System Desiderata for XML Databases

Airi Salminen* Frank Wm. Tompa

Department of Computer Science
University of Waterloo
Waterloo, ON
Canada
{asalminen,fwtompa}@db.uwaterloo.ca

Abstract

There has been much progress made towards defining query languages for structured document repositories, but emerging prototypes, products, and even proposed specifications too often assume overly simplistic data models and application needs. In this paper we explore the requirements for a general-purpose XML database management system, taking into account not only document structure and content, but also the presence of XML Schemas, Namespaces, XML entities, and URIs. Furthermore, the requirements accommodate applications that create, modify, and maintain complex units of data and metadata that co-exist with numerous versions and variants. Our discussion addresses issues arising from data modelling, data definition, data manipulation, and database administration.

1. Introduction

Two extreme positions can be heard regarding the role of XML in databases. One view is that XML is merely an encoding representation for exchanging data; therefore an XML database system is one that is able to import and export data or programs and to convert them to and from internal forms. The other extreme is that XML is merely an encoding representation for formatting documents; therefore an XML database system is one that is able to store such documents and to retrieve them on demand in order to present them to a browser. Our vision is for a database system that can manage XML data on behalf of applications that are far more demanding than either of these extremes.

An *XML database* is 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 semi-structured 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. Even for many applications in which XML is used as a transient data exchange format, there remains the need for persistent storage in XML form to preserve the communications between different parties in the form understood and agreed to by the parties.

David Maier proposed a list of language properties that are implied by the need to query collections of XML data [Mai98], and these have largely been adopted by the W3C XML Query Language Working Group [CFR01]. In this paper we propose further capabilities that must be provided by database management systems that purport to support XML databases and their applications.

The XML:DB initiative (http://www.xmldb.org) has defined three classes of XML database system, supporting native XML databases (designed to store and manipulate XML documents), XML-enabled databases (providing XML interfaces to other forms of stored data), and hybrid XML databases (accessible through XML and other interfaces), and there are a variety of database system prototypes and products in each of these classes (as documented in *The XML Cover Pages* [Cov01]). Unfortunately, emerging prototypes, products, and even proposed specifications too often assume overly simplistic data models and application needs.

A problem in applying traditional database technologies to the management of persistent XML data

**Proceedings of the 27th VLDB Conference,
Roma, Italy, 2001**

* On leave from the University of Jyväskylä.

lies in the special characteristics of such data, not typically found in others databases. XML documents are complex units of information, consisting of formal and natural languages, and often including multimedia entities. The units as a whole may be important legal or historical records. The production and processing of XML documents in an organization may create a complicated set of versions and variants, covering both basic data and metadata. Applications depend on systematic, controlled, and long lasting management technologies for all such collections.

These characteristics of document data also applied to SGML documents [Gol90], long before XML evolved. However XML imposes yet further demands:

- Closely related specifications that extend the capabilities specified in XML 1.0 [BPS00], such as XML Namespaces [BHL99] and XML Schema [Fal00, TBM00, BiM00], must be accommodated when developing XML database solutions, since they are expected to be widely used.
- Because references in XML documents refer to internet resources, general-purpose XML database systems must include internet resource management.

In the following sections we explore many capabilities needed to manage XML databases. After further elaborating on the special characteristics of XML data, the discussion addresses database system characteristics required for appropriate data definition, data manipulation, and database administration.

2. Modelling XML data

2.1 Modelling documents as well as enterprises

The complexity of XML-related data repositories and their management creates a special challenge for the underlying data model. 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 must 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 [CoW97].

For example, 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* [RTW96]: when are two documents or document parts the same? This question 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 defines the components of 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. The specification is explicit in stating which of these components are to be provided to an application by an XML processor and which text representations must be considered to be identical by all XML processors. It is expected that all XML applications will "view" documents in terms of these components, which we shall refer to as the *abstract structure*.

It is overly simplistic to assume that general-purpose XML database systems can be built using models that ignore the documents' structure, just as it is misguided to use models that ignore the enterprise data represented by those documents' content. In general, reconstruction of an original XML document from the data stored in an XML database must be possible.

2.2 Three-level architecture for XML data

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

Applying these principles to an XML database necessitates that the conceptual model incorporates not only all the entities 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 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. It is in such external models that the various notions of data equivalence can be encoded by defining mappings from the conceptual model that mask differences that are irrelevant to particular classes of applications.

Unfortunately, *complete* models for describing XML documents have not yet been developed, in spite of the fact that abstract structures for XML documents have been developed for four different W3C specifications: the Infoset model [CoT01], the XPath data model [ClD99], the DOM model [ABC98, LLW00], and the XML Query

	XML Information Set (Draft)	**XPath 1.0 Data Model**	**DOM 1.0 Level 2**	**XML Query Data Model (Draft)**
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 provide a formal definition for the information contained in the input to an XML Query processor, and a foundation for the XML Query Algebra
What is modelled?	XML document	XML document	XML (or HTML) document	Collection of XML documents or parts
Specification technique	Attaches descriptions (in terms of information items with a set of properties) to a set of syntactic items specified in XML 1.0	Informal description of the data structure and the string value of a set of node types	IDL description for a set of object interfaces	Functional description of constructors and accessors
# of node types in the tree structure	17	7	11	9
DTD or XML Schema validity required?	no	no	no	yes
DTD represented?	yes , but the order of declarations, content models of elements, grouping of attribute declarations, and ignored declarations are not represented	no	yes, representing its name, general entities and notations declared, identifier(s) for the external subset, and the internal subset as a string	no

Table 1. Characteristics of the four XML data models

Data Model [FeR01]. Brief descriptions of these models are given in the Appendix, and Table 1 summarizes their features. All four models describe an XML document as a tree structure, but there are differences in the trees and in the information available in the trees.

A common problem in the four models of Table 1 is the lack of information from the document type definition (DTD). Two of the models have a special node for the DTD, but only a subset of it is represented. None of the models includes explicit reference to an associated XML Schema. (Although the XML Query Data Model requires the existence of an XML Schema against which documents have been validated, the information in the schema is not accessible in the model, except to the extent that it is itself in a separate document.) Furthermore, each of the models in Table 1 also lacks some information from the content parts of XML documents.[1] Finally none of the models incorporate any semantics to support the need to relate document content to enterprise information.

Thus one challenge for the database community is to develop an acceptable data specification model (e.g., extensions to the E-R model) that can be used to capture the conceptual model of an XML database, including features to represent detailed document structure, enterprise information, and their interrelationships. Subsequent challenges are to develop mapping languages and techniques that can be used to define internal and external models and to design tools to support those who are charged with creating appropriate models for specific databases.

[1] The development of Canonical XML is intended to allow the comparison of two documents within a given application context [Boy00]. The specification describes a method for generating a text representation, the canonical form, of an XML document that accounts for permissible changes. However, the current candidate recommendation is based on the XPath Data Model, and thus there is a loss of information, including, for example, loss of the XML declaration, DTD, and all information about entities.

2.3 XML Schema volatility

The XML 1.0 specification states:

> "The function of the markup in an XML document is to describe its storage and logical structure and to associate attribute-value pairs with its logical structure. XML provides a mechanism, the document type declaration, to define constraints on the logical structure and to support the use of predefined storage units. An XML document is **valid** if it has an associated document type declaration and if the document complies with the constraints expressed in it."

The storage units may be text entities or other types of resources such as images, animations, or binary software applications. An XML document can be *well-formed* even when a DTD has not been declared. Alternatively, any of several more powerful constraining capabilities (such as XML Schema) may be applied [LeC00]. XML Schema, developed at W3C, is the most accepted of these, extending the definition capabilities of DTDs, by allowing the use of a variety of data types (e.g. boolean, float, int, date) and corresponding data validation and by providing mechanisms to define additional simple and compound data types.

When considering XML in the context of databases, it is tempting to treat a collection of documents as database instance values and associated DTDs (or XML Schemas [2]) as database schemas. However, there are two features that distinguish DTDs from database schemas:

- Documents may or may not be associated with a DTD, at the discretion of the application that created it.
- Any association with a DTD is represented within a document, not by the document's insertion into a class or collection.

As a result, in practice DTDs are created and modified at a rate that is more closely approximated by the volatility of database instances than by the stability of database schemas. For example, a relatively homogeneous collection of technical documents produced by a prolific organization typically includes many associated DTDs that reflect the evolution of requirements over the period of time during which the documents were created. Of particular note is that older documents are usually *not* updated to conform to newer versions of the constraints, either because the effort to change the documents is deemed to exceed any potential benefit or because there is a requirement to preserve the documents in their original form.

Elsewhere we have proposed that DTDs be stored and manipulated as part of object-relational database instances [BCD98], and we reiterate this as a desired characteristic for modelling DTDs in XML databases more generally. We note that neither Oracle8i nor the DB2 XML Extender represent the DTD associated with a document as part of the schema; however the DTD is stored in a separate file, which is unavailable to application programs as part of the instance that can be queried and manipulated by SQL.

Even if DTDs are stored as part of a database instance, they must still be used by the database system in the role usually reserved for the schema. As such, they provide a basis to formulate

- constraints for data input and thus for validity checking,
- meaningful queries, updates, and views,
- text transformations,
- query optimization strategies, and
- presentations of documents and of query results for subsequent browsing or other processing.

Thus a challenge for the database community is to determine how best to provide DTDs as part of the database instance but still to treat them as reliable constraint mechanisms for other parts of the instance. Because XML Schemas are themselves represented as XML documents, parallels can likely be drawn with system-defined "table tables" that provide read-only access to relational schema information maintained in the form of relational tables.

2.4 Versions and variants

The processing of structured documents often requires the use of multiple DTDs. As mentioned above, the production of large and complicated technical documents typically involves the need for different DTDs for the same material.

To illustrate the problem, we cite an analysis of the production of manuals for machines produced by a paper machine factory [KaT01]. Since each paper machine is unique in some of its aspects, the manual for each machine is also unique. Five different DTDs were required for producing the operation and maintenance manual for a single machine, covering various stages of document development. One of the DTDs included over 200 element names used for writing the content with the support of an SGML editor, a second was designed especially for publishing on paper, a third was HTML for publishing in the form readable by Web browsers, a fourth was a more generic DTD used as an intermediary between the detailed DTD used for creating the document and the ones used for publishing it, and the last was a DTD for metadata required to manage the transformations and files involved.

In an ideal world multiple presentations could be produced from a single data source as required, either through the use of style sheets or through automated transformations. In practice, however, publishing a document often involves human intervention to "clean

[2] For simplicity and to avoid confusion between XML schemas and database schemas, we henceforth use the term *DTD* to refer generically to any form of XML schema or document type definition facility.

up" the output of an initial transformation. Furthermore, each published form must be supported by a corresponding DTD.

As an additional complication, at the time of the analysis of the paper machine factory, the detailed DTD used for data entry was in its fourth version, and there were instances in the document repository created with respect to each of the different versions. In this factory, documenting a new machine usually proceeds by adapting the manual of an older machine that has the closest specifications to the new one, even if some aspects of documentation practices have since evolved. Therefore older documents conforming to older versions of the schema must be accessible to users. Because of ongoing requirements to access the contents of a manual, the various forms of the data are not merely historical versions; information from all forms must remain accessible concurrently.

Other case studies confirm that publishing documents often requires multiple DTDs that represent various *versions* developed over time as well as several *variants* covering different phases of document production [Fah99, SLT00]. Furthermore, these studies confirm that the data content must be preserved in its variant forms corresponding to different DTDs.

Thus, schema management is even more problematic in XML databases than in traditional environments. Not only do changes in business requirements cause the design of new schemas, but also publication requirements may impose the need for multiple schemas. Not only does the integration of data from different partners cause the need to accommodate multiple schemas, but also access to multiple pieces of data from a single source imposes the same requirement. Not only are different schemas created over time, but also all past schemas must be maintained. Not only are different schemas needed to access different pieces of information about an enterprise, but also several schemas are needed to access a single piece of enterprise information in each of its various forms.

3. Desired DDL characteristics

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. The purpose of the XML schema languages is to allow the validation of a given well-formed XML document against the schema. Below we discuss the DDL characteristics needed, in addition to the constraining capabilities, to instantiate an XML database.

3.1 Data collections

A possible approach for defining an XML database structure is as a single XML document, similarly to defining a relational database as a universal relation. With this approach any of the schema definition languages could be used. However, if the database were restricted to being one universal document (and all applications were defined in terms of access to such a single document), then any collection of views of the database that are to be made accessible to an application must be interpreted as if it were a single document and all intermediate results would need to be (logically) inserted somewhere in the document in order to be accessible to subsequent processing.

As a general solution, therefore, 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. The W3C proposal for the XML Query Data Model specifies that a data instance is logically an ordered or unordered collection of complete XML documents or document parts. To be able to apply a query language to such a database, the DDL should offer the capability to define such collections. In addition, because document management systems usually organize documents into a hierarchy of folders, the capability to declare that a particular XML document represents a folder hierarchy may ease interoperability with external sources.

3.2 DTD collections

As mentioned in Section 2, structured document management often requires a versatile collection of DTDs. Therefore the DDL of an XML database system should support the definition of multiple DTDs, 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 must provide capabilities to manage different DTD versions and variants and do so as new DTDs are created and existing ones are updated.

The need for several DTDs for the same material and for different DTD 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 of the diversity of presentation media, it is clear that the need for managing rich collections of DTDs 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 continuingly changing international and industry-level standards, DTDs will be "alive," and the database system must support the management of their evolution.

3.3 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 the logical structure of a database 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 the 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 storage decisions made at the internal level of an XML database to application programs.

One approach for general-purpose XML database systems is to separate the naming structure of URIs from the file-naming mechanism used by the database system. That is, the database system must include an internal mapping from URIs to files (or equivalent storage units) so that the names used as resource indicators can remain independent of the names used for addressing units of storage. The DDL must then also provide a capability to define the URIs used for naming entities in the database and for mapping such URIs to the structures used for storing the collection of entities in the database.

3.4 Multiple levels of validity

Maier's list of desired characteristics includes two concerning XML schemas: the query language should be usable on XML data when there is no schema known in advance, and when schemas are available it should be possible to judge whether a query is correctly formed with respect to the schemas and to derive a schema for the query's result. In light of these characteristics, 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 DTD provided by a user or application,
- valid XML documents, each associated with a DTD 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.

Recall that our use of the term *DTD* encompasses any mechanism to define an XML schema. In this context, therefore, the DDL must 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 datatype 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 datatypes used in the schema and ensure that the documents they create conform to the richer constraint mechanisms. Thus, the extent to which rich datatypes 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.

3.5 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. XML Schema as a schema language allows the use of namespace names in schemas. 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 must 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).

3.6 Document indexing

Some relational data definition languages allow the definition of indexes. Such indexes affect performance efficiency but not query semantics. Traditional document indexing, however, implicitly affects the set of documents that are retrieved in response to various queries.

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. Some full-text indexing systems also apply morphological and lexical analyses to the documents' contents before extracting index terms (thus, for example, converting the presence of the word "data banks" in a document to the term "database" in the index). The choice of words and phrases for the index determines which query terms will select which documents. As a result, a document index is much more than a performance-enhancing device.

It is important to realize that 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. Among other aspects, the stop words (i.e., words considered to carry no important information) left out from an index vary from language to language and from one collection to another even when all documents share a common language.

The nature of a document index, therefore, is closer to that of instance data than is true of traditional database indexes. The DDL for an XML database should allow application programs to specify the rules for indexing documents (and the DML should provide facilities for querying the indexing rules and for choosing which indexes to use to execute a given query). 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. An open problem is to develop general techniques that will allow database designers to specify precisely how document indexing is to be applied to structured documents that include arbitrary character data together with values taken from other types (such as dates, numbers, and URIs).

3.7 Metadata

In database management systems, some metadata, such as that stored in a data dictionary, is created by the system as a result of compiling DDL statements; other metadata, such as the time of last update, results from executing DML statements. Often metadata is not accessed directly by applications, but rather it is consulted by the system before instance data is accessed and perhaps updated thereafter. In document repositories, metadata is data about the documents, and the metadata is an important means for end users to manage a repository. The schemas that describe document structures and data types, as well as the indexes describing the content of documents, can be regarded as two common forms of metadata.

In an XML database, the system should provide all metadata in XML form and allow querying and manipulating through the DML (subject to permissions). The DDL should include not only a capability to define metadata, but also a capability to define the vocabularies used for the metadata. Suitable features are available in RDF, a W3C Recommendation for describing metadata for Web resources in XML format [LaS99], and the RDF Schema language, a W3C Candidate Recommendation for defining metadata vocabularies [BrG00].

One way to manage inter-document references is to separate them from documents and to express them as metadata that is stored in separate documents. Such "link documents" can be defined as *linkbases*, following the recommendation of the XLink language [DMO00]. Declaring a link document as a separate object in an XML database also simplifies the definition of views in which a subset of links can be excluded even if the resources they connect are included in the view. The DDL must support the declaration of linkbases and their relationship to other data.

4. Desired DML characteristics

The development of XML query languages has been based on extensive discussion about the desired characteristics of such languages [Mai98, CFM01]. We will not repeat all those characteristics here; instead we discuss those characteristics 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 [GMW99], Quilt [CRF00], XQuery [CRF00], and the XML Query Algebra [FFM01] all omit some of the data from XML documents. The DTD, 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.

XQuery and the XML Query Algebra are based on the XML Query Data Model, which assumes that the data is validated against an XML Schema. If there is no schema explicitly associated with a document, then a default schema is used. They incorporate datatypes as defined in the XML Schema [BiM00], but the schema associated with a document is not accessible to an application (unless the XML form of the schema is explicitly stored by the application as XML data, and the association of the document to the schema is preserved by the application). As discussed in Section 2, an XML database having a rich collection of DTDs must be able to access DTD information.

Also following Section 2, it is important to be able to test the equivalence of documents or their parts, and also test the equivalence of DTDs [RTW96]. However, equivalence comparison capabilities are limited in the proposed query languages. Not only do the models not encompass all data from XML documents, but also there are only limited ways for testing equivalence. The XML Query Data Model includes one equality operator to test node identity and another to test equality of node values. Semantics for the equality of node values is not yet defined, but we note that in an XML database there is a need for multiple forms of value equivalences. For example, before adding a new document to the database we might want to find out if the same document is already in the database. Various applications may wish interpret the "same" in different ways: some of which may correspond to the model's meaning and some requiring testing equivalence with respect to some view that may ignore differences in various components of the document structure or enterprise data. A query language that supports application-dependent choices for testing equivalence, for example by ensuring that equivalence can be tested with respect to an external view, would benefit the management of a large collection of XML data and DTDs.

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 [TaT01].

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 must 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 [ABC00] together with XSLT transformation descriptions [Cla99] and for dynamic production of external presentations.

Integration support. An XML database system must 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 2, we discussed the problem of multiple DTDs in XML document production environments. Because changes in DTDs are quite common, there is often a need to transform existing data to correspond to a new DTD.

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 far more appropriate in an XML database system to base view definitions on a transformation language.

4.3 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 must 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. For example, during the development of an operation and maintenance manual for a machine, the content production may include:

- an engineer inserting pieces by writing directly to the repository or by uploading from a word processor,
- a technical writer inserting pieces by modifying an earlier "model" manual,
- a technical writer updating a preliminary text written by an engineer,
- an editor updating a part to polish the text,
- a translator to convert the text to another language, and
- an engineer updating a part to correct the text before approving it.

Such processes rely on XML editors and support for workflow management, which should be integrated with XML database systems, as is common in content management systems such as Chrystal's Astoria (http://www.chrystal.com), Interleaf's Information Manager (http://www.interleaf.com), and Open Text's Livelink (http://www.opentext.com).

An XML database may contain various forms of reference: entity references, intra-document 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 must be supported.

A major concern in updating traditional databases has been transaction management. Database systems include capabilities with their DMLs for applications to specify the scope of each transaction. In XML database systems, an XML document is a natural unit for specifying a transaction, and thus the DML should include a mode 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 common to various business sectors are being continuously introduced (see, for example, [Cov01]).

5. Desired administrative capabilities

One of the most characteristic features of the data on the Web is the lack of centralized control. XML emerged in this environment to support information distribution and data integration. To be able to manage XML information assets in the intranets and extranets of organizations, however, systematic ways to control the information and its accessibility is needed. This requires an XML database system to provide support for the specification of administrative and user roles, specification of authorization for data, and specification of the metadata needed to control, use, and preserve the information.

5.1 Administrative roles

Database administration tasks are often divided into two roles: data administrator and database administrator. The *data administrator* is a person making strategic and policy decisions regarding the data of the organization. An XML data administrator in an extranet environment must also negotiate with other parties outside the enterprise to establish common rules for sharing and distributing data over the extranet and common interpretations for the metadata vocabularies. These tasks impose further reliance on effective management of metadata. The *database administrator* is a technical person who provides necessary technical knowledge and support for implementing the decisions of the data administrators. Tasks of the database administrator include defining the database schemas and corresponding security and integrity checks, defining backup and recovery procedures, liaising with users, monitoring performance, and responding to changing requirements. All of these functions are needed to manage an XML database.

Legal and historic records stored as documents must be accessible in their original form for decades. Thus, for an XML database it is useful to specify a third administrative role: the document administrator. The *document administrator* is responsible for record-keeping and archiving, as is typically provided by records managers in many organizations. A central task of the document administrator is to decide, in collaboration with the data administrator and the database administrator, which strategies to adopt for preserving digital documents throughout changes to hardware and software, to design the metadata needed to assure full evidentiality and long-term accessibility of the information assets, and to plan the responsibilities to update the metadata [BeS01, Mur98, HeR98, CGM00].

5.2 User roles and access rights

One of the tasks of a database administrator has been granting of authorization to data. An XML database including both data and metadata for a variety of purposes and diverse users needs role-based access control [SCF96]. 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 in very large database environments with very many users, each shifting among many possible roles.

6. Conclusion

Database systems were designed to manage large bodies of information about one enterprise at a time and to provide integrity for the information despite many users sharing the data and changes in technology. More recently XML emerged as a universal metalanguage to be used as a common format for information in various application areas. 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. We addressed some of the major requirements for appropriate data definition, data manipulation, and database administration and demonstrated the complexity of the area.

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. We look forward to innovative solutions being developed to address the problems identified, including problems of equivalence, versions, and variants for XML data.

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 Science and Engineering Research Council of Canada, and Bell University Labs.

References

[ABC00] S. Adler, A. Berglund, J. Caruso, S. Deach, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, and S. Zilles (Eds.). Extensible Stylesheet Language (XSL) Version 1.0, W3C Candidate Recommendation 21 November 2000. http://www.w3.org/TR/2000/CR-xsl-20001121/.

[ABC98] V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson and L. Wood (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/.

[BeS01] D. Bearman and K. Sochats. Metadata requirements for evidence, http://www.lis.pitt.edu/~nhprc/BACartic.html, retrieved Feb. 24, 2001.

[BiM00] B.V. Biron and A. Malhotra (Eds.). XML Schema Part 2: Datatypes, W3C Candidate Recommendation 24 October 2000. http://www.w3.org/TR/2000/CR-xmlschema-2-20001024/.

[Boy00] J. Boyer (Ed.). Canonical XML Version 1.0, W3C Candidate Recommendation, 12 December 2000. http://www.w3.org/TR/2000/CR-xml-c14n-20001212/.

[BHL99] T. Bray, D., Hollander, and A. Layman (Eds.). Namespaces in XML, World Wide Web Consortium, 14 January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/.

[BPS00] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler (Eds.). Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000. http://www.w3.org/TR/2000/REC-xml-20001006/.

[BHL99] T. Bray, D. Hollander, and A. Layman (Eds.). Namespaces in XML, World Wide Web Consortium 14 January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/.

[BrG00] D. Brickley and R.V. Guha (Eds.). Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation, 27 March 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[BCD98] L. J. Brown, M. P. Consens, I. J. Davis, C. R. Palmer, and F. W. Tompa. A structured text ADT for object-relational databases. *Theory and Practice of Object Systems* 4(4) (1998), 227-244.

[CFM01] D. Chamberlin, P. Fankhauser, M. Marchiori, and J. Robie. XML Query Requirements, W3C Working Draft 15 February 2001. http://www.w3.org/TR/2001/WD-xmlquery-req-20010215/.

[CFR01] D. Chamberlin, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu (Eds.). XQuery: A Query Language for XML, W3C Working Draft, 15 February 2001. http://www.w3.org/TR/2001/WD-xquery-20010215/.

[CRF00] D. Chamberlin, J. Robie, and D. Florescu. Quilt: An XML Query Language for heterogeneous data sources. *WebDB (Informal Proceedings)* 2000, pp. 53-62.

[Cla99] J. Clark (Ed.). XSL Transformations (XSLT) Version 1.0, W3C Recommendation, 16 November 1999. http://www.w3.org/TR/1999/REC-xslt-19991116.

[ClD99] J. Clark and S. DeRose. XML Path Language (XPath),Version 1.0 W3C Recommendation 16 November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116

[CoW97] J. Coleman and D. Willis. SGML as a framework for digital preservation and access, The Commission on Preservation & Access, July 1997.

[Cov01] R. Cover (Ed.). The XML Cover Pages. http://www.oasis-open.org/cover/, retrieved Feb. 24, 2001.

[CoT01] J. Cowan and R. Tobin (Eds.). XML Information Set, W3C Working Draft 2 February 2001. http://www.w3.org/TR/xml-infoset/.

[CGM00] A. Crespo and H. Garcia-Molina. Modeling archival repositories. *Proc. Fourth European Conf. on Digital Libraries* (*ECDL*). September, 2000.

[DMO00] S. DeRose, E. Maler, and D. Orchard (Eds.). XML Linking Language (XLink) Version 1.0, W3C Proposed Recommendation, 20 December 2000. http://www.w3.org/TR/2000/PR-xlink-20001220/.

[Fah99] S. Fahrenholz. SGML for electronic publishing at a technical society – Expectations meets reality. *Markup Languages: Theory and Practice* 1(2) (Spring 1999), 1-30.

[Fal00] D.C. Fallside (Ed.). XML Schema Part 0: Primer, W3C Candidate Recommendation 24 October 2000. http://www.w3.org/TR/2000/CR-xmlschema-0-20001024/.

[FFM01] P. Fankhauser, M. Fernández, A. Malhotra, M. Rys, J. Siméon, and P. Wadler (Eds.). The XML Query Algebra, W3C Working Draft, 15 February 2001. http://www.w3.org/TR/2001/WD-query-algebra-20010215/.

[FeR01] M. Fernández and J. Robie (Eds.). XML Query Data Model, W3C Working Draft 15 February 2001. http://www.w3.org/TR/2001/WD-query-data-model-20010215/.

[Gol90] C.F. Goldfarb. *The SGML Handbook*. Oxford University Press, Oxford, UK (1990).

[GMW99] R. Goldman, J. McHugh, and J. Widow. From semistructured data to XML: Migrating the Lore model and query language. *WebDB (Informal Proceedings)* 1999, pp. 25-30.

[HeR98] A.R. Heminger and S.B. Robertson. Digital Rosetta Stone: A conceptual model for maintaining long-term access to digital documents. *Proc. 31st Annual Hawaii International Conference on System Sciences* (1998).

[KaT01] A. Karjalainen and P. Tyrväinen. Defining genres and their features for studying information reuse: Preliminary findings. To appear in *Proc. of IRMA 2001, Information Resources Management Association International Conference*, Toronto, May 20 – 23, 2001.

[LaS99] O. Lassila and R. R. Swick (Eds.). Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[LeC00] D. Lee and W. W. Chu, Comparative analysis of six XML schema languages. *ACM SIGMOD Record* 29 (3) (2000), 76-87.

[LLW00] A. Le Hors, P. Le Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion, and S. Byrne (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

[Mai98] D. Maier. Database desiderata for an XML query language. QL'98 – The Query Language Workshop, W3C, Boston (December 1998).

[Mur98] L. D. Murphy. Digital document metadata in organizations: roles, analytical approaches, and future research directions. *Proc. 31st Annual Hawaii International Conf. on System Sciences* (1998), 267-276.

[RTW96] D. R. Raymond, F. W. Tompa, and D. Wood. From data representation to data model: meta-semantic issues in the evolution of SGML. *Computer Standards & Interfaces* 18 (1996), 25-36.

[SLT00] A. Salminen, V. Lyytikäinen, P. Tiitinen, and O. Mustajärvi. SGML for E-Governance: The case of the Finnish Parliament. *Proc. 11th International Workshop on Data and Expert Systems Applications* (2000), 349-353.

[SCF96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer* (February 1996), 38-47.

[TaT01] X. Tang and F. W. Tompa. Approaches towards developing transformation languages for structured documents. In preparation, February 2001.

[TBM00] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn (Eds.). XML Schema Part 1: Structures, W3C Candidate Recommendation 24 October 2000. http://www.w3.org/TR/2000/CR-xmlschema-1-0001024/.

Appendix

The XML Information Set (*Infoset*) [CoT01] is the most central of the four models even though it is still a work in progress. The specification has been in the Working Draft phase since May 1999. It is intended to provide a set of definitions for use in other specifications that need to refer to the useful information in an XML document. The specification draft describes the information in an XML document as a set consisting of information items of various types. For each type, a set of properties is defined. The properties describe the information accessible from an information item of the corresponding type.

The XPath Data Model [ClD99] is included in the XML Path Language (XPath) specification. The model defines an XML document as a tree containing nodes, describes the nodes, their relationships, and a string value for each node. The purpose of the XPath Specification is to offer a common syntax and semantics for addressing parts of an XML document in other specifications, initially for XSLT and XPointer specifications. There are no formal dependencies between the XPath and Infoset models. The XPath specification, however, describes in a non-normative appendix how the nodes in the XPath data model can be derived from the information items provided by the XML Information Set.

The Document Object Model (DOM) [ABC98, LLW00] is an application program interface for valid HTML and well-formed XML documents. Thus it is not itself a data model, but instead it defines an object model for documents: the logical structure of documents and the way a document is accessed and manipulated. DOM is intended to provide a standard programming interface that can be used in a wide variety of environments and applications, with any programming language. Implementations are provided for Java and ECMAScript. The DOM Level 2 specification consists of five parts: Core, Views, Events, Style, and Traversal and Range. The underlying data model for XML documents is included in the DOM Level 2 Core Specification [LLW00], which builds on the DOM Level 1 Specification [ABC98]. The DOM Level 2 Specification defines structural isomorphism between two DOM implementations with respect to the XML Information Set: for a given document the implementations have to create the "same structure model" in accordance with the XML Infoset. However, the precise meaning of this requirement has not been specified.

The XML Query Data Model [FeR01] is part of the W3C activity for specifying an XML query language. It is a work in progress and serves as the foundation for the XML Query Algebra [FFM01]. The Query Data Model is intended to define formally the information contained in the input to an XML Query processor so that a query can be evaluated against a database instance. An instance of the Query Data Model is logically derived from an instance of the XML Infoset after validation against an XML Schema. Thus it is based on the XML Information Set, but it requires support for XML Schema types, for representing collections of documents and collections of simple and complex values, and for references within an XML document and from one XML document to another. Such an instance is called a post-schema-validated information set, or "PSV Infoset."