BINF 8211/6211
Design and Implementation of Bioinformatics Databases
Lecture #19

Dr. D. Andrew Carr
Dept. Bioinformatics and Genomics UNCC
Spring 2016

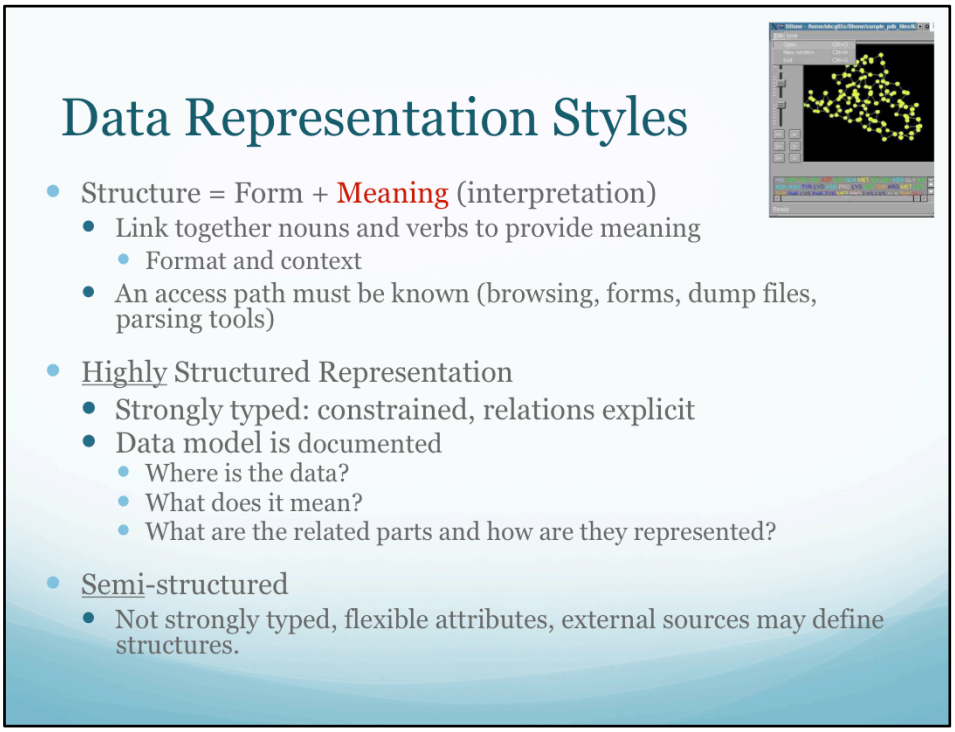
Review Relational Model

Today Topics

- SQL
- Project information review
  - Homework 6,7
- Data Representation

SQL

- TRIGGER
  - Used to autom atically update after insert of data.
  - CREATE or REPLACE TRIGGER
    - AFTER INSERT or UPDATE OF ...
    - BEGIN
      - UPDATE ...
    - END
- Stored procedures
  - Internal SQL that are contained with the database
- Stored Functions
- Embedded SQL



What is it that you have to map? That is, the material, the form that you are working with – has a shape, or constraints, and properties

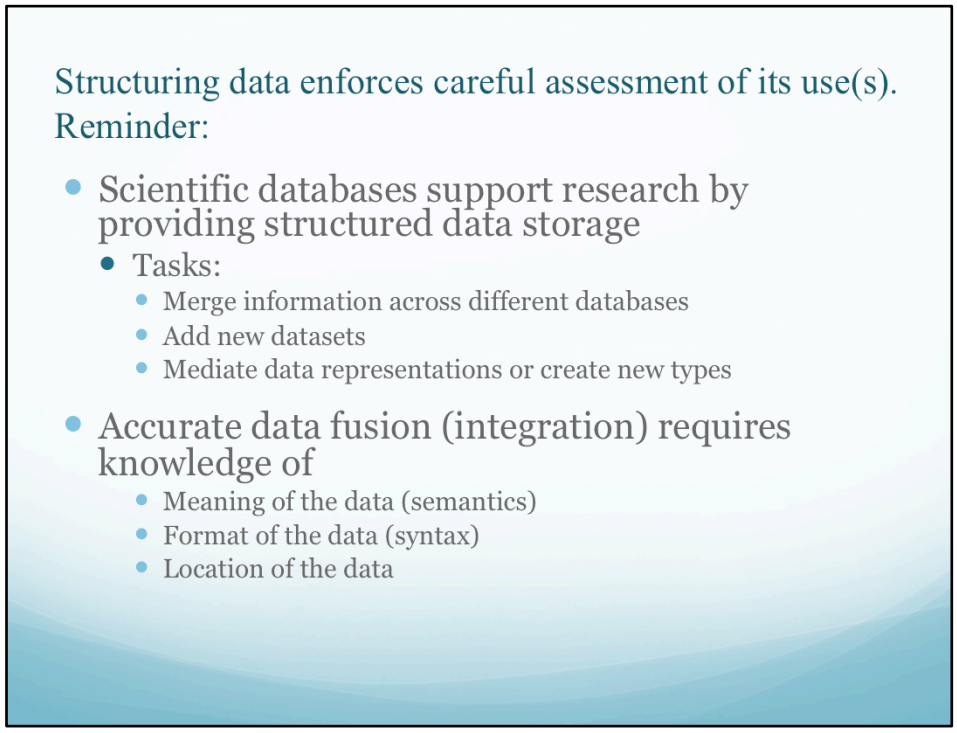
Strongly typed says what types of values are possible, what operations are valid, the meanings that can be associated with the data and how the data must be stored (floating point vs integer vs character for example).

What structures can the material support – what can it interact with

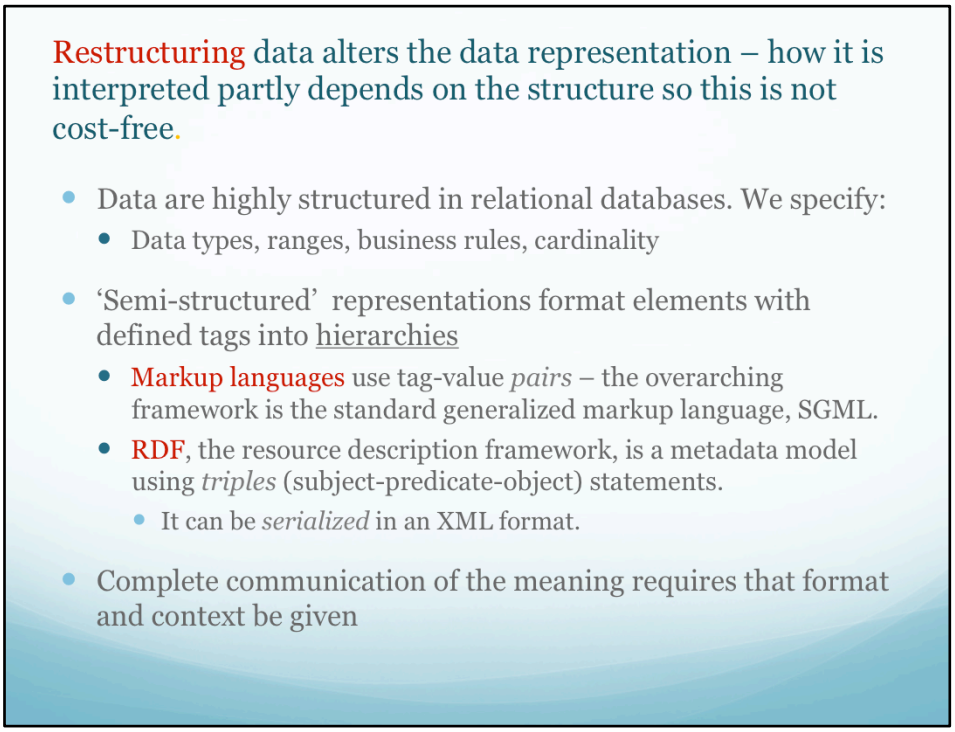
Representation means partly how thoroughly you understand it and how well you described it

If you only know a little bit then trying to put it into a framework that demands a great deal of detail can be frustrating (or you guess)

Semi-structured that allows for flexibility may be more efficient in the long run, even if it is not as elegant



With scientific data the goal is to uncover new information, find new relationships by making new combinations and transformations
Merge outcomes, extract new values
Much easier to create a mud pie than a pot that can stand under its own weight or even contain other substances– the architecture matters, material and connections



standard generalized markup language

Interpretation: substance vs style

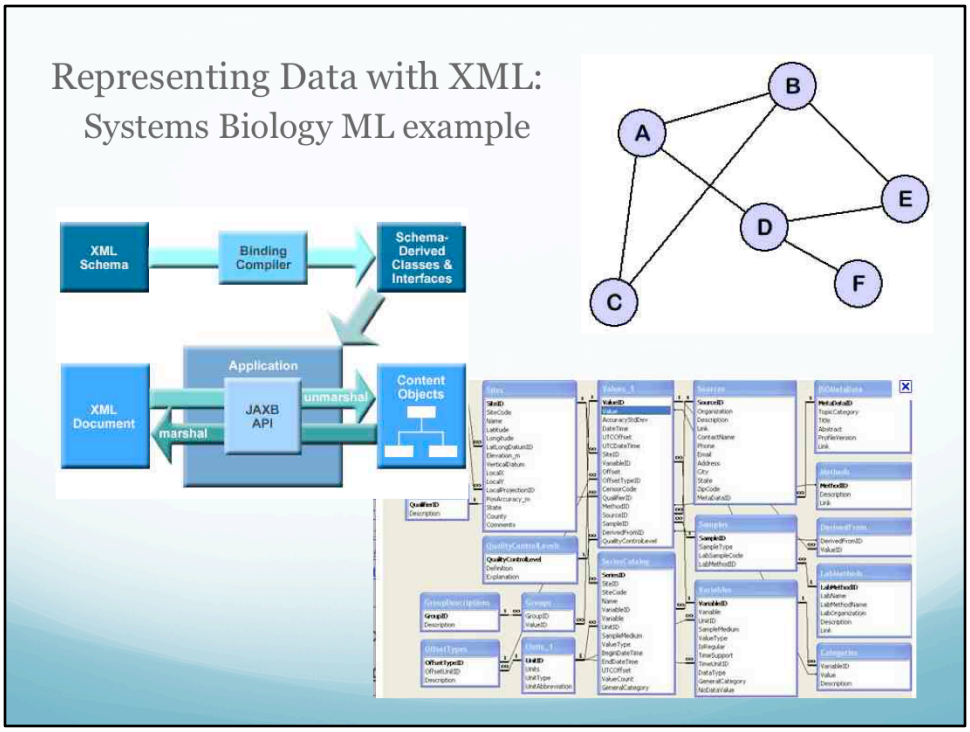
- Tags are familiar:
  - HTML uses tags to describe how things should look
    hi for

    hi
  - XML uses tags to describe meaning

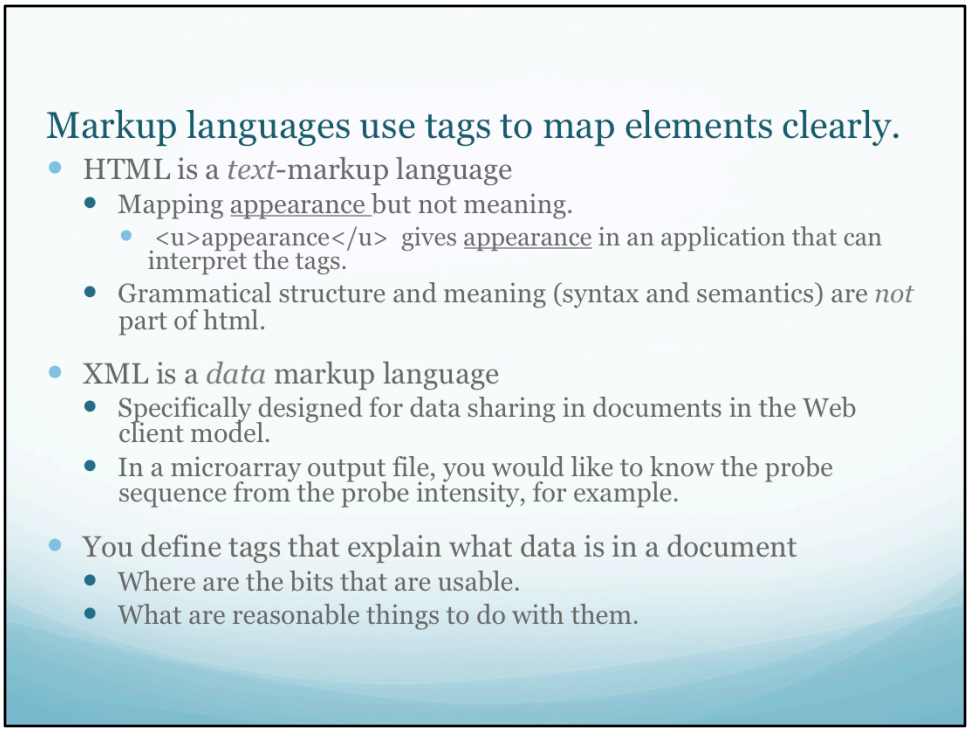
    <emphasis>hi</emphasis> HI!

    <greeting>hi</greeting> Hi
- An XML *processor* provides presentation
  - Read file, convert the data into representation appropriate for another program
  - Tags → symbols for actions
    - web browsers, XML editors, data systems, etc.

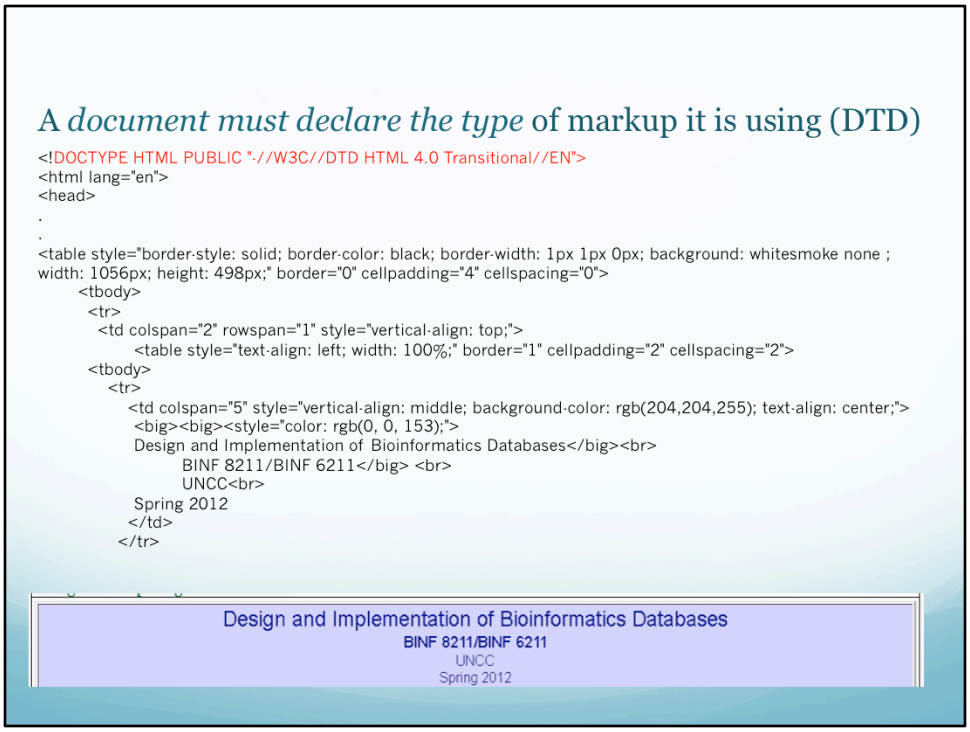
3/31/16



Different ways to structure data allow choices for downstream applications, exchange, sharing.
Main ways to discriminate representation modes= structured vs unstructured



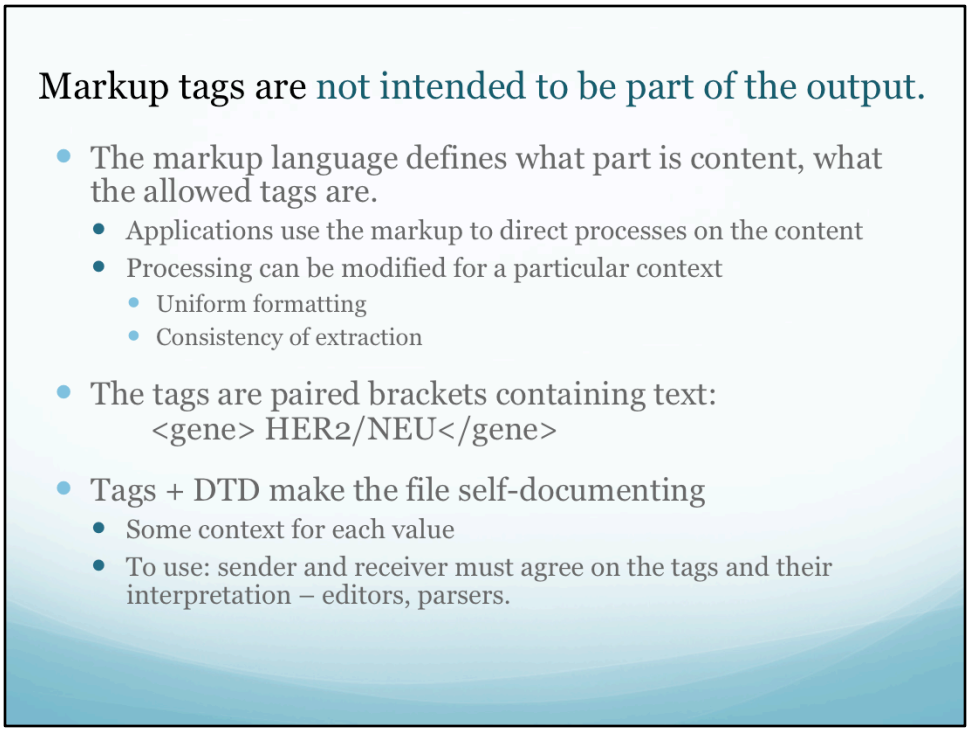
Syntax is the part of speech (noun, verb – do what to whom) and semantic is what it means in context.



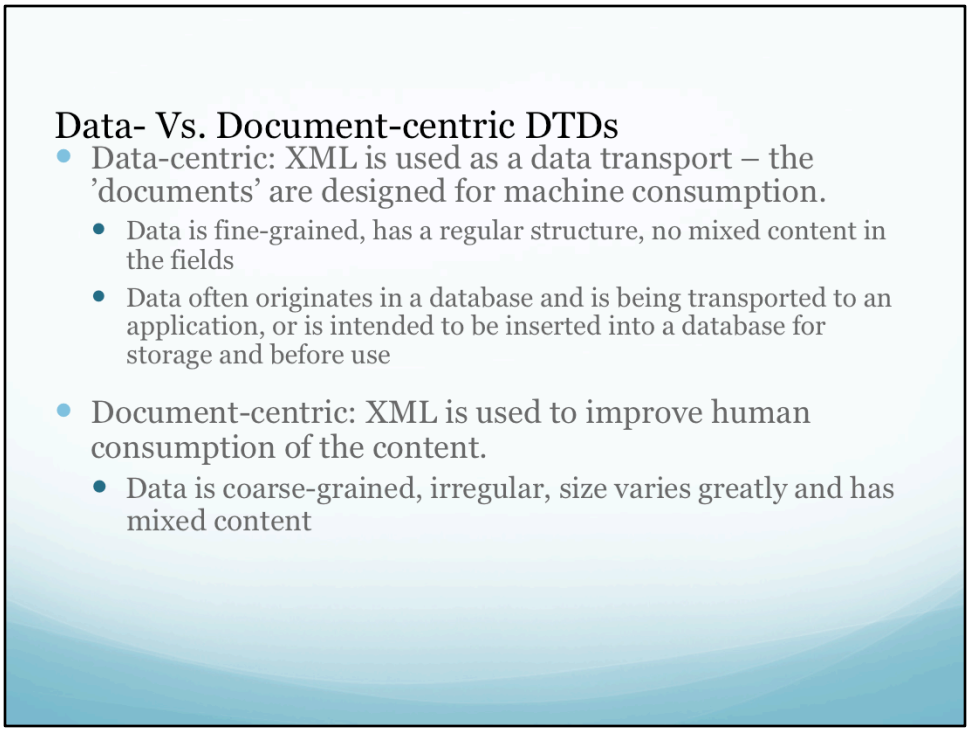
Document Type Definition: In this case I declare that I am using an HTML declaration, for version 4.0 and English conventions.
This is self-describing
So now we have a document that is a collection of data and that is self-describing and that is portable (Unicode based) and that can describe data in tree (graph) structures – is that a database?

It is pretty verbose (at least triple the size) and getting to the data can require text conversion and does require parsing.

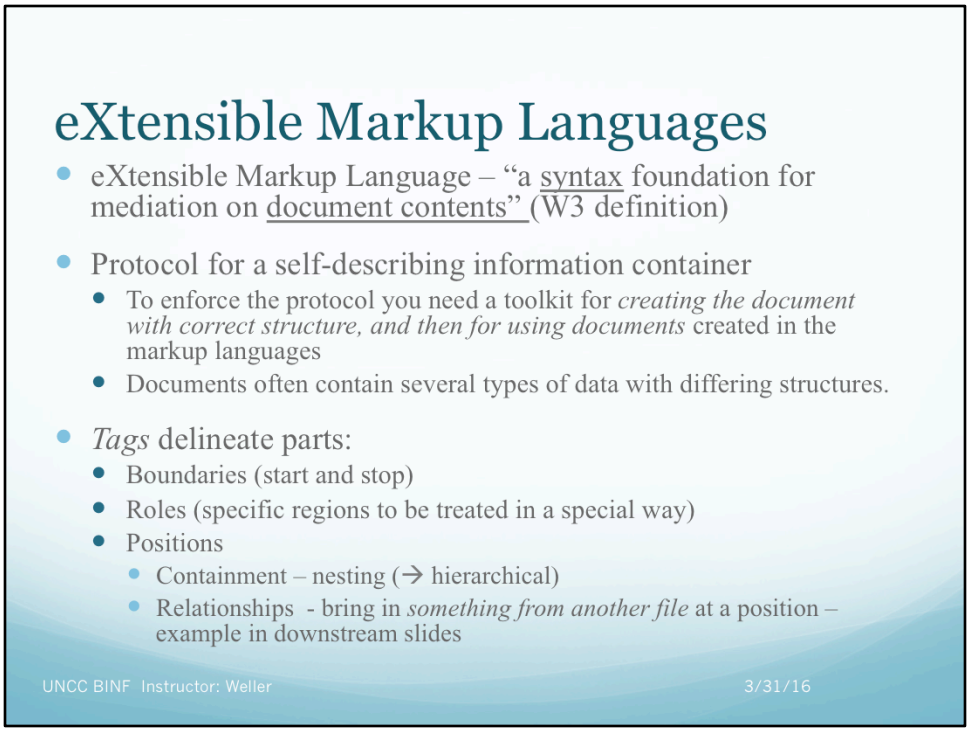
Instead XML and its technologies provide a lot of the technology of a database management system.



An application will look for tags that are meaningful to it, extract the values but not the tags (parse) and then do something with those values – perhaps reformat, perhaps carry out calculations. You might pass values to the application that alters how if filters some of the values, or which tags it selects.

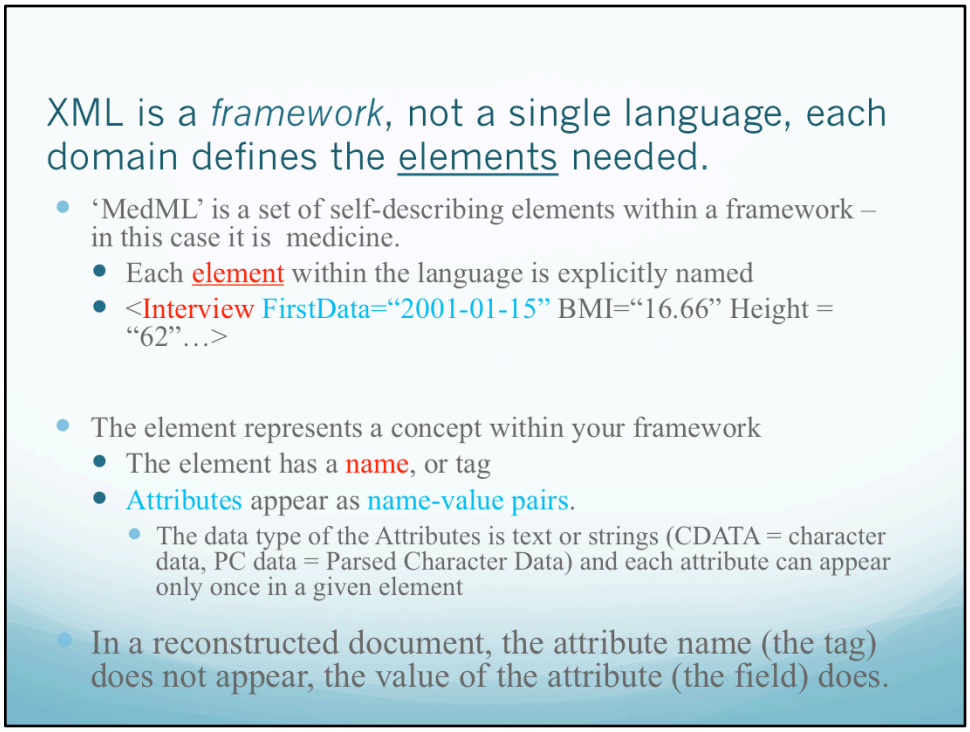


Essentially – the normalization has been pretty complete and the records are pretty complete. We know the expected size of each record.

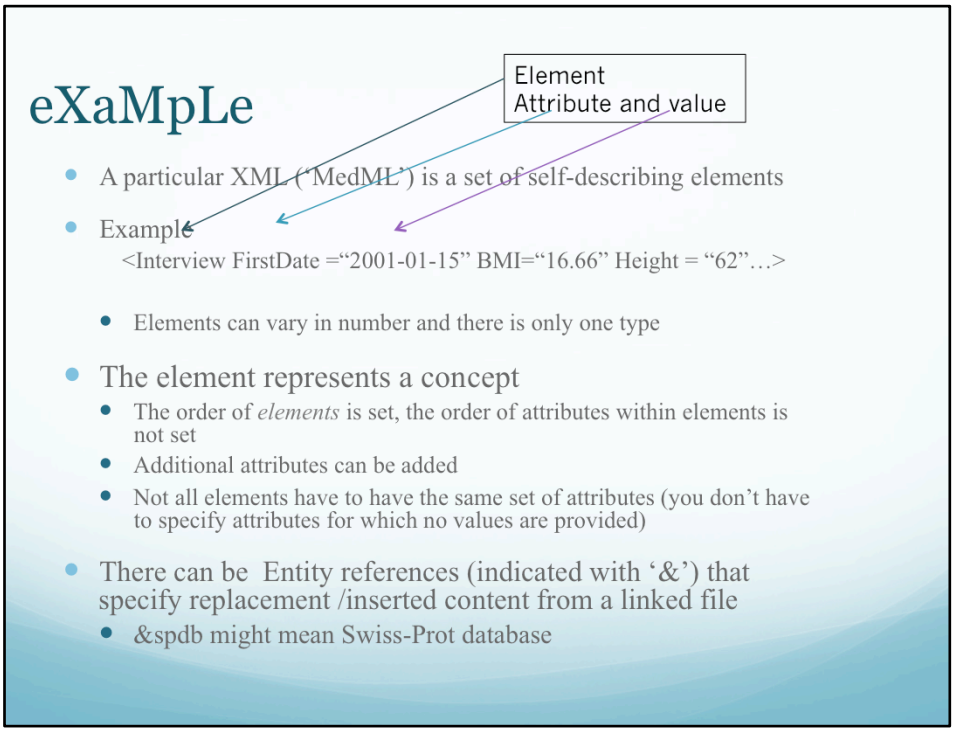


A file is a package of data treated as a continuous unit (a physical structure)
An XML document is a logical structure that integrates the contents of multiple files.
It is allowable to make up a freeform XML, following a basic set of rules prescribing the proper form and using the tags.
It is more common to provide a DTD for document modeling to achieve:

A set of explicit rules (declarations) about how the tags look and what they can contain.
XML Schema is a standard for such document modeling, providing templates as examples to follow for good practice.



In a database the distinction between subelement and attribute is less clear – generally only identifiers are stored as attributes.



The elements resides in some knowledge domain, each element within the language is explicitly named

An XML document is self-describing: when you exchange it the necessary meta-data comes along.

- Representation is flexible and *extensible.*
  - Can change order within level of container
  - Allows multi-valued attributes
  - Omit Attributes
  - Comments can be inserted at any point
- A lot of scientific examples are provided and full descriptions of tags in particular application areas are here:

  http://mariovalle.name/sdm/scientific-data-management.html
- Allows computer processing of file with a generic tool that does not care *at all* about specific content (values), only how they should be treated.

XML DTD structure

- In order to understand the way tags are used and organized in an XML file you provide a second file, which can be either a DTD (file.dtd) or XML Schema (file.xsd)
- Document Type Definition (has a .dtd Extension)
  - Provides the logical model and set of valid tags
    - First line declares the root element

```
<!ELEMENT        Root (child +)>
        <!ELEMENT        Children (sub1, sub 2, sub3,)>
                <!ELEMENT        sub1(#PCDATA)> → leaf
```

- In the XML document you reference the dtd as follows

```
<?xml version =“1.0”?>
<!DOCTYPE sbml SYSTEM “sbml.dtd”>
```

<sbml> corresponds to root element of XML documents that will use the DTD for validation

XML DTD file structure

```
<!ELEMENT ProductList (Product+)>
<!ELEMENT Product (P_code, P_Describe, P_indate?, P_expire*, P_Min?, P_Price)>
<!ELEMENT P_code        (#PCDATA)>
<!ELEMENT P_Describe    (#PCDATA)>
<!ELEMENT P_indate      (#PCDATA)>
<!ELEMENT P_expire      (#PCDATA)>
<!ELEMENT P_Min         (#PCDATA)>
<!ELEMENT P_Price       (#PCDATA)>

<?xml version ="1.0"?>
<!DOCTYPE ProductList SYSTEM "ProductList.dtd">
<ProductList>
        <Product>
                <P_code>23109-AA</P_code>
                <P_Describe>Hammer</Describe>
                <P_price> 6.00</P_price>
        </Product>
        <Product>
                <P_code>40002-AB</P_code>
                <P_Describe>Hex Wrench</Describe>
                <P_price> 12.95</P_price>
        </Product>
</ProductList>
```

#PCDATA is the keyword indicating where the text data will go and indicates a leaf node.

In the xml document the dtd has been *referenced* in the second

- The top has the dtd and the bottom the resulting xml
  - + means the element occurs one or more times (required multivalued element)
  - * means the element occurs zero or more times (optional nultivalued element)
  - ? Means the element is optional (optional single-valued element)
  - If no symbol is given the element must occur exactly once (required single-valued element)
  - a|c gives you the OR option (Minimum price could be a= 0.99 Or c= 6.00)
  - (e) lets you group elements (as in the second line)

The DTD is a set of explicit rules (declarations) about how the tags look and what they can contain.

This has limited expressivity – root, parent, child, mandatory, optional and no good way to state data type or data validation rules.

Also, this has a different syntax from the xml document itself, so it is not self-describing. Since DTD elements must follow the specified order you cannot actually use the unordered elements that were meant to be a strength of the XML approach.

Therefore we move on to the XML Schema construction.

Specifying Attributes in the DTD

- The previous example did not include any attributes for the Elements – what do you do with those?

```
<!ELEMENT height (#PCDATA)>
        <!ATTLIST height
                Dimension   CDATA   #REQUIRED
                accuracy    CDATA   #IMPLIED >
```

- In this case Implied means Optional.
  - CDATA is the type, which is character, or string.
- The syntax is
- <!ATTLIST element_name
  - Attribute_name attribute_type default_value>
- Attribute types include
  - CDATA, ENUMERATION, ID, IDREF, IDREFS, NOTATION, NMTOKEN, NMTOKENS, ENTITY, ENTITIES
- Attribute Default Values include
  - #IMPLIED
  - #REQUIRED
  - #FIXED
  - #LITERAL

Interpretation requires DTDs and Schemas.

Entity References in the DTD

- There are sometimes very long names that you would like to abbreviate – the method for doing this is to declare an entity reference
- Declare the ENTITY as an element in the DTD and then use the abbreviation in the XML document.

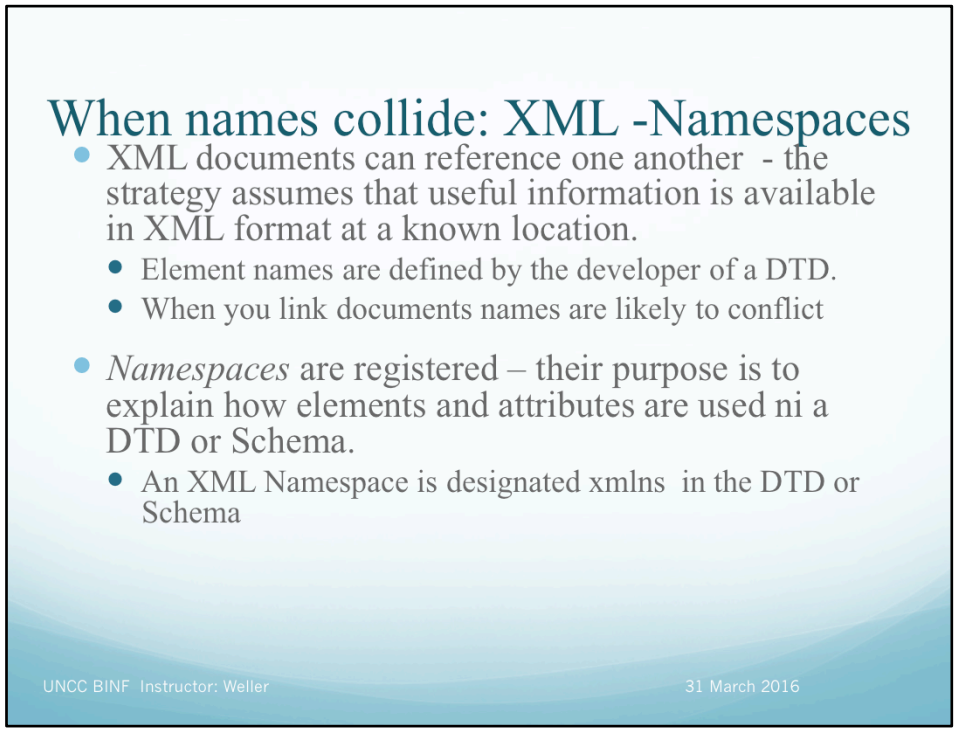
```
<?xml version ="1.0" encoding ="UTF-8"?>
<!DOCTYPE long-words [
<!ELEMENT long-words (word+)>
<!ELEMENT word (#PCDATA)>
<!ENTITY ade "antidepressant">
<!ENTITY pph "pseudohypoparathyroidism">
<!ENTITY pums "pneumonia">
]>
<long-words>
        <word>&ade;</word>
        <word>&pph;</word>
        <word>&pums;</word>
</long-words>
```

The DTD is specified in the DOCTYPE
The element long-words contains at least one word.
Define 3 entities – an abbreviation, followed by the word in quotes

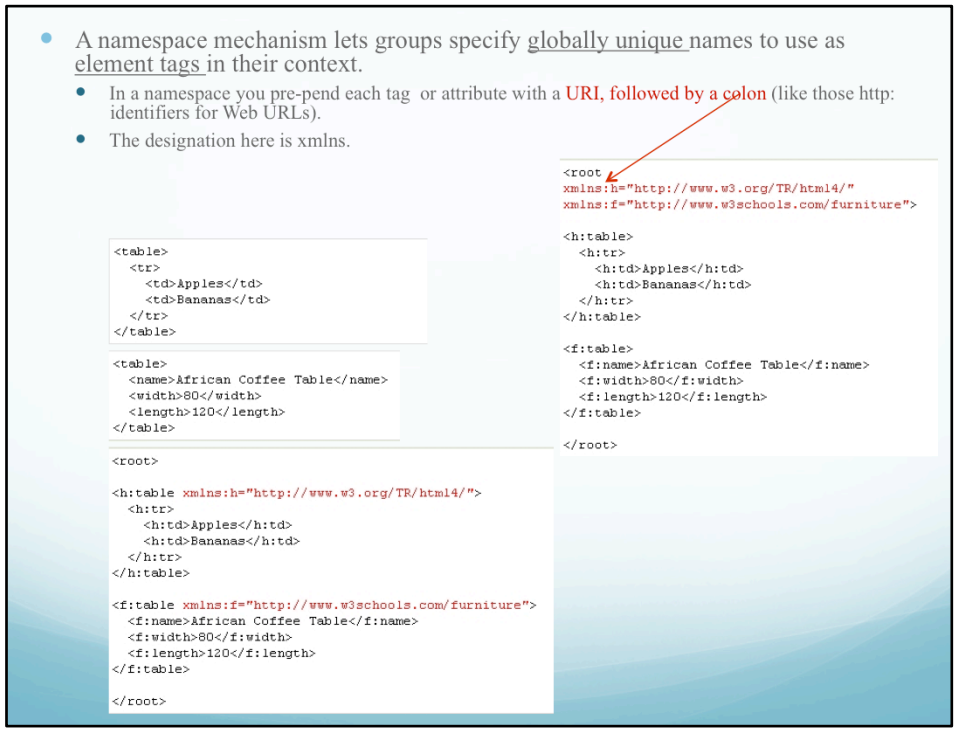
The abbreviation can be used in the XML document – it is preceded by an ampersand and followed by the semi-colon.

If you want to use the reserved characters you do something similar:
< > & ' "

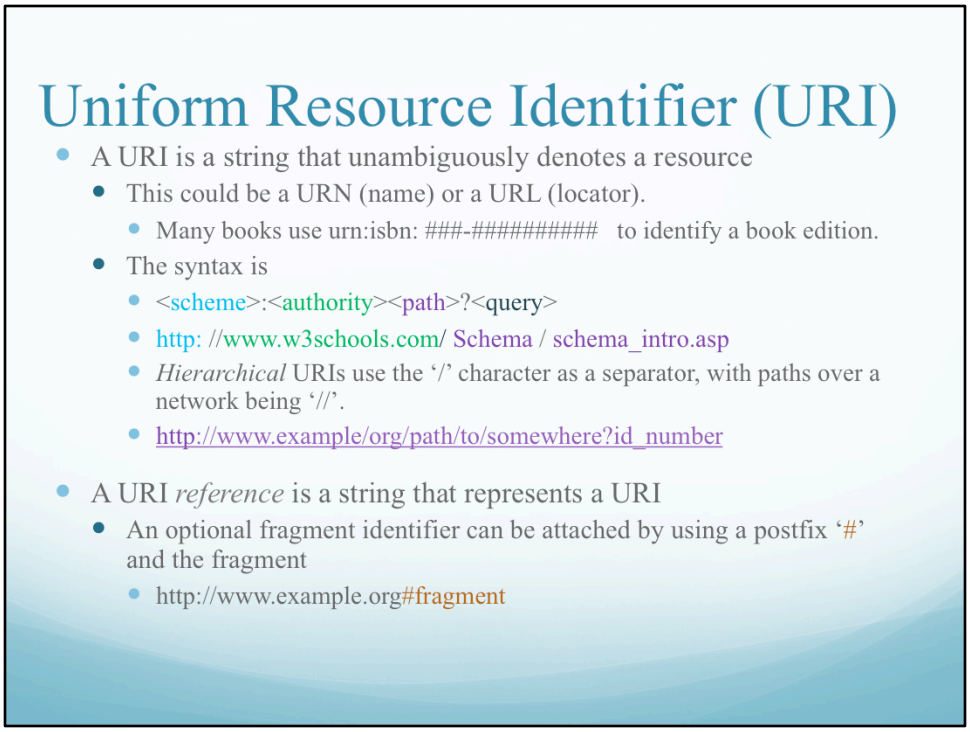
Interpretation requires DTDs and Schemas.



XML is one of a set of semantic Web technologies. Resources are addressable at Web-accessible servers.



You can define an abbreviation for the namespace here we have bound the prefix h to the first URL and f to the second URL.
Use of the prefix is optional in the subelements of each set (unless I am switching out in the middle)



Universal Resource Identifiers

URL is a special case specifying the location of a Web resource, used by a Web browser to find and download the specified resource

The URI identifies but does not necessarily download the resource (a resource might have a URI not the same as the URL)

A Uniform Resource Name omits the protocol specifying how to obtain the

Reading about XML

- Chapter 14 Section 3, Rob and Coronel (8th edition) "Database Systems"
- Online: www.w3schools.com – has tutorials on XML, XML DTD and XML Schema