XML: Changing the data warehouse

Deliver new levels of business analysis and bring users closer to their data
Executive summary
Some saw it coming years ago. Others are just beginning to realize what lies ahead. We’re talking about the impact that XML is having on data warehousing environments.

For IT leaders building data warehouses that meet the evolving demands of their business environments, integration of XML data into their infrastructures is critical. With XML, organizations can support evolving business reporting and analysis requirements without incurring significant database schema changes or rewriting applications. Simultaneously, they can bring their users closer to the information that they need to make accurate business decisions while providing the user with a result set that is similar to a search engine on the Internet.

XML has become the preferred data exchange format across many industries. As a result, organizations must find ways to efficiently manage and manipulate XML within their data warehouses. In fact, leading-edge firms in retail, finance, healthcare and other industries already have production environments that leverage both relational and XML database technologies. While their implementations vary, early adopters are often striving to promote greater business agility, providing decision makers with more accurate and timely information and improving IT staff productivity.

IBM® DB2® pureXML® makes it possible for organizations to manage XML data as well as relational data. This increases database efficiency, improves the user experience and increases their competitive advantage by fully exploitign data interchange standards.

Data warehouses and evolving business needs
One of the primary goals of data warehousing is to make it as easy as possible for users to get the information that they need when they need it. However, traditional relational schemes can make this difficult to achieve.

Consider a data warehouse in the retail industry that tracks sales information. Using a typical star schema database design, as pictured in Figure 1, a “fact” table might contain sales data by product, region and time period. Such data would typically be joined with data in multiple “dimension” tables to obtain specific details pertaining to various products, regions and so on. Unfortunately, different products have different attributes, which presents challenges both in database design and in figuring out how to expose the data to users.

With a relational-only design, each product attribute must be captured in its own column. But because attributes vary widely from one type of product to another, the result could be an inefficient, unwieldy product table with thousands of sparsely populated columns. And, as new products and new product attributes are introduced over time, the database schema—and any applications that depend on it—would need to be altered, which can be quite costly.
Augmenting a data warehouse schema with XML

![Diagram](image)

Figure 1: In a typical star schema warehouse, millions of transactions—known as facts—are in a purchase table. This table joins via keys to dimension tables that store attributes for time, store location and product (left). Using XML data warehousing methods, the organization captures additional product attributes as a single XML column in the product dimension table (right).

Presenting this data to decision makers is a challenge in this environment. Programmers and database analysts must determine which attributes to expose to which decision maker’s application or executive dashboards—not an easy task. Also, business analysts may want to “drill down” or “slice-and-dice” product sales data in unexpected ways, such as exploring sales of women’s shirts by size, color, fabric, neckline and sleeve length. But investigating warehouse data in this way requires an intimate knowledge of how the warehouse schemas are constructed. For example: Are leather coats filed under outerwear, fall/winter or designer collections? There’s often no easy, efficient or effective way in today’s table-based warehouses for developers to create a search function that works like a Web search: Type in a term and all relevant results are returned.
Extending a relational data warehouse schema with one or more XML columns avoids those problems. Commonly used attributes can be stored in relational columns, while additional details can be maintained in an XML column, which readily accommodates variable structures and is easily accessible for queries and reports. Using the previous example, the dimension table for product data could be extended with an XML column. New product attributes would simply be included as new elements in the appropriate XML documents. The database schema would not need to change.

Although this example focuses on a retail scenario, similar scenarios apply to other industries. Medical records, financial instruments, employee profiles, traveler profiles and income tax data are just a few examples of information that has a wide range of possible attributes able to change over time.

Real-time analysis in retail

In the United States, a popular retailer captures point-of-sale data in XML and posts this data to message queues: These sales records represent important real-time business data that analysts need to review along with historical information about sales.

Using a popular extract/transform/load (ETL) offering to transform this data and load it into a relational-only data warehouse proved time consuming to develop and failed to meet the firm’s need for real-time analysis. Instead, this retailer built an operational data store using DB2 pureXML. Stored procedures read transactional XML data from queues and store the data intact into a DB2 database. Analysts can now combine data about intra-day sales (managed by this operational data store) with historical data previously loaded into the data warehouse. Periodically, operational data is mapped into tables and loaded into the warehouse. The firm reduced their ETL development cycle by several months and provided real-time analysis capabilities to their business users for the first time.
Why XML?
As more and more critical business data is captured and exchanged in XML, it’s not surprising that firms are recognizing the need to manage, share, query and report on XML data. Message-based applications, Service Oriented Architectures (SOA), Web-centric applications and application integration projects increasingly rely on XML to define how important business data is represented and exchanged, and industry-specific XML standards abound. Examples include the Financial Products Mark-up Language (FpML) for over-the-counter derivatives trading, Health Level 7 (HL7) and Clinical Data Interchange Standards Consortium (CDISC) specifications for healthcare, Association for Cooperative Operations Research and Development (ACORD) specifications for insurance, Financial Information Exchange Markup Language (FIXML) for securities transactions, ISO 20022 for banking payments, Standards in Automotive Retail (STAR) for automotive manufacturing and others.

The increased use of XML standards for data interchange creates storage and management challenges; the highly variable, nested structures that typify XML data are difficult to accommodate using traditional relational database techniques. One approach to storing XML in a traditional relational database is using large objects to manage XML, which prevents the database management system (DBMS) from understanding the internal structure of stored data. Therefore, it can’t provide optimized access to specific XML elements or attributes contained within a message or document.

Some firms prefer to “shred” or decompose XML data into multiple columns of one or more tables. In doing so, they “flatten” the XML hierarchy and convert the data values of XML elements and attributes into traditional SQL data types, such as an integer or varying-length character strings. But, a single XML message structure (XML schema) may need to be mapped to dozens, even hundreds of tables, driving up application development and database administration costs. Those complex, labor-intensive mappings are difficult to adjust as XML messaging formats change over time. Finally, reconstructing the data into an XML format for downstream applications can be slow and complex.

For those reasons, and others, many firms are storing XML in its native hierarchical format alongside relational data so that both types of data may be managed in an optimal manner. With this approach, XML data is stored intact with full DBMS knowledge of its internal hierarchical structure. IBM DB2 supports native XML storage alongside its relational storage, which helps organizations manage, share, query and report on data modeled in tables as well as data contained in XML hierarchies. Labor-intensive document decomposition and reconstruction processes aren’t needed. In addition, certain performance advantages and programming productivity enhancements are possible, thanks to greater DBMS knowledge of XML technology.
Deliver new levels of business analysis

Improving productivity with XML

A European financial services firm adopted a SOA to promote better customer service across its various lines of business (LOB). The firm defined a standard XML messaging format for data exchange among applications and needed to maintain an operational data store to manage this data. After comparing DB2 pureXML to relational technologies, the firm concluded that it could improve the productivity of its Web service developers and database administrators through pureXML.

DB2 pureXML

IBM DB2 provides firms with a common application programming interface and database management platform for data modeled in tables as well as XML hierarchies. This hybrid database management architecture, pictured in Figure 2, helps firms to extend their traditional relational database environments to directly manage XML messages and documents without the need to shred data into columns of various tables. Applications can retrieve relevant portions of the XML data easily and efficiently, as well as integrate XML and relational data with little effort.

Figure 2: DB2 9 architecture with built-in support for relational and XML data helps extend traditional relational database environments.
DB2 pureXML includes many sophisticated features to provide high levels of performance and scalability for query-intensive workloads common in data warehousing environments. Such features include:

- **Cost-based query optimization**—helps enable DB2 to select an efficient path for accessing requested data
- **Specialized XML indexing**—speeds retrieval of queries over XML data as well as queries over relational views of XML data
- **Hash-based partitioning**—provides significant scalability gains
- **Range-based partitioning**—helps firms “roll in” and “roll out” data over time (a common requirement in data warehouses)
- **Multi-dimensional clustering**—often improves performance of analytic queries
- **Compression of XML data and indexes**—reduces storage costs, improves storage efficiency and speeds runtime performance for many common workloads

Recent benchmarks conducted by IBM and Intel confirm the strong runtime performance, throughput and efficiency of running DB2 pureXML on Intel® Xeon® processor 5500 series. In addition, DB2 helps enable firms to easily create relational views of their XML hierarchical data, which allows relational tools and applications to access stored XML data in a straightforward manner. Indeed, popular query/reporting tools, such as IBM Cognos®, benefit from DB2’s ability to dynamically transform XML structures to relational result sets when needed. Finally, many extract/transform/load tools, such as IBM InfoSphere” IBM DataStage® and IBM Infosphere Warehouse Design Studio, can access pureXML data through such views or through built-in wizards.

**Summary**

Increased use of XML as a preferred format for data exchange is prompting data architects and administrators to evaluate options for integrating business-critical XML data into their IT infrastructures. Sophisticated features for indexing, query optimization, compression and physical database design options provide for strong runtime performance and scalability. Also, various complementary software offerings can readily access pureXML data, providing firms with an easy means to integrate pureXML into their existing IT infrastructures.

**For more information**

For more information on DB2 pureXML, visit [ibm.com/software/data/db2/xml](http://ibm.com/software/data/db2/xml)
1 Intel and IBM Collaborate to Boost Performance and Lower Power Consumption
download.intel.com/business/software/testimonials/downloads/xeon5500/ibm_db2.pdf
May 2009