CARDINALITY-AWARE PURELY RELATIONAL XQUERY PROCESSOR

Sherif Sakr, University of New South Wales, Australia

ABSTRACT

Recently, the use of eXtensible Markup Language (XML) continues to grow in popularity, large repositories of XML documents are going to emerge, and users are likely to pose increasingly more complex queries on these data sets. In 2001, XQuery is decided by the World Wide Web Consortium (W3C) as the standard XML query language. In this article, we describe the design and implementation of an efficient and scalable purely relational XQuery processor which translates expressions of the XQuery language into their equivalent SQL evaluation scripts. The experiments of this article demonstrated the efficiency and scalability of our purely relational approach in comparison to the native XML/XQuery functionality supported by conventional RDBMSs and has shown that our purely relational approach for implementing XQuery processor deserves to be pursued further. [Article copies are available for purchase from InfoSci-on-Demand.com]

Keywords: Relational Database; XML; XPath; XQuery Processor

INTRODUCTION

The eXtensible Markup Language (XML) (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2006) was introduced by the end of the 1990s in order to create a standard data-format for the World Wide Web which could be easily handled by computers as well as by humans. In recent years, XML has found practical application in numerous domains including data interchange, streaming data, and data storage. The semistructured nature of XML allows data to be represented in a considerably more flexible nature than in the traditional relational paradigm. However, the tree-based data model underlying XML poses many challenges, especially with regard to the problem of performing efficient query evaluations.

As XML continues to grow in popularity, large repositories of XML documents are going to emerge, and users are likely to pose increasingly more complex queries on these data sets. Consequently, there is a great demand for efficient XML data management systems for managing complex queries over large volumes of the XML data. In 2001, XQuery was decided by the World Wide Web Consortium (W3C) as the standard XML query language (Boag, Chamberlin, Fernandez, Florescu, Robie, & Sim’eon, 2006).

XQuery is based on a hierarchical and ordered document model which supports a wide
variety of constructs and use cases. The language addresses a wide range of requirements, thus incorporating a rich set of features.

The work of this article was developed within the Pathfinder project (Pathfinder, 2003). The aim of the Pathfinder project is to implement XQuery as a query language that can be used to query XML data stored on relational database systems. The architecture of Pathfinder is designed in a front-end/back-end fashion. Pathfinder receives an XQuery expression, which is parsed, normalized, and translated into XQuery Core. The Core expression is then simplified, type checked optimized, and translated into an intermediate algebraic plan. Initially, Pathfinder used the MonetDB main memory RDBMS as its target back-end. In this development branch (Boncz, Gust, van Keulen, Manegold, Rittinger, & Teubner, 2006), the Pathfinder intermediate algebraic plan is translated into Monet Interpreter Language, or MIL, code (Boncz & Kersten, 1999) which is then executed by the kernel of MonetDB. The MIL code generated by the Pathfinder compiler relies on some extensions added to the MonetDB back-end such as the staircase join algorithm (Grust, van Keulen, & Teubner, 2003) which is designed as an efficient algorithm to evaluate XPath expressions. Although the approach of Pathfinder/MonetDB XQuery processor has been shown to be highly efficient and scalable, it is very tightly bound to the Monet DBMS and thus can not be used with any other relational back end. Another disadvantage of using MonetDB is that it requires huge main memory sizes to store large XML documents. The limitation of this approach was the main motivation behind our purely relational approach for implementing XQuery processor described in this article.

In this article, we describe the design and implementation of an efficient and scalable purely relational XQuery processor which translates expressions of the XQuery language into their equivalent SQL evaluation scripts. The proposed XQuery processor is enhanced with an accurate algebraic-based cost model (Sakr, 2007, 2008a; Teubner, Grust, Maneth, & Sakr, 2008) which facilitates the processor’s ability to generate enhanced cardinality aware SQL translation scripts. Figure 1 illustrates the different alternative back-ends for the Pathfinder XQuery compiler. In particular, the main contribution of the work of this article is that it describes the design and implementation of an efficient and scalable purely relational XQuery processor. The proposed relational XQuery processor stores the source XML documents in a relational repository using a tree aware relational encoding scheme and translates the XQuery expressions into SQL evaluation scripts. The main features of the proposed XQuery processor are:

- It supports an almost complete dialect of the XQuery language.
- It can reside on any relational database system and exploits its well known matured query optimization techniques as well as its efficient and scalable query processing techniques.
- It can target any RDBMS which supports the standard SQL:1999 language interface with no need for the relational database back-end to support the SQL/XML standard or to provide an XML column type of any kind.
- The relational database kernel remains untainted as there is no need for additional query processing operators or special structural join algorithms to be injected.
- It exhibits good performance characteristics when run against high-volume XML data as well as complex XQuery expressions.

The rest of this article is organized as follows. In the following section, we give an overview of the XPath accelerator relational encoding scheme for the XML documents as the basis of our proposed XQuery processor. We then give an overview of the loop-lifting compilation technique which translates XQuery expressions into their equivalent intermediate relational algebraic plans and describe the design and the implementation of the cardinality-aware and purely relational XQuery engine which translates the intermediate Pathfinder algebraic plans into their equivalent SQL evalu-
Related Content

Intension Mining
[www.irma-international.org/chapter/intension-mining/11163/](www.irma-international.org/chapter/intension-mining/11163/)

Service Composition and Interaction in a SOC Middleware Supporting Separation of Concerns with Flows and Views
[www.irma-international.org/article/service-composition-interaction-soc-middleware/52992/](www.irma-international.org/article/service-composition-interaction-soc-middleware/52992/)

The Expert’s Opinion
[www.irma-international.org/article/expert-opinion/51100/](www.irma-international.org/article/expert-opinion/51100/)

Physical Modeling of Data Warehouses Using UML Component and Deployment Diagrams: Design and Implementation Issues
[www.irma-international.org/article/physical-modeling-data-warehouses-using/3351/](www.irma-international.org/article/physical-modeling-data-warehouses-using/3351/)

MECP: A Memory Efficient Real Time Commit Protocol
[www.irma-international.org/chapter/mecp-memory-efficient-real-time/20760/](www.irma-international.org/chapter/mecp-memory-efficient-real-time/20760/)