Submission for SIGMOD'2002 Demonstration

Rainbow: Mapping-Driven XQuery Processing System *

Xin Zhang, Mukesh Mulchandani, Steffen Christ, Brian Murphy, and Elke A. Rundensteiner
Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609
Tel.: (508) 831–5815, Fax: (508) 831–5776
{xinz, mukesh, stchrist, rudie, rundenst}@cs.wpi.edu

1 Introduction

Touted as the ASCII format of the future, the Extensible Markup Language (XML) [3] has recently been adopted as markup language for information modeling and exchange in many web-based industries.

Commercial relational database products embody years of research and development in areas as diverse as modeling, storage, retrieval, update, indexing, and transaction processing, to just name a few. With more and more data being captured in XML formats, it is an obvious next step to extend relational systems to accommodate XML data. Such an approach may succeed in taking advantage of the power of relational database technology and the wealth of experience in optimizing and using the technology.

There are however a number of technical issues to overcome. These include the data model mapping strategies, the transformation of XML queries into SQL queries, XML order handling, XML update propagation, and XML query optimization. They all are affected by the mapping strategy chosen to translate the more flexible XML model into the rigid relational structure, with many strategies being possible.

2 Rainbow System

We have developed an XML repository management system, called Rainbow, designed to solve the issues mentioned above based on a flexible XML mapping strategy. As shown in Figure 1, the Rainbow system is composed of three sub-systems, e.g., a loading manager, a mapping manager, and an XML query engine build on top of a relational database engine.

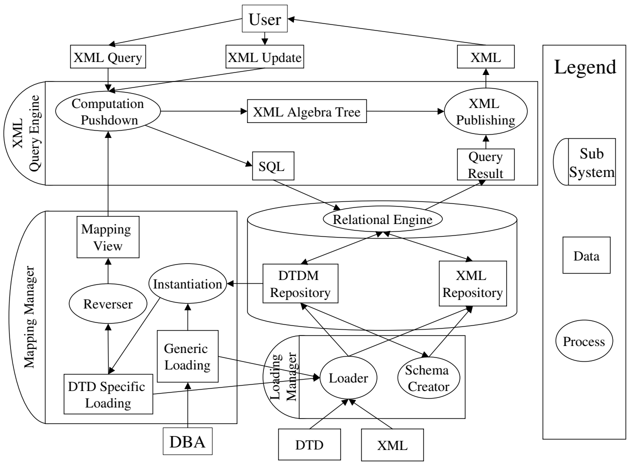

Figure 1: Architecture of the Rainbow System.

Rainbow will first load XML and DTD into the relational database by the loading query provided by the loading manager. Loading queries are expressed in XQuery [8]. Then the mapping manager will provide a mapping view query generated from the loading query. The XML query engine can accept an XML query or an XML update request from the user in an extended XQuery syntax. It then combines the user's request with the mapping view, pushes down the computation to the underlying relational database, and returns the XML results back to the user.

As illustrating examples, Figure 2 represents a book list document (Figure 2.1) with its DTD (Figure 2.2), which has a list of books with their authors. The XQuery (Figure 2.3) is querying the second author of book whose title is "Texas Holdem". The result is shown Figure 2.4.

*This work was supported in part by several grants from NSF, namely, the NSF NYI grant IIS-979624, the NSF CISE Instrumentation grant IRIS 97-29878, and the NSF grant IIS-9988776. Mr. Zhang would like to thank Verizon Inc. for partial support.

1. XML Document

```
<BOOKLIST>
    <BOOK TITLE="Texas Holdem">
        <AUTHOR>David Sklansky</AUTHOR>
        <AUTHOR>Straight Flush</AUTHOR>
    </BOOK>
    <BOOK TITLE="The Tale of Six Cities">
        <AUTHOR>Charles Dickens</AUTHOR>
        <AUTHOR>Joe Smith</AUTHOR>
    </BOOK>
</BOOKLIST>
```

2. DTD

```
<!ELEMENT BOOKLIST (BOOK+)>
<!ELEMENT BOOK (AUTHOR+)>
<!ATTLIST BOOK
            TITLE CDATA #REQUIRED>
<!ELEMENT AUTHOR #PCDATA>
```

3. XQuery

```
<RESULT>
FOR $book in document("bib.xml")/BOOK
WHERE $book/TITLE="Texas Holdem"
RETURN $book/AUTHOR[2]
</RESULT>
```

4. XQuery Result

```
<RESULT>
    <AUTHOR>Straight Flush</AUTHOR>
</RESULT>
```

Figure 2: XQuery Illustrating Example.

2.1 Management of Mapping Queries

The Rainbow system supports the flexible mapping approach of mapping XML data into a flat relational schema, including inlining, shredding, joining, universal tables, and the necessity of order handling of attributes and elements.

The selection of the appropriate mapping depends on the structure of the XML data and the expected type of workload, That is, we have various loading strategies from the literature, including typically recursive XQuery that utilize the XML schema of the original XML data for schema checks.

This procedure has two major advantages. First, Rainbow can flexibly switch between different mapping strategies by simply exchanging the loading XQuery expression. Second, by using recursive expressions it is possible to create loading queries that pinpoint a particular mapping strategy, but are general enough to work for all kinds of DTD. Nevertheless, with the knowledge of the XML Schema of the XML document to be loaded, the Rainbow system can construct DTD-specific, non-recursive XQueries at a finer granularity that allow Rainbow for later tunings.

For the purpose of later query and update processing against the relational database, Rainbow generates a reversed mapping query from the selected loading query that reconstructs virtual XML document views from the relational data.

2.2 Query Processing in Rainbow

For basic query processing, Rainbow employs a strategy similar to XPERANTO [4]. In particular, its XML Algebra Tree (XAT) is inspired by both the algebras proposed by XPERANTO [4] and NIAGARA [5]. Figure 3 shows the XAT tree for the XQuery in Figure 2.

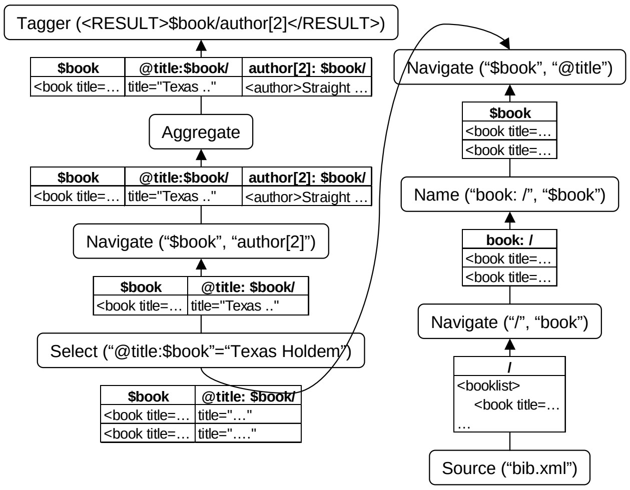

Figure 3: XML Algebra Tree Example.

Rainbow employs SQL pushdown strategies based on algebra equivalence rules to extract maximal SQL expressions from the incoming XQuery, while the query executor implements the remaining tagger nodes in memory.

2.3 XML Update Propagation

Similar to [7, 9, 6] we extend the XQuery language specifications [8] by revising the *returnClause* grammar to incorporate the update statements. We have four update primitives namely *Create*, *Modify*, *Move* and *Delete*. Shown below is an example of the XQuery statement to modify the contents of the bib.xml document.

FOR $book **IN** document("bib.xml")/book[/author/text()= 'Joe Smith']
MODIFY $book/@title **WITH** 'My Title'.

Further, we extend the XAT to handle the XML updates. For example, the **Modify**(*context*, *attribute*, *arithmeticExpr*) operator will change the value of the *attribute* of the element found at the *context* to value given by *arithmeticExpr*.

Since the XQuery mappings from the XML views to the relational data we consider are object preserving, the

XML views are found to be updatable. Based on the mapping query, update statements are pushed down to SQL updates. For the XQuery Modify statement shown earlier, the translated SQL statement will be:

```
UPDATE booklist.book SET title = 'My Title'
WHERE author = 'Joe Smith'.
```

Propagation of updates into all appropriate places in the mapped relational data is supported, including consistency checking to assure correction after each update.

2.4 Order Handling in XML Query Processing

While XML and XQuery are order sensitive, relational databases are not. Unfortunately, order handling has frequently been ignored to date [4, 5, 7]. Rainbow, however, now provides a comprehensive solution. We note that our loading queries must be order sensitive, otherwise user queries that are order sensitive cannot be executed appropriately. For these queries, we have developed a general scheme to add or remove order handling for a data loading query.

Once order is captured, Rainbow addresses all relevant issues including order-aware algebra operators, order sensitive SQL pushdown, and order sensitive SQL execution. With Oracle, our following example query would retrieve the 2^{nd} tuple of the BOOK table:

```
SELECT * FROM BOOK
WHERE rowid = ( SELECT rowid
        FROM ( SELECT rowid, rownum FROM BOOK ORDER BY position)
        WHERE rownum =2)
```

3 Demonstration

Rainbow is implemented with JDK1.2, JDBC, and Oracle *8i*. The system runs on both Windows NT and Linux. The system demonstration will be based on real data sets, e.g., the plays of Shakespeare [2] and the Microsoft SkyServer [1].

- At least three different loading strategies will be exercised. For each strategy, we will then show the following tradeoffs: order sensitive vs. order insensitive, and generic loading vs. DTD specific loading.
- The strategy for reversing a loading query into a specific mapping view will be shown.
- Computation pushdown of different query plans will be shown using a graphical tree viewer.
- Update propagation showing the original and updated XML documents as well as the resulting SQL statements and consistency checks will be shown.
- Order-sensitive queries and the trade-offs in managing order in these mapping queries will be shown.

References

[1] Skyserver sql search. http://skyserver.final.gov/en/tools/search/sql.asp.

[2] J. Bosak. Shakespeare 2.00. http://metalab.unc.edu/bosak/xml/eg/shaks200.zip.

[3] E. T. Bray, J. Paoli, and C. Sperberg-McQueen. Extensible Markup Language (XML), 1997. http://www.w3.org/TR/PR-xml-971208.

[4] M. J. Carey, J. Kiernan, J. Shanmugasundaram, E. J. Shekita, and S. N. Subramanian. XPERANTO: Middleware for Publishing Object-Relational Data as XML Documents. In *The VLDB Journal*, pages 646–648, 2000.

[5] 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. http://www.cs.wisc.edu/niagara/papers/algebra.pdf, 2001.

[6] H. Su, D. Kramer, L. Chen, K. T. Claypool, and E. A. Rundensteiner. XEM: Managing the Evolution of XML Documents. In *IEEE RIDE–DM*, pages 103–110, April 2001.

[7] I. Tatarinov, Z. G. Ives, A. Y. Halevy, and D. S. Weld. Updating xml. In *SIGMOD*, pages 413–424, 2001.

[8] W3C. XQuery: A Query Language for XML. http://www.w3.org/TR/xquery/, February 2001.

[9] X. Zhang, G. Mitchell, W.-C. Lee, and E. A. Rundensteiner. Clock: Synchronizing Internal Relational Storage with External XML Documents. In *RIDE-DM*, pages 111–118, April 2001.