

JSIM Database: A Web-based Database Application Using XML

Xueqin Huang, University of Georgia, xueqin@cs.uga.edu

Hui Tian, University of Georgia, hui@cs.uga.edu

Xiangming Xu, University of Georgia, xu@math.uga.edu

Abstract- This paper presents an approach of enabling object-relational databases with XML by introducing a three-tier Web-based database application using an XML-SQL utility. We hope to identify some problems arising from managing XML documents as well as the benefits of this new technology. Our purpose is to contribute to a better understanding of this promising technology among the database community and to introduce one practical approach to applying XML to databases. We are fully aware that our approach represents only one partial solution to the problem of how to bridge the two worlds of XML and databases. In fact, this is still an unsolved problem. Currently, research work on XML Schema and XML Query languages are being actively carried out by the World Wide Web Consortium and some research labs from the academia and industry.

1. Introduction

XML (Extensible Markup Language) is an effort by the World Wide Web Consortium aimed at facilitating the exchange of information on the web. Like HTML, it is a subset of the Standard Generalized Markup Language (SGML). But more than HTML, XML is a meta-language which allows you to define your own set of tags and describe your data in your own logic. With its entire specification covered in about 20 pages, it could not be simpler. Yet, the linking and nesting mechanisms embodied in its logical and physical structure are sophisticated enough to capture complex relationships in the real world. It offers you the ultimate flexibility in choosing how simple or how complex the structure of your data should be, and how to present your data in different ways without changing the structure of your data. Above all, XML is a standard for information interchange. It frees you from worrying about how to exchange your data with other XML-aware applications.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

©2000 ACM 1-58113-250-6/00/0004

\$5.00

1.1 Why XML

XML has been developed out of the pressing need of today's enterprises to make better use of the exploding information on the Web. According to CNN, the non-Internet sectors of the United States economy today are at best not growing while the Internet sector has been witnessing a strong momentum for rapid growth. To strive for survival, today's enterprises are forced to plan for enterprise-wide computing environments that can provide their customers and suppliers easy access to information about the enterprises and quickly process feedback from their customers and suppliers. While the Web has brought about unprecedented opportunities, there are apparently some open problems with the current Web technology. One of the most obvious problems is the inadequacy of HTML functionality to support the requirements of complex Web-based applications. HTML has a fixed tag set, a flat structure, and tight coupling of structure and presentation of information. It is primarily designed to present information to human beings. It works well when Web contents are meant for humans to read. However, as the Web continues to grow and becomes a valuable asset to the enterprises, to automatically process web contents is the only choice for sophisticated applications, such as search engines. HTML pages are difficult to understand for an application program. HTML is also too simple for those applications that need to dynamically retrieve information from heterogeneous databases and present different views of the same data to different users.

On the other hand, XML is a promising technology that vows to be a better alternative to HTML. XML documents are both human- and machine-readable. In fact, to facilitate automatic processing of XML documents is a design goal of XML. As a standard for information interchange, XML allows easy integration of data from heterogeneous databases that support XML. The separation of structure and presentation of XML data makes it possible for a client to dynamically change the view of an XML document upon the user's request. By shifting processing load to the client side, it is expected to significantly improve response time while reducing network traffic. In addition, most XML parsers support the Document Object Model (DOM) API, making it easy for XML documents to be manipulated by application programs as DOM objects.

1.2 How to Manage XML Documents

With so many benefits and tremendous industry support, XML is becoming real. But how to manage XML documents? Are there any tools and technologies out there that can help translate existing relational data into XML documents? These are legitimate questions to ask if you are seriously considering using XML for your application. There are basically two kinds of documents, data-centric documents and document-centric documents [1]. Data-centric documents are highly structured and can be easily mapped to a database structure, whereas document-centric documents are loosely structured and better be managed using a content management system, which is a document management system and an underlying database management system combined into one.

A natural way to manage XML documents is to use an object-oriented database management system. By storing an XML document as an object, no information will be lost due to the mapping between a document structure and a database structure. Unfortunately, although there are some OODB standards, such as the ODMG2.0, not many available OODB products fully support the standard. This is mainly because an OODBMS is not backward compatible with legacy systems and partly because it does not have the capability to support sophisticated queries [2].

On the other hand, there are some inherent difficulties in mapping between the structure of an XML document and a relational database structure due to the restrictions of both XML and relational database. An XML document doesn't contain adequate information about data type and storage size that is required by a database, and a relational database cannot handle arbitrarily nested structures that are characteristic of XML documents. A good news is, an XML schema modeling language is being developed by the W3C which will incorporate data type and storage size information into an XML document to make it more suitable for storing in database [3]. Also, efforts are being made by some object-relational database vendors to extend the capability of relational databases by supporting part of the object-oriented features of SQL3. Although the current situation is far from what is desired, at least you can transform existing relational data into XML documents with some ease if you pick the right tools for your application.

1.3 XML Database Tools

Currently, there are numerous tools that support storing XML documents into database and/or retrieving XML documents from database. To name a few, ASP2XML developed by Stonebroom, DBIx::XML_RDB by Matt Sergeant, XML SQL Utility by Oracle, XMLDB and XML Servlet by Cerium Component Software Incorporated, and XML-DBMS by Ronald Bourret. These are all middle-wares that support both directions using relational or object-relational databases [1].

Which tool to use depends on the needs of your application. Our implementation of the JSIM database uses Oracle's XML

SQL Utility. This utility can output XML documents from SQL queries and insert XML documents into tables or object views [4]. It maps an XML document into a set of tables in database. Based on these tables, you can create an object view that is a virtual object table or a snapshot of the data from a certain point of view. When an XML document is stored into an object view, an INSTEAD OF TRIGGER is fired and the data is decomposed and stored in the corresponding tables. When a SQL query is made on an object view, the data from the underlying tables is reassembled by the utility and an XML document is returned. A Document Type Definition (DTD), which specifies the structure of the XML document, can be optionally used. A style sheet using XML Stylesheet Language (XSL) that defines the presentation of the XML document can also be specified if desired. By using the object view concept, you can normalize the base tables to make them optimized for update, and at the same time, present the data as objects in views to database applications. A nice feature of object view is that you can quickly apply the object model to existing relational data without changing its underlying structure and physical storage [5].

2. JSIM: A Web-based Simulation Environment

JSIM [6] is a Java-based simulation environment implemented in Java using applets, JavaBeans [7], and Enterprise Java Beans. It is an ongoing project led by Dr. John A. Miller from the University of Georgia and actively worked on by several graduate students. JSIM is intended to automate those repetitive, time-consuming, and error-prone tasks that are sometimes manually performed by simulation analysts [8]. Our goal is to build a distributed, cooperative, and inter-operable simulation environment with a web-accessible database that provides the desired simulation functionality and a highly intuitive user interface to the simulation analysts. It should be designed to be highly extensible to accommodate for future changes in technology and allow for easy customization.

JSIM supports a point-and-click model designer that can be used to design simulation models and code-generate model applets and model beans. A generated model can run either as an applet or as a bean in a bean builder. A model bean is controlled by a model agent, which specializes in a statistical analysis method. A model agent can also dynamically adjust the properties of a model, collect statistical results from the model, and fire an event to a database bean for storing the model parameters, agent properties, and statistical results into a database. A model bean and a model agent can be dynamically wired in a bean builder so that statistics on the same model but different scenarios, each with different parameters, and perhaps different model agents, can be collected in database and then compared to find the best scenario for this model.

In the future, as JSIM becomes truly distributive, it will need to integrate simulation results stored in different and perhaps heterogeneous databases. To inter-operate with other simulation systems, JSIM also needs to be capable of exchanging model information with other simulation systems.

Thus, choosing a standard format for information interchange is critical to the long-term goal of JSIM.

2.1 Application Design

As we have indicated above, JSIM has several design goals: distributive, cooperative, inter-operable, and a highly intuitive user interface. We decided to enable the JSIM database with XML technology because it is a universal standard. Given the fact that OODB standards are inadequately supported and some RDBMSs have been extended to support some object-oriented features, we choose to use an object-relational database management system, Oracle8i, as our backend system. We use Oracle's XML SQL Utility to store and retrieve JSIM data from an object view. To support a highly intuitive user interface, we choose a query-language independent criteria-based search interface to the JSIM database, and display the returned query results as a table instead of a tagged document whenever an XSL-aware browser is used.

This project has two major parts, store and query the scenarios. The parameters and statistical results of a scenario, collected by a model agent, are stored into Oracle8i using JDBC by the database bean. The query part is mainly done by an HTTP server, which accepts and parses queries from a browser, queries the database using the XML SQL Utility, and then sends XML-enabled query results to the browser. If the browser doesn't support XSL, it displays the document as plain text. Otherwise, it either formats the document by applying the embedded XSL style sheet, or displays the document as an XML tree by default. Currently, the HTTP server is hard-coded for this project, we hope to replace it later with a real web server and an application server supporting Java servlets.

2.1.1 A Three-Tier Model

To build a web application, we must select a correct web-architecture to get the most benefits for our application. Currently, corporations are increasingly moving from classic client/sever two-tier application models to three-tier models, in which a browser front end interacts with a middle-tier Web-server and it in turn communicates with a back-end database server. One of the benefits from the three-tier model is the integration of data from heterogeneous databases, which is needed in future JSIM development. Also, XML can serve as the standard for the data exchange among databases and communication between end user and middle-ware. Hence, The architecture of our application follows a three-tier model, with a web browser as the front-end, an HTTP server and Oracle's XML SQL Utility as the middle-tier, and Oracle8i as the backend storage system, as shown in Figure 1.

2.1.2 Database Design

To find the best scenario for a model, we need to keep track of information about the model, the parameters, the statistical results, and the agents that control the scenarios. In our implementation, each model has a list of nodes representing the

entities participating in the model. Each node is associated with a set of parameters on the corresponding entity and a set of statistical results. A model can have many scenarios. The parameters and statistical results associated with each node differ from scenario to scenario. The resulting database design is presented in UML in Figure 2.

We obtained the schema from ER design and verified it to be in the 3rd Normal Form using the 3NF Synthesis technique. This design is proved to be lossless by the Tableau Lossless Test Algorithm [2]. No functional dependency is lost according to a functional dependency test with the Restricted Closure technique [9].

2.1.3 Creating a Scenario View of the JSIM Database

Object view is Oracle's solution to quickly applying the object model to relational data. An object view is a view of data from the base tables from a certain angle. You can create different views on the same base tables. We use object view because Oracle's XML SQL Utility only supports query or insert on one table or view. (We do not intend to criticize Oracle for this, on the contrary, we agree that objects are more natural ways to model real world entities.)

Since JSIM users will be mainly interested in analyzing the statistical results of the different scenarios of a model, we have created a scenario view on the base tables. Each scenario consists of a model, a list of nodes belonging to the model, each of which representing a set of parameters and the corresponding statistical results, other properties of the scenario, including scenarioID, scenarioName, and date, and the model agent controlling the scenario. The full detail of the scenario view is shown in Figure 3.

2.2 Querying the JSIM Database

We build an intuitive point-and-click query interface for accessing the JSIM database so that users don't have to learn any query language and can be totally unaware of the underlying technology.

At this point, we support searching by model name and number of scenarios. We plan to support more search criteria, such as searching by model location, searching by model agent, and etc., when we have access to an application server supporting Java servlets. Criteria-based searching is made easy and efficient by the meaningful tag names of XML documents.

2.3 Presenting JSIM Data with XSL

Returned query results will be formatted according to an XSL style sheet and presented to a client in a table by an XSL-aware browser. Figure 4 is a snapshot of some query results displayed by a browser using an XSL style sheet.

Unfortunately, only Internet Explorer 5.0 supports XSL at the date of writing. For those who are using other browsers, the query results will be displayed as plain XML documents. A

get-around would be translating the XML document into an HTML style sheet using the DOM APIs.

Since the query results are returned as an XML document, different views of the same query results can be dynamically created at the client side by applying different style sheets to the XML document. Again, we leave this for future work.

3. Summary

This paper presents a prototype three-tier web-based database application using XML. By this application, we have demonstrated how it is technically possible to translate existing relational data into XML documents with the help of some tools and technologies, such as Oracle's XML SQL Utility and the object view concept. We have also pointed out some benefits of using XML, such as meaningful representation of data, integration of data from heterogeneous data sources, and client-side processing of multiple views of the same data. In addition, we have used a query-language independent criteria-based search interface to this new technology. Future work on this project include migrating from the hard-coded HTTP server to a general purpose web server and application server supporting Java servlets, adding support for criteria-based searching, and possibly supporting multiple views of the results of a single query.

Acknowledgements

We owe numerous thanks to Dr. John A. Miller for his invaluable comments on the final draft of this paper and his help on every aspect of this project, especially the database

schema design. We also want to thank him for providing all the resources that we needed to complete the project.

References

1. Ronald Bourret. "XML and Databases". <http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases.htm>
2. Ramez Elmasri, Shamkant B. Navathe. "Fundamentals of Database Systems", 3rd edition. Addison-Wesley publishing Company. 1999.
3. XML Schema Group. 1999. <http://www.w3.org/TR/NOTE-xml-schema-req>
4. XML SQL Utility. 1999. http://technet.oracle.com/tech/xml/oracle_xsu
5. Oracle's Object View Concept. 1999. http://technet.oracle.com/doc/server.815/a68003/01_17obj.htm
6. Rajesh Nair, John A. Miller and ZhiweiZhang. "A Java-Based Query Driven Simulation Environment". Proceedings of the 1996 Winter Simulation Conference (WSC '96), Coronado, California. December 1996. pp. 786-793.
7. John A. Miller, Andrew F. Seila and Xuewei Xiang. "The JSIM Web-Based Simulation Environment". Future General Computer Systems. 1999.
8. Andrew F. Seila and John A. Miller, "Scenario Management in Web-Based Simulation," Proceedings of the 1999 Winter Simulation Conference (WSC'99), Phoenix, Arizona. December 1999.
9. Class notes from Advanced Databases. Taught by Dr. John A. Miller. University of Georgia. Fall 1999.

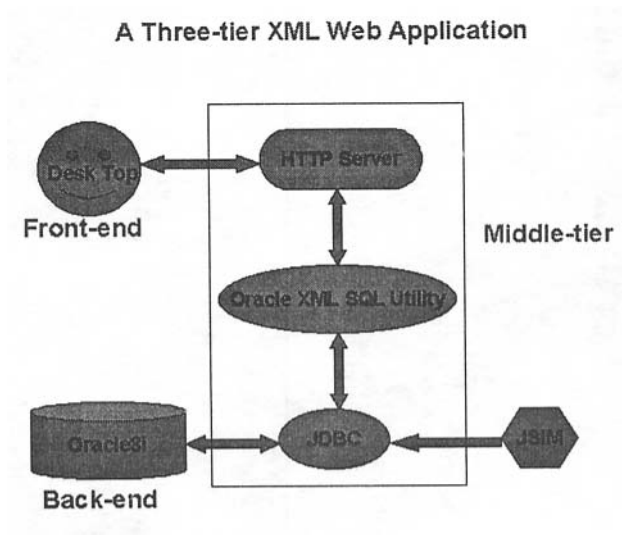
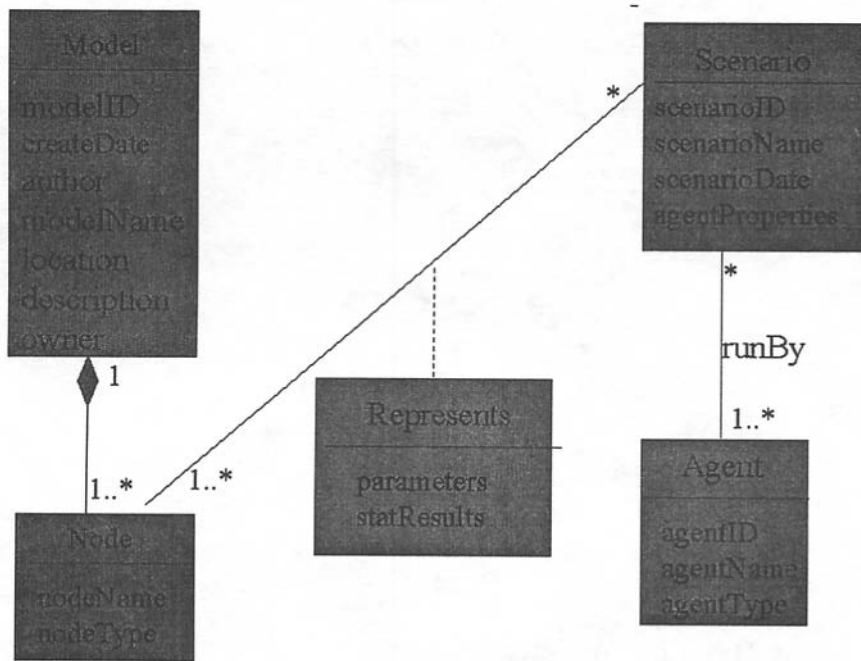


Figure 1: The Three-Tier Architecture for the JSIM Database Application.



Attributes not explicitly drawn in the above diagram:

Parameters (numTokens, distribution(alpha, beta, gamma, stream))

StatResults (timeStat(statName, noSamples, min, max, mean, stdDev, interval, precision),

occuStat(statName, noSamples, min, max, mean, stdDev, interval, precision))

AgentProperties (batchSize, numBatches, replicationSize, numReplications, confidenceLevel, relativePrecision, transientPeriod)

Figure 2: JSIM Database Design in UML

```

CREATE OR REPLACE VIEW scenario_view OF scenario_t
WITH OBJECT IDENTIFIER(scenarioID)
AS SELECT model_t(m.modelID, m.modelName, m.description, m.createDate,
                m.location, m.author, m.owner),
        CAST (MULTISET (
            SELECT n.nodeName, n.nodeType,
                represents_t(r.numTokens, r.alpha, r.beta, r.gamma, r.stream,
                    r.timeStatName, r.tNoSamples, r.tMin, r.tMax,
                    r.tMean, r.tStdDev, r.tInterval, r.tPrecision,
                    r.occuStatName, r.oNoSamples, r.oMin, r.oMax,
                    r.oMean, r.oStdDev, r.oInterval, r.oPrecision)
            FROM node n, represents r
            WHERE n.mID = m.modelID    AND
                  r.mID = n.mID      AND
                  r.nName = n.nodeName AND
                  r.sID = s.scenarioID)
        AS node_list_t),
        agent_t(a.agentID, a.agentName, a.agentType),
        s.scenarioID, s.scenarioName, s.scenarioDate, s.batchSize, s.numBatches,
        s.replicationSize, s.numReplications, s.confidenceLevel, s.relativePrecision,
        s.transientPeriod
FROM model m, scenario s, agent a
WHERE a.agentID = s.aID;

```

Figure 3: Scenario View for the JSIM Database

http://orion.cs.uga.edu:6002/ - Microsoft Internet Explorer

Address http://orion.cs.uga.edu:6002/

JSIM XML Database Search Result

Model Name	Bank	Author	John Miller	Owner	John Miller	Create Date	1999-11-13 23:33:44.0					
AgentName	ScenarioDate	BatchSize	NumberBatches	ReplicationSize	NumberReplications	ConfidenceLevel	RelativePrecision	TransientPeriod				
BatchMeansAgent	1999-11-13 23:33:44	4	8	0	0	0.95	0.10	0				
NodeName	NodeType	NumTokens	Alpha	Beta	Gamma	Stream	TimeStatName	TNoSamples	TMin	TMax	TMean	TStdDev
Customer	3	100	2000	1000	0	0	Customer (dur)	25	1497.83287546	2374.43232320	1993.84326167	241.3892181020
Teller	1	1	2000	1000	0	0	TellerQ (dur)	10	1207.25444909	10041.407602	5115.53984212	3028.46356712
Exit	4	0	0.10	1000	0	0	Exit (dur)	25	5057.65389621	15843.44096580	9573.16995414	3062.91919279

Model Name	Bank	Author	John Miller	Owner	John Miller	Create Date	1999-11-13 23:33:44.0					
AgentName	ScenarioDate	BatchSize	NumberBatches	ReplicationSize	NumberReplications	ConfidenceLevel	RelativePrecision	TransientPeriod				
BatchMeansAgent	1999-11-13 23:55:45	4	8	0	0	0.95	0.10	0				
NodeName	NodeType	NumTokens	Alpha	Beta	Gamma	Stream	TimeStatName	TNoSamples	TMin	TMax	TMean	TStdDev
Customer	3	100	2000	1000	0	0	Customer (dur)	25	1497.83287546	2374.43232320	1993.84326167	241.3892181020
Teller	1	1	2000	1000	0	0	TellerQ (dur)	10	1207.25444909	10041.407602	5115.53984212	3028.46356712
Exit	4	0	0.10	1000	0	0	Exit (dur)	25	5057.65389621	15843.44096580	9573.16995414	3062.91919279

Model Name	Bank	Author	John Miller	Owner	John Miller	Create Date	1999-11-13 23:33:44.0					
AgentName	ScenarioDate	BatchSize	NumberBatches	ReplicationSize	NumberReplications	ConfidenceLevel	RelativePrecision	TransientPeriod				
BatchMeansAgent	1999-11-14 00:08:34	4	8	0	0	0.95	0.10	0				
NodeName	NodeType	NumTokens	Alpha	Beta	Gamma	Stream	TimeStatName	TNoSamples	TMin	TMax	TMean	TStdDev
Customer	3	100	2000	1000	0	0	Customer (dur)	25	1497.83287546	2374.43232320	1993.84326167	241.3892181020
Teller	1	1	2000	1000	0	0	TellerQ (dur)	10	1207.25444909	10041.407602	5115.53984212	3028.46356712
Exit	4	0	0.10	1000	0	0	Exit (dur)	25	5057.65389621	15843.44096580	9573.16995414	3062.91919279

Done

Start http://orion.cs.uga.edu:60... Microsoft PowerPoint - [Pr... http://orion.cs.uga.o... 11:45 PM

Figure 4: Snapshot of returned query results displayed in Internet Explorer using an XSL style sheet