DUMAX: A Dual Mode Algebra for XML Queries

Dunren Che and Radiya M. Sojitrawala
Department of Computer Science
Southern Illinois University Carbondale, USA
(01) 618 453 6046
{dche, rsoji}@cs.siu.edu

ABSTRACT
In the paper, we present a new algebra called DUMAX designed for XML and XML queries. An important feature of this algebra is its dual mode, which is introduced to help fuse node-based features and tree-based features (both are essential for XML) and to achieve accelerated execution of XML queries in large XML databases or repositories. We also briefly discuss the potentials of DUMAX for XML query processing and optimization.

Categories and Subject Descriptors
H.2.4 [**Database Management**]: Systems – Query processing, F.4.3 [**MATHEMATICAL LOGIC AND FORMAL LANGUAGES**] Formal Languages – Algebraic language theory

General Terms
Theory.

Keywords
Algebra, XML Algebra, Query Algebra, XML database, Query processing, Query optimization.

1. INTRODUCTION
While the web-published data (in the form of XML data for example) keeps mounting up, advanced scalable techniques are especially needed for efficiently querying XML data sources (typically stored and managed by a DBMS).

The algebraic approach has been proven to be an effective way for query processing and optimization in a variety of database systems, including RDBs and OODBs. A related key issue is to how to design an apposite query algebra for XML. On the one hand, XQuery [4] has been proposed and accepted as a pre-standard language for querying XML data; on the other hand, although numerous algebras have been proposed, no algebra has been commonly accepted as the query algebra for XML data (this is in contrast to the situation of relational databases and queries). In this paper, we report the result of our effort toward developing a proper algebra to facilitate effective XML query optimization and evaluation.

Conference'04, Month 1–2, 2004, City, State, Country.

Because of the semi-structured nature of XML, XML query processing heavily depends on dealing with two kinds of trees: *pattern trees* and *operation trees*. A pattern tree specifies a tree-shaped structure pattern used in an XML query to screen the data sources. As in traditional database query processing, an operation tree forms the basic manipulation space for query optimization. Query optimization typically relies on transforming the operation trees to identify improved (if not optimal) alternative query plans. Previous proposals for XML algebra emphasized on either pattern trees (like TAX [11]) or operation trees (like XAL [9]), but not on both. An ideal algebra for XML queries shall posses the features of both types of trees.

In this paper, we first introduce a new algebra, called DUMAX (a DUal-Mode Algebra for Xml), which is designed to provide integrated support to both types of trees in a single algebraic framework. Based on DUMAX, we then develop algebraic transformation strategies (formatted in the form of query equivalences using DAMAX expressions) to facilitate XML query transformation and optimization.

The remainder of this paper is organized as follows: First (in Section 2), we review related work by drawing a general picture of the algebras proposed for XML. Second (in Section 3), we present our initial design of the DUMAX algebra, mainly by presenting the main operations provided by this algebra. Third (in Section 4), we use DUMAX as s formalism to address the algebraic optimization strategies appropriate for XML queries.

2. REVEW OF RELATED WORK
Algebra has been a very important issue for database query optimization. There appear to be plenty of algebras proposed in the context of XML. Fernandez *et al* proposed an algebra [8] tailored for Quilt. This algebra later on formed the intellectual basis for the W3C working group's algebra document [7], which is centered on providing formal semantics, but not on efficient implementation of XQuery [4]. Other XML algebras were proposed with more or less emphasis on the support for XML query implementation. These algebras can be roughly classified into three categories: (1) extended relational (or object-relational) algebras, e.g., YAT [6], Xtasy [3], XAT [19], etc., (2) node-based algebras, e.g., PAT [1, 17] and XAL [9], and (3) tree-based algebras, e.g., Aqua [18] and TAX [11]. In the following, we review several selected algebras proposed for XML.

The Aqua project algebra [18] focuses on the identification of pattern matches and rewriting in the style of grammar production rules. The YAT algebra [6] is defined as an extension to the familiar relational and object-oriented algebras so that the common relational and object-oriented optimization techniques

can be retained. To fuse the XML data model with relational model, the algebra introduced two border operators: *bind*, to extract XML data to an intermediate structure, and *tree*, to rebuild an XML structure from the intermediate structure. That intermediate structure is named Tab and is essentially a set of tuples for holding the bindings to specified labeled tree nodes. The main object of manipulation in this algebra is nodes (tuples). As pointed out in [11], due to the consequent loss of structure, this scheme very quickly breaks down when complex analyses are required. The Xtasy algebra [3] was proposed with the same spirit, and actually can be considered as a variant of the YAT, but additionally supports direct evaluation of recursive XPath patterns. XAT [19] and several other algebras are all extended relational algebras. In [12] a node-based navigational algebra was proposed. It treats individual nodes as the basic units of manipulation. There are three other well-know algebras, Niagara [10], XAL [9], and TAX [11]. Due to space concern, we are no going to provide more details of these algebras.

Generally, there are two kinds trees related to an XML query — one (or more) pattern tree implied by the query and an operation tree that forms the search space for query optimization. This observation seems to be widely ignored so far. The focus of this writhing is to extend the enriched version [2, 1] of the PAT algebra [17] as adopted in our previous work [2, 1], and incorporate important ingredients from other influential algebras, such as XAL and TAX, to form a new algebra that has more expressive power and can better facilitate XML query processing.

3. DUMAX: THE ALGEBRA

In this section, we present the major operations in the DUMAX algebra. DUMAX operations can work in two different modes: *shallow* mode and *deep* mode. Shallow mode is a natural feature of node-based algebra like XAL [9], and deep mode is an important feature enabled only by tree-based algebra like TAX [11]. In shallow mode, an operation does not see the levels beyond the current node, while in deep mode, an operation perceives the entire tree as referenced, i.e., the details about all the sub-trees underneath the current node. Each operation assumes the default mode. Only when it does not work in the default mode, is the non-default mode explicitly specified in the form of a superscript parameter.

3.1 Data Model

DUMAX is designed for XML, and we assume a data model that deals with collections of node-labeled and ordered trees. In the relational data model, a relation consists of a set of tuples with identical structure. The counterpart in the DUMAX data model is a collection of trees, with structures that satisfy a given DTD/XSD (XML Schema Definition). Trees in the same collection must satisfy the same DTD/XSD.

In the XML, an element can have children, which are either sub-elements or attributes. In our data model, elements are represented as tree nodes, and attributes are always represented as tree leaves (nodes) because attributes do not have children. Elements and attributes are both stored as name-value pairs.

In brief, DUMAX assumes a data model which is collection of labeled and ordered trees. A database is a collection of collections of such trees, and each collection satisfy the same DTD/XSD.

3.2 Symbols and Grammar

We adopt a number of meta-symbols in our presentation and their interpretations are given below:

- '|' – or
- etn – an element-type-name
- @atn – an attribute-name
- [...] – the content in the brackets is optional.
- $\{...\}^+$ – the content in the braces repeats more than one time.
- $\{...\}^n$ – the content in the braces repeats n times for n = 0,1, …

The syntax of algebraic expressions in DUMAX is summarized below and explained in the following subsections.

$E ::= etn \mid (E) \mid I(E) \mid [\,]$ /* access ops */
$\pi_{pl}(E) \mid$ /* projection */
$\sigma_{target,\ predicate,\ modifier}(E) \mid$ /* selection */
$E1 \supset E2 \mid E1 \subset E2 \mid E1 \supseteq E2 \mid E1 \subseteq E2 \mid$ /*containments */
$E1 \bigcup E2 \mid E1 \cap E2 \mid E1 - E2 \mid$ /* set ops */
$\Sigma_{base,\ order}(E) \mid$ /* sorting ops */
$\gamma_{bl,\ ol}(E) \mid$ /* group-by */
$E1 \backsim_c E2 \mid E1\ X_c\ E2$ /* structural & unstructural joins*/
$\mu_{etn}(E) \mid$ /* construction */
$\chi(E) \mid \perp(E) \mid -(E) \mid \&$ /* miscellaneous */
$E1 + E2 \mid E1 - E2 \mid E1 * E2 \mid E1 / E2$ /* arithmetic ops */
$E1 >> E2 \mid E1 << E2$ /* predicates */

DUMAX operations are classified into the following categories: access operations, projects, selections, containments, set operations, ordering, joins, and miscellaneous operations. Due to space limitation, we only present selected operations from each category below.

3.3 Access Operations

All access operations are atomic and they form the basic set of operations in DUMAX.

etn, as a basic access operation, it simply returns all the elements of the represented type from the database.

(E), as a sub-expression, returns the same result as *E*. Its main usage is for composition.

I(E) is an index operation. It uses a specified index to retrieve the elements decided by the expression *E* from the database.

E[x] is the array operation. Typically, it retrieves a particular element from a sequence of elements decided by the expression *E*. For example, *author[2]* retrieves only the second author in the current context. The array operation's subscript can be other meaningful regular expressions, such as *+, *, 2+, n, $*, and *[n1-n2]*, which are interpreted in a similar way as with the pattern match in Perl language.

3.4 Projection

Projection is an operation for discarding unwanted parts (sub-elements and attributes) from its input.

Syntax: $\pi_{pl}(E)$ where $pl ::= \{@atn \mid etn \mid etn^*\}+$

This operation always works in deep mode. It eliminates nodes other than those specified in the projection list *pl*.

3.5 Selection

A selection operation screens the elements in an input collection based on a screening predicate.

Syntax: $\sigma_{target,\ predicate,\ modifier}(E)$

The *target* parameter specifies the targets that the *predicate* parameter is to be applied to. The grammar of target is: *target ::= e | a |**. When it is *e*, the predicate applies to the content of the elements decided by the subexpression *E*; when is *a*, the predicate applies to a named attribute (prefixed by @); when it is *, the predicate recursively applies to the entire trees in the input collection.

The predicate parameter specifies the filtering condition of the selection operation. It can be any meaningful form of predicates. For example, "@year = 1991", which limits the *year* attribute to a specific value, *1991*.

The modifier parameter modifies the implication of the predicate while applied to the target in a similar way as in Perl pattern match.

3.6 Containment operations

Containment is a key operation for XML and XML queries. We differentiate several versions of this operation.

$E_a \supset E_d$ describes an ancestor-descendant relationship that must he held. The operation returns the ancestor elements only. Accordingly, we have $E_d \subset E_a$, which returns descendants only. $\supseteq$ and $\subseteq$ may be used only for parent-child relationships.

3.7 Set Operations

DUMAX adopts the three standard set operations: union ($\cup$), intersection ($\cap$), difference (–).

3.8 Sorting

Sorting is an important operation for database access. In DUMAX, it is denoted by $\Sigma_{base,order}(E)$, which has a sorting base and an order parameter.

3.9 Grouping

DUMAX provides its group-by operation in the same flavor as in relation databases: $\gamma_{bl,\ ol}(E)$, which contains a grouping-base list parameter *bl* and an output list parameter *ol*.

3.10 Joins

DUMAX provides two types of joins: structural joins and nonstructural joins via a single generic operation $\bowtie_c$.

Structural Join ($E_i \bowtie_c E_j$): When the join condition *c* takes the form of either $\supset$, $\subset$, $\supseteq$, or $\subseteq$, the operation is interpreted as a structural join and the structural relationship must be satisfied.

Nonstructural Join ($E_i\ X_c\ E_j$): When the join condition *c* specifies a non-structural relationship, the operation is interpreted as a nonstructural join, which can be used to accomplish any form of joins, including the joins in relational databases.

3.11 Construction

With XML queries, there is a strong desire for constructing new of XML elements using the obtained query results as materials. In DUMAX, the construction operation takes the form: $\mu_{etn}(E)$. This operation wraps around the output of the expression *E* using the tag name indicated by the parameter *etn.*

3.12 Miscellaneous Operations

There are miscellaneous operation such as unorder **(χ), f**latten **(⊥)**, the unary – (for negation), and a variety of predicates: >> (follows), << (precedes), =, !=,<, <=, >, >=, AND, OR, and NOT.

3.13 Aggregate Functions

The common aggregation operations used in relational databases, *count*, *max*, *min*, *average* are all supported in DUMAX as well.

3.14 Example

In the following, we select a few queries from the XMark benchmark [91] (which assumes an online auction database) to illustrate DUMAX as a powerful algebraic query language.

Q4.1 Return the name of the item with ID 'item20748' registered in North America.

$name \subset ((\sigma_{natural\_order\ =\ 'item20748'}(item)) \subset namerica)$

Q4.2 Return the initial increases of all open auctions.

$increase \subset (bidder[1] \subset open\_auction)$

Q4.3 How many sold items cost more than 40?

$COUNT\ (closed\_auction \supset (\sigma_{>40}\ (price)))$

Q4.4 List all persons according to their interest.

$\gamma_{[(interest,\ @category)],\ person}\ (person)$

Q4.5 For each person, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.

$\gamma_{[(person,\ name)],\ COUNT(*)}(person\ X_{profile@income\ >\ 5000*initial}\ (open\_auction)$

Q4.6 Print the keywords in emphasis in annotations of closed auctions.

$keyword \subset (emph \subset (annotation \subseteq closed\_auction))$

Q4.7 Confer Q15. Return the IDs of the sellers of those auctions that have one or more keywords in emphasis.

$\pi_{@person}\ (\sigma_{(seller//emph/keyword)}\ (seller \subset closed\_auction)$

Q4.8 Which persons don't have a homepage?

$- (person \supseteq homepage)$

Q4.9 Give an alphabetically ordered list of all items along with their location.

$\Sigma_{asc}(item \bowtie_{\subset} region)$

4. XML Query Optimization

In an algebraic optimization approach, XML queries are represented as algebraic expressions, and transformations are then performed on the query expressions according to algebraic equivalences. The best alternative query expression is ultimately decided from usually a very large pool of candidates enumerated based on cost analysis or optimization heuristics. Candidate enumeration relies on equivalent query transformation. Our work emphasizes heuristic-based optimization strategies due to the multiple sources of heuristic knowledge as recognized in [1] that can be used to quickly reach a "sufficiently good" alternative query plan without resort to exhaustive plan enumeration. In the following we briefly outlook DUMAX's possible application in XML query optimization.

4.1 Optimization through Mode Switching

Maintaining the deep mode for DUMAX operations during query evaluation is obviously costly. So, switching from deep to shallow mode is an important aspect of query optimization. In the course of query evaluation, mode analysis must be performed in order to identify the opportunities so that beneficial mode switching can be made. A general heuristics with regard to mode-centered optimization is "push down shallow mode operations to the bottom and pull up deep mode operations to the top in the query's operation tree" (in this way data tree materialization is postponed to the last). We are currently developing a runtime evaluation optimization procedure that uses the dual mode feature offered by DUMAX to obtain acceleration of XML query execution.

4.2 Query Equivalences

As a rather expressive algebra, DUMAX is capable to accommodate the common equivalences that can be used for query optimization, including De Morgan's Laws, projection decomposition, selection cascading, and the communativity law of various operations. Due to space limitation, we refer interested readers to a previous paper [1], where we presented a large set of equivalences and deterministic transformation rules based on a rather simpler algebra PAT [17]. Our work is continuing – we are adapting the equivalences and transformations rules to DUMAX.

5. SUMMARY

In the paper, we presented a newly designed algebra called DUMAX for XML queries. An important feature of this algebra is its dual mode, which is introduced to fuse node-based features and tree-based features and can be used to achieve runtime evaluation optimization for XML queries, in addition to query plan optimization. We hope our work will inspire more interests in the research community toward a more suitable XML algebra.

6. REFERENCES

[1] D. Che, K. Aberer, and M. T. Özsu. Query Optimization in XML Structured-Document Databases. *VLDB Journal*, 15(3): 263-289, September 2006.

[2] D. Che. Efficiently Processing XML Queries with Support for Negated Containments. *International Journal of Computer & Information Science*, 6(2): 109-120, June 2005

[3] C. Sartiani, A. Albano. Yet Another Query Algebra for XML Data. In *Proc. of IDEAS 2002*, Edmonton, Canada, July 2002.

[4] S. Boag, D. Chamberlin, M. F. Fernndez, D. Florescu, J. Robie, J. Simon. XQuery 1.0: An XML Query Language. *W3C Working Draft 04*, April 2005 (http://www.w3.org/TR/xquery).

[5] 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 *Proc. of VLDB'03*, Berlin, Germany, September 2003.

[6] V. Christophides, S. Cluet, and J. Simeon. On wrapping query languages and efficient XML integration. In *Proc. of SIGMOD*, pages 141-152, May 2000.

[7] D. Draper, P. Fankhauser, M. Fernndez, et al. XQuery 1.0 and XPath 2.0 Formal Semantics. *W3C Working Draft 3*, June 2005 (http://www.w3.org/TR/query-semantics/).

[8] M. Fernandez, J. Simeon, P. Wadler. An Algebra for XML Query. In *Proc. of FST TCS*, Delhi, December 2000.

[9] F. Frasincar, G.-J. Houben, C. Pau. XAL: an algebra for XML query optimization. In *Proc. of the13th Australasian Conference on Database technologies*, January 2002.

[10] L. Galanis, E. Viglas, D.J. DeWitt, J.F. Naughton and D. Maier. Following the Paths of XML Data: An Algebraic Framework for XML Query Evaluation. 2001 (http://www.cs.wisc.edu/niagara/papers/algebra.pdf).

[11] H. V. Jagadish, L. V. S. Lakshmanan, D. Srivastava and K. Thompson. TAX: A Tree Algebra for XML. In *Proc. of DBPL Conf.*, Rome, Italy, September 2001.

[12] B. Ludascher, Y. Papakonstantinou, and P. Velikhov. Navigation-driven evaluation of virtual mediated views. In *Proc. of EDBT*, pages 150-165, 2000.

[13] University of Michigan. The TIMBER system (http://www.eecs.umich.edu/db/timber/).

[14] S. Paparizos, S. Al-Khalifa, H.V.Jagadish, A. Nierman and Y. Wu. A Physical Algebra for XML. *Technical report* (http://www.eecs.umich.edu/db/timber/).

[15] S. Paparizos, Y. Wu, V.S. Lakshmanan and H.V. Jagadish. Tree Logical Classes for Efficient Evaluation of XQuery. In *Proc. of SIGMOD Conf.*, Paris, France, June 2004.

[16] S. Paparizos and H.V. Jagadish. Pattern tree algebras: sets or sequences? In *Proc. of VLDB Conf.*, Trondheim, Norway, September. 2005.

[17] A. Salminen and F. W. Tompa. PAT Expressions: an Algebra for Text Search. *Acta Linguistica Hungarica*. 41(1): 277-306, 1994.

[18] B. Subramanian, T. W. Leung, S. L. Vandenberg, S. B. Zdonik. The AQUA approach to Querying Lists and Trees in Object-Oriented Databases". In *Proc. of ICDE*, 1995.

[19] X. Zhang, B. Pielech, E. A. Rundesnteiner. Honey, I shrunk the XQuery!: an XML algebra optimization approach. In *Proc. of the 4th Intl. Workshop on Web Information and Data Management,* pp15-22. McLean, Virginia, USA, 2002.