SEYBOLD
CONSULTING

Oracle XML DB: Uniting XML Content and Data

By Victor Votsch and Mark Walter

Seybold Consulting Group

March 2002

The Oracle XML DB technology in Oracle9i Database Release 2 marks a noticeable improvement over Oracle's previous efforts to incorporate XML support in its database product. More significant, however, is that Oracle XML DB's uniting of XML documents and data is at the forefront of a new trend to merge the repository functions of these previously disparate uses of XML. This paper briefly traces the history of XML development and the evolution of Oracle's XML support; outlines the benefits of Oracle XML DB integrated approach; and explains in the context of larger industry trends why Oracle's new XML architecture and features mark a milestone in database technology that anticipates future market requirements.

The Evolution of XML

To understand the current market forces that bear on XML within databases, it's instructive to briefly outline the history of the language and its uses. XML is used today in ways far beyond its original purpose, and, as a result, the requirements for direct XML support from within database products have risen to unprecedented levels.

From documents to data

When the core protocols that define the World Wide Web were created in 1991, no one thought it would lead to a ubiquitous, global communications network that would create a backbone for information technology development. Yet as the Web became an integral part of business, computing, and publishing, it became apparent that there was more to content than static HTML pages and that there was no good way to handle it.

At XML's conception five years ago, its authors were searching for a way to bring the portability and vendor neutrality of SGML (the Standard Generalized Markup Language) into a simpler form that would be easier for Web developers to adopt. The XML authors wanted a meta language that would provide a consistent syntax for describing a variety of documents and their structures, just as SGML had done. Their motivations were the same as those that drove the development of SGML: these users had voluminous, high-value content (Sun's documentation manuals, for example), and they wanted to use generic markup to describe the structure of that content in a way that would automate the process of publishing the content in multiple media. The XML authors were highly motivated in 1996—everyone wanted to publish on the Web. At the same time, SGML's design proved too cumbersome for many software developers—Netscape and Microsoft included—to adopt. As envisioned, XML would be a simpler version of SGML that would attract a broader audience of developers to recognize the benefits of structured content.

However, the need to share information on the Web was so great that shortly after the World Wide Web Consortium (W3C) formally adopted XML in early 1998, people began to view XML as a way to describe structured data—the sort of information that lives in rows and tables of relational databases. This use of XML for data grew exponentially, and quickly surpassed narrative content as the driving force behind adoption of XML in the vendor community.

Two camps divided

By 2000, a schism had emerged between those primarily interested in XML for use with content and those primarily interested in applying it to data. The former had XML DTDs (document type definitions, a carryover from SGML) in place as a way to describe document structures. Their documents tended to have narrative, heterogeneous content designed for direct human consumption, as in documentation, electronic mail, books, press releases, or news articles, for example. In these

applications, XML typically was used to aid an organization internally—making the material easier to repurpose and reformat for a variety of output media.

In contrast, the latter group found DTDs, with their unique syntax and lack of data types, too restrictive for database information, so they developed XML Schema, an XML-compliant way to describe relational data as document classes. Data-centric XML applications tend to contain fine-grained, structured content that is fairly homogeneous and easily fits into the database model: purchase orders, flight schedules, scientific data, and stock quotes, for example. Often called "transactive content," such material is ideally suited to storage in relational databases. Organizations often apply XML to transactive content not only to improve output to the Web but also to make it easier to exchange the information with other servers, both inside and outside the organization.

While the two camps working on XML from different vantages didn't necessarily fight, they formed committees to work on standards relevant to their tasks, and vendors necessarily made choices as to where their priorities lie. Until recently, that meant that most vendors chose to focus on either document-centric or data-centric applications. In the database market, for example, products did not treat XML data and rich content in a consistent way. For the most part, they stored XML narrative content as single binary objects, and broke data described by a schema into tables of rows and columns. It was left up to third-party application developers to manage narrative content in more granular chunks, or to provide hierarchical, DOM and XPath views of XML data stored in a SQL database.

Heading toward convergence

Going forward, however, it will be increasingly common for XML implementations to employ both types of content. Many catalogs contain both narrative descriptive content and tightly constrained data, such as model numbers and prices, to name just one example. Rather than treat these components of the catalog as distinct data sources, as is typically done today, in the future users may choose to store them together in the database in a way that supports both document-centric and SQL interfaces. As users make this shift, XML Schema will overtake DTDs as the preferred method of delineating the structure of complex documents. By 2005, we expect that at least 70 percent of new XML-based content-management implementations will be based on schemas, rather than DTDs.

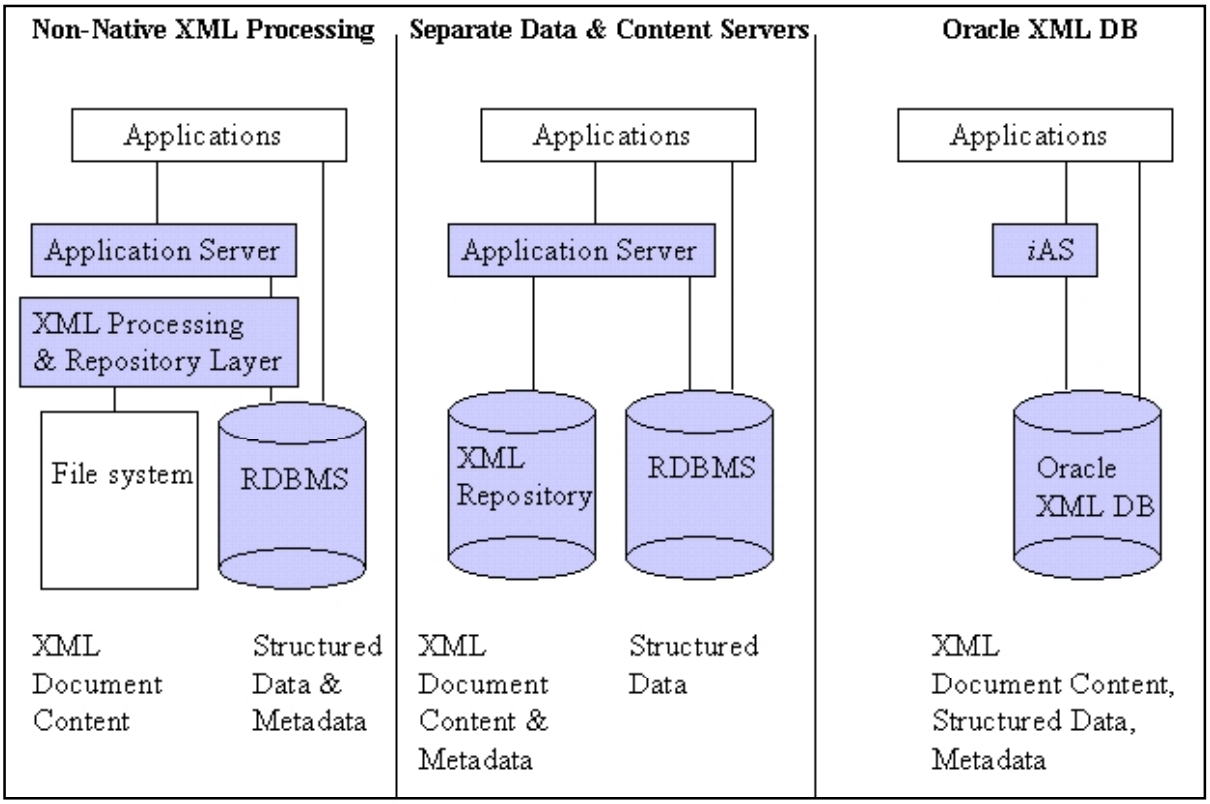
The XML DB technology of Oracle9*i* Database Release 2 represents the first integration of these distinct uses of XML by the world's leading database vendor. In our view, it's a milestone that will bring specific benefits to developers and end users, and one that is consistent with key trends in the computer and publishing industries, as we'll discuss later in this paper.

Evolution of Oracle XML technology

Oracle has been steadily moving toward integrating XML with its core database products. Its progress has paralleled the maturation of XML itself and the family of

standards that have grown up around it over the last several years, as shown in Figure 1.

Figure 1. Evolution of Oracle's XML technology, from Oracle8*i* to Oracle9*i* Database Release 2.



Oracle8i: XML middleware

Oracle introduced its first support for XML with the release of Oracle8*i* in late 1999. It was a relatively loose integration, featuring content transformation performed externally to the database itself, and was designed primarily to support emerging requirements for exchange of data in XML form. The XML components included an XSQL servlet, an XML parser and an XSL transform engine, all of which ran separate from the database itself. The XML Developers Kit—a set of APIs (available in multiple flavors: Java, C, C++, and PL/SQL)—was provided to help developers access the functions by manual coding. At the same time, Oracle made its *inter*Media full-text indexing product (now called Oracle Text) cognizant of XML tags.

Another component of Oracle8*i* was the Oracle Internet File System (*i*FS), which enabled developers to map the hierarchical view of the file system inside the structure of the database. Introduced in 2000, this innovation gave end users the

advantages of a database (including extensible metadata) within the familiar and readily understood interface of folders on the file system.

The combination of XML utilities and Oracle Internet File System made Oracle8*i* an attractive platform for XML-based content management. For example, Arbortext, a leading provider of XML authoring tools, developed an Oracle8*i* plug-in that enabled a team of authors to manipulate native XML documents in Oracle Internet File System through a user interface that looked just like a shared file system.

Oracle9i: Integrated XML processing

In Oracle9*i*, released in June 2001, Oracle added XML support directly to the database, primarily to improve performance for accessing transactional content.*

The Oracle9*i* Database Release 1 XDK contains a number of tools for processing XML into and out of the database, several of which were enhanced from the Oracle8*i* release. They include:

- XML Parsers: Written in Java, C, C++ and PL/SQL, these components create and parse XML using industry-standard DOM and SAX interfaces.
- XSLT Processor: The engine transforms or renders XML into other text-based formats such as HTML.
- XML Schema Processor: Supporting Java, C, and C++, it allows use of simple and complex XML datatypes.
- XML Class Generator: This routine automatically generates Java and C++ classes from schemas to send XML data from Web forms or applications.
- XML Transviewer Beans: These routines display and transform XML documents and data via Java components.
- XML SQL Utility: This Java utility generates XML documents, DTDs and schemas from SQL queries.
- XSQL Servlet: The Java servlet combines XML, SQL, and XSLT to deliver dynamic Web content.

In addition, new datatypes—one for XML (XMLType) and one for logical pointers (URI-Ref)—were added to the kernel for direct XML storage. (The limitation was that XML datatypes must be stored as binaries, with text indexes, in Oracle9*i* Database Release 1.) To facilitate the import of XML-encoded records, new Table Functions were introduced that could be used to decompose XML documents across multiple tables, and SQL operators could be applied to query and extract data stored in the tables. Oracle also introduced SQL operators that made it easy for SQL programmers to extract data as XML documents using familiar SQL syntax. The URI-Ref datatype introduced a vendor-neutral way to specify pointers to information both inside and outside the database. This change was the start of opening up the database for XML-based navigation; it also simplified accessing remote data over HTTP.

* It's important to note that Oracle9*i* Database users still have the option of configuring XML operations to run outside of the database, as they did in Oracle8*i*. Database administrators determine which configuration is most appropriate, depending on compatibility needs and application requirements.

Oracle9i Database Release 2: Oracle XML DB

Oracle9*i* Database Release 2, slated to become commercially available in the first half of 2002, will improve the performance and tighten the integration of XML handling by the database core. The new features, collectively referred to as Oracle XML DB , unite the SQL and XML metaphors for XML documents and structured data. The new features embrace the W3C data model, providing structured storage for XML data and documents within the repository. Among the features introduced in this release are automatic identification and parsing of XML schemas, navigation that adheres to the XPath standard written by the W3C, and substantial performance enhancements.

Oracle9*i* Database Release 2 adds object-relational storage, with relational indexing, as a second option for storing XML objects. Oracle's object-relational method, consistent with the SQL:1999 standard, maintains fidelity with the Document Object Model (DOM) and helps the database recognize important features of XML documents, such as distinguishing between elements and attributes and determining the order of child elements.

Schema support in Oracle9*i* Database Release 2 means that for the first time developers can separate the data definition from the physical storage of the data in an Oracle database. This separation allows developers to keep data definitions in an industry-standard, vendor-neutral form; it also simplifies the process of setting up the database at the outset. As an example, developers can use any of the popular XML schema editors—some of which have Oracle-specific extensions—to map the document tree to the database syntax and edit field definitions from within a graphical user interface (see Figure 2).

Oracle9*i* Database also supports the tree-like approach of XPath as an alternative to rows and columns as a navigation paradigm for the database. The tree metaphor provides another layer of abstraction, separating the details of physical storage from the logic of application programming. (Developers can write queries that specify an XPath location, and let the database take care of finding the matching records.) Another advantage of XPath is that developers can use it to create applications that update individual elements or attributes of XML documents, without requiring the rest of the document to be checked out at the same time. As developers begin to mix XML data and documents together in the repository, the Oracle9*i* XPath support gives them XML views of relational data and provides an interesting alternative method for working with specific nodes of information, especially fragments of large XML documents.

Figure 3 illustrates the different methods of access Oracle XML DB provides, contrasting connectivity via content-oriented tools with connections made via data-oriented tools. In Oracle XML DB, XML data and content can be accessed by both sets of tools, which gives developers greater freedom in developing user interfaces.

Figure 2. Registering an XML schema into Oracle9*i* Database using the graphical schema editor, XML Spy. Oracle extensions to the editor enable the user to map the schema (in the foreground window) to specific database fields (detail palette at right).

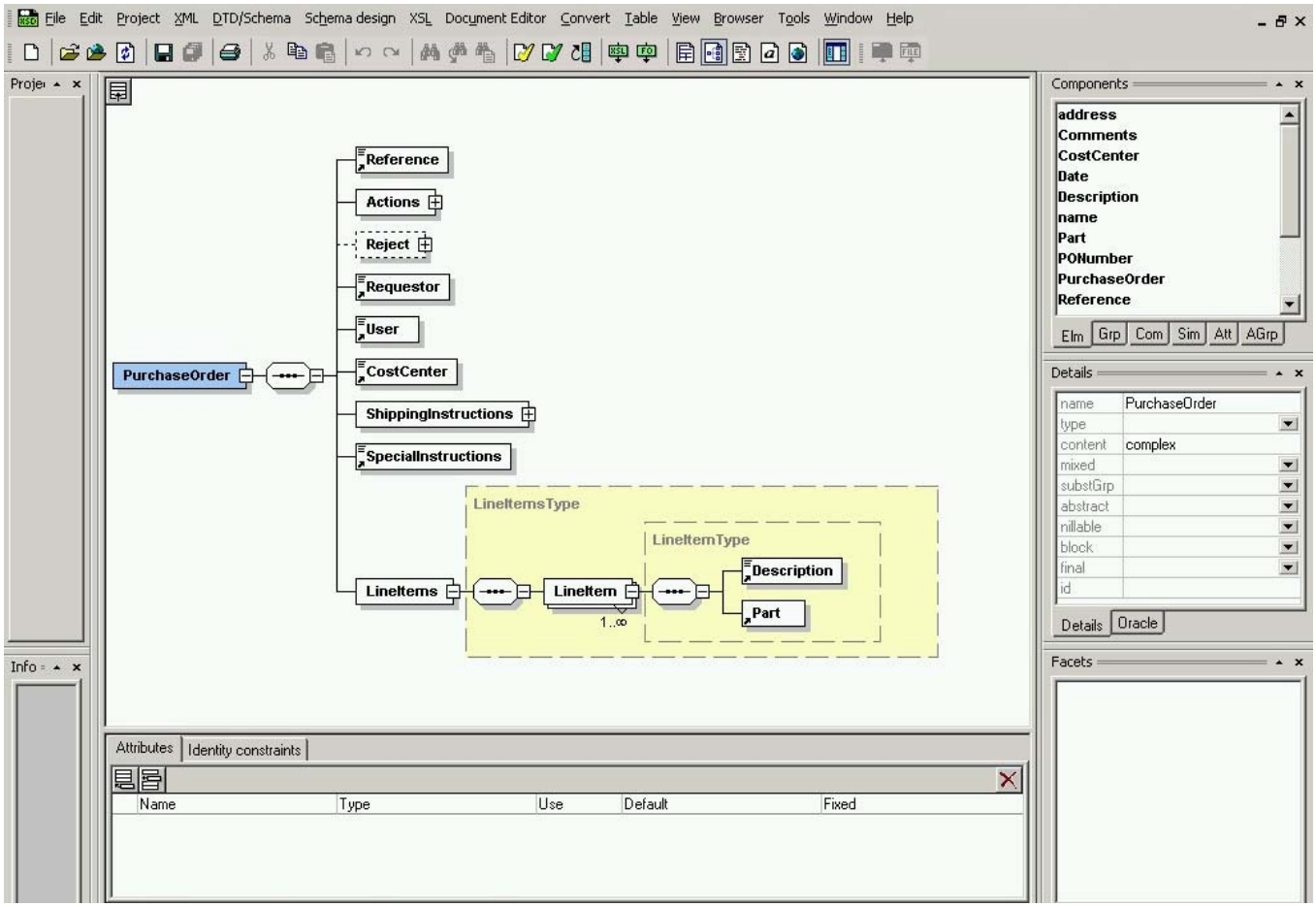


Figure 3A. Oracle XML DB architecture. In content-oriented access, Web clients, including desktop tools, access the Oracle9*i* Database directly.

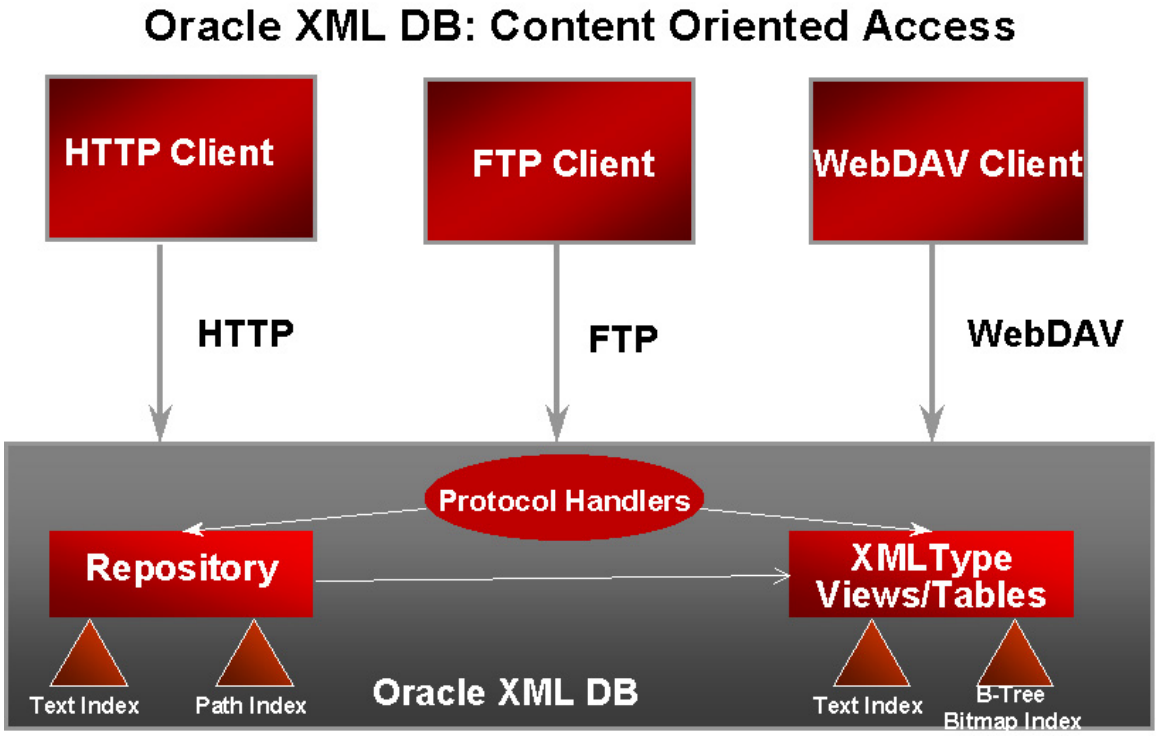
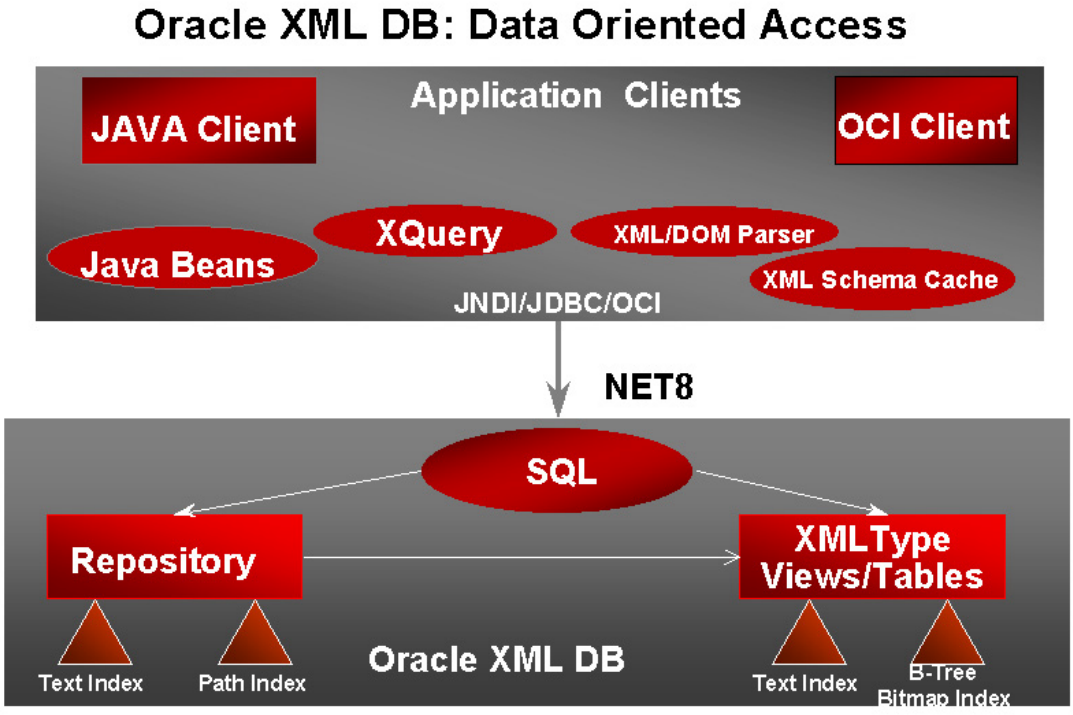


Figure 3B. Oracle XML DB architecture. In data-oriented access, JDBC or OCI clients gain access to both XML documents and relational data using traditional SQL calls as well as through XML-based XPath views.



XML DB in Relation to XML Server Applications

Because of the high number and wide diversity of XML applications, vendors have flooded the technology market with server products that offer XML-related functionality. Again, because of the diversity of applications, there has been a lack of clear segmentation among products: some are clearly very specialized; others are, by design, more general-purpose and may play several roles. This lack of clear segmentation will continue into 2002, in part because large server applications, like leading databases, will increasingly absorb XML-based functionality that was once found only in specialized products. Under such pressure, specialized XML databases won't necessarily go away, but they will face an increasingly competitive market, one in which customers find that building XML applications on top of general-purpose products is a viable option.

Oracle9*i* Database Release 2 with XML DB fits this pattern. XML functions once housed outside the database have been brought inside to boost performance, and new capabilities have been added that extend the underlying database into areas that overlap with server products in different sectors. In fact, the new release is designed to address key requirements in virtually all of the XML server market segments that we've identified. As a result, Oracle is poised to capture a significant market position in segments of the database market where it has not been a leader.

Publishing (content management)

XML has surpassed SGML as the preferred method of adding application- and vendor-neutral descriptive markup to documents. The publishing industry, where XML began, was quick to recognize the value of separating form and content, and also quick to adopt databases as a convenient way to attach metadata to documents. Because content *is* the product in the publishing business, publishers have been an early adopter of XML-based content management. At the same time, corporations across the globe are realizing that the production and dissemination of information to employees, partners and customers puts them in the "publishing" business, and encoding documents in XML is an investment in improving the efficiency of that aspect of their business.

The past several years have seen the increasing popularity of content-management systems geared toward managing various forms of content. Many content-management products are Web-specific, meaning that they manage content that gets published to the Web; others support multiple media, including mobile devices, CD-ROM and print. In general, these content-management systems, particularly those focused on Web content, have not had native XML support, so they break documents up into fields, sacrificing the ability to describe the rich hierarchies that XML allows. However, what they gain by the use of the database is an easy way to attach attributes (known as metadata) to elements stored at the field level. As content management continues to evolve, organizations will seek a way to extend the repository to embrace the richness of XML, including not only coarse identification of elements but also fine-grained tagging, such as identifying unique terms (*e.g.,* company names or glossary terms) inside a paragraph or list.

Content-management servers support the creative and production processes associated with material that is authored for publication. Content-management servers typically wrap workflow, status tracking and library services around a database. Elements of these systems include integration with the authoring process, maintenance of folders and file abstraction, integrated repository functionality (including versioning), workflow integration, extensible metadata, and support for both SQL and full-text querying.

The complete Oracle9*i* package (Oracle9*i* Database and Oracle9*i* Application Server) provides most of these core content-management functions, and the new XML facilities in Oracle9*i* Database Release 2 strengthen them with key underlying functionality that will be critical to emerging content-management implementations based on XML Schema. In addition, Oracle XML DB's support for Web transport protocols (FTP, HTTP and WebDAV) enables common desktop authoring tools to access and edit information in the database, removing the need for specialized client applications.

Today's market also offers a separate class of XML text servers designed for fast search and retrieval of XML documents. Against these products Oracle offers its own Oracle Text full-text indexing engine, which now indexes XML text documents stored in the Oracle9*i* Database. The improvement gains that Oracle expects Oracle9*i* to provide in this area help strengthen its product as an alternative to third-party run-time XML text servers.

Messaging (application-to-application communication)

XML is at the core of the emerging Web services model of application development. Protocols such as SOAP, XML-RPC, and JMS all enable software components to communicate with each other via XML dialects. While this category of content is in its infancy, we know that in production environments messaging applications will require high throughput, rapid generation and ingestion of messages, the ability to query message payloads, extensible attributes, integrated XSLT transformation capabilities, and interfaces with standard APIs.

Oracle XML DB provides a native XMLType datatype to store messages by the terabyte. Generation and aggregation are handled via native SQL operators running in the kernel. XML Schema support allows constraining of messages. SQL (including full-text) and XPath querying methods are supported. Application server interfaces to JMS and SOAP are provided. These features, combined with optimizations (XML-optimized memory layout, XML indexing, etc.), create a high-volume messaging server that can scale to meet enterprise requirements.

Business-to-business data exchange (next-generation EDI)

A very promising application for XML is as a low-cost replacement for electronic data interchange (EDI) implementations. Structured business documents, such as purchase orders and bill presentments, can be expressed as XML documents that can be delivered asynchronously and without the need for direct application-to-application integration, as was done with first-generation EDI implementations. In

XML-based implementations, the systems are loosely coupled, rather than tightly integrated, because the data can be passed as an XML document that can be validated against a schema to ensure common definitions and enforce DOM fidelity (order of elements, namespaces, etc.) as well as to maintain fidelity to the original form of the data.

Oracle XML DB addresses this type of business-to-business interchange on several levels. In addition to the messaging support described above, XML DB supports dynamic discovery of data structures through DOM and JNDI API implementations and, at a Web services level, it provides API access for UDDI and WSDL calls to and from other applications. Applications requiring higher performance can rely on static access furnished by a Java Bean interface. The built-in XML Schema tools help ensure document validity, and the integrated XSLT processor speeds data transformations when mapping data from one schema to another.

E-business (tying legacy systems to Web applications through XML)

Increasingly, XML is being used as "glue" to bind legacy software applications to e-business front ends that deliver information to customers over the Web. A typical scenario is to transform the data in the legacy application to XML in order to hand it off to the new e-business application. As e-business projects grow in complexity, developers will want support for generating XML views over relational and other existing data. To be done efficiently, such application development requires integration with adaptors or gateways to create normalized XML views over multiple heterogeneous data sources.

With XML DB, Oracle has made it much easier to create XML views of mixed content stored in the database or accessed from other servers via Oracle gateways. It also has opened up its repository to Web protocols and, by adding the URI-Ref datatype, simplified the process of linking information in the database to external sources around the world. The new architecture gives developers tremendous flexibility in writing programmatic access to the database: The DOM interface allows developers to write applications without knowledge of the physical data structures of the database, while the Java Bean interface bolsters developers' ability to tune applications for performance when the physical data structure is known. The net effect will be that Oracle9*i* may be seen as a platform for building enterprise-level, XML-based e-business applications.

Fighting silos of XML

Widespread interest in XML has both positive and negative repercussions. On the plus side, XML helps organizations improve quality and reduce labor costs by giving them a consistent syntax for describing a variety of unstructured and structured information. The downside is that the various commercial and homegrown systems that utilize XML don't necessarily communicate with each other. Even when they do, their schemas and DTDs (their definitions of the information structure) may be inconsistent. Achieving consistency and integration across multiple systems requires extensive custom development, either to define structure and metadata up front or to retrofit translation and transformation functionality to new definitions. Either way, eliminating "XML silos" can be quite expensive.

What we are left with is a collection of independent XML data stores that are unable to communicate with each other. A Web publishing system, for example, can transform content and syndicate it to clients worldwide, but it cannot directly query the client's purchase order in the accounting system that tracks the business-to-business relationship. Such silos create a plateau of efficiency, where individual applications are optimized via XML but the enterprise as a whole is unable to elicit additional gains without increased development costs.

Integration of schema handling and other query and transformation tools in Oracle9*i* places the XML functionality inside the core infrastructure of the enterprise. The effect of bringing XML awareness to an underlying platform is that it will enable enterprises to continually reap increasing benefits from their investments in XML.

The Benefits of XML DB in Context

XML functionality ratchets up

The industry momentum behind XML has reached a critical mass that will be self-perpetuating, as all of the major suppliers of infrastructure products embrace XML and related standards as core technologies. The benefits of this trend are twofold. First, the tools will be refined to the point that data will pull away gradually from the programming logic surrounding it. Second, the skill set of developers working with XML will continue to increase. This combination of talent and industrial-strength XML support in products creates an environment where the promise of the technology can realistically be achieved. Thus, as more infrastructure and application vendors support XML and related standards, it will become increasingly easier to implement XML across an enterprise and between organizations. The cycle ratchets up with each iteration of product releases.

Oracle's strategic melding of SQL and XML technologies in XML DB is such a move upward. It gives developers a standard data model for both structured and unstructured data stored in the same repository, rather than treating the two types of data differently, as has been done in the past. This change should be a boon for shops seeking to leverage their current staff expertise in SQL for new XML-based projects. The fact that this union will provide a two-way street—SQL access to XML documents, and XML access to relational data—increases the developer's flexibility and speeds application development.

Silos of data and content consolidate

With XML DB, content-management solutions will be able to take a more universal approach than is available from the current crop of data- and document-centric XML servers. In Oracle9*i*, Oracle now offers a baseline XML repository functionality that supports editing of complex documents that mix fielded data with narrative content and other document structures, such as hyperlinks and tables. End users won't be restricted to HTML forms for editing XML documents. Instead, they'll have richer, XML-based document user interfaces, many of which will be standard desktop tools

connecting directly to the database. Programmers also benefit from a unified approach that provides efficient retrieval for both XML and SQL queries, with the least amount of coding. By harmonizing structured and unstructured objects in a single repository, Oracle XML DB lays the groundwork for unified, enterprise-wide solutions. At the same time, consolidating existing smaller departmental systems provides an opportunity to improve system administration by placing it on hardware that is easier to maintain and scale. The repository then becomes the key building block for other integration and delivery projects, such as portals or automated customer support systems. As the database gains increasing functionality, it will displace the basic file system as the core technology on which to build content-intensive applications in the future.

XML spreads to new applications

At the same time, exposing XML DB functionality as standard Java APIs extends support to the popular JSP Web-development environment. Applications such as e-commerce, Web publishing and content syndication are obvious candidates for XML DB. But we think that its utility will extend to targets in traditional non-Web applications as well, such as enterprise-application integration, process control and data acquisition.

Demand for adherence to Web standards continues

Oracle9*i* and Oracle XML DB respond to growing customer demand for vendor compliance with industry standards. Standards promote connectivity across products, and from that perspective, a critical feature of Oracle9*i* is that editing tools talk to the database via Web protocols: HTTP, FTP, and WebDAV. This use of standard protocols for connecting desktop authoring applications to the repository complements the built-in support for SOAP, which is designed for server-to-server communication. Inside the database itself, the new XML architecture of Oracle XML DB is consistent with customer demand for open methods for storing and accessing their information. With Oracle XML DB, that information may be both content and data, all unified under a vendor-neutral object model (the W3C's DOM). The recognition of XML Schema embraces open data definitions, and the new XPath navigation methods in XML DB equip customers with a XML-based view of the database that's emerging as an important alternative to SQL.

As complexity grows, so does the need to enforce integrity

As XML migrates to increasingly complex applications, users need flexible methods for ensuring the integrity of their data. Integrated support for XML schema will improve data integrity across several planes. It provides document validity, matching the data instance against the schema to ensure that all required elements are completed. It ensures referential integrity, verifying, for example, that an XML fragment refers to an element that exists and is well formed. Most importantly, it addresses semantic integrity, by allowing stored procedures to validate input based on business logic, such as matching postal codes to localities. While it's possible to do all of these things via stored procedures, the integration of XML support provided by XML DB provides a cleaner, more efficient way to navigate and manipulate data.

Conclusion

Oracle XML DB resonates with a trend that is driving IT expenditures: integrate technologies to improve performance and control costs. While integration still requires a good deal of work from consultants or in-house developers, the infrastructure must be optimized to support such connectivity, or it will not be cost effective. The technologies that must be part of this next-generation infrastructure include an integrated repository for data and content; multiple, nonproprietary methods of access; and support for the Web family of standards (HTTP, FTP, WebDAV, SOAP, etc.) to handle content and data exchange among servers and client productivity tools. The integrated and optimized XML handling provided by Oracle XML DB addresses the need for a standards-based, scalable infrastructure to support increasingly complex uses of XML.

We are moving to a new age of data portability, one in which XML will play a key role. In the past, customers often selected repository and data-processing systems with little regard for data portability. Large systems were not replaced very often, and there was little emphasis on sharing information among applications. Today, companies are paying more attention to transforming their business into e-business, and two foundations for that transformation are portable data and standards for inter-application communication. In addition, customers are concerned whether even market leaders will be able to sustain themselves in a shifting business climate. Their concerns foster unrelenting pressure on vendors to back W3C and IETF standards to ensure interoperability among products and to provide a migration path for content in the future. Ultimately, enterprises have come to recognize that the key to minimizing risk and maintaining flexibility and stability in times of rapid change is to transform their content *and* data into neutral, well-described forms that exist independent of the products and programming code used to store and manipulate the information. By melding XML data and content in a unified architecture, Oracle has taken an important leading step in helping customers achieve that objective.

The Seybold Consulting Group
428 East Baltimore Avenue, Media, PA 19063 USA
{1} 610 565-2480
www.seyboldreports.com

Mark Walter is a Senior Editor for *The Seybold Report* and a Senior Analyst with the Seybold Consulting Group, covering content management, publishing technology and XML-related software and topics. He can be reached at mwalter@seyboldreports.com

Victor Votsch is a Contributing Analyst for the Seybold Consulting Group and Principal Analyst for V-Square Solutions, focusing on strategic use of cross-media publishing technology, e-commerce, and related emerging technologies. He can be reached at victor@v-square.org.