An Introduction to the e-XML Data Integration Suite

Georges Gardarin, Antoine Mensch, Anthony Tomasic

e-XMLMedia, 29 Avenue du Général Leclerc, 92340 Bourg La Reine, France
georges.gardarin@e-xmlmedia.fr

Abstract. This paper describes the e-XML component suite, a modular product for integrating heterogeneous data sources under an XML schema and querying in real-time the integrated information using XQuery, the emerging W3C standard for XML query. We describe the two main components of the suite, i.e., the repository for warehousing XML and the mediator for distributed query processing. We also discuss some typical applications.

1 Introduction

In the past few years, XML has become the standard for exchanging data on the Internet and on intranets in the enterprise. Different approaches have been proposed for efficiently managing, storing, querying and presenting XML data from diverse sources [1,2,3]. e-XMLMedia believes that building on strong foundations is the best approach for developing efficient and powerful web-enabled information systems. Thus, as relational and object-relational database systems are providing successful solutions for handling data, the e-XML suite of e-XMLMedia is a collection of components built to work and cooperate with relational DBMSs, such as Oracle, DB2, SQL Server, TimesTen and Postgres. More precisely, the suite is composed of Java components able to work with any database supporting a JDBC 2.0 driver. In addition, extensive attention has been given to the incorporation of legacy data sources.

The e-XML suite provides an efficient way to integrate access to multiple data sources on the Internet or on intranets through XML queries. The suite components can be assembled together or used separately in different application contexts. Possible applications include the constitution of electronic libraries from existing sources, the collect of information for trading systems, the assembling of fragments of data for automatic report generation, and more generally knowledge management and information diffusion in the enterprise. Used together, the components form a unique solution to support iterative definition and manipulation of virtual semi-structured databases derived from existing data sources or loosely structured files in Internet/Intranet environments.

This paper is an overview of the e-XML component suite. It presents information that will help you understanding the suite main objectives, functionalities and architectures. The topics in this paper include:

1. A summary of the functional objectives and architecture of the e-XML suite.
2. A description of the main e-XML components: Repository and Mediator.
3. A presentation of typical applications in which e-XML components are currently being integrated

2 Principles of the e-XML Suite

This section summarizes the objectives, presents the architecture, and discusses the functionality of the e-XML component suite.

2.1 Objectives

Many research projects have focused on logical data integration using the relational model or the object model as integration model [4,5,6]. Most of them have difficulties in integrating a large number of data sources with very heterogeneous data including documents, as they exist in many enterprises and on the Web. In addition, performance is a real issue for distributed query processing on multiple sources. The main objective of the e-XML suite is to address these diversity and scalability problems through the use of XML as an exchange format associated with an efficient XQuery processing technology.

More precisely, the objectives of our component suite can be formulated as follows:

– **Use XML as integration model.** By using XML and its derivatives (XQuery, XSchema, DOM, SAX, SOAP, XForms, XSL), a wide range of data sources with different data formats can be integrated and multiple client tools can be interfaced.
– **Process queries in real-time.** Queries are as much as possible distributed to local sources and processed against the current data, thus delivering up to date information. However, components in the suite are able to warehouse XML data, which can be included in the query process somehow as a cache.
– **Optimize query performance.** Maximum delegation of sub-queries, minimum network traffic, event driven approach to XML processing, are key features in query processing performance. A simple but efficient capability-based distributed query-processing algorithm should minimize response time.
– **Facilitate user acceptance.** The e-XML Data Integration Suite can be used incrementally, first to solve simple problems (e.g., XML-ize your relational database), then to query uniformly heterogeneous sources, finally to set a global e-business infrastructure in your enterprise.

2.2 Architecture

The overall global functional architecture is shown in Figure 1.

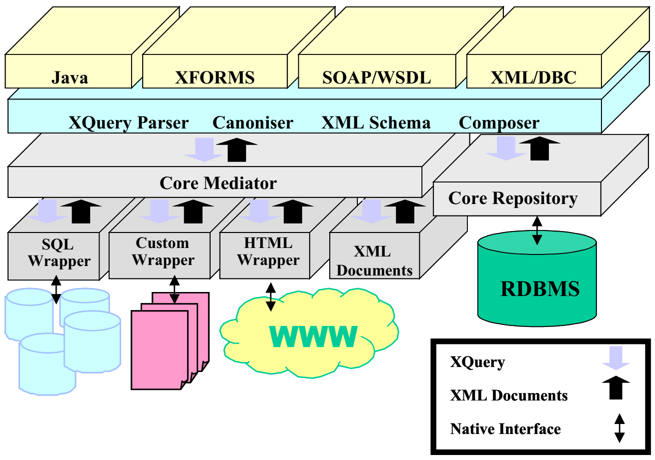

Fig. 1. The e-XML Data Integration Suite Global Architecture

The top layer consists of four client interfaces: a Java-based client interface, an XForms client interface for forms processing, a SOAP/WSDL client interface for binding RPC calls to arbitrary client programming languages (for example to .NET clients), and the XML/DBC interface consisting of a JDBC style client interface supporting XQuery and XML. The first three interfaces are standardized. The last interface is an adaptation of the JDBC interface for XML document repositories. This interface is uniformly used to access wrappers, mediators and repositories.

The second layer consists of the functionality shared by the Mediator and the Repository products. An XQuery Parser generates parse trees of queries. These are handed to Canoniser that generates a canonical form for the query. Since both the Mediator and Repository understand XML Schemas, a common library of Schema management routines is used, insuring identical semantics for all products. Finally, when multiple answers are returned from the Mediator or Repository, the Composer combines them into a single answer.

The next layer consists of the core functionality. The Repository functionality is described in Section 3. The Mediator functionality is described in Section 4.

The final layer consists of the collection of wrappers – SQL wrappers access RDBMS, Custom wrappers access arbitrary applications, and a specially designed HTML wrapper access the WWW. In addition the core Repository manages XML documents in an object/relational DBMS.

2.3 The XML/DBC Uniform Interface

All components that process queries in the toolkit architecture are accessible through XML/DBC. Thus, XQuery queries can be submitted through a uniform interface to a mediator, a repository or an SQL Wrapper. Thus, XML/DBC is both the input interface and the wrapper interface of a mediator. As a consequence, a mediator can call another mediator or itself as a wrapper. This is quite powerful. However, wrappers are generally not able to support fully XQuery. For example, a file wrapper cannot process joins. Thus, a capability interface is included in XML/DBC, to be able to get the supported functionalities of a wrapper.

The basic functionalities of XML/DBC are the following:

– Management of connections to an XQuery data source (repository, mediator and wrappers),
– Submission of XML queries and updates;
– Fetch of XML fragments as answers to queries;
– Retrieval of the XML schema describing the data source.

A SOAP version of XML/DBC is available, i.e., the API functions can be invoked through SOAP, making the tools ideal components for developing Web Services.

3 The e-XML Repository

In this section, we describe the e-XML Repository, which is able to store XML documents in any object-relational DBMS and which provides XQuery access to retrieve XML document fragments and assemble them. The XML Repository component manages persistent collections of XML documents in a relational database and provides query access to them. A name and an owner identify an XML collection. By default this owner is the same as the database user. Each collection is backed by a set of relational tables. In a generic mapping, the tables are fixed by the Repository implementation. In the case of a schema mapping, the tables consist of a combination of implementation dependent tables and user tables defined in the mapping.

3.1 Storage module

The Repository storage module performs the mapping of XML documents either into generic tables or into existing relational tables when the schema mapping option has been selected. The storage module, when storing XML documents without schema, builds metadata information under the form of a dataguide. The dataguide is constructed dynamically for repositories without associated schemas. The Repository provides several functions to retrieve metadata.

The storage module is natively designed to use the SAX 2.0 standard as an external interface with applications. Helpers are provided with the product in order to supply conversion between SAX 2.0 and DOM Level 2. As a consequence, the storage module relies on an external SAX parser, which is responsible for XML parsing and validation. The Apache Xerces parser by default is used as a parsing device.

Note that order information is maintained, even for attributes, so that the original document can be accurately reconstructed.

3.2 Query module

The query language supported is XQuery, as described in [7]. Document insert and deletion is supported, however, no sophisticated updates are allowed in the current version. The Query object gives a read-only access to data stored in the Repository. Queries can perform joins over several XML collections. Query results are XML document fragments.

Query processing is efficient due to some sophisticated optimization features. Generally, relational systems are considered inefficient for processing queries containing path traversals. However, in the context of XML documents, some characteristics of relational systems can compensate for this handicap. In particular, the XML document graph is encoded in an efficient manner that translates path traversal operations into index driven selection on tables. Thus, inefficient solutions such as complex join queries or recursive query processing are avoided.

In XML databases, regular path expressions with Kleene star operators have special importance. XML data have irregular and changing structures. By means of regular path expressions, users can query the database without the full knowledge on data structures. Although convenient to use, regular path expressions in general cannot be evaluated directly due to their prohibitive cost. To avoid this cost, the Repository query processor expands regular path expressions using metadata.

3.3 Mapping XML Objects to Tables

Current applications are often built on object-relational DBMS. Databases are composed of tables with columns of various types, including numeric and string, but also full text, images, and geometry. Thus, there is a strong demand in mapping XML documents to tables and vice-versa.

As explained above, an XML object is a tree with leaves bearing data. Trees can be mapped into tables in many ways. Hence, several mapping schemes are possible for storing XML objects in an object-relational DBMS. Our experience confirms that mapping schemes offer a wide range of performance for a given query workload according to very specific detailed choices, such as OIDs assignment, clustering and indexing schemes, etc. In addition, applications differ in mapping requirements; some may require specific table schemas to process data through standard tools, while others have no such requirement. However, mappings require administrative effort.

To comply with these different requirements, the e-XML Repository provides two types of mappings: generic and schema-based. With the generic mapping, application tables are automatically created and mapping directives are implicitly given. With the schema-based mapping, the documents have a schema, mapping directives are given, and an administrator creates application tables. Generic mappings are convenient and schema-based mappings offer maximum flexibility and performance.

This sophisticated mapping method provides capabilities to map part of XML documents to existing relational tables. Unmapped elements are stored in internal generated tables (as with the generic method), allowing the complete restoration of original documents. To make it possible, documents must be validated against an XML Schema. This XML Schema is unique for a Repository. The validation is performed by an external validating parser. Mapping directives are defined through an XML document with a specific grammar. A specific namespace (represented by the xrm prefix) has been defined for XML relational mapping directives.

4 The e-XML Mediator

In this section, we describe the e-XML Mediator, which provides a uniform view of multiple heterogeneous data sources and query access with the XQuery language. To accomplish this task, the Mediator manages data sources and processes queries The Mediator is composed of several packages, among them:

– The administration console is a simple interface to administrate the Mediator and register wrapped data sources.
– The evaluator is responsible for query processing, including parsing, decomposition, optimization, and composition of results.
– The exchanger provides the communication interface between Mediators and Wrappers based on XML/DBC, and manages the metadata cache including wrapper capabilities.
– A Wrapper is able to execute a query sent by the Mediator on a specific data source and to give back the results as a compressed DOM tree of objects.

4.1 Wrapping Data Sources

Data sources are managed through an XML based configuration file. Data sources are wrapped with a standard interface. This interface uses dynamic attachment of wrappers to the mediator. Metadata from the source is loaded at attachment time. In particular, configuration of the Mediator is entirely dynamic; no static “compile time” configuration is required. Data sources are distinguished as pure relational data sources accessible through JDBC, wrapped data sources translated in XML and often not supporting complex queries, and XML Repositories supporting XQuery via the XML/DBC interface. Wrappers are available for SQL (object-relational DBMS accessed through JDBC), XQuery (Repository and Mediator), and HTML pages. Specific wrappers can be added; they have to comply with the XML/DBC interface.

4.2 Query Processing

The query processing component is composed of several packages in charge of decomposing XQuery queries into mono-source sub-queries, efficiently shipping local queries to data sources, gathering results, processing and assembling them. In particular this component chooses an execution strategy for the query.

The Mediator makes three key decisions during query processing. First the Mediator determines the localization of the relevant sources to process a query among those accessible by the Mediator. Using the metadata information on each data source, the location of data that appears in a query is determined. Second the

Mediator determines the amount of sequential or parallel execution that occurs during execution. The mix between parallel and sequential is determined by the binding relationship between different data sources as expressed by join predicates. Independent data sources are contacted in parallel. Dependent data sources are accessed in a pipelined fashion. Third, the Mediator determines the amount of query processing that can be offloaded onto the data source. This determination is the result of a comparison of the requirements of a particular query and the query processing power available to a data source.

The Mediator also provides a way to structure the result in an integrated XML document, according to the RETURN clause of the query. The three main facilities provided at this level are the following:

– Renaming Tags.
– Changing the initial data structure.
– Composing results when coming from multiple data sources.

5 Examples of Applications

In this section, we present an overview of several applications that have been or are prototyped with the e-XML component suite.

5.1 Semi-Structured Document Manager

This scenario is derived from a major legislation assembly system in Europe that plans to manage textual law amendments (with structured descriptions) in a central Repository. In this case, semi-structured XML documents include a mix of free text and tabular data, to be stored in a relational database. Such an application requires a flexible mapping capability between the XML documents and the underlying database. Transactions are developed on an application server to query and update the database. To publish in HTML, PDF, RTF, etc., the XML results of queries are fed to an XSL transformer. XSL style-sheets guide the transformation process. This architecture has three advantages – flexible representation of documents that permit easy future extension, independence from the document presentation format, and independence from the on-line interaction process of document management.

5.2 XML Web Cache

This scenario is derived from a major financial institution that plans to introduce a new product: a WWW based information delivery system for its clients. The institution's goal is to improve the information delivered to its clients with minimal programming expenditures. The institution currently has a large legacy infrastructure. Clients are given Internet access to the information in the legacy infrastructure through a WWW server that operates through multiple firewalls – first through the security server, then through the client server, and finally through the legacy infrastructure. The use of a Repository for XML data transmitted throughout the system speeds up performance by storing copies of XML data in a central location managed through a main memory database. In addition to a Repository facility, the use of XQuery query access affects the implementation cost of the application logic by saving months of application development time. With the addition of a Mediator to the architecture, the system has the ability to dynamically fetch data from the legacy infrastructure. Thus the ``cache'' is effectively a mini-database with integration capabilities.

5.3 XML Meta-Directory Manager

This scenario is derived from an Internet provider willing to establish a unique identification point for all its clients and applications. The institution already uses some LDAP directories with specific queries. Fast and interoperable identification is required. There exists an XML standard (DTD and schema) for LDAP entries, called DSML. XQuery queries are appropriate for querying, in an interoperable way, LDAP directories. Thus, the XML Repository will be used to identify new customers in a central identification point. Wrappers for existing directories make them XML capable. As a result, identification in the new system will first be evaluated on the XML Repository, and then if an entry is not found, it will be evaluated on the pre-existing directories through the Mediator. If the result of the identification is positive, the central Repository will be updated by inserting the XML retrieved entry. Duplicates can be eliminated. This architecture requires a very fast query engine. To satisfy the engine requirements, a main memory relational database with the Repository running on top is used.

5.4 Source Specific Search Engine

Search engines are in general based on robots that crawl the WWW and build intelligent indexes of URLs. Currently, as the WWW is HTML-based, search engines are based on information retrieval technology that provides a fairly "flat" interpretation of the WWW. That is, the documents contain little structure. However, when data sources are well defined and structured WWW sites (financial WWW sites, electronic libraries, business sites using XML based document exchange), documents can be translated into XML on the fly (with the aid of the appropriate extraction language or tool). In this case, the e-XML components are an ideal platform to develop a search engine dealing with multiple (possibly heterogeneous) sources. The mediator acts as a search engine dealing with multiple HTML sites and associated extractors. The Repository is used for storing the results of robot crawling. The query manager manages search requests by issuing queries to the Mediator – note that these requests reference both the HTML sites and the Repository. The robot manager handles both periodic robot crawling and the administration console for the robot. Results of crawling are stored in the Repository.

6 Comparison with other Approaches

DBMS supporting XML can be classified in four categories:

1. XOR-DBMS (XML Object relational DBMS). This approach consists in extending object-relational DBMSs to support storage of XML documents in tables and publication of SQL query results in XML. Most RDBMSs suppliers are proposing simple solutions to map flat XML to tables and tables to XML. Non-flat XML can be flattened using XSL or specific nesting of tables, and vice-versa.
2. XML-DBMS (XML DBMS). Pure XML DBMSs are brand new systems developing specific storage and access methods for XML. In general, XML is stored in files and specific indexing techniques are used for improving query performance. These systems claim to be efficient in query processing, but this is not proved yet and even difficult to prove as XML queries can be very diverse.
3. XOO-DBMS (XML Object-Oriented DBMS). This class corresponds to the enhancement of previously existing Object-Oriented DBMSs to support XML. Generally, XML is transformed in DOM objects that are made persistent in the object database. Query languages can be extensions of OQL.
4. XML-DOCM (XML DOcument Manager). This class corresponds to document management systems that have been extended to handle XML. Previous versions where often based on SGML. Queries are based on keywords. They generally return a list of qualifying documents with ranking. Their interfaces have some similarities with that of search engines.

The e-XML-SUITE does not belong to any of these categories, but rather take advantages of several of them. The first advantage of the e-XML-SUITE is that it is a middleware product, which does not require a new DBMS: you can built and query XML views on top of your favorite RDBMS. Second, it is a suite of Java components that integrate several tools accessible through APIs. The suite includes the Repository and the Mediator, as described below, but also other tools such as the XMLizer to extract or map XML from/to relational databases and the XForms component to enter data in XML through forms from Web clients. Table 1 summarizes some determinant advantages of the e-XMLMEDIA approach.

Table 1. Functionality Comparison Between Classes of XML Database Software

	XOR-DBMS	XML-DBMS	XOO-DBMS	XML-DOCM	e-XMLSuite
Standards Compliant	No	Possible	Possible	No	Yes
Query Language	SQL+	XQuery	OQL+	Doc+	XQuery
Application compatible	Possible	No	No	No	Yes
Order Preserving	No	Yes	Yes	Yes	Yes
Distributed Queries	No	Possible	Possible	No	Yes
Heterogeneous Data	No	No	No	No	Yes
Adaptable	No	No	No	No	Yes

7 Conclusion

In this paper, we have presented the e-XML component suite, a suite of components based on XML. The e-XML Mediator federates XML documents, HTML pages, and SQL tables; it unifies them in a virtual XML digital library. It offers the XQuery query language to query the virtual libraries of XML documents. The e-XML Repository provides the necessary functionalities to store and manipulate XML documents on top of an object-relational DBMS. Coupling the capabilities of the Repository with multimedia features now available in most object-relational DBMS gives powerful functions to retrieve documents by contents, e.g., using similarity search on keywords or image features.

The suite can be used to built various architectures for:

- Developing modern web sites;
- Federating heterogeneous data sources;
- Building XML datawebhouse for decision support, data presentation or publication;
- Caching Web data in main memory databases;
- And much more.

In addition, the suite pushes the development of software towards component-ware, thus making possible the cross-fertilization with external components, including EJB libraries.

References

1. M. Fernandez, A. Morishima, D. Suciu : Efficient Evaluation of XML Middleware Queries. In: Proc. of ACM SIGMOD Conf. On Management of data, Santa Barbara, CA (2001) 103-114
2. V. Christophides, S. Cluet, J. Simeon : On Wrapping Query Language and Efficient XML Integration. In: Proc. of ACM SIGMOD Conf. On Management of data, Dallas, TX (2000) 141-152
3. I. Manolescu, D. Florescu, D. Kossman : Answering XML Queries over Heterogeneous Data Sources. In : Proc. of VLDB 27th Conf. On Very Large Data Bases, ROMA, IT (2001) 241-250
4. H. Naacke, G. Gardarin, A. Tomasic : Leveraging Mediator Cost Models with Heterogeneous Data Sources. ICDE Conf. On data Engineering, Orlando, FL (1998) 351-360
5. O. Bukhres and A. Elmagarmid Editors : Object Oriented Multidatabases Systems : A Solution for Advanced Applications, Book, Prentice Hall (1996).
6. M. Carey, L. Haas, et. al. : Towards Heterogeneous Multimedia Information Systems – The Garlic Approach. In: Proc of International Workshop on Research Issues in Data Engineering - Distributed Object Management, Taipei, Taiwan (1995).
7. World Wide Web Consortium (W3C) : XQuery 1.0 – An XML Query Language, W3C Working Draft (2001).