XParent: An Efficient RDBMS-Based XML Database System*

Haifeng Jiang Hongjun Lu Wei Wang
The Hong Kong University of
Science and Technology,
Hong Kong, China
{jianghf,luhj,fervvac}@cs.ust.hk

Jeffrey Xu Yu
The Chinese University of Hong Kong,
Hong Kong, China
yu@se.cuhk.edu.hk

1. Introduction

The Extensible Markup Language (XML) is an emerging standard for data representation and exchange on the Internet. In order to facilitate the task of querying XML documents, efficient storage models for storing XML documents in database systems were studied. There are basically three alternatives: storing XML data in repositories designed for semistructured data [7, 9], in object-oriented databases [1], and in relational systems [3, 4, 10, 11]. A number of systems, such as Lore [7], XML_GL and Tamino [9] also provide query interfaces for users.

This paper will present, XParent, an XML document management system built on top of RDBMS. In comparison to other systems, XParent has its unique features as follows:

- XML data is stored in relational tables according to the **XParent mapping schema** [6], an efficient, model-mapping-based approach without assistance of DTD.
- The visual query interface of XParent provides both expressive power for professionals and user friendliness for naive users.
- We propose a flexible multi-level query translation scheme for model-mapping-based approaches. With such a scheme, it becomes possible to develop a generic XML application that supports multiple XML query languages and mapping schemas.

The rest of this paper is organized as follows. In section 2, the XParent system is introduced. Section 3 shows experimental results and section 4 concludes the paper.

2. XParent: the system

Figure 1 outlines the architecture of XParent. In addition to the underlying RDBMS, there are three main components: *Data Loader*, *Query Interface* and *Query Translator*. The Data Loader takes as input an XML document, parses it and stores it into relational tables according to the XParent schema. The Query Interface provides both visual and command line query facilities. The current implementation supports XPath [2]. With the visual query facility, users can view the structures of stored XML documents, form queries by clicking the relevant data elements and entering conditions. Advanced users can also input XPath queries directly. The Query Translator translates an XPath query into a corresponding SQL query ready to submit to the underlying RDBMS. The returned results are constructed and delivered to the user through the query interface.

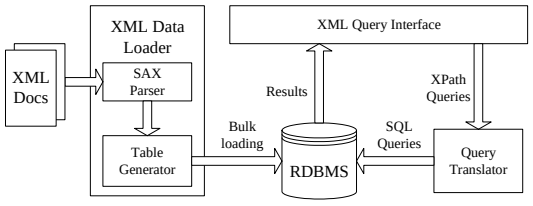

Figure 1. XParent: The system architecture

2.1. XParent schema

XParent adopts the data model of XPath to represent XML documents [2], which models a document as an ordered tree. XParent schema maps such an ordered tree into four relational tables as follows:

```
LabelPath(ID, Len, Path)
DataPath(Pid, Cid)
Element(PathID, Did, Ordinal)
Data(PathID, Did, Ordinal, Value)
```

The table `LabelPath` stores label-paths, each of which is identified with a unique `ID`. Pairs of node identifiers are stored in the table `DataPath`. In `Element` and `Data`

*This work is partially supported by the Research Grant Council of the Hong Kong Special Administrative Region, China (grant AOE 98/99.EG01 and HKUST6060/00E).

tables, `PathID` is a foreign key to the `ID` in table `LabelPath`. The `Did` (for data-path id) is a node identifier which also serves as a unique data-path identifier for the data-path ended at the node itself. Details of XParent mapping schema can be found in the full paper [6].

2.2. XParent query mapping

XPath queries are translated into corresponding SQL queries that conform to the XParent mapping schema. We implemented the query translator with a multi-level architecture as shown in Figure 2.

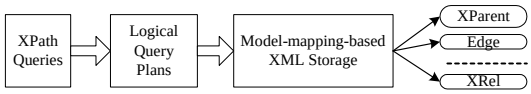

Figure 2. Multi-level query translation

An XPath query is first represented with a logical query plan, which mainly consists of a series of basic operators. SQL queries are generated by resolving these operators based on different mapping schemas. The unique feature of our approach is that the logical plan is independent from the underlying XML to relational mapping schema. Details of the query translation can be found in [5].

3. A performance study

The design objective of XParent is to provide efficient software that can use commercially available RDBMS to manage XML documents. Comprehensive experiments were conducted to study the performance of XParent, in comparison with other approaches. One observation is that RDBMS-based approaches can outperform special-purpose XML repositories such as Lore and Tamino. While among those RDBMS-bases approaches, XParent performs significantly better than other model-mapping-based approaches such as Edge [4] and XRel [11].

All experiments were conducted on top of Microsoft SQL Server 7.0 on a 450MHz PC with 256M RAM, 30G hard disk.

Figure 3 presents some sample results of experiments with XParent and XRel using the Benchmark database [8] with factor 0.4 (BENCH0.4). It can be seen that, XParent outperforms XRel for most of the queries. In certain cases, such as QB2, QB8, QB17 and QB19, XParent can be faster than XRel up to 15 times (QB2), for instance.

4. Conclusions

This paper presents, XParent, an XML document management system built on top of RDBMS. It is based on an efficient, model-mapping-based approach that uses a fixed database schema to store any XML documents without assistance of DTD. The visual query interface of XParent provides both expressive power for professionals and user friendliness for naive users. The proposed multi-level query translation scheme makes it possible to develop a generic XML application that supports multiple XML query languages and mapping schemas.

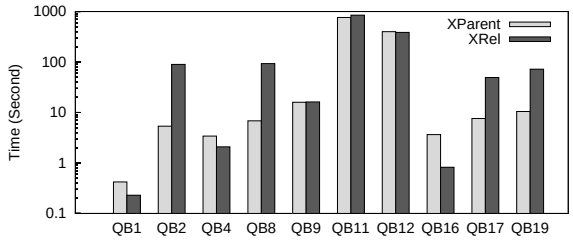

Figure 3. Total Elapsed Time: XParent vs. XRel using BENCH0.4

References

[1] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *Proceedings of the 20th ACM SIGMOD International Conference on Management of Data*, 1994.

[2] J. Clark and S. DeRose. XML path language (XPath). In *W3C Recommendation*, 1999.

[3] A. Deutsch, M. F. Fernandez, and D. Suciu. Storing semistructured data with STORED. In *Proc. of SIGMOD*, 1999.

[4] D. Florescu and D. Kossmann. A performance evaluation of alternative mapping schemes for storing xml data in a relational database. Technical report, INRIA, 1999.

[5] H. Jiang, H. Lu, W. Wang, and J. X. Yu. Mapping-independent XML query processing: From XPath to SQL (submitted for publication). 2002.

[6] H. Jiang, H. Lu, W. Wang, and J. X. Yu. Path materialization revisited: An efficient storage model for XML data. In *Proc. of ADC*, 2002.

[7] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3), 1997.

[8] A. Schmidt, F. Waas, M. Kersten, D. Florescu, L. Manolescu, M. J. Carey, and R. Busse. The XML benchmark project. Technical report, CWI, 2001.

[9] H. Schöning. Tamino - a DBMS designed for XML. In *Prof. of ICDE*, 2001.

[10] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *Proc. of VLDB*, 1999.

[11] M. Yoshikawa, oshiyuki Amagasa, T. Shimura, and S. Uemura. XRel: A path-based approach to storage and retrieval of XML documents using relational databases. *ACM TOIT*, 1(1), 2001.