XQuery & OLAP

Michael Seiferle, michael.seiferle@uni-konstanz.de

2009-04-09

Abstract

Online Analytical Processing on XML became a topic in the database research community[1; 2; 3; 4; 5], which is mostly due to the possibilities of XML to overcome the constraints of the relational multidimensional model[6; 7].

This paper gives an overview of chances and challenges in terms of storing multidimensional heterogenous data in XML and how to query it. We use BaseX, a native XML database developed at the chair for Database and Information Systems (U Konstanz) as a back-end to store and query the data.

The first part of this paper briefly deals with modeling aspects and provides an overview on how to store metadata for non-conventional (i.e. text, compound measures, irregular dimensional hierarchies) data in XML. While we do propose a metaschema for this multidimensional model, we try to allow a certain degree of freedom with respect to the characteristics of the actual data to be stored inside the data warehouse. Throughout this paper XML data from an example Bookstore will be used to support our concepts.

The second part focusses on querying heterogenous and imprecise data. As XQuery 1.0[8], at the moment, only offers very basic support for multidimensional querying, we first explore what enhancements are needed to meet up with Relational OLAP in terms of querying. In contrast to Relational OLAP where missing, incomplete and de-normalized data is avoided at the schema layer, *OLAP-enabled* XQuery offers a whole new perspective on querying non conventional data. We will show how heterogenous schemata, possibly derived from numerous data sources may, to some extent, be queried without prior normalization. We achieve this with the help of relaxing the group by operation. We add an optional granularity argument such that summarizability requirements are still met. Although our research on OLAP-enabled XQuery in BaseX has just begun, we think that our approach is promising for real world applications.

The last section offers some very basic ideas on integrating Full-Text features in multidimensional queries. We will give an future prospect on how XQuery's Full-Text abilities in conjunction with OLAP may be used to analyze textual contents.

1 Introduction

XML has become one of the premier formats for data interchange and information representation on the web[1]. The main reasons that led to this success are mostly due to XMLs simplicity and expressiveness: In fact XML allows us to store arbitrary information with respect to some simple rules. Wether it is relational data, semi-structured data or even binary objects – they are all represented in a uniform manner and syntax. Lots of XML datasets are available today that would not have to converted to the relational model while on the other hand the ETL process could easily be changed to create XML instead of relational data.

In accordance to [6; 7] there are challenges that data warehouses for real world data still face today, among them are:

1. Warehousing imperfect and unstructured data
2. Querying imperfect and unstructured data

3. Full Text Capabilities

Researchers propose new concepts[5; 6; 7] where data and data models are in fact two views upon the same entity and should not be constrained by the strict rules of multidimensional model. We think that XML might be a huge step in overcoming these constraints. The increased complexity of data representation in XML over relational storage in terms of hierarchical modeling and compound data-types leads to a much more flexible data-model.

2 Application Scenario

2.1 Modeling & Storage Aspects

> The fundamental goals of a data warehouse can be developed by walking around the halls of any large organization and listening to the management talk. [...]
> These concerns are so universal that they drive the bedrock requirements for the data warehouse. [9, S. XXV]

Kimball's quote elucidates how versatile, and thus universal, both possible queries and data structures inside the data warehouse are. Up until today developing a suitable data model for complex business data is a challenging task. During model development the user makes assumptions and simplifications that might prove as a restriction in later stages of data warehousing. Wether this happens because of ignorance or simply because there is no need to analyze that particular facet of data at that given time: Making changes to the conceptual model afterwards often becomes even more of a challenge. Throughout this paper we will use a minimal bookstore example to illustrate basic concepts and challenges that arise when performing OLAP on XML data.

2.1.1 XQuery Enriched Metadata

As we plan to use XQuery for analysis queries using XQuery in conjunction with XML as a metadata layer comes in handy. Our approach is *XQuery enriched metadata* where Facts, Measures & Dimensional Hierarchies are accessed using XQuery / XPath expressions. These *access paths* into *primary data* may be defined straightforward in terms of XPath locations steps or might as well be more complex algorithmic XQuery expressions.

This approach offers three advantages, first and foremost we use *one* well defined interface to access both metadata and primary data. Second we may use the information coded in our metadata layer to generate the graphical user interface and provide the user a visual exploration tool to analyse his data. Third we may use the XQuery expressions stored in our metadata to automatically expand parts of our queries according to our metadata. If, for example, the user wanted to group the data based on the hour the sale took place we could either explicitly model a hierarchy that contains *hour members* (Houro, ..., Hour23) or we could implicitly model the *hour members* as a XQuery function i.e.`fn:hours-from-dateTime(facts/pathTo/dateTime)`. In addition this would offer us the possibility to perform some very basic schema integration tasks, although it is not very likely that this can be done with sufficient performance for real time analysis.

2.2 OLAP Extensions for XQuery

For XML to meet up with Relational OLAP several query language solutions have been proposed, among them by Hümmer et. al. [3] the formulation of OLAP queries as XML-fragments sent to a database server that in return sends a XML-fragment containing the inquired analysis data. However solutions that build upon XML databases and a standardized query language (e.g. XQuery) are rarely used.

We will however, show that using XML in conjunction with XQuery will be able to meet up with the capabilities provided by relational OLAP implementations. In order to do so we first gather the required tools and operators known from relational OLAP and adapt them for XQuery. As for today in [10] there exists a working draft that provides all operations necessary for OLAP. In the following section we will concisely discuss those concepts and how they resemble to the relational OLAP analysis workflow.

2.2.1 Heterogenous & Imprecise data

In contrast to Relational OLAP where missing, incomplete and de-normalized data is avoided at the schema layer, XML offers a whole new perspective to store and query non conventional data. Heterogenous schemata possibly derived from numerous data sources may, to some extent, be queried without prior normalization. If we look at the simple XML tree storing sales information from two example bookstores (Figure 1), we notice the heterogenous structure of the XML file, where a book has one or more authors.

2.2.2 Value Based Grouping

Value Based Grouping is described in [10, Section 3.8.7]. The XQuery `group by` works similar to the standard SQL operator as it only allows grouping based on (XQuery) atomic values.

A simple query, getting the per author, per date sales of books in traditional OLAP has to take the 1:n relation between book and author into account. XQuery allows us to do this in a much more convenient way. We "relax" the grouping criterion on author by skipping the intermediate `authors`-Node, using the following group by clause:

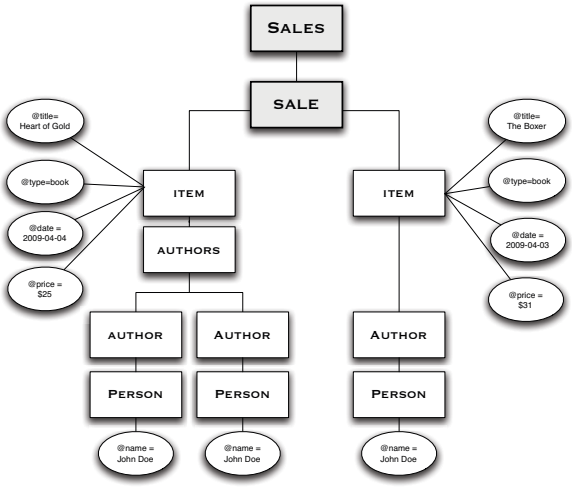

Figure 1: Example XML Document

Example 1

```
group by /sale/item[@type="book"]//author, /sale/item[@type="book"]/@date
```

which as a result yields three tuples for the two
books, each one uniquely defined by its unique (Author, Date)-pairs. This might seem counter intuitive for our analysis requirement as the first book ("Heart of Gold") would return two distinct results for either author. This is why we need to define a concept of uniqueness for any given node with respect to the measure under consideration. To achieve a both flexible and easy to use solution we propose a modification to the relational grouping operator that adds adjustable granularity:
`group by` $/step_0/.../step_n$ [`granularity` $/step_0/.../step_n$] It defines which node is used as the unique group identifier.

The modification `granularity` $/step_0/.../step_n$ resembles to the traditional `group by`.

Reclaiming **Example 1** we would define the granularity level by:

Example 2

```
group by /sale/item[@type="book"]//author, /sale/item[@type="book"]/@date
  granularity /sale/item
```

which would result in only two groups for the nodes satisfying `/sale/item[@type="Book"]`. Please note that the granularity definition does not take the heterogenous structure of the author nodes into account, as we only take their (common) parent nodes into account. This extension is similar to the tree pattern relaxations proposed by [11] and exploits the XML structure in a straight forward way.

Using this concept all subsequent aggregate functions like `sum`, `avg` object to summarizability and return intelligible results to the user.

2.2.3 Partitioning: The Window Clause

With the introduction of the window clause in [10, Section 3.8.4] we gain the ability to perform *time series analyses*. XQuery 1.1 defines two kinds of window clauses that allow us to iterate over a defined sequence of items. We can use both, overlapping (sliding) and exclusive (tumbling) windows to compute Measures.

A typical scenario might be generating an moving three day average of the per author, per date sales from Example 1. In order to so we partition the the resulting items using the window clause:

```
for tumbling window $w in $resultsFromExample1
                                    (: $resultsFromExample1 contains: SUM of sales per
                                      day per author :)
    start at $s when fn:true()      (: set $s to the first item :)
    only end at $e when $e - $s eq 2 (: put the subsequent 2 items into $w :)
                                    (: $w now contains 3 subsequent cells per day :)
return
 <cell info="3 day sales average for $s/author from {$s//date} to {$e//date}">
   { avg($w) }
 </cell>
```

2.3 Group by Cube & Rollup

As the Cube & Rollup operations are special cases of concatenated GROUP BY clauses we can, for the moment, simply mimic this behaviour: For a Rollup over n-Attributes we calculate the (n+1)-combinations of

$$(A_1, \ldots, A_n) \ldots (A_1, \ldots, A_{n-1}), \ldots (A_1, A_2), (A_1), ()$$

to compute each super aggregate for the given dataset and union the respective results. As manually formulating this (n+1)-UNIONS would be very lengthy we add the convenience operator GROUP BY ROLLUP that automatically expands to (n+1)-UNIONS.

The same holds for the GROUP BY CUBE operation that expands to all 2^n possible concatenations of grouping attributes.

2.4 Full-text operations

Besides traditional OLAP we think Full-Text analysis capabilities are crucial to suite warehousing needs and provide a convenient way to analyze non conventional textual data. Full-Text Operations could be used to:

(1) Group the nodes based on their textual content, based on ad-hoc user defined criteria.

(2) Provide custom consolidation paths in conjunction with a thesaurus, based on the specified nodes textual contents.

(3) Criteria for Slice & Dice operation to confine the analyzed nodes.

Consider for example the following XQuery code snippet querying real world patent data:

```
group by  $x//case-file-statement/text[. ftcontains "database"
      and(. ftcontains "biologic" or "chemical")]
```

In case of grouping the following code partitions our nodes into two distinct spaces, each containing the nodes that met the requirements.
A slightly more advanced approach could be grouping the nodes based on their relevance ranking:

```
let $score := score
group by  fn:floor($score * 10)
where $x//case-file-statement/text[. ftcontains "database"
      and(. ftcontains "biological" or "chemical")]
```

Where each group consists of members with a similar relevance for the given keywords. Although these ideas do not conform to the traditional OLAP tasks they might offer new possibilities during the analysis of data.

3 Conclusion

We have shown that XML/XQuery is suitable for OLAP operations and is able to meet up with relational OLAP in terms of analysis needs.

Thanks to XQuery's capabilities as a programming language OLAP operations may – at least in theory – be as complex as the user wants them to be, in both metadata modelling and primary data analysis.

With *XQuery enhanced Metadata* we are able to allow more complex metadata definitions that may be used to easily access complex underlying primary data.

The first steps will be taken with the implementation of the XQuery 1.1 Draft. After these initial steps we will need benchmarks to identify if there are typical access patterns to work on further optimizations. The second step will include building a metadata driven GUI to visually explore OLAP aggregates and provide the user with an appropriate tool for querying this data.

Another big challenge for future work will be the integration of materialized cubes, that by design, are not organized hierarchically and thus require special treatment in serialization.

References

[1] H. Wang, J. Li, Z. He, and H. Gao, "Olap for xml data," *Computer and Information Technology*, Jan 2005.

[2] B. Park, H. Han, and I. Song, "Xml-olap: A multidimensional analysis framework for xml warehouses," *Lecture Notes in Computer Science*, Jan 2005.

[3] W. Hümmer, A. Bauer, and G. Harde, "Xcube: Xml for data warehouses," *... 6th ACM international workshop on Data warehousing and OLAP*, Jan 2003.

[4] M. Jensen, T. Møller, and T. Pedersen, "Specifying olap cubes on xml data," *Journal of Intelligent Information Systems*, Jan 2001.

[5] A. M. Tjoa, A. Rauber, P. Tomsich, and R. Wagner, "Olap of the future," pp. 153–166, 2007.

[6] T. Pedersen, "Warehousing the world: a few remaining challenges," in *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pp. 101–102, ACM New York, NY, USA, 2007.

[7] S. Mansmann, *Extending the OLAP Technology to Handle Non-Conventional and Complex Data*. PhD thesis, Universität Konstanz, Universitätsstr. 10, 78457 Konstanz, 2008.

[8] W3C Recommendation, "XQuery 1.0: An XML Query Language," 23 January 2007.

[9] R. Kimball, *The data warehouse toolkit: practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc. New York, NY, USA, 1996.

[10] W3C Draft, "XQuery 1.1 W3C Working Draft," 3 December 2008.

[11] N. Wiwatwattana, H. Jagadish, and L. Lakshmanan, "X3: A cube operator for xml olap," *23rd International Conference on Data Engineering (ICDE 07)*, Jan 2007.