An introduction to XML

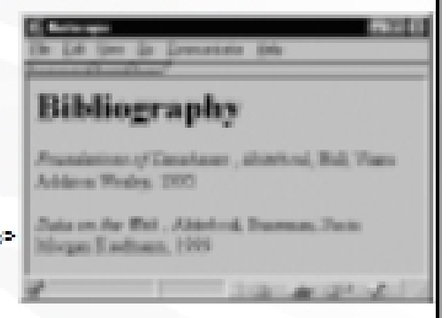
CPS 296.1

Topics in Database Systems

From HTML to XML

- 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 allows the 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 its own schema, you can change it easily
  - Relational schema is rigid and difficult to change
- "Publishability": We need new data models, query languages, query processing and optimization techniques

XML terminology

```
<bibliography>
  <book ISBN="..." price="80.00">
    <title>Foundations of Databases</title>
    <is_textbook>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
    <publisher>Addison Wesley</publisher>
    <year>1995</year>
  </book>...
</bibliography>
```

- Tag names: book, title, author
- Start tags: <book>, <title>, <author>
- End tags: </book>, </title>, </author>
- 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>
- 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="p12"><name>Homer</name>...</person>
<person id="p34"><name>Marge</name>...</person>
<person id="p56" father="p12" mother="p34"><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

- 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.

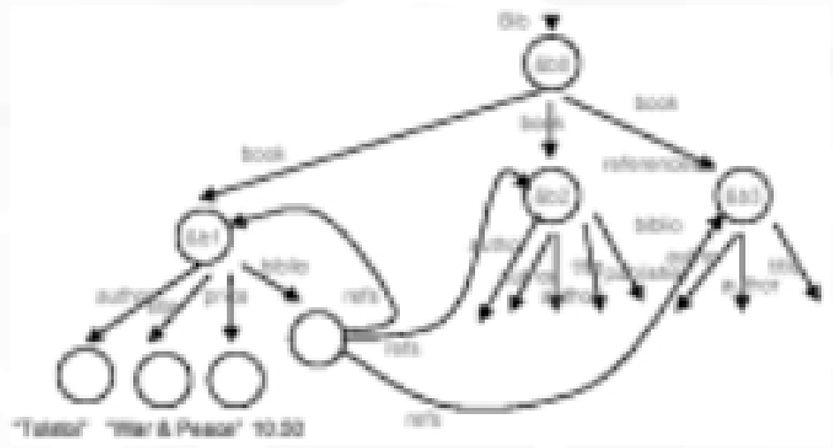
```
<?XML version="1.0"?>
<!DOCTYPE book [
        <!ELEMENT book (title, author*, publisher?, section+)>
        <!ATTLIST book ISBN CDATA #REQUIRED>
        <!ATTLIST book year CDATA #IMPLIED>
        <!ELEMENT title (#PCDATA)>
        <!ELEMENT author (#PCDATA)>
        <!ELEMENT section (#PCDATA | title | section)*>
]>
...
```


Data models for XML

- Graph and tree models used in research
  - Semistructured model of Lore and TSIMMIS (Stanford)
  - Ordered tree model of YAT (INRIA)
- Document Object Model (DOM)
  - Object-oriented programming interface for XML
- XML Infoset
- Data models for various XML query languages
  - Data model defined by XML Query Working Group for XPath and XQuery

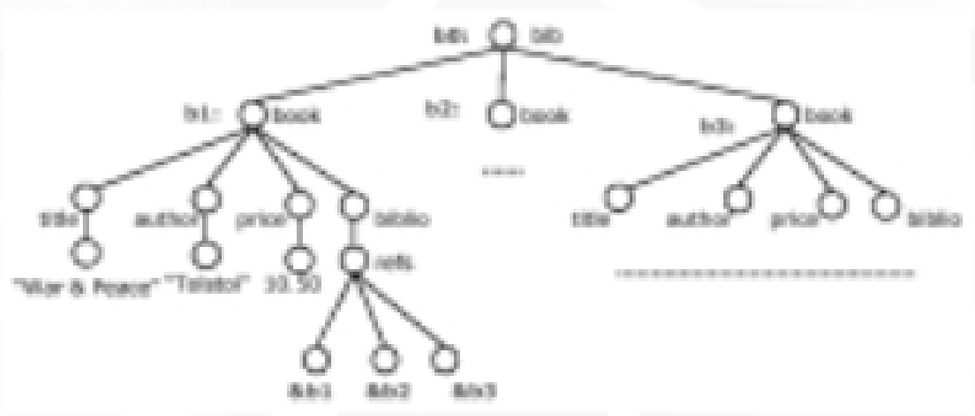
Semistructured data model of Lore

- Graph-based, unordered, edge-labeled



Ordered tree model of YAT

- Tree-based, ordered, node-labeled, with references



Data model of XML Query WG

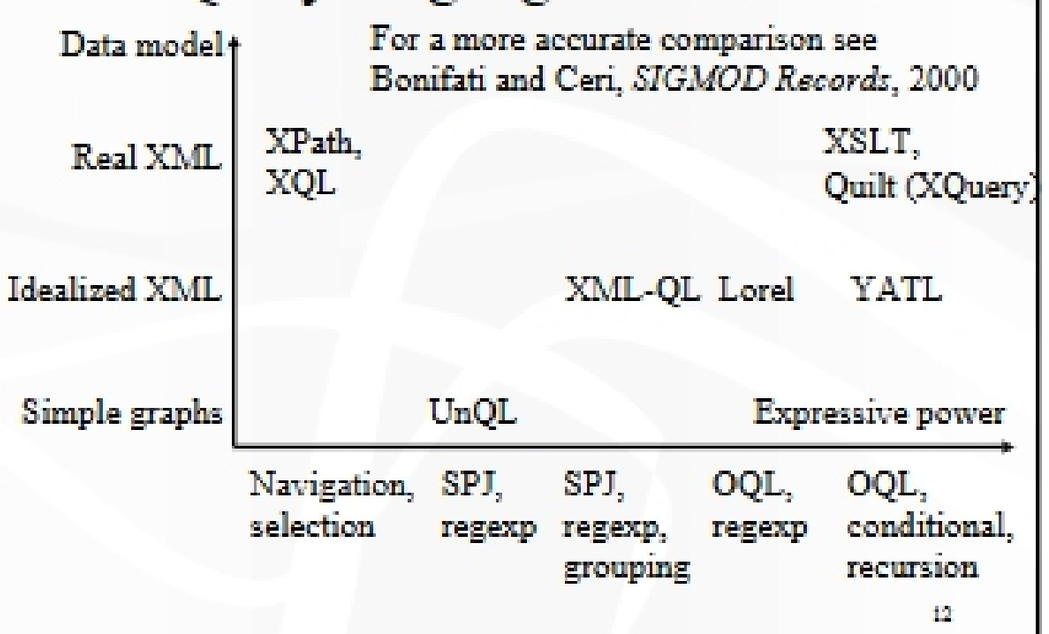
- Conceptually, also tree-based, ordered, and with references
- Functional notation (think ML); no explicit data structure
- Example

```
<book price="10.50">
  <title>...</title><author>...</author><author>...</author>...
</book>
```

→

e_1 = elemNode(qnameValue(null, "book"), { a_1 }, [e_2, e_3, e_3])
a_1 = attrNode(qnameValue(null, "price"), decimalValue(10.50))
e_2 = elemNode(qnameValue(null, "title"), …)
e_3 = elemNode(qnameValue(null, "author"), …)
e_4 = elemNode(qnameValue(null, "author"), …)

Query languages for XML



XML-QL

- Deutsch, Fernandez, Florescu, Levy, and Suciu. "XML-QL: A Query Language for XML." *WWW*, 1999
- Data model: a (totally) ordered or a (totally) unordered graph
- Query language
  - WHERE clause to bind variables and test predicates
  - CONSTRUCT clause to build output XML structures
- Features
  - XML patterns, regexp path expressions
  - Joins on multiple input sources
  - Skolem functions for grouping

XML-QL: XML patterns

- Retrieve the titles of the books written by Abiteboul before 2000

Scan bib.xml: match the pattern to obtain all ($y, $isbn, $t) bindings

Select those that pass the predicate

Construct output for each ($y, $isbn, $t) binding obtained in WHERE

```
WHERE
  <bib>
    <book year=$y ISBN=$isbn>
      <title>$t</title>
      <author><lastname>Abiteboul</lastname></author>
    </book>
  </bib> IN "bib.xml",
  $y < 2000
CONSTRUCT
  <resultBook ISBN=$isbn>
    <resultTitle>$t</resultTitle>
  </resultBook>
```


XML-QL: joins

- Retrieve all reviews for the books written by Abiteboul

```
WHERE
  <bib>
    <book ISBN=$isbn>
      <author><lastname>Abiteboul</lastname></author>
    </book>
  </bib> IN "bib.xml",
  <reviews>
    <review ISBN=$isbn></review> ELEMENT_AS $e
  </reviews> in "reviews.xsl"
CONSTRUCT
  $e
```


XML-QL: outerjoins (slide 1)

- Retrieve the titles of the books written by Abiteboul, together with their reviews, if any

```
WHERE
  <bib>
    <book ISBN=$isbn>
      <title>$t</title>
      <author><lastname>Abiteboul</lastname></author>
    </book>
  </bib> IN "bib.xml",
  <reviews>
    <review ISBN=$isbn></review> ELEMENT_AS $e
  </reviews> in "reviews.xsl"
CONSTRUCT
  <resultBookWithReview ISBN=$isbn>
    <title>$t</title>
    $e
  </resultBookWithReview>
```

If a book has no review, it will not be in the output

Book title appears multiple times if there are multiple reviews

What is wrong?

XML-QL: outerjoins (slide 2)

- Use nested queries with outerjoin semantics

Okay for subquery to return empty result

```
WHERE
  <bib>
    <book ISBN=$isbn>
      <title>$t</title>
      <author><lastname>Abiteboul</lastname></author>
    </book>
  </bib> IN "bib.xml",
CONSTRUCT
  <resultBook ISBN=$isbn>
    <title>$t</title>
    (WHERE
      <reviews>
        <review ISBN=$isbn></review> ELEMENT_AS $e
      </reviews> IN "reviews.xsl"
     CONSTRUCT
      $e)
  </resultBook>
```


XML-QL: regexp, metadata queries

- What kind of elements are found in the content of the element corresponding to the book with ISBN of 10?

Regexp that matches any sequence of elements

```
WHERE
  <$*>
    <book ISBN=$isbn>
      <$tagname></>
    </book>
  </> IN "bib.xml",
CONSTRUCT
  <result>$tagname</result>
```