Objectives:

1. To introduce XML
2. To discuss the relationship between XML and databases
3. To introduce DTD’s

Materials:

1. Handout: Example XML representation of a book plus DTD
2. Transparencies: Mapping XML to relational tables and to objects

I. Introduction

A. If you look at articles in the trade press or on the web, you will likely see a lot of mention of XML.

ASK Class for examples

To begin our discussion of XML, it is appropriate to consider several key questions:

1. What is XML?
2. Why is there so much interest in it? (I.e. what problem(s) does it solve?)
3. How does XML relate to databases?

B. What is XML?

1. It is perhaps best to begin by saying what XML is not. XML is not another data model (like the hierarchical, network, relational, and OO models). That is, we do not currently have - nor are we likely to have - database systems based on actually storing all information in a format like the example I am about to hand out.

2. Rather, XML is a language. The name XML is an abbreviation for eXtensible Markup Language.
a) XML is, then, a specification for how one goes about constructing a representation of information using a tagged representation.

Distribute example of an XML representation for a book. Ignore <!DOCTYPE ...> at the top and DTD at the bottom for now.

b) You may notice some similarity between the example and HTML. That is not accidental.

(1) XML and HTML share a common ancestor - SGML (Standard Generalized Markup Language).

(2) There is a variant of HTML - called XHTML - that is also valid XML (i.e. it is both).

   a) Every XHTML document is also an HTML document.

   b) But not every HTML document is a valid XHTML document.

   c) Example: All my ATM Example system pages are actually valid XHTML.

3. Actually, XML is not just a language - it is a meta-language - i.e. a language that can be used for defining other languages.

   a) For example, the handout I passed out presupposes that one has defined a language for talking about library books, in which words like “book”, “call_number”, and “author” have special meanings.

   b) It is also possible to create an XML language for talking about, say, computer-aided design of automobiles, in which words like “engine” and “chassis” and “piston” would have special meanings.

   c) XHTML that I alluded to earlier is a language in which words like “head”, “body”, “table”, and “ol” have special meanings.

4. Note that a key concept in XML is the use of tags (like “author”). This attaches semantic information to data - so that we know what the data actually means. The personal name information is not just information about any person - it is about a person who is related to the book we are describing in a specific way.
Example: Suppose we had a book description in which we recorded binding colors. Now the word “Black” would have very different meanings if it occurred as a “binding_color” rather than as the “last_name” value of an “author” or as or perhaps as part of a “title”. This facilitates semantic searches - i.e. if we are interested in a book written by someone whose name is Black, we only care about occurrences of “Black” as an author; on the other hand, if we are interested in books about “Black Bears” we are only interested if the word occurs in the title, and if we are writing a detective mystery we may care about books whose binding color is black.

Contrast this with how web search engines typically handle queries!

C. Why is there so much interest in it? (I.e. what problem(s) does it solve?)

1. One reason for interest in XML is to facilitate the interchange of information between diverse sources.
   a) In a networked world, it is common to want to combine information from several different sources.

      Example: Searching the card catalog of several different libraries for books on a given topic, and then combining the results into a single list.

   b) It is also common to find information being transmitted from one organization to another.

      (1) Example: B2B applications

      (2) Example: an item may come with descriptive information - e.g. our library has access to standardized catalog entries (MARC records) for books so that the cataloging librarian doesn’t have to figure out all the information to be entered into the catalog for a newly-received book.

   c) Raw data is not particularly useful unless it is interpreted somehow.

      (1) Example: Suppose we retrieved the following data on a book:

      QL737.123 Black Bears

      Is it a book whose title is “Black Bears”, or a book by an author named “Black” whose title is “Bears”, or perhaps a book with a black binding whose title is “Bears”?
That, of course, is the role of headings in a table. But, in a large table, the headings may be far removed from the data they describe, and it may be hard to associate the correct heading with the data, either for a human or for a computer system.

(2) Life is further complicated by differences in conventions between sites - e.g. two different sites might record an individual named “Alexander George” as

- Alexander George
- George Alexander

(3) Finally, even if we have headings associated with an object, different information sources may use different names.

Example: Suppose we are trying to combine information from three sources, one of which calls the name attributes of a person “last_name” and “first_name”, while another calls them “lname” and “fname” and another calls them “last” and “first”.

2. Another reason for interest in XML is to separate semantics from presentation.

a) Example: The html <table>, <tr>, and <td> tags specify how data is to be displayed, but tell nothing about what it means. On the other hand, the <title> tag tells us what the data means. Html mixes the two - sometimes even offering alternatives (e.g. <b> specifies boldface presentation while the <strong> tag that most browsers render as boldface is more semantic, though still not very much so.)

b) An XML document uses semantic tags, and is typically associated with a style sheet that specifies how various elements are to be displayed. Different systems can use different style sheets - e.g. one sort of style may be appropriate for a computer web browser and another for a cell phone.

D. How does XML relate to databases?

1. As I noted earlier, XML is not a data model. Though XML data can be stored in a database, one wouldn’t think of building a database system whose architecture is XML.
2. OTOH, DBMS’s may offer a mechanism for converting between the internal form used for storing data and XML.

a) Data retrieved from the database can be converted to XML to facilitate integrating it with other data from other sources or use on systems storing data in a different sort of database.

b) Arriving XML data can be converted to the format required by the database.

Example: the book example I gave you earlier could be converted to several relational tuples: a row to insert into the book table, a row to insert into the book_title table, and three rows to insert into the book_author table.

TRANSPARENCY

<table>
<thead>
<tr>
<th>Book</th>
<th>accession_number</th>
<th>call_number</th>
<th>copy_number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>42</td>
<td>QA76.9 D3</td>
<td>S5737 2002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book_title</th>
<th>call_number</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QA76.9 D3</td>
<td>Database System Concepts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book_author</th>
<th>call_number</th>
<th>author_last_name</th>
<th>author_first_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA76.9 D3</td>
<td>KORTH</td>
<td>HENRY</td>
<td></td>
</tr>
<tr>
<td>QA76.9 D3</td>
<td>SILBERSCHATZ</td>
<td>ABRAHAM</td>
<td></td>
</tr>
<tr>
<td>QA76.9 D3</td>
<td>SUDARSHAN</td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>

MAPPING THE BOOK DESCRIPTION IN XML TO RELATIONAL TABLES
Example: the book example I gave you could also be converted to a Book object to be stored in an OO database, to be associated with several Author objects that might either already exist (and be matched by their names) or might need to be created as well.

TRANSPARENCY

<table>
<thead>
<tr>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>accession_number</td>
</tr>
<tr>
<td>call_number</td>
</tr>
<tr>
<td>copy_number</td>
</tr>
<tr>
<td>title</td>
</tr>
<tr>
<td>authors</td>
</tr>
</tbody>
</table>

Author

<table>
<thead>
<tr>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
</tbody>
</table>

Author

<table>
<thead>
<tr>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
</tbody>
</table>

Author

<table>
<thead>
<tr>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
</tbody>
</table>

MAPPING THE BOOK DESCRIPTION IN XML TO OBJECTS

c) As an example of a commercial implementation, IBM offers a package called XML Extender for use with db2, which allows specification of the mappings needed to convert data between XML and relational table forms. This allows an application to communicate with the database in relational terms, and with the outside world in XML.
3. Having said this, it should be noted that a DBMS may make it possible to store data in a database in XML form. (DB2 XML Extender allows this, for example.) This would be appropriate if the database basically served to store information that was to be distributed via the worldwide web, but would not allow local processing of the information using traditional relational operations.

II. More about XML

A. A well-formed XML document consists of an **element**, which may contain other elements nested within it.

Example: Our example was an XML document describing a single book, with author elements nested within. Alternately, we could have an XML document describing all the books in the catalog, where the “top-level” element is the catalog, containing book elements nested within, each of which, in turn contain author elements within them.

B. Every element in an XML document is bracketed by start and end tags (e.g. `<book>`, `</book>`; `<author>`, `</author>`).

1. In the case where nothing needs to appear between the tags, the start and end tags can be combined into one.

   Example: `<author ... /></author>`

2. Start and end tags must be properly nested - e.g.

   `<book> ... <author> ... </book> ... </author>`

   is not properly nested and not allowed.

C. An element may incorporate any or all of the following:

1. Nested elements.

   Example: the example book element contains a call_number, copy_number, title, and three author elements within it.

2. Attributes  (name/value pairs)

   Example: the book element has an accession_number attribute and the author elements have at least a last_name attribute and sometimes a first_name attribute as well.
3. Text

Example: several of the elements (call_number, copy_number, and title) contain textual information.

4. Note that an element can contain any or all of the above.

5. A document that conforms to the basic syntax of XML is called a well-formed document.

D. XML is case sensitive, so the tags <author> and <AUTHOR> are not the same.

E. An aside on differences between XHTML and HTML:

1. Ramifications of case-sensitivity. Tags are all-lowercase in XHTML.

2. Ramifications of requiring closing tags.
   a) In some cases, end tags are optional in HTML. (E.g. </p>, </li>). In XHTML they are required.
   b) Some HTML tags do not have an end form (e.g. <br>). But both HTML and XML allow a simple tag that has no content to be self-terminating - e.g. <br />.

III. DTD’s

A. One key issue in the usefulness of XML is standardization of names for elements - e.g. we should consistently use one name like <last_name> rather than different names like <last_name>, <lname>, or <last>.

B. To facilitate this, XML allows a document to be associated with a DTD (document type description.)

1. Note the <!DOCTYPE ... reference in the example I handed out. (This must always occur first in the document.)

2. A DTD may either be part of a document, or it may be a reference to a file stored elsewhere, either in a file on the same system or at a URL on the net.
C. The handout contains an example of a DTD that might be referenced by the book_example document.

1. A DTD defines various elements that may occur in the document, using <!ELEMENT ... declarations.

2. The first element defined is the root element; a valid XML document that conforms to this DTD will consist of a single instance of this element with other elements nested within.

3. The structure of an element is defined by listing the things that can occur within it. Note that, in the example, a book element consists of a call_number element, a copy_number element, a title element, and 0 or more author elements - in that order. (The syntax here resembles that used for regular expressions - e.g. the use of + for “one or more” and * for “zero or more” plus the use of | for “or” - not shown here.

4. In an element declaration, #PCDATA stands for “parsed character data” - i.e. that the XML parser will not try to interpret, leaving it to the application that processes the document.

5. If an element is allowed to have attributes, these are specified in an <!ATTLIST ... declaration. Note that a given element can have an <!ELEMENT declaration (if it contains other elements or character data within it, an <!ATTLIST declaration (if it has attributes), or both. (Book has both; call_number and copy_number have only an <!ELEMENT declaration; and author has only an <!ATTLIST declaration.)

D. Standardized DTD’s have been developed for many fields of endeavor. By conforming to such a standardized DTD, an XML document can ensure that it is “understandable” to any audience that uses the same DTD.

EXAMPLE: There is a standard DTD for XHTML (actually three, with different levels of strictness.) An XHTML page will reference this.

```xml
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml11-strict.dtd">
```
E. A well-formed XML document that contains a DTD (or a reference to one) and that conforms to that DTD is said to be valid.

1. Validity is only possible if the document is well formed.
2. Validity requires a DTD.