XML's Impact on Databases and Data Sharing

The Extensible Markup Language reduces the obstacles to sharing data among diverse applications and databases. However, understanding XML's benefits requires evaluating which system challenges it actually solves.

Len Seligman
Arnon Rosenthal
MITRE Corporation

The Extensible Markup Language, HTML's likely successor for capturing much Web content, is receiving a great deal of attention from the computing and Internet communities. Although the hype raises unrealistic expectations, XML does reduce the obstacles to sharing data among diverse applications and databases by providing a common format for expressing data structure and content. Although some benefits are already within reach, others will require new database technologies and vocabularies for affected application communities.

HTML DATA-SHARING DILEMMA

The Web has greatly facilitated the sharing of data across distributed, heterogeneous hardware and software environments. Rather than having to search for data from only one location, businesses can now use a browser interface to access data from sources around the world. Unfortunately, HTML, the predominant format for Web and intranet publishing, has several serious shortcomings that limit its use for representing information from diverse sources.

- *Presentation rather than content orientation.* HTML uses presentation-oriented markup tags, such as <H2> for a second-level heading, that tell a browser how to display data to human users. However, it gives no information about the data's meaning—for example, "this is a warranty description for a retail product." Because HTML focuses on the computer-to-human interface, it has limited value as a data format for computer-to-computer applications such as transferring information between databases. In addition, because it tightly couples content and presentation, HTML does not effectively support alternate presentations of the same underlying content for different audiences or media.
- *No extensibility.* HTML has a fixed set of markup tags. It lacks support for creating new, application-specific tags—for example, Patient_ID for medical applications—that help communicate data content.
- *No data validation capabilities.* HTML does not help applications validate data as it is entered or imported.

Although instrumental in creating a new perception among businesses that data can and should come from many diverse sources, HTML is poorly suited for building systems in which applications, not users, interpret the data. Because of these limitations, building and maintaining complex data-access applications—such as a comparison-shopping agent—based on HTML documents is cumbersome. HTML-based applications require brittle, handcrafted code to screen-scrape information from Web pages—for example, "find the third column and second row of the fourth HTML table in this page; that's usually the price."

Such convoluted techniques are especially frustrating given that much Web content derives from structured databases. Structural information would vastly simplify data extraction by applications, but content

Table 1. Advantages of XML over HTML.

Feature	HTML	XML
Extensibility	Fixed set of tags	Extensible set of tags
Presentation/content	Tags for presentation only	Tags describe data content
Views	Single presentation of each document	Multiple views of the same document (provided by XSL)
Document/data orientation	Document orientation only	Support for documents plus extensive infrastructure for exchange and validation of structured data
Search/query	Search only	Search plus field-sensitive queries and later update

providers discard most such information when they publish the data in HTML—or via dynamic scripting languages like ColdFusion and Active Server Pages.

ENTER XML

To address HTML's limitations, the World Wide Web Consortium (W3C) created XML, a language similar in format to HTML but more extensible. This new language lets information publishers invent their own tags for particular applications or work with other organizations to define shared sets of tags that promote interoperability and that clearly separate content and presentation.

XML is a simplified subset of the earlier, document-structuring Standard Generalized Markup Language. Developers have used SGML to create large information collections such as encyclopedias and multivolume case law books, but its complexity has discouraged widespread adoption. In addition to being extensible, XML addresses only content; Cascading Style Sheets, Extensible Stylesheet Language, or Extensible HTML handle presentation separately.

XML also supports validation in two ways. Application developers can associate an XML document with a document type description (DTD) that describes the structure to which the document should conform. In addition, because DTDs were intended for document management and cannot adequately model complex data, the W3C subsequently developed an XML schema specification, which adds data types, relationships, and constraints. Applications can use off-the-shelf XML parsers to validate imported data for conformation to a DTD or schema.

Widespread acceptance

Although a young standard, XML already exerts significant influence on intranets and the Web. Businesses appreciate its elimination of the many costly and fragile workarounds needed to represent rapidly changing data in HTML. A vibrant XML marketplace is providing inexpensive tools for preparing, validating, and parsing XML data. Application developers praise XML's extensibility, and communities that share common data, such as the chemical industry, like XML's support for well-defined, common data representations. XML should continue to have strong support throughout the next decade.

XML benefits

As Table 1 shows, XML offers many advantages over HTML.

- *Support for multiple views of the same content for different user groups and media.* As Adobe chairman John Warnock said in his keynote address at XML 98, "To date we have had a separate workflow for each output format.... We are switching to XML because it will allow us to have a single workflow with multioutput."
- *Selective (field-sensitive) queries over the Internet and intranets.* For example, a search for documents with an author field containing "Kissinger" would only return documents that mention Kissinger within an author tag. This capability depends on agreements within communities on the meaning of certain widely used tags.
- *An increasingly visible semantic structure for Web information.* This will decrease the need for brittle screen-scraping parsers.
- *A standard data and document interchange infrastructure.* This infrastructure includes freely available parsers that can validate conformance with a DTD.

Several related standards will greatly increase XML's data sharing and management utility. For current information on these standards and their supporting tools, see http://w3.org/xml and http://www.xml.org.

Extensible Stylesheet Language. XSL expresses rules that indicate how to transform an XML document to a presentation format such as HTML or PDF, or to an alternate representation of the content such as an XML document with a different DTD. Developers can manage content independently of its presentation, and they can use different XSL style sheets to produce alternate views of that content.

Document Object Model. The initial XML standard gives enough information to drive a parser but does not specify the parser's output form, either as a data structure or in terms of operations. We predict that developers

Will XML “Disappear”?

Paradoxically, the Document Object Model and Extensible Markup Language query language will reduce the use of XML text. DOM defines a standard wrapper for XML text, above which most services can work—for example, XSL, constraint checking, and even linking care about abstract elements, not uninteresting syntactic details such as “</”. These services can use an off-the-shelf parser that produces DOM instead of each parsing XML to its own internal form.

Services must next recognize that a giant XML text string or Character Large Object within a database management system is an inefficient representation for a large, complex, updatable structure, which requires indexes, clustering, free-space management, transactions, and so on. Vendors can replace stored XML text with their own specialized storage formats under the DOM or query language abstraction.

Increasingly, tools will access document set abstractions through standard interfaces, as Figure A shows. In the end, XML text might be used mainly for document management applications and at the interfaces between loosely coupled systems—for example, for data exchange over the Web.

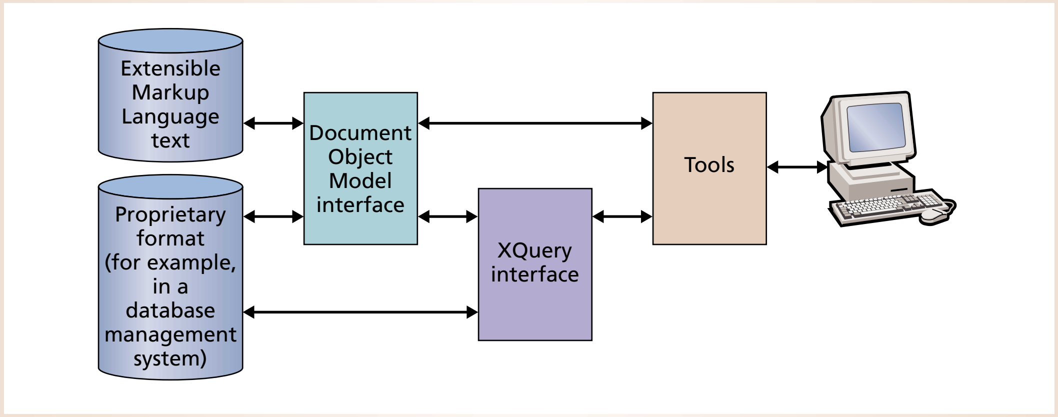
Extensible Markup Language text
Proprietary format (for example, in a database management system)
Document Object Model interface
XQuery interface
Tools

Figure A. Abstract interfaces replacing XML text. Web developers will increasingly access XML document collections through standard interfaces such as Document Object Model and XQuery, a query language the World Wide Web Consortium is developing.

will express most XML-related work in terms of tree and graph abstractions that hide such details. The Document Object Model (DOM) provides a tree-based application programming interface to XML with methods for traversing the tree, such as getParentNode() and getChildNodes().

XML query language. A W3C working group is developing the XQuery language for extracting data from XML document collections as well as encapsulating non-XML data via mappings (see the sidebar, “Will XML ‘Disappear’?”).

XML AND DATABASES

XML and database technology complement rather than compete with each other. Because XML makes the structure in nontabular data explicit, database technologies can provide some of the amenities found in relational databases. Conversely, database techniques can improve the integrity and semantic integration of XML resource sets. The research community and W3C working groups recognize this synergy and are adapting database ideas to provide XML schema and query technologies.

Well-structured data

Today’s broad, mature database management systems (DBMSs) will dominate critical enterprise data management in the foreseeable future. They are rapidly widening their scope to serve newer areas—for example, electronic commerce has become a major revenue and development focus. DBMSs offer high integrity, read, write, and—increasingly—subscribe-to-changes processing of large amounts of regularly structured data. Data that supports critical but routine processing will continue to require these features.

DBMS facilities include highly tuned query and transaction processing, recovery, indexes, integrity, and triggers. DBMSs exploit the relational data model—regular tables, no queries over element tags, weak support for paths—to simplify semantics and improve performance. They even optimize load and dump util-

Relational systems often use highly tuned indexed structures rather than simply storing XML as text.

ities. Providing similar functionality over XML's more intricate structures is more complex.

For applications involving regularly structured data, XML tools will not replace such DBMSs because there is too much functionality to implement rapidly and migration would be too traumatic. Still, XML is rapidly gaining a role as an interface format for even highly structured data.

To meet specific Web publishing, e-commerce, or other application demands, developers can create XML versions of appropriate data views. Applications can evaluate these data views on the fly in response to queries—or in advance when handling nonvolatile data or supporting users who do not need completely current information. Major vendors such as Oracle and IBM have already released tools for creating XML database extracts, and these tools will become even more powerful. Vendors are also customizing import utilities to accept XML.

User organizations are committing to XML, with some already beginning large-scale implementation. For example, the US Air Force Global Combat Support System uses XML messages among applications. In addition, the US Department of Defense is developing a registry, including tools and process guidance, for XML components—schemas, elements, attributes, document type definitions, style sheets, and so on—for its Joint Technical Architecture.

Publishing database contents as XML has other benefits. The XML output includes its own schema information; for anyone who understands the tags the schema uses, the information describes itself. Also, by keeping part of the format open, XML's schema reduces the need for multisite systems to simultaneously migrate to a new interface. If an information provider inserts new tags, sites equipped to use the new information can do so while parsers for other sites will ignore the new tags.

Semistructured data

Relational DBMSs contain only a fraction of the world's data, for several good reasons. Data must be tabular and conform to a prespecified schema, which promotes integrity but discourages rapid development and change for irregular data or data with a rapidly evolving structure. Further, DBMS purchase prices are often high, and they tend to require professional administration. Semistructured data models address all but the administration issue, but they currently lack the features needed for robust systems.

Managing semistructured data. As semistructured data becomes more widely shared and its processing more automated, organizations will need the capability to manage it through powerful queries, integrity, updates, and versioning. Applications can store XML data directly in relational systems by encoding its graph, but relational operators are insufficient for the manipulations users want.

Object-oriented database vendors such as Poet (http://www.poet.com) and eXcelon (http://www.exceloncorp.com) address this need by extending their capabilities to support XML. (Formerly ObjectDesign, eXcelon has redefined its identity to focus on XML data management rather than object databases.) Relational systems are also increasing their support of XML. To achieve efficiency, these products often use highly tuned indexed structures rather than simply storing XML as text.

The future. The database research community provides the best indicators of the long-term direction for database support of XML and other semistructured data. Many researchers are addressing the challenges of interfacing with and managing semistructured data.[1] For interfacing, wrappers can mine data with implicit structure and make the structure explicit and machine-readable.[2,3] Other projects[4] have investigated the use of graph-structured data models—such as that underlying XML—as a common representation for heterogeneous information sources, including both structured and semistructured sources.

Finally, several groups are developing prototype DBMSs that manage semistructured data with new query languages and optimization techniques.[5,6] These researchers have converged on the use of graph-structured data models, especially XML, in which labeled, directed graphs represent all data.

Research prototypes. Graph-structured DBMSs handle semistructured data from ordinary documents, Web sites, biochemical structures, and other data difficult to describe with a fixed schema. Data may be irregular or the structure may evolve rapidly, lack an explicit machine-processable description, or be unknown to the user. Even a known structure can appear hierarchical, which makes having operations that understand the hierarchy advantageous.

Compared with an ordinary relational or object database, semistructured databases offer several capabilities.

- *Irregular structure*. For example, a short string such as Good, Bad, or Ugly can describe the attribute Weather for one data source, while another might provide a collection of tuples—date, time, temperature, humidity, wind speed, and remarks. Document data often varies in structure, especially if assembled from multiple sources. The structure can also change—for example, by adding figures. Relational systems can model some irregularity by having missing attributes as nulls, but SQL's null values cause awkwardness, while current relational database storage structures can have excessive overhead.

- *Tag and path operations.* Conventional database languages allow manipulation of element values but not element names. Semistructured databases provide operators that test tag names—for example, "find all documents that have a ReferenceList or Bibliography element." They also include operators that manipulate paths. For example, path expressions with wild cards can ask for a Subject element at any depth within a Book element.
- *Hierarchical model.* Some data is most naturally modeled as a hierarchy. For this data, hierarchical languages simplify data manipulation.
- *Sequence.* Because document sections, unlike tables, are ordered, they must represent sequence. Sequence complicates query processing, especially for joins and updates.

Research prototypes have demonstrated these features, which will likely appear in commercial products in the next few years. How the market will segment among the three approaches—layered over an object database, over a relational database, or directly over a new data manager—remains unclear.

XML AND DATA SHARING

Some industry observers have heralded XML as the solution to data-sharing problems—for example, one observer asserted that XML together with XSL will bring "complete interoperability of both content and style across applications and platforms."[7] In reality, XML technologies will contribute only indirectly to meeting many of the toughest data-sharing challenges.

Architectures

Users want seamless access to all relevant information about their domain's real-world objects. Several general architectures and hybrids are available for this purpose.

- *Integration within the application.* An application or Web portal uses each source's native interface to communicate directly with source databases and reconciles the data it receives.
- *Data warehouses.* Administrators define a global schema for the shared data. They provide the derivation logic to reconcile data and pump it into one system; often the warehouse is read only, with updates made directly on the source systems. As a variation, data marts give individual communities their own subsets of global data.
- *Federated databases.* These virtual data warehouses do not populate the global schema. Instead, the source systems retain the physical data, and a middleware layer translates all requests to run against the source systems.
- *Messaging.* One application or database uses structured messages to pass data to others. Enterprise application integration products tend to support this architecture.
- *Parameter passing.* One application invokes another and passes data as parameters. EAI products also support this architecture.

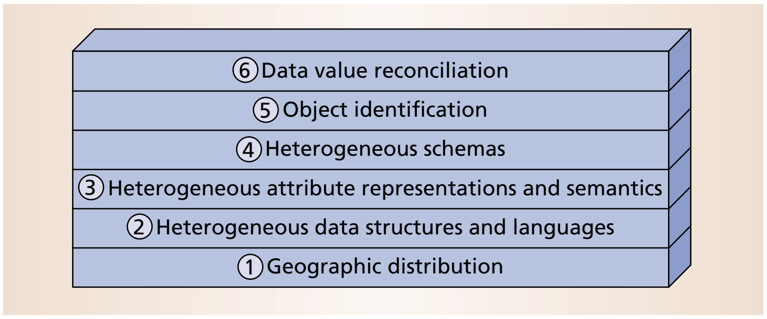
6 Data value reconciliation
5 Object identification
4 Heterogeneous schemas
3 Heterogeneous attribute representations and semantics
2 Heterogeneous data structures and languages
1 Geographic distribution

Figure 1. Levels at which data reconciliation must occur. Reconciliation generally must address challenges in order from the lowest to the highest level.

Challenges

Regardless of the distributed architecture chosen, someone—a standard setter, application programmer, or warehouse builder—must reconcile the differences between data sources and the consumer's view of that data so users can share it. This reconciliation must insulate applications from several forms of diversity. The insulation mechanisms also provide an interface for programmers to look beneath and see the diversity.

Data reconciliation must overcome challenges at multiple levels, as Figure 1 shows. Typically, reconciliation must address these challenges in order, from lowest to highest. For example, unless the reconciliation meets the challenge of geographic distribution, the lowest level, resolving heterogeneous data structures—the next-higher level—will yield little benefit.

Level 1: Geographic distribution. Data can be widely distributed geographically. Off-the-shelf middleware products handle most of the challenges at this level, often supporting standard protocols such as HTTP, the simple object access protocol (SOAP), or the common object request broker architecture.

Level 2: Heterogeneous data structures and languages. Diversity here includes different data-structuring primitives—such as tables versus objects—and data manipulation languages—such as SQL versus a proprietary language versus file systems with no query language. Standards such as open database connectivity (ODBC) and middleware products increasingly handle this difficulty. However, the middleware can be costly, lack advanced features such as triggers, and be inefficient compared with native interfaces.

Level 3: Heterogeneous attribute representations and semantics. Integrators often must reconcile different representations of the same concept. For example, one system might measure altitude in meters from the earth's

Table 2. XML's contributions to data sharing.

Level	Challenge	Contribution
1	Geographic distribution	Assists with remote function invocation—for example, via SOAP
2	Heterogeneous data structures and languages	Provides convenient, neutral, self-describing syntax for heterogeneous data structures
3	Heterogeneous attribute representations and semantics	Provides convenient way to attach and reference metadata that describes data representation and semantics Ubiquitous Web infrastructure eases compliance with data standards
4	Heterogeneous schemas	Rich environment facilitates building tools that encourage resource reuse Makes it easier to map to some well-understood schema Mechanisms for expressing interschema mappings, such as the Common Warehouse Metadata model, can leverage XML
5	Object identification	Provides a convenient mechanism for attaching metadata
6	Data value reconciliation	Provides a convenient mechanism for attaching metadata

surface while another measures it in miles from the earth's center. In the future, application developers may define interfaces in terms of abstract attributes with self-description—for example, Altitude (datatype=integer, units=miles). Mediators can use these descriptions to shield users from the representational details.[8]

Differences in semantics offer greater challenges than representation heterogeneity. For example, two personnel systems include an Employee Compensation attribute. One might be gross salary plus annual bonus, while the other is net salary after taxes. Transformations can sometimes resolve such differences—for example, rederiving gross salary. However, automated transformation often is impossible, and the integrator must simply indicate whether it is possible to use a particular attribute for a particular purpose.

Level 4: Heterogeneous schemas. Systems can assemble the same information elements into many different structures. For example, one system might store all customer account information in one denormalized table, while two others split it several ways among several tables. Each has chosen a schema that is natural for its own use rather than one designed for interchange with other systems. Many applications communities are addressing this challenge by defining standard interface schemas, expressed as Unified Modeling Language models, XML schemas, or SQL tables. Such standards reduce the number of external interfaces a system must support.

Level 5: Object identification. Object identification determines if two objects, usually from different data sources, refer to the same real-world object. For example, if the CriminalRecords database has "John Public, armed robber, born 1 Jan. 1970" and the MotorVehicleRegistry database has "John Public Sr., license plate JP-1, born 9 Sept. 1939," should a police automobile-check view consider the tuples to refer to the same person and return—"John Public, armed robber"?

Level 6: Data value reconciliation. After object identification, the different sources can disagree about particular facts. Suppose three sources report John Public's height to be 180, 187, and 0 centimeters, respectively. What value or values should the search return to the application?

Reconciliation at this level can require detailed application knowledge. "Data-cleaning" researchers and vendors are increasing their efforts to help administrators specify the desired policy, semiautomatically identify candidate objects to be merged, and—if cost-justified—resolve individual instances.[9] Reconciliation rules should be flexible, modular, and displayable to domain experts who lack programming skills.

Where can XML help?

Given these challenges, how can XML help improve data sharing? As Table 2 shows, using XML and related tools often eliminates the problems associated with heterogeneous data structures. In addition, data administrators can use XML to express results that help at other levels, especially levels 3 and 4.

Level 1: Geographic distribution

XML indirectly assists with distribution by supporting mechanisms for remote function invocation across the Web. For example, SOAP specifies an XML vocabulary for representing method parameters, return values, and exceptions (http://www.w3.org/TR/SOAP). Data sharing also requires functions that actually create, send, and read interchange files. For example, application developers must know the syntax and exact semantics for "Send." Database-oriented data sharing can use submittal protocols like ODBC. XML does not provide these functions, but middleware vendors will likely layer them on top of XML-based invocation mechanisms such as SOAP.

Level 2: Heterogeneous data structures and languages

XML provides a neutral syntax for describing graph-structured data as nested, tagged elements with links. Because developers can transform diverse data structures into such graphs, XML—along with DOM and XQuery—provides the operations users need to access these heterogeneous data structures. Microsoft's ODBC and OLE DB offer analogous functionality for accessing flat and nested data sources as well as a model for describing server search capabilities at fairly low cost.

Although applications can use XML for relational data, it really shines in other settings. When a source or recipient views the world hierarchically—for example, as formatted messages—XML technologies can help restructure the information between relational and hierarchical formalisms. For example, the US military and its coalition partners are transitioning their Message Text Format to an XML-based infrastructure. XML's strong base of freeware and commercial tools affords flexibility at greatly reduced development costs. In another example, XML provides a useful common representation for integrating semistructured text data sources.[3]

Observers who point to XML as a panacea for interoperability usually refer to level 2. For the purpose of representing data structures, XML—right out of the box—provides both a representation and, with its current and future query languages, a manipulate-transform capability. A recipient can reassemble, in the form of a labeled graph, the same data structure sent. Many organizations and data exchange standards[10] employ XML in this way, thereby removing the obstacles to interoperability at this level. Interpreting the meaning of the graph is a substantial task for subsequent levels.

Level 3: Heterogeneous attribute representations and semantics

This level deals with atomic concepts. Transmitting a fact between systems requires relating each system's semantics as well as their representations. The computer does not need to "understand" either the source or target concept; rather, it only needs to know whether they are identical or how to convert them. XML provides a convenient mechanism for attaching descriptive metadata to both source and target schemas' attributes.

For semantics, the key is knowing whether the source concept is good enough for the target, not necessarily that two concepts mean the same thing. For example, an instrument landing system might measure altitude from the current lowest point of the aircraft, but any part of the aircraft can suffice for air traffic control.

When a source and target database disagree about representation, each should explicitly describe representation details, for example:

```
<Altitude>38500
  <LengthUnit>feet</LengthUnit>
  <MeasurePoint>lowest</MeasurePoint>
</Altitude>
```

Standards should make this subsidiary information sharable—for example, by clarifying the meaning of "LengthUnit." The descriptions determine what transformations are necessary. Increasingly, integration tools include libraries of such conversions and insert them automatically. However, we need more than a mechanism to collect metadata. Repositories, for example, have not yielded the expected interoperability benefits. Without effective metadata location and exploitation tools, organizations lack sufficient incentives to collect and update accurate metadata. As a result, repositories have not greatly eased interoperability.

Fortunately, typical XML environments have universal connectivity and rich toolsets that provide wide accessibility and ease construction of interoperability tools. The universal connectivity of Web environments facilitates pointing to standard element definitions, conversion function libraries, and other resources that promote interoperability. Also, because of XML's ubiquity, tool builders benefit from a large marketplace of high-quality, inexpensive commercial development tools that simplify interoperability tool construction.

XML's strong base of freeware and commercial tools affords flexibility at greatly reduced development costs.

Level 4: Heterogeneous schemas

Developers are increasingly aware that schema diversity will be a serious problem even if XML schemas achieve wide usage.[11] To support interoperability at this level, a way to describe and share community schemas and to express mappings across schemas is necessary. Various communities have defined XML schemas that provide a neutral model for describing data structures. Communities developing standard schemas include e-commerce, healthcare, and data-warehousing vendors. Such schemas will reduce diversity among interfaces and ease data sharing. Oasis and BizTalk are examples of XML repository environments that map among XML elements and models.

XML does not provide intrinsically simpler model standardization than object systems, but its ubiquity and cheap tools have sparked enthusiasm, motivating some communities to agree on standards when previously they could not. Because it facilitates the sharing of information, the Web will dramatically increase the impact of community standards. Organizations will need to map their schemas and non-DBMS data to the standards, which may spur creation of a new generation of schema integration tools.

To cope with diversity, organizations must describe interschema mappings. Too often these are specified in code tied to specific proprietary tools, especially data-warehousing vendors' extract-transform-load tools. In recently merged efforts, the Metadata Coalition's Open Information Model (OIM) and the Object Management Group's Common Warehouse Metadata pursue a better approach that represents mappings declaratively and generates glue code from reusable and vendor-independent mappings.[10] CWM supports

XML's very real benefits apply not only to the Web but also to databases and applications.

relational, XML, and several legacy data sources, using SQL-99 to express mappings and XML for data interchange. Other mechanisms for expressing mappings include XSL and XQuery, although it may take some time before the latter receives efficient support from off-the-shelf query processors.

Level 5: Object identification

Improvements in describing attribute representation and semantics can remove one source of object misidentification—for example, is the date in a payment in US or European format? Also, XML makes it easy to attach uncertainty estimates as subsidiary elements to any output—although to be useful, the recipient must be prepared to interpret them.

Level 6: Data value reconciliation

Many strategies for data value reconciliation depend on having metadata such as time stamp and source quality attached to the data. In addition to attaching such annotations, XML makes it easy to return a set of alternative elements for an uncertain value if the recipient can use such output.

OUTSTANDING ISSUES

XML leaves unresolved how to best specify intersystem mappings. First, analysts need tools to help identify candidate relationships across systems. Second, the tools need to specify mappings declaratively, not in procedural or vendor-dependent code about which optimizers and other automated tools cannot reason. SQL and, hopefully, the upcoming XQuery language, are examples of such a declarative language. Of the two efforts to standardize the expression of intersystem mappings—OIM and CWM—only the latter directly supports XML data sources. Finally, analysts need tools to record intersystem relationships and mappings to community standard DTDs or schemas and to make them available for reuse.[12]

XML-based interoperability often means using XML as a message syntax for bulk transfers among systems. However, this approach often supports only predefined requests, notably: "Generate the standard message XYZ." Although XML tools will eventually support ad hoc queries, transactional updates, and subscriptions to specific kinds of changes, these capabilities currently lag behind comparable offerings in relational databases.

As with any hot new technology, XML has generated exaggerated claims. In reality, XML does not come close to eliminating the need for database management systems or solving large organizations' data-sharing problems. Some developers have a tendency to use XML as an excuse for skipping rigorous data modeling, which remains a critical activity for enterprise systems. However, exaggerated claims and occasional misuse do not negate XML's very real benefits, which apply not only to the Web but also to databases and applications:

- XML reduces the work of reconciling heterogeneous data structures. For example, XML will soon underlie a ubiquitous and inexpensive infrastructure for exchanging self-describing messages. In fact, participants can use XML as a neutral format to describe almost any data as a graph if they can agree on which schema to use and on what the elements mean.
- Data administrators can use XML to conveniently attach metadata, thereby simplifying other kinds of reconciliation—especially for heterogeneous attribute representations, semantics, and schemas.
- The enthusiasm surrounding XML motivates some communities to agree on standards when previously they could not.
- XML tools contribute greatly to organizations' ability to manage and share semistructured data—including much Web content—that is difficult to describe with a prespecified schema.

Although researchers and vendors have made great strides, realizing XML's full potential requires more: DBMS products that manage semistructured data; standards that provide shared vocabularies and schemas; and administrative tools that map heterogeneous attributes and schemas and resolve object identity and conflicting values. ✷

Acknowledgments

The authors thank Terry Bollinger, Frank Manola, Roger Costello, John Schneider, Kit Lueder, Lisa Harper, and the anonymous reviewers for their helpful comments.

References

1. J. Widom, "Data Management for XML: Research Directions," *IEEE Data Eng.*, Sept. 1999, pp. 44-52.
2. B. Adelberg, "NoDoSE: A Tool for Semi-Automatically Extracting Structured and Semistructured Data from Text Documents," *Proc. ACM Sigmod Int'l Conf. Management Data,* ACM Press, New York, 1998, pp. 283-294.
3. D. Mattox, L. Seligman, and K. Smith, "Rapper: A Wrapper Generator with Linguistic Knowledge," *Proc. 2nd Int'l Workshop Web Information and Data Management,* ACM Press, New York, 1999, pp. 6-11.

4. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Fusion in Mediator Systems," *Proc. Int'l Conf. Very Large Databases,* Morgan Kaufmann, San Francisco, 1996, pp. 413-424.
5. P. Buneman et al., "A Query Language and Optimization Techniques for Unstructured Data," *Proc. ACM Sigmod Int'l Conf. Management Data,* ACM Press, New York, 1996, pp. 505-516.
6. M. Fernandez et al., "Catching the Boat with Strudel: Experiences with a Web-Site Management System," *Proc. ACM Sigmod Int'l Conf. Management Data,* ACM Press, New York, 1998, pp. 414-425.
7. J. Bosak, "Media-Independent Publishing: Four Myths about XML," *Computer,* Oct. 1998, pp. 120-122.
8. E. Sciore, M. Siegel, and A. Rosenthal, "Using Semantic Values to Facilitate Interoperability among Heterogeneous Information Systems," *ACM Trans. Database Systems,* June 1994, pp. 254-290.
9. S. Sarawagi, ed., *IEEE Data Eng.*, Special Issue on Data Cleaning, Dec. 2000.
10. T. Vetterli, A. Vaduva, and M. Staudt, "Metadata Standards for Data Warehousing: Open Information Model versus Common Warehouse Metadata," *Sigmod Record,* Sept. 2000, pp. 68-75.
11. A. Gonsalves and L. Pender, "Schema Fragmentation Takes a Bite out of XML," *PC Week Online,* ZDNet, May 3, 1999, http://www.zdnet.com/pcweek/stories/news/0,4153,401355,00.html.
12. A. Rosenthal, E. Sciore, and S. Renner, "Toward Unified Metadata for the Department of Defense," *IEEE Metadata Workshop,* Silver Spring, Md., 1997, http://computer.org/conferen/proceed/meta97/papers/arosenthal/arosenthal.html.

Len Seligman *is a principal scientist at MITRE Corp. His research interests include heterogeneous databases, semistructured data, and large-scale information dissemination. Seligman received a PhD in information technology from George Mason University. Contact him at seligman@mitre.org.*

Arnon Rosenthal *is a principal scientist at MITRE Corp. His research interests include data administration, interoperability, distributed object management, legacy system migration, and database security. Rosenthal received a PhD in electrical engineering and computer science from the University of California, Berkeley. Contact him at arnie@mitre.org.*