A Query Processing Architecture for an XML Data Warehouse

Nuuwee Wiwatwattana #1, H. V. Jagadish #2
#Computer Science and Engineering, University of Michigan
Ann Arbor, Michigan, USA
1nuwee@umich.edu
2jag@eecs.umich.edu

Abstract—Data warehousing accounts for a significant fraction of database use today. As XML becomes ever more popular, more and more XML data finds its way into data warehouse repositories. This paper examines the modeling mismatch between the tree structure of XML data model and the multidimensional model of a typical data warehouse, and proposes an XML warehouse model based on the Multi-Colored Trees (MCT) logical data model that resolves the modeling issue naturally. Furthermore, this data model ameliorates some well-known modeling limitations of the XML data warehouse.

To cope with ad-hoc OLAP queries, we extend bitmap join indices to the XML context. We then tackle the difference between the bit-map and the stack-based structural join processing paradigm popular in XML query processing, permitting both styles of query processing to be used seamlessly in consort to evaluate queries. We demonstrate experimentally the benefit of bitmap join indices for typical queries, and particularly those with low cardinality or high selectivity.

I. INTRODUCTION

Data warehousing stands out as an arena in which XML has not been able to make many inroads, in spite of the many modeling benefits that it offers. One major barrier to XML adoption has been a mismatch between the XML data model and that of a typical data warehouse. A typical warehouse has a very large “fact” table, and multiple smaller tables arranged along dimension hierarchies, in a star or snowflake schema. XML has a tree-structured hierarchical model, which is an excellent match for any one dimension hierarchy, but does not do well at all in representing the large fact table. If we use multiple XML hierarchies, one for each dimension hierarchy, we can actually get excellent representation of dimension information, with flexibility and expressiveness going beyond what current databases provide, and meeting the desiderata for expressive hierarchies. However, in traditional databases, each fact table entry ties together multiple leaves, one from each hierarchy, and the only way to do that in XML is through an inelegant and inefficient reference structure (i.e., ID/IDREF).

To overcome inefficiencies associated with XML for reference structures, our recent work on multi-colored XML (MCT) [1] has proposed a more expressive model for XML data. This MCT model permits multiple hierarchies, in different colors, and allows nodes to be shared among different hierarchies.

Fig. 1. The MCT Warehouse Schema of the TPC-W benchmark schema. Nodes/edges shown repeated per color for clarity. Each color represents one dimension hierarchy. Attributes are suppressed for a clearer representation.

Each fact table entry becomes a multi-colored node in this model, and can be expressed conveniently.

Although MCT addresses the modeling dilemma, we are still faced with the issue of query processing. XML query processing has traditionally performed stack-based computations with node identifiers that are created and ordered in a way that makes ancestor-descendant relationships easy to determine. Data warehouse query processing usually relies on bitmap join indices to be able to compute efficiently the combination of multiple query conditions. These are two very distinct query processing paradigms, and it is not clear how to put them together in a single system. We develop an architecture for query processing in a multi-color XML warehouse, seamlessly integrating both styles of query processing mentioned above. We show experimentally that the integrated architecture is capable of good query processing performance, whereas an alternative architecture using only one of the two query processing techniques does poorly.

II. MODELING XML WAREHOUSE

In common OLAP warehouse applications, users often ask questions about fact attributes according to particular dimension attributes. In such a schema, facts are small multi-colored subtrees, which include measure attributes. These facts are leaves in each of the multiple dimension hierarchies. Each dimension hierarchy is a colored tree. Fig. 1 depicts the XML warehouse schema of the TPC-W benchmark. The order and order_line subtrees are facts. An attribute
such as total in the order subtree is a measure attribute. The country of customer and billing address are dimension attributes, to name a few.

[Dimension Hierarchy]: A dimension hierarchy $D$ is a rooted colored tree $T_d$ with a main color $c_d$. All nodes in $D$ have at least one color, the color $c_i$. Facts (and their attributes) are subtrees in $D$, and have all the colors of the dimensions they are associated with.

[MCT Data Warehouse Schema]: A Data Warehouse Schema is an MCT schema with dimension hierarchies $D_1, D_2$.

The proposed model provides the following notable benefits in warehousing. All of the followings are accomplished while keeping the data normalized.

- Hierarchical organization between all dimension attributes of all dimension hierarchies and fact attributes accommodates query writing and efficient query evaluation via structural joins.
- Fact elements for any given group in any given dimension are clustered together. This increases the performance of any structural joins and navigational operations occurring in any of these dimensions.
- The clustered hierarchy also requires no additional sorting when facts are grouped and aggregated by dimension attributes. As a result, the grouping operations become non-blocking.

III. OLAP QUERY PROCESSING

Decision support queries in the warehouse are usually complex, typically ad-hoc, requiring multiple joins along multiple dimension paths (same-color and cross-color hierarchical joins). Since the number of fact elements is usually very large, to avoid actual joins for faster processing, bitmap join indices are widely used for this class of queries.

A. Bitmap Join Indices

In a relational database system, a join index is defined as an index on one table for a quantity that involves a column value of a different table through a commonly encountered join path [2]. Because of the tree heterogeneity, XML join paths are less rigid between elements. The join paths between the same pair of XML elements can be different. The data portion of the index is represented in a bit vector. One bit corresponds to an element, and a bit vector is a series of bits of the indexed elements.

Even though bitmap join indices can be expected to provide a significant performance gain, leveraging the indices into an MCT/XML context is not straightforward. In a relational database system, mapping between bit positions and rows is effectively done through row identifiers. The entire query processing has to be done on values. Thus, the order of rows in a table is not important in the evaluation. Instead, it is the value itself which is already grouped in bit vectors. In XML and MCT, in contrast, the majority of query processing is done on the tree structures. The main technical challenge is that the order of elements (attributes) in bit vectors has to be globally fixed to be manipulated and compared, but the order of the same element in different colored trees can be different. Therefore, one of the colored orders is chosen as a global order for the bitmap. This order difference affects the evaluation of any queries that construct results for the color not chosen for the global order. In the evaluation plan, we need to convert bit positions into node information (i.e. start key, end key and level) to be able to (structurally) join with each of the elements.

We use two maps to translate between bit positions and XML object identifiers (which in our case are node information): XML Object ID → Bit Position and Bit Position → XML Object ID. For a one-condition query, when the color returned is different from the global bit order, tree nodes are not ordered according to the designated color, but the global bit color. As a result, an expensive sorting on node identifiers has to be applied after the translation. When we have many group-by conditions, each group produces one bit vector, the order of nodes returned is in accordance with the many groupbys (sale orders by country ‘and’ billing address), not one of the group-by conditions. Therefore, if we need to process the returned nodes at a tree level, no matter which dimension we are returning we need to sort the nodes by that dimension. Hence, the key to bridging the difference between the global bit order and tree orders is bit-to-tree translation coupling with node sorting, which can be a bottleneck in the evaluation.

IV. ARCHITECTURE EVALUATION

We implemented the query processing engine discussed in this paper in C++ on the TIMBER native XML database. We experimented with synthetic and real datasets: TPC-W benchmark and molecular interaction database (MiMI) [3].

A. TPC-W Benchmark:

We generated an XML data file as in our MCT warehouse model (Fig. 1) with 5 colors. When loaded in TIMBER, the data size is 1.4 Gbytes including necessary tags and content indices, with 2.6 million elements. There are together five variables used to categorize 108 queries: (1) the number of dimensions (or colors) used in the query (one(1C), two(2C), three(3C), or four(4C)), (2) whether the order of nodes in the results needed to be the SAME or NOTSAME as the bit order, (3) the aggregate operation (NONE, COUNT, or SUM) , (4) the cardinality of the bitmap condition (usually a dimension attribute) (SMALL, or LARGE), and (5) the selectivity of the bitmap condition in the query (LOW, MEDIUM, or HIGH) 1. Fig. 2 presents the percentage of performance gains of a bitmap join plan over a non-bitmap join plan across some of these variables. The minus bar reflects that the non-bitmap join plan performs better.

Unsurprisingly, bitmap join plans almost always win more than 80 percent with small cardinality conditions, because the index with a small cardinality is more clustered, has smaller

1Cardinality is SMALL if the number of conditions indexed is within the order of $10^2$. Otherwise, it is LARGE. The result size is HIGH selective if it is within $10^2$, $10^4$ for MEDIUM, and larger than $10^4$ for LOW.
were taken from a log file and their running time is shown in GeneOntology-Variant. 7 popular non-simple-selection queries (content indices), with over 3 million elements. We only need a reordering. However, at low query selectivity, the advantage of index accesses outshines the reordering glitch. A higher selectivity does not always come with a higher performance gain. Fundamentally, not only bitmap join plans but non-bitmap join plans benefit from higher query selectivity (Fig. 2(a - d)).

Simple queries with one dimension (1C) are already efficient with structural and navigational processing, as MCT allows queries to be solved in a few steps. Thus, bitmap join indices should not be much better in these queries. Yet, queries with MEDIUM or HIGH selectivity for LARGE cardinality can be run faster in bitmap join plans, especially when the number of colors is two (2C; Fig. 2(b)). At these points, bitwise operations on a small or medium result set perform better than color crossings. Comparing SAME and NOTSAME in SMALL cardinality situations (Fig. 2(c) and (d)), the difference is only a little, except in 1C , since the sorting of nodes into a tree order is always necessary.

B. Real World Data (MiMI):

The MiMI data set size is 2.35 GBytes (with tags and content indices), with over 3 million elements. We only extracted the interaction information, molecular information and gene ontology data. This created 3 colors MCT data: two colors for Organism-Molecule-Variant-Interaction (each color for each participant of the interaction) and one color for GeneOntology-Variant. 7 popular non-simple-selection queries were taken from a log file and their running time is shown in Fig. 3.

![Fig. 3. Performance in seconds for MiMI queries](image)

Bitmap join plans perform better in every query. Unsurprisingly, biologists are more concerned with certain genes and/or specific organism, which can be regarded as highly selective and low cardinality search, respectively. Another advantage bitmap join indices provided over structural joins can be seen in Q4, Q5 and Q6, where multiple colors are involved, and thus, in addition to the ever fast bitwise operations, single-color ancestor-descendant joins are inferior to bitmap join plans which provide fewer steps across multiple colors. For instance, we had a bitmap join index on molecular interaction of gene ontology’s ids. It would require 2 color crossings (from gene ontology’s dimension to each interaction participant) without the bitmap.

In summary, bitmap join indices are very efficient for low cardinalities bitmap conditions. Bitmap join index plans can perform well in some cases where the cardinalities are large but the query condition is highly selective as well. The indices are also advantageous in aggregate operations.

V. Conclusions

This paper recognizes the considerable benefits of the Multi-colored XML data model (MCT) when used as an XML warehouse model. The MCT warehouse model comprises dimension hierarchies, fact and measure attributes similar to relational database. The use of bitmap join indices in MCT warehouse model generally improves query time, exhibiting similar performance patterns as in a relational warehouse model. Towards this end, we have proposed a new flexible bitmap index data structure and order-variant access method. We showed that numerous variables altogether dictate the performance of bitmap join indices. To further improve and automate the query processing, a cost-based index selection coindex for XML tree processing is needed to determine whether bitmap join indices should be used.

ACKNOWLEDGMENT

H.V. Jagadish is supported in part by NSF under Grants No. IIS-0219513 and IIS-0438909.

REFERENCES

