An Interpreter Approach for Exporting Relational Data into XML Documents with Structured Export Markup Language

Joseph Fong, City University of Hong Kong, Hong Kong
Herbert Shiu, City University of Hong Kong, Hong Kong

ABSTRACT

Almost all enterprises use relational databases to handle real-time business operations and most need to generate various XML documents for data exchanges internally among various departments and externally with business partners. Exporting data in a relational database to an XML document can be considered a data conversion process. Based on the four approaches for data conversion: Customized program, Interpretive transformer, Translator generator, and Logical level translation, this paper proposes a new interpretive approach using Structured Export Markup Language (SEML) interpreter for converting relational data into XML documents. The frameworks and languages proposed by other researchers are neither generic nor able to generate arbitrary XML documents. Therefore, SEML interpreter is a simple, user-friendly, and complete solution with a new mark-up language — SEML — for data conversion. The solution can be used as a generic tool for extracting, transforming, and loading (ETL) purposes. In other words, the SEML interpreter is a solution for relational databases similar to what X-Query is for XML databases.

Keywords: Data Conversion, Interpreter Transformer, Relational Database, Structured Export Markup Language, XML Document

1. INTRODUCTION

While a relational database is the major facility for storing operational data, more and more data are handled in XML format that needs to be composed according to particular XML schema definitions. The following presents the general problems that software developers are facing while composing XML data based on data stored in a relational database.

No Common Facilities for Exporting Desired XML Documents from RDB into XML Databases

Even though XML databases are becoming more popular because XML documents can be retrieved quickly from the XML database servers with query languages such as XQuery
(Boag et al., 2007), a common scenario is that XML documents are generated by exporting real time data from a relational database whose data is constantly changing through all the business operations.

Software Developers are not Knowledgeable Enough

Software developers are usually required to write dedicated software applications to use (a) common input/output libraries or (b) XML specific libraries such as DOM (Le Hégaret 2005) and SAX (Megginson, 2004) to construct the XML documents (Fong, Wong, & Cheng, 2003). The former approach is more error prone, because the software developers must understand the XML specifications thoroughly. The latter approach constructs the corresponding DOM tree data structure in the computer memory before the document is serialized (stored). However, it is resource-demanding at run-time, especially for constructing very large XML documents.

Tradeoff Between Data Concurrency and Query Optimization

Nonetheless, there are issues in concurrency and query optimization. In a typical programmatic approach, software developers mostly have to connect to the underlying relational databases and send SQL statements repeatedly for the necessary data to construct various parts of the XML document regardless of whether they are using common input/output libraries, XML related libraries, or DOM/SAX. If the XML document construction program is the only client that manipulates the relational database server, such an approach works without any problem. However, if the relational database is updated by other software applications at the same time, the query statements sent by the XML document construction process must be transaction protected such that all data obtained from the relational database are of the same timestamp.

Performance Issues Regarding the On-the-Fly SQL Construction Approach

Software developers may overlook such performance issues. During the construction of an XML document, the usual approach that sends SQL query statements to the relational database starts by getting data from the parent tables followed by child tables. A common practice is to add a foreign key in the child table that addresses the primary key of the parent tables. The SQL query statements filter the records based on the foreign key column values. As it is not a common necessity to build indexes on foreign keys, queries based on the foreign key column will need a table scan of the child table. Therefore, such an approach does not make use of the existing primary key indexes, and the performance is therefore unacceptable, especially for constructing huge XML documents involving many SQL queries. Many existing commercial software applications and frameworks/languages proposed by various researchers (Shanmugasundaram et al., 2000; Balmin & Papakonstantinou, 2004), however, are either (a) proprietary for not generating arbitrary XML documents (b) complicated for introducing their own languages, or (c) inefficient.

As a result, a new SEML interpreter is proposed as a solution that can fulfill most requirements for exporting XML documents. The language is designed based on the following criteria:

1. The interpreter is simple, easy to learn and write.
2. It can provide all the necessary information for constructing SQL statements to be used for obtaining data from the relational database.
3. It should be in XML format so that software developers are not required to learn a new format.
4. Facilities or utilities should be available to make use of the existing schema, either
Similarity Search in Time Series Databases
www.igi-global.com/chapter/similarity-search-time-series-databases/11218?camid=4v1a

Visual Query Languages, Representation Techniques, and Data Models
www.igi-global.com/chapter/visual-query-languages-representation-techniques/7955?camid=4v1a