XML and Relational Data Integration: Best Practices and the Database Course Experiences

David Olsen, Utah State University, 3515 Old Main Hill, Logan, Utah 84322, USA; E-mail: david.olsen@usu.edu
Vance Cooney; E-mail: vcooney@mail.ewu.edu

ABSTRACT

Many database courses focus on fundamental aspects of relational design, data modeling, transaction processing, and backend database issues. Given the ever increasing importance of web enabled databases generally but particularly the influence of XML (eXtensible Markup Language) as a database enabling technology, the authors felt that an XML module should be integrated into both introductory and advanced database offerings. The focus of this paper is discuss issues related to the integration of XML and relational databases in an undergraduate IT curriculum.

1. INTRODUCTION

Advanced database courses traditionally cover the theory and concepts behind the design and implementation of relational databases. In recent years there has been mounting pressure to weave into this discussion content on integration with the Web as much e-commerce is facilitated by web-enabled databases. The purpose of this paper is to show how the authors enriched an advanced database class by adding an XML module that extended a typical a real world database project example with an integrated XML component. To do this we developed a database using Microsoft's SQL Server 2000, referenced several online tutorials, and created assignments to "pull" data from this database using the XML technology [1].

The remarkable history of the Internet and WWW, well-documented elsewhere, is marked in the mid 1990s by the increasing realization that for the web to facilitate e-commerce in a meaningful way, that production databases and web servers would have to be integrated so that ordering et. al. would be from real time inventories. With the explosion in web sites and web services designed to support this functionality, the limitations in HTML, the web's first language, became obvious [2]. Specifically, many businesses needed to pass data from dissimilar information systems (typically relational databases) via the web and HTML based web pages were not well suited to this requirement. The reason is that such transfers are greatly facilitated by a medium that describes the structure of data (so as to assist the receiving end in parsing / processing the data stream), a requirement that vanilla HTML cannot meet. This is where XML comes in.

Prior to the development of XML there were efforts to adapt the precursor of HTML, SGML (Standard Generalized Markup Language) to accommodate the requirements of web – enabled database applications. SGML was a descendant of IBM's Generalized Markup Language (GML), originally developed in the 1960s to enable the sharing of machine-readable documents in large projects in government. It had also been used extensively in the printing and publishing industries, but its complexity had prevented its widespread application for small-scale general-purpose use [cite = http://en.wikipedia.org/wiki/SGML]. So, when efforts to adapt SGML itself proved fruitless, Jon Bosak, Tim Bray, C. M. Sperberg-McQueen, and Jean Paoli of Microsoft, designed a simplified version of XML based on SGML that has since evolved into the standard XML that we have today [3]. Mind you, we say standard with some trepidation as there continue to be turf wars over how XML ought to be implemented and what valid extensions are, what the best support tool are and etc., but as is typical with important technologies, XML has taken on a life if it's own and isn't waiting for all the warring sides to completely define it but is being and has been usefully deployed in any number of organizational settings over the last dozen years; we try to acquaint our students with this reality as well.

So, XML today is not merely an extension of HTML, it is a meta-language that can be used to define a language particular to a business domain and allow the exchange of data using this defined language. For example, instead of just, say, the <H1></H1> tag pair that traditional HTML would offer that only has to do with how the text within tags is displayed, an XML tag has semantic value. An XML tag example might be <Whsle_Unit_Cost></Whsle_Unit_Cost> and such a tag would define the meaning of the data as well as the display format; i.e., data between the pairs would mean the wholesale unit cost of something, say a prescription drug, and an organization receiving an XML file so designed could easily import the data into their database systems. In short, XML allows a user to separate the presentation of data from its storage, meaning and management. This allows users a vast array of opportunities for using XML to exchange data with trading partners. This phenomenon alone justifies inclusion of XML technology in an advanced database class, in the authors' opinions. The XML enabled Medical Informatics Networks, Automotive parts networks and XML enabled ERPs (Enterprise Resource Planning Systems) that are evolving worldwide all give support to the penetration of XML within the database development community. [Both authors schools have adopted Banner, an Oracle based, XML enabled ERP for the academic environment].

Since its inception, XML has been adopted by many developers as a way to describe data sets and their contents and to define how the data should be output or displayed on a web page (or, significantly, a cell phone, PDA or any of a number of other human readable or machine readable devices). Before XML, a client application accessing a database needed to translate the result set returned to a format that could be understood and displayed by a web application. XML removed the need for client side processing (given a XML compliant client) as the data and its formatting were defined in the XML markup.

The importance of XML is further supported by the fact that Microsoft SQL Server 2005 supports XML, and the result sets can be returned directly in XML format, or data can be retrieved from an XML document as if it were a SQL Server table. Oracle offers a similar suite of functions. A list of XML terms and definitions provided to students in the described database module are included in Appendix A. These terms are a good reference for anyone wishing to begin the study of XML.

2. BRIEF LITERATURE REVIEW

After Codd's seminal 1970 paper on relational theory, hundreds of books and articles on relational databases have been published [5]. Knowledge of relational database design and normalization are staples of advanced database courses, and are key components of the IS2002 model curriculum and guidelines for undergraduate degree programs in information systems curriculum [6]. Likewise, many computer science departments rely strongly on the Computing Curriculum 2001 (or the 2005 version) [7]. We believe that while model curricula such as IS2002 are extremely valuable guidelines, they are by their nature conservative, and likely to be slow in responding to emerging technologies such as XML. However, we believe that XML is going to be integral to future web enabled database architectures and so

believe that universities need to instruct students in XML. As a first pass at such development we have developed an XML Module that can be integrated into an advanced database management course. This will help students prepare to use modern development tools using XML such as Microsoft's .NET framework, and Sun Microsystems's J2EE platform. The need to teach XML in database courses is further demonstrated by the inclusion of XML column types, XML views on relational data, relational views on XML data, XML Schema, and XML based query languages [8] in Microsoft SQL Server 2000 and Oracle 10g. What follows is a description of the teaching module we developed for advanced undergraduates and graduate students in an advanced database management course.

3. XML EXPERIENCES AND BEST PRACTICES

We have reported on the Moab Medical Clinic XML case in [7] as a method for integrating relational and semi-structured data for instructional purposes. We also have collected many semesters of pre and post test data regarding different methods of integrating relational and XML data. In the following section, we present best practices regarding both using and teaching XML and relational database theory based upon the prevailing literature and our own conclusions from the data we have collected.

During the time we have been using and teaching XML in various courses, the following is a list of some XML practices we have observed / evolved:

1. Storing data as an XML data type in an XML column facilitates the DBMS engine checking that the data is well-formed or valid according to the specified XML schema.
2. If large amounts of XML data are input and output on a regular basis, it is better to store the data as an XML data type as it is far more scalable.
3. Storing data as an XML data type in an XML column allows for indexing data for efficient query processing, scalability, query optimization.
4. We encourage students to use the XML output option that creates all XML in the form of attributes as opposed to attributes and entities or raw XML output. There is little advantage to using entities or raw XML output if the desire to integrate with relational data is desired.
5. If small amounts of XML data are inserted or if XML data is infrequently inserted, it is better to use a conversion function and a mapping tool and simply map and transform the XML data into relational tables. Additionally, relational data can be transformed into XML data and used for Web Services. Transforming data back and forth between relational and XML formats can be messy, time consuming, unwieldy, and not scalable.
6. Several third party mapping and transformation tools can be immensely useful. Our students did very well with the Altova XML Spy tool that has numerous features related to database mapping. Altova has a free, limited version that students find valuable for completing their assignments.
7. Students that had significant experience with relational database theory or experience with tag-based languages such as HTML, PHP, Cold Fusion etc. did significantly better than students than students without such experience.
8. Using XML for interoperability between heterogeneous databases looks very promising. Indeed, with numerous disciplines (like accounting creating the XBRL standards) creating XML standards for data formatting, interoperability and common methods of exchange seems inevitable.

4. CONCLUSION

We have previously presented a comprehensive teaching case that demonstrates the integration of a XML module into an advanced database course [7]. Here we discuss how XML is becoming an increasingly important technology, yet few schools are teaching students how to retrieve, format and display XML data. Based on a sound relational database built in earlier assignments, students using our modules proceed through a number of tutorials that develop their familiarity with key components of XML technology. We believe that this method offers a couple of advantages. First, students are reminded of the importance of good relational design principles as they build the MMC database into SQL Server 2005. Second, students learn XML technology in the context of an advanced database class. Third, this instructional approach lays an effective foundation for later courses in web services development or web application programming, without losing the course's focus on advanced database management.

With literally hundreds of competing software packages, languages, operating systems, and networking technologies, we believe students need to integrate and apply new learning within the context of previous skills. XML then becomes a part of student's fundamental understanding of data retrieval and formatting, rather than just one more language or tool.

5. REFERENCES

[1]Marshall, B. 2004 "XML Module." Available from http://olsen.usu.edu/Olsen/XMLModule/index.htm [accessed 1-05].

[2]Wagner, P.J., Moore, T.K. 2003 "Integrating XML into a Database Systems Course." Paper presented at 34th SIGCSE technical symposium on computer science education, Reno.

[3]Sperberg-McQueen, C.M. & Bray, T. 1997. "Extensible Markup Language." Available from http://www.cs.queensu.ca/achallc97/papers/p050.html [accessed 1-05].

[4]Codd, E.F. 1970. "A Relational Model of Data for Large Shared Data Banks." Comm. ACM 13:6.

[5]Gorgone, J.T., Davis, G.B., Valacich, J.S., Topi, H., Fenistein, D.L. & Longnecker, H. E. 2002. "IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems." Available from http://www.acm.org/education/is2002.pdf [accessed 1-05].

[6]Computing Curriculum 2001. Available from http://www.computer.org/education/cc2001/ [accessed 1-05].

[7]Olsen, D., Cooney, V., Marshall, B., and Richard Swart, (Fall 2005), "Towards Full Integration of XML and Advanced Database Concepts," Review of Business Information Systems, vol. 9 no. 4.