XML

eXtensible Markup Language

XML Motivation

- Huge amounts of unstructured data on the web: HTML documents
  - No structure information
  - Only format instructions (presentation)
- Integration of data from different sources
  - Structural differences
- Closely related to semistructured data

Semistructured Data

- Integration of heterogeneous sources
- Data sources with non rigid structures
  - Biological data
  - Web data
- Need for more structural information than plain text, but less constraints on structure than in relational data

The Idea Behind XML

- Easily support information exchange between applications / computers
- Reuse what worked in HTML
  - Human readable
  - Standard
  - Easy to generate and read
- But allow arbitrary markup
- Uniform language for semistructured data
  - Data Management

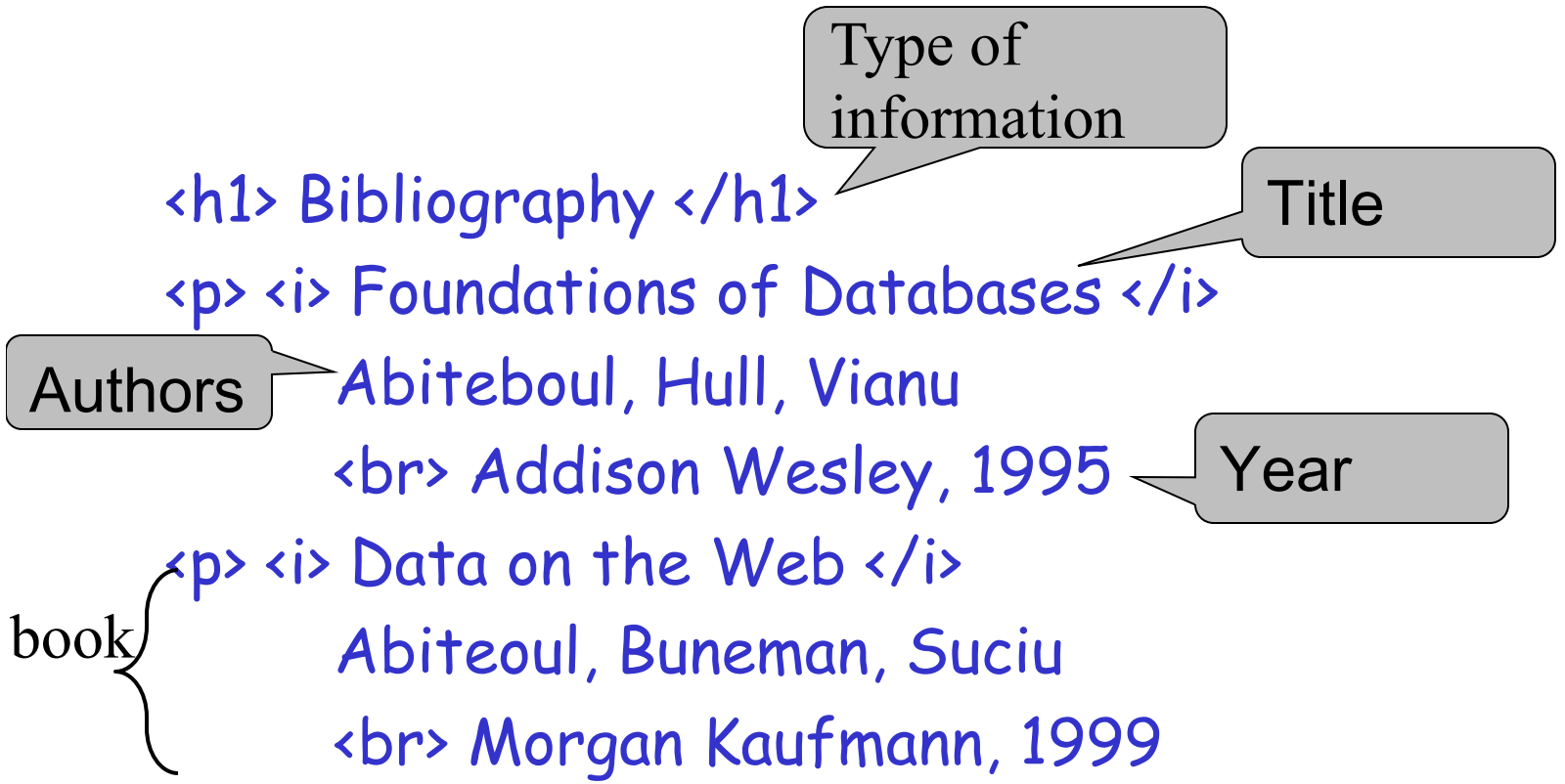
XML

- eXtensible Markup Language
- Universal standard for documents and data
  - Defined by W3C
- Set of technologies
  - XLink, XPointer, XSchema, DOM, SAX, XPath, XQuery, XSL, XSLT, …
- XML gives a syntax, not a semantic
- XML defines the structure of a document, not how it is processed
- Separate structural information from format instructions

Difference between XML and HTML

- XML is not a replacement for HTML.
- XML and HTML were designed with different goals:
  - XML was designed to transport and store data, with focus on what data is
  - HTML was designed to display data, with focus on how data looks
- HTML is about displaying information, while XML is about describing/carrying information.

HTML Document Example



An Address Book as an XML document

```
<addresses>
        <person>
                <name> Donald Duck</name>
                <tel> 414-222-1234 </tel>
                <email> donald@yahoo.com </email>
        </person>
        <person>
                <name> Miki Mouse</name>
                <tel> 123-456-7890 </tel>
                <email>miki@yahoo.com</email>
        </person>
</addresses>
```

Main Features of XML

- No fixed set of tags
  - New tags can be added for new applications
- An agreed upon set of tags can be used in many applications
  - *Namespaces* facilitate uniform and coherent descriptions of data
    - For example, a namespace for address books determines whether to use <tel> or <phone>
- XML has the concept of a schema
  - *DTD* and the more expressive *XML Schema*
- XML is a data model (define how data is connected to each other and how they are processed and stored inside the system.)
  - Similar to the *semistructured data model*
- XML supports internationalization (Unicode) and platform independence (an XML file is just a character file)

XML is the Standard for Data Exchange

- Web services (e.g., ecommerce) require exchanging data between various applications that run on different platforms
- XML (augmented with namespaces) is the preferred syntax for data exchange on the Web

The Structure of XML

- 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 ... ... in HTML)

XML text

XML has only one "basic" type -- text.

It is bounded by tags e.g.

<title> The Big Sleep </title>

<year> 1935 </ year> --- 1935 is still text

XML text is called PCDATA (for parsed character data). It uses a 16-bit encoding.

XML structure

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>
```

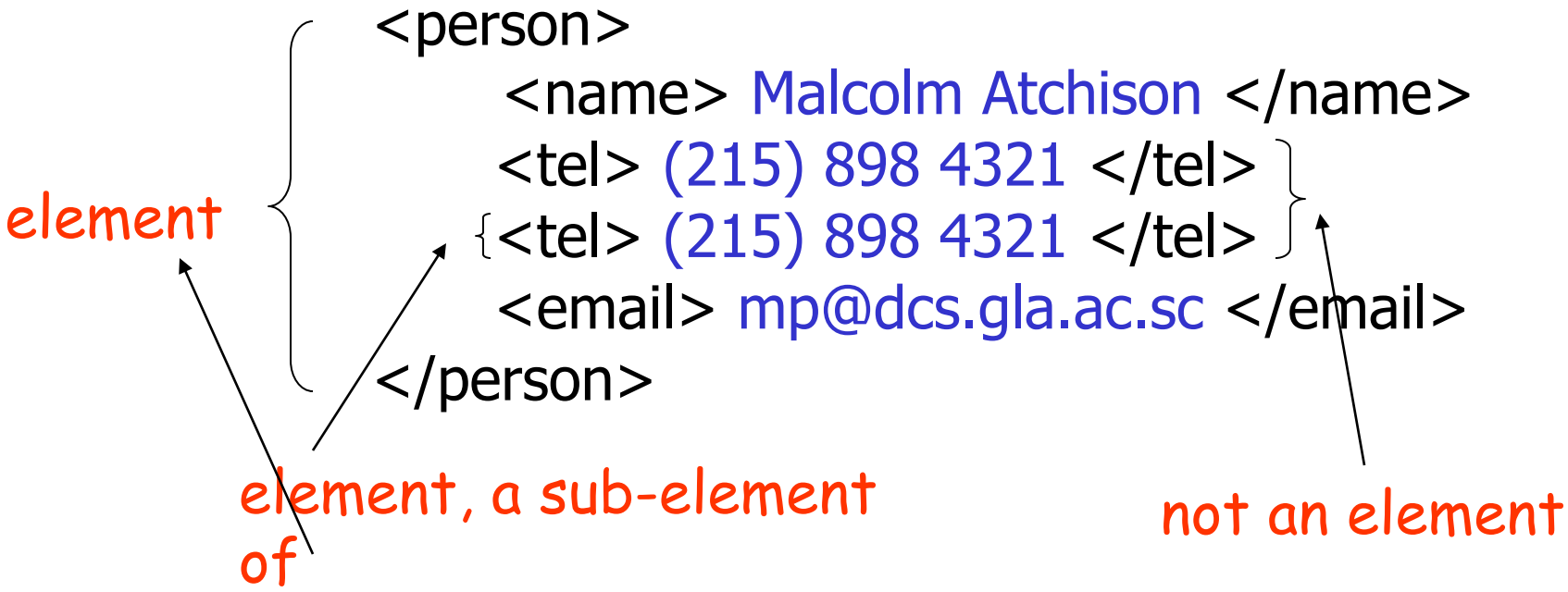
XML structure

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

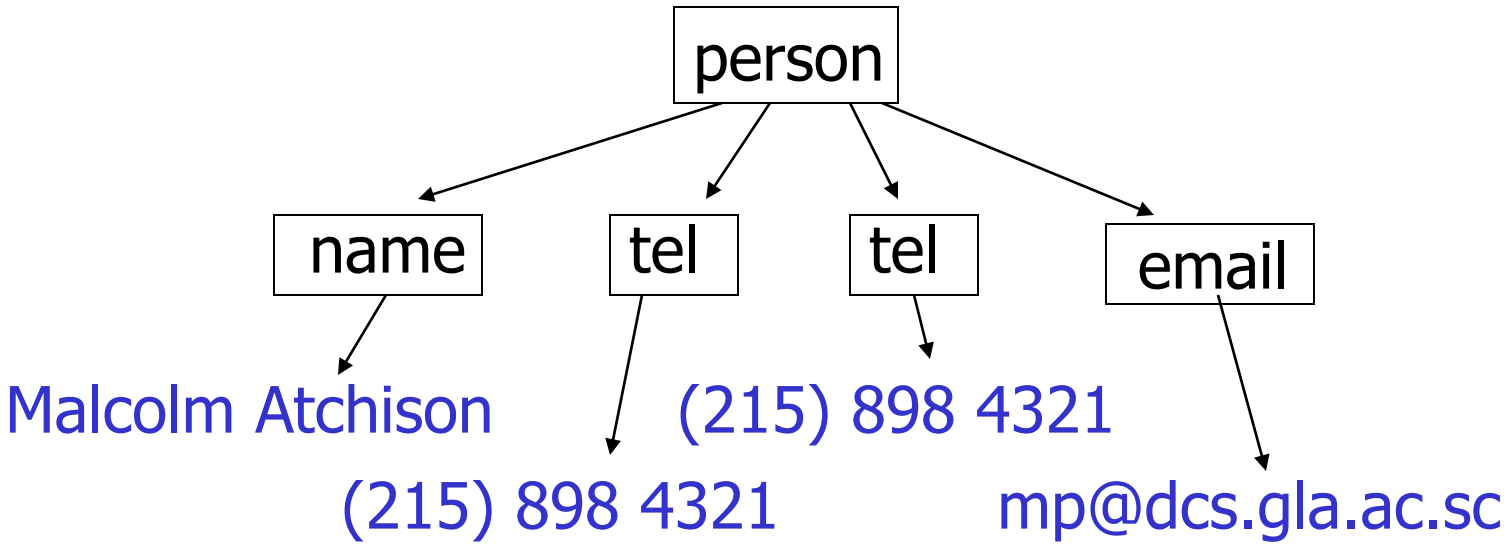
```
<addresses>
   <person> ... </person>
   <person> ... </person>
   <person> ... </person>
   ...
</addresses>
```

Terminology

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



XML is tree-like



Mixed Content

An element may contain a mixture of sub-elements and PCDATA

```
<airline>
    <name> British Airways </name>
    <motto>
    World's <dubious> favorite</dubious> airline
    </motto>
</airline>
```

Data of this form is not typically generated from databases. It is needed for consistency with HTML

A Complete XML Document

```
<?xml version="1.0"?>
<person>
  <name> Malcolm Atchison </name>
  <tel> (215) 898 4321 </tel>
  <email> mp@dcs.gla.ac.sc </email>
</person>
```

Representing relational DBs: Two ways

projects:

title	budget	managedBy

employees:

name	ssn	age

Project and Employee relations in XML

Projects and employees are intermixed

```
<db>
  <project>
    <title> Pattern recognition </title>
    <budget> 10000 </budget>
    <managedBy> Joe </
   managedBy>
  </project>
  <employee>
    <name> Joe </name>
    <ssn> 344556 </ssn>
    <age> 34 < /age>
  </employee>
  <employee>
    <name> Sandra </name>
    <ssn> 2234 </ssn>
    <age> 35 </age>
  </employee>
  <project>
    <title> Auto guided vehicle </title>
    <budget> 70000 </budget>
    <managedBy> Sandra </managedBy>
  </project>
  :
</db>
```

Project and Employee relations in XML (cont' d)

Employees follows projects

```
<db>
   <projects>
     <project>
          <title> Pattern recognition </title>
          <budget> 10000 </budget>
          <managedBy> Joe </managedBy>
     </project>
     <project>
          <title> Auto guided vehicles </title>
          <budget> 70000 </budget>
          <managedBy> Sandra </
   managedBy>
     </project>
    :
   </projects>
```

```
<employees>
  <employee>
     <name> Joe </name>
     <ssn> 344556 </ssn>
     <age> 34 </age>
  </employee>
  <employee>
     <name> Sandra </name>
     <ssn> 2234 </ssn>
     <age>35 </age>
  </employee>
 :
  <employees>
</db>
```

Project and Employee relations in XML (cont' d)

Or without "separator" tags …

```
<db>
  <projects>
    <title> Pattern recognition </title>
    <budget> 10000 </budget>
    <managedBy> Joe </managedBy>
    <title> Auto guided vehicles </title>
    <budget> 70000 </budget>
    <managedBy> Sandra </
   managedBy>
    :
  </projects>
  <employees>
    <name> Joe </name>
    <ssn> 344556 </ssn>
    <age> 34 </age>
    <name> Sandra </name>
    <ssn> 2234 </ssn>
    <age> 35 </age>
    :
  </employees>
</db>
```

Attributes

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>
```

Attributes (cont' d)

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* .

When to use attributes

It's not always clear when to use attributes

```
<person ssno= "123 45 6789">
    <name> F. MacNiel </name>
    <email> fmacn@dcs.barra.ac.sc </email> ...
</person>
```

OR

```
<person>
  <ssno>123 45 6789</ssno>
   <name> F. MacNiel </name>
   <email> fmacn@dcs.barra.ac.sc </email> ...
</person>
```

Using IDs

```
<family>
    <person  id="jane"  mother="mary" father="john">
        <name> Jane Doe </name>
    </person>
    <person id="john" children="jane jack">
        <name> John Doe </name>
    </person>
    <person id="mary" children="jane  jack">
        <name> Mary Doe </name>
    </person>
        <person  id="jack"  mother=”mary" father="john">
        <name> Jack Doe </name>
    </person>
</family>
```

An example

```
<db>
  <movie id="m1">
    <title>Waking Ned Divine</title>
    <director>Kirk Jones III</director>
    <cast idrefs="a1 a3"></cast>
    <budget>100,000</budget>
  </movie>
  <movie id="m2">
    <title>Dragonheart</title>
    <director>Rob Cohen</director>
    <cast idrefs="a2 a9 a21"></cast>
    <budget>110,000</budget>
  </movie>
  <movie id="m3">
    <title>Moondance</title>
    <director>Dagmar Hirtz</director>
    <cast idrefs="a1 a8"></cast>
    <budget>90,000</budget>
  </movie>
  :
  <actor id="a1">
    <name>David Kelly</name>
    <acted_In idrefs="m1 m3 m78" >
    </acted_In>
  </actor>
  <actor id="a2">
     <name>Sean Connery</name>
     <acted_In idrefs="m2 m9 m11">
     </acted_In>
     <age>68</age>
  </actor>
  <actor id="a3">
      <name>Ian Bannen</name>
      <acted_In idrefs="m1 m35">
      </acted_In>
  </actor>
   :
</db>
```

Summary - XML Data Components

XML includes two kinds of data items:

Elements

```
<article mdate="2002-01-03" …>
    <editor>Paul R. McJones</editor>…
</article>
```

- Hierarchical structure with open tag-close tag pairs
- May include nested elements
- May include attributes within the element's open-tag
- Multiple elements may have same name
- Order matters

Attributes

```
mdate="2002-01-03"
```

- Named values – not hierarchical
- Only one attribute with a given name per element
- Order does NOT matter

Combining XML from Multiple Sources with the Same Tags: Namespaces

- *Namespaces* allow us to specify a context for different tags – (1) resolving name conflict, (2) group elements relating to a common idea together
- The namespace can be defined by an xmlns attribute in the start tag of an element – working like "link the following letters to a URI"
- The namespace declaration has the following syntax -

  xmlns:*qualifier*="URI" with two parts:
  - Binding of namespace to URI
  - Qualified names

Default namespace for non-qualified names

```
<root xmlns="http://www.first.com/aspace" xmlns:otherns="..">
 <myns:tag xmlns:myns="http://www.fictitious.com/mypath">
    <thistag>is in the default namespace
                (www.first.com/aspace)</thistag>
    <myns:thistag>is in myns</myns:thistag>
    <otherns:thistag>is a different tag in otherns</otherns:thistag>
 </myns:tag>
</root>
```

Defines "otherns" qualifier

Combining XML from Multiple Sources with the Same Tags: Namespaces

Note:

(1) The namespace URI is not used by the parser to look up information.
(2) The purpose of using an URI is to give the namespace a unique name.
(3) However, companies often use the namespace as a pointer to a web page containing namespace information.