

Preserving State Government Digital Information

Minnesota Historical Society

XML Native Databases and Legislative Documents: A White Paper

Abstract

XML native databases can improve access to and use of text-based XML encoded information by providing full-text indexing and native storage, and they can be even more powerful when used with other XML standards and tools such as XQuery and XForms.

Any comments, corrections, or recommendations may be sent to the project team, care of:

Nancy Hoffman
Project Analyst
Minnesota Historical Society
nancy.hoffman@mnhs.org / (651) 259-3367

The Problem

Electronic legislative records present a classic information management problem – how can text-based information be as easy to access and use as data stored in traditional database systems?¹ A common solution has been to take data out of a document and place it into defined fields in a traditional database management system - a process called “shredding.” Removing and storing data elements in this way is not only difficult and time consuming, but more importantly, the database cannot take full advantage of the value of the text because the actual documents are stored in a separate file system or are completely obscure when stored as objects (called binary large object types or BLOBs) inside the database.²

XML Changes Everything

The adoption of XML bill drafting systems for legislation and increasing use of XML to structure related information opens new possibilities afforded by XML-related tools and

¹ http://www.cs.ubc.ca/grads/resources/thesis/May04/Fengdong_Du.pdf [accessed 12/18/2009]

² <http://www.ibm.com/developerworks/xml/library/x-xml2008prevw.html> [accessed 12/22/2009]

standards that have developed over the ten-plus years since its introduction. XML changes how we can think of text-based information. XML specifies a system of user-generated tags organized in a hierarchical or tree-like fashion. The W3C Schools' introduction to XML points out that, "XML is nothing special. It is just plain text. Software that can handle plain text can also handle XML. However, XML-aware applications can handle the XML tags specially."³

An aspect of XML that sets it apart from virtually all other electronic text encoding is that, as writer and developer Gavin Powell points out, "Essentially, every XML document is a form of an XML database. Why? Because every XML document contains both data [text] and metadata [tags]."⁴ XML itself is not very efficient as a database though because XML is verbose; it creates long strings that can be time consuming for XML-aware applications to read (a process called parsing).

XML native databases improve on this basic functionality by using indexing, which improves the speed of search. Indexing creates simply binary (ones and zeros) lists that can be searched instead of requiring the application to traverse the entire document for each query, greatly improving the response time. Indexing eliminates the need to "shred" document information to get search and retrieval functional equivalent to a traditional database while keeping all elements of a document intact.

XML and Databases

Use of XML tends to fall into two general categories, data-centric and document-centric.⁵ For data-centric use, XML is just one of many tools available to describe a well-defined data set. Likewise, many database options are available for storing and using this type of information. On the other hand, electronic encoding options for the large blocks of text found in most legislative documents have not offered good data management choices because this kind of text does not contain well-defined data elements. Use of document-centric XML has steadily increased because the flexible mark-up system of XML is well suited to describing text-based information. Prior to the introduction of XML databases however, management of document-centric XML usually did not include the kind of search and reporting functionality afforded by traditional databases.⁶

³ http://www.w3schools.com/xml/xml_what_is.asp [accessed 12/22/2009]

⁴ "Beginning XML databases," Gavin Powell, John Wiley & Sons, 2006, p 384

⁵ <http://www.rpbouret.com/xml/XMLAndDatabases.htm#datavdocs> [[accessed 12/22/2009]

⁶ <http://www.openxmlcommunity.org/documents/IDC%20Document%20Adoptions%20White%20Paper.pdf> [accessed 12/22/2009]

For example, Elliotte Rusty Harold, a programmer and author of several books on XML, noted in 2007, “fields like publishing have not even had a database backend. It’s not that they didn’t need one. It’s just that the databases of the day couldn’t handle their needs, so content was simply stored in Word files in a file system.” He predicts that these fields will be revolutionized by the use of XML databases.⁷ XML databases fall into one of two basic types: XML-enabled and XML native. Each of these types tends to work better with one of the XML use categories than the other; data-centric XML with XML-enabled databases and document-centric with XML native databases.

XML Enabled Databases

Databases that can accept XML input and can produce XML output, but do not store data in XML form, fall into a category called XML-enabled.⁸ The increasing use of XML has led traditional database vendors, such as Oracle, to add the ability to handle XML encoded data. Users with data-centric XML, often those who want to take advantage of XML data transmission tools, have found XML-enabled databases useful. Fundamentally, XML is just another tool supported by these databases and they do not seek to utilize the XML data model or employ XML-related technologies.

XML-enabled databases can introduce errors when translating data into and out of XML format. Details like element order, processing instructions, comments, and white space that do not contribute to defining the data in the document are discarded during the transformation.⁹ This can be particularly problematic when dealing with highly formatted text-based material like legislative documents.

XML Native Databases

XML native databases use the XML document as the basic unit of storage and keep all components of the XML model intact. They do not require shredding (placing data into separate fields) to utilize the meaning of the tags, and most create a full-text index when files are added to the database.¹⁰ Documentation on the computer code sharing website Sourceforge explains that

⁷ <http://cafe.elharo.com/xml/the-state-of-native-xml-databases/> [accessed 12/22/2009]

⁸ http://en.wikipedia.org/wiki/XML_database [accessed 12/22/2009]

⁹ www.ibm.com/developerworks/xml/library/x-mxd4.html [accessed 12/22/2009]

¹⁰ <http://www.xml.com/pub/a/2001/10/31/nativexml.db.html> [accessed 12/22/2009]

an XML native database, “Defines a (logical) model for an XML document — as opposed to the data in that document — and stores and retrieves documents according to that model.”¹¹

The ability to group documents together in collections is another feature supported by XML native databases that implement XQuery (see below). Collections can be searched as a group, but searches can also be limited to single document files, made across collections, or across the entire database. This allows a variety of XML encoded information to be aggregated while still providing the ability to perform targeted searches.

Because XML native databases preserve document formats while offering improved access to and manipulation of the information in the documents, they are a good fit for XML-encoded legislative documents.

Advantages of XML Native Databases

One reason XML native databases serve as such effective tools for managing text-based XML data is their ability to create indexes based on the XML data model. Indexes improve the speed at which a database functions because the size of the index is a fraction of the size of the full file. This allows them to process queries on very large documents and very large numbers of documents efficiently.¹²

In a paper written in 2005, three Australian researchers explained indexing differences in various XML native databases, “Searching a native XML database is handled in different ways, depending on the vendor of the database. Some native XML databases require the user to select the elements or attributes to be indexed. This information is then used to build an index that the searching mechanism can use to faster locate matching documents. Other native XML databases simply index all elements in a document, which obviously causes the need for more storage space. Indexing all elements in native XML databases, however, has more sense compared with indexing of all columns in a relational database.”¹³ Regardless of the mechanism employed, indexes offer a significant improvement over the speed required to parse an entire document or file during a search and make XML native databases a far more effective way to retrieve data from XML files than from within a traditional file system.

¹¹ <http://xmldb-org.sourceforge.net/faqs.html> [accessed 12/22/2009]

¹² <http://gilbane.com/whitepapers.pl?view=28> [accessed 12/22/2009]

¹³ http://espace.uq.edu.au/eserv/UQ:8997/n60_stantic.pdf [accessed 12/22/2009]

Ronald Bourret has written extensively on XML and XML databases. He outlines a number of specific advantages XML native databases can offer. His list includes:

- 1) *Storing and querying document-centric XML.* Functions which involve managing, finding, retrieving information from documents, and allow reuse of content are facilitated as a result of the full-text indexing feature of XML native databases. The query syntax used by XML native databases allows more complex structured queries of markup, text, or both.
- 2) *Working with very large documents.* Large documents are difficult to query due to the time it takes to parse them. XML native databases solve this problem by parsing and indexing documents when they are inserted.
- 3) *Integrating data.* Bringing together data that has different kinds of information and different ways of representing information is one of the biggest problems faced by data integration projects. Resolving this problem primarily involves making some decisions about what information represents the same or similar concepts. These decisions can be implemented by converting all the data to a single standard format when it is brought together, but because XML native databases allow XML-aware full-text searches, these differences can be ignored in exchange for less precision in the query construction. This feature can facilitate combining a wide variety of non-XML information, because many tools can convert other file formats into XML.
- 4) *Handling semi-structured data.* Data integration and long-term data management can face an additional layer of complexity when changes are made to the structure and definition of the data. Rigid structures found in relational databases cannot easily adapt to such changes. They require updates to old data to make it compatible with the new structure. This is not possible for some types of information. Changing a contract, for example, invalidates it. Queries over unchanged fields continue to work in the native XML database, but fail in the relational database because they cannot include the new data types.

Another aspect of semi-structured data may be that some data types occur infrequently. XML represents infrequently occurring data types — sparse data — efficiently. Relational databases do not handle sparse data efficiently: the choices are a single table with lots of nulls, which wastes space, or many sparsely populated tables, which are difficult and time consuming to join. XML-aware and full-text searches make this kind of information easy to find in an XML native database.

- 5) *Running websites.* Native XML databases can be used to build web sites because data stored in the database as XML can easily be transformed into XHTML with standard tools such as XQuery (see below).¹⁴

Disadvantages of XML Native Databases

Because designers of XML databases intended them to work well with hierarchical data and text, they generally do not offer features that can in any way improve access to or use of highly-structured non-hierarchical data. For example, Kimbro Staken, a developer writing for O'Reilly, has pointed out that, "An [XML native database] can store any type of XML data, but probably isn't the best tool to use for something like an accounting system where the data is very well-defined and rigid."¹⁵

XML based technologies such as XML native databases require programmers who are proficient in working with XML and related tools. This means that anyone working with text-base information not in XML format would have to take into account the learning curve for this technology in order to benefit from the solutions it offers. This would not pose a significant barrier though to anyone already using XML and XML tools for information management.

XML native databases do not have the history of development behind them that traditional relational databases have. While many are mature enough to have supported commercial applications for a number of years,¹⁶ there are not as many implementations of XML native systems as of relational or object-oriented databases.

Emerging XML-related standards mean that varying implementation of these functions across different XML native databases may hinder interoperability and result in changes when the standards are finalized. Some of the standards still in development as listed in a paper published by XML native database vendor Documentum include:

Versioning. "As one version is updated, the need to decide whether related versions also need to be updated, or if the relationship needs to be continued, can be supported by the database application. Different XML databases manage these relationships and decisions to varying degrees."

¹⁴ Paraphrased from "Going native: Use cases for native XML databases
<http://www.rpbouret.com/xml/UseCases.htm> [accessed 12/22/2009]

¹⁵ <http://www.xml.com/pub/a/2001/10/31/nativexml.db.html> [accessed 12/22/2009]

¹⁶ <http://www.scribd.com/doc/14776004/XML-Impacting-the-Enterprise> [accessed 12/22/2009]

Transaction management. Document-level locking can limit concurrency. This is not particularly problematic for systems that do not require frequent access to document editing, but can pose significant workflow management issues for systems where more than one person may need to access and change a document at the same time.¹⁷

The Documentum writers go on to say, “As users begin using XML databases, they will be faced with some design and development issues that are not currently addressed by a consistent and open standard. In these situations, they will have to rely on proprietary capabilities of the databases they deploy. But, different databases may support the functionality in wildly differing ways, which reduces the interoperability of systems and increases development and integration efforts.”¹⁸

XML Based Standards

Standards ensure that different systems and applications can work together. XML is not just a World Wide Web Consortium (W3C) standard for text encoding; it also defines a data model based upon the properties of XML’s tree-like structure. This model is called the Infoset. Clearly defining the elements and properties of this model has facilitated the development of a number of XML-related standards such as XPath¹⁹ (see below). Using XML based standards together takes fullest advantage of XML’s superior ability to support access to and reuse of text based information.

XML Native Databases and XRX

The use of XForms, REST, and XQuery in combination, called XRX, draws upon a set of XML standards that, when used together with an XML native databases, can produce information delivery services that are simple to develop and that maintain the integrity of XML encoded data.²⁰ The XForms and XQuery components are tools for manipulating XML data. REST names the type of technical architecture employed. Enterprise data architect Dan McCreary characterizes it this way, “Because XRX uses a single model for data (XML) it avoids the translation complexity of other architectures.”²¹ McCreary compares the process used by a typical web application to language translation, “Select any passage from any book and enter it into a translation program such as Google Translate. Perform a translation from English to

¹⁷ <http://xml.coverpages.org/Gilbane-EMC-WaldtTrippe-XML-andDatabases.pdf> [accessed 12/22/2009]

¹⁸ Ibid.

¹⁹ <http://www.w3.org/TR/xpath20/> and XQuery <http://www.w3.org/TR/xquery> [accessed 12/22/2009]

²⁰ [http://en.wikipedia.org/wiki/XRX_\(web_application_architecture\)](http://en.wikipedia.org/wiki/XRX_(web_application_architecture)) [accessed 12/22/2009]

²¹ <http://knol.google.com/k/dan-mccreary/xrx/2urpkmpv14a33/2#> [accessed 12/22/2009]

Spanish and from Spanish to German. Then reverse the process by translating the German to Spanish and the Spanish back to the original English. The result will have little resemblance to the original text and will require manual cleanup.”²² XRX systems do not introduce these kinds of errors because the data never needs to change formats.

REST

REST stands for Representational State Transfer. While not an XML standard, it is the design principle upon which the internet is built. Exchanging information between computers via the internet’s Hypertext Transfer Protocol (HTTP) requires only two things: the identifier of a resource, and the type of action to be taken, and there are just four: ‘Get’ (display or show a resource representation), ‘Post’ (add or create a new resource), ‘Put’ (update a resource), and ‘Delete’ (remove a resource). A system that employs this type of architecture, called RESTful, can take advantage of simple interfaces using open web standards.²³

This is in marked contrast to more common procedural programming approaches that tend to be proprietary, vary from one implementation to the next, and do not support stable resource references, creating systems that can hinder access to information rather than make more available, as intended.

The REST architectural model offers a number of valuable characteristics for systems that seek to provide a flexible information platform. RESTful systems keep copies of resources in memory, a process known as caching. Caching improves response time for requests because the system can quickly find and use the copy when it’s available rather than spending time searching for the original resource. Each request can be handled independently of every other, allowing the use of multiple servers. This not only improves the response time, but also allows the system to be highly scalable because actions can be “farmed out” across any number of processors.

As they scale, RESTful systems have the capacity to accommodate evolution of document types such as HTML without breaking backwards- or forwards-compatibility and have the ability to add support for new content types as they are defined without dropping or reducing support for older content types.²⁴

²² http://www.oreillynet.com/xml/blog/2008/05/xrx_a_simple_elegant_disruptiv_1.html [accessed 12/22/2009]

²³ <http://knol.google.com/k/dan-mccreary/xrx/2urpkmpv14a33/2#> [accessed 12/22/2009]

²⁴ http://en.wikipedia.org/wiki/Talk:Representational_State_Transfer [accessed 12/22/2009]

A disadvantage of RESTful systems is that security has depended upon secure internet protocols like https. New tools, however, allow role-based permissions to be set-up, improving control over access and security.²⁵

XQuery and XForms

XQuery²⁶ and XForms²⁷ are both W3C standards based upon XPath,²⁸ the XML language for defining paths to and selecting nodes²⁹ in an XML tree.

XQuery

The XQuery standard resulted from a W3C initiative to develop a query language for XML that combined the best characteristics of several other query languages. XQuery extends the basic selection tools of XPath. As Ronald Bourret points out, XML-based queries are, “fundamentally different from those that return whole documents to be read or modified. Instead they answer questions, create reports, or construct entirely new documents.”³⁰ This is possible because they work with only the section of the document or documents containing the information of interest. XQuery can be used to filter, sort, and divide existing XML content.³¹

XML can produce results in other formats such as HTML or plain text³² and it can be relatively easy to generate web pages with XQuery, for example, by specifying XHTML as the output format.³³ XQuery can also be used to support interoperable machine-to-machine interaction over a network³⁴ when XML query results sets are exposed on the web.^{35 36} Similarly, XQuery can be used to combine data from multiple independent sources. System architect Michiel van

²⁵ <http://www.oreillynet.com/pub/e/1225> [accessed 12/22/2009]

²⁶ <http://www.w3.org/TR/xquery/> [accessed 12/22/2009]

²⁷ <http://www.w3.org/TR/xforms/> [accessed 12/22/2009]

²⁸ <http://www.w3.org/TR/xpath> [accessed 12/22/2009]

²⁹ A node is an abstract basic unit used to build linked data structures such as trees -

[http://en.wikipedia.org/wiki/Node_\(computer_science\)](http://en.wikipedia.org/wiki/Node_(computer_science)) [accessed 12/22/2009]. “In XQuery, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document (root) nodes. XML documents are treated as trees of nodes. The root of the tree is called the document node (or root node)”.

http://www.w3schools.com/xquery/xquery_terms.asp [accessed 12/22/2009]

³⁰ <http://www.xml.com/lpt/a/1559> [accessed 12/22/2009]

³¹ <http://www.oreillynet.com/pub/e/1225> [accessed 12/22/2009]

³² <http://databits.lternet.edu/node/52> [accessed 12/22/2009]

³³ <http://en.wikibooks.org/wiki/XQuery/Benefits> [accessed 12/22/2009]

³⁴ http://en.wikipedia.org/wiki/Web_service [accessed 12/22/2009]

³⁵ <http://www.w3.org/TR/xquery-sx-10-use-cases/#webservice> [accessed 12/22/2009]

³⁶ <http://www.soamag.com/I10/0907-3.asp> [accessed 12/22/2009]

Otegem has noted, “Any system that exposes XML over HTTP is a potential source of data for XQuery.”³⁷

An important feature of XQuery is its support for the concept of collections or groups of files that may be queried as a discrete set. “Besides organizing documents, collections are useful for restricting or “scoping” queries to include a specific subset of documents.”³⁸ When used in a RESTful system, the URL for an XQuery can act as an XML web service. This allows XQuery to be used as a building block for other XML tools like XForms (see below).³⁹

Like all of the components of this system, it requires a good understanding of XML, and XPath in particular. The relatively recent development of XQuery means that, as with XML native databases, not as many examples and implementations exist to use as models. Finally, XQuery support in XML native databases is widespread but not universal.⁴⁰

XForms

Forms provide an easy way to create a user interface for a system. On the internet, forms play a big role in interactive web applications.⁴¹ Most web-based forms however, leave much to be desired; they are usually embedded in a single web page, they are built with programming that is not friendly to users employing assistive technology such as screen readers, and they are not optimized for working with data in a seamless fashion. Data entry in HTML forms needs to be checked and translated by a middle layer of programming before it can be sent to a server and responses from the server must go back through that same layer.⁴² XForms was designed to address these kinds of problems. XForms is an XML based web application.

The W3C standard for XForms explains that splitting traditional XHTML forms into three parts — the XForms model, the instance data, and the user interface — allows the separation of presentation from content, reduces the number of round-trips to the server, and offers device independence.⁴³ McCreary points out that XForms also, “enables the developer to reuse business rules encapsulated in XML Schemas (xsd) and XML Transforms (xslt), reduces duplication, ensures that a change in the underlying business logic does not require rewriting [forms] in

³⁷ <http://www.devx.com/codemag/Article/16124/1954> [accessed 12/22/2009]

³⁸ <http://databits.lternet.edu/node/52> [accessed 12/22/2009]

³⁹ <http://broadcast.oreilly.com/2008/11/xrx-and-context-delivery-archi.html> [accessed 12/22/2009]

⁴⁰ <http://www.oreillynet.com/pub/e/1225> [accessed 12/22/2009]

⁴¹ <http://www.w3.org/TR/xforms/> [accessed 12/22/2009]

⁴² <http://broadcast.oreilly.com/2009/01/analysis-2009-xforms-and-xml-e.html> [accessed 12/22/2009]

⁴³ <http://www.w3.org/TR/xforms/> [accessed 12/22/2009]

another language, and supplies the control layer that moves data elements to and from the [data] model.”⁴⁴

As with XQuery, XForms becomes especially valuable when paired with an XML native database because the XML standard used throughout the system eliminates data translation. The W3C standard notes, “XML submission means the XML instance document received by the server can be directly validated and processed by the application back-end.”⁴⁵ And because it is designed as a web-based interface, XForms fits well with RESTful applications. Developer Kurt Cagle explains how it works. By using RESTful actions, “GET as XML is a specific schematic representation of the resource – GET as XForms provides the editor for that representation and PUT or POST accepts that representation as input.”⁴⁶

Despite these advantages, XForms currently has limited (though increasing) implementations. Support for XForms is included in IBM Lotus Forms, Orbeon Forms, and the Mozilla XForms 1.1 extensions for Firefox 2 and 3.⁴⁷

Summary

Daniela Florescu, one of the authors of the W3C XQuery standard, points out that XML is the only flexible and adaptable abstract information model that is not based upon the traditional relational database paradigm. Further, XML is the only format that can be processed automatically while preserving the structure and essence of natural language. She also notes that XML blurs the distinction between data, metadata and code and can handle dissimilar or changing schemas by allowing the user to ignore them and work with the data directly. She emphasizes that no other technology offers similar advantages and says that, “For the first time, text-based information has a data model perfectly suited to it.”⁴⁸

XML native databases expand upon the advantages XML offers by using the XML model to index and store documents. No other system facilitates access and reuse of text based information as well, especially large documents. They make it easy to expose resources on the web and because XML native databases support searches that allow use of the structure and meaning embedded in the XML tags, without being restricted by them, they serve as excellent tools for data integration.

⁴⁴ <http://www.danmccreary.com/presentations/xrx/otug.ppt> [accessed 12/22/2009]

⁴⁵ <http://www.w3.org/TR/xforms/> [accessed 12/22/2009]

⁴⁶ <http://www.oreillynet.com/pub/e/1225> [accessed 12/22/2009]

⁴⁷ <http://www.w3.org/MarkUp/Forms/wiki/FrontPage> , <http://www.w3.org/MarkUp/Forms/> [accessed 12/22/2009]

⁴⁸ <http://www.webservicesummit.com/Trends/DeclarativeXMLwithXQuery.PDF> [accessed 12/22/2009]

Combining XML native databases with a suite of tools based on XML standards creates a system that is greater than the sum of the parts. The combination of XForms, RESTful interfaces, and XQuery (XRX) exemplifies this principle. System architect Dan McCreary describes XRX as, “A web development architecture, an architecture based on international standards that is designed to minimize the probability of vendor-lock in, that gives a rich user experience ... [provides] portability on both the client and the server using a variety of forms players and XQuery databases, [and] the option of avoiding costly shredding (and reconstitution) of complex XML documents into [relational database] tables. A community of standards/tools and a "complete solution" ecosystem that can give you a proven [return] on your IT investment.”⁴⁹

⁴⁹ <http://datadictionary.blogspot.com/2007/12/introducing-xrx-architecture.html> [accessed 12/22/2009]