XML

eXtensible Markup Language

Xin Chen - xinchen@zju.edu.cn

Semester 2, 2018/19

LAST TIME

- **function** - `call`/`scope`…
- **class** - `attributes`/`method`
- **algorithms** - `characteristic`/`pseudocode`
- **big oh notation** - `orders of growth`
- **evaluate efficiency of programs** - `bubble sort`/`merge sort`

TODAY

- **xml concepts** - `extensible markup language`
- **xml applications** - `data interchange…`
- **xml structure** - `prolog`/`root`/`element`/`tags`/`attributes…`
- **DTD** - `document type definition`
- **GO** - `gene ontology`

GAINING ACCESS TO DIVERSE DATA

- if we focused on data integration in the relational model

  Simplest model to understand

A	B
a_1	b_1
a_2	b_2

- but real-world data is often not in relational form

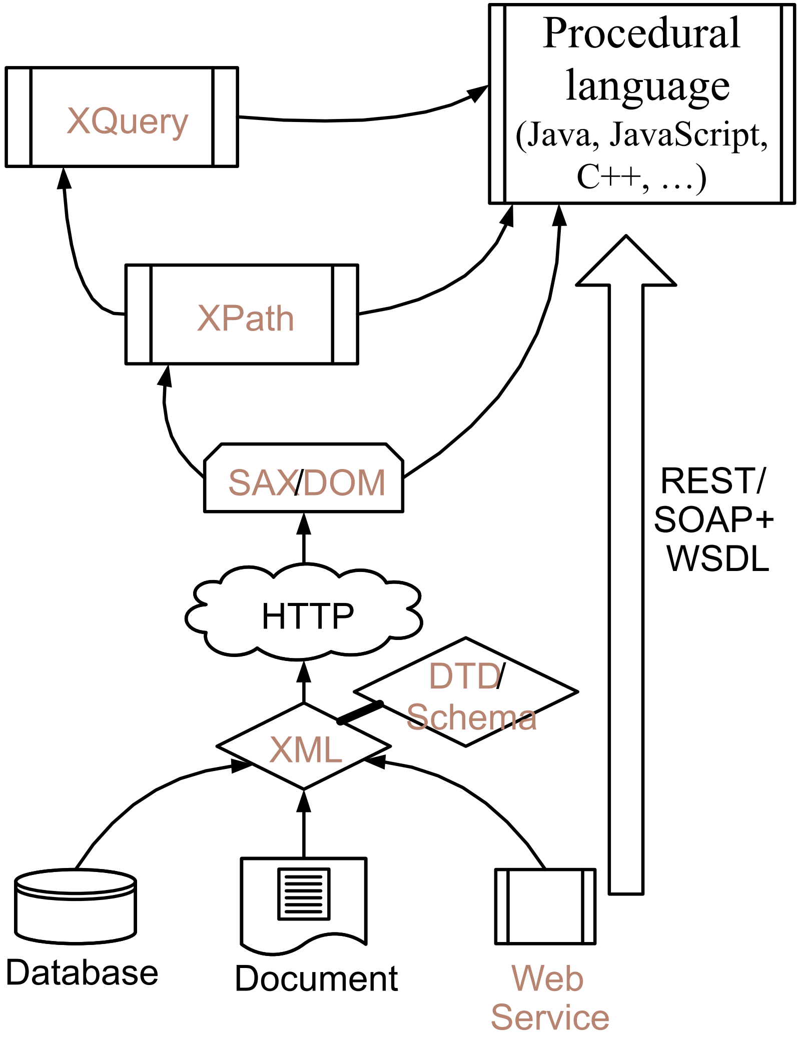
  e.g., Excel spreadsheets, Web tables, Java objects, RDF, …

  - One approach: convert using custom wrappers
  - But imagine tools would adopt a standard export (and import) mechanism?

… This is the role of XML, the eXtensible Markup Language

WHAT IS XML

- Hierarchical, human-readable format
  - A “sibling” to HTML, always parsable
  - “Lingua franca” of data: encodes documents and structured data
  - Blends data and schema (structure)
- Core of a broader ecosystem
  - Data： XML
  - Schema： DTD and XML Schema
  - Programmatic access： DOM and SAX
  - Query： XPath, XSLT, XQuery
  - Distributed programs： Web services



WHAT IS XML

- Extensible Markup Language (XML) is:

– a World Wide Web Consortium (W3C) standard for

– a file format to

– easily and cheaply distribute electronic documents on

the World Wide Web

– extensible, not frozen like HTML

– supporting rich structure, like objects or hierarchies or relationships

– supporting validation and well-formed properties

– avoiding applets, scripts, plug-ins, etc.

– separating form (how it looks) from content (what it is)

XML APPLICATIONS

- XML applications
  - Data **interchange** format between computers
    - Using Web server as data channel between databases
    - Automated processing of documents exchanged
  - **Common format** for Web, electronic, paper documents, ...
    - XML as a general markup language
    - XML used for manuals, CDs, help and other text documents
    - Handled by standard browsers (IE, Firefox, Chrome, ...)
  - Remote procedure call/invocation protocol
    - Executes Web services or processes on other computers

EXAMPLE

- Pre-XML representation of data:

```
“BOOK”,”author: Hull”,
"title:California","year:1995"
```

- XML representation of the same data:

```
<BOOK>
   <author>Hull</author>
   <title>California</title>
   <year> 1995 </year>
</BOOK>
```

SYNTAX AND STRUCTURE

```
<?xml version=“1.0” ?>

<BOOKS>

<book id=“123” loc=“library”>

   <author>Hull</author>

   <title>California</title>

   <year> 1995 </year>

</book>

</BOOKS>
```

Prologue
(processing instructions)

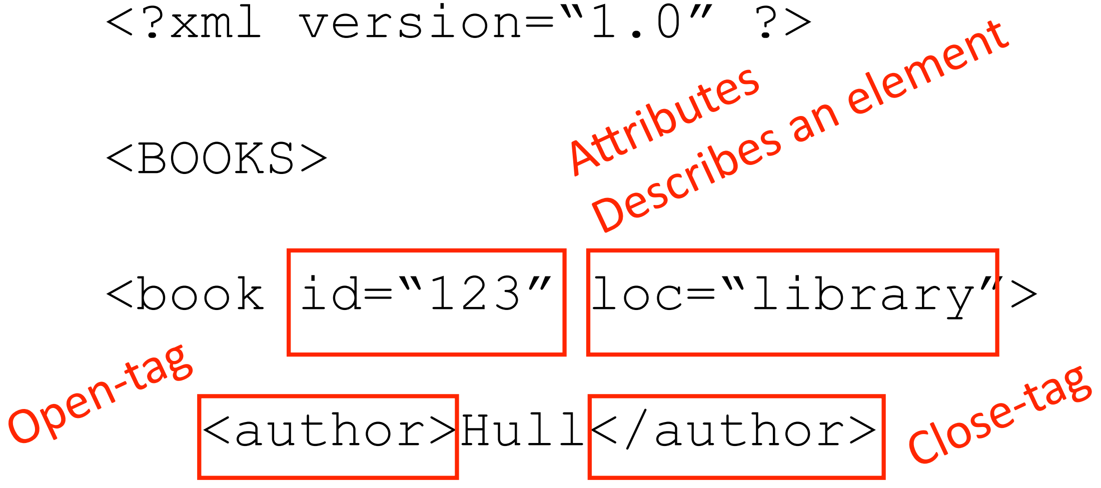
root

Elements with
Attributes

Elements

Elements can be empty
(<TAG_NAME />)

SYNTAX AND STRUCTURE



```
<?xml version=“1.0” ?>

<BOOKS>

<book id=“123” loc=“library”>

   <author>Hull</author>

   <title>California</title>

   <year> 1995 </year>

</book>

</BOOKS>
```

Attributes

Describes an element

Open-tag

Close-tag

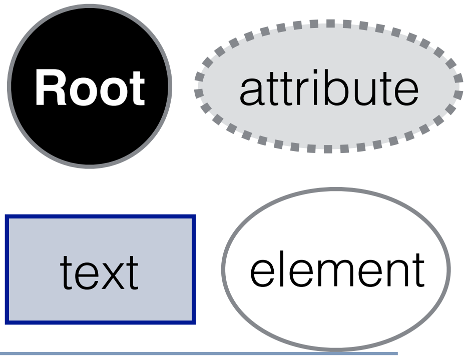
SYNTAX AND STRUCTURE

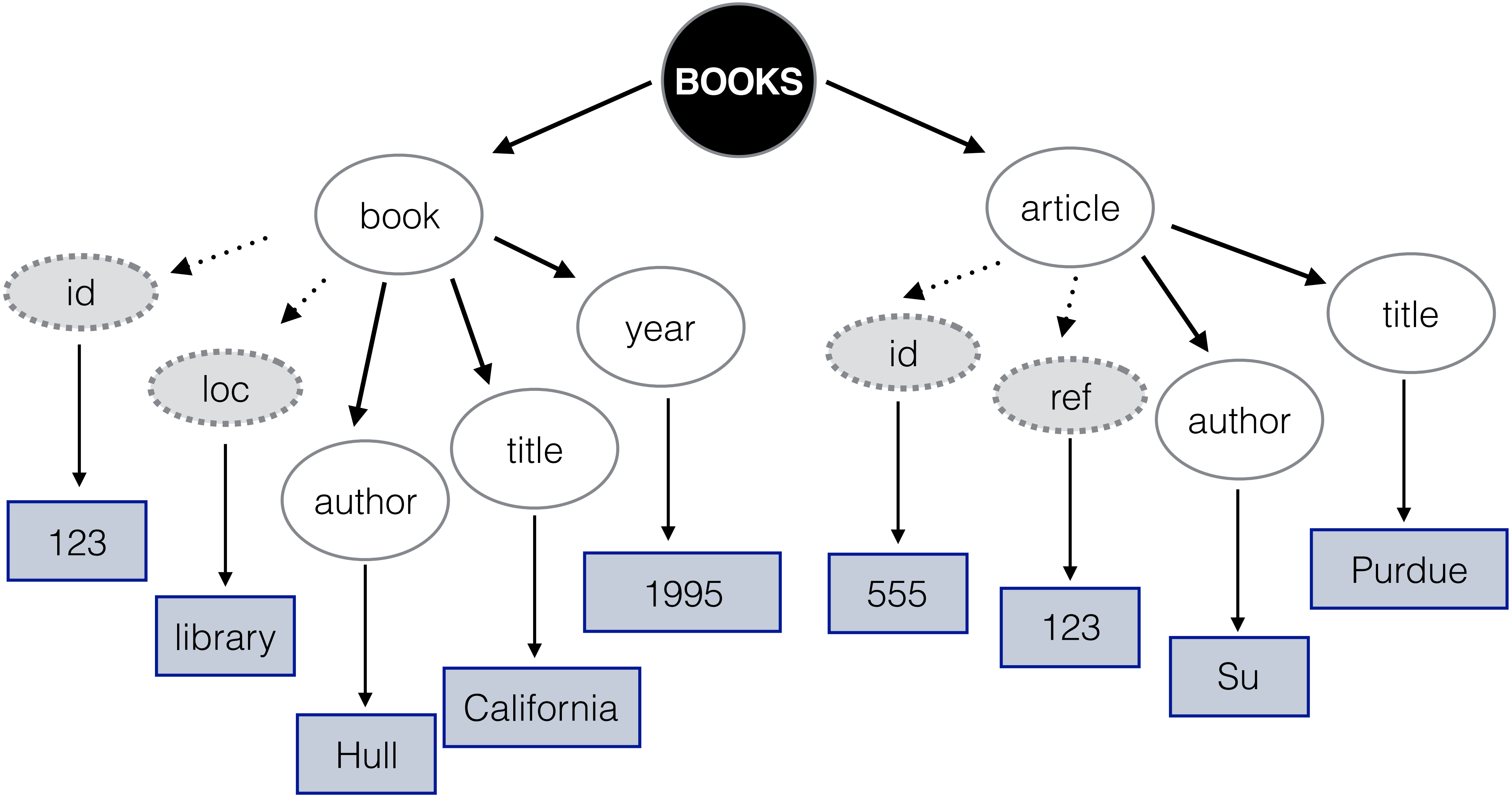
```
<?xml version=“1.0” ?>
<BOOKS>
<book id=“123” loc=“library”>
   <author>Hull</author>
   <title>California</title>
   <year> 1995 </year>
</book>
<article id=“555” ref=“123”>
   <author>Su</author>
   <title> Purdue</title>
</article>
</BOOKS>
```

eLement 1:
book

eLement 2:
article

DATA VISUALIZED





RULES

- There must be **one**, and only one, root element
- Sub-elements must be properly **nested**
  - A tag must end within the tag in which it was started
- Attributes are optional
  - Defined by an optional schema
- Attribute values must be enclosed in **“”** or **‘’**
- Processing instructions are optional
- XML is **case-sensitive**
  - `<tag>` and `<TAG>` are not the same type of element

WELL FORMED XML?

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

<PARENT>

  <CHILD1>This is element 1</CHILD1>

  <CHILD2><CHILD3>Number 3</CHILD2></CHILD3>

</PARENT>
```

✗

- No, CHILD2 and CHILD3 do not nest properly

WELL FORMED XML?

```
<xml? Version=“1.0” ?>

<PARENT>

   <CHILD1>This is element 1</CHILD1>

</PARENT>

<PARENT>

   <CHILD1>This is another element 1</CHILD1>

</PARENT>
```

- No, there are two root elements

WELL FORMED XML?

```
<xml? Version=“1.0” ?>

<PARENT>

  <CHILD1>This is element 1</CHILD1>

  <CHILD2/>

  <CHILD3></CHILD3>

</PARENT>
```

Empty element

✓

- Yes

COMBINE XML
WITH THE SAME TAGS

- What if two different xml documents have the same tags?

table.xml

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

furniture.xml

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

USING PREFIX

- Use **prefixes** to avoid conflicts

table.xml

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

furniture.xml

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

XML NAMESPACES

- **Namespaces** allow authors to differentiate between tags of the same name

```
<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>
```

An attribute named ‘xmlns’

table.xml

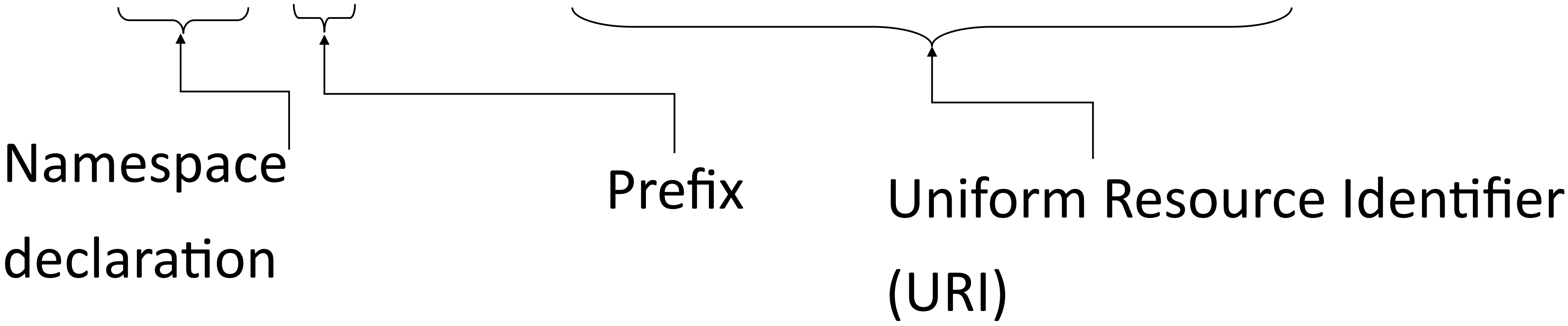
XML NAMESPACES

- Namespace declaration examples:

```
xmlns: bk = “http://www.example.com/bookinfo/”

xmlns: bk = “urn:mybookstuff.org:bookinfo”

xmlns: bk = “http://www.example.com/bookinfo/”
```



Namespace declaration

Prefix

Uniform Resource Identifier (URI)

DEFAULT NAMESPACES

- An XML namespace declared without a prefix becomes the **default namespace** for all sub-elements
- All elements without a prefix will belong to the default namespace:

```
<BOOK xmlns=“http://www.bookstuff.org/bookinfo”>
   <TITLE>All About XML</TITLE>
   <AUTHOR>Joe Developer</AUTHOR>
```

XML IS NOT ENOUGH ON ITS ON

- It's too unconstrained for many cases!
  - How will we know when we're getting garbage?
  - How will we know what to query for?
  - How will we understand what we have received?

- We also need:
  - An idea of (at least part of) the structure
  - Some knowledge of how to interpret the tags…

DOCUMENT TYPE DEFINITIONS (*DTDs*)

- The DTD is an EBNF grammar **defining XML structure**
  - The XML document specifies an associated DTD, plus the root element of the document
  - DTD specifies children of the root (and so on)

- Advantages for DTDs:
  - A single DTD ensures a common format for each XML document that **references** it
  - A description of legal, valid data further contributes to the interoperability and efficiency of using XML

DTD CONTENT

- **DOCTYPE**: class (type) of document
  - Placed in XML file, refers to DTD file to be used to validate
- **ELEMENT**: object in document
  - Either all valid values are given in a list in (), or
  - The element is defined later in the DTD file
  - Symbols: +: 1 or more, *: 0 or more, ?:0 or 1, none: exactly 1
- **ATTLIST**: valid attribute list for element
  - #CDATA: character data
  - #PCDATA: parsed character data (can't have < > &...)
  - #REQUIRED: element must be present
  - #IMPLIED: element optional, no default value
  - #FIXED: attribute value is fixed

DTD EXAMPLE

- Example DTD: (written in xml)

```
<?xml version=“1.0"?>
<!DOCTYPE BOOK [
<!ELEMENT BOOK (author,title,year)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
]>
<BOOK>
   <author>Hull</author>
   <title>California</title>
   <year> 1995 </year>
</BOOK>
```

DTD

DTD EXAMPLE

- Example DTD: (not written in xml)

```
<?xml version=“1.0"?>
<!DOCTYPE BOOK SYSTEM "BOOK.dtd">
<BOOK>
   <author>Hull</author>
   <title>California</title>
   <year> 1995 </year>
</BOOK>
```

DTD declaration

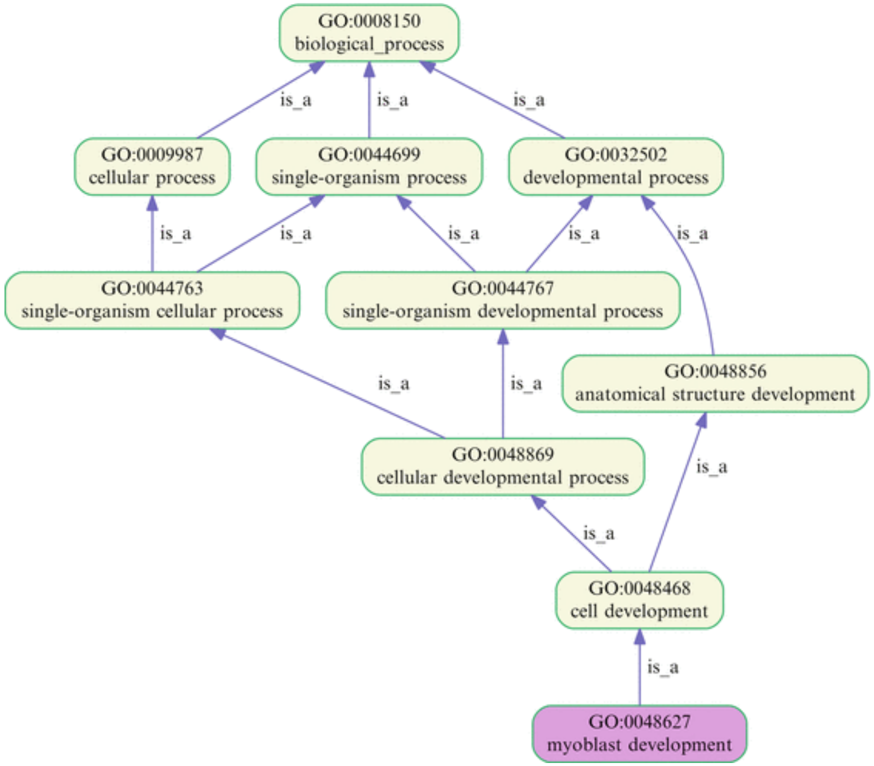
BOOK.xml

```
<!ELEMENT BOOK (author,title,year)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

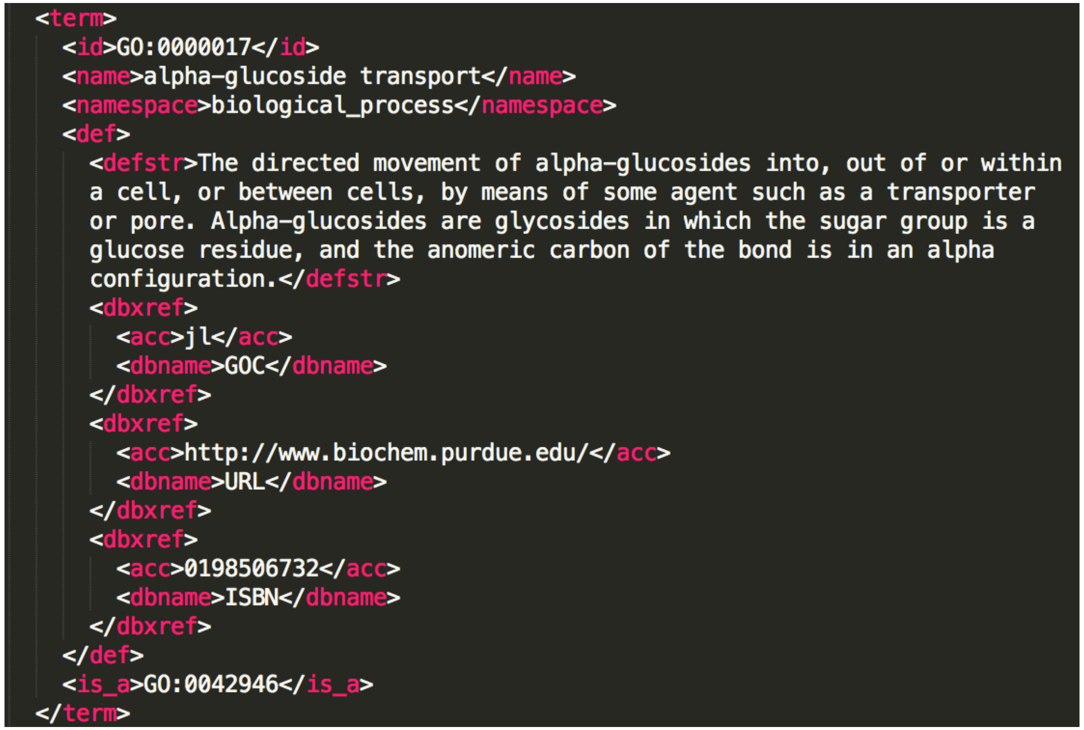
BOOK.dtd

EXAMPLE: GO DATABASE

Human-readable



Computer-readable



```
<term>
  <id>GO:0000017</id>
  <name>alpha-glucoside transport</name>
  <namespace>biological_process</namespace>
  <def>
    <defstr>The directed movement of alpha-glucosides into, out of or within
    a cell, or between cells, by means of some agent such as a transporter
    or pore. Alpha-glucosides are glycosides in which the sugar group is a
    glucose residue, and the anomeric carbon of the bond is in an alpha
    configuration.</defstr>
    <dbxref>
      <acc>jl</acc>
      <dbname>GOC</dbname>
    </dbxref>
    <dbxref>
      <acc>http://www.biochem.purdue.edu/</acc>
      <dbname>URL</dbname>
    </dbxref>
    <dbxref>
      <acc>0198506732</acc>
      <dbname>ISBN</dbname>
    </dbxref>
  </def>
  <is_a>GO:0042946</is_a>
</term>
```

SUMMARY

- XML documents can be:
  - – Defined by anyone: tags and business rules
  - – Sent and received by databases using SQL and HTTP
  - – Validated by DTD or XSD files
  - – Transformed and styled by XSLT files
  - – Placed on a server for clients to attach to, such as blogs
- XML files are little pieces of a database that can be shared.

Typically they represent:

  - – Rows from a single table, or
  - – Rows from two tables in a many-to-one relationship
  - – Any arbitrary set of tables/relationships can be sent