Example 2. A more complex document system (World Bank)

A frame/object hierarchy of document templates and documents

A document template is a frame with a slot for each part of the document (a part can be a single line or part of a line). Many slots have a procedure attached; the procedure obtains the information from a database, if it is available, or displays a menu of possible values, or asks the user a question. The document templates are arranged in a hierarchy, so that the slots in common to all documents of a class, such as meeting announcements, need to be specified only once; these slots than inherit down to all descendants of the class.

Figure WB1. **Frame/object hierarchy of document templates and documents**

Template for generic meeting announcement (Figure WB2)

- Template for announcement of any meeting of group A
  - Announcement of a specific meeting of the group (Figure WB3)
- Template for announcement of any meeting of group B
- etc.

Template for generic meeting agenda

- Template for the agenda of a meeting of a specific group (Figure WB4)

Template for an internal report

- Template for a progress report
  - Template for a market study progress report
- Template for market study report
  - Template for a market study progress report
- Template for a technical development report

Note: This figure presents the "deep structure", the slots with a brief description of their attached procedures. The next figure presents an example with filled-in values.

Figure WB2. **Meeting announcement template**

Author: Fill in on sign-on (*Most other instructions are to carried out by the system*)

Parent document (class hierarchy): Announcement

Child document (class hierarchy): Country Review Committee meeting announcement

Group that is meeting: Possible values: Get list of groups for which the author calls meetings

Note: "Get" is an instruction to the system to get this information from database. A complete template includes the database query to be used. Possible values are automatically displayed in a pop-up menu with multiple selection.

Business function served: Possible values: Get list of business functions in which this group is involved.

Note: The meeting must deal with one or more of these business functions

Purpose: Possible values: Get list of purposes served by this group

Receiver: Possible values: Get list of group members, of regular guests, and of others who should receive agendas of this group's meetings

Date and time: Fixed date and time (on the 15th of each month) or relative date and time (e.g., every second Tuesday at 10 am) or

Get schedules of all participants and determine the date(s) and time(s) at which everybody can make it.

Note: System fills in time or presents pop-up menu if there are several possibilities.

Room: Default room. If not available, select a big enough room that is available at the meeting time from room scheduling database.

Deadline:

Time needed for preparation:

Starting date for preparation:

Status:

Includes document: Agenda

Attached document:

Figure WB3. **Country Review Committee meeting announcement — template filled in**

Author: R. Singh (*in this more specific template, filled in by system on sign-on*)

Parent document (class hierarchy): Meeting announcement

Child document (class hierarchy): Specific Country Review Committee meeting announcements

Group that is meeting:

Country Review Committee
(*filled in by system since this is the only group for which R. Singh calls meetings*)

Business function served:

(*System retrieves business functions served by this group from database and displays pop-up menu. Author selects the business function(s) applicable to this meeting.)*

2.1 Develop country operations strategy

2.3 Approve projects

2.4 Supervise projects through completion

Purpose:

Receiver:

Members: R. Singh, B. Smith, J. Dubois
Guests: D. Suarez
(*All filled in by system from database information*)

Date and time:

Monday, November 30, 1992, 10 am
(*Determined by system based on schedules of participants and general instruction: End of every month*.)

Room: F1057 (*Determined by system*)

(continued on next page)

Figure WB3. **Country Review Committee meeting announcement — continued**

Deadline:

Monday, November 2, 1992
(4 weeks before meeting date)

Time needed for preparation:

3 days (elapsed time)

Starting date for preparation:

Thursday, October 26, 1992

Status:

In process

Includes document:

Agenda for Country Review Committee

Attached document:

Determined based on agenda

Figure WB4. **Agenda for Country Review Committee — template**

Information needed:

Status of country operations strategy

From: Country desk

If decisions needed and all necessary documents are ready, put on agenda

Documents needed for deliberation (attachments to meeting announcement)

Status of projects in the appraisal process

From: Project management database

Get projects for which the appraisal is completed. Put on agenda

Appraisal report as attachment

Status of operating projects

From: Project management database

Get projects for which a review is due. Put on agenda

Project progress report as attachment

Note: Once the agenda is complete, it can be used to automatically generate deadlines for documents needed at the meeting (or a specified time prior to the meeting) and send appropriate messages to the authors of these documents. Such a message, in conjunction with the template for the requested document, can in turn be used to automatically update the work plan of the recipient.

Lecture 12b (for information only)
(You may want to look at this when you take 690 to see the connection)

Formatting documents for interpretation by computer programs. Document markup languages

HTML (Hypertext Markup Language) / XML (eXtensible Markup Language) and

SGML (Standard Generalized Markup Language)

Objectives

1 The student should understand the principles of markup languages and their importance for the implementation of good document design as a basis of further study.

2 Students should be able to create simple Web pages using HTML markup.

Practical significance

Databases of machine-readable text are undergoing an unprecedented explosion, not only on the Web, but also in intranets and in efforts of creating large corpora of text for linguistic and literary studies (the Text Encoding Initiative). Conventions for marking the structure of documents are a prerequisite for creating such databases and for common access and data exchange. Most students will find themselves in situations where they need to access such texts and assist users in the further processing such texts; some students might participate in the setup of text databases.

Furthermore, markup languages are now expanded to specify any kind of data structure, blurring the boundary between text and formatted data.

Definition

Markup is the insertion of tags (codes) into a document text or other data stream to specify a structure which can then be used for further processing, in particular for controlling the appearance (or rendering) of a document when it is printed or displayed on a screen.

Note: The term *markup* derives from typesetting. An editor put marks in a manuscript that specified for the typesetter the fonts to be used for a portion of text and other matters of appearance. The meaning of the term has much expanded since then, particularly in the last few years.

General introduction

HTML markup tags are designed primarily to allow the display of documents. HTML tags also control links to other documents to be included automatically at display time (such as images) or available to the user by clicking on the link symbol. Tags defined through XML are much more powerful for expressing document and data structure.

Principles

Physical markup. Tags specify actual appearance properties, such as

Indent .3", *Center*, *Bold*, *Font Times Roman 12*

Problem: What if display device cannot show Times Roman?

Logical markup. Tags specify the logical structure of the document, including importance of certain pieces of text. The display is done by a program, possibly in conjunction with style sheets, that renders logical elements in a format determined at output time.

Formal (or syntactic) logical elements. Tags specify units such as

heading level 1, *paragraph*, *numbered list*, *emphasize*

Content logical elements. Tags specify units such as

From, *to*, *subject*, *Recommendations*, *warning*, *methods*, *conclusion*,

defining the content structure of a document. These tags can be used to define record formats even for highly structured data. (XML is used increasingly as a language to define the structure of data in Web-based database applications.)

The display program then determines the physical appearance in accordance with the capabilities of the display device and the preferences set by the user. Examples

A *heading level 1* may appear in Times Roman 16 pt bold or in all caps

A new *paragraph* may start with a blank line and no indention (block style) or without a blank line with the first line indented.

The document element *warning* may be displayed in a box with light gray background and a heading **Warning**.

Since logical content markup makes the logical structure of a document explicit, it can be used for information organization and retrieval as well. It can be used to define record formats for straightforward data to be processed by a database management system or to define templates for complex documents (see the examples in Lecture 11b). Organizations use SGML- or XML-derived markup languages to organize large databases of documentary, especially textual information.

HTML is a markup language; all tags are predefined. HTML emphasizes logical markup, but the logical elements are primarily formal, and HTML includes an increasing number of physical markup tags (but still not enough to provide tight control over the appearance of a page).

SGML and XML ("SGML lite", see the main XML Web site at http://www.w3.org/XML/) are not markup languages but *markup metalanguages*: there are no predefined tags; authors and system administrators define their own tags. Many specific markup languages can be defined using SGML or XML This makes it possible to represent more of a document's semantic structure than HTML does. HTML is one of many markup languages defined in terms of SGML.

Examples

HTML example

```
<HTML>
   <HEAD>
      <TITLE>What XML can do for us</TITLE>
      <META NAME="Author" CONTENT="Bob Boiko">
      <META NAME="keywords" CONTENT="XML; content management; document structure;
      databases on the Web">
      <META NAME="GENERATOR" CONTENT="" >
   </HEAD>
   <BODY>
      <H1><Center>Memorandum</Center></H1>
      To: Sue Feldman, CIO <BR>From: Bob Boiko<BR>
      Date: February 7, 2003<BR><BR>
      Subject: <EM>What XML can do for us</EM>
      <P>XML allows us to define document structures that will make it easier to create documents.
      Once a document is created, it can be displayed in many different ways (Web page in multiple
      formats, print, etc.) through applying style sheets (the simple Cascading Style Sheets, CSS2, or
      the more powerful eXtensible Stylesheet Language for document Transformation, XSLT). . . .
      </P>
   </BODY>
</HTML>
```

Document displayed

Memorandum

To: Sue Feldman, CIO
From: Bob Boiko
Date: February 7, 2003
Subject: **What XML can do for us**

XML allows us to define document structures that will make it easier to create documents. Once a document is created, it can be displayed in many different ways (Web page in multiple formats, print, etc.) through applying style sheets (the simple Cascading Style Sheets, CSS2, or the more powerful eXtensible Stylesheet Language for document Transformation, XSLT). . . ..

****

XML example

In XML one must first create an **XML schema** which specifies tags for the parts of a document (and thus document structure), in the example for a document type (or class) called *Memo*. The example assumes that the Memo schema is stored at location http://www.jasca.com/cm/memo.xsd..

Note: Most document structure definitions still use a document type definitions (DTD), but XML schemas are more powerful and will replace DTDs. The XML schema syntax is defined in the W3C Recommendation XML Schema (appr. May 2, 2001); the special tag definitions are in a public file.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!-- w3 schema file defines an XML name space; we use prefix xsd. –>
  <xsd:element name="memo" type="memoType"/>
  <xsd:complexType name="memoType">
     <xsd:sequence>
        <xsd:element name="metadata" type="metadataType"/>
        <xsd:element name="memoBody" type="memoBodyType"/>
     </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="metadataType">
     <xsd:sequence>
        <xsd:element name="to" type="xsd:string"/>
        <xsd:element name="from" type="xsd:string"/>
        <xsd:element name="subject" type="xsd:string"/>
        <xsd:element name="date" type="xsd:date"/>
        <xsd:element name="keywords" type="xsd:string"/>
     </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="memoBodyType">
     <xsd:sequence>
        <xsd:element name="plainText" type="xsd:string"/>
     </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

An XML schema defines a document structure and identifies each element of the structure by a tag. This XML code creates a **memo schema**. The documents in the memo class must contain one top-level element, *memo*, which in turn consists of two subordinate elements, *Metadata* and *memoBody* (exactly one of each in this order), which in turn contain subordinate elements.

A document instance of type *memo*

```
<?xml version="1.0"?>
<?xml:stylesheet type=”text/XSLT”
     xlink:href=”http://www.jasca.com/cm/memo.xslt”?>
<memo xmlns=”http://www.jasca.com/cm/memo.xs”>
     <metadata>
          <to>Sue Feldman, CIO</to>
          <from>Bob Boiko</from>
          <subject>What XML can do for us</subject>
          <date>February 7, 2003</date>
          <keywords>XML; content management; document structure; databases on the Web"</keywords>
     </metadata>
     <memoBody>
          <plainText>XML allows us to define document structures that will make it easier to create
          documents. Once a document is created, it can be displayed in many different ways ...</plainText>
     </memoBody>
</memo>
```

The *document instance* gives the content of the document. The document is structured as a record with data fields, named by the tags, in a database of documents. We must still tell a display program (printing, online display) exactly how each document element should be rendered. This is done through an **XSL style sheet**:

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/TR/WD-XSLT"
  xmlns="http://www.w3.org/TR/REC-html40">
  <xsl:template match="/">
     <HTML>
        <HEAD>
        <TITLE><xsl:value-of select="memo/metadata/subject"/></TITLE>
           <META NAME="Author" CONTENT="{<xsl:value-of select="memo/metadata/from"/>}"/>
           <META NAME="keywords"
              CONTENT="{<xsl:value-of select="memo/metadata/keywords"/>}"/>
           <META NAME="GENERATOR" CONTENT=""/>
        </HEAD>
        <BODY>
           <H1><Center>Memorandum</Center></H1>
           To: <xsl:value-of select="memo/metadata/to"/><BR/>From: <xsl:value-of
           select="memo/metadata/from"/><BR/>
           Date: <xsl:value-of select="memo/metadata/date"/><BR/><BR/>
           Subject: <EM><xsl:value-of select="memo/metadata/subject"/></EM>
           <P><xsl:value-of select="memo/memoBody/plainText"/></P>
        </BODY>
     </HTML>
  </xsl:template>
```

Following the instructions in the style sheet, the XML document is transformed into an HTML document (the HTML document that was given as an example above):

```
<HTML>
   <HEAD>
      <TITLE>What XML can do for us</TITLE>
      <META NAME="Author" CONTENT="Bob Boiko">
      <META NAME="keywords" CONTENT="XML; content management; document structure;
      databases on the Web">
      <META NAME="GENERATOR" CONTENT="" >
   </HEAD>
   <BODY>
      <H1><Center>Memorandum</Center></H1>
      To: Sue Feldman, CIO <BR>From: Bob Boiko<BR>
      Date: February 7, 2003<BR><BR>
      Subject: <EM>What XML can do for us</EM>
      <P>XML allows us to define document structures that will make it easier to create documents.
      Once a document is created, it can be displayed in many different ways (Web page in multiple
      formats, print, etc.) through applying style sheets (the simple Cascading Style Sheets, CSS2, or
      the more powerful eXtensible Stylesheet Language for document Transformation, XSLT). A
      table of contents can be created automatically. Moreover, the document can be displayed
      selectively using just the parts most appropriate for a given audience. Parts of one document
      can be reused in another document. . . . </P>
   </BODY>
</HTML>
```

XML schema for a self assessment memo

Since a self assessment memo is a specific type of memo, we can define its schema by adding to the memo schema; the memo schema is **reused** .

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.jasca.com/cm/memo.xsd">

    <include xsd:schemaLocation="http://www.jasca.com/cm/memo.xs"/>

    <xsd:element name="selfAssessmentMemo" type="selfAssessmentMemoType"/>

    <xsd:complexType name="selfAssessmentMemoType">
        <xsd:sequence>
            <xsd:element name="metadata" type="metadataType"/>
            <xsd:element name="memoBody" type="memoBodyType"/>
        <xsd:sequence>
    </xsd:complexType>

    <!– redefinition of memoBodyType –>
    <xsd:complexType name="memoBodyType">
        <xsd:complexContent>
            <xsd:extension base="memoBodyType">
                <xsd:sequence>
                    <xsd:element name="accomplishments" type="xsd:string"/>
                    <xsd:element name="goals" type="xsd:string"/>
                    <xsd:element name="trainingNeeds" type="xsd:string"/>
                </xsd:sequence>
            </xsd:extension
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

Reuse is a big theme in the application of XML, and there are other mechanisms for it. There are whole collections of defined data types, such as a data type for US states, with the values restricted to a list of two-letter abbreviations of US states, or data types for US address, UK address, France address, Germany address, etc. (all derived from a generic address as a common parent). These type definitions are collected into *vocabularies*, from which they can be included in any XML document schema, saving the schema creator a lot of work.

Note: The syntax of the XML examples may not be correct to the last dotted I and crossed t, but it gives the general idea.

Large XML example: A database of foods

Food database schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <xsd:element name="foodDatabase" type="foodDatabaseType"/>

   <xsd:complexType name="foodDatabaseType">
      <xsd:sequence>
         <xsd:element name="foodProduct" type="foodProductType" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="foodProductType">
      <xsd:sequence>
         <xsd:element name="name" type="xsd:string"/>
         <xsd:element name="isa" type="IDREF"/>
         <xsd:element name="origin" type="originType" minOccurs="0"/>
         <xsd:element name="form" type="xsd:string" minOccurs="0"/
         <xsd:element name="processedBy" type="processedByType" minOccurs="0"
         maxOccurs="unbounded"/>
         <xsd:element name="packedIn" type="xsd:string"minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="foodID" type="ID"  use="required"/>
   </xsd:complexType>

   <xsd:complexType name="originType">
      <xsd:choice>
         <xsd:element name="foodSource" type="xsd:string"/>
         <xsd:element name="part" type="xsd:string" maxOccurs="unbounded"/>
         <xsd:element name="extractedSubstance" type="xsd:string" minOccurs="0"
         maxOccurs="unbounded"/>>
         <xsd:element name="madeFrom" type="xsd:string" minOccurs="0"
         maxOccurs="unbounded"/>
         <xsd:element name="ingredient" type="ingredientType"maxOccurs="unbounded"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="ingredientType">
      <xsd:attribute name="intensity" type="xsd:string" use="optional"/>
      <xsd:attribute name="purpose" type="xsd:string" use="optional"/>
   </xsd:complexType>

   <xsd:complexType name="processedByType"  mixed="true">
      <xsd:attribute name="purpose" type="xsd:string" use="optional"/>
   </xsd:complexType>
```

Not shown in this example are two options (think back to frames):

(1) Declaring restrictions on an element or attribute (by form, e.g. min and max length of a string, min and max values of numbers, or by an enumerated authority list of allowed values, see text, Section 9.1.1).

(2) Declaring a default value.

The food data from Lecture 3

```
<?xml version="1.0"?>
<?xml:stylesheet type=”text/XSLT”
    xlink:href=”http://www.afw.com/it/database.xslt”?>
<!-- File Name: foodDatabase.xml -->
<!-- Element content shown in italics for clarity -->

<foodDatabase xmlns=”http://www.afw.com/it/database.xsd”>
    <foodProduct foodID=”FP0">
        <name>Food product</name>
    </foodProduct>

    <foodProduct foodID=”FP1">
        <name>Vegetable product</name>
        <isa>FP0</isa>
        <origin>
            <foodSource>Plant</foodSource>
        </origin>
    </foodProduct>

    <foodProduct foodID=”FP2">
        <name>Meatproduct</name>
        <isa>FP0</isa>
        <origin>
            <foodSource>Animal</foodSource>
            <part>Carcass</part>
        </origin>
    </foodProduct>

    <foodProduct foodID=”FP3">
        <name>Egg product</name>
        <isa>FP0</isa>
        <origin>
            <foodSource>Bird</foodSource>
            <part>Egg</part>
        </origin>
    </foodProduct>

    <foodProduct foodID=”FP4">
        <name>Prepared food</name>
        <isa>FP0</isa>
        <processedBy>Process</ProcessedBy>
    </foodProduct>
```

```
<foodProduct foodID="FP5">
    <name>Soup</name>
    <isa>FP4</isa>
    <processedBy>Process</ProcessedBy>
    <form>Liquid or semiliquid</form>
</foodProduct>

<foodProduct  foodID="FP11">
    <name>Diced carrots</name>
    <isa>FP1</isa>
    <origin>
        <foodSource>Carrot plant</foodSource>
        <part>Root</part>
    </origin>
    <form>Diced</form>
</foodProduct>

<foodProduct foodID="FP12">
    <name>Cut green beans</name>
    <isa>FP1</isa>
    <origin>
        <foodSource>Bean plant</foodSource>
        <part>Immature fruit</part>
    </origin>
    <form>Cut</form>
</foodProduct>

<foodProduct foodID="FP13">
    <name>Chicken broth</name>
    <isa>FP2</isa>
    <origin>
        <foodSource>Chicken</foodSource>
        <part>Meat</part>
        <part>Bones</part>
    </origin>
    <extractedSubstance>Fat</extractedSubstance>
    <extractedSubstance>Protein</extractedSubstance>
    <extractedSubstance>Flavor</extractedSubstance>
    <processedBy>Cooking</processedBy>
    <form>Liquid</form>
</foodProduct>

<foodProduct foodID="FP14">
    <name>Cubed cooked chicken</name>
    <isa>FP2</isa>
    <origin>
        <foodSource>Chicken</foodSource>
        <part>Skeletal meat</part>
    </origin>
    <processedBy>Cooking</processedBy>
    <form>Cubed</form>
</foodProduct>

<foodProduct  foodID="FP15">
```

```
    <isa>FP3</isa>
    <name>Eggs</name>
    <origin>
        <foodSource>Chicken</foodSource>
        <part>Egg</part>
    </origin>
</foodProduct>

<foodProduct  foodID="FP16">
    <isa>FP1</isa>
    <name>Durum wheat flower</name>
    <origin>
        <foodSource>Durum wheat</foodSource>
        <part>Seed, kernel</part>
    </origin>
    <form>Ground</form>
</foodProduct>

<foodProduct foodID="FP17">
    <isa>FP4</isa>
    <name>Noodles</name>
    <origin>
        <ingredient>FP16</ingredient>
        <ingredient>FP15 </ingredient>
    </origin>
    <processedBy>Mixing</processedBy>
    <processedBy>Extruding</processedBy>
    <processedBy>Drying</processedBy>
</foodProduct>

<foodProduct  foodID="FP18">
    <name>Flavoring</name>
    <isa>FP0</isa>
</foodProduct>

<foodProduct  foodID="FP19">
    <name>BHT</name>
    <isa>FP0</isa>
</foodProduct>

<foodProduct foodID="FP20">
    <name>Chicken noodle soup</name>
    <isa>FP5</isa>
    <origin>
        <ingredient>FP13</ingredient>
        <ingredient>FP14</ingredient>
        <ingredient>FP11</ingredient>
        <ingredient>FP12</ingredient>
        <ingredient>FP17</ingredient>
        <ingredient>FP18</ingredient>
        <ingredient purpose="Preservation">FP19</ingredient>
    </origin>
    <processedBy purpose="Make edible" purpose=" Preservation">Sterilizing by heat
    </processedBy>
```

```
    <form>Liquid with Solid Pieces</form>
</foodProduct>

<foodProduct  foodID="FP21">
    <name>Diced parsley</name>
    <isa>FP1</isa>
</foodProduct>

<foodProduct foodID="FP22">
    <name>Campbell's chicken noodle soup</name>
    <isa>FP20</isa>
    <origin>
        <ingredient> FP13"</ingredient>
        <ingredient> FP14 </ingredient>
        <ingredient> FP11 </ingredient>
        <ingredient> FP12 </ingredient>
        <ingredient> FP22 </ingredient>
        <ingredient> FP17 </ingredient>
        <ingredient> FP18 </ingredient>
        <ingredient purpose="Preservation"> FP19</ingredient>
    </origin>
    <processedBy purpose="Make edible" purpose=" Preservation">Sterilizing by heat
    </processedBy>
    <form>Liquid with Solid Pieces</form>
    <packedIn>Steel can</packedIn>
</foodProduct>

<foodProduct foodID="FP23">
    <name>Frozen cut green beans</name>
    <isa>FP12</isa>
    <origin>
        <foodSource>Bean plant</foodSource>
        <part>Immature fruit</part>
    </origin>
    <form>Cut</form>
    <processedBy>Freezing</processedBy>
    <packedIn>Carton</packedIn>
</foodProduct>
```

Style sheet for displaying food data

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/TR/WD-XSLT"
  xmlns="http://www.w3.org/TR/REC-html40">
  <xsl:template match="/">
     <HTML>
        <HEAD>
        <TITLE>Food database listing</TITLE>
           <META NAME="Author" CONTENT="Association of Food Wholesalers"/>
           <META NAME="keywords"
              CONTENT="food products; formulation; ingredients"/>
           <META NAME="GENERATOR" CONTENT=""/>
        </HEAD>
        <BODY>
           <H1><Center>Association of Food Wholesalers Product List</Center></H1>
           <H1><Center>Table of Contents</Center></H1>
           <xsl:for-each select="foodDatabase/foodProduct">
              <xsl:value-of select="@foodID"/> 
              <xsl:value-of select="name"/><BR/>
           </xsl:for-each>

           <H1><Center>Full Food Product Listing<Center></H1>
           <xsl:for-each select="foodDatabase/foodProduct">
              <B><xsl:value-of select="@foodID"/> 
              <xsl:value-of select="name"/> </B><BR><I>isa <I> 
              <xsl:value-of select="isa"/><BR/>
              <I>Food source or ingredients:<I> 
              <xsl:value-of select="origin/foodSource"/>
              <xsl:for-each select="origin//part">
                 <xsl:value-of select="."/>
              </xsl:for-each>
              <xsl:for-each select="origin/ingredient">
                 <xsl:value-of select="."/>
                 <xsl:value-of select="@purpose"/>, 
              </xsl:for-each><BR/>
              <I>Extracted substance:<I> 
              <xsl:for-each select="origin/extractedSubstance">
                 <xsl:value-of select="."/>, 
              </xsl:for-each><BR/>
              <I>Processsed by:<I> 
              <xsl:for-each select="processsedBy">
                 <xsl:value-of select="."/>
                 <xsl:value-of select="@purpose"/>
              </xsl:for-each><BR/>
              <I>Form:<I> 
              <xsl:value-of select="form"/><BR/>
              <I>Packed in:<I> 
              <xsl:value-of select="packedIn"/>
              <BR/><BR/>
           </xsl:for-each><BR/>
```

```
        <H1><Center>Food Name Index</Center></H1>
        <xsl:for-each select="foodDatabase/foodProduct" order-by'"+Name">
            <xsl:value-of select="name"/> 
            <xsl:value-of select="@foodID"/><BR>
        </xsl:for-each>
      </BODY>
    </HTML>
</xsl:template>
```

Display of the HTML document produced by the style sheet

Association of Food Wholesalers Product List

Table of Contents

Full Food Product Listing

FP0 Food product

FP1 Vegetable product
isa: FP0
*Food source or ingredients:*Plant

FP2 Meat product
isa: FP0
Food source or ingredients: Animal Carcass

FP3 Egg product
isa: FP0
Food source or ingredients: Bird Egg

FP4 Prepared food
isa: FP0
Processed by: Process

FP5 Soup
isa: FP0
Processed by: Process
Form: Liquid or semiliquid

FP11 Diced carrots
isa: FP1
Food source or ingredients: Carrot plant Root
Extracted substance:
Processed by:
Form: Diced
Packed in:

FP12 Cut green beans
isa: FP1
Food source or ingredients: Bean plant Immature fruit
Extracted substance:
Processed by:
Form: Cut
Packed in:

FP13 Chicken broth
isa: FP2
Food source or ingredients: Chicken Meat Bones
Extracted substance: Fat, Protein, Flavor,
Processed by: Cooking
Form: Liquid
Packed in:

FP14 Cubed cooked chicken
isa: FP2
Food source or ingredients: Chicken Skeletal meat
Extracted substance:
Processed by: Cooking
Form: Cubed
Packed in:

FP15 Eggs
isa: FP3
Food source or ingredients: Chicken Egg
Extracted substance:
Processed by:
Form:
Packed in:

FP16 Durum wheat flower
isa: FP1
Food source or ingredients: Durum Wheat Seed, kernel
Extracted substance:
Processed by:
Form: Ground
Packed in:

FP17 Noodles
isa: FP4
Food source or ingredients: FP16, FP15,
Extracted substance:
Processed by: Mixing, Extruding, Drying
Form:
Packed in:

FP18 Flavoring
isa: FP0
Food source or ingredients:
Form:
Extracted substance:
Processed by:
Packed in:

FP19 BHT
isa: FP0
Food source or ingredients:
Form:
Extracted substance:
Processed by:
Packed in:

FP20 Chicken noodle soup
isa: FP5
Food source or ingredients: FP13, FP14, FP11, FP12, FP17, FP112, FP113 Preservation,
Extracted substance:
Processed by: Sterilizing by heat Make edible, Preservation
Form:
Packed in: Steel can

FP21 Diced parsley
isa: FP1
Food source or ingredients:
Form:
Extracted substance:
Processed by:
Packed in:

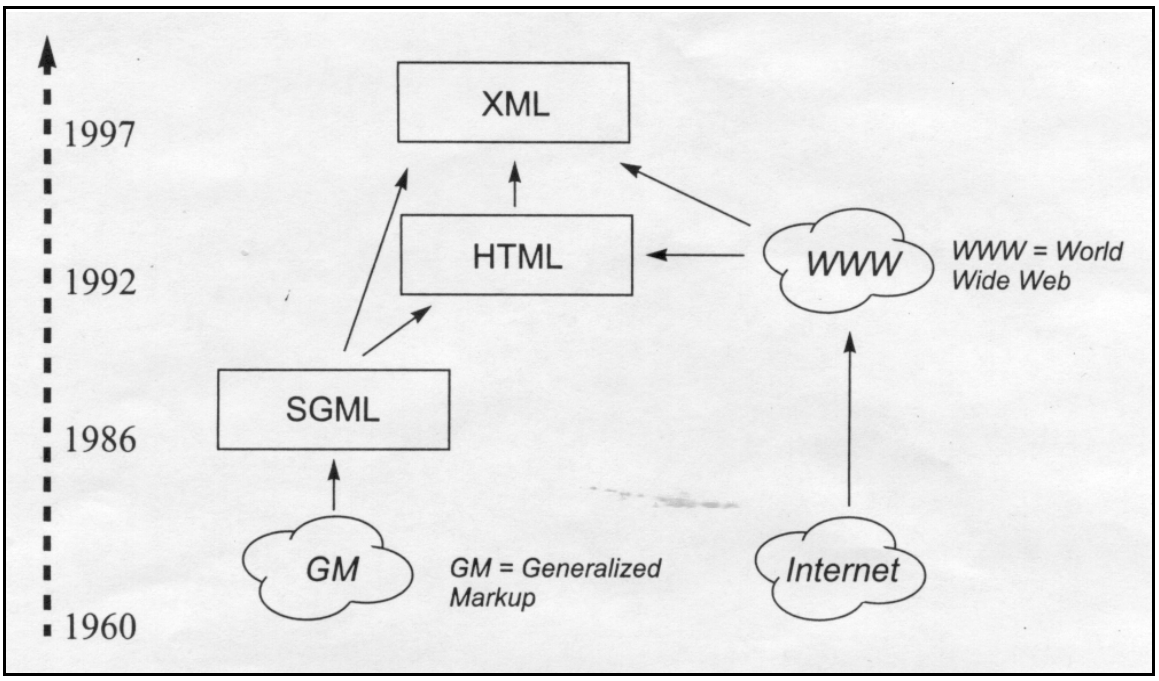
FP22 Campbell's chicken noodle soup
isa: FP20
Food source or ingredients: FP13, FP14, FP11, FP12, FP22, FP17, FP18, FP19 Preservation,
Extracted substance:
Processed by: Sterilizing by heat Make edible, Preservation
Form:
Packed in: Steel can

FP23 Frozen cut green beans
isa: FP12
Food source or ingredients: Bean plant Immature fruit
Extracted substance:
Processed by: Freezing
Form: Cut
Packed in: Carton

Food Name Index

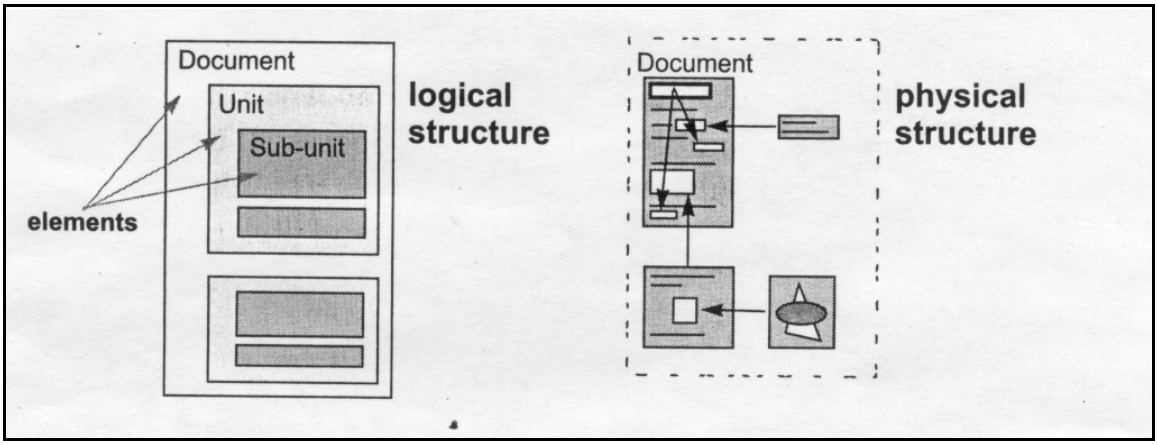
XML in a larger context

History

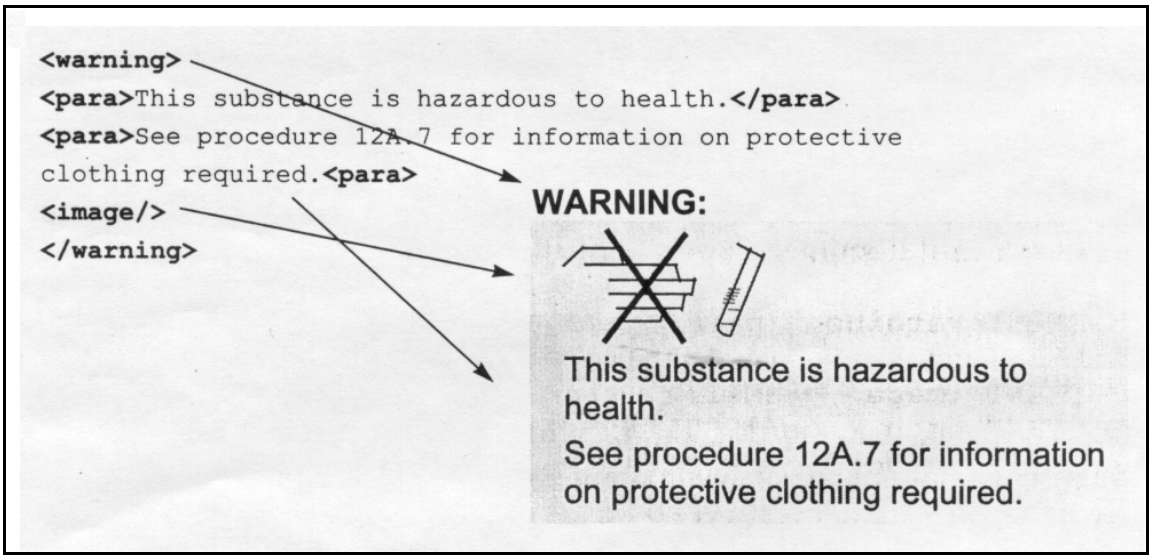


XML: 80% of the functionality, 20% of the complexity of SGML

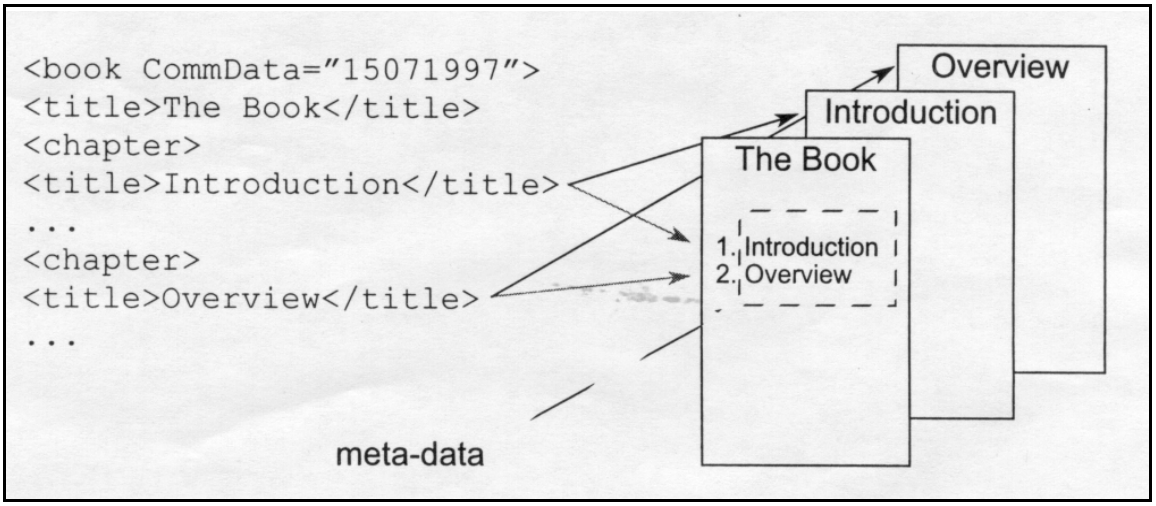
Document management: Logical and physical structure



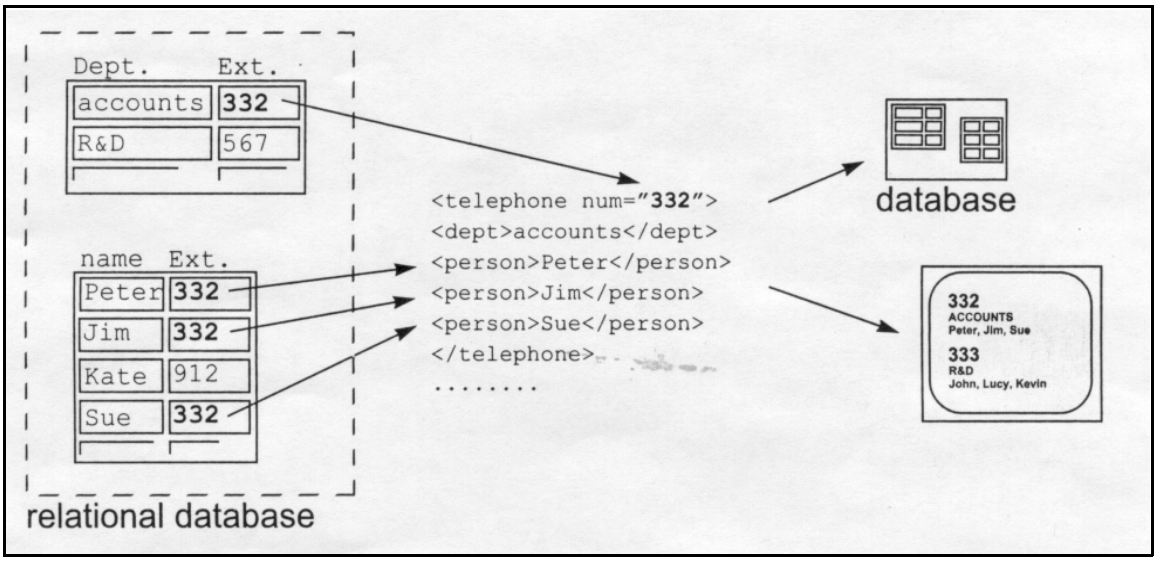
From XML to display (style sheets are used to govern this mapping)



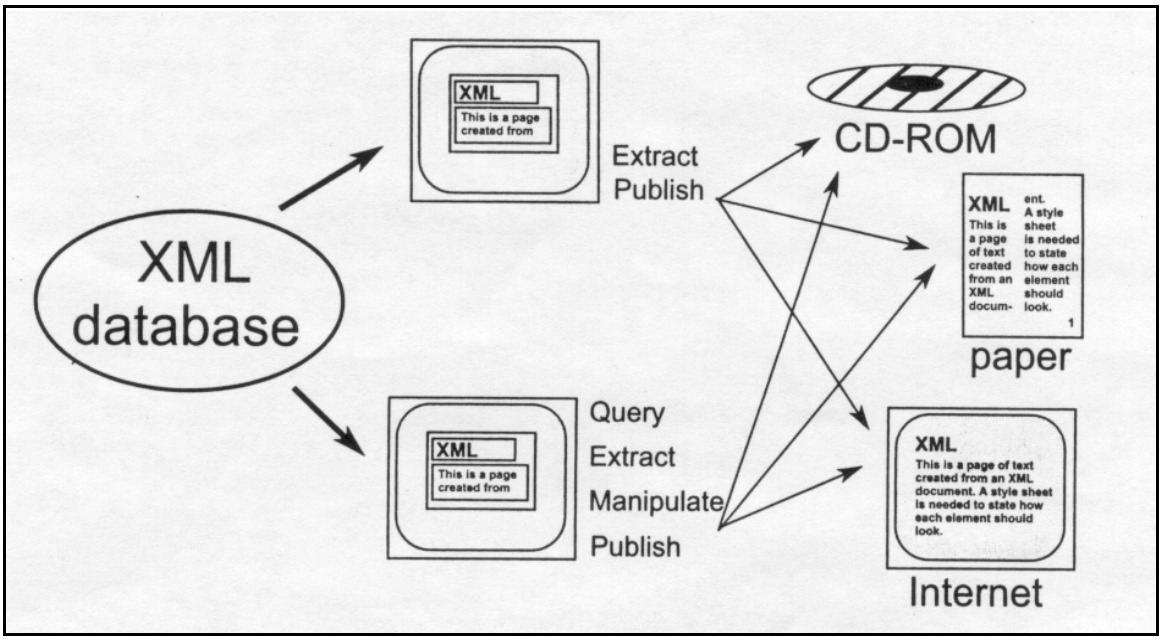
From XML to display: Multiple use of data (also: extraction of cataloging data)



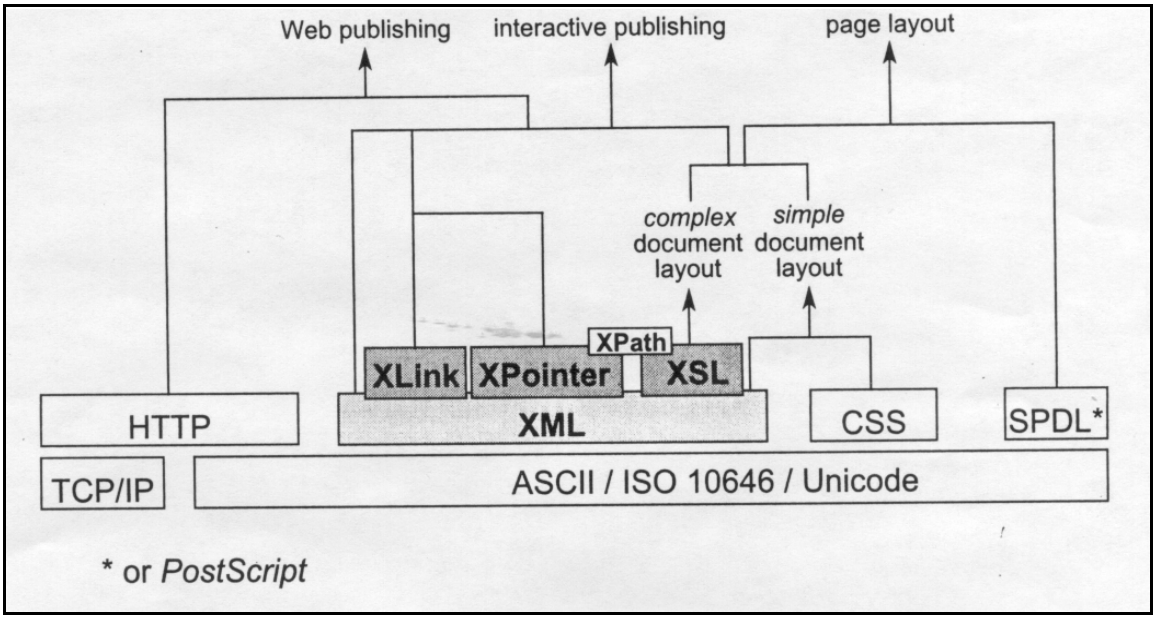
From database to XML, from XML to display or a different database

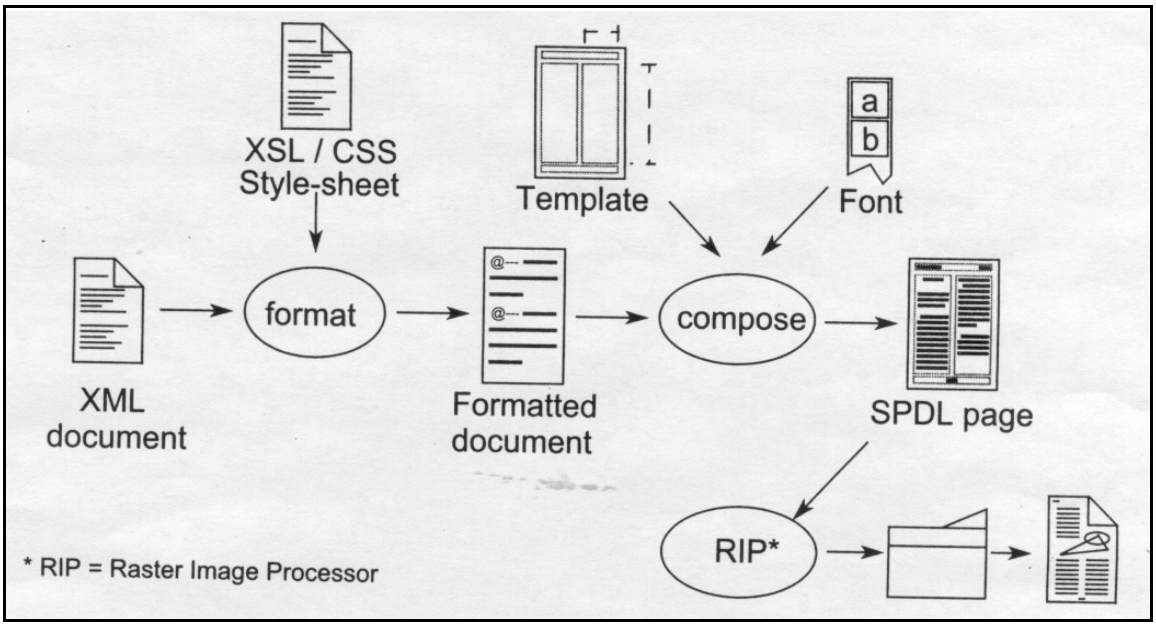


XML database for multiple uses through simple extraction or more complex manipulation



Alphabet soup: Many standards working together





	Transfer protocols
TCP/IP	Transfer Control Protocol / Internet Protocol
HTTP	Hypertext Transfer Protocol
	Character code standards
ASCII	American Standard Code for Information Interchange common characters used in English. 7 bit (extended ASCII 8 bit). Almost identical to ISO/IEC 646:1991 (7 bit, 128 characters) ISO International Standards Organization IEC International Electrotechnical Commission
ISO/IEC8859/1	8 bit, 256 characters. Superset of ASCII / ISO/IEC646:1991. Adds accented characters and such. Used in HTML and Windows. One of family of 8 bit codes
Unicode	16-bit superset of ISO/IEC8859/1. Extended character sets for many languages. 64,000 characters
ISO/IEC 10646	32-bit superset of Unicode, 4 billion characters
	Formatting standards
SGML	Standard Generalized Markup Language
HTML	Hypertext Markup Language, an SGML application
XML	eXtensible Markup Language
MathML	Mathematical Markup Language, an XML application
SMIL	Synchronized Multimedia Integration Language, an XML application
CSS	Cascading Style Sheets (for use with HTML)
XSL	XML Stylesheet Language
XSLT	XSL Transformation. A standard for transforming any XML document into a specific XML format suitable for output according to a style sheet
SPDL	Standard Page Description Language
PostScript	ADOBE's proprietary but widely used page description language
XLink	XML Linking Language (more complex linking than HTML), an XML application
XPointer	Complementary to XLink, allows linking to a specific position within a document
	XML related query language standards
XQL	eXtensible Query Language - a general query language for XML documents and structured data
XSL Pattern	A specific implementation of XQL
XPath	An expression language for querying XML documents and data structures

	Image format standards
JPEG	Joint Photographic Experts Group 24-bit raster format for image compression, preferred when high resolution is needed
GIF	Graphic Interchange Format, 8-bit raster format, preferred on the Web for simple images
PNG	Portable Network Graphics. Developed for the Internet, consistent appearance across platforms, my replace GIF.
MPEG	Moving Picture Experts Group. Developed a number of standards for the physical storage and logical description of moving images
	General Models and frameworks
RDF	Resource Description Framework. A framework for the representation of metadata, uses an entity-relationship approach
DOM	Document Object Model. A software interface standard for accessing and manipulating document elements as objects

References

Bradley, Neil.
The XML companion.
2. ed. Harlow, England: Addison-Wesley / Pearson Education; 2000. 566 p.
ISBN0-201-67486-6

Pardi, William J.
XML in action.
Redmond, WA: Microsoft Press; 1999. 329 p., CD
ISBN 0-7356-0562-9

Young, Michael J.
Step by step XML
Redmond, WA: Microsoft Press; 2000. 382 p., CD
ISBN 0-7536-1020-7

Simpson, John E.
Just XML. 2. ed.
Upper Saddle River, NJ: Prentice Hall; 2003. 417 p.
ISBN 0-13-018554-X

Morrison, Michael et al.
XML unleashed.
Sams; 2000. 958 p., CD
ISBN 0-672-31514-9

These books are on reserve as Personal Copies.

RDF schema definition for the food database from Lecture 12b

```
<?xml version="1.0"?>
<rdf:RDF
      /* Name space decalarations */
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

      /* Entity type definitions */

<rdf:Description rdf:ID="FoodProduct">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Organism">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="OrganismPart">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Substance">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Purpose">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Process">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="ProcessIntensity">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Form">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Container">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

/* Property (relationship type) definitions */

```
<rdf:Description rdf:ID="hasName">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Description>

<rdf:Description rdf:ID="isa">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#FoodProduct"/>
</rdf:Description>

<rdf:Description rdf:ID="comesFromSource">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#Organism"/>
</rdf:Description>

<rdf:Description rdf:ID="comesFromPart">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#OrganismPart"/>
</rdf:Description>

<rdf:Description rdf:ID="isExtractedSubstance">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#Substance"/>
</rdf:Description>

<rdf:Description rdf:ID="isMadeFrom">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#FoodProduct"/>
</rdf:Description>

<rdf:Description rdf:ID="hasIngredient">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#FoodProduct"/>
</rdf:Description>

<rdf:Description rdf:ID="processedBy">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#Process"/>
</rdf:Description>

<rdf:Description rdf:ID="hasForm">
```

```
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#Form"/>
</rdf:Description>

<rdf:Description rdf:ID="packedIn">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#FoodProduct"/>
    <rdfs:range rdf:resource="#Container"/>
</rdf:Description>

<rdf:Description rdf:ID="eats">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#Organism"/>
    <rdfs:range rdf:resource="#FoodProduct"/>
</rdf:Description>

</rdf:RDF>
```

This is stored at

http://www.clis.umd.edu/faculty/soergel/FoodSchema

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:food="http://www.clis.umd.edu/faculty/soergel/FoodSchema”>

      /* Entity values assigned to entity types */

<rdf:Description rdf:ID="CarrotPlant">
     <rdf:type rdf:resource="http://www.clis.umd.edu/faculty/soergel/FoodSchema#Organism"/>
</rdf:Description>

<rdf:Description rdf:ID="Root">
     <rdf:type rdf:resource="http://www.clis.umd.edu/faculty/soergel/FoodSchema#OrgaismPart"/>
</rdf:Description>

<rdf:Description rdf:ID="Diced">
     <rdf:type rdf:resource="http://www.clis.umd.edu/faculty/soergel/FoodSchema#Form"/>
</rdf:Description>

      /* Statments on Food Products */

<rdf:Description rdf:ID="FP0">
     <rdf:type rdf:resource="http://www.clis.umd.edu/faculty/soergel/FoodSchema#FoodProduct"/>
      <food:hasName>”Food Product”</food:hasName>
</rdf:Description>

<rdf:Description rdf:ID="FP1">
     <rdf:type rdf:resource="http://www.clis.umd.edu/faculty/soergel/FoodSchema#FoodProduct"/>
      <food:hasName>”Vegetable product”</food:hasName>
      <food:isa rdf:resource=”#FP0"/>
</rdf:Description>

<rdf:Description rdf:ID="FP11">
      <rdf:type rdf:resource="http://www.clis.umd.edu/faculty/soergel/FoodSchema#FoodProduct"/>
      <food:hasName>”Diced carrots”</food:hasName>
      <food:isa rdf:resource=”#FP1"/>
      <food:comesFromSource rdf:resource=”#CarrotPlant"/>
      <food:comesFromPart rdf:resource=”#Root"/>
      <food:hasForm rdf:resource=”#Diced"/>
</rdf:Description>


</rdf:RDF>
```

RDF schema definition example: Dublin Core

```
<?xml version='1.0'?>
 <rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"/>


 <rdfs:Class rdf:ID="Book">
    <rdfs:label>Book</rdfs:label>
    <rdfs:comment>The class of books</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
 </rdfs:Class>

 <rdf:Property rdf:ID="title">
   <rdfs:label>Title</rdfs:label>
   <rdfs:comment>The name given to the resource</rdfs:comment>
   <rdfs:domain rdf:resource="#Book"/>
   <rdfs:range
      rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
 </rdf:Property>

 <rdfs:Property ID="author">
   <rdfs:label>Author</rdfs:label>
   <rdfs:comment>A person responsible for creating a written document</rdfs:comment>
   <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/dcmes.rdf#Creator"/>
   <rdfs:domain rdf:resource="#Book"/>
   <rdfs:range
      rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
 </rdfs:Property>

 <rdf:Property ID="organisation">
   <rdfs:label>Organisation</rdfs:label>
   <rdfs:comment="The author's affiliation."/>
   <rdfs:domain rdf:resource="#Author"/>
   <rdfs:range rdf:resource="#OrgNames"/>
 </rdf:Property>

 <rdfs:Property rdf:ID="OrgNames"/>

 <OrgNames rdf:ID="OCLC"/>
 <OrgNames rdf:ID="Cornell University"/>
 <OrgNames rdf:ID="DSTC"/>

 </rdf:RDF>
```

Below is the corresponding XML Schema representation

```
<schema xmlns="http://www.w3.org/1999/XMLSchema"
        targetNamespace="http://www.dstc.edu.au/"
        xmlns:dstc="http://www.dstc.edu.au/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"/>

  <import namespace="http://purl.org/dc/elements/1.1/"/>

  <element name="Book">
    <annotation>
      <documentation>The class of books</documentation>
    </annotation>

    <sequence>
      <element ref="dc:title" minOccurs="1" maxOccurs="1"/>
      <element name="author" type="author" maxOccurs="4"/>
    </sequence>
    <attribute name="id" type="uriReference"/>
  </element>

  <complexType name="author">
     <extension base="dc:creator">
       <element name="organisation" type="OrgNames"/>
     </extension>
  </complexType>

  <simpleType name="OrgNames">
    <restriction base="string">
      <enumeration value="OCLC"/>
      <enumeration value="Cornell University"/>
      <enumeration value="DSTC"/>
    </restriction>
  </simpleType>

 </schema>
```

References

Resource Description Framework (RDF) Model and Syntax Specification
W3C Recommendation 22 February 1999
http://www.w3.org/TR/REC-rdf-syntax/

Resource Description Framework (RDF) Schema Specification 1.0
W3C Candidate Recommendation 27 March 2000
http://www.w3.org/TR/rdf-schema

Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles
http://archive.dstc.edu.au/RDU/staff/jane-hunter/www10/paper.html

Jane Hunter
DSTC Pty Ltd
University of Qld, Australia
jane@dstc.edu.au

Carl Lagoze
Digital Library Research Group
Cornell University, NY
lagoze@cs.cornell.edu

(Source of the RDF schema example)

An Idiot's Guide to the Resource Description Framework
Renato Iannella
DSTC Pty Ltd
The University of Queensland, 4072, AUSTRALIA
Email: renato@dstc.edu.au
To Appear in: The New Review of Information Networking, Vol 4, 1998
First Published: 1998-09-03
Updated: 1999-01-25
Status: Conforms with RDF Syntax Specification 1999-01-05
http://archive.dstc.edu.au/RDU/reports/RDF-Idiot/

W3C Recommendation 22 February 1999: www.w3.org/TR/REC-rdf-syntax/