Algebra-Based Optimization of XML-Extended OLAP Queries

Xuepeng Yin Torben Bach Pedersen

Department of Computer Science, Aalborg University, 9220 Aalborg Ø, Denmark, {xuepeng, tbp}@cs.aau.dk

Abstract

In today's OLAP systems, integrating fast changing data physically into a cube is complex and time-consuming. Our solution, the "OLAP-XML Federation System," makes it possible to reference the fast changing data in XML format in OLAP queries without physical integration. In this paper, we introduce the novel query optimization techniques specialized for the federation system including a query optimizer and plan transformation rules. We also show the experimental results which suggest that our approach, unlike the physical integration, is a practical solution for integrating fast changing data into OLAP systems.

1 Introduction

Current OLAP systems have a common problem in handling the situations where changes in data requirements are common and data changes frequently. Physical integration of new data into OLAP systems is a long and time-consuming process. The increasing use of XML suggests that the required data will often be available in XML format. Therefore, a logical integration of OLAP and XML data is desirable.

Our overall solution is to logically federate the OLAP and XML data sources, decorating the OLAP cube with virtual dimensions based on external XML data, and thereby allowing selections and aggregations to be performed over the decorated cube. In this paper, we extend previous work [10, 14] by presenting the novel query optimization techniques specialized for the logical federation system, including a functioning implementation of the query optimizer for the OLAP-XML query engine and a set of plan transformation rules based on the logical algebra of OLAP-XML federations. We also show the experiments on the query engine implemented with the above techniques, with respect to federation performance, optimization effectiveness, and feasibility, suggesting that the logical OLAP-XML federation system can be the practical solution to gaining flexible access to fast changing data in XML format from OLAP systems.

There has been a great deal of previous work on data integration, for instance, on relational data [4, 5, 9], semi-structured data [6], and a combination of relational and semi-structured data [1, 7]. However, none of these handles the advanced issues related to OLAP systems. Some work concerns integrating OLAP and object databases [3, 8] with complex associations. In comparison, using XML as data source, as we do, enables the federation to be applied on any data as long as the data allows XML wrapping, greatly enlarging the applicability.

International Conference on Management of Data
COMAD 2006, Delhi, India, December 14–16, 2006

2 OLAP-XML Federations

A federation contains an OLAP *cube* and the XML documents. A cube contains measured values (*measures*) that are characterized by *dimensions*. A dimension is structured using *levels* of different details. The fundamental part of an XML document is the element node, which can contain other element nodes. Another component of a federation is *links*. Links are created by users or DBAs between existing dimensions and XML data, allowing external data to characterize the OLAP measures as extra dimensions. The fundamental linking mechanism is a relation between one dimension value in a cube and one node in an XML document. Suppose TC is a cube based on the TPC-

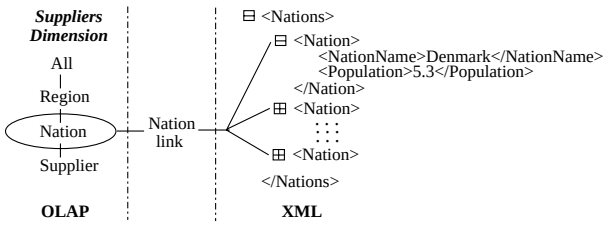

Figure 1: Linking OLAP and XML

H benchmark [12] with the dimensions, Suppliers (All-Region-Nation-Supplier), Parts (All-Manufacturer-Brand-Part), Orders (All-Customer-Order), and Time (All-Year-Month-Day) charactering the measure, Quantity (of parts in orders). Nation names and corresponding population data (in millions) are available in an XML document. Figure 1 shows an example link, Nation link (Nlink), that connects the dimension values of Nations to the Nation nodes that have the same text values in the sub-nodes, NationName, in the XML document. The plus/minus symbol in a box indicates whether the element is folded/unfolded. With Nlink, the population information about nations

can be referenced in OLAP queries. The example federation query, "SELECT SUM(Quantity), Brand, Nation/Nlink/Population FROM TC WHERE Nation/Nlink/Population<30 GROUP BY Brand, Nation/Nlink/ Population," shows the total quantity of the parts of each brand sold by each nation, where a nation is decorated with its population. Nation/Nlink/Population, is a *level expression*, where Population is a relative XPath expression applied to the XML nodes in Nlink to identify new nodes. A level expression allows decoration of dimension values (e.g., nation names) with XML values (e.g., populations) in the context defined through links.

3 The OLAP-XML System

The federation system has three major components: *query analyzer*, *query optimizer*, and *query evaluator*. Given a query, the query engine parses and analyzes the query, and generates the initial logical plan. The query optimizer generates a plan space for the initial plan and searches for the best execution plan (which has the least execution time) and then passes the plan on to the query evaluator, which generates the final results. The query optimization is a Volcano-like [2], *rule-based* process. However, the optimizer is a totally different implementation specialized for OLAP-XML federations, including: 1) faster logical plan enumeration (by a factor of five or more), because the logical plans are considerably smaller than physical plans without integrating the detailed data retrieval and manipulation algorithms, 2) novel *transformation rules* (see below) specialized for OLAP-XML federations, 3) a novel cost-model for the federation components [13], which have a high degree of autonomy , and 4) the inlining technique that rewrites selection predicates (see below).

4 Logical Algebra and Plan Transformation

Decoration A decoration operator, $\delta_{l_{xp}}$, builds a virtual (decoration) dimension using the XML data referenced by a level expression l_{xp}. The decoration operator enables the external XML data to become virtually a part of the cube, thereby allowing the following operations involving XML data to be performed on the federation.

Federation Selection A federation selection operator, $\sigma_{Fed[\theta]}$, allows the facts from the cube to be filtered using the external XML data as well as the regular cube data.

Federation Generalized Projection The generalized federation projection operator, $\Pi_{Fed[\mathcal{L}]<F(M)>}$, also let the federated cube be aggregated over the external XML data. Here, $\mathcal{L}$ is a set of levels to which the federation will be rolled up to, intuitively, the levels in the GROUP BY clause where level expressions can be present. *F(M)* is a set of aggregate functions over the specified measures, i.e., the aggregates in the SELECT clause.

When an SQL$_{XM}$ query is parsed and analyzed by the query engine, it is expressed in the logical algebra above and turned into a logical query tree. Figure 2 shows the query tree for the example query in Section 2, where $\mathcal{F}_{TC}$ represents the federation involving the TC cube. The query tree implies the query can be evaluated in three major steps. First, the decoration operator instantiates the level expression and builds a virtual dimension consisting of the population data to decorate the associated suppliers' nations through Nlink. The dimension schema is (All-Nation/Nlink/Population) or simply (All-Population). Then, the selection operator slices the cube using the population data, and finally the federation generalized projection operator rolls up the levels of dimensions to Brand, Population, and All (the top level of the dimensions not referenced in the SELECT clause), which leaves only the Parts and Suppliers dimensions in the federation, and produces the final aggregate results.

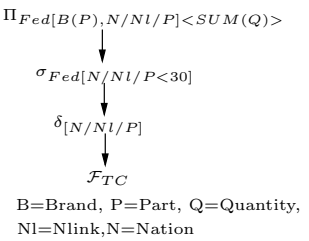

Figure 2: The logical plan

Transformation Rules Transformation rules are used during query optimization to enumerate equivalent plans that generate less intermediate data. Although some rules are variants of existing rules in relational systems [11], all the rules are specialized for OLAP-XML federation with the hierarchical structure and the virtual dimensions taken into consideration. Following this, Rules 1 and 2 can be thought of as being developed from well-known relational rules on select and group-by. Rules 3, 5, and 6 aim to optimize cases when the special OLAP-XML decoration operator is involved, and thereby are novel. Rule 4, which concerns inlining, is also novel.

In the following, let l denote a dimension level and $\sqsubset$ denote the partial order between two levels. we say $l'_i \sqsubset_i l_i$ holds if and only if the values of the higher level l_i contain the values of the lower level l'_i. For example, in the Time dimension, $Day \sqsubset_{Time} Year$ because years contain days. Also, the plan expression on one side can be reconstructed to the other side following an arrow.

A federation generalized projection cannot roll up the dimensions to levels higher than those referenced by the federation selection executed afterwards because these levels would no longer exist after the roll-up. Therefore, a federation generalized projection operator and a selection operator are commutative if the levels referred by the selection are not projected away by the projection.

Rule 1 (Commutativity of Federation Generalized Projection and Selection) *If θ does not reference measures, and for each level l_i in θ there exists a level $l'_i \in \mathcal{L}$ such that $l'_i \sqsubseteq_i l_i$, the following rule holds:* $\Pi_{Fed[\mathcal{L}]<F(M)>}(\sigma_{Fed[\theta]}(\mathcal{F})) \leftrightarrow \sigma_{Fed[\theta]}(\Pi_{Fed[\mathcal{L}]<F(M)>}(\mathcal{F}))$.

Rule 1 does not apply if the projection above rolls up the cube to the levels higher than those referred by the selection's predicate. However, a part of a federation generalized projection can be performed prior to the federation selection below, meaning that the cube can be rolled up to certain levels without interfering the selection.

Rule 2 (Pushing Federation Generalized Projection Below Selection) *If θ does not reference measures, and for a level l_i in θ there exists a level $l_i' \in \mathcal{L}$ such that $l_i \sqsubset_i l_i'$, the following rule holds:* $\Pi_{Fed[\mathcal{L}]<F(M)>}(\sigma_{Fed[\theta]}(\mathcal{F})) \rightarrow \Pi_{Fed[\mathcal{L}]<F(M)>}(\sigma_{Fed[\theta]}(\Pi_{Fed[\mathcal{L}']<F(M)>}(\mathcal{F}))$, *where, $\mathcal{L}'$ are the highest levels that the new generalized projection can aggregate the cube to and still allow θ to be evaluated, i.e., levels referenced by θ still exist in the cube.*

A federation selection reduces the tuples while they are passed onto the upper part of the plan. To use less temporary space and reduce data transfer between the OLAP component and the temporary component, we try to push a federation selection below a decoration operator if the predicate does not reference the level expression.

Rule 3 (Pushing Federation Selection Below Decoration) *Let S_θ be the set of levels involved in θ and l_{xp} be a level expression. If $l_{xp} \notin S_\theta$, the following rule holds:* $\sigma_{Fed[\theta]}(\delta_{l_{xp}}(\mathcal{F})) \rightarrow \delta_{l_{xp}}(\sigma_{Fed[\theta]}(\mathcal{F}))$.

Example 4.1 $\sigma_{Fed[Customer="Customer\#01"]}(\delta_{Nation/Nlink/Population}(\mathcal{F}_{TC}))$ is equivalent to $\delta_{Nation/Nlink/Population}(\sigma_{Fed[Customer="Customer\#01"]}(\mathcal{F}_{TC}))$. The second plan first filters the facts before it is decorated and the same decoration dimension can still be created using the unchanged dimension values and linked data.

As Rule 3 implies, a federation selection operator cannot be pushed below a decoration operator if the predicate references a level expression. However, the rule can still be applied if a new predicate which does not involve the level expression but selects the same facts is used to replace the old predicate. The technique to generate the new predicate based on the original one is inlining [13].

Rule 4 (Inlining of Decoration in Federation Selection) *If the predicate θ contains references to the level expression l_{xp}, the following rule holds:* $\sigma_{Fed[\theta]}(\delta_{l_{xp}}(\mathcal{F})) \rightarrow \delta_{l_{xp}}(\sigma_{Fed[\theta']}(\mathcal{F}))$, *where, θ' no longer refers to XML data, and is a placeholder at optimization time for the real predicate having the same filtering effects as θ, with references only to regular dimension levels and constants.*

Example 4.2 $\sigma_{Fed[l_{pop}=1017645163]}(\delta_{l_{pop}}(\mathcal{F}_{TC})) = \delta_{l_{pop}}(\sigma_{Fed[(l_{pop}=1017645163)']}(\mathcal{F}_{TC}))$ where l_{pop} represents *Nation/Nlink/Population*. The predicate $(l_{pop} = 1017645163)'$ becomes a placeholder in the second plan and will be rewritten to *Nation="India"* by inlining at execution time. The two predicates have the same selection effect because only India has exactly a population of 1017645163.

A logical federation generalized projection removes all dimensions that are not present in the parameters, and rolls up the remaining dimensions to the specified levels. Therefore, a decoration operator can be removed if the federation generalized projection above does not reference the decoration data through the level expression.

Rule 5 (Redundant Decoration Below Federation Generalized Projection) *If $l_{xp} \notin \mathcal{L}$, then* $\Pi_{Fed[\mathcal{L}]<F(M)>}(\delta_{l[SEM]/link/xp}(\mathcal{F})) \rightarrow \Pi_{Fed[\mathcal{L}]<F(M)>}(\mathcal{F})$ *holds.*

There are situations where the federation projection operator above a decoration operator references the data of the dimension produced by the decoration operator below through the level expression. In that case, the equivalent plan has a new federation generalized projection below the decoration, which aggregates the cube as much as possible, while still allowing the decoration operator to be applied.

Rule 6 (Pushing Federation Generalized Projection Below Decoration) *If the level expression of the decoration operator is a projection parameter, i.e., $l_{xp} \in \mathcal{L}$, the following holds:* $\Pi_{Fed[\mathcal{L}]<F(M)>}(\delta_{l_{xp}}(\mathcal{F})) \leftrightarrow \Pi_{Fed[\mathcal{L}]<F(M)>}(\delta_{l_{xp}}(\Pi_{Fed[\mathcal{L}']<F(M)>}(\mathcal{F})))$, *where $\mathcal{L}'$ does not contain l_{xp} but still allows $\delta_{l_{xp}}$ to build the decoration dimension.*

Example 4.3 $\Pi_{Fed[Customer, Brand, l_{pop}]<SUM(Quantity)>}(\delta_{l_{pop}}(\mathcal{F}_{TC}))$ is equivalent to $\Pi_{Fed[Customer, Brand, l_{pop}]<SUM(Quantity)>}(\delta_{l_{pop}}(\Pi_{Fed[Customer,Brand,Nation]<SUM(Quantity)>}(\mathcal{F}_{TC})))$, where l_{pop} represents *Nation/Nlink/Population*. The new projection operator also has the levels Customer and Brand. But the level expression is replaced by Nation, which still allows the decoration dimension to be built through Nlink.

5 Performance Study

Here, the query engine is observed w.r.t., 1) the query evaluation performance, 2) the effectiveness of the query optimization, and 3) the feasibility of the federation system.

Query Evaluation Performance To study the behavior of the query engine, four groups of queries of different types and selectivities were evaluated. Groups 1 to 4 have selectivities of 0.01%, 0.1%, 1% and 10%, respectively. Each group has ten types of queries. Queries with larger type numbers tend to return more data than the smaller ones. The line chart in Figure 3 illustrate execution performance of queries with different selectivities. The chart shows that the more the cube is reduced by selection and aggregation, the better the query performs.

Query Optimization Effectiveness The same types of queries as above were evaluated on the query engines with and without optimizations. except that the WHERE clause now refers to external XML data and only has a selectivity of 10%, e.g., WHERE Nation/Nlink/Population IN (45860000, 59128187, 31787647). Figure 4 presents the executions of the initial plans and the optimized plans by the top and bottom lines respectively. The lines indicates optimized plans are executed seven to fifty times faster than the straightforward initial plans. The experiments show the optimization is effective in reducing intermediate data and data-transfer.

Federation Feasibility We compare the performance when the external XML data is in 1) the XML component (federated), 2) in the local, relational temporary component (cached), and 3) physically integrated in the OLAP cube itself (integrated). The same query types are defined as above and evaluated with optimizations. A large(11.4 MB) XML document and and a small (2KB) one were used, both generated from the TPC-H benchmark [12]. As dimensions in OLAP cubes are not typically very large, we believe the

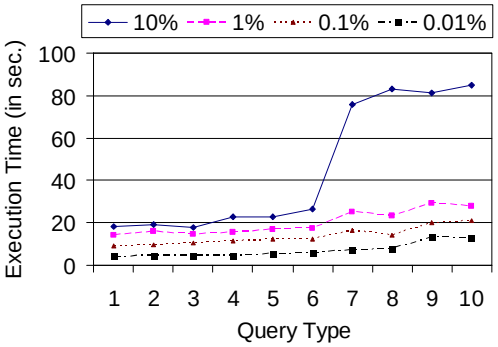

Figure 3: Line chart for 4 selectivities

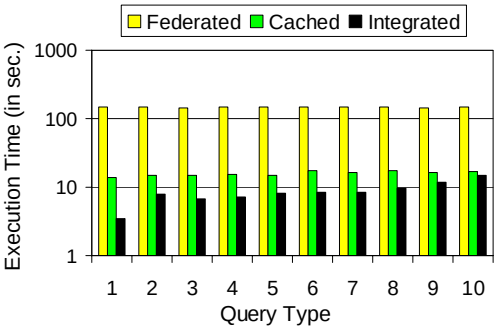

Figure 5: Using 11.4MB XML data

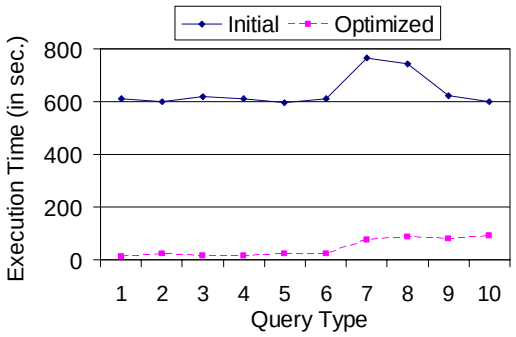

Figure 4: Execution time comparisons

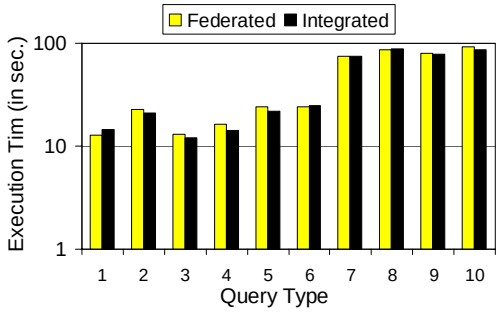

Figure 6: Using 2KB XML data

amounts of XML data used for virtual dimensions are realistic. As Figure 5 indicates, the cost of querying the federation (shown in logarithmic scale), where XML processing takes most of the time (i.e., about 135 seconds), exceeds the cost of querying the physical integration by a factor of ten to twenty. The "Cached" bars stay in between but much closer to the "Integrated." Figure 6 demonstrates comparisons of queries on two other federated/integrated levels. The chart suggests that querying the logical federation with a virtual dimension has almost the same performance as on the physical integration, when the amount of the XML data is small, i.e. a few kilobytes. In summary, the federation can be queried just as if it was a local cube when involving a small amount of XML data. More efficient query performance can be gained by caching the external data locally, which will be the most common case in the applications of OLAP-XML federations.

6 Conclusion

We have presented query optimization techniques specialized for the OLAP-XML federation system and experimental results, which suggest that our approach, unlike physical integration, is a practical solution for integrating fast changing data into OLAP systems. Our future work will focus on enhancing the query engine with more advanced query evaluation techniques and also integrating XML-based measure data with OLAP systems.

Acknowledgements

This work was supported by the Danish Research Council for Technology and Production Sciences under grant no. 26-02-0277.

References

[1] R. Goldman and J. Widom. WSQ/DSQ: A Practical Approach for Combined Querying of Databases and the Web. In *Proc of SIGMOD*, pp. 285–296, 2000.

[2] G. Graefe and W.J.McKenna. The Volcano Optimizer Generator: Extensibility and Efficient Search. In *Proc. of ICDE*, pp. 209–218, 1993.

[3] J. Gu, T. B. Pedersen, and A. Shoshani. OLAP++: Powerful and Easy-to-Use Federations of OLAP and Object Databases. In *Proc of VLDB*, pp. 599–602, 2000.

[4] J. M. Hellerstein, M. Stonebraker, and R. Caccia. Independent, Open Enterprise Data Integration. *IEEE Data Engineering Bulletin*, 22(1):43–49, 2000.

[5] IBM corp. Information integrator. www.ibm.com/software /data/integration. Current as of Oct. 20, 2006.

[6] Chawathe S. et al. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Proc. of IDS of Japan*, pp. 7–18, 1994.

[7] Lahiri T. et al. Ozone: Integrating Structured and Semistructured Data. In *Proc of DBLP*, pp. 297–323, 1999.

[8] Pedersen T. B. et al. Extending OLAP Querying to External Object Databases. In *Proc of CIKM*, pp. 405–413, 2000.

[9] Oracle corp. Gateways. www.oracle.com/gateways. Current as of Oct. 20, 2006.

[10] D. Pedersen, K. Riis, and T. B. Pedersen. XML-extended OLAP Querying. In *Proc of SSDBM*, pp. 195–206, 2002.

[11] R. Ramakrishnan. *Database Management Systems*. McGraw-Hill, 2003.

[12] Transaction Processing Performance Council. TPC-H. www.tpc.org/tpch. Current as of Oct. 20, 2006.

[13] X. Yin and T. B. Pedersen. Algebra-Based Optimization of XML-Extended OLAP Queries. DBTR-17, 2006. www.cs.aau.dk/DBTR. Current as of Oct 20, 2006.

[14] X. Yin and T. B. Pedersen. Evaluating XML-Extended OLAP Queries Based on a Physical Algebra. *Journal of Database Management*, 17(2):85–116, 2006.