Data Bases II
XML Data Management

Michele Beretta

michele.beretta@unibg.it

XML

- e**X**tensible **M**arkup **L**anguage
- Data representation format proposed by W3C

Original idea: a *meta-language* used to specify markup languages. XML was designed to **describe data** and focus on what data is.

HTML (HTML5 at least) is a “special case” of XML.

XML can be used to:

- **Exchange data** with text files (software-independent and hardware-independent)
- **Store data** in text files
- **Separate** data from its representation

Syntax

Very simple, and very strict. An example

```
<?xml version="1.0" encoding="UTF-8" ?>
<note>
  <to>John</to>
  <from>Jane</from>
  <heading>Reminder</heading>
  <body>Never gonna give you up!</body>
</note>
```

First line

```
<?xml version="1.0" encoding="UTF-8" ?>
```

is the **declaration**, defines the version and the encoding.

Syntax

Then, we have the root element `<node> [...] </node>`, which has 4 children:

- `<to>John</to>`
- `<from>Jane</from>`
- `<heading>Reminder</heading>`
- `<body>Never gonna give you up!</body>`

Each one with some text.

Some rules:

- All elements **must have a closing tag**
- Tags are **case sensitive** (unlike HTML)
- `<!-- This is a comment -->`
- All elements **must be properly nested**
- All documents **must have a single root node**
- All elements must be inside this root node (either directly or indirectly)

Elements

Elements are comprised of everything from (including) the start tag to (including) the end tag, and can have **different content types** (other elements, text, mixed, and so on).

Elements have some naming rules:

- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation
- Names must not start with xml, XML, Xml, …
- Names cannot contain spaces
- The colon : should not be used as it is reserved for namespaces

An element can have **attributes**, which are always quoted (either double or single). They contain only a single value, and cannot be structured. Also, they're not easy to test against a DTD. As a rule of thumb:

- Elements describe data
- Attributes provide information that is not relevant to the data
- Attributes provide information that is for metadata (e.g., ID)

Elements

Another example

```
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>
    Introduction to XML
    <section>What is HTML</section>
    <section>What is XML</section>
  </chapter>
  <chapter numberOfPages="23">
    XML Syntax
    <section>Elements must have a closing tag</section>
    <section>Elements must be properly nested</section>
  </chapter>
</book>
```

Namespaces

Namespaces are used to solve **name conflicts** (e.g., what happens if we merge two documents that have a `<table>`?) by introducing a **prefix**.

Take the element `<ns:name>` – here, `ns` is the namespace.

The XML Namespace (xmlns) attribute

A *Uniform Resource Identifier* (**URI**) is a string of characters which identifies an Internet Resource. The most URI is the *Uniform Resource Locator* (**URL**) which identifies an Internet domain address.

The `xmlns` is spaced in the start tag of an element and appears as so

```
xmlns:[namespace-prefix]="[namespace-URI]"
```

All child elements with the same prefix are associated with the same namespace.

Note that the address is not “used” by the parser to look up information, it’s just a string (although the URL usually contains some information about the namespace).

DTD

An XML document can be:

- **well formed** if it has correct XML syntax
- **valid** if it also conforms to a DTD

A *Document Type Definition* (**DTD**) defines the legal building blocks of an XML document. Can be declared inline or in a separate file and referenced.

An example

```
<?xml version="1.0" ?>
<!DOCTYPE note
  [ <!ELEMENT note (to, from, heading, body)>
    <!ELEMENT to (#PCdata)
    <!ELEMENT from (#PCdata)
    <!ELEMENT heading (#PCdata)
    <!ELEMENT body (#PCdata) ]>
<note>[...]</note>
```

DTD Syntax – Elements

- Empty elements: `<!ELEMENT name EMPTY>`
- Elements with only character data `<!ELEMENT name (#PCdata)>`
- Elements with any content `<!ELEMENT name ANY>`
- Elements with only specific children `<!ELEMENT name (child1, child2)>` in that order
- Elements with alternative specific children `<!ELEMENT name (child1 | child2)>`
- Children can have a sign after the name
  - `+` for 1 or more occurences
  - `*` for 0 or more occurences
  - `?` for 0 or 1 occurence
  - Nothing for exactly one

DTD Syntax – Attributes

The syntax is simply

```
<!ATTLIST element attribute type default-value>
```

Attribute types:

- `Cdata`: character data
- `(en1|en2|..)`: one from the list
- `ID`: a unique id
- `IDREF`: the id of another element
- `IDREFS`: a list of other ids
- `NMTOKEN`: a valid XML name
- `NMTOKENS`: a list of valid XML names
- `ENTITY`: an entity
- `ENTITIES`: a list of entities
- `NOTATION`: a name of a notation
- `xml`: a predefined xml value

Default values:

- A specific value
- `#REQUIRED`
- `#IMPLIED`: the attribute doesn't have to be included
- `#FIXED [...]`: the attribute always has the same value

XML Schema

XML schema is a **richer** XML-based alternative to DTD, it uses XML itself. The language is also called *XML Schema Definition* (**XSD**).

An example:

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" …>
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML Schema – Elements

The `note` element is a *complex type* because it contains other elements.

Other elements (`to`, `from`, `heading`, `body`) are *simple types* because they do not contain other elements, “only text” (it can be a boolean, a string, a date, …).

You can add restrictions (*facets*) to a ddata type to limit its content, e.g., it must match a pattern.

Some allowed types:

- `xs:string`
- `xs:decimal`
- `xs:integer`
- `xs:boolean`
- `xs:date`
- `xs:time`

XML Schema – Elements

Moreover, simple elements can have a *default value*, a *fixed value*, *restrictions*, and so on.

Examples:

```
<!-- With a default value -->
<xs:element name="color" type="xs:string" default="red" />
<!-- With a fixed value -->
<xs:element name="color" type="xs:string" fixed="red" />
<!-- With restrictions -->
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0" />
      <xs:maxInclusive value="100" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML Schema – Elements

Example continued:

```
<!-- With enumeration -->
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi" />
      <xs:enumeration value="Golf" />
      <xs:enumeration value="BMW" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<!-- With pattern -->
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string"> <xs:pattern value="[a-z]" /> </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML Schema – Attributes

The syntax is similar to that of elements:

```
<xs:attribute name="..." type="..." />
```

With the same types as elements.

XML Schema – Complex elements

A complex element simply contains other elements and/or attributes. There are four kinds of complex elements:

- Empty elements
- Elements with only text
- Elements with only other elements
- Elements with mixed content

Example:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string" />
      <xs:element name="lastname" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Schema – Complex elements

The empty element is defined as

```
<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger" />
  </xs:complexType>
</xs:element>
```

If the `xs:complexType` has the attribute `mixed=true`, then the content can contain text too.

XML Schema – Complex elements

Up until now we have used the `<xs:sequence` element, which wants all child to appear once in that order. The `<xs:all>` indicator specifies that the children can appear in any order and that each child must appear once.

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string" />
      <xs:element name="lastname" type="xs:string" />
    </xs:all>
  </xs:complexType>
</xs:element>
```

The `<xs:choice>` specifies a choice of one among the children. The `maxOccurs` and `minOccurs` attributes on an element specify the min and max number of times that element can appear (can be `unbounded`).

The `<xs:any>` enables us to use elements that are not defined by the schema. Similarly, we have `<xs:anyAttribute>`.