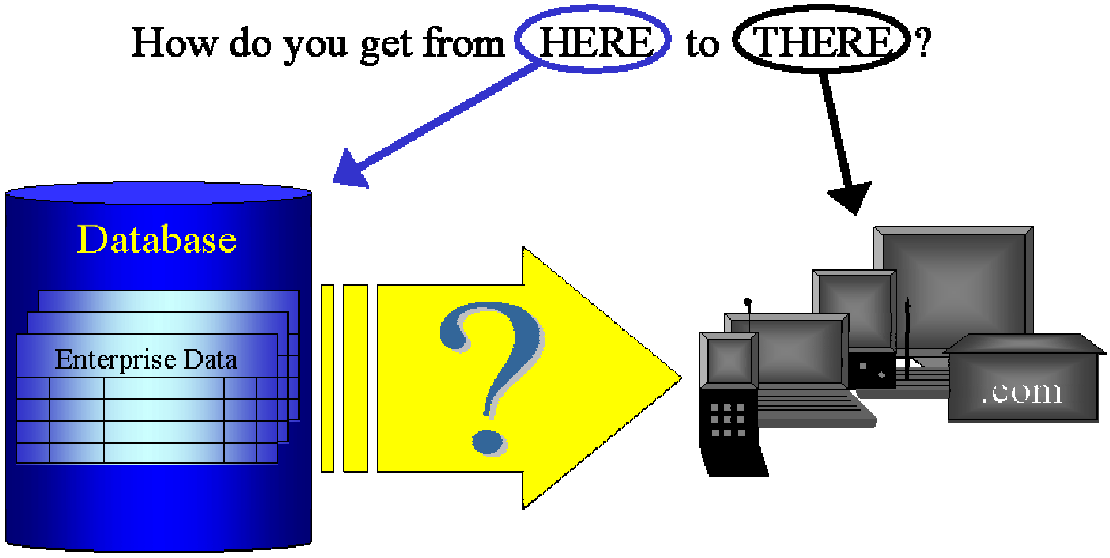
XML and Database Integration



Robert Burgess
robert.burgess@informix.com

Informix Software, Inc.
www.informix.com

Introduction

For a long time, organizations have relied on enterprise databases to maintain information relevant to the whole corporate structure. Increasingly over the last several years, corporate citizens have accessed that information through Web browsers, and data access applications have been designed to deliver database query results to this now-ubiquitous client platform. Today, however, users are beginning to adapt to new-technology client gadgets -- like cell phones and PDAs -- that have presentation and navigation capabilities very different from the "traditional" Web browser.

The critical question for the enterprise now becomes, *How can information stored in the corporate database be made available and accessible from these new devices?*

About This Paper

This discussion/demonstration touches on three currently timely topics for database vendors: XML, Open Source, and multi-client (including wireless) support.

The discussion begins with an overview of XML and explains current database issues with respect to XML inside the server. Next, the prospect of applying business logic in the middle (Apache Tomcat) tier is diagrammed and a simple application is outlined. Finally, putting these concepts together, an application is modeled and demonstrated for delivering data from the database to a heterogeneous client base; XML results from the database are transformed per the requirements of the requesting client device.

Terms

The following abbreviations appear throughout this presentation:

SGML, a standard which has been around a long time, is a language for describing tag-delimited content in a general way.

HTML (a subset of SGML) became popular with the advent of browser technology and is today the de facto standard for information presentation on the Web. HTML describes both the *content* and the *presentation* for the content.

XML (a subset of SGML) is poised to overtake HTML in importance as the de facto language for describing content on the Internet. XML describes only the content. The presentation is left up to the consumer.

A *DTD* formally describes what content is legally allowed in an XML document. It is used primarily for data validation.

XSL describes how the content of an XML document should be transformed from it's original state to another (e.g., XML-to-XML, XML-to-HTML, etc.)

XSLT is the engine that physically transforms XML based on an XSL document.

JSP is a mechanism for creating Java Servlets.

HTML and XML

Before diving into the discussion of applying XML as a enterprise application solution, it is important to understand the distinction between HTML and XML.

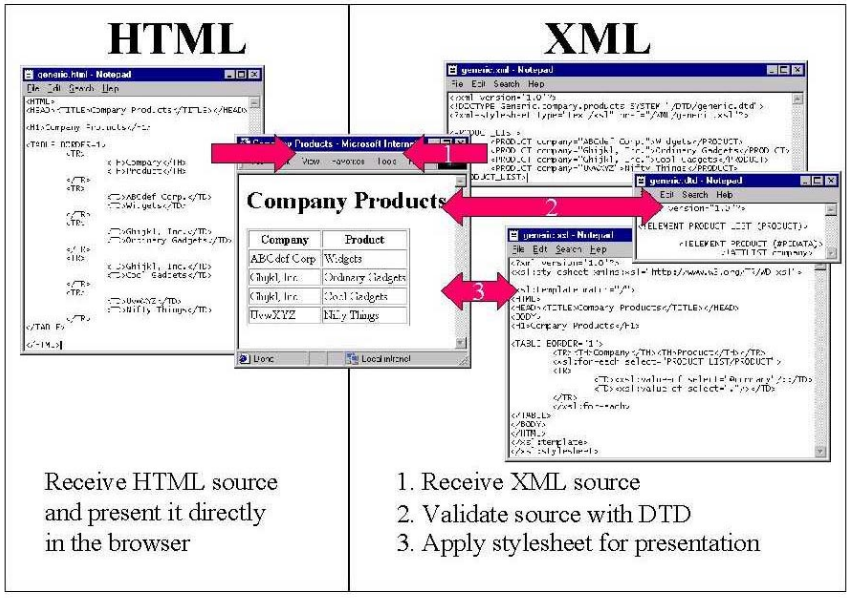
The ubiquity of HTML has made it the de facto vehicle for communicating content on the Internet. However, with its flexibility and platform-neutral information representation, XML is projected to overtake HTML in importance soon.

HTML bundles information as a combination of content and presentation, meaning that it describes not only what the information is about, but also how it should be displayed at the receiving client. XML, on the other hand, is concerned just with content, without regard for how a receiving client will treat it. Perhaps a good distinction to draw between the two is that HTML offers information in a *presentation format*, whereas XML describes information in a *representative hierarchical structure*.

HTML does not force strict syntax on all its tag elements. For instance, it is alright for an HTML document to have a beginning paragraph (`<p>`) or horizontal rule (`<hr>`) tag without the formal ending tags (`</p>`, `</hr>`). But XML enforces strict rules for ending every element tag appropriately.

The single greatest aspect of XML is its ability to transform. Clients receiving an XML document can transform the original structure of the document to a completely different structure by a process of transformation, which is achieved through applying XSL to the structure.

Although HTML and XML are designed for different purposes, it is possible to use them for a common objective. For instance, the below graphic illustrates how to achieve the same visual result on a Web browser from both HTML and XML data sources. It also highlights the various components of each technology.



To visually represent data using HTML, the browser needs just the HTML page. Remember that HTML is a packaging of content plus presentation, so the browser simply follows the rules contained in the HTML page to render the presentation.

Achieving the same result through XML is a different matter. Since XML only describes the content, the browser must perform the work of applying the presentation. A three-step process is used to reach the same result as the one-step HTML process.

First, the browser ingests the XML document. Second, to ensure the data contained in the XML document are valid, the DTD is referenced and the complete XML document is evaluated for proper structure and content. (Note: DTD validation is optional.) Third, to transform the XML document into the visual representation desired on the browser, XSL is applied to the document. The XSL document used in this case describes a resulting HTML document, highlighting the transformation abilities of XML.

Serving Content From the Database

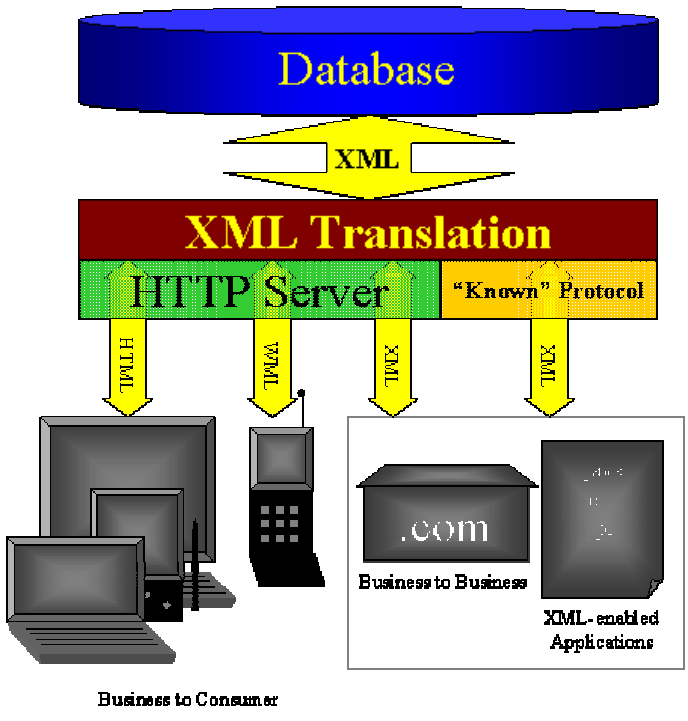
HTML Directly to Clients

Now that the differences and similarities between HTML and XML have been described, it is possible to see how the two can be used in conjunction with database results. It will soon be apparent where XML plays a critical role in the overall strategy of generating client-friendly database output.

It is fairly easy to imagine the delivery of HTML results from a database to a particular client. Results are tailored according to the client's presentation capabilities, satisfying the client's needs directly. However, note that for every unique client platform, a similarly unique result must be generated. This model may work in certain constrained environments, but it does not scale well in a more open or global environment.

XML Translation to HTML

For a given application, multiple clients access the same database. To simplify database server application development, it is reasonable to generate results that every client will understand. The best way to do this is for the database server to deliver query results as XML documents.



Since all clients ask similar questions of the database, results of a specific query are relevant to all clients, regardless of their presentation requirements. By describing these common results in XML, it is easy to apply a transformation to address the presentation needs of a specific client.

(Note: in certain circumstances, it is possible to apply XSL transforms directly in the database engine; however, the transformation may impose unnecessary burden on the database server. Hence, running XSLT inside a database should be considered carefully.)

XML As a Foundation

Although HTML clients have been used up to now to feature the usefulness of XML database results, the proposed model can be used to transform results for any number of non-HTML clients as well. Remember, *the greatest strength of XML is its ability to transform*. This axiom applies to any model where XML is used as an information transport vehicle.

XML and Databases

It has been generally demonstrated that XML database results are useful. But what does XML *in* the database server mean?

"XML in the database" must be viewed from two perspectives: getting XML out from the database, and storing XML in the database. Generating XML results is a relatively simple task, nothing more than restructuring the traditional result set as a formally structured XML document and returning the single result. There are several ways to achieve this in various database servers, as described in the next few slides.

XML Into the Database

The notion of storing XML in a relational database is fraught with complexities, although there are some rudimentary ways of doing it today. The difficulty lies in effectively managing the contents of an XML document such that indexing and searching strategies provide the performance expected from a relational database. This is a problem for all relational database vendors, and one that all are seeking to address.

There are several methods for storing XML in a relational database today, although none offer ideal robust functionality. Storing XML as a Blob or Clob is reasonable, but to perform reasonable business logic on the object requires the creation and use of user-defined routines (UDR) or other middleware solutions.

If XML is stored as and extended Text data type, SGML-like filters can be applied, but the structural hierarchy of a document may compromise the effectiveness of search strategies. Also, the organization of an XML document is completely different from that of a true text document, so text query strategies are not appropriate for XML documents.

Recently, clever tools for mapping between XML elements and a relational database schema have reached the market. These tools employ various methods for making associations between XML elements and database fields, modeling the relationship in software (usually a middle tier), and persisting a run-time component that manages the transfer of data between the application and the database.

Although it is possible to create an effective XML storage solution with today's technology, database vendors have yet to offer a strategy that implements "XML as XML" in the database while delivering the performance expected from a relational

query. Ideally, the data should look like XML to the database user, with query results as true XML. Inside the database, the document must be stored in a way to facilitate indexing and, therefore, quick searching. The language for querying XML via SQL has yet to be determined, standards bodies are working on such specifications and it is expected that products will support whatever language is adopted as a standard.

XML Out From the Database

Generating XML results from SQL queries is very easy to do with any database engine; the capability dose not provide significant differentiation for any database vendor. Some vendors provide XML results as an extension of the structured query language (SQL). Others provide it as a feature of database access tools like JDBC and ODBC. A database schema-to-XML representation can take a variety of forms, each of which are valid and acceptable. The simplest representation to make is essentially a one-to-one mapping from schema tables and columns to XML elements.

Open Source

The feasibility of retrieving database query results as XML has now been shown. But the query result is useful only if the requesting client can understand how to make sense of the XML document. In most cases, the query result will need transformation before it is truly useful to the client.

The Open Source community as an excellent resource for quality tools and components for newer technologies like XML. Not only does Open Source software have a broad and eager installed base, the software itself evolves and matures much more rapidly than proprietary software simply because of the cooperative nature of Open Source development. Adopters benefit from the growing popularity and credibility of Open Source code as enterprise components because they generally contribute to integration with best-of-breed solutions.

Apache Open Source

Perhaps the best known Open Source effort in XML technology is Apache Software Foundation. The demonstration referred to in this document incorporates Apache components in the architecture. Certainly any comparable components may be applied to this architecture, but for the purposes of this discussion Apache will be used.

Three Apache components are used in the architecture of the demonstration application.

Apache Tomcat basically functions as a Web server, accepting and processing URL requests. As a JSP implementation, it is specifically suited to manage Java servlets that are initially defined as and subsequently compiled from JSP.

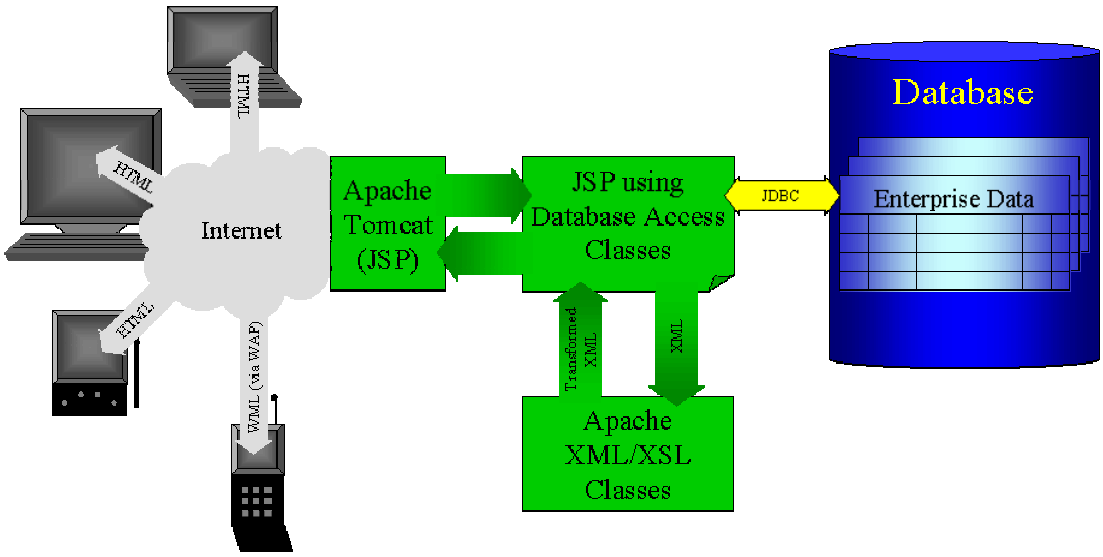
Apache Xerces is a project code name for an XML parser, used for evaluating raw XML documents. XSL requires services from the XML parser.

Apache Xalan is a project code name for an XSLT engine, used for transforming XML documents from its original form to another.

Demonstration Architecture

The architecture described here employs a middle tier to perform most of the business logic for the application to be built. The "Internet" portion of the model implies that the application is generally available and any client on the Internet can make a request for information. On the other side of the model, the "Database" indicates where the core information resides.

Between the "Internet" and "Database" is the middle tier, consisting of Apache components, JavaServer Pages, and data access facilities. Regardless of how XML is extracted from the database, JDBC is the connection mechanism between the application and the database.



Apache Tomcat functions as a Web server, receiving requests for information from "Internet" clients. Tomcat references the appropriate Java servlet, which interacts with the database to retrieve query results and passes the results through XSLT for transformation appropriate to the requesting client. The decision of which stylesheet to apply is determined by the client's "user agent" (browser type). When transformation is complete, the result is handed back to the client via Tomcat.

The complete construction of this model includes HTTP, JDBC, and Open Source convenience and helper classes, and the exercise is left for the reader. This simple yet effective design is appropriate for basic enterprise solutions. These core pieces can easily be extended to built enterprise applications -- primarily by adapting database queries and creating application-specific XSL designs.

Summary

This presentation outlines just one possible approach to addressing the need for delivering enterprise information to disparate clients. An XML database query result is easily transformed for any purpose. The combination of an enterprise database and Open Source is a good platform for delivering on these needs. Incorporating these concepts, users employ any number of devices to access information of interest to themselves.

XML, in conjunction with XSL, is an excellent way to address the needs of users while preserving the integrity of existing systems. With XML and XSL, it *is* possible to get your enterprise data from "here" to "there".