PROMISING TECHNOLOGY MAKES IT EASIER TO BUILD SERVICES THAT WORK WITH XML

Level: Intermediate
Frank Cohen (frank.cohen@rainingdata.com), Director, Solutions Engineering, Raining Data Corporation

06 May 2005 (Updated 07 Jul 2005)

XQuery shows much promise for software architects and developers because it greatly reduces the amount of code you need to write to build services that work with XML. You might think XQuery does everything and is well understood, but misconceptions and misunderstandings still exist in the software development community about XQuery. In this article, Frank Cohen details and clarifies many of the myths and misunderstandings that surround XQuery.

If you work with XML, Web services, or Service Oriented Architecture (SOA), you will likely benefit from the emerging XML Query (XQuery) standard. XQuery is not even a formally accepted standard, yet dozens of implementations help software architects and developers every day. What began as a standard for querying XML documents now includes the next-generation standards for XML selection (XPath 2), XML serialization, full-text search, and functional XML data modeling. A project of this size is bound to have much myth and misunderstanding that needs to be debunked. Here are some of the more common myths and misunderstandings surrounding XQuery.
TABLE OF CONTENTS

Misunderstanding - Database companies see XQuery as direct competition to their core businesses ................................................................. 2
Myth - XQuery will replace XSLT .......................................................................................................................... 2
Myth - XQuery will replace SQL ...................................................................................................................... 3
Misunderstanding - To adopt XQuery, you must abandon procedural programming ........................................... 3
Myth - XQuery is more difficult to use than JDOM, JAXP, and other XML parsing APIs ................................. 4
Myth - XQuery is difficult to learn to use ........................................................................................................ 4
Misunderstanding - XQuery is not product; XQuery is just a layer in a stack somewhere .............................. 4
Misunderstanding - XQuery is not important in improving Service Oriented Architecture (SOA) performance and complexity ........................................................................................................ 4
Misunderstanding - XQuery does not have a significant role to play in data transformation ........................ 5
Misunderstanding - XQuery offers nothing to On-Line Analytical Processing (OLAP) and data warehousing applications ...................................................................................................................... 5
Myth - XQuery will not scale to handle large datasets; XQuery will never be as fast as relational databases ........................................................................................................................................ 6
Misunderstanding - Aren't XPath and XQuery the same thing? ...................................................................... 6
Misunderstanding - XQuery lacks an update mechanism ............................................................................... 7
Myth - The XQuery specification will never achieve RFC status .................................................................. 7
Myth - XQuery supports token-based text searches ....................................................................................... 7
Summary ......................................................................................................................................................... 7
Resources .................................................................................................................................................... 8
About the author ........................................................................................................................................ 9
MISUNDERSTANDING: DATABASE COMPANIES SEE XQUERY AS DIRECT COMPETITION TO THEIR CORE BUSINESSES

Database companies see XQuery as an opportunity to augment their core solutions.

To software architects and developers, XQuery delivers increased productivity and agility. It makes sense that tool vendors (see Resources) are eager to get behind XQuery.

To a developer, XQuery looks a lot like SQL, so it’s natural for comparisons to be made. Additionally, more and more data is being marked-up using XML; this puts pressure on database companies to add XML storage, persistence, and query capabilities to their products. XQuery has so much developer momentum behind it that IBM and Oracle put their rivalries on the back burner to extend their core database products to offer XQuery capabilities.

Database companies also see an opportunity to be the first -- and eventually market-dominating -- supplier of a database that takes full advantage of the XML format. Data stored in relational databases today is normalized around rows and fields. In the XML world, each row contains an unlimited number of fields and each field is part of a parent/child hierarchy. The database vendor that delivers fast performance and XQuery flexibility first will win a huge new market.

As evidence of this opportunity, XQuery has united IBM and Oracle -- otherwise fierce competitors -- to jointly propose JSR 225 (see Resources), the XQuery API for Java (XQJ). And on the .NET side, Microsoft and IBM have teamed to submit an XQuery test suite to the World Wide Web Consortium (W3C).

MYTH: XQUERY WILL REPLACE XSLT

XQuery and XSLT have enough developer momentum behind them that both will co-exist. In fact, the most recent specifications for XQuery 1.0 and XSLT 2.0 are being produced in tandem.

Where XQuery and XSLT overlap is in the problems they solve: transformation of XML data, federation of XML collections, and advanced query of XML data. Developers will continue to see debates about the capabilities of each, including plenty of myth and misunderstanding. For example, I often see the claim that XQuery’s ability to query multiple, disparate sources in one pass gives it a distinct advantage over XSLT. In fact, XSLT 2.0 processors can have multiple nodes supplied as an input sequence. XSLT 1.0 has the document() function for accessing multiple sources within a single transformation, and XSLT 2.0 supports the new collection() function. I also often hear the claim that while XQuery syntax looks nicer, it lacks XSLT’s template style pattern matching. While this might be true, I fully expect XQuery to add that function too. In the end, developers can expect improvements and challenges in both technologies that will keep them close to one another in terms of function and capability.
Finally, there is the issue of the developer's thick skull. The XSLT presentations I have attended have left me feeling like I didn't really get it. XSLT is a transformation syntax that does not have a main() or start method like the Java and Jython code I normally write. At times, an XSLT script looks non-deterministic to me. So I haven't really gotten my head into XSLT. XQuery looks like SQL and solves many of the SQL problems that cause me to run back to my bookshelf for answers.

**MYTH: XQUERY WILL REPLACE SQL**

XQuery is best suited for XML just as SQL is best suited for relational data. XQuery brings SQL-like querying power to applications that require access, selection, integration, and transformation from one or more XML collections. While XML enthusiasts have begun to see everything in the world encoded with XML tags, the relational database model is entrenched and most of the world's digital data is encoded in tables that are composed of rows and fields. SQL will not go away anytime soon. Instead, extensions to XQuery have already appeared that allow queries to treat the results of an SQL call as part of an XML document collection.

As I mentioned, XQuery is to XML what SQL is to relational data. However, sometimes XQuery is easier to use, even when working with relational data. For example, using SQL to create a multi-table outer join that outputs its results to a new XML document is much more complicated to the average developer than writing XQuery.

XML’s popularity has spurred standards body working groups to expand the SQL specification to include XML processing functions. The SQLX Group, INCITS H2 group, and ISO/IEC JTC1/SC32/WG2’s standardization of SQL/XML are all working to extend the SQL standard to handle XML data.

**MISUNDERSTANDING: TO ADOPT XQUERY, YOU MUST ABANDON PROCEDURAL PROGRAMMING FOR OBJECT PROGRAMMING**

XQuery works within procedural scripting languages and object-oriented programming languages alike. If you are satisfied writing PHP scripts, then you should continue to do so. XQuery implementations exist for most available programming languages.

XQuery benefits developers by reducing the amount of code needed to perform a query. Sometimes a developer will have relational data in two or more databases, and need to produce a report that shows the union of those databases. A developer who is comfortable using a procedural programming language like Python may write 100 or more lines of code to retrieve, parse, and process the data. Or that developer can write a few lines of XQuery.
MYTH: XQUERY IS MORE DIFFICULT TO USE THAN JDOM, JAXP, AND OTHER XML PARSING APIs

XQuery is less difficult to use with XML data than XML parsing APIs. JDOM, JAXP, and other XML parsing APIs provide Java code with methods for working with XML data. Many object-oriented design patterns propose writing objects that will deal with complexity within an XML document. Writing Java objects takes time, effort, and expertise. Any small change to the underlying XML data format requires maintenance of the object. XQuery enthusiasts bet that an XQuery script will more rapidly find the XML data that an application needs to present than if a developer wrote a Java object using JDOM. Additionally, many XQuery libraries provide Java interfaces so that the XQuery code you write appears in your Java classes and delivers the result set to you just as if you called a method. Your Java class then processes the result.

MYTH: XQUERY IS DIFFICULT TO LEARN TO USE

Software developers who write in Java code, .NET, and other languages find XQuery easy to learn. XML has many components that no one would describe as elegant, including parts that were holdovers from the earlier SGML standard. XQuery uses a concise set of commands to make it easy to work with XML data. While the average developer faces certain challenges in adding XQuery to his or her repertoire, the learning curve is not very steep or prolonged.

MISUNDERSTANDING: XQUERY IS NOT PRODUCT; XQUERY IS JUST A LAYER IN A STACK SOMEWHERE

Whenever XML data needs to be managed and manipulated, XQuery is a specification for functions that a library or an application programming or service stack can provide. However, the underlying mechanism for storing, retrieving, and indexing the XML data makes a huge difference in the function, performance, and scalability of the XQuery implementation. For example, early attempts at storing XML data in varchar2 fields of a relational database resulted in poor query performance when an XQuery engine was simply layered on top. This led to specialized XQuery solutions in content management, data persistence, Web services and Service Oriented Architecture (SOA), data warehousing, online analytical processing (OLAP), extract/transform/load (ETL), enterprise application integration (EAI), and supply-side management.

MISUNDERSTANDING: XQUERY IS NOT IMPORTANT IN IMPROVING SERVICE ORIENTED ARCHITECTURE (SOA) PERFORMANCE AND COMPLEXITY

Software architects and developers are turning to XQuery to solve performance and complexity problems as the systems they build handle huge amounts of XML data. Consider the following scenarios and XQuery solutions:
Early analysis shows XQuery plays an important roll where payload sizes and XML schema complexity in ebXML- and UBL-based services are exploding.

XQuery greatly enhances UDDI solutions as it better manages and governs the resources listed in a UDDI registry.

Software architects find slow-moving data caching is one way to accelerate SOA performance.

In an analogous situation to Web page edge caching, services that receive many requests for the same information can use an XQuery engine to temporarily cache the XML data. XQuery implementations usually deliver both the XQuery scripting capability and a data persistence or storage facility. The service exposes the XQuery as a service and uses the underlying XQuery XML database to temporarily cache the XML data.

Additionally, in the supply chain application space, XQuery XML storage and retrieval has the potential to play an important role in accelerating overall system performance. Imagine the advantage XQuery-based XML storage and query functions can have in supply chain transactions where every product is tracked within the context of a business relationship that is described in an XML document.

MISUNDERSTANDING: XQUERY DOES NOT HAVE A SIGNIFICANT ROLE TO PLAY IN DATA TRANSFORMATION

XQuery plays an important role in data transformation as new schemas are adopted and existing schemas evolve. For businesses that need to build a supply chain application, the most costly area is transforming incompatible message formats. For example, imagine a buyer who standardizes on a standard like RosettaNet and is away from the original in-house developed schema. As the supplier, you now need to make your supply chain application compatible with RosettaNet but want to avoid the cost and risk of moving your existing system to RosettaNet. XQuery is a solution that enables your business to migrate to the new standard without stopping your existing sales operation.

XQuery provides you with a way to map, or transform, your existing schema to the RosettaNet format without having to write a huge library of new code. Instead, you write an XQuery that transforms your existing response data into a RosettaNet response.

MISUNDERSTANDING: XQUERY OFFERS NOTHING TO ONLINE ANALYTICAL PROCESSING (OLAP) AND DATA WAREHOUSING APPLICATIONS

XQuery does provide needed linking capabilities to OLAP and data warehousing applications. For instance, the average enterprise commonly has more than one data warehouse to track and analyze company data. These warehouses act like data silos and require effort, money, and expertise to mine for business knowledge. Linking one silo to another is usually a huge and expensive effort. XQuery offers a solution to assist OLAP by providing a query-based link
between multiple data warehouses. For example, one data warehouse stores products shipped for a home supply retail chain, and a second data warehouse stores product support call logs for the products offered in the retail chain. XQuery bridges these data warehouses by showing which products cause the most unresolved support calls. This illustrates the XQuery advantage in logical data warehousing, analytics, extract/transform/load (ETL), and enterprise application integration (EAI).

**MYTH: XQUERY WILL NOT SCALE TO HANDLE LARGE DATASETS; XQUERY WILL NEVER BE AS FAST AS RELATIONAL DATABASES**

In many ways, the XQuery standards industry looks at the Internet as one big distributed XML database. In this view, a query language appears in the role of a browsing capability where users find data in one or more retrieved documents. From the database view, XQuery is a tool for structural and content-based querying over the large dataset that is the world-wide XML database. The view is really that big.

Scalability and performance of XQuery solutions depend on the target of the XQuery implementation. For example, some XQuery implementations focus on content management and integration services. These are best used for publishing Web sites and Web portals to limited-sized audiences. XQuery implementations that focus on XML database functions are best used for handling large datasets efficiently.

An easy way to learn the focus of an XQuery implementation is to look at its origins. For example, looking at the XQuery working group shows two very distinct constituencies: those that come to XQuery from the XML document space and those working with XML as data. The document-oriented members come from an SGML past where agile access to a relatively small amount of XML data is important. The database-oriented members come from a hierarchical, relational, and XML database past and recognize the importance of indexing, extensions for text search, transactions and two-phase commit, external indexes, and an SDK/API for developers.

**MISUNDERSTANDING: AREN'T XPATH AND XQUERY THE SAME THING?**

Actually, XQuery builds on XPath and XSLT. Software architects and developers use XPath as a query language to find elements in an XML document and transform them into XHTML or another XML format using XSLT. For example, a developer uses XPath to find the dental records of a patient in an XML file and uses XSLT to package the patient information in an HTML view that is displayed in a browser. This works fine where the data is already in XML form, but XPath and XSLT only work on XML files.

XPath is selection-oriented while XSLT is transformation-oriented; both technologies still need an efficient way to select, join, and transform the data into the desired form. XQuery addresses the data needs of an application by enabling access to multiple sources, selecting...
information from them, and joining the data. This is true even for non-XML data -- sources include forms, Web pages, and other loosely structured data.

**MISUNDERSTANDING: XQUERY LACKS AN UPDATE MECHANISM**

It is true that the XQuery specification does not include an update mechanism. Additionally, at the time this article is being written, the XQuery working group is on "Last Call" status for the main XQuery specification and few working group members are willing to spend their time on the update specification. I expect the SQL-style approach will wind up in the XQuery specification. Updates will likely be expressed in a set of standalone operations that mimic and support existing relational database commands. However, some implementers and existing implementations offer a more free-form way of composing an update with XQuery.

It is important to note that most XQuery implementations provide an update mechanism of their own. For example, one popular XQuery engine implements an extension that provides Create, Read, Update, and Delete (CRUD) operations on XML and non-XML data.

**MYTH: THE XQUERY SPECIFICATION WILL NEVER ACHIEVE RFC STATUS**

It seems like it’s taken forever, but the XML Query Working Group and the XSL Working Group are at "Last Call" status on the XQuery, XPath, and XSLT languages at the time of this writing. Also, a variety of mature XQuery offerings already exist.

**MYTH: XQUERY SUPPORTS TOKEN-BASED TEXT SEARCHES**

While a specification for XQuery full-text search does define token-based text search, the XQuery working group intentionally left certain areas underspecified. For example, XQuery offers no standard mechanism for loading a document or viewing a list of available documents. From my perspective, not specifying everything provides fluidity to XQuery. Current XQuery implementations vary in their focus as well as their underlying data management facilities. This fluidity makes XQuery just as appropriate as a database search system as it is to queue filtering. That is powerful.

**SUMMARY**

Again, XQuery shows great promise because it reduces the amount of code you need to write to build services that work with XML. The greater XQuery ecosystem provides a unified way to query XML documents, including XML selection, serialization, full-text search, and functional data modeling. Work continues at the XQuery specification Working Group, and this will lead to even more benefits for software developers who work with XML.
RESOURCES

> Review the **XQuery**, **XSLT**, and **XPath 2.0** specifications on the W3C site.
>
> Visit **XQueryNow**, an online community site for developers building XML-based applications. It includes a blog by author Frank Cohen.
>
> Read Howard Katz's "**introduction to XQuery**" here on developerWorks (September 2003).
>
> Read **JSR 225**, the XQuery API for Java.
>
> Check out the **XQuery Test Suite**, the proposed test suite for XQuery from IBM and Microsoft.
>
> Explore these groups that are working to extend the SQL standard for XML operations:
>  > The SQLX Group
>  > The INCITS H2 group
>  > ISO/IEC JTC1/SC32/WG2
>
> Download **XQEngine**, an open-source Java component for querying XML documents, now hosted on SourceForge.
>
> Check out the **XQuery Normalizer and Static Analyzer (XQNSTA)**, a Java API and GUI for normalizing and computing the static type of XQuery expressions (alphaWorks, March 2004).
>
> Visit the author's Web site, **PushToTest.com**, where you'll find TestMaker, a free open-source test tool that now includes an XQuery engine for parsing Web service responses.
>
> Take a look at IBM's **Xperanto project**, which leverages XML, XQuery, text search capabilities, and Web services technology to enable users to search XML documents, flat files, spreadsheets, and other sources of information housed in a single database.
>
> Find more XML resources on the **developerWorks XML zone**.
>
> **Browse for books** on these and other technical topics.
>
> Find out how you can become an **IBM Certified Developer in XML and related technologies**.
ABOUT THE AUTHOR

Frank Cohen (frank.cohen@rainingdata.com) is the go-to guy when enterprises need to build, test, and solve problems in complex interoperating information systems. Frank is author of Java Testing and Design: From Unit Tests to Automated Web Tests available now at http://thebook.pushtotest.com. He is also the principal maintainer of the popular TestMaker open-source test utility and framework, and Director of Solutions Engineering at Raining Data Corporation, publisher of the TigerLogic XDMS XQuery engine and native XML database.