XML TECHNOLOGIES AND/OR RELATIONAL DATABASES: A CLASSROOM EXPERIENCE

Ahmad Abuhejleh
Associate Professor
College of Business & Economics
Computer Science & Information Systems Department
University of Wisconsin – River Falls
River Falls, WI 54022, USA
Email: Ahmad.Abuhejleh@uwrf.edu

Abstract

Extensible Markup Language (XML) is rapidly emerging as the standard for exchanging business data on the World Wide Web. For the foreseeable future, however, most business data will continue to be stored in relational database systems. This paper discusses the emergence of XML and the benefits of its use in conjunction with relational database management systems in the development of web applications in information systems courses. The author will begin with a brief overview of the application and its architecture and will then discuss the various XML technologies that have been used (XSL, XSLT, XML Schema, and XQuery), as well as the experiences with each. The author will explore the relationship between native XML databases and relational databases. Finally, the author will discuss whether XML technologies are robust enough to completely replace relational systems and the benefits/detriments of such a replacement.

Keywords: XML databases, relational databases, XSL, XSLT, XQuery, XML technologies

1. INTRODUCTION

Web pages are created using a document layout language known as Hypertext Markup Language (HTML). Related Web pages are grouped together and called a Web application. HTML is used to create static Web pages, which contains fixed content. Today, individuals and businesses require dynamic Web applications. Dynamic Web sites contain Web pages that exchange information between the Web site and the user. Dynamic Web sites are usually driven from data stored in databases. Examples of Dynamic Web applications include new services that provide access to large data repositories, e-commerce applications such as online stores, and business-to-business (B2B) support products (Doke, 2002; Williams, 2002). This paper describes and compares two approaches for implementing Dynamic Web site. The two approaches are: the traditional relational database theory approach and the new XML technologies approach. The paper also discusses the lessons learned from using various XML technologies (XSL, XSLT, XML Schema, and XQuery) and dynamic relational databases in the development of a web application in an information systems course.

2. THE PROBLEM DESCRIPTION

The application used in this paper can be described as a web based timesheet entry system and material receipt-reconciliation system available via mobile devices.

Background

RCH Builders is a large custom home builder in the Minnesota, Iowa, and Wisconsin tri-state area. Currently, RCH builds 500 custom homes each year, each home valued between $350,000 and $1,000,000. Over the past 5 years, the company has realized 200% growth and predicts another 100% over the next 3 years. RCH currently employs over 1250 employees.

Because of RCH's rapid growth it has become increasingly difficult to disseminate to and retrieve information from hundreds of jobsites being worked on concurrently by RCH staff. Two pieces of critical information, labor costing and site inventory, have become time consuming to gather, inefficient to utilize for jobsite decisions and inaccurate for accounting purposes.

The labor costing process is made up of two major components, time entry from the field and budget reporting from the corporate office. Each is a manual process which is time consuming and error prone.

While carpenters are responsible for recording time on paper, Foremen are responsible for collecting and auditing hard copies of time sheets and submitting to general accounting at corporate headquarters. Accountants then audit for accuracy, code entries, and input into the general accounting system. The process of time sheet entry could take from 12 hours a week at a small contractor to 40 hours for larger firms. The current entry form consists of several different methods of job accounting and numbering. If an error exists in a carpenter's time sheet, extra time is added for reconciliation. In addition, job numbers are now not clearly communicated to each employee, which makes accurate recording difficult.

Job reporting is currently not possible from the field unless a hard copy of a report from the accounting system is given to the foreman. Without this information readily available, Foreman is not able to make critical decisions that affect the homeowners' budget. In addition, because of the problems inherit in the time reporting process, reports given to the foremen are often too out-of-date to be of any use.

Jobsite inventory also poses problems in the construction process. Typically, a foreman will accept materials on-site and the confirmation of that receipt may not be known to accounting for several days. In addition, packing slips from shipments may be lost or destroyed and not available for reconciliation of invoicing. Job tracking is difficult when the accounting office does not immediately know what materials have been delivered and what have not.

We design and implemented a web based timesheet entry system and material receipt-reconciliation system available via mobile devices from the field are developed to eliminate the problems listed above. The two systems will be integrated with RCH's current general accounting and purchasing systems.

The benefits received from an investment in such systems include the reduction of errors in costing and inventory management, increased efficiency in gathering and disseminating information and increased availability of accurate information for critical decisions. In addition, these systems would give RCH a groundwork that opens the capability to serve our clients better in the future by potentially giving them real time access to information related to their construction.

Application Design

Figure 1 shows the core classes of the application. The class diagram provides the developer with an object oriented interface. There are five classes in the application: TimeEntry, Employee, Payroll, Job, and Management. Notice that the Employee, Payroll, and Job classes are inherited from the TimeEntry class. The Management class is inherited from both the Employee and the Job classes.

3. APPLICATION ARCHITECTURE

The Relational Database Approach

Figure 2 describes the overall application architecture, showing both the client and the server side using the relational database approach. The client computer first issues a request to the Web server. The request specifies the address of the Web page to be loaded and could include information supplied by the user, such as the quantity of a product to be purchased. The Web server gathers necessary data from the database server, carries out the logic needed to fulfill the request, and returns a response to the client.

The XML Technologies Approach

Figure 3 describes the overall application architecture, showing both the client and the server side using the XML technologies approach.

The visitor's Web browser requests the Web page using a standard URL. The Web server software such Apache or IIS, recognizes that the requested file is an ASP script (see Figure 4), and so the server interprets the file using ASP plug-in, before responding to the page request. Through Certain ASP commands connect to the XML interpreter and request the content that belongs in the Web page. The XML interpreter (MSXML 4.0) responds by sending the requested content. The XML interpreter outputs the content as part of the Web page. The XML interpreter plug-in finishes up by handing a copy of the HTML it has created to the Web server. Finally The Web server sends the HTML to the Web browser as it would a plain HTML file (see Figure 5), except that instead of coming directly from an HTML file, the page is the output provided by the parser plug-in.

4. XML TECHNOLOGIES

One of the challenges of working with an XML document is presenting the XML data in a useful format, for users of the Web. The method we chose is the Extensible Style sheet Language (XSL). One of the components of XSL is XSLT (Extensible Style sheet Language Transformation). XSLT is used to transform XML content into a presentation format. In our project we used XSLT style sheets. The fact that all the application's data is available in XML leads to the obvious decision to use XSLT for printing and report generation.

Among the important things that we faced during the implementation phase is the parsing of the XML Schema documents describing the various object classes and building an efficient object-based representation of them. One of the major contributors to the implementation was the querying of XML Schema documents with XPath expressions. XPath expressions allows for quickly finding the element associated with a specific class attribute or relationship.

Describing Queries

All queries within the application are described in an XML syntax. The result of a query is an XML document. Queries are used in forms, they are used internally by the application logic and they are used for user-defined queries. However, users do not write their queries in XML. Users can formulate their queries in a Query-by-example style through the browser. The user queries are then transformed into XML and sent to the server. On the server, the XML query syntax is not executed directly, but transformed into the underlying system's native query language, which in our case is XSL.

5. RELATIONAL APPROACH VS. XML APPROACH

For many people, especially those with relational database backgrounds, XML databases raise a number of controversial issues, particularly with respect to issues surrounding the storage of data, mainly normalization and referential integrity.

Normalization

Normalization is a process that reduces data redundancies and helps eliminate the data anomalies that result from those redundancies. Normalization does not eliminate data redundancies; instead, it produces the controlled redundancy that lets us link database tables. Normalization works through a series of stages called normal forms. They are described as first normal form (1NF), second normal form (2NF) and third normal form (3NF) (Rob, 2002).

In our application the group that used the relational approach normalized the tables. For the group that used the XML technologies, we found that there was nothing in XML databases that forces you to normalize your data. There is no systematic approach in XML databases to normalize your table. Therefore, we found great difficulties in eliminating redundancies in the

XML approach. In more involved applications, there are no clear answers. Real-world relational data is often non-normal for performance reasons, so having non-normal XML data might not be as bad as it sounds. It is a decision that the designer need to make based on the application.

Referential Integrity
Closely related to normalization is referential integrity. In relational database, referential integrity means that a foreign key may have either a null entry or an entry that matches the primary key value in a table to which it is related. The enforcement of the referential integrity rule makes it impossible to delete a row in one table, whose primary key has mandatory matching foreign key values in another table. For example, a customer might not have an assigned sales representative but it will be impossible to have an invalid sales representative (Rob, 2002).

In our application the group that used the relational approach had no problem of enforcing the referential integrity rule since the access database enforces such a rule. On the other hand, the group that used the XML approach had to mechanism to enforce the referential integrity rule. The reason for this is that most XML databases do not currently support linking mechanisms, so there are no references whose integrity they can check. Therefore, we have to enforce the referential integrity rule by regular high level coding using ASP scripting language.

6. CONCLUSION AND OUTLOOK

Building a complex application completely based on XML technologies can be done successfully. However, there are many things that must be considered to lead such a project to successful completion. Like HTML, XML documents are text files. Therefore, an XML author needs no more than a simple text editor, like Notepad or vi to get started. There are more sophisticated XML editors available that may make it easier to design a document, but they are not required. After the XML document is created, it needs to be evaluated by an application known as an XML processor, or XML parser. Part of the function of the parser is to interpret the document's code and verify that it satisfies all of the XML specifications for document structure and syntax. XML parsers are strict. If one tag is omitted or a character is lowercase when it should be uppercase, the parser will report an error and rejects the document. This may see excessive, but that rigidity was built into XML to correct the flaw in HTML that gave Web browsers too much discretion interpreting HTML code. The end result is that XML code accepted by the parser is sure to work the same everywhere. In our project we used a parser developed by Microsoft called MSXML. We believe that XML is something of a hybrid. XML is probably most similar to object databases in data modeling, in as much as it also consists of nodes, and nodes can contain heterogeneous data. On the other hand, the degree of heterogeneity of nodes depends a lot on the particular DTDs or schemas used to define the structure of an XML document.

XML is an extremely versatile data *transport* format, but despite high hopes for it, XML is mediocre to poor as a data storage and *access* format. It is not nearly time to throw away your (SQL) relational databases that are tuned to quickly and reliably query complex data. So just what is the relationship between XML and the relational data model? The problem for many XML-everywhere (and XML-only) aspirations is that at the core of an RDBMS are its *relations* -- in particular, the set of constraints that exists between tables. Enforcing the constraints is what makes RDBMSs so useful and powerful. While it would surely be possible to represent a constraint set in XML for purposes of communicating it, XML has no inherent mechanism for enforcing constraints of this sort (DTDs and schemas are constraints of a different, more limited sort). Without constraints, you just have data, not a data model. Overall, we have been quite satisfied with the results of our two approaches and we will use this technologies in future projects too.

7. REFERENCES

Doke, E. Reed, John W. Satzinger, and S. R. Williams (2002), Object-Oriented Application Development Using Java. Boston, MA: Course Technology.

Rob, P., and C. Coronel (2002), Database Systems Design, Implementation & Management. Boston, MA: Course Technology.

Williams E. Hugh, and David Lane (2002), Web Database Applications with PHP & MySQL. Sebastopol, CA: O'Reilly.

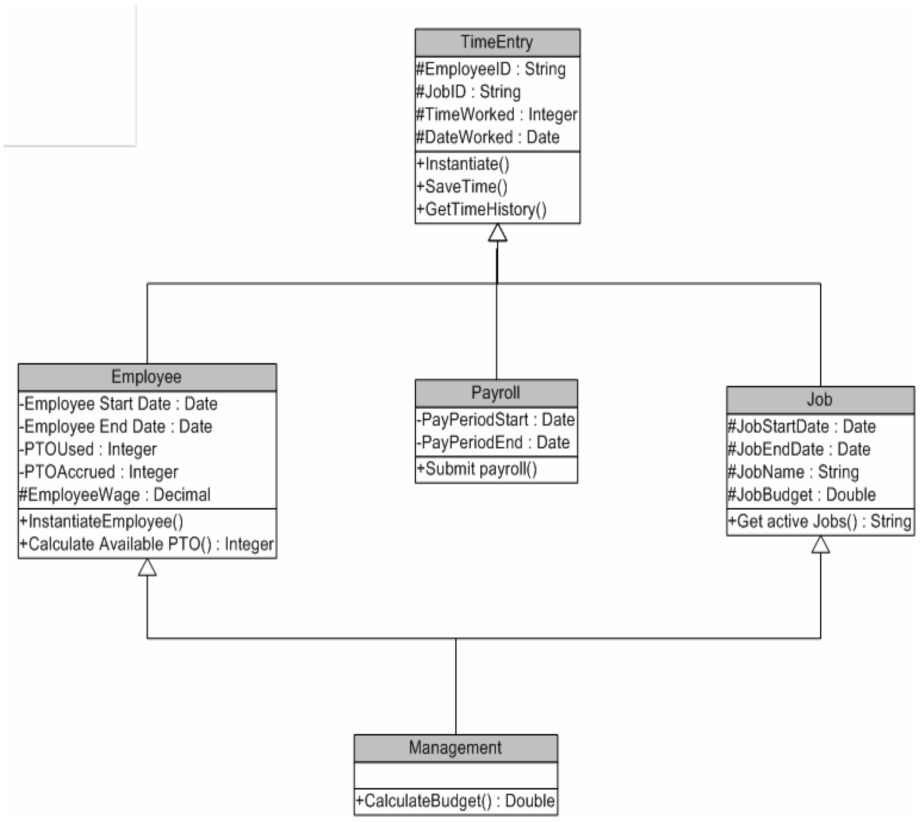


Figure 1: Class Diagram

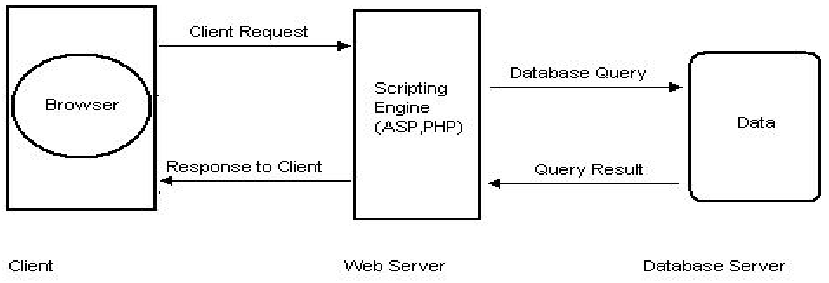


Figure 2: The Relational Approach

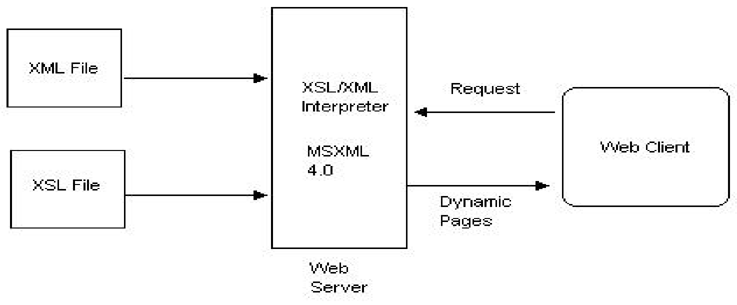


Figure 3: The XML Approach

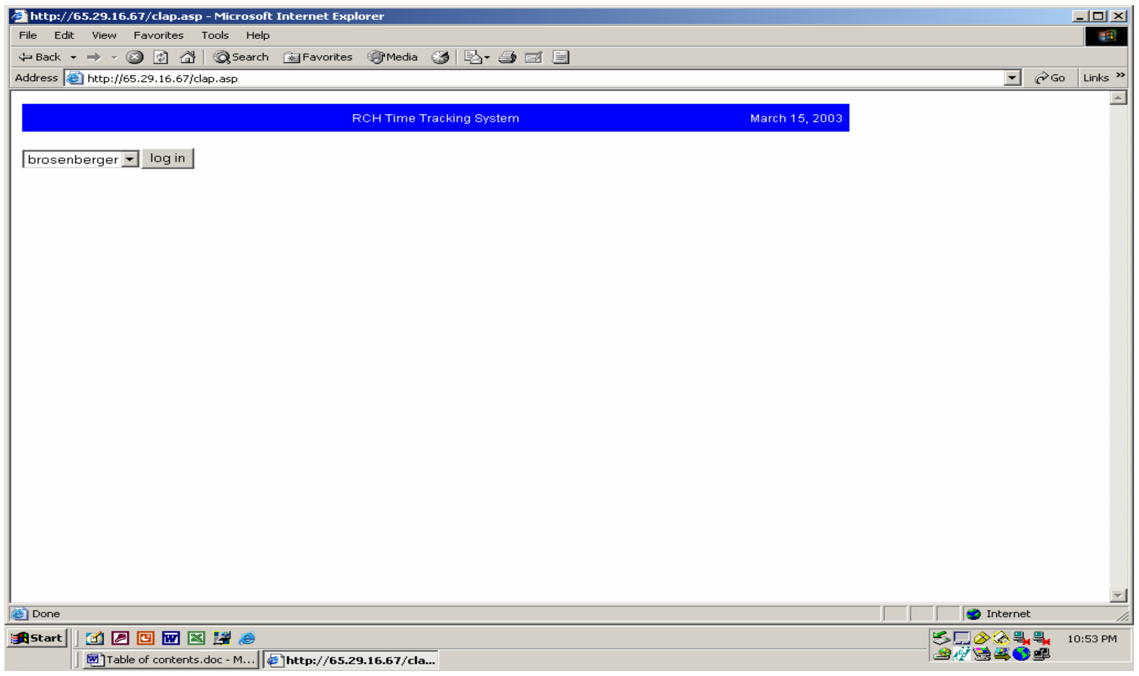
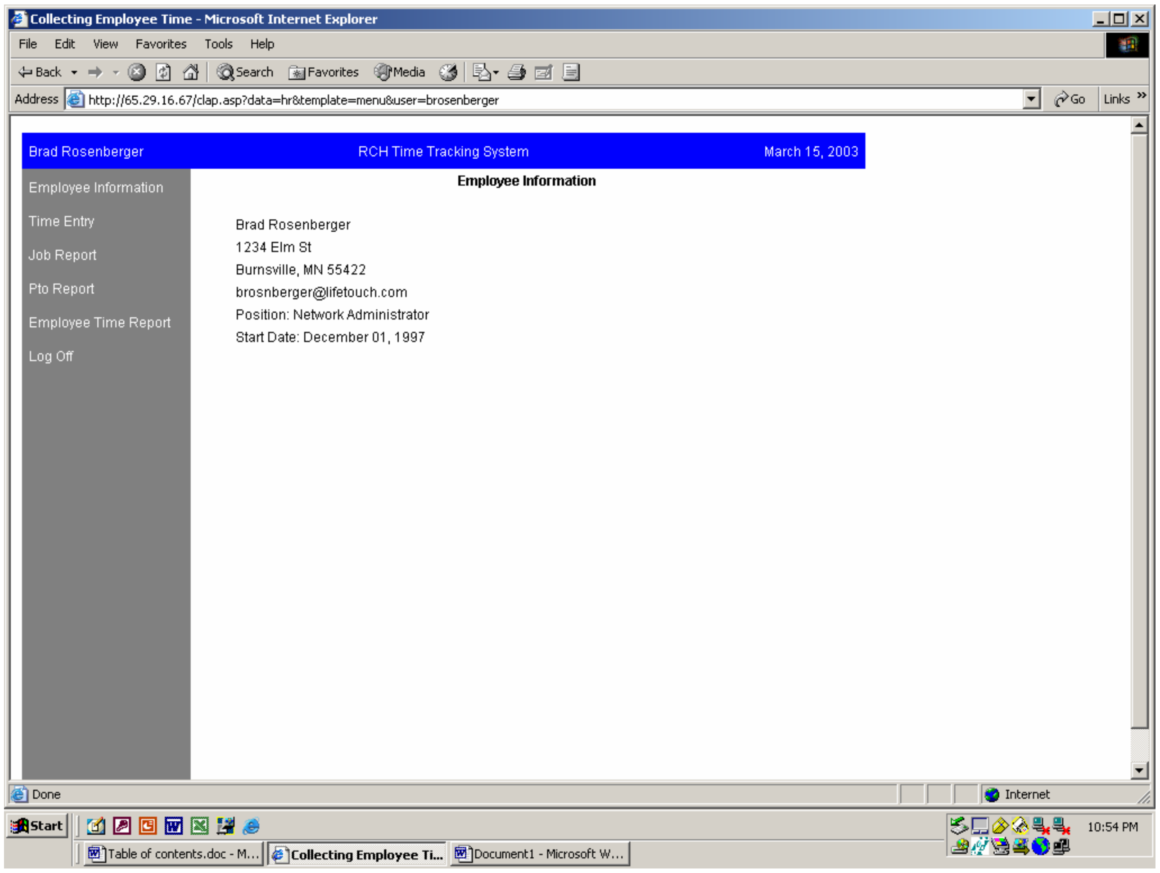

http://65.29.16.67/clap.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://65.29.16.67/clap.asp
Go Links
RCH Time Tracking System March 15, 2003
brosenberger
log in
Done
Internet
Start
Table of contents.doc - M...
http://65.29.16.67/cla...
10:53 PM

Figure 4: Requesting a Page


Collecting Employee Time - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://65.29.16.67/clap.asp?data=hr&template=menu&user=brosenberger
Go Links

Brad Rosenberger RCH Time Tracking System March 15, 2003

Employee Information
Time Entry
Job Report
Pto Report
Employee Time Report
Log Off

Employee Information

Brad Rosenberger
1234 Elm St
Burnsville, MN 55422
brosnberger@lifetouch.com
Position: Network Administrator
Start Date: December 01, 1997

Done
Internet
Start
Table of contents.doc - M...
Collecting Employee Ti...
Document1 - Microsoft W...
10:54 PM

Figure 5: Server Response (HTML format)