Scientific Data Formats

IST400/600

Jian Qin

Review of data concepts

Scientific data formats

Metaformats

Data structures

Physical data

Common Data Formats
Image formats
Matrix formats
Microarray file formats
Communication protocols

Data model

Hierarchical
Relational
Object-oriented
Network

DSV CSV XML

Tuple Set List Array Tree

Bits…Bytes…characters…strings
Extensible Markup Language (XML)

Let’s look at this familiar code...

HTML Syntax

```html
<H1 ALIGN="center">The Heading Text</H1>
```

Telling computer how to present data in a browser
Components

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!--This is a very simple example -->
<wines>
  <wine grape="chardonnay">
    <winery>Benziger</winery>
    <product>Carneros</product>
    <year>1997</year>
    <desc>Well-textured flavors, good finish.</desc>
    <prices>
      <list>10.99</list>
      <discounted>9.50</discounted>
      <case>114.00</case>
    </prices>
  </wine>
</wines>
```

Tags

- Case sensitive
  `<wines>` is different from `<Wines>`
- Descriptive
  `<customerLastName>` or `<customer_last_name>` compared to `<custLName>`
- Use lower-case to begin and upper-case to begin later words in the tag
  `<journalPublisherAddress>`
- Beginning and closing tags
  `<title>Forrest Gump</title>`
- No blank space in tag labels
  `<addressState>` not `<address State>`
Tag examples

- Some of the following examples are correct and some are not. Can you tell which are correct or incorrect?

```xml
<books>...</Books>
<ISBN />
<author>...</author>
<title>
<price$/>
</publisher>
<pubDate>...</pubDate>
```

Data formats

---

Structuring XML

- Single root element
- Tree structure
  - Branches (also called parents)
  - Leaves (also called children)
  - Nested

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wine SYSTEM "wine.dtd">
<wines>
  <wine grape="Chardonnay">
    <winery>Benziger</winery>
    <product>Carneros</product>
    <year>1997</year>
    <desc>Well-textured flavors, good finish.</desc>
    <prices>
      <list>10.99</list>
      <discounted>9.50</discounted>
      <case>114.00</case>
    </prices>
  </wine>
  <wine grape="Cabernet">
    <winery>Benziger</winery>
    <product>Cabernet Sauvignon</product>
    <year>2001</year>
    <desc>Turns rich and dry in the mouth, classically structured, with firm tannins and a finish of pure black currants and sweet oak.</desc>
    <prices>
      <list>35.99</list>
      <discounted>32.50</discounted>
      <case>250.00</case>
    </prices>
  </wine>
</wines>
```
XML elements are extensible

<note>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>

<note>
  <date>2002-08-01</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>

(source: XML elements from http://www.w3schools.com/xml/xml_elements.asp)

Data formats

XML elements have relationships

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

(source: XML elements from http://www.w3schools.com/xml/xml_elements.asp)
The need for DTD

- What is a DTD?
  - Document Type Definition

- Why do we need a DTD?
  - Consistency in data and document structure
    • (Why is consistency important?)
  - Well-thought-out relationships between data elements rather than ad hoc relationships
  - Tool development
  - Interoperability

Document Type Definition (1)

- Contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or DTD.
  - Markup declaration includes:
    • element-type declaration <!ELEMENT ...>
    • attribute-list declaration <!ATTLIST ...>
    • entity declaration <!ENTITY ...>
    • notation declaration <!NOTATION...>
  - These declarations may be contained in whole or in part within parameter entities
Document Type Definition (2)

- An optional part of the document prolog, as specified by the XML standard.
- Specifies constraints on the valid tags and tag sequences that can be in the document.

**Internal subset:** defined in the current file

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)> ]>
<greeting>Hello, world!</greeting>
```

**External subset:** consists of the definitions contained in external .dtd files that are referenced in the local subset using a parameter entity

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

Element Declarations (1)

```xml
<?xml version='1.0' encoding='UTF-8'?>
<!-- DTD for a simple "slide show". -->
<!ELEMENT slideshow (slide+)>  
<!ELEMENT slide (title, item*)>    
<!ELEMENT title (#PCDATA)>       
<!ELEMENT item (#PCDATA | item)* >
```

**Qualifier** | **Name** | **Meaning**
--- | --- | ---
? | Question Mark | Optional (zero or one)
* | Asterisk | Zero or more
+ | Plus Sign | One or more
Element Declarations (2)

- slideshow (slide+):
The slideshow element may have one or more slide sub-elements

- #PCDATA:
Parsed Character Data, i.e., plain text data. “#” means that what follows is a special word, not an element name.

- |
Vertical bar, indicating an OR condition. In this case, either PCDATA or an item can occur.

Creating a DTD (1)

- Start by creating a file named slideshow.dtd.
- Enter an XML declaration and a comment to identify the file, as shown below:

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

<!-- DTD for a simple "slide show". -->
```
Creating a DTD (2)

• Add the text highlighted below to specify that a slideshow element contains slide elements and nothing else:

<!-- DTD for a simple "slide show". -->

<!ELEMENT slideshow (slide+)>
Creating a DTD (4)

• Defining Attributes in the DTD:

```xml
<!ELEMENT slideshow (slide+)>
<!ATTLIST slideshow
  title CDATA #REQUIRED
  date CDATA #IMPLIED
  author CDATA "unknown"
>
<!ELEMENT slide (title, item*)>
<!ATTLIST slide
  type (tech | exec | all) #IMPLIED
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT item (#PCDATA | item)* >
```

Creating a DTD (6)

The complete DTD for slide show

```xml
<?xml version='1.0' encoding='us-ascii'?>
<!-- ***DTD for a simple "slide show".***-->
<!ELEMENT slideshow (slide+)>
<!ATTLIST slideshow
  title CDATA #REQUIRED
  date CDATA #IMPLIED
  author CDATA "unknown"
>
<!ELEMENT slide (image?, title, item*)>
<!ATTLIST slide
  type (tech | exec | all) #IMPLIED
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT item (#PCDATA | item)* >
<!ELEMENT image EMPTY>
<!ATTLIST image
  alt CDATA #IMPLIED
  src CDATA #REQUIRED
  type CDATA "image/gif"
>
```
What’s wrong with DTDs?

- DTDs are good for narrative-style documents, however, they are:
  - Unable to define data types used in data-centric documents
  - Not in XML syntax
  - Only marginally extensible, low scalability
  - Inflexibility to use namespace

What is a schema?

- Schema:
  A collection of statements that define the structure of a set of data, as well as constraints on that structure

- Schema language:
  A language used to define a schema, such as the XML DTD language or an XML schema language

The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.
Namespaces

• What is it?
  - A collection of names, identified by a URI reference.
    • Has structure
    • A DTD can be a namespace
      http://www.myserver.com/xml/catalog.dtd

BookList document

```xml
<booklist>
  <book ISBN="1-558-28592-x" availability="instock">
    <title>XML: A Primer</title>
    <price>24.99</price>
    <author>
      <name>Simon St. Laurent</name>
      <contactinfo>
        <email>simonstl@simonstl.com</email>
        <website>http://www.simonstl.com</website>
      </contactinfo>
    </author>
  </book>
  <book ISBN="0-130-81152-1" availability="instock">
    <title>The Xml Handbook</title>
    <price>44.95</price>
    <author>
      <name>Charles F. Goldfarb</name>
      <name>Paul Prescod</name>
    </author>
  </book>
</booklist>
```
BookList DTD

<!ELEMENT bookList (book*)>
<!ELEMENT book (title, price, author+)>
<!ATTLIST book
  ISBN CDATA #REQUIRED
  bookAvailability (InStock | OutOfStock) "InStock">
<!ELEMENT title (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT author (name, contactInfo?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT contactInfo (address?, telephone*, email*, webSite*)>
<!ELEMENT address (street, city, country?, postalCode)>
...

BookList Schema

<x:schema version="1.0" name="http://www.booklist/booklist.xs">
  <xs:complexType name="Book">
    <xs:sequence>
      <xs:element name="Title" maxOccur="1"/>
      <xs:element name="Price" maxOccur="1"/>
      <xs:element name="Author" minOccurs="1" maxOccurs="*"/>
    </xs:sequence>
    <xs:attribute name="ISBN" use="required" type="xs:string"/>
    <xs:attribute name="bookAvailability" use="required" type="availability"/>
  </xs:complexType>
  <xs:simpleType name="availability">
    <xs:restriction base="xs:string">
      <xs:enumeration value="InStock"/>
      <xs:enumeration value="OutOfStock"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
Element declarations (1)

<?xml version="1.0"?>
<xs:schema>
  ...
  ...
</xs:schema>

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

Elements and data types from W3C

Elements come from the default namespace

Any element used by the XML instance document which were declared in this schema must be namespace qualified.

Element declarations (2)

Standard schema language

Name to be used in XML document

Data type

<xs:element name="dateReceived" type="xs:date"
  minOccurs="0" maxOccurs="*" />

Cardinality

Or

<xs:element name="dateReceived" type="xs:date"
  minOccurs="0" maxOccurs="unbounded" />

View the wine schema file
Review of data concepts

Scientific data formats

Metaformats

Data structures

Physical data

Data model

Hierarchical

Relational

Object-oriented

Network

Common data formats
Image formats
Matrix formats
Microarray file formats
Communication protocols

DSV
CSV
XML

Tuple Set List Array Tree

Bits...Bytes...characters...strings

Scientific data formats

• Major formats:
  - Common Data Format (CDF)
  - Flexible Image Transport System (FITS)
  - GRid In Binary (GRIB)
  - Hierarchical Data Format (HDF)
  - Network Common Data Format (netCDF)
Common Data Format (CDF)

- A conceptual data abstraction for storing, manipulating, and accessing multidimensional data sets
- Basic component: a software programming interface that is a device-independent view of the CDF data model
- Current version: 3.2 as of October 12, 2007
- Used by hundreds of government agencies, universities, and private and commercial organizations as well as independent researchers on both national and international levels
- CDF has been adopted by the International Solar-Terrestrial Physics (ISTP) project as well as the Central Data Handling Facilities (CDHF) as their format of choice for storing and distributing key parameter data.
Flexible Image Transport System (FITS)

- The standard data format used in astronomy
- Endorsed by NASA and the International Astronomical Union
- Much more than just another image format (such as JPEG or GIF)
- Used for the transport, analysis, and archival storage of scientific data sets
  - Multi-dimensional arrays: 1D spectra, 2D images, 3D+ data cubes
  - Tables containing rows and columns of information
  - Header keywords provide descriptive information about the data

Sample FITS file:
http://fits.gsfc.nasa.gov/cgi-bin/browse?file=/samples/WFPC2u5780205r_c0fx.fits

Hierarchical Data Format (HDF)

- Self-describing
  - Comprehensive information about the data and its location in the file (metadata)
- Multiple data types
  - Symbolic
  - Numerical
  - Graphical

HDF data structures

The HDF Group. (2007). HDF4 user's guide. [Link]

HDF file format

Scientific data set content

The HDF Group. (2007). HDF4 user's guide. [Link]
Network Common Data Format (netCDF)

- A set of interfaces for array-oriented data access and a freely-distributed collection of data access libraries for C, Fortran, C++, Java and other languages.
- Data structure that supports the creation, access, and sharing of scientific data:
  - interfaces
  - Libraries (of programming codes)
  - formats

Network Common Data Format (netCDF)

- Self-Describing
- Portable
- Direct-access
- Appendable
- Sharable
- Archivable
Development of data formats

- XML-based scientific data formats
  - CDF Markup Language (CDFML)
    [CDFML](http://cdf.gsfc.nasa.gov/html/cdf_xml.html)
    - DCFML DTD
    - CDFML Schema
  - HDF5 [HDF5](http://hdf.ncsa.uiuc.edu/HDF5/XML/)
    - HDF5 DTD
    - HDF5 Schema
Summary

• Scientific data formats are closely tied to software tools
  - Each format is facilitated by a software library
  - Runs on different platforms
  - Represents very complex data objects and a wide variety of metadata
  - Offers tools and applications for managing, manipulating, viewing, and analyzing the data in the collection

• Scientific data formats contain metadata, structure, and data
  - Self-describing

• XML is increasingly becoming the encoding language for scientific data and metadata