An analytic study of XML database techniques

Jia-Lang Seng
Department of Management Information Systems, National Chengchi University, Taipei, Taiwan, ROC
Yu Lin
Department of Management Information Systems, National Chengchi University, Taipei, Taiwan, ROC
Jessie Wang
Department of Management Information Systems, National Chengchi University, Taipei, Taiwan, ROC
Jing Yu
Department of Management Information Systems, National Chengchi University, Taipei, Taiwan, ROC

Keywords
Computer languages, Integration, Electronic commerce, World Wide Web

Abstract
XML emerges and evolves quick and fast as Web and wireless technology penetrates more into the consumer marketplace. Database technology faces new challenges. It has to change to play the supportive role. Web and wireless applications master the technology paradigm shift. XML and database connectivity and transformation become critical. Heterogeneity and interoperability must be distinctly tackled. In this paper, we provide an in-depth and technical review of XML and XML database technology. An analytic and comparative framework is developed. Storage method, mapping technique, and transformation paradigm formulate the framework. We collect and compile the IBM, Oracle, Sybase, and Microsoft XML database products. We use the framework and analyze each of these XML database techniques. The comparison and contrast aims to provide an insight into the structural and methodological paradigm shift in XML database technology.

Industrial Management & Data Systems
103/2 [2003] 111-120

Introduction

As a consequence of the progression of the Web (World Wide Web, Web or WWW) and wireless information technology, the application of information systems is not standalone but shared and integrated. In the technology management area, there is a dramatic growth in the database system applications. Although more and more powerful database management systems are produced, database access still faces severe complications. The first complication is distribution. Not every query can be answered by the data in one single database system. Relations may be broken into fragments that are distributed among distinct databases.

The second complication in database integration is heterogeneity. This heterogeneity may be notational or conceptual. Notational heterogeneity concerns the access language and protocol. For example, one source is a Sybase database using SQL while the other is an IBM DB/2 database using SQL and another is an Object Store using OQL. This sort of heterogeneity can usually be handled through the middleware products (such as the Sybase OpenServer). Nevertheless, this kind of transforming and exchanging happens dynamically and interactively. The economical method to solve the problem is to exchange only flat-text files among heterogeneous databases. However, this approach requires much extra work, such as defining flat-text file formats, writing programs to extract flat-text files, and transforming heterogeneous files. It is obviously not an efficient and effective way. Fortunately, the emergence of eXtensible Markup Language (XML) standard gives us a new direction as the common base of heterogeneous data exchange. XML is a markup language and a subset of Standardized General Markup Language (SGML). XML is more complete and disciplined, and it allows you to define your own application-oriented markup tags. Its properties make XML particularly suitable for Web and wireless data interchange. Through a proposed XML-based transformation mechanism, the data integration/exchanging process among heterogeneous databases is easier and more favorable.

This paper aims to present the framework of XML-based transformation mechanism. We apply this framework to introduce how commercial database systems nowadays support XML. An analytic study is conducted to examine the current and standard XML database techniques. This paper is organized as follows. We review XML in section 2. We describe XML-based transformation framework in section 3. We discuss how XML can be applied to heterogeneous database integration. We analyze the current commercial XML database products in section 4, including Oracle database server, IBM DB2, Sybase Adaptive Server, and Microsoft SQL Server. We present a comparative analysis of key product features with management implications in section 5. We conclude the paper with a brief summary in section 6.

2. An overview of XML

This section presents an overview of XML. XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or a restricted form of SGML. By construction, XML documents conform to SGML formats.

2.1 Origin and goal

XML was developed by an XML Working Group (originally known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) in 1996. It was chaired by Jon Bosak of Sun Microsystems with the active participation of an XML Special Interest Group (previously known as the SGML Working Group) also organized by the W3C.

The design goals for XML are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

2.2 XML documents

A data object is an XML document if it is well-formed. A well-formed XML document may in addition be valid if it meets certain further constraints. Each XML document has both a logical structure and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a “root” or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions. All of those are indicated in the document by the explicit markups.

2.3 DTD and schemas

The XML document type declaration contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a DTD. The document type declaration can point to an external subset (a special kind of external entity) containing markup declarations, or can contain the markup declarations directly in an internal subset, or can do both. The DTD for a document consists of both subsets taken together. A markup declaration is an element type declaration, an attribute-list declaration, an entity declaration, or a notation declaration. For example, if you want a document type to be able to describe, users use lists that contain items.

```
<!ELEMENT List (Item)+>
<!ELEMENT Item (#PCDATA)>
```

XML schema has recently been approved as a W3C Recommendation. XML schemas are an XML language for describing and constraining the content of XML documents. Those offer several features which are urgently required in the data processing applications such as to define a sophisticated set of basic data types including dates, numbers, and strings with facets that can be superimposed with the minimum and maximum ranges and lengths.

2.4 Extensible stylesheets language

XSL is a language for expressing stylesheets. It consists of three parts:

1. XSL Transformations (XSLT): a language for transforming XML documents.
2. The XML Path Language (XPath): an expression language used by XSLT to access or refer to parts of an XML document. XPath is also used by the XML Linking specification.
3. An XML vocabulary for specifying formatting semantics (XSL Formatting Objects).

An XSL stylesheet processor accepts a document or data in XML and an XSL stylesheet and produces the presentation of that XML source content that was intended by the designer of that stylesheet. There are two aspects of this presentation process (see Figure 1). First, we construct a result tree from the XML source tree. Second, we interpret the result tree to produce formatted results suitable for presentation on a display and onto other media. The first aspect is called tree transformation and the second is called formatting.

You can use XSL to format XML documents. XSL specificiations consist of a set of rules that define the transfromation of an XML document into either an HTML document or a different XML document. You can create you own stylesheets for the

Industrial Management & Data Systems
103/2 [2003] 111-120

display of particular classes for particular applications. XSL is normally used with the presentation applications rather than with the applications for data interchange or storage.

3. XML-based transformation framework

This section outlines the framework for XML-based transformation mechanisms. To use XML documents for data exchange among databases, you must be able to store XML documents or the data that XML documents contain in the database. Therefore, we will discuss techniques about XML and databases, such as how to store XML documents and data in databases. For the XML-based transformation framework, we discuss it from three different levels. First is transferring data between XML documents and databases. Second is XML technology applied to heterogeneous database integration. Third is to XML technology applied to the B2B (business to business) EC (electronic commerce) environment? The objective of this paper centers on the XML technology applied to heterogeneous database integration with a focus on relational products, since they represent the key player in the marketplace.

3.1 XML and database

The most important factor in choosing a database is whether you are using the database to store data or documents. To store and retrieve data for XML documents, you can use a database and middleware. To store documents, you will need a content management system or a persistent document object management (DOM) implemnation (the native XML database). Basically, commercial relational databases store XML documents by extracting data from an XML document and organizing them as data rows and columns in databases. Some aslo provide the method to store an entire XML document in a single column such as binary large object (BLOB).

To transfer data between XML documents and databases also relies on a mapping between the document and the database. For example, this might map the <SalesOrder> element to the SalesOrders table and the <OrderDate> element to the SalesOrders.Date column. One kind of mapping is to embed product-specific commands in a template that is processed by the data transfer software. For example, the following template has embedded SELECT statements in the form of <SelectStmt> elements:

```
<?xml version="1.0"?>
<FlightInfo>
<Introduction>The following flights have
available seats:</Introduction>
<SelectStmt>SELECT Airline,
FltNumber, Depart, Arrive FROM
Flights</SelectStmt>
<Conclusion>We hope one of these meets
your needs</Conclusion>
</FlightInfo>
```

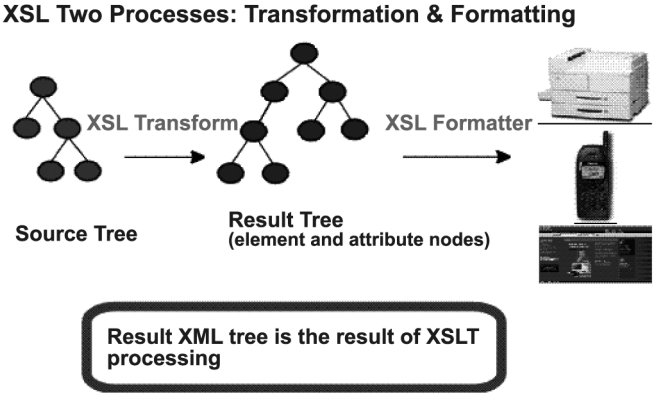
Another mapping is the data in an XML document modeled according to some predefined model. This is mapped either implicitly or explicitly to the database. Because data are transferred according to a predefined model and many documents may not conform to this model, XSLT is commonly integrated into these products. This allows applications to transform documents to the model before transferring data to the database and vice versa. There are two common models for the data in an XML document: the table model, which is the basis for table-based mappings, and the object model, which is the basis for object-relational mappings.

The table-based mapping is used by many of the middleware products that transfer data between an XML document and a relational database. It models XML documents as a single table or a set of tables. The object-relational mapping models the data in the XML document as a tree of objects that is specific to the data in the document.

3.2 Transformation between XML documents and databases

According to the Universal Data Interchange Model (UDIM), it is an approach based on Meta Data Model, and an environment for data interchange between different formats. In this environment, data activities between models like the re-regineering, modeling,

Figure 1
XSL processes



Source: W3C (2001 a, b)

Industrial Management & Data Systems
103/2 [2003] 111-120

retreiving, analysis, encapsulation, model optimizing and data transformation will proceed. Because the purpose of UDIM is universal data interchange, the file transform rules process module mainly produce the file transformation rules of physical files. The archiecture and modules of UDIM is described as follows.

- File Transform Rules Process: XML Parser, File Transform Rule Generatior.
- Data Transform Process: File Transform Processor, Data Integrator.
- Data Interchange Process: Data Interchanger Process.

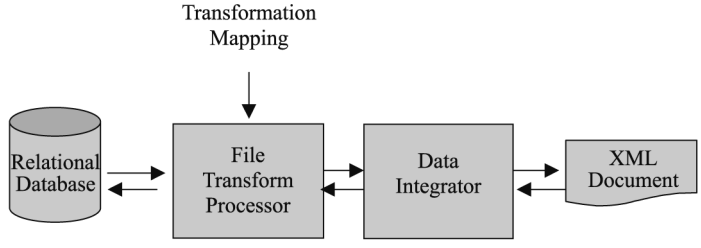
For the transformation between XML and relational databases, we do not care about the physical files. What we need to know is the mapping between XML documents and databases. Therefore, we simplify the UDIM archiecture and modules for the transformation between XML and relational databases. There is only one module left: Data Tranform Process including File Transform Processor and Data Integrator (see Figure 2).

In Figure 2, for retrieving data from databases to XML documents, the File Transform Processor receives the transformation mapping, according to the mapping to retrieve data from databases, then through the Data Integrator to produce a new XML document. When storing data from XML documents to databases, first, the Data Integrator reads the XML document, then the File Transform Processor inserts or updates database tables according to the transformation mapping.

3.3 XML in heterogeneous database integration

Basically, the transformation between XML documents and databases is the foundation of this section. A key strength of XML is a common format for data exchange between databases. An example showed in Figure 3 transfers data between the Oracle9i database server and Lotus Domino server.

Figure 2
Transformation between XML and databases



Source: Dayen (2000)

In the demonstration of the XML-based transformation mechanism, a human resource Web site is built up as a demo site of their heterogeneous environment. A summary oriented report generating an application running on the Oracle9i server is developed to show that the XML documents can act as suitable data exchange bases. Although this demonstration is quite successful, there are two issues which need to be noticed.

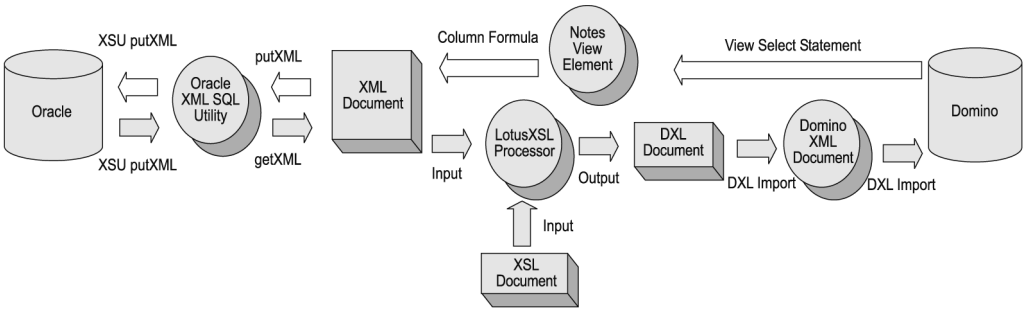
In its experience of data transformation, Oracle recommended that it is better to share a common DTD to make exchanging data more reliable and easier to manage. To share a common DTD requires intermediate processing and transformation of the XML. For example, if you want to write XML data to an object view or table, but the XML data do not match the underlying table structure, you need to transform the XML data before writing them to the database. A good approach for doing this is to use an XSL stylesheet to transform the XML document into a new XML document that conforms to the underlying database schema.

3.4 XML applied to the B2B environment

For the business requirement, XML technology is not only the XML document. There are more protocols, standards and frameworks provided for the B2B environment.

- XMLP/SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized and distributed environment using XML. XMLP/SOAP does not itself define any application semantics such as a programming model or implementation specific semantics; rather it defines a simple mechanism for expressing application semantics by providing a modular package model and encoding mechanisms for encoding data within modules. This allows XMLP/SOAP to be used in a large variety of systems ranging from messaging systems to RPC.
- The United Nations body for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS) have joined forces to initiate a worldwide project to standardize XML business specifications. UN/CEFACT and OASIS have established the Electronic Business XML (ebXML) initiative to develop a technical framework that will enable XML to be utilized in a consistent manner for the exchange of all electronic business data.

Figure 3
Data transfer between Oracle database and Lotus Domino server



Source: Oracle (2001)

- ebXML is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet. Using ebXML, companies now have a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define and register business processes.
- Another XML e-business standard is the BizTalk framework, promoted by Microsoft and partners. It aims to make e-business XML easier for individual companies to mix and match XML message formats from different vendors and standards groupings, and to pick out the sets that best meet their business needs and application mix. It does so in three ways. First, it sets out a "canonical form," in which any application-specific set of XML message formats can be defined. Second, it provides a public repository at http://www.BizTalk.org where BizTalk-conformant XML message format sets can be validated, lodged, retrieved, and freely used. Third, the creators of BizTalk-conformant message standards are encouraged to lodge XSLT-based translations between their own formats and others' standard formats.

4. Product analysis

In this section we will describe and discuss four XML-enabled databases that can provide different methods to transfer data between the XML documents and relational databases. We examine and analyze key products in the marketplace. They are IBM DB2, Sybase Adaptive Server, Oracle8i/9i Server, and Microsoft SQL Server. We will have an analytic study in three aspects as the dimensions. They are the storage method, the transformation technique, and the application profile.

4.1 IBM DB2

IBM DB2 develops the DB2 XML Extender product to provide the function that transfers data between XML documents and relational databases.

4.1.1 Storage method

With DB2 XML Extender, it provides document-centric storage methods. There are two options in DB2:

1 *XML columns*. This method is document-centric. The documents are inserted into columns that are enabled for XML, and can be updated, retrieved, and searched. Element and attribute data can be mapped to DB2 tables called side tables, which can then be indexed for fast searches.
2 *XML collections*. You have to map XML document structures to DB2 tables so that XML documents can be composed from existent DB2 data.

4.1.2 Mapping between XML document and database

The Document Access Definition file, or DAD file, itself is an XML document. It specifies how the XML documents are stored in the database and handled. The example of person column DAD file is shown as follows.

```
<xml version="1.0"?>
<!DOCTYPE DAD SYSTEM
"c:\dxx\dtd\dad.dtd">
<DAD>
<dtdid>C:.dtd</dtdid>
<validation>YES</validation>
<Xcolumn>
<table name="person_names">
<column name="fname"
type="varchar(50)"
path="/person/firstName"
multi_occurrence="NO"/>
<column name="lname"
type="varchar(50)"
path="/person/lastName"
multi_occurrence="NO"/>
</table>
<table name="person_phone_number">
```

```
<column name="pnumber"
type="varchar(20)"
path="/person/phone/number"
multi_occurrence="YES"/>
</table>
<table name="person_phone_type">
<column name="ptype"
type="varchar(20)"
path="/person/phone/type"
multi_occurrence="YES"/>
</table>
</Xcolumn>
</DAD>
```

4.2 Sybase adaptive server enterprise

Sybase Adaptive Server introduces the ResultSetXml Java class as a base for processing XML documents in both directions.

4.2.1 Storage method

There is a basic way to store XML data in Adaptive Server: element storage. In this method, you extract data elements from an XML document and store them as data rows and columns in Adaptive Server. For example, using the XML Order document, you can create SQL tables with columns for the individual elements of an order: Date, CustomerId, CustomerName, ItemId, ItemName, Quantity, and Units. You can then manage those data in SQL with normal SQL operations.

4.2.2 Mapping between XML document and database

Sybase employs an XML document type, ResultSet, to describe both XML document metadata (element name, type, size, etc.) and actual row data. Here is an excerpt from a hypothetical FxTradeSet.xml:

```
<?xml version="1.0"?>
<!DOCTYPE ResultSet SYSTEM
"ResultSet.dtd">
<ResultSet>
  <ResultSetMetaData>
    <ColumnMetaData
    ...
    getColumnLabel="CURRENCY1"
    getColumnName="CURRENCY1"
    getColumnType="12"
    ... />
  ...
  </ResultSetMetaData>
  <ResultSetData>
    <Row>
      <Column name="CURRENCY1">
      GBP</Column>
    ...
    </Row>
  </ResultSetData>
</ResultSet>
```

4.2.3 XML Transformation in database applications

There are tree applications of XML transformation in Sybase database. Theses applications are organized in three layers.

1. Transact-SQL statements: such as insert, select, and update for referencing SQL columns and variables that contain XML documents. These SQL operations use Java classes and methods to manipulate the XML documents.
2. Java classes: XML documents and elements of those documents to be updated. If there is an application-specific class for the Order document type and there will be a general class for arbitrary SQL result sets.
3. An XML parser: Java classes use it to analyze and manipulate XML documents.

4.3 Oracle 8i/9i Server

Oracle8i/9i provides a number of XML-related tools. The Oracle XML Developer's Kits (XDK) contain the basic building blocks for reading, manipulating, transforming, and viewing XML documents. The XML SQL Utility for Java is a set of Java classes for transferring data between a relational database and an XML document. This processor is implemented as a Java servlet and takes as its input an XML file containing embedded SQL queries. Oracle9i includes "Native XML Database Support (XDB)", which introduces a new object data type (XML Type). It can store XML in the database via SQL and render traditional database data as XML.

4.3.1 Storage method

The basic strategy for storing XML documents in the database is to store an XML document as a single, intact object with its tags in a CLOB or BLOB. That is, storing an intact XML document in a CLOB or BLOB is a good strategy if the XML document contains static content that will only be updated by replacing the entire document. Examples include written text such as articles, advertisements, books, legal contracts, and so on.

4.3.2 Mapping between XML document and database

In XML SQL Utility for Java, the structure of the resulting XML document is based on the internal structure of the database schema that returns the query results. Columns are mapped to top-level elements. Scalar values are mapped to elements with text-only content. Object types are mapped to elements with attributes appearing as sub-elements. The utility can generate either a string representation of the XML document or an

in-memory XML DOM tree of elements. XSQL Servlet is a Java servlet that uses the XML SQL Utility for Java to transfer data from a relational database to an XML document. The servlet has SELECT statements embedded in the template as <query> elements; when processed, these are replaced by the result for the query formatted as XML. Support for passing the query parameters through HTTP and for processing the output document with XSL is provided.

4.3.3 XML transformation in database applications

For the applications based on Oracle database, the Oracle XML Developer's Kits (XDK) contain the basic building blocks for reading, manipulating, transforming and viewing XML documents. Oracle XDKs consist of the following components:

- XML Parsers: supports Java, C, C++, and PL/SQL.
- XSLT Processor: transforms or renders XML into other text-based formats such as HTML.
- XML Schema Processor: supports Java, C, and C++.
- XML Class Generator: automatically generates Java and C++ classes from DTDs and Schemas to send XML data from Web forms and applications.
- XML Transviewer Java Beans: visually views and transforms XML documents and data via Java components.
- XML SQL Utility: supports Java and generates XML documents, DTDs, and Schemas from SQL queries.
- XSQL Servlet: combines XML, SQL, and XSLT in the server to deliver dynamic Web content.

4.4 Microsoft SQL Server

Microsoft SQL Server supports XML in three ways: the FOR XML clause in SELECT statements, the XPath queries that use annotated XML-Data Reduced schemas, and the OpenXML function in stored procedures. SELECT statements and XPath queries can be submitted via HTTP, either directly or from a template file.

4.4.1 Storage method

Storing XML documents employs OPENXML, a row set function, similar to table or view. OPENXML can be used for inserts, updates, or SELECT INTO target tables. OPENXML simplified syntax is shown as follows:

> OPENXML (<XML document handler>, <path pattern>, <flags>) WITH (Schema | Table).

4.4.2 Mapping between XML document and database

The FOR XML clause has three options, which specify how the SELECT statement is mapped to XML. RAW models the result set as a table with one element (named "row") returned for each row. Columns can be returned either as attributes or child elements. AUTO is the same as RAW except: the row elements are named the same as table name and the resulting XML is nested in a linear hierarchy in the order in which tables appear in the select list.

Annotated XML-Data Reduced schemas, also known as mapping schemas, contain extra attributes that map elements and attributes to tables and columns. These specify an object-relational mapping between the XML document and the database and are used to query the database using a subset of XPath. A tool exists to construct the mapping schemas graphically.

4.4.3 XML transformation in database applications

Extraction extends a SELECT-clause by use of the FOR XML construct. Storing introduces a row set function OPENXML analogous to a table or view. Extracting XML from the database refers to mapping between database columns and XML elements or attributes by means of AS aliases in a SELECT.

> <database column> AS [Element Name! Nesting Level! Attribute Name! Directive]

Storing XML documents employs OPENXML that is a new row set function similar to table or view. OPENXML can be used for inserts, updates, and SELECT INTO target tables. OPENXML simplified syntax is shown as follows:

> OPENXML (<XML document handler>, <path pattern>, <flags>) WITH (Schema | Table).

5. Comparative analysis

We point out the main advantage and disadvantage of each product described above. We analyze the pros and cons of the main feature. A comparative framework of analysis is developed with four columns of storage method, mapping method, transformation approach, and promise and pitfall. The discussion is prepared from either the technical viewpoint or managerial perspective.

5.1 IBM product promise and pitfall

IBM provides SQL mapping not RDB node mapping in XML transformation. SQL

mapping is a template-based language and can only be used to transfer data from the database to an XML document. RDB node mapping is model-based language and uses an object-relational mapping. It is a better approach to be used to transfer data to and from the database. A simple comparison is provided as follows:

- *SQL mapping*. In order to compose the XML document and to create the DAD file with SQL mapping, we have to use the <SQL_stmt></SQL_stmt> to specify the SQL used for mapping the relational data to the XML document.
- RDB node mapping: instead of using SQL stmt in the SQL mapping, the RDB_node element for the element_node, text_node, and attribute_node are adopted. In RDB node mapping, we can specify all tables that are associated with the XML document.

5.2 Sybase product promise and pitfall

Sybase provides element storage not document storage. In element storage, you extract data elements from an XML document and store them as data rows and columns in Adaptive Server. You can then manage those data in SQL with normal SQL operations. However, without document storage, you cannot store an entire XML document in a single SQL column. There will always be mapping mismatch between XML documents and database since 1:M, M:1 and M:N mapping creates problems in the element storage and schema evolution. More, some vendors may adopt the hybrid storage while Sybase has not yet considered it. In hybrid storage, you can store an XML document in an SQL column and extract some of its data elements into separate columns for faster and more convenient access.

5.3 Oracle product promise and pitfall

Oracle stores XML documents in a document-centric manner. Storing this kind of document intact within Oracle9i has the advantages of an industry-proven database and its reliability over file system storage. However, it does not allow you to choose to store an XML document outside of database. It is not possible to efficiently retrieve the document through the use of BFILES, URLs, and text-based indexing. More, typically, the XML document contains elements or attributes that have complex structures. Without the option and with its object-relational extensions, Oracle does not have the ability to capture the structure of the data outside the database using object types, object references, and collections. Further, views enable you to construct an object on the "fly" by combining XML data stored in a variety of ways. But, at present time, Oracle does not support this important feature. So, when you need to store structured data (such as employee data, customer data, and so on) in one location within object-relational tables, and store related unstructured data (such as descriptions and comments) within a CLOB, you cannot do that. When you need to retrieve the data as a whole, you cannot construct the structure from the various pieces of data with the use of type constructors in the view's select statement.

5.4 Microsoft SQL server product promise and pitfall

The process of storing an XML document in SQL Server is tedious and troublesome. It involves three steps. We point out the potential pitfall of each step as follows:

- SQL Server has to first obtain an XML document handler by compiling XML document into internal DOM representation. A set of stored procedures has to be compiled and executed.
- SQL Server schema evolution must associate each schema field with its respective atomic XML element or elements and change the corresponding database field or fields at the same time in order to keep data integrity.
- SQL Server needs another set of stored procedures in order to handle the removal and alteration of XML elements with their corresponding database fields.

5.5 A summary

A summary of the above XML database techniques developed in key products is analyzed in Table I. The summary is prepared based on the above comparative study. The storage method, the mapping method, and the transformation approach have been compared.

6. Concluding remarks

XML is evolving. The W3C XML Schema Working Group has drafted specifications to define the structure, content, and semantics of XML documents. The XML Schema specification addresses areas that are currently lacking in DTDs, including the definition and validation of data types. XML

Table I
An analytic comparative summary of XML database products

Database product	Mapping method	Storage method	Transformation method	Promise and pitfall
Oracle 8i/9i	Implicitly constructing object-relational data model	Single CLOB or BLOB column A relational table or an object table Oracle 9i : native XML database support	Oracle XML Developer's Kits (XDK)	Inefficient retrieval of outside database XML documents
IBM DB2	Using data access definition file	XML columns XML collections	Designated sets of stored procedures	SQL mapping instead of RDB node mapping
Microsoft SQL Server	Applying SQL extension; row set function	A relation table	SQL constructs FOR XML and row set OPENXML	Weak handler of schema evolution
Sybase Adaptive Server	Developing result set DTD	Element storage Document storage	Java classes and Java methods	Possible mapping mismatch between elements and fields

Schema was shaped by both the real-world experience of the enterprise and by scientific disciplines including formal language theory, computational linguistics, and type theory. The standard will play an important role in bridging the gap between traditional database-like information and document-like content that is typically expressed using markup languages. XML Schema enables a whole new generation of content and e-business applications. Therefore, there are still some issues related to transferring data between XML documents and databases not mentioned in this paper, such as data types, character sets, etc. Our aim in this paper is to present the framework for an XML-based transformation mechanism and apply this framework to analyze how commercial databases nowadays support and power XML.

References

Dayen, I. (2000), "Storing XML in relational databases", available at: www.xml.com/pub/a

World Wide Web Consortium (W3C) (2001a), Extensible Stylesheet Language, available at: www.w3.org/Style/XSL

World Wide Web Consortium (W3C) (2001b), Extensible Stylesheet Language (XSL) Version 1.0., available at: www.w3.org/TR/2000/CR-xsl-20001121/xslspec.html

Further reading

Bourret, R. (2001), "XML and databases", available at: www.rpbourret.com/xml/XMLAndDatabases.htm

Bourret, R. (2001), "XML database products", available at: www.rpbourret.com/xml/XMLDatabaseProds.htm

Eisenberg, A. and Melton, J. (2001), "SQL/XML and the SQL/X," *ACM SIGMOD RECORD*, Vol. 30 No. 3.

Genesereth, M.R., Keller, A.M. and Duschka, O.M. (1997), "Infomaster: an information integration system", *ACM SIGMOD International Conference Proceedings*, pp. 539-42.

IBM (2000), *Integrating XML with DB2 XML Extender and DB2 Text Extender*.

Klettke, M. and Meyer, H. (2000), "XML and object-relational database systems – enhancing structural mappings based on statistics", *ACM SIGMOD International Conference Proceedings*, pp. 63-8.

Kotok, A. (2001), "ebXML ropes in SOAP", available at: www.xml.com/pub/a/2001/04/04/ebXML.html

Kumar, A. and Palvia, P. (2001), "Key data management issues in a global executive information system", *Industrial Management and Data Systems*, Vol. 101 No. 4, pp. 153-64.

OASIS and UN/CEFACT (2001), "ebXML", available at: www.ebxml.org

Oracle (2001), "Using XML in Oracle Database Applications Part 2: About Oracle XML Products", available at: http://technet.oracle.com/tech/xml/info/index2.htm?Info&htdocs/otnwp/about_oracle_xml_products.htm

Shanmugasundaram, J., Shekita, E., Kiernan, J., Krishnamurthy, R., Viglas, E., Naughton, J. and Tatarinov, I. (2001), "A general technique for querying xml documents using a

relational database system", *ACM SIGMOD RECORD*, Vol. 30 No. 3.
Sybase (2001), "Using XML in Adaptive Server Database", available at: http://manuals.sybase/com/onlinebooks/
Worden, R. (2000), "XML E-business standards: promises and pitfalls", available at: www.xml.com/pub/a/2000/01/ebusiness/index.html
World Wide Web Consortium (W3C) (2001), XMLP/SOAP/, available at: www.w3.org/2000/xp/Group/1/04/17/xmlp-soap-01.html
World Wide Web Consortium (W3C) (2001), Extensible Markup Language (XML), available at: www.w3.org/TR/2000/REC-xml-20001006
World Wide Web Consortium (W3C) (2001), The XML FAQ, available at: www.ucc.ie/xml/index.htm
Zhang, Q. (2001), "Object-oriented database systems in manufacturing: selection and applications", *Industrial Management & Data Systems*, Vol. 101 No. 3, pp. 97-105.