Big Data and Data Mining

Introduction to XML

Flavio Bertini

flavio.bertini@unipr.it

The origins of XML

- XML (eXtensible Markup Language) derives from SGML (Standard Generalized Markup Language)
  - Both with XML and SGML it is possible to define markup languages specific to several domains, such as finance or math
    - For instance, HTML is one of the languages derived from SGML
  - With respect to SGML, XML is easier to use, and it’s designed to specify markup languages to be used on the Internet
  - Maybe it is a little hard to understand, but XML does not DO anything
- Initially XML was born as a format for data exchange and XML is often used to separate data from presentation
- We will look at XML for its data storage and data lookup capabilities
- Consequently, we will not go through all the in-depth details but only the aspects relevant to data management

Example of XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```

https://www.w3schools.com/xml/

Markup and *character data*

```
<?xml version="1.0" encoding="UTF-8"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```

Prolog (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```

PROLOG

Prolog (2/2)

- The prolog contains information useful for the interpretation of the document
- In particular, it can contain:
  - A declaration that the document is in XML format (optional)
  - A *grammar* (DOCTYPE) that allows to validate the content of the document (optional)
  - Comments and information for software applications that will use the document (Processing Instructions, or PI) (zero or more)

Body of the document (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
    <!-- editor is missing! -->
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```

DOCUMENT BODY

Body of the document (2/2)

- The body of the document is made of one **element**, which itself can contain other nested elements in its content, and also comments
- We will see, through some examples, how to write *well-formed* elements

Elements

- An element can be of two forms:

```
<name attributes_list>        → extended form
    content…
</name>

<name attributes_list />      → short form or self-closed
```

Well-formed elements and attributes (1/5)

- Each element must be contained between an opening tag and a closing /tag or with a *short form*

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
      <title xml:lang="en">
      The silence of the lambs
      </title>
      <price euro="7"/>
</doc>
```

```
<BR>
```

ERROR: this tag (BR) is opened but never closed!

Well-formed elements and attributes (2/5)

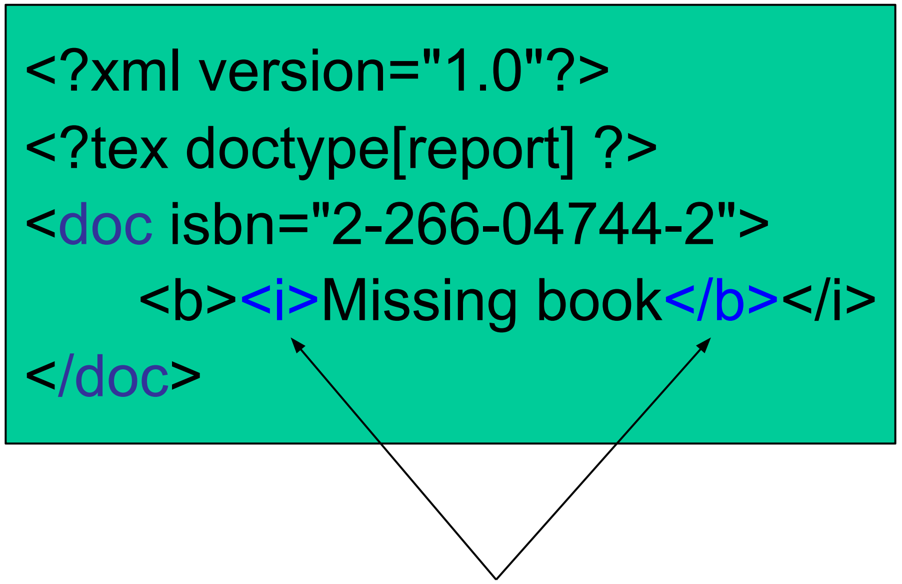
- The names of elements and attributes are *case-sensitive*

ERROR

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
        <title xml:lang="en">
        The silence of the lambs
        </title>
        <price euro="7"/>
</DOC>
```

Well-formed elements and attributes (3/5)

- Elements must be nested correctly, and there is only one element that contains all the other elements



```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
      <b><i>Missing book</b></i>
</doc>
```

Well-formed elements and attributes (4/5)

- Values of the attributes must be contained between quotes or double quotes

```
<?xml version="1.0"?>
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">
      <title xml:lang=en>
      The silence of the lambs
      </title>
      <price euro="7"/>
</doc>
```

ERROR

Well-formed elements and attributes (5/5)

- An element cannot have more than one attribute with the same name

WRONG

```
<book author=“Doe” author=“Blake”/>
```

CORRECT

```
<book>
   <author>Doe</author>
   <author>Verdi</author>
</book>
```

Document Type Declaration

- In the beginning of an XML document there can be a **document type declaration**
- This declaration contains a grammar, named **Document Type Definition**, or **DTD**, with the double purpose of *constrain* and *complete* the documents
- The DTD is made of **markup declarations**, which define what can be and cannot be written in the related XML document
- The DTDs determine which elements can be included in the document, how they can be used, what are the default values of the attributes of the elements, and other constraints

DTD example (1)

XML

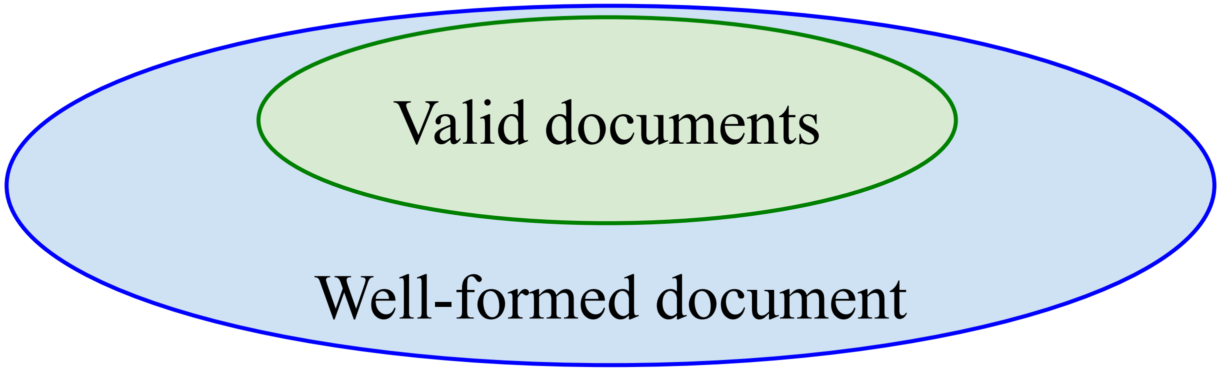
```
<?xml version="1.0"
encoding="UTF-8"?>
<!DOCTYPE note SYSTEM
"Note.dtd">
<note>
 <to>Tove</to>
 <from>Jani</from>
 <heading>Reminder</heading>
 <body>Don't forget me this
       weekend!</body>
</note>
```

DOCTYPE

```
<!DOCTYPE note
[
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
```

Valid and well-formed documents

- An XML document is valid if:
  - It contains a DTD
  - It complies to it



The following is a well-formed document but it’s not valid, because it doesn’t have a DTD:

```
<greetings>Hello, world!</greetings>
```

Examples of Document Type Declaration 1/2

- We define a type of document named *greetings* that contains an element <greetings>, which do not contain any other element. Then, the attributes of <greetings> are declared, in this example only the *id* attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE greetings [
    <!ELEMENT greetings (#PCDATA)>
    <!ATTLIST greetings
              id          ID       #REQUIRED>
    ]>
<greetings id="0001">Hello,
world!</greetings>
```

D
T
D

Examples of Document Type Declaration 2/2

- The same constraints can be specified inside an external file (hello.dtd) and declared in the following way

```
<?xml version="1.0"?>
<!DOCTYPE greetings SYSTEM "hello.dtd">
<greetings>Hello, world!</greetings>
```

DTD example (2)

XML

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

<!DOCTYPE note [
  <!ENTITY nbsp "&#xA0;">
  <!ENTITY writer "Writer: Donald Duck.">
  <!ENTITY copyright "Copyright: W3Schools.">
]>

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  <footer>&writer;&nbsp;&copyright;</footer>
</note>
```

OUTPUT

```
<note>

  <to>Tove</to>

  <from>Jani</from>

  <heading>Reminder</heading>

  <body>Don't forget me this
weekend!</body>

  <footer>Writer: Donald Duck.
Copyright: W3Schools.</footer>

</note>
```

DTD and XML Schema

- A DTD constrains an XML document
- It is however possible to specify constraints more complex than the one allowed by DTD
  - For instance: imported keys, uniqueness constraints, or the domains of elements and attributes (as in SQL)
- The **XML Schema** allows to specify this kind of constraints, and it is therefore an alternative to the DTD
- Moreover, the constraints of XML Schema are expressed in XML
  - XML Schemas are extensible to additions
  - XML Schemas support data types
  - XML Schemas support namespaces

XML Schema: an example

- The line xmlns:xsi="..." tells the parser that this document should be validated against a schema
- The line xsi:noNamespaceSchemaLocation="shiporder.xsd" specifies WHERE the schema resides (here it is in the same folder as "shiporder.xml")

```
<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
...
```

XML Schema

XML

```
<?xml version="1.0"?>

<note
xmlns="https://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.w3schools.com/xml
note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XML SCHEMA

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">

<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>
```

DTD and XML Schema - comments

- XML does not require a DTD or XML Schema. When you are experimenting with XML, or when you are working with small XML files, creating DTDs may be a waste of time
- However, with DTD and XML Schema, XML files can carry a description of its own format
  - Independent groups of people can agree on a standard for interchanging data
  - Data receive from the outside world can be verified
- One of the greatest strengths of XML Schemas is the support for data types:
  - It is easier to describe document content
  - It is easier to define restrictions on data
  - It is easier to validate the correctness of data
  - It is easier to convert data between different data types

JSON vs XML

JSON	XML
It is JavaScript Object Notation, based on JavaScript language	It is Extensible markup language, derived from SGML
It is a way of representing objects	It is a markup language and uses tag structure to represent data items
It does not provides any support for namespaces*	It supports namespaces
It supports array	It doesn't supports array
Its files are very easy to read as compared to XML	Its documents are comparatively difficult to read and interpret
It doesn't use end tag	It has start and end tags
It doesn't supports comments	It supports comments
It supports only UTF-8 encoding	It supports various encoding

**Valid Data Types In JSON: string, number, (JSON) object, array, boolean, and null*

JSON vs XML: an example

JSON

```
{"employees":[

  {"name":"Shya",
"email":"shy@mai.com"},

  {"name":"Bob",
"email":"bob@mail.com"},

  {"name":"Jai",
"email":"jai@gmail.com"}

]}
```

XML

```
<employees>
    <employee>
        <name>Shya</name>
        <email>shy@mai.com</email>
    </employee>
    <employee>
        <name>Bob</name>
        <email>bob@mail.com</email>
    </employee>
    <employee>
        <name>Jai</name>
        <email>jai@gmail.com</email>
    </employee>
</employees>
```

Proper and improper usage of XML

- XML allows a great degree of freedom to the designers of XML documents
- The designer can decide the tags and attributes to use, and where to put the data
- When XML is used to store data, particular attention must be paid to the correct usage of the elements of the data model: elements, attributes, contents, hierarchies
- We will now provide some guidelines

1 – Data in elements content

- Data must be stored in the content of the elements, for instance:

```
<book>
    <title>The Great Gatsby</title>
</book>
```

Must be preferred to

```
<book title="The Great Gatsby" />
```

AVOID THIS

2 – Metadata in attributes or names of elements

```
<book>
    <property>
        <name>title</name>          TO AVOID
        <value>The Great Gatsby</value>
     </property>
</book>
```

- In this example, title is a metadata, therefore it must not appear as content

3 – (In)Correct usage of hierarchies

```
<db>
    <title>The Great Gatsby</title>          TO AVOID
    <author>F Scott Fitzgerald</author>
    <title>For Whom the Bell Tolls</title>
    <author>Ernest Hemingway</author>
</db>
```

- The two titles and the two authors are only separated by the order of elements, while they compose well distinct objects (the refer to different books)
- <title> and <author> must be child of one element <book>, and not child of the element <db>

4 – Correct use of hierarchies

```
<db>
    <book>
        <title>The Great Gatsby</title>
        <author>F Scott Fitzgerald</author>
    </book>
    <book>
        <title>For Whom the Bell Tolls</title>
        <author>Ernest Hemingway</author>
    </book>
</db>
```

Example:

https://www.w3schools.com/xml/Books.xml

https://www.w3schools.com/xml/cd_catalog.xml

XML Namespaces - Name Conflicts

- In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

XML carries HTML table information

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

XML carries information about a generic table

- If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning

Solving the Name Conflict Using a Prefix

- Name conflicts in XML can easily be avoided using a name prefix

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

XML Namespaces - The xmlns Attribute

- When using prefixes in XML, a namespace for the prefix must be defined using a xmlns attribute in the start tag of an element

```
<root>
  <h:table xmlns:h="http://www.w3.org/TR/html4/">
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>
  <f:table xmlns:f="https://www.w3schools.com/furniture">
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```