Requirements for XML Document Database Systems

Airi Salminen
Dept. of Computer Science and Information Systems
University of Jyväskylä
Jyväskylä, Finland
+358-14-2603031
airi@cs.jyu.fi

Frank Wm. Tompa
Department of Computer Science
University of Waterloo
Waterloo, ON, Canada
+1-519-888-4567 ext. 4675
fwtompa@db.uwaterloo.ca

ABSTRACT
The shift from SGML to XML has created new demands for managing structured documents. Many XML documents will be transient representations for the purpose of data exchange between different types of applications, but there will also be a need for effective means to manage persistent XML data as a database. In this paper we explore requirements for an XML database management system. The purpose of the paper is not to suggest a single type of system covering all necessary features. Instead the purpose is to initiate discussion of the requirements arising from document collections, to offer a context in which to evaluate current and future solutions, and to encourage the development of proper models and systems for XML database management. Our discussion addresses issues arising from data modelling, data definition, and data manipulation.

Categories and Subject Descriptors
H.2.4 [**Database Management**]: Systems – *textual databases*; I.7.1 [**Document and Text Processing**]: Document and Text Editing – *document management*.

General Terms
Management, Design.

Keywords
XML, structured documents, XML database systems, data modelling, data definition, data manipulation.

1. INTRODUCTION
SGML has been a widely used markup language for defining and representing structured documents since its publication in 1986 [30]. The ongoing shift from SGML to XML is creating new demands for the management of structured documents. Compared to SGML, the variety of applications expected to use XML is much wider. On the one hand, XML will have an extended use in the application areas where SGML and HTML have already been commonly used, for example, in technical documentation of manufacturing companies, book publishing, and Web publishing. On the other hand, XML will also be used in ways SGML and HTML were not, most notably as the data exchange format between different applications. As was the situation with dynamically created HTML documents, in the new areas there is not necessarily a need for persistent storage of XML documents. Often, however, document storage and the capability to present documents to a human reader as they are or were transmitted is important to preserve the communications among different parties in the form understood and agreed to by them.

Effective means for the management of persistent XML data as a database are needed. We define an *XML document database* (or more generally an *XML database*, since every XML database must manage documents) to be a collection of XML documents and their parts, maintained by a system having capabilities to manage and control the collection itself *and* the information represented by that collection. It is more than merely a repository of structured documents or of semistructured data. As is true for managing other forms of data, management of persistent XML data requires capabilities to deal with data independence, integration, access rights, versions, views, integrity, redundancy, consistency, recovery, and enforcement of standards.

A problem in applying traditional database technologies to the management of persistent XML documents lies in the special characteristics of the data, not typically found in traditional databases. Structured documents are often complex units of information, consisting of formal and natural languages, and possibly including multimedia entities. The units as a whole may be important legal or historical records. The production and processing of structured documents in an organization may create a complicated set of documents and their components, versions and variants, covering both basic data and metadata. Thus, to accommodate structured documents and support typical applications' needs, Arnold-Moore, Fuller, and Sacks-Davis have described a structured document management system as an "authoritative document repository" that includes the following features [5]:

- on-the-fly creation of renditions
- automatic transformations
- access control at the element level
- access to elements (component versioning)
- intensional versioning
- human-readable description of changes
- extended search capabilities
- document-based workflow

However, XML imposes yet further demands:

- Closely related W3C specifications that extend the capabilities specified in XML 1.0 [12], such as XML Namespaces [11], XML Schema [8, 27, 48], and XLink [24], should be accommodated when developing XML database solutions. The accommodation should adapt to the continuing development and re-development of the specifications.
- XML is intended especially for use on the Internet. References in XML documents refer to Internet resources, and thus XML database systems should include Internet resource management. In the Internet environments integration of the management of structured documents with the management of other kinds of documents and data is also important.
- An SGML document was always associated with a DTD[1], and the DTD could be used in many different ways to support the data management. XML documents do not always have an associated DTD.

The database research community has been actively investigating XML (see, for example, [1] and [49]). Much of the effort has been directed at using XML as a database wrapper and mediation medium, using XML to describe Web resources, storing and indexing XML in traditional database systems, understanding the interaction of DTDs with constraint and typing mechanisms, and designing query languages for XML. In an influential paper, Maier examined XML query language proposals from the database perspective [38], but broader management issues peculiar to XML databases have not yet received much attention.

2. THE DATA MODEL

A well-defined database system is based on a well-defined data model. The complexity of XML-related data repositories and the need to integrate the management of structured documents with the management of other types of data creates a special challenge for the underlying data model. In research papers the XML data model is often simplified to a labeled tree, or a directed graph, including elements with their character data, and attributes with their values. Sometimes the elements are ordered (e.g. [28]), and other times they are not (e.g. [3]). This kind of simplified model may be sufficient for developing capabilities dealing with the hierarchic structure of elements. To be able to manage XML documents as a database, however, requires a richer data model. We present three data model features that we regard as important for the underlying data model.

2.1 Modelling Document Collections As Well As Enterprises

Unlike conventional databases, the data in a document database does not represent an enterprise directly. Instead it represents a collection of documents, which, in turn, captures the information embodying the enterprise. The data model should support the description of the documents as they are built from multimedia storage units and symbols, as well as the description of the enterprise reflected by the information in the documents' contents. This has always been a major challenge for text data modelling [42].

The XML 1.0 specification defines the components of individual XML documents, partitioning them into *logical structures* ("declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup") and *physical structures* (entities, which may include entity references). The text stored within these structures may represent character data, markup, white space, or end-of-line markers. These two structures are described by grammar rules in the XML 1.0 specification, and these rules define what is an XML document. The specification is the basis for standardized data exchange between different types of applications, and therefore XML database systems must preserve and present the standard format.

Abstract structures for XML documents have been developed in four different specifications proposed through W3C: the Infoset model [21], the XPath data model [18], the DOM model [6, 37], and the XQuery 1.0 and XPath 2.0 Data Model [29]. These models do not describe explicit text markup, but they do describe document structure, which is often used to encode enterprise data. Markup languages like XML can be used in various ways, but typically markup is intended for computers to process documents, and the character data is intended to be represented to human readers. Following this convention the character data, in other words the content of XML elements without markup, often represents enterprise data. Thus the element declarations of the document type declaration typically define the structure of an enterprise, and comments and processing instructions are not part of the enterprise data. Attributes, however, are sometimes used for metadata and sometimes used to encode enterprise data components (either to avoid imposing an order or to refer to externally stored components).

Table 1 summarizes the features of the four W3C data model specifications. Each of the models describes an XML document as a tree, but there are differences in the trees. Although these variations may not impact the models' ability to represent enterprise data for most applications, they do impact the ability to provide consistent and uniform management of documents across diverse applications.

It is important to note that among the four models only the XQuery 1.0 and XPath 2.0 Data Model acknowledges that the data universe includes more than just a single document. Furthermore, it is also the only model that includes interdocument and intradocument links in a distinct node type (i.e., *Reference*). An XML database system should be built on a model that supports collections of inter-related documents, only some of which are validated against document type declarations, together with document fragments and other related forms of data.

2.2 Conceptual Model for Documents

It is well accepted by the database community that data should be managed through a three-level architecture that separates the conceptual model from an internal model and a set of external models. Furthermore, it is understood that data independence relies on the principle that the conceptual model is shielded from the physical arrangement of the data on storage devices and

[1] A DTD was required in the original specification of the SGML standard [30]. Annex K published in 1997 [33] distinguishes two kinds of SGML documents: *type-valid* and *fully-tagged*. A fully-tagged SGML document does not require an associated document type declaration.

Table 1. Characteristics of the four XML data models

	XML Information Set [21]	**XPath 1.0 data model** [18]	**DOM 1.0 Level 2** [37]	**XQuery 1.0 and XPath 2.0 data model** [29]
Purpose	To provide a set of definitions for use in other specifications that need to refer to the information in an XML document.	To provide the basis for the XPath language specification, which in turn is intended to be a component that can be used by other specifications, primarily by XPointer and XSLT.	To provide the basis for a platform- and language-neutral interface that allows programs and scripts to access and update the content and structure of documents dynamically.	To define precisely the information contained in the input to an XSLT or XQuery processor, and to define all permissible values of expressions in the XSLT, XQuery, and XPath languages.
Development phase	Proposed Recommendation	Recommendation	Recommendation	Working Draft
What is modelled?	XML document	XML document	XML (or HTML) document	Sequences of XML documents or parts
# of node types in the tree structure	11	7	12	8
Node types	Document Document Type Declaration Unparsed Entity Notation Element Attribute Namespace Processing Instruction Comment Unexpanded Entity Reference Character	Root Element Attribute Namespace Processing instruction Comment Text	Document DocumentFragment DocumentType Entity Notation Element Attr ProcessingInstruction Comment EntityReference CDATASection Text	Document Element Attribute Namespace Processing instruction Comment Reference Text
DTD or XML Schema validity required?	no	no	no	validity required if there is an associated XML Schema or DTD; otherwise no
Examples of information omitted	- the order of declarations within the DTD - content models of elements - document type name - difference between two forms of empty elements - comments in the DTD - the boundaries of CDATA marked sections - the order of attributes within a start tag - the location of declarations (whether in internal or external subset or parameter entities)	- all what is listed missing in the Infoset model - all information about entities - distinction between default attribute values and specified attribute values - information about the type of an attribute (e.g. ID, IDREF, ENTITY)	- all information about parameter entities - information about the external subset - declarations as such and their location - types of attributes	- all information that is missing from the Infoset model - all information about entities - distinction between default attribute values and specified attribute values

embodies the "universe of discourse" for all applications, which must access the data through the external models [34].

Applying these principles to an XML database necessitates that the conceptual model incorporates not only all the objects and relationships that are to be modeled in the enterprise, but also all the document components that are to be made available to any XML application. With such a universal conceptual model at its core, an XML database can then include external models that view the database as having only document features or only enterprise features, or any combination of document and enterprise features that are required for various classes of applications. The production rules of the XML 1.0 specification offer a basis for such a universal model, since they define all of the information encoded in an XML document. Unfortunately, in contrast to the relational model, a model covering the physical and logical structure of the XML specification is intricate and very detailed. Nevertheless, the details are needed if the model is to serve as a basis both for views describing the document and for views describing the enterprise. Both views are also important to describe collections digitized legacy paper documents, often having a requirement to capture both the enterprise features and the original rendition features.

2.3 Well-defined Equivalence

Electronic documents are often legal, historic, or business transaction records, and queries against such documents typically involve entities and relationships that represent features of the text itself as well as features of the businesses involved in the contractual agreements. For an XML database one fundamental semantic issue is *document equivalence* [40]: when are two