

# Efficient Management of XML Documents

## Extended Abstract

Hongjun Lu

Department of Computer Science  
Hong Kong University of Science and Technology, China  
luhj@cs.ust.hk

XML has become a de facto standard for data representation and exchange on the World-Wide-Web. Unlike HTML tags that are mainly used to describe presentations, tags in XML capture some semantics, especially when domain-specific common DTDs are used when authoring XML documents. Since industries are indeed enthusiastic about XML, and more and more XML documents have been generated, we have to deal with the issues related to efficient management of XML documents.

**Storing XML documents.** A number of approaches have been proposed to store XML documents. Those approaches can be categorized along two dimensions. One dimension is *how an XML document is modelled*. XML documents can be managed as text files, or by a DBMS, or by a native XML engine. When managed as text files, they are viewed as character strings. When DBMS is used, they are transformed to conform a specific data model, e.g., the relational model. Most native XML engines use trees to model XML documents since elements in an XML document are ordered and strictly nested. Orthogonal to this dimension is *whether DTD is used in the storage model*. For example, when XML data is stored in a relational system, the relational schema can be generated either using or not using the type information of elements in DTD. When the schema is generated based on the DTDs, XML documents with different DTDs will have different schemas, hence the schema will be *document dependent*. On the other hand, since any XML document can be modelled as an ordered tree, a relational schema that is able to describe the tree structure, and the position of elements in such a structure is sufficient. Using this approach, no DTD information is required, and all XML documents will have the same relational schema. That is, such schema is *document independent*.

Recently, we conducted a benchmarking study to investigate the comparative performance of various schema mapping and storage methods implemented in three experimental XML database systems: VXMLR (Zhou, Lu, Zheng, Liang, Zhang, Ju & Tian 2001), XParent (Jiang, Lu, Wang & Yu 2002), and XBase (Lu, Wang, Yu, Bao, Lv & Yu 2002). VLXMR and XParent were built on top of RDBMS, and XBase is a native XML engine. Based on the categorization mentioned, the storage models compared can be summarized as in Figure 1. The benchmarks used in the study are the data centric XMark Benchmark (Schmidt, Waas, Kersten, Florescu, Manolescu, Carey & Busse 2001) and the document centric XMach benchmark (Böhme & Rahm 2001). We will

present the main results and analyze the factors that affect the system performance.

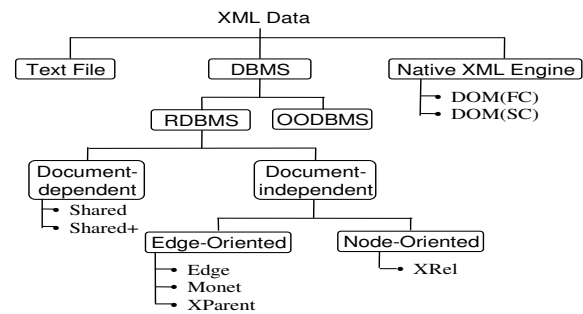


Figure 1: Managing XML Documents

**Accelerating XML query processing.** A number of XML query languages have been proposed, including XPath (Clark & DeRose 1999) and XQuery (Chamberlin, Florescu & et al 2001). It is obvious that different storage model will require different query processing techniques. For example, if a relational DBMS is used to manage XML documents, queries in XPath or XQuery need to be converted to SQL queries, whose processing will be handled by the underlying DBMS. Furthermore, like in relational systems, various access methods and indexing techniques play an important role in query processing.

Elements in XML documents are ordered and nested. That is, structural relationship exists among XML elements, and XML database systems have to support both content based queries as well as structural queries. Since content-based query processing has been well studied in both relational systems and information retrieval systems, recent work on XML query processing has been focusing on structural query processing, for which structural join is identified as a core primitive. Structural join can be defined as follows. Given two sets of XML elements,  $A$  and  $B$ , find all pairs of  $(a, b)$  such that  $a \in A$ ,  $b \in B$ , and  $a$  and  $b$  satisfy the specified structural relationship, such as, **parent-child**, and **ancestor-descendant**.

It is well accepted that region-based coding scheme is an efficient mechanism to facilitate structural join processing. Using this scheme, the position of an element occurrence in an XML document is represented by  $(start, end, level)$ , where  $start$  and  $end$  are the positions of the starting point and ending point of the element in the document, respectively, and  $level$  is the nesting depth of the element in the document. Structural relationships between elements whose positions are represented in such a way can be determined easily. For example, an element  $a$  is an ancestor of another element  $d$  if and only if  $a.start < b.start$  and  $b.end < a.end$ . An element  $p$  is the parent of

