WRITTEN BY **JERRY KING**

The Case for XQuery

XML adoption is growing steady

XML use is widespread across modern information systems in all industry, government, and academic sectors. The core technologies for processing XML (XML, XSLT, XPath, XML Schema, and others) are maturing steadily – thanks to support from standards bodies like the W3C and OASIS, and from major industry players such as IBM, Microsoft, and Oracle. XML is also the basis for a growing body of industry standards for data exchange, and it is well on its way to becoming a mainstream technology for data integration. XML is transforming not just data – it is transforming information processing in general.

However, XML as we know it today is not the whole answer. It is simply a way to represent data in a self-describing format that is easy to interpret across diverse systems. The demands of today's advanced data integration challenges require a flexible XML standard for data aggregation and transformation, one that can seamlessly work with relational and legacy data sources, as well as new Web service technologies.

XQuery Unlocks the Power of XML

One of the most recent developments in XML is the emergence of a native XML query and transformation language – XML Query, or XQuery. XQuery is under development by the W3C and is nearing Candidate Recommendation status. Even now, XQuery is poised to become the standard query language by which enterprises access and manipulate disparate data and content repositories. With XQuery, the query and transformation logic operates on XML views of the data. It does not depend on the data's physical structure. If this approach to data query and manipulation sounds familiar, it is: in SQL, a query describes in a declarative fashion the mapping between a set of input tables, and the output table, or result. The underlying data provider (a JDBC driver, a database client, and so on) takes the SQL and executes it against the relational database. To varying degrees, the application is thereby shielded from the underlying database platform. XQuery's similarities to the SQL paradigm can only help speed its adoption.

One important difference is that in SQL, everything has to look like a relational table, no matter how it is stored physically. In XQuery, you can have completely different storage systems and wildly different data structures, and, as long as the underlying data can be exposed as XML, it all still works.

XQuery Simplifies Data Integration

XQuery is designed to support multiple XML information sources as input. An XQuery program's primary functions are to select, filter, transform, join, and aggregate data across multiple data sources. The trick is that these data sources must be represented to the XQuery program as XML. Fortunately, there are also several products on the market, including Stylus Studio, that provide visual tools for building adapters that transform non-XML data sources (flat files, EDI, relational databases, and others) into XML. The net result is that a developer using current technology can build an XQuery program to join or aggregate data from diverse data sources, and produce XML as output. That XQuery program can then be deployed as a Web service that can be imported into another XQuery program, one that creates a composite view by combining this data source with other data sources in its integration scheme. This is just one example of the value of reusable, standards-based XQuery code.

XQuery's inherent data integration capability makes it a powerful tool for the modern application developer. Take service-oriented architecture (SOA) applications, for example. Data integration in the emerging SOA world means dealing with data from multiple sources (relational databases, XML files, legacy applications, and Web services, to name a few). XML is the perfect language for uniformly expressing all of this data, and XQuery is the easiest and most powerful way to process it.

To best appreciate the problem XML and XQuery solve, consider how much time developers today spend dealing with dynamic requirements for inter- and intra- enterprise information flows that must be integrated, current,

AUTHOR BIO

Jerry King leads DataDirect's XML Products group and is responsible for bringing the company's innovative XML technologies to market. When not pondering the emerging XML market, Jerry enjoys life in the great state of Maine.

and correct. A prime example of this can be seen in supply-chain management applications and the many other applications that integrate data from various sources in order to present unified customer and product information.

Building this data integration logic can be a costly, complex, and time-consuming process – some analysts believe that up to 70 percent of the effort on a typical systems integration project is devoted to "hand-coding" data-level integration logic. What's more, this code is very sensitive to any kind of change in the environment or even in the intended application usage. The net result is that developers often end up writing throw-away code – and spending 70 percent of their time doing it.

XML solves some of this problem by providing a *lingua franca* for data integration. To this end, XML Schemas exist for almost every industry sector imaginable to facilitate data exchange within organizations, as well as among customers, partners, distributors, and suppliers.

Even with XML, many developers are using hand-coded programming approaches that incorporate Java, DOM, XPath, XSLT, and other methods, all in an effort to query and manipulate XML data. Low-level approaches like DOM are difficult to write and maintain because the query expressions, the aggregation, and the transformation logic to be evaluated (the what) are so tightly bound to the underlying query processing strategy (the how), that even small changes in application requirements can require substantial recoding efforts.

XQuery greatly simplifies XML querying and transformation by virtue of its simple and concise syntax. In addition, the developer who is using XQuery works with all data as an XML abstraction and can expect the underlying XQuery implementation to deal with accessing the physical data sources appropriately.

XQuery Will Simplify SOA Data Services

A key value proposition of SOA is the idea of creating loosely coupled, composite applications to bridge existing information systems and brand-new applications. Adherence to SOA principles also often requires the ability to aggregate and transform legacy, relational, and XML data sources to expose new federated views of data – usually as XML – that can be consumed by higher-level applications. To this end, XQuery may well emerge as a preferred method for building data-level SOA data services.

Developers building these *data services* will find that XQuery greatly simplifies data aggregation and transformation logic. This task will also require the ability to abstract relational and XML data sources to ease integration challenges. XQuery, fortunately, provides the foundation for vendors to deliver tools and components that do just this.

Vendor Solutions for the Future XQuery World Are Emerging

For example, products that can provide unified data access across XML and relational data, such as Data Direct XQuery, will be in high demand by developers tasked with assembling XQuery-based data services. A related specification – XQuery API for Java, XQJ – will provide a standard interface for easily embedding these XQuery programs in any Java program, much as JDBC does for SQL.

"XML is transforming not just data – it is transforming information processing in general"

Of course, performance will be another key factor in XQuery's adoption. Performance of data integration logic can be unacceptable in many cases due to the excessive network traffic and local memory consumption needed to process queries across disparate data sources. There are many products on the market today that deal with this problem by offering server-based solutions that separate the data integration logic from the application. Many of these platforms are XML-aware and have plans for supporting XQuery.

For example, Microsoft, IBM, and Oracle have all staked out a position in the XML world so that by morphing their respective databases at the API level, their platforms can easily serve as big and fast file servers for any data type. In this world, XQuery is a natural API for accessing the disparate data types stored in those servers, as well as for accessing external data sources such as file systems and WebDAV repositories. XQuery implementations from integration vendors like BEA, Ipedo, Actuate, and OpenLink are also on the market today. The good news is that many vendors are actively developing useful products and helping promote the use of XQuery. The bad news is that some of these vendors are using proprietary XQuery extensions and highly purposed implementations to deliver working products, with the unfortunate result that the XQuery services offered by these vendors are bound within the context of their platform-based solutions.

Other solutions coming to market will offer XQuery- and XQJ- compliant data access technology as an embeddable, high performance component. Interestingly, the DataDirect Technologies' XQuery implementation will expose as XML the relational data stored on any of the major database platforms. In other words, using relational data in an XQuery will not be dependent on the database vendor's support of XML. Also, DataDirect XQuery will provide hard-to-match performance benefits by pushing much of the distributed query and join operations to the underlying relational database platforms involved in the query. This fast and lightweight approach to data integration will be a natural fit for developing rich, data-level SOA services.

Summary

XQuery promises unprecedented productivity for developers solving data integration problems. Delivery on that promise by W3C and industry stakeholders will be the key to XQuery's success. As this article discussed, there are many arguments for the success of XQuery as a widely adopted programming language: ease of use, similarity to SQL, the demand for data integration, enhanced developer productivity, an active vendor community, interoperability with legacy data, and widespread use of XML are just a few of them.

jking@datadirect.com