XDB-IPG: An Extensible Database Architecture for an Information Grid of Heterogeneous and Distributed Information Resources

David A. Maluf Ph.D.[1], David G. Bell Ph.D.[2], Chris Knight[1], Peter Tran[3], Tracy La[4], Jenessa Lin[1], Bill McDermott[1], Barney Pell Ph.D.[2]

[1]*NASA Ames Research Center (NASA ARC)*
Mail Stop 269-4
Moffett Field, CA 94035-1000
USA

[2]*Research Institute for Advanced Computer Science at NASA ARC*
[3]*QSS Group Inc. at NASA ARC*
[4]*Computer Sciences Corporation at NASA ARC*

Abstract

This paper describes XDB-IPG, an open and extensible database architecture that supports efficient and flexible integration of heterogeneous and distributed information resources. XDB-IPG provides a novel "schema-less" database approach using a document-centered object-relational XML database mapping. This enables structured, unstructured, and semi-structured information to be integrated without requiring document schemas or translation tables. XDB-IPG utilizes existing international protocol standards of the World Wide Web Consortium Architecture Domain and the Internet Engineering Task Force, primarily HTTP, XML and WebDAV . Through a combination of these international protocols, universal database record identifiers, and physical address data types, XDB-IPG enables an unlimited number of desktops and distributed information sources to be linked seamlessly and efficiently into an information grid. XDB-IPG has been used to create a powerful set of novel information management systems for a variety of scientific and engineering applications.

Introduction – *The Information Grid*

The Information Power Grid (IPG) is the National Aeronautics and Space Administration's (NASA) project for providing seamless access to distributed information resources regardless of location [29]. The project addresses three major categories of distributed resources: 1) hardware resources, such as super computers and scientific instruments, 2) software resources, such as teamwork and CAD programs, and 3) data resources, such as data archives and document databases.

This paper is concerned with data resources. While much recent work in this area is focused on access to structured data archives, our focus is on integrating structured, semi-structured, and unstructured information. As with many enterprises, information and information processes services at NASA are highly distributed. NASA and its contractors have hundreds of databases with millions of records and hundreds of desktop computers with millions of files. The formats and structures of the information are diverse with hundreds of file-types and hundreds of thousands of explicit and implicit structures. The decision making applications that utilize this information are numerous with hundreds of procedures and guidelines and hundreds of thousands of diverse work practices. We seek to create an Information Grid that will provide seamless integration of these distributed heterogeneous information resources for distributed heterogeneous scientific and engineering applications.

XDB-IPG is a novel architecture that enables the creation of such an Information Grid. XDB-IPG is built upon three standards from the World Wide Web Consortium (W3C) Architecture Domain and the Internet Engineering Task Force: 1) HTTP: Hypertext Transfer Protocol – a successful and stable request/response protocol standard, 2) XML: Extensible Markup Language – A ubiquitous five-year old standard that defines a syntax for exchange of logically structured information on the web, and 3) WebDAV – A widely supported four-year old standard that defines HTTP extensions for distributed management of web resources. While the third of these standards was primarily designed for distributed authoring and versioning of web content, XDB-IPG leverages WebDAV for management of arbitrary information resources including information processing services.

Through a combination of these international protocols, universal database record identifiers, and physical address data types, XDB-IPG provides a number of capabilities for managing distributed and heterogeneous information resources such as the following:

- storing and retrieving information about resources using properties
- locking and unlocking resources to provide serialized access
- getting and putting information in heterogeneous formats
- copying, moving and organizing resources through hierarchy and network relations
- automatic decomposition of information into a query-able XML database
- context+content querying of information in the XML database
- sequencing workflows of information processing tasks
- seamless access to information in diverse formats and structures
- a common protocol for human and computer interface to grid services

XDB-IPG enables an unlimited number of desktops and distributed information sources to be linked seamlessly and efficiently into a highly scalable information grid as shown in Figure 1. XDB-IPG thus represents a flexible, high-throughput open architecture for managing, storing, and searching unstructured or semi-structured data. XDB-IPG provides automatic data management, storage, retrieval, and discovery [27] in transforming large quantities of highly complex and constantly changing heterogeneous data formats into a well-structured, common standard.

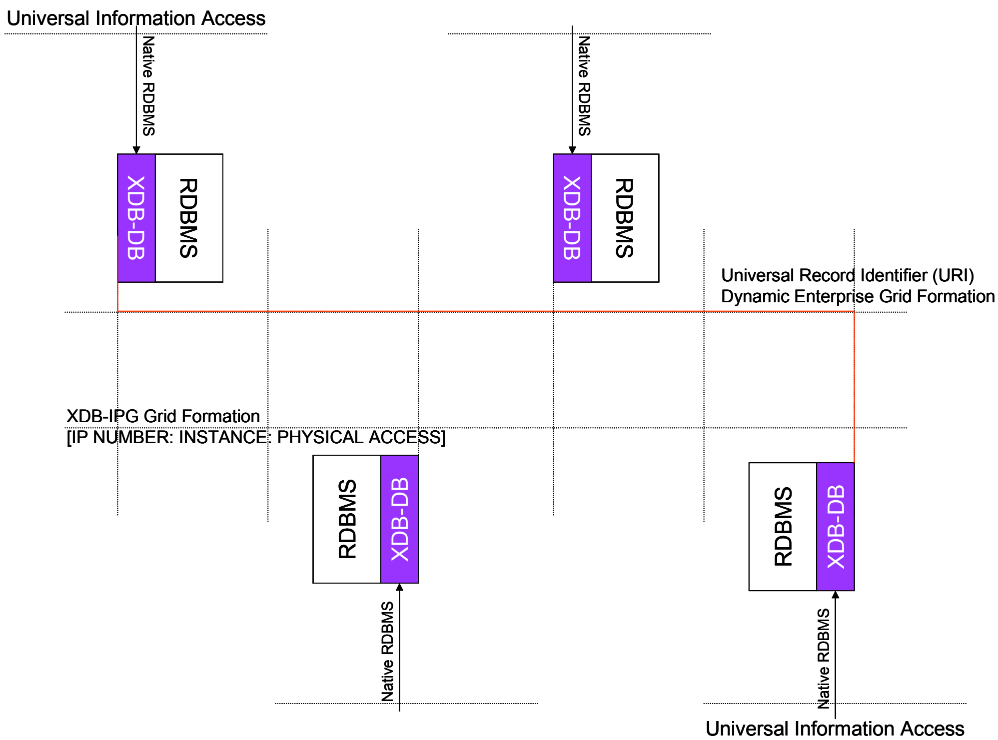

Figure 1: Information Grid using the XDB-IPG architecture

In the next section, we provide background on the primary building blocks for XDB-IPG.

Object-Relational Database Technology

During the early years of database technology, there were two opposing research and development directions, namely the relational model originally formalized by Codd [1] in 1970 and the object-oriented, semantic database model [2][3]. The traditional relational model revolutionized the field by separating logical data representation from physical implementation. The relational model has been developed into a mature and proven database technology holding a majority stake of the commercial database market along with the official standardization of the Structured Query Language (SQL)[1] by ISO and ANSI committees.

The semantic model leveraged from the object-oriented paradigm of programming languages, such as the availability of convenient data abstraction mechanisms, and the realization of the impedance mismatch [4] dilemma faced between the popular object-oriented programming languages and the underlining relational database management systems (RDBMS). Impedance mismatch here refers to the problem faced by both database programmers and application developers, in which the way the

---

[1] The Structured Query Language (SQL) is the relational standard defined by ANSI (the American National Standard Institute) in 1986 as SQL1 or SQL86 and revised and enhanced in 1992 as SQL2 or SQL92.

developers structure data is not the same as the way the database structures it. Therefore, the developers are required to write large and complex amounts of object-to-relational mapping code to convert data, which is being inserted into a tabular format the database can understand. Likewise, the developers must convert the relational information returned from the database into the object format developers require for their programs. Today, in order to solve the impedance mismatch problem and take advantage of these two popular database models, commercial enterprise database management systems (DBMS), have an integrated hybrid cooperative approach of an object-relational model [5].

The object-relational model takes the best practices of both relational and object-oriented, semantic views to decouple the complexity of handling massively rich data representations and their complex interrelationships. ORDBMS employs a data model that attempts to incorporate object-oriented features into traditional relational database systems. All database information is still stored within relations (tables), but some of the tabular attributes may have richer data structures. It was developed to solve some of the inadequacies associated with storing large and complex multimedia objects, such as audio, video, and image files, within traditional RDBMS. As an intermediate hybrid cooperative model, the ORDBMS combined the flexibility, scalability, and security of using existing relational systems along with extensible object-oriented features, such as data abstraction, encapsulation, inheritance, and polymorphism.

In order to understand the benefits of ORDBMS, a comparison of the other models need to be taken into account. The 2x3 database application classification matrix [6] shown in Figure 2 displays the six categories of DBMS applications—simple data without queries (file systems), simple data with queries (RDBMS), complex data without queries (OODBMS), complex data with queries (ORDBMS), distributed complex data without queries (Grid FTP), and distributed complex data with queries (Grid XDB-IPG). For the upper left-handed corner of the matrix, traditional business data processing, such as storing and managing employee information, with simple normalized attributes such as numbers (integers or floats) and character strings, usually needs to utilize SQL queries to retrieve relevant information. Thus, RDBMS is well suited for traditional business processing; but this model cannot query complex data such as word processing documents as if they were a structured database. The lower middle cell describes the use of persistent object-oriented languages to store complex data objects and their relationships. The lower middle cell represents OODBMS which either have very little SQL-like queries support or none at all. The upper middle with the light blue colored cell is well suited for complex and flexible database applications that need complex data creation, such as large objects to store word processing documents, and SQL queries to retrieve relevant information from within these documents. XDB-IPG is the upper right-handed corner with the light blue colored cell as indicated in Table below, and supports queries over complex data that is distributed.

Figure 2: Mapping the evolution of data complexity and database trends (adapted).

The main advantages of ORDBMS, are scalability, performance, and broad support by vendors. ORDBMS handle very large and complex applications. Most ORDBMS supports the SQL3 [10] specifications or its extended form. The two basic characteristics of SQL3 are crudely separated into its "relational features" and its "object-oriented features". The relational features for SQL3 consist of new data types [such as large objects or LOB and its variants]. The object-oriented features of SQL3 include structured user-defined types called abstract data types (ADT) [11][13] which can be hierarchical defined (inheritance feature), invocation routines called methods, and REF types that provides reference values for unique row objects defined by object identifier (OID) [13]. These new features are heavily exploited by XDB-IPG.

In order to take advantage of the object-relational (OR) model defined within an object-relational database system (ORDBMS) [5][6], a standard for common data representation and exchange is needed. Today, the emerging standard is the eXtensible Markup Language (XML) [7][8][9], commonly viewed to be the next generation of HTML for placing structure within documents.

Structuring with XML

XML is known as the next generation of HTML and a simplified subset of the Standard Generalized Markup Language (SGML)[1]. XML is both a semantic and structured markup language [7]. The basic principle behind XML is simple. A set of meaningful, user-defined tags surrounding the data elements describes a document's structure as well as its meaning without describing how the document should be formatted [16]. This enables XML to be a well-suitable meta-markup language for handling loosely structured or *semi-structured data*, because the standard does not place any restrictions on the tags or the nesting relationships. Semi-structured data here refers to data that may be irregular or incomplete, and its structure can be rapidly changing and unpredictable [16]. Good examples of semi-

[1] The Standard Generalized Markup Language (SGML) is the official International Standard (ISO 8879) adopted by the world's largest producers of documents, but is very complex. Both XML and HTML are subsets of SGML.

structured data are web pages and constantly changing word processing documents being modified on a weekly or monthly basis.

XML encoding, although more verbose than database tables or object definitions, provides the information in a more convenient and usable format from a data management perspective. In addition, the XML data can be transformed and rendered using simple eXtensible Stylesheet Language (XSL) specifications [8]. It can be validated against a set of grammar rules and logical definitions defined within the Document Type Definitions (DTDs) or XML Schema [19] much the same functionality as a traditional database schema.

Since XML is a document and not a data model, the ability to map XML-encoded information into a true data model is needed. XDB-IPG allows this to occur by employing a customizable data type definition structure defined by parsing dynamically the hierarchical model structure of XML data instead of any particular persistent schema representation. The customizable driver simulates the Document Object Model (DOM) Level 1 specifications [20] on parsing and decomposition of elements. The XDB-IPG driver is more effective on decomposition than most commercial DOM parsers, since it is less syntax sensitive and guarantees an output ("garbage in, garbage out"), when compared to most commercial parsers. The node data type format is based on a simplified variant of the Object Exchange Model (OEM) [28] researched at Stanford University, which is very similar to XML tags. The node data type contains an object identifier (node identifier) and the corresponding data type. Traditional object-relational mapping from XML to relational database schema models the data within the XML documents as a tree of objects that are specific to the data in the document [19]. In this model, element type with attributes, content, or complex element types are generally modeled as object classes. Element types with parsed character data (PCDATA) and attributes are modeled as scalar types. This model is then mapped to the relational database using traditional object-relational mapping techniques or via SQL3 object views. Therefore, classes are mapped to tables, scalar types are mapped to columns, and object-valued properties are mapped to key pairs (both primary and foreign). This traditional mapping model is limited since the object tree structure is different for each set of XML documents. On the other hand, the XDB-IPG SGML parser models the document itself (similar to the DOM), and its object tree structure is the same for all XML documents. Thus, XDB-IPG parser is designed to be independent of any particular XML document schemas and is termed to be schema-less.

UNIVERSAL DATABASE RECORD IDENTIFIER (UDRI)

Universal Database Record Identifier (UDRI) is intended to be a subset to the Uniform Resource Locator (URL) and provide an extensible means for identifying universally database records. This specification of URI syntax and semantics is derived from concepts introduced by the World Wide Web global information initiative, and is described in "Universal Resource Identifiers in WWW" [RFC1630].

Universal access provides several benefits: it allows different types of databases to be used in the same context, even when the mechanisms used to access those resources may differ. It allows uniform semantic interpretation of common syntactic conventions across different types of records identifiers; and, it allows the identifiers to be reused in many different contexts, thus permitting new applications or protocols to leverage on pre-existing, largely, and widely-used set of record identifiers.

The UDRI syntax is designed with a global transcribability and adaptability to URI standard as one of its main principles. A UDRI is a sequence of characters from a very limited set, i.e. the letters of the basic Latin alphabet, digits, and special characters. A UDRI may be represented in a variety of ways as a sequence of coded character set. The interpretation of a UDRI depends only on the characters used.

The UDRI syntax is a scheme derived from URI. In general, absolute URI are written as follows:

<scheme>:<scheme-specific-part>

An absolute URI contains the name of the scheme being used (<scheme>) followed by a colon (":") and then a string (the <scheme-specific part>) whose interpretation depends on the scheme. The XDB-IPG delineates the scheme to IPG where the scheme-specific-part delineates the ORDBMS static definitions.

RELATIONAL DATABASE ROWID SUPPORT

ROWID is a data type that stores either physical or logical addresses (row identifiers) to every row within the relational database [15]. Physical ROWIDs store the addresses of ordinary table records, clustered tables, indexes, table partitions and sub-partitions, index partitions and sub-partitions, while logical ROWIDs store the row addresses within indexed-organized tables for building secondary indexes. Each relational database table must have an implicit pseudo-column called ROWID, which can be retrieved by a simple SELECT query on the particular table and by bypass index search or table scan. Physical ROWIDs provide the fastest access to any record within an Oracle table with a single read block access, while logical ROWIDs provide fast access for highly volatile tables. A ROWID is guaranteed to not change unless the rows it references is deleted from the database.

XDB-IPG INTERFACES

The XDB-IPG API contains two major sets of interfaces: the first is the JDBC, ODBC, and C/C++ API for application writers, and the second is the lower-level SQL and corresponding server level procedure language API for driver writers. XDB-IPG drivers fit into one of four categories. Applications and servers (server-to-sever) can access XDB-IPG using standard compliant SQL technology-based drivers in particular for XDB-IPG core schema-less configuration.

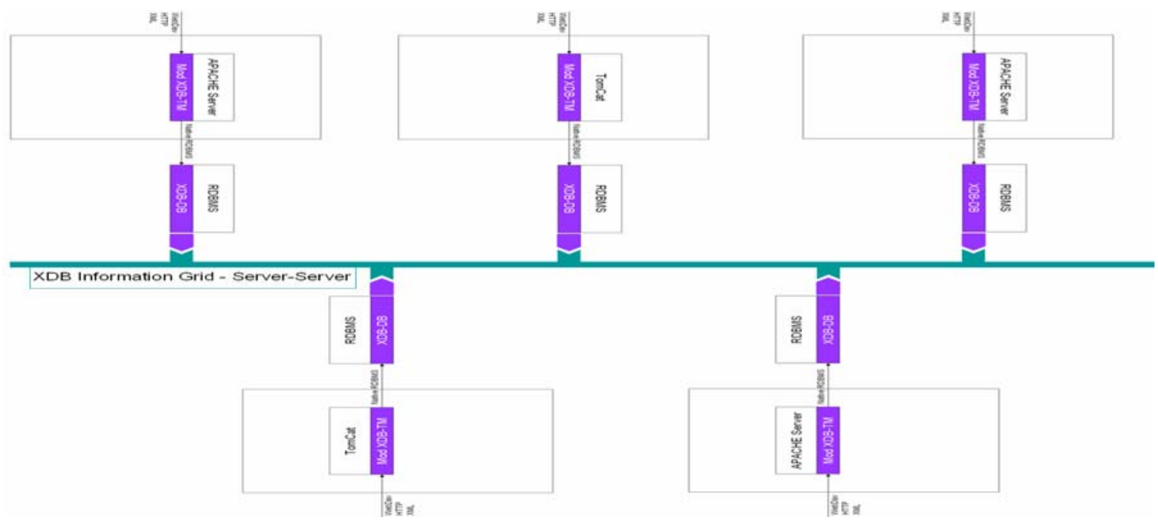

Figure 3: server to server mapping and providing a practical solution for intranet access.

Direct-to-Database: This style of driver converts JDBC, ODBC and C/C++ calls into the network protocol (or localized proprietary protocols) used directly by the data management system, allowing a direct call from the client machine to the server , server to server mapping and providing a practical solution for intranet access.

Transaction level Driver for Multipurpose Middleware: This style of driver translates JDBC, ODBC and C/C++ calls into the vendor's middleware protocol, which is then translated to a XDB-IPG by the middleware server. The middleware provides connectivity to many systems including file systems.

Benefits of XDB-IPG

With the XDB-IPG driver API, no configuration is required on the server side. With a driver written in the C/C++ programming language, all the information needed to map the information content is completely seamless as defined by the markup language or the Direct Access Virtual Information Directory object to be registered with XDB-IPG. The XDB-IPG driver library does not require special installation. It can be set to be automatically downloaded as part of the system that makes the XDB-IPG calls. This reduces the complexity of many data access tasks, and reducing both the upfront development costs for applications and the follow-on database administration maintenance costs.

The XDB-IPG API provides metadata access that enables the development of sophisticated applications that need to understand the underlying facilities and capabilities of specific information, a typical example is the metadata used in for Web-based Distributed Authoring and Versioning (WebDav). XDB-IPG technology also exploits the advantages of existing definition of Enterprise database management systems standard to manage information objects. The XDB-IPG API includes an even better way to identify and connect to a data source, using Direct Access Virtual Information Directory objects that make code even more portable and easier to maintain.

NASA is working with an array of companies in the industry to create and rapidly establish a NASA leadership API as the industry-standard, open interface for XDB-IPG applications to access databases. These leading middleware and tool vendors will provide support for XDB-IPG technology in many new products. This ensures that government, academic, and industrial entities can build portable applications while choosing from a wide range of competitive products for the solution best suited to their needs with little need to export and import the actual content and information.

Building Applications with XDB-IPG: Example 1 – Netmark

Netmark is an initial application that was built using the current XDB-IPG architecture. Netmark is comprised of a distributed, *information on demand* model for document management. Modules in the Netmark example are extensible and adaptable to different data sources. This example consists of (1) a set of interfaces to support various communication protocols (such as HTTP, WebDAV, FTP, RMI-IIOP and their secure variants), (2) an information bus to communicate between the client interfaces and the XDB-IPG core components, (3) the daemon process for automatic processing of inputs, (4) the XDB-IPG search on both document context plus content, (5) a set of extensible application programming interfaces (APIs), (6) and the XDB-IPG implementation for Oracle backend ORDBMS.

The three core components of XDB-IPG consist of the high-throughput information bus, the asynchronous daemon process, and the set of customizable and extensible APIs built on Java enterprise technology (J2EE) [22] and Oracle PL/SQL stored procedures and packages [13]. The XDB-IPG information bus allows virtually three major communication protocols heavily used today—namely HTTP/WebDAV and its secure HTTP web-based protocol, the WebDaV, and the new Remote Method Invocation (RMI) over Internet Inter-Orb Protocol [23][24] from the Object Management Group (OMG) Java-CORBA standards—to meet the information on demand model. The XDB-IPG daemon is a unidirectional asynchronous process to increase performance and scalability compared to traditional synchronous models, such as Remote Procedure Call (RPC) or Java RMI [23] mechanisms. The set of extensible C/C++, JNI and Java and PL/SQL APIs are used to enhance database access and data manipulation from most applications.

The information bus is comprised of a XDB-IPG module Apache HTTP web server and JNI interface integrated with Tomcat Java-based JSP/Servlet container engine [25]. It waits for incoming requests from the various clients, such as an uploaded word processing document from a web folder enabled browser. The bus performs a series of format conversions based on file-type. For instance, the XDB-IPG information bus will automatically convert a semi-structured Microsoft Word document into either a well structured HTML or XML format either on the client side or on the server side. A copy of the original word document, the converted HTML or XML file, and a series of dynamically generated configuration files containing additional pertinent but optional meta-data are added.

A novel aspect of Netmark is that the daemon process automatically converts heterogeneous documents such as word processing documents, presentations and spreadsheets into HTML or XML, the parser then stores the content in a connected node structure in the ‘schema-less’ database, and then a toolbar enables users to query the heterogeneous documents as if they were a database. By entering context+content keywords and phrases into the toolbar, users can retrieve the specific records of data from the heterogeneous documents (e.g., a section of a word processing document, or a cell of a spreadsheet).

CONCLUSION

XDB-IPG provides an extensible, schema-less, information on demand architecture that enables distributed and heterogeneous information resources to be integrated for a variety of scientific and engineering applications. XDB-IPG is a scalable, high-throughput open database framework for transforming unstructured or semi-structured documents into well-structured and standardized XML and/or HTML formats, and for managing, storing and retrieving unstructured and/or semi-structured data. Future plans for XDB-IPG include extending additional node data types, such as the SIMULATION node for handling other complex data sources.

ACKNOWLEDGEMENTS

XDB-IPG is the derivative of the NASA Information Technology-Base program, now part of the overall NASA Computing, Information, and Communication Technologies (CICT) program.

REFERENCES

[1] E. F. Codd, “A Relational Model of Data for Large Shared Data Banks”; Communications of the ACM, Vol. 13, No. 6, pp. 377-387, June 1970

[2] R. Hull and R. King, “Semantic Database Modeling: Survey, Applications, and Research Issues”; ACM Computing Surveys, Vol. 19, No. 3, pp. 201-260, September 1987

[3] A. F. Cardenas and D. McLeod (Editors), “Research Foundations in Object-Oriented and Semantic Database Systems”; pp. 32-35, Prentice-Hall, 1990

[4] J. Chen and Q. Huang, “Eliminating the Impedance Mismatch Between Relational Systems and Object-Oriented Programming Languages”; Monash University, Australia, 1995

[5] R. S. Devarakonda, “Object-Relational Database Systems – The Road Ahead”; ACM Crossroads Student Magazine, February 2001,
http://www.acm.org/crossroads/xrds7-3/ordbms.html

[6] M. Stonebraker, “Object-Relational DBMS - The Next Wave”, Informix Software (now part of the IBM Corp. family), Menlo Park, CA

[7] E. R. Harold, “XML: Extensible Markup Language”; pp. 23-55, IDG Books Worldwide, 1998

[8] Extensible Markup Language (XML) World Wide Web Consortium (W3C) Recommendation, October 2000, http://www.w3c.org/TR/REC-xml

[9] “The XML Industry Portal”; XML Research Topics, 2001, http://www.xml.org/xml/resources_cover.shtml

[10] A. Eisenberg and J. Melton, “SQL:1999, formerly known as SQL3”; 1999,
http://www.incits.org/press/1996/pr96067.htm

[11] ISO/IEC 9075:1999, “Information Technology—Database Language—SQL—Part 1: Framework (SQL/Framework)”, 1999

[12] American National Standards Institute (ANSI), http://web.ansi.org

[13] K. Loney and G. Koch, “Oracle 8i: The Complete Reference”; Oracle Press Osborne/McGraw-Hill, 10th Edition, pp. 69-85; pp. 574-580; pp. 616-644; pp. 646-663, 2000

[14] D. Megginson, “Structuring XML Documents”; pp. 43-70, Prentice-Hall, 1998

[15] Oracle Technology Network (OTN), “Oracle 8i Concepts Release 8.1.5”; Ch. 12 Built-In Data Types, pp. 9-

14, Oracle Corp. 1999,
http://technet.oracle.com/doc/server.815/a67781/c10datyp.htm

[16] J. Widom, "Data Management for XML Research Directions"; Stanford University, June 1999, http://www-db.stanford.edu/lore/pubs/index.html

[17] Lore XML DBMS project, Stanford University, 1998, http://www-db.stanford.edu/lore/research/

[18] H. F. Korth and A. Silberschatz, "Database System Concepts"; pp. 173-200, McGraw-Hill, 1986

[19] R. Bourret, "Mapping DTD to Databases"; O'Reilly & Associates, 2000, http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html

[20] L. Wood et al., "Document Object Model (DOM) Level 1 Specification", W3C Recommendation, October 1998, http://www.w3c.org/DOM/

[21] R. Bourret, "XML and Databases"; XML-DBMS, February 2002, http://www.rpbourret.com/xmldbms/

[22] Java 2 Enterprise Edition (J2EE) technology, Sun Microsystems, http://java.sun.com/j2ee/

[23] ava-CORBA RMI-IIOP Protocol, Sun Microsystems and IBM Corp., http://java.sun.com/products/rmi-iiop/

[24] Java Language to IDL Mapping, Object Management Group (OMG), July 2000, http://cgi.omg.org/cgi-bin/doc?ptc/00-01-06

[25] Apache Software Foundation, Jakarta-Tomcat JSP/Servlet Project, 2000 http://jakarta.apache.org/tomcat/index.html

[26] M. B. Jones, C. Berkley, J. Bojilova, and M. Schildhauer, "Managing Scientific Metadata"; IEEE Internet Computing, pp. 59-68, October 2001

[27] D. A. Maluf and P. B. Tran, "Articulation Management for Intelligent Integration of Information";IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews, Vol. 31, No. 4, pp. 485-496, November 2001

[28] R. Goldman, S. Chawathe, A. Crespo, and J. McHugh, "A Standard Textual Interchange Format for the Object Exchange Model (OEM)"; Database Group, Stanford University, 1996, http://www-db.stanford.edu/~mchughj/oemsyntax/oemsyntax.html

[29] NASA Information Power Grid (IPG);http://www.ipg.nasa.gov/