Lessons Learned from DB2 pureXML Applications: A Practitioner’s Perspective

Matthias Nicola
IBM Silicon Valley Lab,
555 Bailey Ave, San Jose, CA 95123, USA
mnicola@us.ibm.com

Abstract. Beyond using XML as a message format, more and more companies are storing XML data permanently in a database. Database researchers and vendors alike have spent a lot of effort on designing, studying, and implementing XML database technology. In this paper we report our experiences from working with a broad variety of companies that have developed and deployed XML applications on top of DB2. We discuss three real-world XML database scenarios and their design considerations, and describe recurring patterns in XML applications. We highlight common concepts and observations, and document challenges that point to future work for the database community.

Keywords: XML, database systems, experiences, XQuery, SQL/XML.

1 Introduction

XML has become the de-facto standard format for information exchange between organizations, business processes, and applications. Companies in virtually all commercial and industrial sectors have defined XML formats to standardize and streamline information management within their industry. The main reasons include that XML is extensible, flexible, self-describing, and suitable for combining structured, unstructured and semi-structured information. XML has become the fabric of Service-Oriented Architectures (SOA), which are prevalent in today's IT world.

Beyond XML as a message format, there is a continuous need to store XML permanently: sometimes for auditing and compliance reasons, sometimes because XML is a more flexible and suitable data format than a rigid relational database schema, and sometimes because using XML can simplify applications and improve their efficiency. When companies keep XML in persistent storage, they typically want to insert, index, query, and update XML with the same performance, scalability, and ACID properties as traditional data in relational databases.

In response to this trend, the research community has developed a range of XQuery processors and XML database systems such as Galax, Lore, MonetDB/XQuery, Natix, Pathfinder, Timber, and others [1]. On the commercial side, the major database vendors have added XML capabilities to their products [12][15][19] and several XML-only databases have emerged [8]. The SQL standard has been extended with an XML data type and XML-specific functions, collectively known as SQL/XML [6].
The SQL/XML functions XMLQUERY and XMLTABLE as well as the predicate XMLEXISTS enable users to include XPath, XQuery, or XQuery Updates in SQL statements. SQL/XML also allows users to construct XML from relational tables, using functions such as XMLELEMENT, XMLATTRIBUTES, and XMLAGG. All of these advances have enabled an increasing number of companies to manage XML in a database effectively.

Over several years we have worked with companies in various industries to assist them in their design and implementation of XML applications on DB2. This includes applications in retail, government, health care, finance, and others [9]. Our work with these applications allowed us to expand our understanding of their characteristics and requirements, and to assess the success of XML technologies and languages in the real-world.

After a brief summary of DB2's XML capabilities in Section 2, the main contributions of this paper are the following:

- We describe three real XML database applications, including their motivation and requirements for using XML, the database design decisions, performance trade-offs, and the benefits received from the XML database (Section 3).
- We identify a set of findings as well as recurring challenges and design patterns in XML applications (Section 4).
- Based on our findings we identify future work for the XML database community (Section 5).

2 Overview of DB2 pureXML

This section provides a brief overview of the XML support in DB2 9.7. Further details can be found in [5] and [15]. DB2 pureXML comprises a set of XML features in the DB2 database management system. At the core of pureXML is the XML data type, which can be used to define XML columns in tables and views. XML columns store XML documents in a parsed tree format that corresponds to the XQuery Data Model [24]. Each node in the tree has a pointer to its parent and pointers to all its children. If the tree is larger than a single page in a DB2 table space, DB2 automatically splits the tree into multiple regions and each region is stored on a separate page [15]. Multiple regions can reside on the same page.

When XML documents are parsed and stored, tag names are internally replaced with unique integer numbers. This reduces the space consumption and improves navigation performance through integer-based node comparisons. Additionally, DB2's table compression can reduce XML storage consumption by another 60% to 80%.

The use of XML Schemas in DB2 is optional and does not impact the hierarchical storage format. Users can choose to validate all inserted or updated documents in an XML column against the same schema or to validate different documents against different schemas. This schema flexibility is critical for many XML applications.

To improve search performance, users can define full-text indexes or path-specific XML value indexes. DB2's text indexing capabilities are based on Lucene [10] and commonly used in content-centric XML applications. In contrast, data-centric XML applications typically use XML value indexes (Fig. 1). An XML value index is a