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

documents or document parts or document DTDs the same? For example, before inserting a document into the database, we might want to find out if the same document is already in the database. The question of equivalence is important in satisfying requirements for evidence and archiving, for version management, for metadata management, and (as is true of all forms of data) for query optimization.

The XML 1.0 specification does not define equality of documents or equality of entities, nor do the Infoset, XPath, or DOM models. The XQuery 1.0 and XPath 2.0 Data Model includes one equality operator to test node identity and another to test equality of values. However semantics for the equality of node values does not encompass all data from XML documents. W3C has proposed that Canonical XML [10] be used to compare the equivalence of two documents. The canonical form is created by a process called canonicalization either from an XPath node set or an octet stream containing a well-formed XML document. In both cases canonicalization omits some of the information in the original XML document. Since such a canonical form does not contain all information from an XML document, this definition of equivalence may not satisfy all applications' needs. One solution is to define document equivalence in terms of a model that includes *all* document features, after which application-dependent definitions of equivalence can be specified by applying document equivalence to application-specific transformations of the documents to be compared.

3. DATA DEFINITION

3.1 Data Types

The XML specification offers the capability to define document types, element types, and the type of an attribute (e.g., CDATA, ID, IDREF, ENTITY, NMTOKEN), but the content of atomic elements always consists of text. Maier and others have highlighted the need for broader data typing mechanisms to allow more powerful constraining mechanisms than is available in XML. These require the development of a suitable Data Definition Language (DDL) together with a corresponding Data Manipulation Language (DML) with appropriate operations for each kind of data [38].

The XQuery 1.0 and XPath 2.0 Data Model describes an XML document as one that may be associated with an XML Schema after schema validation. The value of an attribute and an atomic element can be one of nineteen primitive data types defined in the XML Schema specification.

3.2 Document Types

As well as supporting a variety of base types, an XML database system should support collections that include multiple *document* types. The language for describing the types should include all the functionality of DTDs, since they are fundamental to XML definitions and will likely be central in data interchange. Each of the several schema languages developed or under development for XML provides a mechanism to constrain the structure and content of a class of XML documents [20]. The primary purpose of these languages has been to allow the validation of a given well-formed XML document against a schema. However, in the context of XML databases, it is expected that these languages will also serve as the means to specify which operations on the data are valid.

3.3 Data Collections

The DDL should allow the definition of *collections* of XML documents and document parts, together with collections of values of various data types that are not required to be (even logically) a part of any document. Towards this end, the W3C proposal for the XQuery 1.0 and XPath 2.0 Data Model supports a "flat" *sequence* collection, not allowing sequences inside sequences. To be able to apply a query language to collections, the DDL should offer the capability to define such collections. In addition, the capability to declare collection hierarchies is important, for example, to support flexible definition of access rights and views.

3.4 Document Type Collections

Structured document management often requires a versatile collection of document types, even for the same material. Case studies confirm that production and publication of structured documents often requires multiple document type definitions that represent various *versions* developed over time as well as several *variants* covering different phases of document production [26, 35, 43]. Furthermore, these studies confirm that the data content should be preserved in its variant forms corresponding to different document type definitions.

The DDL of an XML database system should support the definition of multiple document type definitions, their organization into manageable collections, their presentation as data (typically in XML format), and their role as metadata constraining other data in the database instance. Furthermore, the DDL should provide capabilities to manage different document type definition versions and variants and do so as new document type definitions are created and existing ones are updated.

The need for several document type definitions for the same material and for different document type definition versions has partly evolved from the immaturity of software and from the experimental nature of SGML and XML solutions for document creation. In light of the growing use of XML for various types of data and the simultaneous increase in the diversity of presentation media, it is clear that the need for managing rich collections of document type definitions in a single environment will increase. Since XML involves many forms of data manipulation, many forms of media, and many persons having diverse qualifications and application needs, all in the presence of continually changing international and industry-level standards, document type definitions will be "alive," and the database system should support the management of their evolution.

3.5 Multiple Levels of Validity

A database should support multiple levels of validity for XML data. For example, we may wish to define a database subcollection or a view consisting of

- non-XML data values from a set of types (e.g., numbers, dates, strings, images, tables),
- well-formed XML documents,
- valid XML documents, each associated with some document type definition provided by a user or application,
- valid XML documents, each associated with a document type definition from a closed set known to the database system

(either predetermined by the database administrator or pre-registered by some application), or
- parts of well-formed or valid XML documents.

In this context, therefore, the DDL should support not only the declaration of XML data and its level of validity but also the declaration of the type of XML schema definition against which validity is to be judged and the schema declaration itself.

We note that the adoption of XML Schema will have a major impact on content authoring, which will increase the need for multiple levels of validity in XML databases. The inclusion of a rich data type mechanism in XML schema languages has been motivated primarily by the needs of electronic commerce, where much data is numeric and produced by software. However, this will make document creation by humans still more challenging than earlier, when constraint checking was restricted to conformance with an XML 1.0 DTD. In the future, authors must also understand the variety of data types used in the schema and ensure that the documents they create conform to the richer constraint mechanisms. Thus, the extent to which rich data types are adopted in document authoring by humans and in which phases of content production they are introduced will influence in how many different stages of validation documents will be stored in the database.

3.6 Entities and URIs

Entities are used in XML document repositories to avoid redundancy. For example, a technical documentation suite may involve thousands of images, and a specific image may be used in several places. Each image is stored once as an image file, and the documents or elements containing the image refer to the file by an *entity reference*. Similarly, pieces of text defined as entities can be reused in different places of documents of a document collection via entity references. In XML, references to entities *internal* to a document are shorthand notations that are replaced by their values in the abstract structure of the document, as if they were parameterless macros. *External* entities, however, are referenced by URIs, and in the abstract structure their contents remain outside the entities from which they are referenced.

The central idea in the specification of XML and the URI addressing mechanism has been to create a human readable notation for information management on the Internet, where readability encompasses the physical structure as well as the logical structure of documents. The URIs of accessible entities must be available to applications, and they will also be stored beyond a single enterprise's control in extranet environments, where several organizations share database resources. In the absence of careful attention, therefore, entities, files, and URIs will violate data independence by exposing to application programs some storage decisions made at the internal level of an XML database.

In conclusion, it is tempting to relegate entities to the internal schema as a means to reduce storage redundancy. However, because XML applications may well rely on resources that are external to the database, they will depend on the entity-URI mechanism. Thus, an XML database system should support entities at the conceptual schema level and use a complementary mechanism internally.

3.7 Support for Namespaces

XML namespaces provide a method for qualifying element and attribute names in XML documents by associating them with namespaces identified by URI references. To be able to use particular namespaces for a specific database, the DDL should include a capability to define the names included in a namespace and optionally the data types that are to be associated with those names (for situations in which applications are dependent on the types). As Maier has already noted, the database system should also provide views of XML documents in which the presence of document-specific namespace identifiers are replaced by document-independent identifiers (i.e., fully expanded URIs in general).

3.8 Document Indexing

Document indexing assigns content indicators, called *index terms*, to documents. These terms are then used by retrieval systems to access the documents. For many applications, a human indexer may choose the terms, as is almost inevitably done for indexing non-text documents. Other applications rely on *full-text indexing*, in which a subset of words (or phrases) occurring in a document are chosen as index terms and assigned to the document. The appropriateness of a full-text indexing method to a specific document repository depends on the language and content domain of its documents. For example, the indexing terms that are effective for a repository of English novels will perform poorly when used against a repository of Finnish technical documentation.

The DDL for an XML database should allow application programs to specify the rules for indexing documents (and the data manipulation language should provide facilities for querying the indexing rules and for choosing which indexes to use to execute a given query). This need extends to indexing structure as well as text content. Furthermore, the DDL should have facilities to bind a collection of such rules to the whole database, to a subcollection of the database, or to a view.

3.9 User Roles and Access Rights

An XML database including data for a variety of purposes and diverse users needs role-based access control [7, 22, 36]. Definition of role hierarchies, the hierarchic structure of XML documents, and hierarchic document containers allow the specification of very fine-grained authorization. The challenge for XML database systems is to support such fine-grained access control efficiently in very large database environments with very many users, each shifting among many possible roles.

4. DATA MANIPULATION

The development of XML query languages has been based on extensive discussion about the desired characteristics of such languages [38, 15]. We will not repeat all those characteristics here; instead we discuss those characteristics of the Data Manipulation Language (DML) that are important for manipulating persistent XML data in a controlled way, in the context of a system having the definition capabilities described in Section 3.

4.1 Queries

In an XML database we should be able to express queries in terms of all data in the database, including entities, URIs, tags,

comments, processing instructions, schemas and other metadata. The latest proposals for XML query languages, including Lorel [31], XML-QL [25], XQL [41], and XQuery [14] all omit some of the data from XML documents. The document type definition, entities, entity references, and notations are not accessible through any of these languages, and Lorel also ignores comments and processing instructions. Each item of data, however, may provide important information for managing parts of the database.

4.2 Transformations

In traditional databases the most important group of operations consists of queries. In structured document management environments transformations are typically at least as important as queries that retrieve a subset of data. Hence the DML should include flexible means to specify transformations for various needs [47].

At some level, there is no clear distinction between queries and transformations, which has led to extensive discussions surrounding the roles of XQuery and XSLT and whether the efforts in one of these directions should be abandoned in favour of the other (http://www.xml.org/xml-dev/). Nevertheless, in a traditional database query we specify the data we wish to retrieve, and the form of the result is of secondary interest, often determined largely by the data model or by the system. In contrast, a transformation specification is primarily concerned with the form of the result and secondarily includes criteria to include or omit various parts. Transformations are needed, for example, for the following purposes:

Rendering. Flexible rendering capabilities are not important in databases where XML is primarily used for data exchange between applications and the database is an archive of transactions. In the rare occasions when the data is presented to a human reader, some simple predefined external format may be appropriate. However, such limited control is not satisfactory for many applications.

Because presentation media for documents are diverse and new media are continuously being developed, there should be flexible means to specify how to render XML on various types of media. The specification may require first a transformation of the content, and then the attachment of layout information. For example, displaying the content of an HTML page on a small screen of a mobile device may require removal of images, partitioning the page into clusters suitable to the small screen, and adding some style information. The XML database system should provide capabilities both for persistent storage of specifications for rendering, such as XSL style sheets [4] together with XSLT transformation descriptions [17] and for dynamic production of external presentations.

Integration support. An XML database system should include capabilities to import and export data between the database and other systems. This typically requires some transformation of the data.

Schema evolution. In Section 3, we discussed the problem of multiple document type definitions in XML document production environments. Because changes in document type definitions are quite common, there is often a need to transform existing data to correspond to a new definition.

Views. In all databases, view definition capability is an important means to provide data independence in the presence of database growth and restructuring, to allow data be seen in different ways by different applications, and to provide security for hidden data. The complexity and evolving nature of XML databases makes a view definition mechanism critical to a system's usability.

Whether defining a virtual or materialized view, a view definition typically hides part of the database. The DDL of an XML database system should allow the hiding of the physical structure of XML documents, specific types of documents or links, specific elements or attributes in documents, all comments and processing instructions in documents, style sheets, or a subset of metadata. However, in addition to removal of data, an XML view specification will typically include other transformations, for example, to change element and attribute names or change the order or hierarchical organization of elements. Whereas SQL and other database systems specify views through their query languages, it may be more appropriate in an XML database system to base view definitions on a transformation language.

4.3 Document Assembly

Document assembly is the process of constructing a new document from fragments of a collection of existing documents. For example, the manufacturer of a complex piece of machinery may create a large collection of documentation, from which several manuals can be assembled to meet the needs of various classes of users. To facilitate flexible creation and update of assembled documents, systems should offer support for the specification of the assembly process, which typically includes a compound operation involving several queries and transformations.

4.4 Update

As for other database systems, the update operations for an XML database include insertion, deletion, and replacement. The data affected can be a whole document, part of a document, a file, a URI, a style sheet, or any other unit. Furthermore the affected component may be either basic data or metadata, such as a DTD, a set of RDF descriptions for resources within the database or outside it, or a set of links. The DML should provide mechanisms for applications to distinguish updates that cause the creation of new documents from those that create new versions or new variants of existing document parts.

An application may activate an update by specifying a transformation that is to persist in the store. In many environments various users in different roles maintain the content of structured document repositories through a complicated process in which documents are developed gradually and collaboratively. Such processes rely on *XML editors* and *support for workflow management and collaboration*, which should be integrated with XML database systems.

An XML database may contain various forms of reference: entity references, intradocument IDREFs, and inter-document links, where the links can be embedded HTML-like links or richer XLink-type links. The requirement of *referential integrity* is an important goal for an XML database, restricting updates such that all entity references, IDREFs, and links to documents within the database have existing targets. Traditional mechanisms to disallow or to cascade updates that would otherwise violate referential integrity should be supported.

A major concern in updating traditional databases has been transaction management. Database systems include as part of their DMLs capabilities for applications to specify the scope of each transaction. In XML database systems, an XML document is a natural unit for specifying the collection of operators that must be executed as an atomic transaction, whether the data units to be updated are documents, document fragments, or nodes. The DML should include a mechanism in which an application request is presented to the database system in the form of an XML document. Examples of XML-based "languages" to express transactions that are common to various business sectors are being continually developed (see, for example, [19]).

5. SYSTEMS FOR MANAGING PERSISTENT XML DATA

In the time since the publication of the SGML standard, no widely accepted single model or technology for the management of SGML/XML document repositories as a database has evolved. The generic names for candidate management systems have varied, and the boundaries between types of systems are fuzzy. For example, under the title of "XML database products" Ronald Bourret separates the following categories: middleware, XML-enabled databases, native XML databases, XML servers, XML application servers, content management systems, and XML query engines [9]. There are also activities towards developing specialized search engines for XML documents on the Web. These systems, such as Xyleme [51] and Niagara [39], bring new information management capabilities to the Web, but they cannot be characterized as complete database systems. Below we consider two broad categories of systems: native SGML/XML systems and extensions of relational and object-oriented database systems. We characterize the categories in terms of the features we discussed in the previous sections: data model, data definition and data manipulation.

5.1 Native SGML/XML Systems

Native SGML/XML systems are designed especially for the management of SGML/XML data. The systems should include capabilities to define, create, store, validate, manipulate, publish, and retrieve SGML/XML documents and their parts. Some of the native systems, such as Astoria [16] and Information Manager [32], are comprehensive document management systems with front-ends for users to work with documents. Some others, such as SIM [44] and Tamino [46], are software packages intended for building applications for the management of SGML/XML data. A few systems, especially those that support semistructured data, such as Lore [31], XYZFind [52], and dbXML [45], provide native support for tree-structured data but are limited in their support of rich XML documents because they do not rely extensively on DTDs or other document type definitions.

The data model. As we discussed in Section 2, there is no single well-defined data model for XML data. The lack of a well-defined universal conceptual model causes problems in the native systems: for example, the underlying model for XML data is not explicitly defined in Astoria or Tamino, and system-specific notions and models have been invented in SIM. Many of the systems consist of packages of tools that do not share a common data model and may be limited in kind of XML documents they are able to store and manipulate. Unfortunately, because the systems do not highlight the details of the data model, such inconsistencies and constraints are often difficult to detect.

In Tamino the database data model is a tree, and a central notion is the "XML object." Nevertheless, the data model is not explicitly described, and the extensive glossary for Tamino's documentation does not include the notion of an XML object. The data model derives from the data model of the XQL language [41], which is the query language in Tamino (with some limitations and some extensions). It consists of nodes, representing elements and attributes in an XML document, and parent-child relationships between them. The Tamino tree structure lacks comments and processing instructions that form part of the XQL structure. On the other hand, Tamino allows the association of a set of data types to nodes in the tree.

Data definition. The capability to define document types is an important characteristic of XML, and we consider the document type definition capability an essential feature in systems of this category. This aspect severely reduces the utility of semistructured approaches for managing persistent XML resources. The systems originally developed for SGML are able to use DTDs directly as the document type definition with no translation to some other form of schema. Additional definitions may be needed, however, to support flexible manipulation and efficient implementation. In Astoria an important extension is provided by *components*, which form the data unit for many operations. For example, access rights are granted at the component level, components can have variants and versions, and simultaneous update to a document by several users is controlled at the component level.

Tamino provides a proprietary schema language for data definition. The database can be defined to contain collections of XML documents of given types and non-XML objects. Thus there can be collections of document types and collections of documents. A document type can be created from a DTD, but the creation is not automatic. Tamino's document type definition capabilities both restrict and extend the definition capabilities of DTDs. A Tamino document type does not include information related to entities, attribute types, or URIs. Furthermore, some information about the content models is simplified; for example, information about optionality of an element is not saved. On the other hand, a document type in Tamino includes definitions related to indexing and data storage. There is no special support for defining namespaces, and there are two essential forms of validity: the validity of non-XML data with respect to their data types and the validity of XML data with respect to the corresponding Tamino document types. XML documents without a Tamino schema can be stored in the database, but they are stored as indexed text with no XML structure and thus are treated as non-XML data. Versions can be defined at the database level, and authorization is restricted to schemas.

Data manipulation. The lack of a standardized XML query language has led to various system-specific query languages. In addition, the simplified data models restrict query capabilities. For example, since Tamino does not store information about attribute types, queries utilizing IDs and IDREFs are impossible. The response to a Tamino query is an XML document containing the query result as tagged text, plus metadata related to the query (e.g. date and time). Thus the query language cannot be applied directly to query results unless a Tamino schema defines them as part of the database. In content management systems such as

Astoria and Information Manager, parts of documents can be updated by structure editors integrated with the systems. In both of them style sheets can be associated with documents in their associated editors, and transformations can be defined by means of style sheets. Both of the systems also offer some capabilities for document assembly. In Tamino, database update is applied at the document level. The data storage mechanism for XML data (called X-Machine) has an associated programming language that includes commands for inserting and deleting documents. XSL is used to transform XML documents to HTML for Web publishing, but there is no additional support for defining transformations.

5.2 Extensions of Relational and Object-oriented Database Systems

In this approach a relational or object-oriented database system is extended to support SGML/XML data management. The proposed SGML extensions included, for example, a system where SGML files were mapped to the O_2 database management system [2], and the extension of operators of SQL to accommodate structured text [13]. All current commercial database systems provide some XML support. Examples of commercial systems are Oracle's XML SQL Utility [50] and IBM's DB2 XML Extender [23]. For the sake of discussion, we consider IBM's DB2 XML Extender as representative of the many systems following this approach.

Data model. When conventional database systems are used for XML, data structuring is systematic and explicitly defined by a database schema. The data model of the original system is typically extended to encompass XML data, but the extensions define simplified tree models rather than rich XML documents. The XML extensions are intended primarily to support the management of enterprise data, wrapped as elements and attributes in an XML document. A problem in using the systems is the need for parallel understanding of two different kinds of data models.

Data definition. The extended systems require explicit definition of transformation of a DTD to the internal structures. XML elements are typically mapped to objects in object-oriented systems, but relational systems require more elaborate transformations to represent hierarchic and ordered structures in unordered tables. In the DB2 XML Extender the whole document can be stored either externally as a file or as a whole in a column of a table. Elements and attributes can also be stored separately in side tables, which can be accessed independently or used for selecting whole documents (as if the side tables were indexes). DTDs, which are stored in a special table, can be associated with XML documents and used to validate them.

Data manipulation. In relational extensions, whole documents and DTDs that are stored in tables can be accessed and manipulated through the SQL database language. As explained above, specific elements of XML data can be extracted when documents are loaded, maintained separately, and accessed directly through SQL. Support for accessing elements that have not been extracted as part of document loading is provided through limited XPath queries, and the DB2 XML Extender can be used together with DB2 UDB Text for full-text search. DB2 also provides document assembly via a function call that can be embedded in an SQL query.

6. CONCLUSION

In many environments collections of XML documents will be carriers of large bodies of information related to a particular enterprise or crossing enterprise boundaries. The information must be securely accessible, often for a long time, despite continuing changes both in technology and in participating enterprises, and despite heterogeneity in the user community. The special characteristics of XML data cause problems when adapting database management principles and systems to XML data. In this paper we have discussed these characteristics and derived a set of desired features for XML database management systems.

Data model, DDL, and DML design must be coordinated if the resulting system is to be consistent. Much effort has been devoted to data definition for the purpose of validation and to query language features. We believe that now the highest priority is to define a complete data model that covers enterprise and document data, serves as a means to define conceptual schemas, and defines the mechanism to answer whether any two items of data are equivalent. We are encouraged by the move towards convergence of the XPath and XQuery data models; if convergence with the DOM and Infoset models were undertaken, a complete and stable database model might evolve. DDLs and DMLs can then be defined to include all components of the model.

We believe that priority should also be given to developing mechanisms to manage collections of DTDs and other document definitions along with managing the documents themselves. This is especially important in the context of managing diverse collections of documents, each of which encompasses many versions and variants and subject to various levels of validity.

The purpose of the paper is to initiate discussion of the requirements for XML databases, 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. A well-defined, general-purpose XML database system cannot be implemented before database researchers and developers understand the needs of document management in addition to the needs of more traditional database applications.

7. ACKNOWLEDGEMENTS

Ideas for this paper have arisen from many sources, and we particularly thank members of the Database Research Group at the University of Waterloo and of the inSGML project at the University of Jyväskylä, as well as colleagues at Open Text Corporation. We gratefully acknowledge the financial support provided by the Academy of Finland (under Project 48989), the University of Waterloo, the Natural Sciences and Engineering Research Council of Canada, and Bell University Labs.

8. REFERENCES

[1] Abiteboul, S., Buneman, P., and Suciu, D. Data on the Web. Morgan-Kaufmann, 2000.

[2] Abiteboul, S., Cluet, S., and Milo, T. Querying and updating the file. in Proc. of the 19th Int. Conf. on Very Large Databases (1993), 73-84.

[3] Abiteboul, S., Segoufin, L., and Vianu, V. Representing and querying XML with incomplete information. in Proc. 20th Symp. on Principles of Database Systems (2001), 150-161.

[4] Adler, S., Berglund, A., Caruso, J., Deach, S., Grosso, P., Gutentag, E., Milowski, A., Parnell, S., Richman, J., and Zilles, S. (eds.). Extensible Stylesheet Language (XSL) Version 1.0, W3C Candidate Recommendation 21 November 2000. http://www.w3.org/TR/2000/CR-xsl-20001121/.

[5] Arnold-Moore, T., Fuller, M., and Sacks-Davis, R. Approaches for structured document management. Markup Technologies (MT'1999).

[6] Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Le Hors, A., Nicol, G., Robie, J., Sutor, R., Wilson, C., and Wood, L. (eds.). Document Object Model (DOM) Level 1 Specification Version 1.0, W3C Recommendation 1 October, 1998. http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/.

[7] Bertino, E., Castano, S., Ferrari, E., and Mesiti, M. Controlled access and dissemination of XML documents. in Proc.2nd International Workshop on Web Information and Data Management (1999), 22 – 27.

[8] Biron, B. V., and Malhotra, A. (eds.). XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[9] Bourret, R. XML Database Products. http://www.rpbourret.com/xml/XMLDatabaseProds.htm, updated August 13, 2001.

[10] Boyer, J. Canonical XML Version 1.0, W3C Recommendation, 15 March 2001. http://www.w3.org/TR/2001/REC-xml-c14n-20010315.

[11] Bray, T., Hollander, D., and Layman, A. (eds.). Namespaces in XML, World Wide Web Consortium, 14 January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/.

[12] Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E. (eds.). Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000. http://www.w3.org/TR/2000/REC-xml-20001006/.

[13] Brown, L. J., Consens, M. P., Davis, I. J., Palmer, C.R., and Tompa, F. W. A structured text ADT for object-relational databases. Theory and Practice of Object Systems 4, 4 (1998), 227-244.

[14] Chamberlin, D., Clark, J., Florescu, D., Robie, J., Siméon, J., and Stefanescu, M. (eds.). XQuery 1.0: An XML Query Language, W3C Working Draft, 07 June 2001. http://www.w3.org/TR/2001/WD-xquery-20010607/.

[15] Chamberlin, D., Fankhauser, P., Marchiori, M., and Robie, J. XML Query Requirements, W3C Working Draft 15 February 2001. http://www. w3.org/TR/2001/WD-xmlquery-req-20010215/.

[16] Chrystal Software, Astoria. http://www.chrystal.com/product/astoria/index.htm, retrieved August 2001.

[17] Clark, J. (ed.). XSL Transformations (XSLT) Version 1.0, W3C Recommendation, 16 November 1999. http://www.w3.org/TR/1999/REC-xslt-19991116.

[18] Clark, J., and DeRose, S. (eds.). XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[19] Cover, R. (ed.). The XML Cover Pages. http://www.oasis-open.org/cover/, retrieved August 2001.

[20] Cover, R. (ed.). XML schemas. http://www.oasis-open.org/cover/schemas.html, retrieved August 2001.

[21] Cowan, J., and Tobin, R. (eds.). XML Information Set, W3C Proposed Recommendation 10 August 2001. http://www.w3.org/TR/2001/PR-xml-infoset-20010810.

[22] Damiani, E., De Capitani di Vimercati, S., Stefano Paraboschi, S., and Samarati, P. Fine grained access control for SOAP E-services. in Proc. Tenth International World Wide Web Conference (Hong Kong, May 2001), 504-513.

[23] DB2 Universal Database XML Extender, XML Extender Administration and Programming. http://www-4.ibm.com/software/data/db2/extenders/xmlext/docs/v71wrk/english/index.htm, retrieved August 2001.

[24] DeRose, S., Maler, E., and Orchard, D. (eds.). XML Linking Language (XLink) Version 1.0, W3C Recommendation, 27 June 2001. http://www.w3.org/TR/2001/REC-xlink-20010627/.

[25] Deutsch, A., Fernandez, M., Florescu, D., Levy, A.Y., and Suciu, D. XML-QL: a query language for XML, Submission to W3C, NOTE-xml-ql-19980819 (19-August-1998). http://www.w3.org/TR/NOTE-xml-ql/.

[26] Fahrenholz, S. SGML for electronic publishing at a technical society – Expectations meets reality. Markup Languages: Theory and Practice 1, 2 (Spring 1999), 1-30.

[27] Fallside, D.C. (ed.). XML Schema Part 0: Primer, W3C Recommendation 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/.

[28] Fan, W., and Libkin, L. On XML integrity constraints in the presence of DTDs. in Proc. 20th Symp. on Principles of Database Systems (2001), 114-125.

[29] Fernández, M., and March, J. (eds.). XQuery 1.0 and XPath 2.0 Data Model, W3C Working Draft 7 June 2001. http://www.w3.org/TR/2001/WD-query-datamodel-20010607/.

[30] Goldfarb, C. F. The SGML Handbook. Oxford University Press, Oxford, UK (1990).

[31] Goldman, R., McHugh, J., and Widow, J. From semistructured data to XML: Migrating the Lore model and query language. in Proc. Int. Workshop on the Web and Databases (WebDB'99) 1999, 25-30.

[32] Information Manager™, Standard-based content management, Interleaf. http://www.interleaf.com/products/im.htm, retrieved August 2001.

[33] ISO/IEC JTC1/WG4 N1955, Document Description Languages, 1997. http://www.ornl.gov/sgml/wg8/document/1955.htm.

[34] Jardine, D. A. (ed.). The ANSI/SPARC DBMS Model. North-Holland, 1977.

[35] Karjalainen, A., and Tyrväinen, P. Defining genres and their features for studying information reuse: Preliminary findings. in Proc. of IRMA 2001, Information Resources Management Association International Conference (Toronto, May 2001), Idea Group Publishing, 346-348.

[36] Kudo, M, and Hada, S. XML document security based on provisional authorization. in Proc. 7th ACM Conf. on Computer and Communications Security (Athens, Nov. 2000), 87-96.

[37] Le Hors, A., Le Hégaret, P., Wood, L., Nicol, G., Robie, J., Champion, M., and Byrne, S. (eds.). Document Object Model (DOM) Level 2 Core Specification Version 1.0, W3C Recommendation 13 November, 2000. http://www.w3.org/TR/2000/ REC-DOM-Level-2-Core-20001113.

[38] Maier, D. Database desiderata for an XML query language. QL'98 – The Query Language Workshop, W3C, (Boston, Dec. 1998).

[39] Naughton, D., DeWitt, D., Maier, D., Aboulnaga, A., Chen, J., Galanis, L., Kang, J., Krishnamurthy, R., Luo, Q., Prakash, N., Ramamurthy, R., Shanmugasundram, J., Tian, F., Tufte, K., Viglas, S., Wang, Y., Zhang, C., Jackson, B., Gupta, A., and Chen, R. The Niagara Internet Query System. IEEE Data Engineering Bulletin 24,2 (2001), 27-33.

[40] Raymond, D. R., Tompa, F. W., and Wood, D. From data representation to data model: meta-semantic issues in the evolution of SGML. Computer Standards & Interfaces 18 (1996), 25-36.

[41] Robie, J. (ed.). XQL (XML Query Language), August 1999. http://www.ibiblio.org/xql/xql-proposal.html.

[42] Salminen, A., and Tompa, F. W. Grammars++ for modelling information in text. Information Systems 24, 1 (1999), 1-24.

[43] Salminen, A., Lyytikäinen, V., Tiitinen, P., and Mustajärvi, O. SGML for E-Governance: The case of the Finnish Parliament. in Proc. 11th International Workshop on Data and Expert Systems Applications (2000), 349-353.

[44] The Structured Information Manager. http://www.simdb.com/, retrieved August 2001.

[45] Staken, K. dbXML Users Guide version 0.9, 2001/06/04. http://www.dbxml.org/docs/UserGuide.html.

[46] Tamino Version 1.2.1. http://www.cs.uni-essen.de/dawis/teaching/ss2000/nsdb/tamino/help/overview.htm, retrieved August 2001.

[47] Tang, X., and Tompa, F. W. Specifying transformations for structured documents. in Proc. 4th Int. Workshop on the Web and Databases (WebDB) 2001, 67-72.

[48] Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (eds.). XML Schema Part 1: Structures, W3C Recommendation 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

[49] Vianu, V. A Web odyssey: from Codd to XML. in Proc. 20th Symp. on Principles of Database Systems (2001), 1-15.

[50] Wait, B. Using XML in Oracle database applications, Nov. 1999. Oracle Corporation. http://technet.oracle.com/tech/xml/info/htdocs/otnwp/about_xml.htm.

[51] Xyleme, L. A dynamic warehouse for XML data of the Web. IEEE Data Engineering Bulletin 24,2 (2001) 40-47.

[52] XYZFind, The schema-independent native XML database. http://www.xyzfind.com/product/, retrieved August 2001.