XML: Data and Document Processing

Some Basic XML Concepts

David Edmond
School of Information Systems
Queensland University of Technology

February 20, 2007

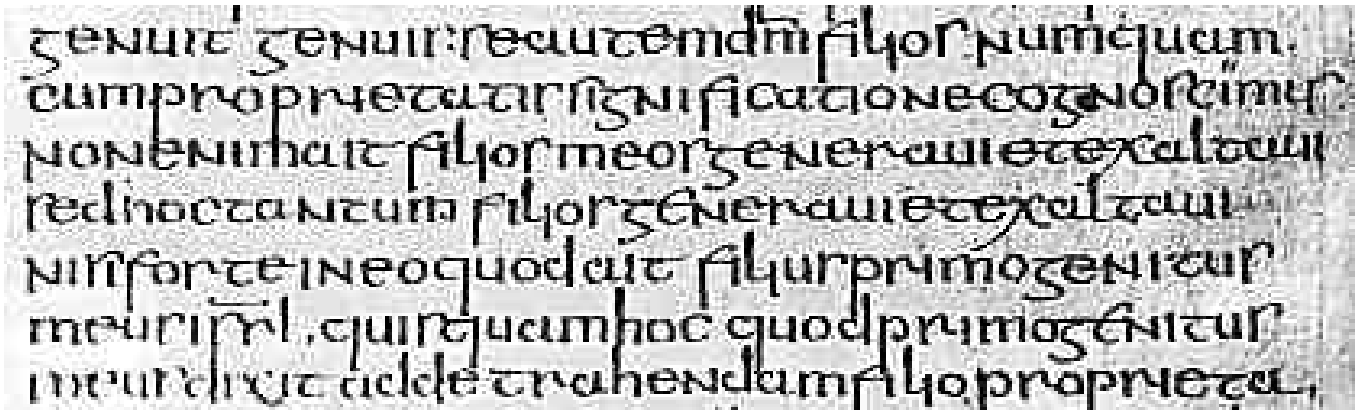
Introduction

This week's topics

- Why XML?
- Elements and attributes
- A range of examples of XML documents
- The original design goals for XML
- The XML family
- Some of the rules for *producing* an XML document

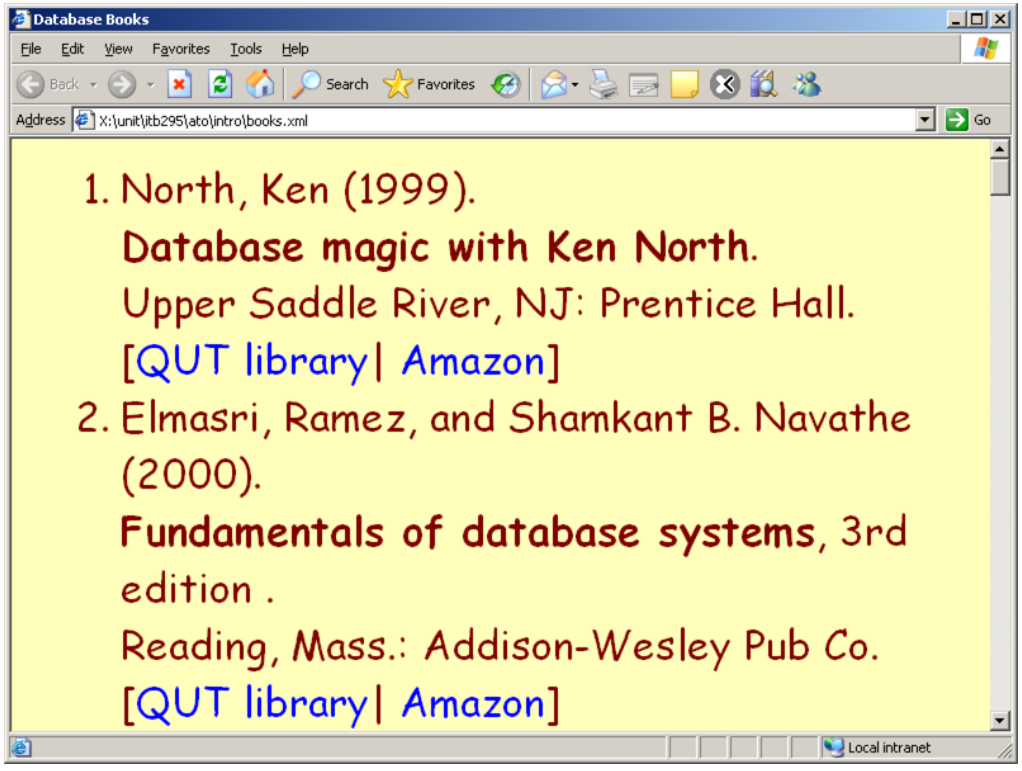
✓ `http://www.w3.org/TR/REC-xml`

What does this have to do with XML?



HTML = Data + Markup

Look at the HTML document shown below.



The links

Each book has two distinct hypertext links. For the first book, these are:
`http://libcat.qut.edu.au/search/i?SEARCH=0136471994>`
`http://www.amazon.com/exec/obidos/ASIN/0136471994/>`

- One links to the QUT library catalogue.
- The other links to Amazon.

The underlying HTML

```
<HTML>
<TITLE>Database Books</TITLE>
<BODY STYLE="margin-left: 40px;
    margin-right: 20px;
    background-color: #FFFFBB;
    color:maroon; font-family:'Comic Sans MS';
    font-size:12pt;
    padding:0px 6px">
<OL>
 <LI>North, Ken (1999).<BR/>
  <B>Database magic with Ken North</B>.<BR/>
  Upper Saddle River, NJ: Prentice Hall.<BR/>
  [<A HREF="....search/i?SEARCH=0136471994">QUT library</A>|
   <A HREF="..../ASIN/0136471994/">Amazon</A>]
  <BR/></LI>
 <LI> Elmasri, Ramez, and Shamkant B. Navathe (2000 ).<BR/>
  <B>Fundamentals of database systems</B>, 3rd edition.<BR/>
  Reading, Mass.: Addison-Wesley Pub Co.<BR/>
  [<A HREF="....search/i?SEARCH=0201542633">QUT library</A>|
   <A HREF="..../ASIN/0201542633/">Amazon</A>]
  <BR/></LI>
  .
  .
  .
    </OL>
</BODY>
</HTML>
```

Only books

```
1.  <?xml version="1.0"?>
2.  <?xml-stylesheet href="BIBstyle3.xsl" type="text/xsl"?>
3.  <bibliography>
4.   <book>
5.    <author>North, Ken</author>
6.    <title>Database magic with Ken North</title>
7.    <address>Upper Saddle River, NJ</address>
8.    <publisher>Prentice Hall</publisher>
9.    <year>1999</year>
10.   <isbn>0136471994</isbn>
11.  </book>
12.
13.  <book>
14.   <author> Elmasri, Ramez, and Shamkant B. Navathe</author>
15.   <title>Fundamentals of database systems</title>
16.   <address>Reading, Mass.</address>
17.   <publisher>Addison-Wesley Pub Co</publisher>
18.   <year>2000</year>
19.   <edition>3rd</edition>
20.   <isbn>0201542633</isbn>
21.  </book>
22.
23. <!-- Lots more <book/> elements here. -->
24.
25. </bibliography>
```

Elements and attributes

Example 1: Tags, tags, tags, ...

Consider the following snippet of information from the QUT phone book:

```
Edgar   Miss Pam   Optometry           KG B501 35695
Edmond  Dr   David Information Systems GP S842 32240
Edmonds Dr   Ian   Physical Sciences   GP M206 32584
```

We could write this as an XML document:

```
<Phonebook>
  <Entry>
    <LastName>Edgar</LastName>
    <Title>Miss</Title>
    <FirstName>Pam</FirstName>
    <School>Optometry</School>
    <Campus>GP</Campus>
    <Room>B501</Room>
    <Extension>35695</Extension>
  </Entry>
<!-- Lots more <Entry/> elements here. -->
</Phonebook>
```

XML elements

- An element is *not* the same as a tag!
- XML is case-sensitive. (`<LastName>` $\neq$ `<lastname>`)
- The names `XML`, `xML` `...xml` are reserved.
- A name cannot start with a digit:
  - illegal: `<911/>`
  - legal: `<_911/>`
- A name may only contain letters, digits, `'-'`, `'_'`, `'.'`
  - illegal: `<Jack's Diary/>`
  - legal: `<Jacks_Diary/>`
- Elements cannot overlap:
  - illegal: `<OutTag><InTag><OutTag/><InTag>`
  - legal: `<OutTag><InTag><InTag/><OutTag>`

Empty elements

- An *empty* element is one that has no “content”. There are two ways of writing them:
  - `<AnEmptyElement/>`.
  - `<AnEmptyElement></AnEmptyElement>`.
- An empty element can be useful. For example, the line-break element in XHTML may be written `<br/>` or `<br></br>`.
- Empty elements may still carry information by means of attributes. For example, the image element in XHTML may be written `<img href="mydog.jpg" alt="woof"/>`.

Example 2: Attributes

We could make a slight variation to each phonebook entry in the following way:

```
<Entry>
  <LastName Title="Miss">Edgar</LastName>
  <FirstName>Pam</FirstName>
  <School>Optometry</School>
  <Campus>GP</Campus>
  <Room>B501</Room>
  <Extension>35695</Extension>
</Entry>
```

- The `<Title/>` element has been turned into an attribute of the `<LastName/>` element.
- Why might we make such a change?

Example 3: Yet more attributes

We could make even more extensive use of attributes, by expressing each entry in the following way:

```
<Entry Extension="35695">
  <Name Title="Miss"
        LastName="Edgar"
        FirstName="Pam"/>
  <School>Optometry</School>
  <Campus>GP</Campus>
  <Room Building="B">501</Room>
</Entry>
```

XML attributes

- The names of XML attributes follow the same conventions as elements.
- Their values must be demarcated by quotes (") or apostrophes (').
  - illegal: `<LastName Title=Mr>`
  - legal: `<LastName Title="Mr">`
  - illegal: `<if test="Age<25">`
  - illegal: `<Album Category='Rock'n'Roll'>`
  - legal: `<Album Category="Rock'n'Roll">`
  - legal: `<Album Category='Rock"n"Roll'>`
- What if you need to use both a quotation mark *and* an apostrophe in an attribute?
- When should you use attributes to convey information?

Some other XML concepts

XML comments

- Comments begin `<!--` and end `-->`.
- For example `<!-- Created \today\ -->`
- Two consecutive dashes (`--`) should *never* appear within a comment.

The XML prolog

At the very beginning of any XML document there may be a “prolog”, that is, some general preliminary information about the document. This could include:

- The way the document is physically encoded.
- A pointer to a stylesheet to be used to transform the document.
- A definition of how the document should be validated before any processing occurs.
- Comments on the document.

Some exercises

Example 4: Duncan

Suppose we want to send a letter to Duncan:

```
1. Duncan Hunter
2. 45 Somerset Street
3. Clayfield
```

Encode the above as XML.

```
1. <envelope>
2.  <name>
3.   <given>...............</given>
4.   <family>...............</family>
5.  </name>
6.  <street>
7.   <number>...............</number>
8.   <name>...............</name>
9.  </street>
10.  <suburb>...............</suburb>
11. </envelope>
```

Example 4: Duncan

Suppose we want to send a letter to Duncan:

```
1. Duncan Hunter
2. 45 Somerset Street
3. Clayfield
```

Encode the above as XML.

```
1. <envelope>
2.  <name>
3.   <given>Duncan</given>
4.   <family>Hunter</family>
5.  </name>
6.  <street>
7.   <number>45</number>
8.   <name>Somerset</name>
9.  </street>
10.  <suburb>Clayfield</suburb>
11. </envelope>
```

Example 5: It's simply not cricket!

Sports results are suitable targets for XML. Consider this cricket innings:

```
Team:                    Australia
Haydon b Jones ..............  58
Sanger c Kent, b Jones ......  87
Wonting not out .............  55
Lemon not out ...............  13
```

Suppose we encoded Haydon's performance as follows:

```
<bat>
 <name>Haydon</name>
 <runs>58</runs>
 <out style="bowled">
  <bowler>Jones</bowler>
 </out>
</bat>
```

How might we encode Sanger?

```
<bat>
 <name>Sanger</name>
 <runs>87</runs>
 <out style=".................">
  <bowler>Jones</bowler>
  <.............>Kent</.............>
 </out>
</bat>
```

Example 5: It's simply not cricket!

Sports results are suitable targets for XML. Consider this cricket innings:

```
Team:                    Australia
Haydon b Jones ..............  58
Sanger c Kent, b Jones ......  87
Wonting not out .............  55
Lemon not out ...............  13
```

Suppose we encoded Haydon's performance as follows:

```
<bat>
 <name>Haydon</name>
 <runs>58</runs>
 <out style="bowled">
  <bowler>Jones</bowler>
 </out>
</bat>
```

How might we encode Sanger?

```
<bat>
 <name>Sanger</name>
 <runs>87</runs>
 <out style="caught">
  <bowler>Jones</bowler>
  <catcher>Kent</catcher>
 </out>
</bat>
```

Example 6: What's XML got to do with SQL?

How might write the following as XML?

```
1. Select   CarNr, Make, Cost
2. From     Cars
3. Order by Cost
```

```
1. <sql>
2.  <select order="...............">
3.   <col>CarNr</col>
4.   <col>Make</col>
5.   <col>Cost</col>
6.  </select>
7.  <..............>
8.   <table>Cars</table>
9.  </..............>
10. </sql>
```

Example 6: What's XML got to do with SQL?

How might write the following as XML?

```
Select   CarNr, Make, Cost
From     Cars
Order by Cost
```

```
<sql>
 <select order="cost">
  <col>CarNr</col>
  <col>Make</col>
  <col>Cost</col>
 </select>
 <from>
  <table>Cars</table>
 </from>
</sql>
```

Example 7:

Consider the following simple program:

```
while (x.test()) {
   y.modify();
   x.meddle();
}
```

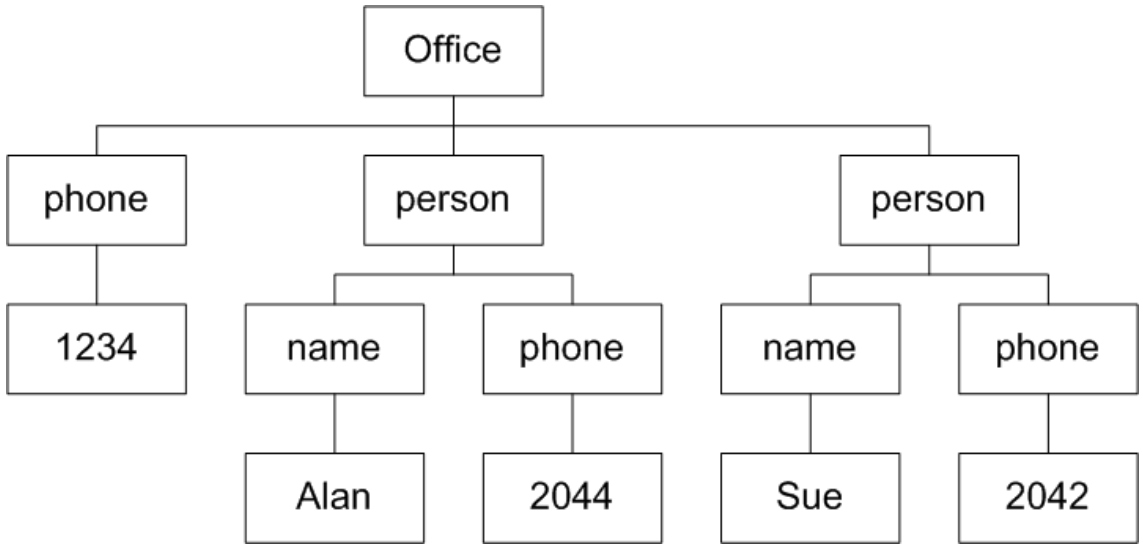
We might encode this, XML style, as follows:

```
<while>
  <condition>
    <var>x</var><method>test</method>
  </condition>
  <body>
    <statement>
      <var>y</var><method>modify</method>
    </statement>
    <statement>
      <var>x</var><method>meddle</method>
    </statement>
  </body>
</while>
```

NB: Most important!

An XML document is a tree

```
1.  <office>
2.    <phone>1235</phone>
3.    <person>
4.      <name>Alan</name>
5.      <phone>2044</phone>
6.    </person>
7.    <person>
8.      <name>Sue</name>
9.      <phone>2043</phone>
10.   </person>
11. </office>
```



XML: at the beginning

XML design goals

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

✓ `http://www.w3.org/TR/REC-xml`

✓ `http://www.xml.com/axml/testaxml.htm`

The XML Family

- XML: for generating languages that describe information.
- DOM: a programming interface for accessing and updating documents.
- XML schema: a language for specifying the structure and content of documents.
- XSLT: a language for transforming documents.
- XPath: a query language for navigating XML documents.
- XPointer: for identifying fragments of a document.
- XLink: generalises the concept of a hypertext link.
- XInclude: for merging documents.
- XQuery: a language for making queries across documents.
- RDF: a language for describing resources.

The XML specification: a start

The XML Specification

- The original specification was first released in 1998.
- It specified both the physical and logical makeup of an XML document.
- A 2nd edition was released in 2000, a 3rd edition in 2004, and a 4th in 2006.
- None of these later editions significantly changed the original.

✓ `http://www.w3.org/TR/REC-xml`

Production rules

The core of the specification is a series of production rules. Each “production” has the form:

1. `symbol ::= expression`

Each production is given a number.

Selected productions

Some of the productions appearing in the XML specification are:

```
1. document ::= prolog element Misc*
2. prolog   ::= XMLDecl? Misc* (doctypedecl Misc*)?
3. element  ::= EmptyElemTag | STag content ETag
4. Misc     ::= Comment | PI | S
5. Comment  ::= '<!--' ((Char - '-') | ('-' (Char - '-')))*
6.              '-->'
7. S        ::= (#x20 | #x9 | #xD | #xA)+
```

The production that defines an XML document suggests that there can only be one element. But we know that is not true. So . . .

EBNF: Core

The rules are expressed in a version of the notation called *Extended Backus-Naur Form*. And this notation allows the following forms of expression:

`A?`	matches A or nothing; optional A.
`A B`	matches A followed by B.
`A | B`	matches A or B but not both.
`A+`	matches one or more occurrences of A.
`A*`	matches zero or more occurrences of A.
`#xN`	matches the ISO/IEC 10646 character corresponding to the hexadecimal number `N`.
`(expression)`	expression is treated as a unit and may be combined as described in this list.

EBNF: Literals, ranges, enumerations, etc

`'string'`	matches the enclosed string exactly.
`"string"`	matches the enclosed string exactly.
`A - B`	matches any string that matches A but does not match B.
`[a-zA-Z]`	matches any Char with a value in the range(s) indicated (inclusive).
`[abc]`	matches any Char with a value among the characters enumerated. Enumerations and ranges can be mixed in one set of brackets.
`[^a-z]`	matches any Char with a value outside the range indicated.
`[^abc]`	matches any Char with a value not among the characters given. Enumerations and ranges of forbidden values can be mixed in one set of brackets.

Names and tokens

The XML specification contains the following set of production rules:

```
1. NameChar ::= Letter | Digit | '.' | '-' | '_' | ':'
2.              | CombiningChar | Extender
3. Name     ::= (Letter | '_' | ':') (NameChar)*
4. Names    ::= Name (S Name)*
5. Nmtoken  ::= (NameChar)+
6. Nmtokens ::= Nmtoken (S Nmtoken)*
```

Example 8: Names and tokens

Classify each of the following as being one of the following: `Name`, `Names`, `Nmtoken` or `Nmtokens`.

1. `xsl:value-of` .....
2. `123ABC` .....
3. `pardon_my_french` .....
4. `_excuse_me` .....
5. `-dash-dash-dash` .....
6. `dash dash dash` .....
7. `15 August` .....
8. `_:_1` .....

Example 8: Names and tokens

Classify each of the following as being one of the following: `Name`, `Names`, `Nmtoken` or `Nmtokens`.

1. `xsl:value-of` — `Name`
2. `123ABC` — `Nmtoken`
3. `pardon_my_french` — `Name`
4. `_excuse_me` — `Name`
5. `-dash-dash-dash` — `Nmtoken`
6. `dash dash dash` — `Names`
7. `15 August` — `Nmtokens`
8. `_:_1` — `Name`

Conclusions

About this week's topics

- XML was designed to allow the data underlying some message or document to be expressed.
- Display or formatting information was to be removed.
- Formatting is the business of the receiving program.
- The most fundamental component of an XML document are its elements and attributes.
- Every XML document is tree-structured (hierarchical).
- The (meta-)language is expressive enough to allow a limitless range of examples.
- The rules for constructing an XML document are written in the form of a set of production rules.

✓ `http://www.w3.org/TR/REC-xml`

Next week's topics

- *XPath:* a language for querying XML documents.
- *Namespaces:* how to mix different vocabularies in the one document.