Introduction to Semistructured Data and XML

Based on slides by Dan Suciu
University of Washington

Overview

- From HTML to XML
- DTDs
- Querying XML: XPath
- Transforming XML: XSLT

How the Web is Today

- HTML documents
  - often generated by applications
  - consumed by humans only
  - easy access: across platforms, across organizations
- No application interoperability:
  - HTML not understood by applications
    - screen scraping brittle
  - Database technology: client-server
    - still vendor specific
**New Universal Data Exchange Format: XML**

- A recommendation from the W3C
  - XML = data
  - XML generated by applications
  - XML consumed by applications
  - Easy access: across platforms, organizations

---

**Paradigm Shift on the Web**

- From documents (HTML) to data (XML)
- From information retrieval to data management
- For databases, also a paradigm shift:
  - from relational model to semistructured data
  - from data processing to data/query translation
  - from storage to transport

---

**Semistructured Data**

Origins:
- Integration of heterogeneous sources
- Data sources with non-rigid structure
  - Biological data
  - Web data
The Semistructured Data Model

Syntax for Semistructured Data


| author: &o52 "Abiteboul",
| author: &o96 { firstname: &o243 "Victor",
| lastname: &o206 "Vianu"},
| title: &o93 "Regular path queries with constraints",
| references: &o12,
| references: &o24,
pages: &o25 { first: &o64 122, last: &o92 133} |

Observe: Nested tuples, set-values, oids!

Syntax for Semistructured Data

May omit oids:

{ paper: { author: "Abiteboul",
author: { firstname: "Victor",
lastname: "Vianu"},
title: "Regular path queries ...",
page: { first: 122, last: 133 } } }
Characteristics of Semistructured Data

- Missing or additional attributes
- Multiple attributes
- Different types in different objects
- Heterogeneous collections

Self-describing, irregular data, no a priori structure

Comparison with Relational Data

<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>

From HTML to XML

Bibliography

Foundations of Databases, Abiteboul, Hull, Vass
Addison Wesley, 1995

Data on the Web, Abiteboul, Duenas, Gus
Morgan Kaufmann, 1999

HTML describes the presentation
<h1> Bibliography </h1>

<p> <i> Foundations of Databases </i>  
Abiteboul, Hull, Vianu  
<br> Addison Wesley, 1995 </p>

<p> <i> Data on the Web </i>  
Abiteoul, Buneman, Suciu  
<br> Morgan Kaufmann, 1999 </p>

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

XML describes the content

- A W3C standard to complement HTML
- Origins: Structured text SGML
- Motivation:
  - HTML describes presentation
  - XML describes content
  HTML4.0 ⊂ XML ⊂ SGML
XML Terminology

- **Tags**: book, title, author, ...
- **Elements**:
  - `<book>…</book>,<author>…</author>`
  - elements can be nested
  - empty element: `<red></red>` (Can be abbrv. `<red/>`)
- **XML document**: Has a single root element
- **Well-formed XML document**: Has matching tags

More XML: Attributes

```xml
<book price="55" currency="USD">
  <title> Foundations of Databases </title>
  <author> Abiteboul </author>
...
  <year> 1995 </year>
</book>
```

Attributes are alternative ways to represent data

More XML: Oids and References

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

oids and references in XML are just syntax
More XML: CDATA Section

- Syntax: `<![CDATA[ .....any text here...]]>`

- Example:

```xml
<example>
  <![CDATA[ some text here </notAtag> <>]]>
</example>
```

More XML: Entity References

- Syntax: `&entityname;`

- Example:

```
<element> this is less than &lt; </element>
```

- Some entities:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td><code>&lt;</code></td>
</tr>
<tr>
<td>&gt;</td>
<td><code>&gt;</code></td>
</tr>
<tr>
<td>&amp;</td>
<td><code>&amp;</code></td>
</tr>
<tr>
<td>'</td>
<td><code>'</code></td>
</tr>
<tr>
<td>&quot;</td>
<td><code>&quot;</code></td>
</tr>
<tr>
<td>&amp;</td>
<td><code>&amp;</code></td>
</tr>
<tr>
<td>'</td>
<td><code>'</code></td>
</tr>
<tr>
<td>&quot;</td>
<td><code>&quot;</code></td>
</tr>
<tr>
<td>&amp;</td>
<td><code>&amp;</code></td>
</tr>
</tbody>
</table>

Xml – Storage

- Storage is done just like an n-ary tree (DOM)
 Xml vs. Relational Model

Computer Table

<table>
<thead>
<tr>
<th>Id</th>
<th>Speed</th>
<th>RAM</th>
<th>HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>800Mhz</td>
<td>256MB</td>
<td>40GB</td>
</tr>
<tr>
<td>102</td>
<td>933Mhz</td>
<td>512MB</td>
<td>40GB</td>
</tr>
</tbody>
</table>

Overview

- From HTML to XML
- DTDs

Document Type Descriptors

- Sort of like a schema but not really.
  ```
  <!ELEMENT Book (title, author*) >
  <!ELEMENT title #PCDATA>
  <!ELEMENT author (name, address,age?)>
  <!ATTLIST Book id ID #REQUIRED>
  <!ATTLIST Book pub IDREF #IMPLIED>
  ```
- Inherited from SGML DTD standard
- BNF grammar establishing constraints on element structure and content
- Definitions of entities
**DTD - An Example**

```
<?xml version='1.0'?>
<!ELEMENT Basket (Cherry+, (Apple | Orange)*) >
<!ELEMENT Cherry EMPTY>
<!ATTLIST Cherry flavor CDATA #REQUIRED>
<!ELEMENT Apple EMPTY>
<!ATTLIST Apple color CDATA #REQUIRED>
<!ELEMENT Orange EMPTY>
<!ATTLIST Orange location 'Florida'>
```

```
<Basket>
  <Apple/>
  <Cherry flavor='good'/>
  <Orange/>
</Basket>
```

```
<Basket>
  <Cherry flavor='good'/>
  <Apple color='red'/>
  <Apple color='green'/>
</Basket>
```

**DTD - !ELEMENT**

```
<!ELEMENT Basket (Cherry+, (Apple | Orange)*) >
```

- !ELEMENT declares an element name, and what child elements it should have
- Wildcards:
  * * Zero or more
  + One or more

**DTD - !ATTLIST**

```
<!ATTLIST Cherry flavor CDATA #REQUIRED>
```

- !ATTLIST defines a list of attributes for an element
- Attributes can be of different types, can be required or not required, and they can have default values.
Attributes in DTDs

Types:
- CDATA = string
- ID = key
- IDREF = foreign key
- IDREFS = foreign keys separated by space
- (Monday | Wednesday | Friday) = enumeration
- NMTOKEN = must be a valid XML name
- NMTOKENS = multiple valid XML names
- ENTITY = you don’t want to know this

Attributes in DTDs

Kind:
- #REQUIRED = optional
- #IMPLIED = default value
- value = 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)
**DTD – Well-Formed and Valid**

```xml
<?xml version='1.0'?>
<!ELEMENT Basket (Cherry+)>
<!ELEMENT Cherry EMPTY>
<!ATTLIST Cherry flavor CDATA #REQUIRED>

Well-Formed and Valid
<Basket>
  <Cherry flavor='good'/>
</Basket>

Not Well-Formed
<basket>
  <Cherry flavor='good'>
  <Location>Home</Location>
</Basket>

Well-Formed but Invalid
<Job>
  <Location>Home</Location>
</Job>
```

**DTDs as Grammars**

```xml
<!DOCTYPE paper [
<!ELEMENT paper (section*)>
<!ELEMENT section ((title,section*) | text)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT text (#PCDATA)>
]

<paper> <section> <text> </text> </section> <section> <title> </title> <section> … </section> <section> … </section> </section> </paper>
```

**DTDs as Grammars**

- A DTD = a grammar
- A valid XML document = a parse tree for that grammar
**DTDs as Schemas**

Not so well suited:
- impose unwanted constraints on order
  
- references cannot be constrained

- can be too vague:
  
```xml
<!ELEMENT person ((name|phone|email)*)>
```

like an upper bound schema

**Shortcomings of DTDs**

Useful for documents, but not so good for data:
- No support for structural re-use
  - Object-oriented-like structures aren’t supported

- No support for data types
  - Can’t do data validation

- Can have a single key item (ID), but:
  - No support for multi-attribute keys
  - No support for foreign keys (references to other keys)
  - No constraints on IDREFs (reference only a Section)

**XML Schema**

- In XML format
- Includes primitive data types (integers, strings, dates, etc.)
- Supports value-based constraints (integers > 100)
- User-definable structured types
- Inheritance (extension or restriction)
- Foreign keys
- Element-type reference constraints
Sample XML Schema

```xml
<schema version="1.0" xmlns="http://www.w3.org/1999/XMLSchema">
  <element name="author" type="string"/>
  <element name="date" type="date"/>
  <element name="abstract">
    <type>
      ...
    </type>
  </element>
  <element name="paper">
    <type>
      <attribute name="keywords" type="string"/>
      <element ref="author" minOccurs="0" maxOccurs="*"/>
      <element ref="date"/>
      <element ref="abstract" minOccurs="0" maxOccurs="1"/>
      <element ref="body"/>
    </type>
  </element>
</schema>
```

Important XML Standards

- XSL/XSLT: presentation and transformation standards
- RDF: resource description framework (meta-info such as ratings, categorizations, etc.)
- XPath/XPointer/XLink: standard for linking to documents and elements within
- Namespaces: for resolving name clashes
- DOM: Document Object Model for manipulating XML documents
- SAX: Simple API for XML parsing

XML Data Model (Graph)

Think of the labels as names of binary relations.

Issues:
- Distinguish between attributes and sub-elements?
- Should we conserve order?
XML vs. Semistructured Data

- Both described best by a graph
- Both are schema-less, self-describing
- XML is ordered, ssd is not
- XML can mix text and elements:
  - `<talk>` Making Java easier to type and easier to type
  - `</speaker>` Phil Wadler
  - `</talk>`
- XML has lots of other stuff: entities, processing instructions, comments

What about XML queries?

- XPath
  - A single-document language for “path expressions”
  - Not unlike regular expressions on tags
  - E.g. `/Contract/*/UnitPrice, /Contract//UnitPrice, etc.
- XSLT
  - XPath plus a language for formatting output
- XQuery