Introduction to XML

Yanlei Diao
UMass Amherst
April 19, 2007

Structure in Data Representation

- Relational data is highly structured
  - structure is defined by the schema
  - good for system design
  - good for precise query semantics / answers
- Structure can be limiting
  - data exchange hard: integration of different schema
  - authoring is constrained: schema-first
  - querying constrained: must know schema
  - changes to structure not easy

Data Integration

1. Find all departments whose total employee salaries exceed 1% of the budget of the company.
2. Find names of employees with the top sales record last month.
Integration of Text and Structured Data

Structured data - Databases

Semistructured Data

WWW

Unstructured Text - Documents

Need for A New Data Model

Loose (and rich) structure
- Integration of structured, but heterogeneous data sources
- Evolving, unknown, or irregular structure
- Textual data with tags and links
- Combination of data models

XML: Universal Data Exchange Format

- XML is the confluence of many factors:
  - Databases needed a more flexible interchange format.
  - Data needed to be generated and consumed by applications.
  - The Web needed a more declarative format for data.
  - Documents needed a mechanism for extended tags.
- XML was originally proposed for online publishing, is becoming the wire format for data exchange.
- W3C Recommendation:
  http://www.w3.org/TR/REC-xml/
From HTML to XML

HTML describes the presentation.

**HTML**

```html
<h1>Bibliography</h1>
<p><i>Foundations of Databases</i>  
Abiteboul, Hull, Vianu  
Addison Wesley, 1995</p>
<p><i>Data on the Web</i>  
Abiteboul, Buneman, Suciu  
Morgan Kaufmann, 1999</p>
```

**XML**

```xml
<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>
```

XML describes the content!
XML: Syntax & Typing

XML Syntax

- Tags: book, title, author, ...
  - start tag: <book>
  - end tag: </book>
  - elements are nested
  - empty element: <red></red>, abbrev: <red/>)
- An XML document: single root element

An XML document is well formed if it has matching tags

Attributes are alternative ways to represent data.
**XML Syntax**

```
<person id="o555"> <name>Jane</name> </person>
<person id="o456"> <name>Mary</name> </person>
<children idref="o123 o555"/>
<person id="o123" mother="o456"> <name>John</name> </person>
```

Oids and references in XML are just syntax.

**XML Semantics: a Tree!**

Order matters! IDREF will turn it to a graph.

**XML Data**

- XML is self-describing
- Schema elements become part of the data
  - Relational schema: `persons(name,phone)`
  - In XML `<persons>`, `<name>`, `<phone>` are part of the data, and are repeated many times
- Consequence: XML is much more flexible

Some real data:
http://www.cs.washington.edu/research/xmldatasets/
Relational Data as XML

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>3634</td>
</tr>
<tr>
<td>Sue</td>
<td>6343</td>
</tr>
<tr>
<td>Dick</td>
<td>6363</td>
</tr>
</tbody>
</table>

XML:

```xml
<peron>
  <row>
    <name>John</name>
    <phone>3634</phone>
  </row>
  <row>
    <name>Sue</name>
    <phone>6343</phone>
  </row>
  <row>
    <name>Dick</name>
    <phone>6363</phone>
  </row>
</peron>
```

XML is Semi-structured Data

- Missing attributes:
  - `<data>`
  - `<person>`: `<name>John</name>`
    - `<phone>1234</phone>`
  - `<person>`: `<name>Joe</name>`
    - `<phone>`

- Could represent in a table with nulls

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1234</td>
</tr>
<tr>
<td>Joe</td>
<td></td>
</tr>
</tbody>
</table>

- Repeated attributes

```
<peron>
  <row>
    <name>Mary</name>
    <phone>2345</phone>
    <phone>3456</phone>
  </row>
</peron>
```

- Impossible in tables:
  - nested collections (non 1NF)
XML is Semi-structured Data

- Attributes with different types in different objects
- Mixed content:
  - `<db>` contains both `<book>`s and `<publisher>`s

Data Typing in XML

- Data typing in the relational model: schema
- Data typing in XML
  - Much more complex
  - Typing restricts valid trees that can occur
    - Theoretical foundation: tree languages
  - Practical methods:
    - DTD (Document Type Definition)
    - XML Schema

Document Type Definitions (DTD)

- Part of the original XML specification
- To be replaced by XML Schema
  - Much more complex
- An XML document may have a DTD
- XML document:
  - well-formed = if tags are correctly closed
  - Valid = if it has a DTD and conforms to it
- Validation is useful in data exchange
**DTD Example**

```xml
<DOCTYPE company [
  <!ELEMENT company ((person|product*))>
  <!ELEMENT person (ssn, name, office, phone?)>
  <!ELEMENT ssn (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT office (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
  <!ELEMENT product (pid, name, description?)>
  <!ELEMENT pid (#PCDATA)>
  <!ELEMENT description (#PCDATA)>
]>...
```

**Example of valid XML document:**

```xml
<company>
  <person>
    <ssn>123456789</ssn>
    <name>John</name>
    <office>B432</office>
    <phone>1234</phone>
  </person>
  <person>
    <ssn>987654321</ssn>
    <name>Jim</name>
    <office>B123</office>
  </person>
  <product>...
  </product>
</company>
```

**DTD: The Content Model**

- Content model:
  - Complex = a regular expression over other elements
  - Text-only = #PCDATA
  - Empty = EMPTY
  - Any = ANY
  - Mixed content = (#PCDATA | A | B | C)*
DTD: Regular Expressions

sequence

XML

optional

Kleene star

Alternation

Attributes in DTDs

Attributes in DTDs
Attributes in DTDs

Types:
- CDATA = string
- ID = key
- IDREF = foreign key
- IDREFS = foreign keys separated by space
- (Monday | Wednesday | Friday) = enumeration

Kind:
- #REQUIRED
- #IMPLIED = optional
- value = default value
- value #FIXED = the only value allowed

Using DTDs

- Must include in the XML document
- Either include the entire DTD:
  - <!DOCTYPE rootElement [ ...... ]>
- Or include a reference to it:
  - <!DOCTYPE rootElement SYSTEM "http://www.mydtd.org">
- Or mix the two... (e.g. to override the external definition)
XML Schema

- DTDs capture grammatical structure, but have some drawbacks:
  - Not themselves in XML, inconvenient to build tools
  - Don't capture database datatypes' domains
  - No way of defining OO-like inheritance...

- XML Schema addresses shortcomings of DTDs
  - XML syntax
  - Subclassing
  - Domains and built-in datatypes
  - min. and max # of occurrences of elements
  - http://www.w3.org/XML/Schema

Basics of XML Schema

- Need to use the XML Schema namespace (generally named xsd)
- simpleTypes are a way of restricting domains on scalars
  - Can define a simpleType based on integer, with values within a particular range
- complexTypes are a way of defining element structures
  - Basically equivalent to !ELEMENT, but more powerful
  - Specify sequence, choice between child elements
  - Specify minOccurs and maxOccurs (default 1)
- Must associate an element/attribute with a simpleType, or an element with a complexType

Simple Schema Example

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="mastersthesis" type="ThesisType"/>
  <xs:complexType name="ThesisType">
    <xs:attribute name="mdate" type="xs:dateTime"/>
    <xs:attribute name="key" type="xs:string"/>
    <xs:attribute name="advisor" type="xs:string"/>
    <xs:sequence>
      <xs:element name="author" type="xs:string"/>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="year" type="xs:integer"/>
      <xs:element name="school" type="xs:string"/>
      <xs:element name="committeemember" type="CommitteeType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```