**FOUNDATIONS OF DATA CURATION**

**WEEK 3. DATA MODELS: TREES**

**1. TEXT AND DOCUMENTS**

**What’s so important about documents?**

The document is the natural unit of textual information. Documents are important because that’s where the information is. Much more information than in databases. That’s where the action is.

Information only has traction on the world when it is communicated in documents. Even databases have effect when a report is generated and read.

That’s where we live, work, and play. We cannot imagine our lives without document-based communication.

Documents and data

Relational databases: when something has a value for an attribute. But most the information is in the text form. Hence, challenges for the relational model: 1) In the text the exact message is not always obvious and requires human analysis. 2) The text itself often needs to be organized and managed, rather than the data in it; most documents are not tabular.

Examples of docs in the pdf file (certificates, notifications, technical specifications, etc.)

Paul Otlet, Vannevar Bush, Douglas Engelbart, Ted Nelson and others thought of the following great promises of digital documents:

• computationally available data items accessible with discipline-specific tools (chemical formulae, proteins, equations, etc.)

• advanced navigation and viewing optimized for domain-specific browsing and analysis,

• typed hypertext linking with links as first class objects,

• data-driven interactive diagrams and graphics

• computable equations,

• supportive ontological inferencing

• thoroughgoing interoperability with other tools

We are not achieving these promise yet. Why? See the topic of the next video.

**2. The Problem**

It’s the same problem as with relational data. The situation circa 1960:tText is stored and processed in radically different ways. Interaction with text is directly via storage. Formal conceptualization of text components is rare and happens mostly in human mind.

This causes operational inefficiencies, lack of functionality, lack of data independence. The promise of digital documents is unfulfilled

Creating complex, high-performance documents is arduous, no sophistication or high-performance information environment. Still the same problem: there are many ways to represent text in documents. Interaction occurs directly with storage structures. Abstraction and indirection are not implemented

The results in: no interoperability and integration of tools, data, applications; documentation and validation are difficult; specialized applications (searching, analysis, etc.) are not supported; schemas are non-existent or unhelpful. Sounds familiar?

Example: electronic publishing in the 1960s. Input file:

.pa odd; .font Times; .size 14;

.it; .ce; .in +5 -5; .sk 2p a; .kp next; .toc include; The Sick Rose[…]

Output:

*The Sick Rose*

**3. The Solution: (1) Descriptive Markup**

The same solution. Recall the previous example. The first improvement there would be:

A macro to abbreviate formatting commands (US Government Printing Office, 1960s) is used in the input:

&format17 = df

“.pa odd;.font Times;.size 14;.it;.ce;.in +5 -5;.sk 3p;.sk 2p a;.kp next;.toc include;”

format17 abstracts the “looks” (can be used for titles, captions, extract labels, etc. But what can be even better? **Identify** not the looks, but **the component itself**!

**A much better improvement**

A macro is defined in the input file to **identify the logical component** of the text itself, not the intended processing or appearance:

.title: The Sick Rose […]

**Abstraction from storage**

Consider this text:

An example of the Tea rose is: <http://www.example.org/rose/hybrid/tea/pink/42>

What if the image moves? Then we use this:  
&rose42 = df http://www.example.org/rose/hybrid/tea/pink/42

The entity name is used in the input:  
An example of the Tea rose is: &rose42;

After processing the output is rendered like this: An example of the Tea rose is: [Image]

Again: **Abstraction and Indirection**

By identifying recurring logical objects (like titles) we abstract away from the varying details of processing and storage. We then exploit abstraction by using indirection: mapping object instances to storage locations, mapping types of objects to processing rules, achieving efficiencies and new functionality.

**Examples of text components**

• Title

• Author

• Date

• Abstract

• Section, subsection, subsubsection

• Section title, subsection title … etc

• Paragraph

• Extract (long quotation)

• Equation

• Diagram

• Footnote

**Genre-specific text components**

Scientific article: title, author, affiliation, address, date submitted, date revised, keywords, abstract, introduction, methodology, results, discussion, conclusion, diagram, equation, plate, graph, chart, bibliography, bibliography item, date

Playscripts: act, scene, stage direction, line, character, cast list

Poetry: title, author, verse, stanza, couplet, line, half-line

Also: 1) Legal and financial documents: contracts, deeds, licenses, writs, tickets, receipts; 2) Office documents: project proposals, reports, performance evaluations, forms, etc.

**An XML example**

<anthology>

<poem>

<heading>THE SICK ROSE</heading>

<stanza>

<line>O Rose thou art sick.</line>

<line>The invisible worm,</line>

<line>That flies in the night</line>

<line>In the howling storm:</line>

</stanza>

<stanza>

<line>Has found out thy bed</line>

<line>Of crimson joy:</line>

<line>And his dark secret love</line>

<line>Does thy life destroy.</line>

</stanza>

</poem>

<!-- more poems go here -->

</anthology>

Other examples (see pdf)

**Descriptive markup describes** the logical components of documents, not processing. Advantages:

Authoring, Editing, Transcribing: simpler composition, supported writing tools, alternative views and links facilitated

Publishing: formatting specified and modified, apparatus automated, output device support enhanced, portability maximized

Retrieval and Analysis: information retrieval and analytical procedures supported

In terms of data curation, descriptive markup makes digital documents: easier to create, maintain, convert (new formats or software), better integrated with workflow, other applications and tools (databases, word processing templates, indexes,), etc.; more accessible to varied audiences; easier to accommodate different technological circumstances (varying hardware, OS, browsers, connectivity (bandwidth)), etc.; easier to accommodate perceptual abilities (blindness, other sight disabilities, dyslexia, etc.)

**4. THE SOLUTION: (2) TREES**

**The OHCO model of text emerges**

Text is an **Ordered Hierarchy** of **Content Objects**:

* content objects = chapters, paragraphs, sentences, stanzas, lines, equations, titles, headings
* hierarchy = sentences inside paragraphs, paragraphs inside sections, sections inside chapters, etc. - nesting with no overlaps
* ordered = objects proceed or follow one another

Most documents can be modeled as trees. A tree, in our specific sense, is a directed acyclic graph with ordered branches and all nodes except one having exactly one parent.

Trees have labelled nodes and possible additional annotations (lang=English)

**Using XML to serialize a tree**

See pdf for XML example. A tree can be serialized with a formal language defined by a context free grammar, such as an **XML language**.

**5. WHY THE SOLUTION WORKS**

**Clarification: these models have two parts**

Relational model = relational data structure + attributes for data values.

Tree model = tree data structure + descriptive markup node labels (“generic identifiers”) such as “stanza”.

We will now refer to the tree model as the ”tree/DM” model.

Why the tree/DM model works so well for data management? Same reason as for RM: Abstraction and Indirection

**Abstraction**

Both models focus on the data itself, separate from storage and processing.

This explicit identification of data attributes in one case, and logical text objects in the other, brings enormous new functionality and efficiency

Once again: **Indirection**

Both models support an indirect relationship to storage and processing, but in practice the emphasis is often different:

For the relational model abstracting away from storage is dominant. For the tree/DM model abstracting away from processing is dominant

In both cases the separation is mediated by a mapping:

*logical schema to physical schema* in the case of the relational model

*text component (type) to processing instructions* in the case of trees/DM

**Formal vs colloquial understanding of these data models**

Although in the relational model we commonly think of **attributes** as representing dyadic properties or relationships in the world, technically they are **names for domains of values**.

Similarly in the tree/DM model we think of these node labels as indicating the kind of enclosed text object (stanza, formula, etc), but in the model they are really simply names.

It is our colloquial understanding of these models that enables us to actually use them to secure useful abstraction and data management.

This may seem a small point now, but it motivates a further advance in abstraction, as we will see when we discuss ontologies.

**6. IMPLEMENTING THE SOLUTION: XML**

**XML** is a **schema language** (or a meta-grammar). An XML document uses a defined set of delimiters with arbitrary element names and attribute value pairs to nest spans of text.

A well-formed XML document fits a formal grammar (along with other constraints) that ensures the document can be parsed as a tree by an XML parser.

NB: a well-formed XML document need not have a schema that defines the element vocabulary and grammar.

it may use arbitrary element, attribute, and value names and arrange text objects in any way that does not violate the tree data structure.

**The two main things in the XML world**

**Schemas** [e.g. Document Type Definitions (DTDs)]

One for each document type (class, category, genre). Defines a markup language for document structures by specifying its vocabulary and syntax (grammar). Describes elements for a particular type of document, patterns for these elements, additional information about the elements.

**Document Instances**

Particular documents, marked up with a markup language that meet well-formedness constraints, and, perhaps, also meets the constraints of a relevant schema.

**A well-formed XML document**

<anthology>

<poem>

<heading>THE SICK ROSE</heading>

<stanza>

<line>O Rose thou art sick.</line>

<line>The invisible worm,</line>

<line>That flies in the night</line>

<line>In the howling storm:</line>

</stanza>

<stanza>

<line>Has found out thy bed</line>

<line>Of crimson joy:</line>

<line>And his dark secret love</line>

<line>Does thy life destroy.</line>

</stanza>

</poem>

<!-- more poems go here -->

</anthology>

**Schemas for Trees**

XML **Document Type Definition (DTD)**, defining an XML document type

<!ELEMENT anthology (poem+)> #+ means one or more

<!ELEMENT poem (title?, stanza+)> #? means 0 or more

<!ELEMENT title (#PCDATA) #PCDATA=Parsed Character Data (mixed content)

<!ELEMENT stanza (line+) >

<!ELEMENT line (#PCDATA) >

This schema specifies element vocabulary and grammar

The **DTD** schema language is based on (extended) **Backus Naur Form (BNF) grammars**

Some XML schema languages provide additional validation and constraints on content, including data types.

*Another DTD*

<!ELEMENT poem (title, author? verse) >

<!ATTLIST poem

editor CDATA #REQUIRED>

<!ELEMENT verse (stanza+)>

<!ELEMENT stanza (line+)>

<!ELEMENT title (#PCDATA | italic | persname)\*>

<!ELEMENT author (#PCDATA)

<!ATTLIST author

sex (male | female) #IMPLIED

dates CDATA #IMPLIED

bio IDREF #IMPLIED>

<!ELEMENT line (#PCDATA | italic | persname)

<!ATTLIST line

lang CDATA "ENGLISH">

<!ELEMENT italic (#PCDATA)>

<!ELEMENT persname(#PCDATA)>

*Another XML document, with attribute/value pairs*

<!DOCTYPE text SYSTEM ”poem.dtd">

<poem editor=“Sara Porter”>

<title>Terence</title>

<author person=“N320”>A. E. Houseman</author>

<verse>

<stanza>

<line>Terence this is stupid stuff </line>

<line>you eat your victuals fast enough</line>

<line>there can’t be much amiss ‘tis clear</line>

</stanza>

<stanza> […]

<line lang=“latin”>The old lie: </line>

<line> in vino veritas </line>

</stanza>

</verse>

</poem>

**Valid XML Documents**

A document instance is valid with respect to some schema if it conforms to the declarations in that schema, which is to say, matches the grammar and other constraints.

. . . nothing out of place, nothing missing, no attributes with values the wrong type, no wrong references (that fail), and so on.

A validating parser applies an XML schema to an XML document and determines whether or not the document conforms to the constraints specified in the schema.

All **valid XML documents are well-formed**, but not vice versa.

**XML Processing** (see pdf for a diagram)

Some XML tools and schema languages

**Two important XML transformation tools**

XSLT: “a language for transforming XML documents into other XML documents” https://www.w3.org/TR/xslt

Xquery: “a standardized language for combining documents, databases, Web pages, and almost anything else” <https://www.w3.org/XML/Query/>

**Two other XML schema languages**

**XML Schema (XSD)**: “the XML Schema Definition Language…offers facilities for describing the structure and constraining the contents of XML documents”

http://www.w3.org/XML/Schema

A more complex schema language than DTDs, but does more than validate. Written in XML. Common for business applications

**RelaxNG**: “a schema language for XML” http://relaxng.org/

Similar expressiveness to XSD, with simpler syntax. Less commercial application support

**Some important XML languages for documents.**

You should be familiar with these. Please explore the websites.

**XHTML**: “a family of current and future document types and modules that reproduce, subset, and extend HTML”  
https://www.w3.org/TR/xhtml1/

**TEI**: Text Encoding Initiative – “a standard for the representation of texts in digital form” http://www.tei-c.org/index.xml

**JATS**: Journal Article Tag Suite “defines a set of XML elements and attributes for tagging journal articles”  
https://jats.nlm.nih.gov/

**XML Languages and Interchange**

There are an enormous number of XML markup languages.

See https://en.wikipedia.org/wiki/List\_of\_XML\_markup\_languages

Most of these languages are not for text, but are interchange and preservation formats for structured data.

XML is an important preservation format. It uses simple ASCII text with inline tags, can be parsed without a schema, and if a schema is available can be validated ensuring a correct grammar (nothing missing, nothing out of place) and data typing.