

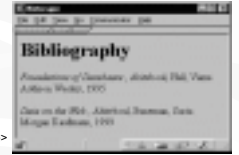
XML & DTD

CPS 196.3
Introduction to Database Systems

From HTML to XML (eXtensible Markup Language)

- ❖ HTML describes the presentation of the content

```
<h1>Bibliography</h1>  
<p><i>Foundations of Databases</i>  
Abiteboul, Hull, and Vianu  
<br>Addison Wesley, 1995...
```



- ❖ XML describes only the content

```
<bibliography>  
<book>  
  <title>Foundations of Databases</title>  
  <author>Abiteboul</author>  
  <author>Hull</author>  
  <author>Vianu</author>  
  <publisher>Addison Wesley</publisher>  
  <year>1995</year>  
</book>  
</bibliography>
```

- ❖ Separation of content from presentation simplifies content extraction and allows the same content to be presented easily in different looks

Other nice features of XML

- ❖ Portability: Just like HTML, you can ship XML data across any platforms
 - Relational data requires heavy-weight protocols, e.g., JDBC
- ❖ Flexibility: You can represent any information (structured, semi-structured, documents, ...)
 - Relational data is best suited for structured data
- ❖ Extensibility: Since data describes itself, you can change the schema easily
 - Relational schema is rigid and difficult to change

XML terminology

- ❖ Tag names: `book`, `title`, ...
- ❖ Start tags: `<book>`, `<title>`, ...
- ❖ End tags: `</book>`, `</title>`, ...
- ❖ An element is enclosed by a pair of start and end tags: `<book>...</book>`
 - Elements can be nested:
`<book>...<title>...</title>...</book>`
 - Empty elements: `<is_textbook></is_textbook>`
 - Can be abbreviated: `<is_textbook/>`
- ❖ Elements can also have attributes: `<book ISBN="..." price="80.00">`

Well-formed XML documents

- A well-formed XML document
- ❖ Follows XML lexical conventions
 - Wrong: `<section>We show that x < 0...</section>`
 - Right: `<section>We show that x < 0...</section>`
 - Other special entities: `>` becomes `>`; and `&` becomes `&`;
- ❖ Contains a single root element
- ❖ Has tags that are properly matched and elements that are properly nested
 - Right:
`<section>...<subsection>...</subsection>...</section>`
 - Wrong:
`<section>...<subsection>...</section>...</subsection>`

More XML features

- ❖ Comments: `<!-- Comments here -->`
- ❖ CDATA: `<![CDATA[Tags: <book>,...]]>`
- ❖ ID's and references

```
<person id="o12"><name>Homer</name>...</person>  
<person id="o34"><name>Marge</name>...</person>  
<person id="o56" father="o12" mother="o34"><name>Bart</name>...</person>...
```
- ❖ Namespaces allow external schemas and qualified names

```
<book xmlns:myCitationStyle="http://...mySchema">  
  <myCitationStyle:title>...</myCitationStyle:title>  
  <myCitationStyle:author>...</myCitationStyle:author>...  
</book>
```
- ❖ Processing instructions for apps: `<? ...java applet... ?>`
- ❖ And more...

Valid XML documents

7

- ❖ A valid XML document conforms to a Document Type Definition (DTD)
 - A DTD is optional
- ❖ A DTD specifies
 - A grammar for the document
 - Constraints on structures and values of elements, attributes, etc.
- ❖ Example


```
<!DOCTYPE bibliography [
  <!ELEMENT bibliography (book+)>
  <!ELEMENT book (title, author*, publisher?, year?, section*)>
  <!ATTLIST book ISBN CDATA #REQUIRED>
  <!ATTLIST book price CDATA #IMPLIED>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT publisher (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
  <!ELEMENT section (title, (#PCDATA)?, section)*>
]>
```

DTD explained

8

- ```
<!DOCTYPE bibliography [
 <!ELEMENT bibliography (book+)>
 <!ELEMENT book (title, author*, publisher?, year?, section*)>
 <!ATTLIST book ISBN ID #REQUIRED>
 <!ATTLIST book price CDATA #IMPLIED>
```
- ↳ bibliography is the root element of the document
  - ↳ One or more bibliography consists of a sequence of one or more book elements
  - ↳ Zero or more book consists of a title, zero or more authors, an optional publisher, and zero or more sections, in sequence
  - ↳ book has a required ISBN attribute which is a unique identifier
  - ↳ book has an optional (#IMPLIED) price attribute which contains character data
- ```
</book>
</bibliography>
```
- Other attribute types include IDREF (reference to an ID), IDREFS (space-separated list of references), enumerated list, etc.

DTD explained (cont'd)

9

- ```
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```
- ↳ title, author, publisher, and year all contains parsed character data (#PCDATA)
- ```
<!ELEMENT section (title, (#PCDATA)?, section)*>
```
- ↳ Each section starts with a title, followed by some optional text and then zero or more subsections
- ```
</section>
</bibliography>
```
- PCDATA is text that will be parsed (<...> will be treated as a markup tag and &lt; etc. will be treated as entities); CDATA is unparsed character data
- ```
<section><title>Introduction</title>
In this section we introduce XML and DTD.
<section><title>XML</title>
XML stands for...
</section>
<section><title>DTD</title>
<section><title>Definitions</title>
DTD stands for...
</section>
<section><title>Usage</title>
You can use DTD to...
</section>
</section>
```

Using DTD

10

- ❖ DTD can be included in the XML source file


```
<?xml version="1.0"?>
<!DOCTYPE bibliography [
  ...
]>
<bibliography>
  ...
</bibliography>
```
- ❖ DTD can be external


```
<?xml version="1.0"?>
<!DOCTYPE bibliography SYSTEM "../dtds/bib.dtd">
<bibliography>
  ...
</bibliography>
```
- ❖ DTD can be external


```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  ...
</html>
```

Why use DTD's?

11

- ❖ Benefits of using DTD
 - DTD can serve as a schema for the XML data
 - Guards against errors
 - Helps with processing
 - DTD facilitates information exchange
 - People can agree to use a common DTD to exchange data (e.g., XHTML)
- ❖ Benefits of not using DTD
 - Unstructured data is easy to represent
 - Overhead of DTD validation is avoided

XML versus relational data

12

- | Relational data | XML data |
|-------------------------------------------------------------|-------------------------------------------------------------------|
| ❖ Schema is always fixed in advance and difficult to change | ❖ Well-formed XML does not require predefined, fixed schema |
| ❖ Simple, flat table structures | ❖ Nested structure; ID/IDREF(S) permit arbitrary graphs |
| ❖ Ordering of rows and columns is unimportant | ❖ Ordering forced by document format; may or may not be important |
| ❖ Data exchange is problematic | ❖ Designed for easy exchange |
| ❖ "Native" support in all serious commercial DBMS | ❖ Implemented as an "add-on," often on top of relations |