XML and Databases for E-Learning Applications

H.F. El-Sofany[1], S.A. El-Seoud[2], F.F.M. Ghaleb[3], S.S. Daoud[3], J.M. ALJa'am[1] and A.M. Hasnah[1]

[1]Department of Computer Science and Engineering, College of Engineering, Qatar University, Qatar
[2]Department of Computer Science, Princess Sumaya University for Technology, Jordon
[3]Department of Mathematics, Faculty of Science, Ain Shams University, Egypt

***Abstract*—XML has become a standard format in information exchange and integration on the Web. Much research has been conducted in recent years on XML technology, which has led to new developments in this field. The effective combination of XML and relational databases for portable Internet data exchange and data management is the technology platform of choice for the Internet Applications. Our paper describes the relationship between XML and relational databases and its potential as an enabling technology to support e-learning. In addition, the paper discusses how relational databases and XML fit together and briefly illustrates their implication in the development of e-learning systems. We introduce our e-learning model, as an example of this new developing methodology.**

***Index Terms*—E-Learning, Relational Databases, XML, XML Databases.**

1. INTRODUCTION

An overall study of the *XML enabled database* systems, such as SQL Server, Oracle and DB2 leads to the fact that, they provide mechanisms to store and query XML data by extending the existing data model [1]. This extension is created mainly by adding *XML data type*, so that a column of this data type can be defined and used to store XML data. In addition, a set of methods is associated with this new XML data type to process, manipulate and query stored XML data. This usually requires various mappings (*e.g.*, *schema mapping*, *data mapping* and *query mapping*) to be performed between the two data models [2,3]. Therefore, the main issue is to develop efficient algorithms to perform these mappings.

One of the important features of XML documents is that we can perform operations based on their logical structures [4]. Based on this feature, databases that manage XML documents have to support queries on their logical structures and on their contents. Considering access based on a logical unit and reusability, it is appropriate to decompose and store XML documents according to their tree structures. They are then stored in appropriate databases. In order to retrieve XML documents from such databases, we have used a dynamic application based on a W3C's Document Object Model (DOM) and database queries (typically in SQL) [5].

The potent combination of XML and databases for portable Internet data exchange and data management is the technology platform of choice for the Internet Applications. The main reason for this is that, XML provides a compelling environment for the integration of disparate forms of data in a platform independent manner. This paper discusses how relational databases and XML fit together and briefly illustrates their implication in the development of e-learning systems.

The paper is organized as follows: in section two we give a brief overview about the XML database systems. In section three we introduce our e-learning model that uses XML and relational databases. In section four, we present a comparison between four virtual learning environment systems. Section five concludes the paper.

2. XML DATABASE SYSTEMS

There are number of options to manage XML data and documents. One way is to use a traditional database system such as relational databases or object-relational databases; the other way is to use a database system designed specially for handling XML documents. In this section, we give a brief overview about the XML databases types, and we present some approaches used to store XML documents in relational databases.

2.1 Types of XML Databases

There are different views on what an XML database is. A good definition is the following: "*a database that stores XML in its native format*".

In general we have two types of XML databases:

- If the XML document is not stored internally as XML data, then it is called an *"XML-enabled database"*. In such databases, XML data is stored as data records into relational tables.
- If an XML document is stored as XML internally (*i.e.*, by the same tree structure of XML document) then we call it a *"native XML database"*.

2.2 XML-enabled databases

We can use the existing database systems to manage XML data. Relational databases are among the first that wanted to represent their data as XML. The storage of XML documents in relational databases means describing hierarchical, tree-type structures with relations. Therefore the first effort was directed towards enabling or extending the capabilities of these databases to support XML. This effort gave birth to the so-called *XML-enabled database*

system. The idea behind such databases is to extend the existing databases in such a way that it would allow efficient storage, querying and publishing of XML in relational databases.
These databases typically accept XML document, and insert it into database tables according to specified database schema. To retrieve the XML document, the data stored in a XML-enabled database are collected back together again. In XML-enabled databases, the XML schema (*e.g.*, DTD or XML Schema) is mapped to the database schema.

2.2.1 Storing XML Documents in a Relational Database

A database schema describes the structure of a database. It gives an abstract view over physical storage mechanisms used by a database. In relational databases, the data model organizes data in table structures. The XML data model, on the other hand, organizes data in highly nested, hierarchical structure. Due to the extreme differences between the two structures, different approaches for converting XML structure to relational structure have been developed. This sub-section provides a brief overview of some approaches used to store XML documents in a relational database.

2.2.1.1 The Edge Approach

Florescu and Kossmann proposed in [6] alternative ways to store XML data in a relational database, one of them is the called *edge approach*. This approach used a *directed labeled graph* data model.

Example 1: Assume that, we want to export information (*e.g.*, `names`, `addresses` and `hobbies`) of the members of a family. A possible way of structuring this information and representing it in XML is as follows:

```
<person><id='1', age='55'>
        <name>Peter</name>
        <address>4711 Fruitdale Ave.</address>
        <child>
          <person><id='3', age='22'>
          <name>John</name>
          <address>5361 Columbia Ave.</address>
          <hobby>swimming</hobby>
          <hobby>cycling</hobby>
        </person>
        </child>
        <child>
          <person><id='4', age='7'>
          <name>David</name>
          <address>4711 Fruitdale Ave.</address>
          </person>
        </child>
        </person>
<person><id='2', age='38', child='4'>
        <name>Mary</name>
        <address>4711 Fruitdale Ave.</address>
        <hobby>painting</hobby>
</person>
```

- the corresponding data graph of the above XML document, is shown in figure 1.

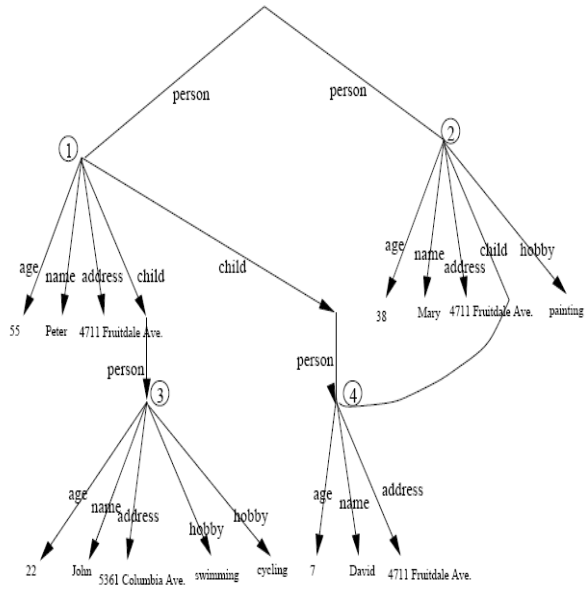

Figure 1. Data graph corresponding to the XML document.

- **In mapping attributes:** all the attributes are stored in a single *Edge* table. The table has the following structure: `Edge (source, ordinal, name, flag, target)`
  - `source` column is used to store the *object identifiers* (oids).
  - `ordinal` column is used to recover all attributes of an object in the right order and to carry out updates if objects have several attributes with the same name.
  - `name` column is used to store the attribute name.
  - `flag` column is used to indicate in which *Value* table a value is stored, a flag can therefore takes values such as *string*, *integer*, *data* or *ref* .
  - `target` column is used to store the oids of the target object of the attribute.
  - The *key* of the *Edge* table is {`source`, `ordinal`}.

- **In mapping values:** two different schemes have been proposed:
  - Storing values in a separate *value* table: V_{type}(`vid`, `value`) and
  - Storing values together with the *attribute*.

- The following tables represent the `Edge`, and V_{type} instance, corresponding to the XML document in Example 1.

Edge

source	*ordinal*	*name*	*flag*	*target*
1	1	age	int	v1
1	2	name	string	v2
1	3	address	string	v3
1	4	child	ref	3
1	5	child	ref	4
2	1	age	int	v4
…	…	…	…	

V_{int}

vid	*value*
v1	55
v4	38
v8	22
v13	7

V_{string}

vid	*value*
v2	Peter
v3	4711 Fruitdale Ave
v5	Mary
v6	4711 Fruitdale Ave.
v7	Painting
…	…
v15	4711 Fruitdale Ave.

2.2.1.2 The Basic, Shared, and Hybrid inlining Approaches

Shanmugasundaram *et al.* presented in [7] an approach to map XML to relational database schema. This mapping scheme was developed to investigate the use of relational database systems to process queries on semi-structured documents. In this approach, the XML documents must have the associated DTDs to map them to a database schema. The database schema is created from the DTD, which is done by building a so-called *DTD graph* and constructing an *element graph* from it.

Example 2: The DTD graph corresponding to the DTD in Figure 2 is given in Figure 3. Cycles in the DTD graph indicate the presence of recursion.

```
<!ELEMENT book (booktitle, author)>
<!ELEMENT article(title, author*, contactauthor)>
<!ELEMENT contactauthor EMPTY>
<!ATTLIST contactauthor authorID IDREF IMPLIED>
<!ELEMENT monograph(title, author, editor)>
<!ELEMENT editor(monograph*)>
<!ATTLIST editor name CDATA#REQUIRED>
<!ELEMENT author(name, address)>
<!ATTLIST author id ID#REQUIRED>
<!ELEMENT name(firstname?, lastname)>
<!ELEMENT firstname(#PCDATA)>
<!ELEMENT lastname(#PCDATA)>
<!ELEMENT address ANY>
```

Figure 2. A sample DTD.

- When the database schema is created from the XML DTD, several *inlining techniques* that destroy nested elements can be used to avoid the fragmentation of XML documents over several tables. The inlining technique keeps as many elements as possible into a single relation.
- Shanmugasundaram proposed in [7] three inlining techniques which can be summarized as follows:
  1. The **basic inlining technique** solves the fragmentation problem, by inlining as many descendants of an element as possible into a single relation. But this technique creates relations for every element because an XML document can be rooted at any element in a DTD.

     ***Example*:** the `author` element in Figure 3 would be mapped to a relation with attributes `firstname`, `lastname`, and `address`. In addition, three relations would be created corresponding to these elements.

  2. The **shared inlining technique** is an improvement over basic inlining technique because it makes sure that an element node is represented exactly in one relation. It identifies the element nodes that are represented in multiple relations in basic inlining technique (*e.g.*, *firstname*, *lastname*, and *address*), and creates a separate relations for them such that those elements can be shared.

  3. The **hybrid inlining technique** is same as the shared inlining except that it inlines some elements that are not inlined in shared inlining technique.

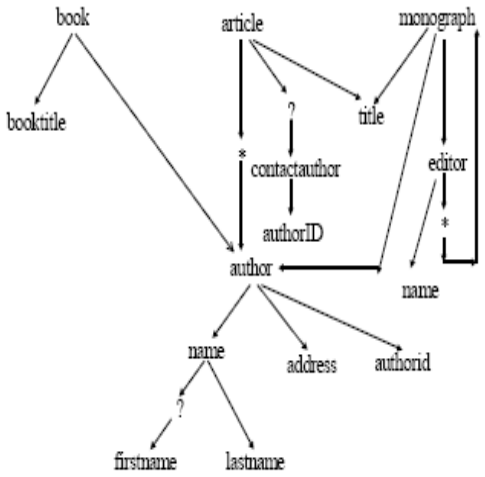

Figure 3. A DTD graph for the DTD in Figure 2.

2.2.1.3 The "New Inlining" Approach

S. Lu., *et al.* introduced in [8] an efficient algorithm which takes an XML DTD as input and produces a relational schema as output for storing and querying XML documents conforming to the input DTD. The algorithm features several significant improvements over the shared-inlining technique explained in the previous section, including eliminating redundancies caused by shared elements, performing optimizations and enhancing efficiency.

- The new inlining algorithm contains the following three steps:
  1. **Simplifying DTDs:** since a DTD expression might be very complex due to its hierarchical nesting capability, this step used to simplify the input DTD.
  2. **Creating and inlining DTD graphs:** create the corresponding DTD graph based on the simplified DTD, and then inline as many descendant elements as possible to an XML element. In contrast to the shared-inlining technique, the new inlining rules eliminate the redundancy caused by shared elements in the generated relational schema and can deal with arbitrary input DTDs including those that contain arbitrary cycles.
  3. **Generating relational schemas:** generate a relational schema based on the inlined DTD graph in step 2.

Example 3: Consider the following input DTD for publications:

```
<!DOCTYPE publication [
<!ELEMENT publication (journal*, conference*)>
<!ELEMENT journal (name, editors, paper+)>
<!ELEMENT conference (name, paper+)>
<!ELEMENT paper (ptitle, authors,(volume, number)?)>
<!ATTLIST paper year CDATA>
<!ELEMENT editors (person+)>
<!ELEMENT authors (person+)>
<!ELEMENT person (pname, institute, techreport*)>
<!ELEMENT techreport (title, references)>
<!ELEMENT references (paper+)>
<!ELEMENT institute (#PCDATA)>
<!ELEMENT pname (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

```
 <!ELEMENT ptitle (#PCDATA)>
 <!ELEMENT title (#PCDATA)>
 <!ELEMENT volume (#PCDATA)>
 <!ELEMENT number (#PCDATA)>
]>
```

After the simplification step, the input DTD is simplified into one with the following new XML element definitions. The definitions for other XML elements remain the same.

```
<!ELEMENT journal (name, editors, paper*)>.
<!ELEMENT conference (name, paper*)>.
<!ELEMENT paper (ptitle, authors, volume, number)>.
<!ELEMENT editors (person*)>.
<!ELEMENT authors (person*)>.
<!ELEMENT references (paper*)>.
```

Finally, the following eight relations will be generated after performing the four steps of generating relational schemas:

```
publication(ID)
conference(ID, name.ID)
journal(ID,nodetype,name.ID
name(ID, PCDATA)
paper(ID, nodetype, ptitle, volume, number,
year)
person(ID, nodetype, pname, institute)
techreport(ID, nodetype, title
edge(parentID,childID, parentType, childType)
```

2.2.1.4 DOM-Based Approach

We have introduced in [5] a novel approach for storage and retrieval of XML documents using relational databases. In this approach, an XML document is decomposed into nodes based on its tree structure, and stored into relational tables according to the nodes types. Our approach enables us to store XML documents using a fixed relational schema without any information about XML schema and DTD. For the processing of XML documents, we proposed two algorithms denoted by "*XtoR*" and "*RtoX*", where the first one is for converting XML data to relational data, while the second one consists of extracting data from a database and insert them into a XML document. The application does not impose any extension of relational databases for storage and retrieval of XML documents. The data model we used for the data mapping algorithms is based on the W3C's DOM given in [9].

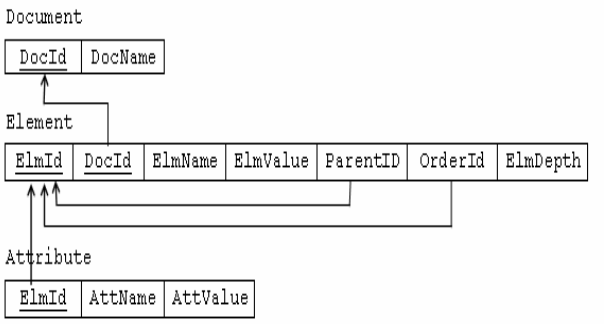

Figure 4. The proposed schema for mapping the XML document to a relational table.

- **Proposed relational schema:** Note that, it is needed to store XML document efficiently, to preserve the order of the document. Hence, we have mapped the XML DOM model onto the relational database schema shown in Figure 4.
  - The core of the mapping is the **`Element`** entity, that contains the following attributes:
    - `ElmId`: A unique identifier for each element,
    - `DocId`: The unique key of the associated document,
    - `ElmName`: The name of this element,
    - `EmlValue`: The text or data of this element (`NULL` for none),
    - `ParentID`: The element which is parent to this element (`NULL` for the root element),
    - `OrderId`: Element id of the next element in scope order (`NULL` for last element), and
    - `ElmDepth`: The scope depth of this element.
  - The `Element` entity is supplemented with two other entities. The first one is the **`Document`** entity, that contains the following attributes:
    - `DocId`: A unique identifier for each document, and
    - `DocName`: The name of this document.

  The second entity is the **`Attribute`** entity, which contains information about element's attribute:
    - `ElmId`: The unique key of the associated element,
    - `AttName`: The name of this attribute, and
    - `AttValue`: The value of this attribute.

Example 4: Consider the following XML document:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<books>
   <book>
      <author> H.M. Deitel and P.J.Deitel </author>
      <title>  C++ How To Program </title>
      <publisher> Prentice Hall Publishing Co. </publisher>
   </book>
   <book>
      <author>Jack Herrington</author>
      <title>PHP Hacks</title>
      <publisher>O'Reilly</publisher>
   </book>
</books>
```

Figure 5. An XML instance of (`books.xml`) document.

- Figure 6 shows a database instance of the *XtoR schema* that explains how the XML document given in Figure 5 is mapped into the relational database schema given in Figure 4, by using the *XtoR* algorithm introduced in [5].

Document:

DocId	DocName
1	books.xml

Element:

ElmId	DocId	ElmName	EmlValue	ParentID	OrderId	ElmDepth
1	1	xml	Null	Null	2	0
2	1	books	Null	1	3	0
3	1	book	Null	2	7	0
4	1	author	H.M. Deitel and P.J.Deitel	3	5	1
5	1	title	C++ How To Program	3	6	1
6	1	publisher	Prentice Hall Publishing Co.	3	Null	1
7	1	book	Null	2	Null	0
8	1	author	Jack Herrington	7	9	1
9	1	title	PHP Hacks	7	10	1
10	1	publisher	O'Reill	7	Null	1

Attribute:

ElmId	AttName	AttValue
1	version	"1.0"

Figure 6. A database instance of the *XtoR schema* storing the XML document in Figure 5.

2.3 Native XML Databases

The need to process and store XML data generates several new types of software tool, one of which is the *"native XML database"*. This section explains the principles behind the native XML database.

2.3.1 What is a Native XML Database?

The term "*Native XML Database*" (NXD) is understandable in many ways. In fact many so-called NXDs aren't really standalone databases at all, and don't really store the XML in true native form (*i.e. text*). To get a better idea of what a NXD really is, let's take a look at the NXD definition offered by the XML:DB initiative [10].

- A *native XML database* has the following features:
  1. Defines a logical model for an XML document. Stores and retrieves documents according to that model. At a minimum, the model must include *elements*, *attributes*, PCDATA, and document order.
     Examples of such models are:
     - The XPath data model,
     - The models implied by the DOM, that proposed by W3C, and
     - The SAX model (Simple API for XML)
  2. Has an *XML document* as its fundamental unit of logical storage, just like a relational database has a *row* in a table as its fundamental unit of logical storage.
  3. Is not required to have any particular underlying physical storage model. For example, it can be built on a relational, hierarchical, or object-oriented database; it can also use a proprietary storage format such as indexed, compressed files.

2.3.2 Native XML Databases Systems

A long list of native XML databases systems is given in [11]. Some of these databases are open source or result of research projects; the rest are commercial native XML databases:

- Native XML databases resulting from research projects: such as, Lore from Stanford University and Timber from University of Michigan.
- Open source native XML databases, such as: Oracle *Berkeley DB XML*, *eXist* XML database [9], and *ozone* database project [12].
- Commercial native XML databases: such as Tamino XML Server (from Software AG).

3. THE E-LEARNING MODEL

Our model given in Figure 7, provides the *student* with two kinds of contents, *Learning content* and *Assessment content*. Each content has different types of services such as:

- ***Learning services*:** *registration*, *online course*, *Interactive tutorial*, *course documents* (is a repository for files that the instructor have made available to the student as a part of your course), *announcements* (displays information to the students that the instructors of the course want him to know), *links* (displays a list of useful URL links that have been identified by the course instructors), *student papers* (students can post/upload requests files to the instructor).
- ***Assessment services*:** provide exercises and quizzes for evaluation of the student knowledge.

During the learning process, a dynamic selection presentation of both contents will be accomplished.

On the other hand, our model, allows teachers (i.e., *professors*, *lecturers* ...) to create and admin course websites through a browser (i.e., *Explorer*, *Netscape*...). The following services are also included in our model:

- Publish documents in any format (Word, PDF, HTML, Video...)
- Admin public or private discussion forums.
- Manage a list of links: displays a list of useful URL links that have been identified by the course instructors.
- Create student groups.
- Compose exercises.
- Structure an agenda with tasks and deadlines.
- Make announcements: This feature displays information to the students that the instructors of the course want them to know.
- Submit files: students can post (i.e., upload) files to the instructor.

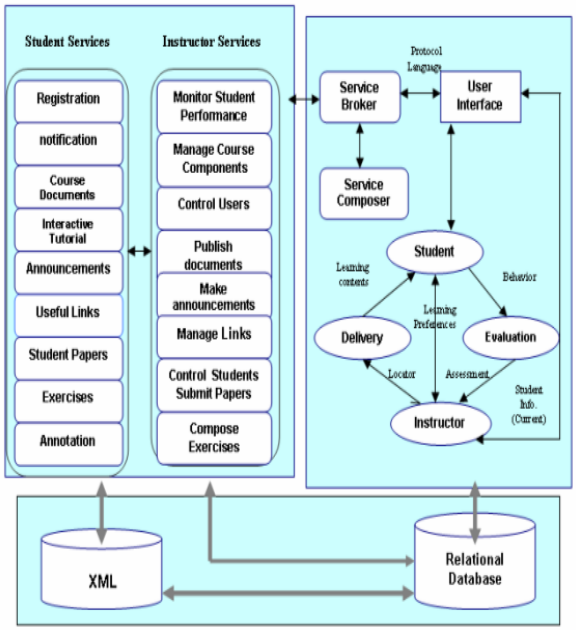

Figure7. Framework for Our Web-Based e-learning system

The e-learning model uses XML, rather than the classical HTML language due to its flexibility and richness to deal dynamically with the questions. In fact, with HTML one must use the established tags, however, with XML we can create specific documents with the tags we need and the features we want. So, XML allows users

to create their own specific language according to their needs. The first step is to create what is called the *Document Type Definition* (DTD) where the features are specified, and include tags and attributes. Once the DTD is created, we can use the new tags for creating web pages. XML was chosen as a basis for system development because of the hierarchical data structure that auto-defines itself. The introduction of XML is performed by means of a DTD, which determines the question structure. The DTD defines the composition of each kind of question considering all the necessary fields [13,14].

4. VIRTUAL LEARNING ENVIRONMENT COMPARISON

Virtual learning environments are software packages designed for helping educators to create and deliver online courses. Such e-learning systems are sometimes called a Learning Management System (LMS), Course Management System (CMS), or Virtual Learning Environments (VLE). The use of e-learning in the majority of universities has begun with the introduction of a VLE and although this project is funded by the European Union.

In this section we give a comparison between four VLE systems: *Moodle* [15], *Claroline* [16], *ATutor* [17] and *Blackboard* [18]. From the proposed comparison, one can decide easily, what is the suitable VLE he may need.

1. **Creation**
   - 1.1 *Claroline*: Consortium of French University faculties.
   - 1.2 *Moodle*: PhD student called Martin Dougiamas.
   - 1.3 *ATutor*: Toronto University in Canada.
   - 1.4 *Blackboard*: Learning Management System, partially owned by Microsoft.

2. **License Agreement**
   - 2.1 *Claroline*, *Moodle*, and *ATutor* are Open source (GPL).
   - 2.2 *Blackboard*: Licensed annually.

3. **Cost**
   - 3.1 *Claroline*, *Moodle*, and *ATutor* are freely downloaded, installed and distributed without charge.
   - 3.2 Blackboard: approximately $8,600 each year.

4. **Common Features**
   - 4.1 Theme feature allows the change of look and feel of the VLE without a new style sheet.
   - 4.2 Agenda feature which allows authors to set weekly announcements in advance.

5. **Common Features between Claroline and Moodle**
   - 5.1 Courses are broken down into component elements and then published to the site under separate areas (announcements, exercise, etc.)
   - 5.2 Courses can start with a limited number of resources, but grow in size and complexity.
   - 5.3 Students have full flexibility in the order that they undertake the elements of the course.
   - 5.4 Categorized links feature to manage relevant URLs.
   - 5.5 Offers fully optional layout for course beneficiaries.
   - 5.6 Ability of the course administrators to set exercise and assign completion deadlines.
   - 5.7 Students can upload their own papers for peer review.

6. **Common Features between Moodle and ATutor**
   - 6.1 Inbuilt glossary function.
   - 6.2 Send course email feature to allow all course students to be contacted simultaneously.
   - 6.3 Flexible assignment creator tool.
   - 6.4 Excellent documentation and help manuals.
   - 6.5 Dynamic site mapping feature that grows as more courses are authored on the system.

7. **Blackboard vs. Moodle**
   - 7.1 Moodle advantages over Blackboard:
     - Providing individualized feedback easily to all assignments
     - Easier to track each student's activity in class
   - 7.2 Blackboard advantages over Moodle:
     - More polished appearance.
     - Better grade book.
     - Threaded discussions easily differentiate between read and unread posts.
     - Announcements are more prominently displayed upon entering the course.

8. **Different Features between the four virtual learning environments**
   - 8.1 *Claroline*:
     - Chat facility for all users (text interface system).
     - Can upload video files for use as course recourses.
     - Statistics function for course administrators to monitor number of courses, courses popularity, etc.
     - Supports multi-languages.
   - 8.2 *Moodle*:
     - Basic security features to limit customer access to particular courses.
     - Journal feature to allow students to post questions, maintain a course diary, or aid revision.
     - Use XML metadata to describe e-learning content within the system.
   - 8.3 *ATutor*:
     - Course construction broken down into a series of 'content pages'.
     - The structure is built on a 'slide by basis' but would require HTML skills.
     - Integrates editor for creating content without the need for HTML knowledge.
     - Print compiler tool-allows student to select pages of text and group them together on one

page.
- Context sensitive help.
- Links to various other educational databases are included with the download package.

8.4 *Blackboard*:
- A Blackboard course consists of a navigation path, a button bar and content frames.
- Faculty type or upload their course materials into Blackboard. Blackboard can accommodate text, graphics and audio mediums.
- The course material does not have to be html documents. Faculty can load Word documents or even PowerPoint presentations.
- The navigation path allows users to return to any page accessed between the main course page and the current pages.
- The button bar links users to the available content areas and tools. The content frame displays web pages accessed through the button or navigation path.
- Typical assessment methods used in determining learning outcomes include tests made up of short answer, multiple choice, True/False or combinations of all three.

5. CONCLUSION

Our paper described the relationship between XML and relational databases and its potential as an enabling technology to support e-learning. Also the paper discussed how relational databases and XML fit together and briefly illustrated their implication in the development of e-learning systems. We introduced our e-learning model, and the comparison between four VLE systems, as an example of this new developing methodology.

REFERENCES

[1] IBM. DB2 extender for XML. http://www-4.ibm.com/software/data/ db2/extenders/xmlext.html.
[2] M. F. Fernandez, Y. Kadiyska, D. Suciu, A. Morishima, and W. C. Tan. Silkroute: A framework for publishing relational data in XML. TODS, 27(4):438–493, 2002.
[3] S. Lu, Y. Sun, M. Atay, and F. Fotouhi. A new inlining algorithm for mapping XML DTDs to relational schemas. In Proc. of the First Int. Workshop on XML Schema and Data Management, in conj. with the 22nd ACM Int. Conference on Conceptual Modeling (ER2003), Illinois, USA, Oct 2003.
[4] I. Tatarinov, S. Viglas, K. S. Beyer, J. Shanmugasundaram, E. J. Shekita, and C. Zhang. Storing and querying ordered XML using a relational database system. In SIGMOD Conference, pages 204–215, 2002.
[5] Sameh S. Daoud, Fayed F. M. Ghaleb, Hosam F. El-Sofany, Ahmad M. Hasna, Jihad M. AL Jaam, and Samir A. El-Seoud, "A DOM-Based Approach of Storage and Retrieval of XML Documents Using Relational Databases". Proc. 9th International Conference on Interactive Computer Aided Learning - ICL 2006. September 27 – 29, Villach, Austria, 2006.
[6] Daniela Florescu and Donald Kossmann. A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in Relational Database. Technical report, INRIA, France, 1999.
[7] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J. Naughton. Relational Databases for Querying XML documents: Limitations and Opportunities. In Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, pages 302–314, 1999.
[8] S. Lu, Y. Sun, M. Atay, and F. Fotouhi. A new inlining algorithm for mapping XML DTDs to relational schemas. In Proc. of the First Int. Workshop on XML Schema and Data Management, in conj. with the 22nd ACM Int. Conference on Conceptual Modeling (ER2003), Illinois, USA, Oct 2003.
[9] W3C site: http://www.w3c.org. (http://www.w3.org/TR/ 2000/ REC-DOM-Level-2-Core-20001113/)
[10] XML:DB Initiative. What is an XML database?, Accessed 2003. http://www.xmldb.org/faqs.html.
[11] Ronald Bourret. XML Databases Products: Native XML Databases. http://www.rpbourret.com/xml/ProdsNative.htm, Accessed August 2003.
[12] http://ozone-db.org/frames/home/what.html, http://www.xmldb.org/ xupdate/
[13] HOSAM F. EL-SOFANY, AHMAD HASNA, JIHAD JAAM, FAYED GHALEB, AND SAMIR A. EL-SEOUD, "A WEB-BASED E-LEARNING SYSTEM EXPERIMENT". INTERNATIONAL JOURNAL OF COMPUTING & INFORMATION SCIENCES (IJCIS), VOL. 4, NO. 1, P22-29, APRIL 2006.
[14] Fayed Ghaleb, Sameh Daoud, Ahmad Hasna, Jihad Jaam, Samir A. El-Seoud, and Hosam El-Sofany, "E-Learning Model Based On Semantic Web Technology". International Journal of Computing & Information Sciences (IJCIS), Vol. 4, No. 2, P63-71, August 2006.
[15] MOODLE: (HTTP://WWW.MOODLE.ORG)
[16] CLAROLINE: (HTTP://WWW.CLAROLINE.NET)
[17] ATUTOR (HTTP://WWW.ATUTOR.CA)
[18] BLACKBOARD: WWW.BLACKBOARD.COM/COMPANY/ACCESSIBILITY.ASPX

AUTHORS

H.F. El-Sofany, Lecturer (helsofany@qu.edu.qa)
A.M. Hasna, Associate Professor (hasnah@qu.edu.qa)
J.M. AL Ja'am, Professor (jaam@qu.edu.qa)
Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha – Qatar.

F. F. M. Ghaleb, Associate Professor
(fmghaleb@yahoo.com)
S.S. Daoud, Associate Professor
Department of Mathematics, Faculty of Science, Ain Shams University, Cairo – Egypt

S. A. El-Seoud, Professor (selseoud@psut.edu.jo)
Computer Science Department, Princess Sumaya University for Technology, Amman – Jordan

Manuscript received 09 May 2007. This work was supported partially by Qatar University and PSUT-Jordan. The authors would like to thank deeply both universities for their support.

Paper presented at ICL2007 conference, Villach, Austria, September 2007.

Published as submitted by the author(s).