XML Notes – 10/31/03

Defined by the WWW Consortium (W3C)
Originally intended as a document markup language not a database language
Documents have tags giving extra information about sections of the document

<title> XML </title> <slide> Introduction …</slide>

Derived from SGML (Standard Generalized Markup Language), but simpler to use than SGML
Extensible, unlike HTML – what do we mean by this?

Users can add new tags, and *separately* specify how the tag should be handled for display

Goal was (is?) to replace HTML as the language for publishing documents on the Web

XML consists of *tags* and *text*

Tags come in pairs <date> ...</date>

They must be properly nested

<date> <day> ... </day> ... </date> --- good
<date> <day> ... </date>... </day> --- bad

(You can't do ... ... as in HTML)

Nesting tags can be used to express various structures. E.g. A tuple (record) :

```
<person>
  <name> Malcolm Atchison </name>
  <tel> (215) 898 4321 </tel>
  <email> mp@dcs.gla.ac.sc </email>
</person>
```

•We can represent a list by using the *same* tag repeatedly:

```
<addresses>
   <person> ... </person>
   <person> ... </person>
   <person> ... </person>
   ...
</addresses>
```

XML provides a hierarchical data model

- two main structuring concepts
  - elements
  - attributes (not same as db terminology)

element – segment of an XML document between an opening and a corresponding closing tag is called an *element.*

```
<person>
      <name> Malcolm Atchison </name>
        <tel> (215) 898 4321 </tel>
        <tel> (215) 898 4321 </tel>
        <email> mp@dcs.gla.ac.sc </email>
  </person>
```

- complex elements – constructed from other elements hierarchically
- (show Toy example XML)

- attribute – provide additional information that describes elements

An (opening) tag may contain *attributes.* These are typically used to describe the content of an element

```
<entry>
    <word language = “en”> cheese </word>
    <word language = “fr”> fromage </word>
    <word language = “ro”> branza </word>
    <meaning> A food made … </meaning>
</entry>
```

Another common use for attributes is to express dimension or type

```
<picture>
   <height dim= “cm”> 2400 </height>
   <width dim= “in”> 96 </width>
   <data encoding = “gif” compression = “zip”>
      M05-.+C$@02!G96YE<FEC ...
   </data>
</picture>
```

A document that obeys the “nested tags” rule and does not repeat an attribute within a tag is said to be *well-formed* .

Document Type Descriptors (DTDs) impose structure on an XML document.

- The type of an XML document can be specified using a DTD
- DTD constrains structure of XML data
  - What elements can occur
  - What attributes can/must an element have
  - What subelements can/must occur inside each element, and how many times.
- DTD does not constrain data types
  - All values represented as strings in XML
- DTD syntax
  - <!ELEMENT element (subelements-specification) >
  - <!ATTLIST element (attributes) >

There is *some* relationship between a DTD and a schema, but it is not close – there is still a need for additional "typing" systems that schemas have and DTDs do not.

The DTD is a *syntactic* specification.

(Show address book example – setup for DTD example)

For Address book DTD example:

name	to specify a name element
greet?	to specify an optional (0 or 1) greet elements
name,greet?	to specify a name followed by an optional greet
addr*	to specify 0 or more address lines
tel | fax	a tel *or* a fax element
(tel | fax)*	0 or more repeats of tel or fax
email*	0 or more email elements

So the whole structure of a person entry is specified by

name, greet?, addr*, (tel | fax)*, email*

This is known as a *regular expression*. Why is it important?

Summary of XML regular expressions

A	The tag A occurs
e1,e2	The expression e1 followed by e2
e*	0 or more occurrences of e
e?	Optional -- 0 or 1 occurrences
e+	1 or more occurrences
e1 | e2	either e1 or e2

(e) grouping

(Show DTD example slides)

Specifying attributes in the DTD

```
<!ELEMENT height (#PCDATA)>
<!ATTLIST height
    dimension CDATA #REQUIRED
    accuracy CDATA   #IMPLIED >
```

The dimension attribute is required; the accuracy attribute is optional.
CDATA is the “type” of the attribute -- it means string.

ID and IDREF attributes:

ID – defines a unique attribute
IDREF – refers to an ID from another element

- similar to primary keys and foreign keys

Consistency of ID and IDREF attribute values

- If an attribute is declared as ID
  - the associated values must all be distinct (no confusion)
- If an attribute is declared as IDREF
  - the associated value must exist as the value of some ID attribute (no dangling “pointers”)
- Similarly for all the values of an IDREFS attribute
- ID *and* IDREF *attributes are not typed*

Connecting the document with its DTD

In line: defined right in the xml document itself – not very flexible or reusable
<?xml version="1.0"?> <!DOCTYPE db [<!ELEMENT ...> …]> <db> ... </db>

Another file: local file or on reachable file system

```
<!DOCTYPE db SYSTEM "schema.dtd">
```

A URL:

```
<!DOCTYPE db SYSTEM
    "http://www.schemaauthority.com/schema.dtd">
```

DTDs v.s Schemas (or Types)

- By database (or programming language) standards DTDs are rather weak specifications.
  - Only one base type -- PCDATA
  - No useful “abstractions” e.g., sets
  - IDREFs are untyped. You point to something, but you don’t know what!
  - No constraints e.g., child is inverse of parent
  - No methods
  - Tag definitions are *global*
- Some of the XML extensions impose something like a schema or type on an XML document.

XML Schema

- XML Schema is a more sophisticated schema language, which addresses the drawbacks of DTDs. Supports
  - Typing of values
    - E.g. integer, string, etc
    - Also, constraints on min/max values
  - User defined types
  - Is itself specified in XML syntax, unlike DTDs
    - More standard representation, but verbose
  - Is integrated with namespaces
  - Many more features
    - List types, uniqueness and foreign key constraints, inheritance ..
- BUT: significantly more complicated than DTDs, not yet widely used.

- xsd – XML Schema Definition

- xmlns – xml namespace

(show schema example slides)

Extracting XML Documents from Relational Databases into XML Schema

- XML uses hierarchical tree model to represent documents
- It is natural to us ER model corresponding to relational DB as a starting point

- example: Toy database (recall the ER diagram – show on slide)

  - extract a document hierarchy from the ER diagram
  - decide which relation to choose as root of the tree (there may be several options)
  - let’s choose toy: (see hierarchical diagram on slide)

- In diagram – consequence of choosing Toy as root – manuf info will have to be stored for each toy in the database