The X files – an XML Xperience

Dave Kennedy, Dr Mike Lance
School of Computing
Christchurch Polytechnic Institute of Technology
Christchurch, New Zealand
kennedyd@cpit.ac.nz

ABSTRACT

XML – eXtensible Markup Language. A way to markup a document for content. A standard for data interchange that is being used for B2B transactions. XML is designed for use with data-centric documents. Williams *et al*, 2000 describe a method for mapping a RDBMS structure to an XML DTD. A non-trivial real-world example was selected, that of course outlines. A RDBMS was designed for course outlines and the structure mapped to a DTD. The DTD plus a sample document was initially validated using Internet Explorer. It was further checked using an on-line validator. The DTD was subsequently revised in line with guidelines for good XML.

1. INTRODUCTION

XML - yet another computer industry acronym. XML - eXtensible Markup Language. So what is it and why would you want to use it?

In spite of its name XML is not a markup language but a set of rules for building markup languages (Ray, 2001, pg2; W3C.org). It enables you to build a standard markup for a document, or to use a standard that someone else has defined, to enable data transfer between business systems. It is a way to define a standard for data exchange. It is a way of marking up a document for content. It is a way of marking up a document that can then be used as a feed to a range of different data presentations.

Most computer professionals are now familiar with HTML - a way of marking a document for display. XML is a way of marking a document for content. There are many confusing articles about XML that talk about XML databases, migrating a database to XML or searching an XML database (Williams, 2000; www.oasis-open.org) that can lead you to think that XML is some new type of database. A collection of XML documents could be described as a database but is a very poor way to store and manage data (Mertz, 2001a). XML is mostly a “standard for automating data exchange between business systems” (Banfield, 2001), i.e. it is an improvement - a big improvement - on using a comma-separated file because the data descriptions are contained within the file (document). It is mostly a format for data exchange, a protocol,

which makes automating the data exchange possible. It is seen as an easier format than EDI to implement as a comms format between businesses (Banfield, 2001). Its key benefits are:

- Flexibility - it can be used to describe any collection of data and in several ways. This description (the Data Type Definition, DTD) then imposes a standard structure on all documents using that particular description.
- Simplicity - it is a text format that is easy to send and process. This has a downside: to find data requires it to be scanned line by line.
- Multipurpose - it is a way of presenting the same data to different users (Banfield, 2001; Punin, 2001).

XML is a way of marking up the data content of a document. A related technology XSLT (XML Style Language Transforms) provides a way of presenting the data. An XML document can be presented in many different ways by using different style sheets or different applications (Ray, 2001, chap 4).

Bourrett, 2000 distinguishes between data-centric documents e.g. sales orders, flight schedules and document-centric documents e.g. books, email, adverts. Data-centric documents are for computer consumption, document-centric documents for human consumption. Furthermore the data in data-centric documents needs to be kept in a DBMS and document-centric documents kept in a context management system (Bourrett 2000, Mertz 2001a). Many of the larger examples of XML are document-centric and are based on the Docbook vocabulary (Ray, 2001). XML is becoming accepted as the standard for data exchange with suppliers now providing support for XML (Punin, 2001). In particular XML documents can be validated and viewed using Microsoft Internet Explorer (Lee, 2001) and SQL server 2000 enables SQL results to be converted to XML (Wahlin, 2001).

XML articles generally use trivial examples to illustrate the principles. Williams et al use a simple customer, invoice, invoiceline example to explain their eleven rules for creating a DTD based on the database structure. Ray 2001 uses a simple chequebook example to illustrate XML concepts. XML may well be relatively simple but implementing a real-world application is obviously not all that easy (Banfield, 2001).

The aim was to learn about XML by applying it to a real-world situation. Course outlines were chosen. This paper documents our experience to date as we work towards a database for course outlines that can be extracted as XML documents which in turn can be used by a number of different applications.

The School of Computing (CPIT) offers a range of computer programmes from CIC to BBComp. Each programme contains a number of courses. For any given semester there are approximately 150 different courses on offer. Each has a course outline, based on a standard MS-Word format, that is updated by the course lecturers. Each student receives a course outline at the beginning of the course. However the information contained in course outlines is used in a number of other areas - brochures, web pages, front office information. In particular it requires skilled workers to take existing information and re-format it for on-line delivery. Staff often spend time cutting and pasting information from course outlines to use in other applications. As course outlines change often the information in brochures etc is not updated and so “Official versions” of the same thing get out of sync. It was this aspect of the information cycle that suggested maybe it would be more efficient if course outlines were converted to XML documents. The information could then be easily presented in a variety of ways. The ideal is to write once and then re-publish in different formats automatically.

2. METHODS

The aim was to produce:

a an XML DTD for course outlines
b a well-formed and valid XML course outline document.

The Document Type Definition (DTD) describes the structure and syntax of an associated XML document. It defines the element and attribute names and their data type, their order within the document and whether they are optional or mandatory components. A well-formed XML document follows the syntax rules for XML as defined by the W3C. i.e. the root element must contain all the other elements, each element must nest inside any enclosing elements properly

and each start tag must have a corresponding end tag (Punin, 2001). An XML document should also be valid i.e. it conforms to the pattern described by the associated DTD. This will provide an automatic check on the completeness of course outlines – an otherwise time consuming manual task. It is interesting to note that a DTD can express constraints and business rules in a simpler and richer vocabulary than can be done in a DBMS. For example writing a trigger to enforce it is more complicated than the equivalent DTD. DTDs are being superseded by XML schema but we decided to start with the simplest XML technology. A simple example would be:

Williams *et al.* (2000, chapt 2) describes a method for taking an existing relational database and moving it to XML The approach decided on was to first build a relational database for course outlines then map this structure to a DTD. If we could transform a collection of Word documents to a collection of XML documents using this approach the next step would be to build the style sheets required to present the documents in a number of different ways. The aim was to automate the process of extracting XML from a database as much as possible. The steps involved: Figure 1.

a. analyse a course outline and design a normalised relational database.
b. build the database.
c. extract the data elements from an existing course outline and insert these into the database.
d. map the database structure to an XML DTD.
e. create a sample XML document and validate it.

3. RESULTS AND DISCUSSION

Bourett, 2000 emphasises that XML is about data-centric documents. Analysis of a course outline revealed a large number of entities (Faculty, School, Programme etc) and attributes. A course outline is definitely a data-centric document. The analysis also raised questions as to the purpose of such a database. It was decided that it would be a production database for course outlines, i.e. it would contain only

```
<?xml version =”1.0”?>

<!–  simple course outline DTD    –>

<!DOCTYPE courseoutline [

<!ELEMENT courseoutline (heading) >
<!ELEMENT heading (programmename) >
<!ATTLIST heading
      coursecode  CDATA #REQUIRED>

<!ELEMENT  programmename      (#PCDATA)>
]>

<!–    and an associated XML document    –>

<courseoutline>
<heading coursecode=”BCIT101” >
<programmename>Computer Applications in Business</programmename>
</heading>
</courseoutline>
```

Figure 1

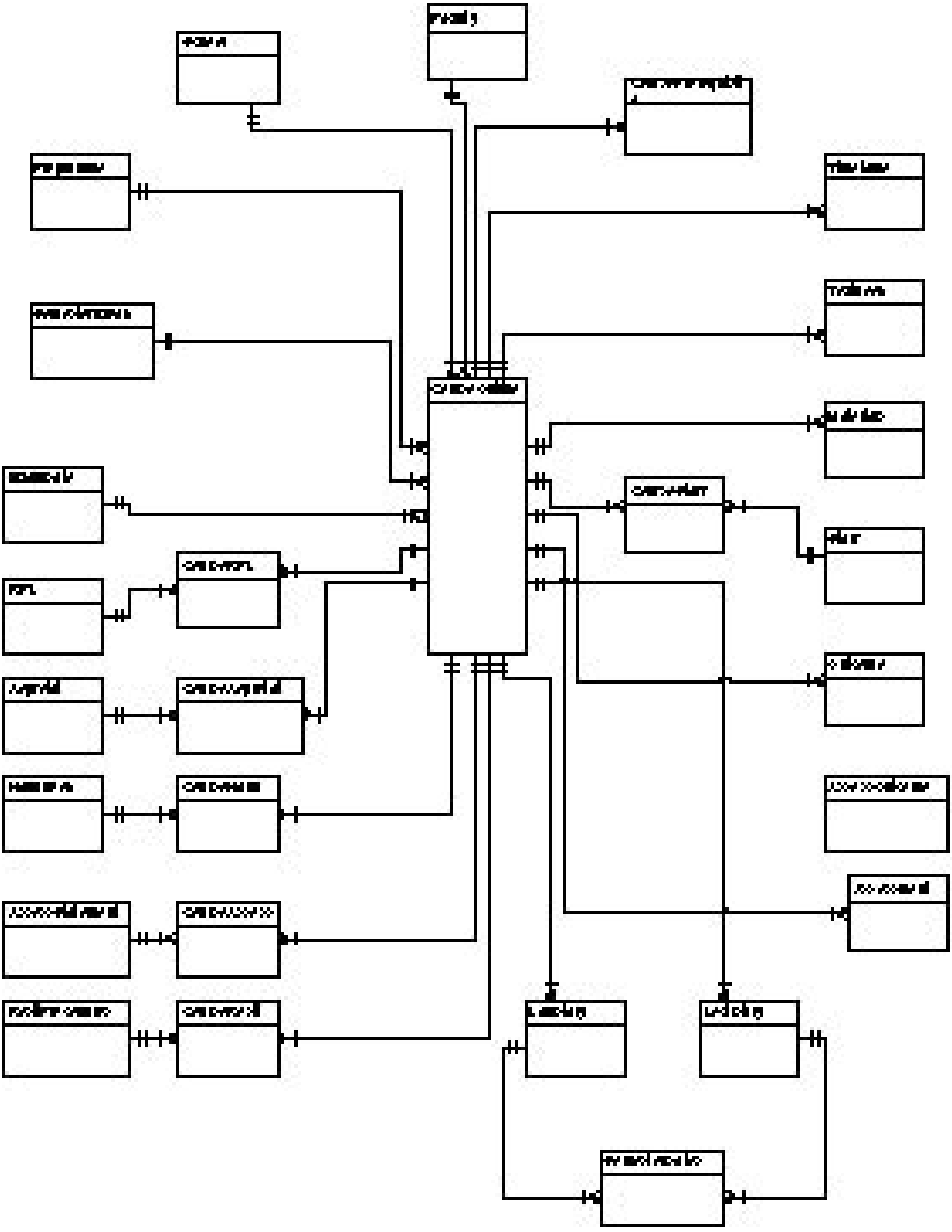

Figure 2

ERD for Course Outlines

outlines for the current semester. The normalised database contained 28 tables (fig 2 ERD for course outlines).

Much of the data (Faculty, School, Programme, Semester dates etc) is the same for all, or a group of, outlines. For a particular outline some data changes very little from one semester to the next and it could be that the semester dates are the only changes. Some of the tables (e.g. Staff, Examdates) are tables that would be maintained by a Programme Leader or Head of School. Particular outlines simply reference these tables. SQL server 2000 can convert the results of an SQL query into XML (Wahlin, 2001). SQL server 7 was available so it was decided to build the database using a script file. Later it would possible to recreate the database in SQL server 2000.

The next step was to insert the data for some course outlines. The outlines currently exist as Word documents. Could a Word document be converted to a sql script file containing the required SQL INSERT statements? Well it could but the first one took some time and manual editing as follows:

a. a script file is a text file. The word document was first saved as text only.
b. the non-data sections were manually deleted.
c. the data elements (table rows) were arranged manually so they were on separate lines.
d. a Find/Replace moved it closer.
   i. Find: paragraph mark
   ii. Replace: “)paragraph markINSERT INTO XXXX VALUES(“
e. Further manual editing was required to produce a working script file.

The transformation process is tantalisingly “mechanical” and considering the structural nature of the Word document, with headers and styles, it deserves to be automated with a macro. The script files would make it easy to create and load the database using SQL server 2000. They also proved useful when the server machine suffered a disk crash.

Creating a script file for a second outline was much less time consuming. Much more of the Word document could be deleted because all the common data was already loaded. At this stage it was realised that it would be easier to create a script file to load the general tables (Faculty, Programme, Staff, RPLStatements etc) and it was only the data specific to each course outline that would need to be extracted from the Word documents. Maybe this process could be automated or semi-automated. That is an area for further research.

If the database was to contain the data for the current semester and XML documents extracted from it then an application would be required to maintain the database. A simple Visual Basic prototype was developed.

A set of SQL SELECT statements would be required to extract all the data for a particular course outline. Obviously the same set of statements would be required to extract the data prior to reformatting it as an XML document.

To construct an XML document for a particular course outline required it to follow the rules for well-formed XML. The sequence of elements and attributes would also be validated against the DTD pattern for a particular course outline. The next step was to create the DTD. This was done by blindly following the eleven rules for moving a relational database to XML (Williams et al, 2000). These are a more detailed sequence of the simplistic three rules of Bourrett, 2000 i.e:

a. for each table create an element.
b. for each column create an attribute.
c. for each PK/FK create a child element.

An initial DTD was produced. It contained elements that mapped to tables and attributes that mapped to columns. The Williams rules produced ID’s and IDREFS for PK/FK links as well as some child elements. I doubt it was a well-formed DTD but it was a start. It was XML but it felt like a relational model. The literature suggested that it is easier and better to use containment (i.e. embedded child elements) rather than ID or IDREFS (Cover, 2000; La Quey, 2001; Park, 2000).

Ray, 2001 and Mertz, 2001 describe XML as a hierarchical data structure. Ray describes it as a tree and as boxes inside boxes (pg 30). He further suggests that you should strive for a wide bushy shrub (pg 172).

With the database in one hand, a real course outline in the other and a shrub in mind a second iteration of the DTD was constructed and then a sample

XML document based on the DTD. Were they well formed?, i.e. did they follow the rules for XML? Was the document valid?, i.e. did it match the pattern described by the DTD?

Internet Explorer (IE) was used as an initial check of the DTD and document (Lee, 2001). A few end tags were required and then IE displayed the XML course outline.

IE is not a strict XML validator. It appears to check mostly for matching start and end tags and correct nesting of elements. It did not check for the correct placement of attributes within the start tag.

It was then checked and validated using the online validator provided by the Scholarly Technology Group at Brown University (http://www.stg.brown.edu/service/XMLvalid/).

Further changes were required to achieve well-formed and valid success.

As Ray, (2001, pg 169) says developing a DTD is part art and part science. The first DTD reflected the influence of Williams et al, 2000 and Bourett, 2000. Mostly tables and columns had been mapped to elements and attributes. Ray, 2001 suggests that you should use an element when the content is more than a few words long and to use attributes as a parameter or to restrict the value (pg 61). Consequently Programmename, RPL statements, Aegrotat statements etc were redefined as elements and the XML changed accordingly. It made for a more concise DTD. This was then validated.

4. A DTD FOR COURSE OUTLINES

```
<?xml version =”1.0”?>

<!DOCTYPE courseoutline [
<!ELEMENT courseoutline (heading, thiscourse) >
<!ELEMENT  heading  EMPTY>
<!ATTLIST  heading
      faculty           CDATA #REQUIRED
      school            CDATA #REQUIRED
      programme         CDATA #REQUIRED>
<!ELEMENT thiscourse (prerequisite*, staff+, outcome+,timetable+, textbook*,
materials*, handbook*, assessment+, assessstatement*, resitprocedure*, rpl*,
aegrotat*, lectdiary, labdiary)>
<!ATTLIST   thiscourse
      coursecode              CDATA #REQUIRED
      coursename              CDATA #REQUIRED
      level                   CDATA #REQUIRED
      credits                 CDATA #REQUIRED
      semester                CDATA #REQUIRED
      year                    CDATA #REQUIRED
      courseaim               CDATA #REQUIRED
      prerequisitecomment     CDATA #IMPLIED
      timetabledlectures      CDATA #REQUIRED
      timetabledlabs          CDATA #REQUIRED
```

```
      selfdirected            CDATA #REQUIRED
      incourseassess          CDATA #IMPLIED
      examdate                CDATA #REQUIRED>
<!ELEMENT   prerequisite      EMPTY >
<!ATTLIST   prerequisite
      precode     CDATA #REQUIRED >
<!ELEMENT   staff EMPTY >
<!ATTLIST   staff
      type  (CASM|PL|CC|HOS) "CASM"
      title       CDATA #REQUIRED
      fname       CDATA #REQUIRED
      sname       CDATA #REQUIRED
      office      CDATA #REQUIRED
      xtn         CDATA #IMPLIED
      email       CDATA #IMPLIED    >
<!ELEMENT   outcome     (#PCDATA) >
<!ATTLIST   outcome
      outcomeid         ID    #REQUIRED >
<!ELEMENT   timetable   EMPTY >
<!ATTLIST   timetable
      lectlab           (LEC|LAB) "LEC"
      coursecode        CDATA #REQUIRED
      streamcode        CDATA #REQUIRED
      dayname           CDATA #REQUIRED
      time              CDATA #REQUIRED
      room              CDATA #REQUIRED  >
<!ELEMENT   textbook    EMPTY >
<!ATTLIST   textbook
      category          (REC|REQ) "REC"
      author            CDATA #REQUIRED
      title             CDATA #REQUIRED
      pub               CDATA #IMPLIED  >
<!ELEMENT   materials (#PCDATA)      >
<!ELEMENT   handbook (#PCDATA) >
<!ELEMENT   assessment  EMPTY  >
<!ATTLIST  assessment
      assesstype        CDATA #REQUIRED
      assessdesc        CDATA #REQUIRED
```

```
      assessweight      CDATA #REQUIRED
      outcomeid         IDREFS      #REQUIRED
      assessduedate     CDATA #REQUIRED
      assesslocation    CDATA #IMPLIED   >
<!ELEMENT   assessstatement   (statement*) >
      <!ELEMENT   statement   (#PCDATA) >
      <!ATTLIST   statement  order CDATA  #REQUIRED  >
<!ELEMENT         resitprocedure    (procedure*)  >
<!ELEMENT   procedure   (#PCDATA)   >
<!ELEMENT   rpl   (rplpara*) >
      <!ELEMENT rplpara (#PCDATA) >
      <!ATTLIST rplpara  order CDATA #REQUIRED>
<!ELEMENT   aegrotat (aegrotatpara*)      >
<!ELEMENT   aegrotatpara      (#PCDATA) >
<!ATTLIST aegrotatpara
      order CDATA #REQUIRED>
<!ELEMENT   lectdiary (diaryline+)  >
<!ELEMENT   labdiary (diaryline+)  >
<!ELEMENT   diaryline   EMPTY >
<!ATTLIST   diaryline
      weekno            CDATA #REQUIRED
      semesterdate      CDATA #REQUIRED
      topic1            CDATA #IMPLIED
      topic2            CDATA #IMPLIED
      notes             CDATA #IMPLIED  >
]>
```

Where to now?

Probably write a VB application that will extract the data from the database and output XML documents. An alternative would be to create the database using SQL server 2000 and use the XML features to convert SQL results into XML. Wahlin, 2001 indicates that this will return data as attributes or as elements but not some combination. The DTD would need to be adjusted accordingly.

5. CONCLUSIONS

XML is a meta-language for defining DTDs. An XML document is a text file containing data and markup. It is a huge improvement on csf in that it is self-describing data. The DTD defines a protocol for a particular document type.

If your data is stored in a RDBMS a DTD can be defined to map the tables and columns to elements and attributes as outlined by Williams et al 2000, Bourrett 2000 but it may not be a “good” DTD according to the guidelines for the use of elements and attributes (Ray 2001, Punin 2001). XML has become the standard for data interchange on the web and is supported by all the major companies (Punin 2001). An XML document is designed to be a self-

contained data set for computer consumption, which is why it is so pedantic. Good XML uses descriptive tags and a hierarchical, nested structure. The automated tools for converting SQL results to XML documents may well work but they do not necessarily produce "good" XML. i.e. some produce tags such as <row…. </row> and <column1> …. </column1> (Mertz, 2001b).

We have created a DTD for course outlines but in the light of current discussion on the use of elements verses attributes it may not be the best or simplest DTD.

REFERENCES

Banfield J. (2001) "XML goes into orbit". Computerworld (New Zealand), 7 May.

Bourret, R. P. (2000) "XML and Databases". Accessed May 2001. <http:/www.rpbourret.com/xml/XMLAndDatabases.htm>

Cover, R. (2000) "SGML/XML: Using elements and attributes". Accessed May 2001.
<http://www.oasis-open.org/cover/elementsandattrs.html>

"Extensible Markup Language (XML) 1.0 (Second Edition)". Accessed May 2001.
<http://www.w3.org/TR/REC-xml>

Ray E. T. (2001) "Learning XML 1st edition". O'Reilly & Associates.

La Quey, R. E. (2001) "SML: Simplifying XML". Accessed May 2001. <http://www.xml.com/lpt/a/1999/11/sml/index.html>

Lee, W. M. (2001) "Microsoft commits to XML". XML Journal 2,3, 10-12.

Mertz, D. (2001a) "Putting XML in context with hierarchical, relational, and object-oriented models". Accessed May 2001.
<http:/www-106.ibm.com/developerworks/xmllibrary/x-matters8/index.html>.

Mertz, D. (2001b) "DTDs and XML documents from queries". Accessed May 2001.
<http:/www-106.ibm.com/developerworks/xmllibrary/x-matters9.html>.

Park D. (2000) "Minimal XML in a nutshell". Accessed May 2001. <http://www.docuserve.com/smldev/minxmlspec.html>

Punin, J. (2001) "Programming XML in JAVA". Accessed May 2001. <http:/www.cs.rpi.edu/~puninj/XMLJ/classes.html>.

Wahlin D. (2001) "Leveraging SQL Server's XML features". XML Magazine, winter, 1, 5, 30-39.

Williams K., Brundage M., Dengler P., Gabriel J., Hoskinson A., Kay M., Maxwell T., Ochoa M., Papa J, Vanmane M. (2000) "Professional XML Databases". Wrox Press Inc.