XML Data Management: Go Native or Spruce Up Relational Systems?

Per-Åke Larson *(Moderator)*
Microsoft Research
One Microsoft Way
Redmond, WA 98052
palarson@microsoft.com

Abstract

XML data is likely to be widely used as a data exchange format but users also need to store and query XML data. The purpose of this panel is to explore whether and how to best provide this functionality.

1. Panelists

Dana Florescu *(Propel Software Corporation)*
Goetz Graefe *(Microsoft)*
Guido Moerkotte *(University of Mannheim)*
Hamid Pirahesh *(IBM Almaden Research Center)*
Harald Schöning *(Software AG)*

2. Panel objectives

XML data is rapidly gaining acceptance. Many recommendations and standards related to XML have been proposed - see http://www.w3.org/XML for more details. It is common wisdom that XML will be the universal data exchange format but users also need to be able to store and manipulate XML data efficiently. The World Wide Web Consortium has recently released a working draft of XQuery http://www.w3.org/TR/xquery), a query language for XML data. However, a query language only defines how to express queries against XML data; we still have to build systems that store the data and evaluate the queries. The question for this panel is how to best provide this functionality.

3. XML data management issues

Like all panels, this one will go wherever the panelists and the audience decide to take it, but here is a list of possible issues to explore. First, several different architectures for XML data management systems are possible:

1. Go native, i.e. build a complete XML data management system from scratch.
2. Add a freestanding XML layer on top of a relational system. What functionality should be included?
3. Add some level of XML functionality to relational systems. If so, what functionality?
4. Modify an object-oriented database system for XML.
5. Something else altogether.

Which architecture provides the best solution hinges on the more fundamental question of what users want to do with XML data, which may depend on what type of XML data is being considered and usage patterns. This raises a number of questions.

1. What do we know about user requirements for XML data management at this stage?
2. Are there different requirements for different types of XML data?
3. If so, what characteristics of XML data are important from a data management point of view: small versus large documents, free-form versus schema-bound documents, many versus few instances, volatile versus stable, short lived versus long lived, deep vs. shallow trees, and so on?
4. Are usage patterns perhaps more important than the characteristics of the data? For example, frequency of access/update, complex queries versus simply retrieving a complete stored document, level of concurrent access, and so on.
5. Are there new system requirements, such as small memory footprint, fast startup time, or ability to run in process with the application?
6. Are clear classes of data types and usage patterns emerging, similar to, say, OLTP and OLAP, whose requirements are well understood?
7. Is there, perhaps, no silver bullet and we need multiple, complementary solutions? If the market isn't big enough to pay for the development cost, what is the best compromise?

ACM SIGMOD 2001 May 21-24, Santa Barbara, California USA