 3.3.

Let us take a look at an example:

- p_f =`doc("sales.xml")//b`
- p_{d1} =`doc("bookinfo.xml")//b[`p_f`/t=t]`

In order to create a data cube, we need to establish the relationships between the fact and the dimension as described in Section 3.3. Here we attempt to use SQL to perform the process. Since fact and dimension paths are given in terms of XPath, we can translate them to the corresponding SQL queries. Having fixed the base relations, we next join the base relations by giving the referential constraints as the join key.

```
SELECT fct.*, dim.*
FROM
  -- Fact path
  (SELECT fct1.*, fct2.value as jkey
   FROM node fct1, node fct2
   WHERE fct1.did = fct2.did
   AND fct1.pexp like '/%/b'
   AND fct2.pexp like '/%/b/t'
   AND fct1.type = 'E' AND fct2.type = 'T'
   AND fct1.pre < fct2.pre
   AND fct2.post < fct1.post) fct,
  -- Dimension path
  (SELECT dim1.*, dim2.value as jkey
   WHERE dim1.did = dim2.did
   AND dim1.pexp like '/%/b'
   AND dim2.pexp like '/%/b/t'
   AND dim1.type = 'E' AND dim2.type = 'T'
   AND dim1.pre < dim2.pre
   AND dim2.post < dim2.post) dim
 -- Join condition
 WHERE fct.jkey = dim.jkey;
```

This query has two subqueries, each of which corresponds to fact and dimension path. The where clause gives the join condition over attributes in the subqueries. Finally we get a new XML table as shown in Table 2. The table contains all attributes from fact and dimension and each record consists of data from fact and dimension which have the same title.

Since the XML data cube is constructed solely by SQL, it is up to us to choose to materialize it for speeding up the rest of the analysis or not.

5.3 Implementing Grouping Operation

We are now ready to apply grouping operation against the generated XML data cube. For value-based concept hierarchy, the task is easy because we can assume that the concept hierarchy is given as an extra relational table. So, we do not go into the detail.

For structure-based concept hierarchy, as discusses in Section 4.3, the clue to perform grouping is the path expressions. 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 `/a/b` out of `/a/b/c/d`, we can achieve this by:

```
SELECT ...
FROM ...
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.

6 Related Work

Rajesh Bordawakar et al. [7] 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. They also proposed new syntactical extensions to XQuery for supporting complex analytical operations, and discussed various challenges.

Mikael R. Jensen et al. [8] proposed a scheme for specifying OLAP cubes on XML data. They integrated XML

表 2: XML Data Cube Example.

did	pexp	cexp	pre	post	type	value	jkey	did	pexp	cexp	pre	post	type	value	jkey
1	/sales/area/kanto/tsukuba/b		5	14	E		A	2	/bookinfo/c/c/b	/math/linear algebra	4	13	E		A
1	/sales/area/kanto/tsukuba/b		15	24	E		D	2	/bookinfo/c/c/b	/cs/db	38	47	E		D
1	/sales/area/kanto/tsukuba/b		25	34	E		C	2	/bookinfo/c/c/b	/math/linear algebra	24	33	E		C
1	/sales/area/kansai/osaka/b		39	48	E		C	2	/bookinfo/c/c/b	/math/linear algebra	24	33	E		C
1	/sales/area/kansai/kyoto/b		49	58	E		B	2	/bookinfo/c/c/b	/math/linear algebra	14	23	E		B
1	/sales/area/kansai/kyoto/b		61	70	E		F	2	/bookinfo/c/c/b	/cs/web	60	69	E		F

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.

Dennis Pedersen et al. [9] proposed a federation of OLAP and XML, which allows external XML data to be presented along with dimensional data in OLAP query results. It enables the uses of external XML data for selection and grouping. It is the same to the third approach mentioned by Rajesh Bordawakar et al. [7]. They allow XML data to be used as "virtual" dimensions, and present a data model and multi-schema query language based on SQL and XPath.

7 Conclusions

In this paper we discussed a model for XML data analysis and implementation issues using relational databases. We first introduced the concepts fact path, dimension path, and XML data cube, and then discussed value- and structure-based concept hierarchy, which is a novel concept in the context of XML. Regarding implementation issues, we utilized the path-approach for mapping XML data to relations, and showed that XML data cube can be constructed using the functionality of SQL on top of the relations. For the future work we plan to evaluate the proposed method. We also plan to investigate how to incorporate textual features such as word vectors of XML data into the analytical processing.

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.

参考文献

[1] World Wide Web consortium: Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/REC-xml. W3C Recommendation 04 February 2004.

[2] World Wide Web consortium: XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/1999/REC-xpath-19991116. W3C Recommendation 16 November 1999.

[3] XQuery: A query language for XML, http://www.w3.org/TR/xquery. W3C working draft 2001.

[4] World Wide Web consortium: XSLT Transformations (XSLT) Version 2.0, http://www.w3.org/TR/2006/CR-xslt20-20060608. W3C Candidate Recommendation 8 June 2006.

[5] Saurajit Chaudhuri and Umeshwar Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, 1997.

[6] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, and Shunsuke Uemura. XRel: A path-based approach to storage and retrieval of XML documents using relational databases. *ACM Transactions on Internet Technology (TOIT)*, 1(1):110–141, 2001.

[7] Rajesh Bordawakar and Christian A. Lang. Analytical Processing of XML Documents: Opportunities and Challenges. *SIGMOD Record*, 34(2):27–32, 2005.

[8] Mikael R. Jensen, Thomas H. Moller, and Torben Bach Pedersen. Specifying OLAP Cubes on XML Data. *SSDBM,* pages 101–112, 2001.

[9] Dennis Pedersen, Karsten Riis, and Torben Bach Perdersen. XML-Extended OLAP Querying. *SSDBM,* pages 195–206, 2002.