

The SOA Magazine

Feature Article



Master Key for Unlocking Enterprise Data

by Sreedhar Kajeepeta

Published: December 3, 2009 (SOA Magazine Issue XXXIV: November 2009)

[Download PDF](#)

[Digg This](#) • [De.licio.us](#) • [Slashdot](#) • [Technorati](#) • [StumbleUpon](#) • [Google Bookmark](#)

Introduction

Of all the integration challenges that enterprises are grappling with today, none are as complex, and getting progressively compounded, as the one related to data.

Data of strategic and tactical interest is everywhere and growing rapidly. It is being generated by mission-critical applications (legacy and new), services (both within and outside the firewall), and surrogate applications (such as desktop databases and spreadsheets). We need to first keep it all in sync through Master Data Management (MDM) or Enterprise Information Management (EIM), Event-Driven Architectures (EDA), and Messaging solutions. And then, we should make it accessible from anywhere - be it through data warehouses, data marts, ad hoc and canned reporting applications, dashboards/portals, Web 2.0/Mash-up applications, or mobile reporting solutions. It would also be nice if we could do the same to the other, unstructured, dimension of data, i.e content that is equally important and is growing even more rapidly.

In a forecast published in April 2008, Gartner predicted that the data integration tools market alone would go beyond \$3 Billion by 2012, with a healthy CAGR of 17%+. Needless to say, those numbers can only portend higher multiples of revenue and growth for the data integration services market.

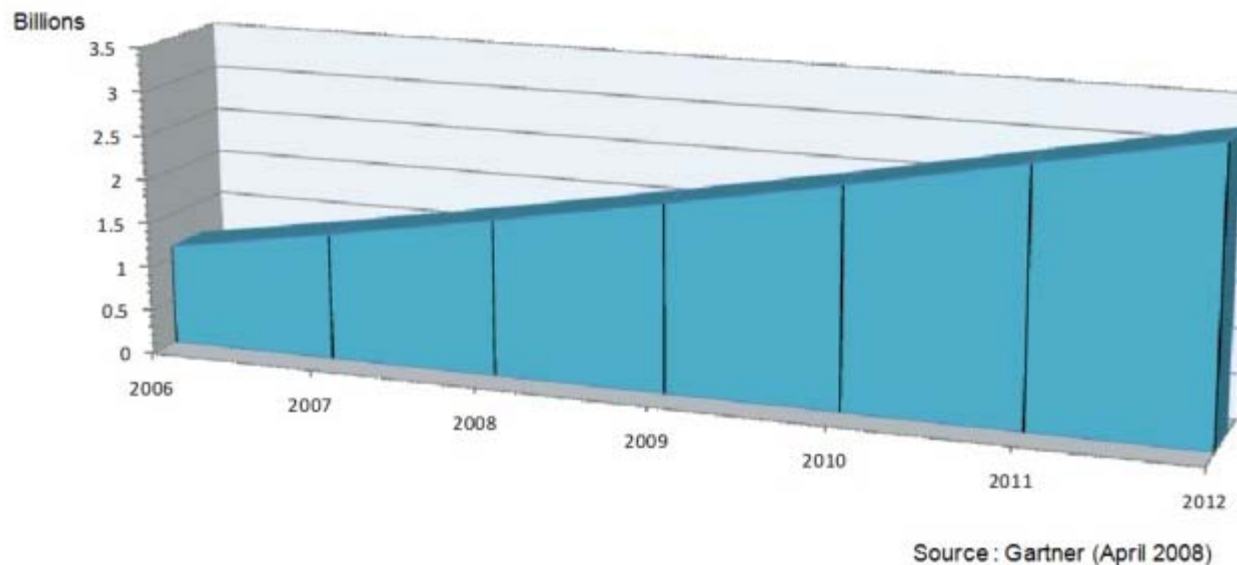


Figure 1: Data Integration Tools Revenue

Need for Data Services

In developing data integration solutions, with all the tools that those billions of dollars can buy, if there is one

practice that reflects the inelegant and/or ignorant nature of our approach to them, it ought to be the lack of attention to developing data services. Much like business/IT services that are addressed as part of the core strategy of building service-oriented architectures (SOA) for improving business agility and operational efficiencies, data services can help us unlock and apply critical data of heterogeneous origins a lot more efficiently. Sometimes, this aspect of data integration is also referred to as the 'last of mile' of the longer journey. A well-written set of data services can become the master key for accessing and manipulating enterprise data - by being very handy and powerful. Most SOA projects apply service-orientation to the business logic alone, through identification and creation of Web services. But unless that discipline is extended to handling data access logic those Web services will most likely be crippled in their ability to process in-coming requests and to create out-going payloads. When combined with a policy-driven approach to SOA, development of data services can also insulate sensitive original data from unfettered direct access.

Enter XQuery

Of all the tools available for building data services, XQuery (short for XML Query), is perhaps the simplest and the most effective to use. Although originally developed only to be a mere alternative to SQL for working with XML databases, XQuery gradually carved out a niche for itself - a broader, and more complementary role of being the language of choice for developing data services. Starting out in its earlier form as Quilt, XQuery has been languishing since May 2000 as a specialized tool while XML databases (along with their Object-Oriented counterparts) kept struggling to be a viable and mainstream alternative to relational structures which continue to rule the roost in database technology. Only recently, in January 2007, XQuery eventually became a W3C recommendation together with its companion standard XSL Transformations 2.0 (XSLT).

Throughout this journey, with pioneering support from vendors like DataDirect, XQuery has been seen as a language that is the most convenient for writing business logic against XML - simple, yet expressive enough as a functional language with an organic connection to XML. As an extension of this capability, and with the growing popularity of XML for enterprise data formats and B2B data exchanges (many industry standard vertical schemas are widely being used to create cross-boundary virtual enterprises), XQuery became an attractive tool for data aggregation and transformation. With traditional data integration techniques, such as Extract Transform and Load (ETL), we can perform the heavy lifting of high-speed bulk transfer and formatting. But, what's missing among them is a nifty technology to get us to the 'last mile' - the ability to build a storage independent mechanism to unlock the data. XQuery offers an open platform to manipulate and integrate any data of interest to the enterprise - just as long as that data can be exposed as XML, which is true with even such obsolete and obscure structures as VSAM let alone all the popular relational and legacy data formats.

A new W3C working group is looking at lending full CRUD support for XQuery making it update-capable as well (some proprietary implementations already do support it, however). Yes, there are other XML processing tools such as Java/JAXP, JavaScript/DOM, XPath (for XML selection), and the afore mentioned XSLT (XML formatting), but only XQuery offers a complete platform to develop composite data services involving business logic, XML serialization/formatting, full-text search, and functional XML modeling.

XQuery Ecosystem

Vendor support for XQuery has been growing steadily in recognition of its unique role in building data integration solutions. Supporters include mainstream database vendors such as Oracle/BEA (Oracle 11g, and Oracle XML DB, and Oracle's open source initiatives such as Berkeley XML DB and XQuilla XQuery Engine), IBM (DB2 Ver9, Optim Development Studio), and Microsoft (with SQL Server 2005 - built into ADO.Net). Among niche supporters, pioneer DataDirect leads the pack (with their Data Integration Suite providing development and deployment support for XQuery), with Software AG (Tamino Server), Ipedo (XIP), Actuate (Actuate 8), and OpenLink (Virtuoso Universal Server) etc., also in the list. A big part of the ecosystem is the XQuery/Java API, called XQJ (or JSR225), which greatly enhances XQuery's capabilities for developing data services in Java. The same is true for .Net environments with the built-in support through ADO.Net.

In his article [REF-1] Binildas Christudas puts together nicely all complementing pieces of the puzzle in the

XQuery ecosystem.

"XPath is optimized for accessing sections or parts of an XML document. Thus we can immediately use XPath if the requirement is just to select a node from within an XML document. But XPath cannot return a part of the selected node (like the node element tag alone, omitting content) and it cannot create new XML.

XSLT includes XPath as a subset to address XML document parts and also includes many other features. XSLT can contain variables and namespaces and can create new documents. XSLT is optimized for recursively processing an XML document or translating XML into HTML, WML, VoiceXML, etc. But writing a user-defined function or other common operations are tedious in XSLT, and XQuery scores here by expressing joins and sorts. It can also manipulate sequences of values and nodes in arbitrary order, not just in the order in the document. It is also easy to write user-defined and recursive functions in XQuery.

XQJ defines a set of interfaces and classes that enables a Java application to submit XQuery queries to an XML data source and process the results of these queries. Queries may be executed against individual XML documents or collections of XML documents. The XQuery standard provides a great degree of freedom for implementers in how they choose to implement many of its features. This means different implementations can differ in how they handle a temporary intermediate result as long as the query produces the correct, final "answer." A few XQuery implementations are available now, amongst which Qexo is worth mentioning. Similarly, Saxon is a collection of XML processing tools by Saxonica for XSLT 2.0, XPath 2.0, XQuery 1.0, and XML Schema 1.0. Saxon also offers two other APIs for XQuery processing: Saxon's own native API, and an early implementation of the XQJ. Saxon is available for both the Java and .NET platforms as two packages: Saxon-B and Saxon-SA. Saxon-B and all its features are available under an open source license to all users, whereas Saxon-SA requires activation by a license key."

For .Net developers, XQSharp might be more appealing as an organic alternative. Saxon is a native Java app that uses .Net JVMs like IKVM files to port to .NET, and that may complicate deployment of applications.

A related development in the world of .Net is Language-Integrated Query (LINQ), which was introduced as a feature in Visual Studio Orcas (now, Visual Studio 2008). LINQ has "LINQ to XML" and VB-XML language extensions that provide functionality similar to XQuery. LINQ one can operate over objects, SQL Databases, and XML documents, to go after virtually any type of data. Hence, as an integrated facility for data access LINQ can be used to work with in-memory data, and in conjunction with ADO.Net to access data in RDBMSs/data sets/ORMs, and XML files. So, much like XQuery, LINQ does offer the versatility needed for accessing diverse data sources, but it does so only in a single (i.e Windows) platform, and it lacks W3C support.

As for the world of popular legacy data sources, let us look at DB2, IMS DB, and VSAM as popular representative examples. IBM's DB2 9.1 (earlier code-named DB2 Viper) comes with a number of significant advances with regard to support for XML, in recognition of the key role XML is already playing, in handling data and content, and as a means of data exchange in SOA platforms and architectures. DB2 9 extends DB2 to allow XQuery operations to be performed against relational data as well. While data sources like IMS and VSAM can always be exposed as virtual relational structures with dedicated ODBC/JDBC drivers (using products like Data Direct's Shadow), new enhancements in IMS Version 10 provide access to IMS Full Function data, including new IMS Xml data through support for standard XQuery expressions.

XQuery Case Studies

Widespread success stories of XQuery solutions are yet to come out in a big way. The technology still remains too esoteric (tucked away as an architect's secret to success in data integration) to attract any broader attention from enterprise applications management teams. Customer quotes on DataDirect's web site [REF-2] and XQuery's W3C site [REF-3] refer to its usefulness in such vertical solutions as ACORD and Basel II implementations, and in many horizontal applications such as Microsoft .Net Framework and

schema-aware applications.

A very exciting case study of XQuery comes from CSC. DynCorp, now an acquisition of CSC, used XQuery to develop statistical analysis for various types of aircraft data for NASA. Michael Ritchson of DynCorp (CSC) had this to say about their experience with XQuery : *'It's rare in this business that you come across a software technology that is more powerful, easier to use, faster, actually allows for increased application requirements, and not to mention, is free to use - but in truth, that was our experience with XQuery and we were thrilled with the results'*.

Future of XQuery

As a standard language to work with XML and to create XML payloads from the multiple data sources and databases that a typical enterprise environment is likely to have, Xquery will continue to have a huge impact in our ability to realize efficiencies in data integration.

In an interview with DataDirect, Jonathan Robie, co-inventor of XQuery, felt very encouraged by the growing popularity of the language as evidenced by the close to 50 implementations of its engine. Here is what Robie had to say on the positioning and power of XQuery:

"XQuery is generally used either in the middle tier or on the server. The XQuery usually needs to be parameterized - you don't just write a query that returns a generic portfolio, it returns a portfolio for a particular user, in a particular time frame, and you have to specify these parameters somehow. One way of doing this is to post an XML document that contains such information as the name of the user and the starting and ending date. Another way to do this is to use URI parameters, and to bind these parameters to external variables in the XQuery before executing the query.

XQuery's prominence in the middle tier is due to the fact that not all of the data necessarily comes from a database, or it might come from more than one database. The XQuery itself is written by developers and executes inside the firewall, where it has access to the data, and people outside the firewall are generally prevented from writing queries. So the user's access to this data is limited to the queries that the developer provides and the parameters that the user is allowed to specify. This lets the developer use a very high level, efficient query language to create XML and to access whatever data is needed, while limiting the user to very well-defined interfaces.

We're dramatically improving our support for very large XML files, and adding filters to convert EDI, Java objects, email mailboxes, and other formats to XML so they can be queried. We're also adding support for MySQL 5, which many customers have been asking for, and for the most recent versions of the Microsoft and Oracle databases."

That optimism is shared by many SOA and data architects, who go on to say that not only is it potent enough to be the all-purpose query language of the future but also that it forms the basis for the often-neglected and highly critical aspects of enterprise architecture (i.e. data and content). To support that goal, there are on-going efforts (eXist XQuery engine) to adopt XQuery to a more RESTful form of enterprise architecture, through such standards as XRX (XForms, REST, and XQuery).

The functional programming side of XQuery is making waves of its own, by attracting the attention of Map-Reduce (or Hadoop) enthusiasts. This interest could be a harbinger of some exciting developments that may lead to the processing of large volumes of distributed data with XQuery on public/private Clouds. This aspect of XQuery's future leads us to the exciting and evolving application of this technology for handling complex distributed data processing applications. Known as DXQ (Distributed XQuery), this technology deploys a network of distributed XQuery servers to remotely process data in the most efficient manner.

The analysis about the future of XQuery does bring up the question about that one single trend (or the so called killer app) that is likely to drive its growth. If there is a simple answer to that question it would be 'Enterprise Information Management (EIM)', a discipline that is being increasingly challenged to present a unified view and means of pervasive access to all the data. The interplay between eCommerce, Web 2.0, social networking, and corporate data is making that challenge a lot tougher by blurring the lines between the structured and unstructured aspects of critical data. XML is emerging as the def-facto standard for

representing all such forms of data, enabling and streamlining intra-company, and inter-company (B2B) communications on the one hand and promoting device independence of technology platforms on the other. XML and its variants such as RDF (Resource Description Format) will be at the core of newer forms Web technology such as Semantic Web which would enable a knowledgebase that is both self-generative and machine-readable. XQuery as the universal language for accessing XML will continue to grow along with it, as it becomes a query tool of choice for heavy-weight (Java and .Net, and such) and light-weight (PHP, Groovy, and such) applications.

References

[REF-1] XQuery For Java, An Enabler for SOA, Java.Net, April 2007, Binildas Christudas

[REF-2] http://www.xquery.com/customers/customer_quotes.html

[REF-3] <http://www.w3.org/XML/Query/#implementations>

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About/RSS](#)
[Legal](#)

Copyright © 2006-2009
SOA Systems Inc.