XML

Syed Gillani
Laboratoire Hubert Curien St-Etienne, France

[Outline]

- XML (*ACK: Alon Halevy, University of Washington*)
  - ✓ Data Model
  - ✓ Examples
- XML Schema
  - ✓ DTD
  - ✓ XSD

[XML]

- ✓ eXtensible Markup Language
- ✓ XML 1.0 – a recommendation from W3C, 1998
- ✓ Roots: SGML (a very nasty language).
- ✓ After the roots: a format for sharing *data*

[XML]

- XML is just syntax for data
  - ✓Note: we have no syntax for relational data
  - ✓But XML is not relational: *semistructured*

Students Table

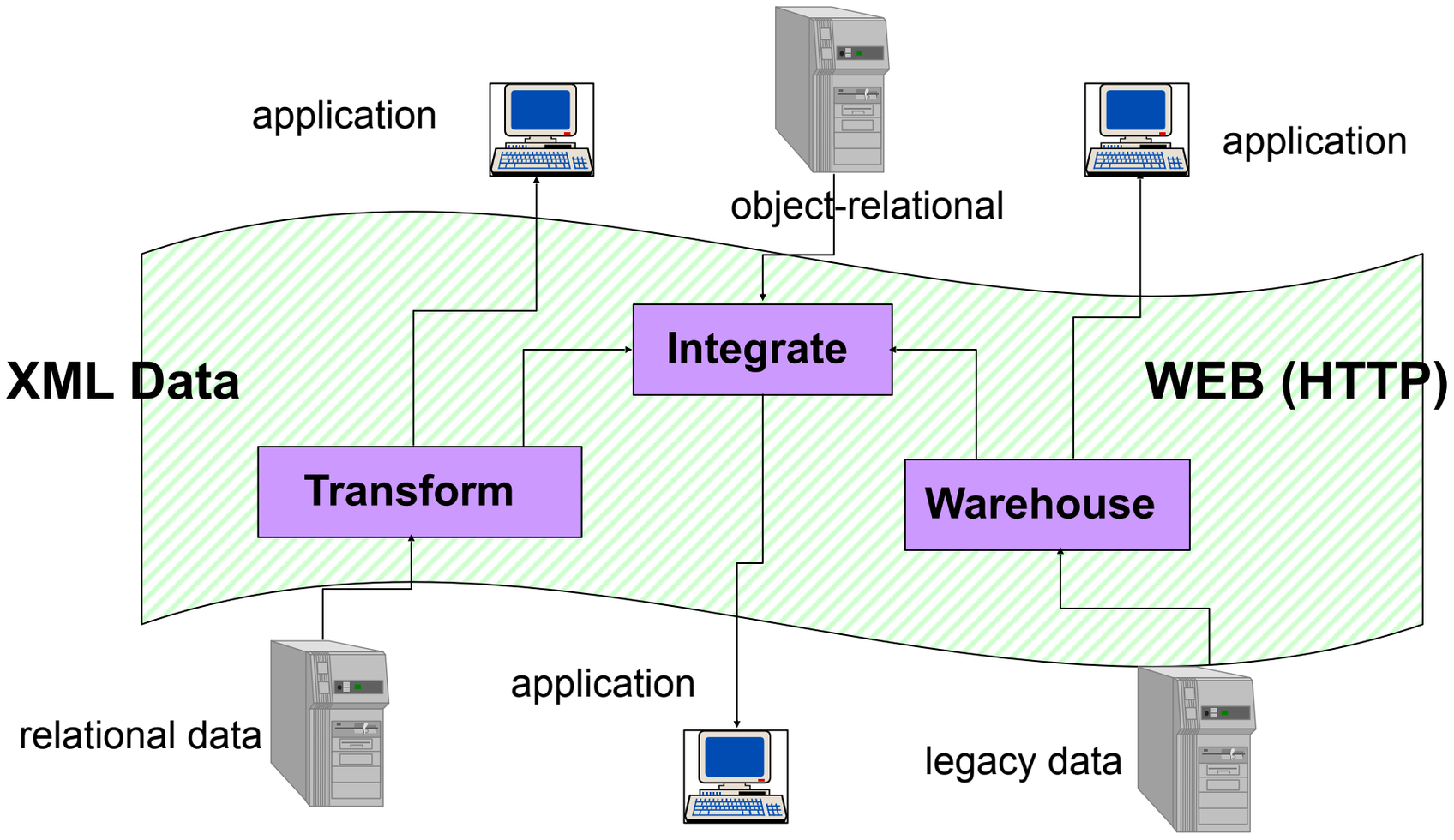
Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Activities Table

ID*	Activity1	Cost1	Activity2	Cost2
084	Tennis	$36	Swimming	$17
100	Squash	$40	Swimming	$17
182	Tennis	$36		
219	Swimming	$15	Golf	$47

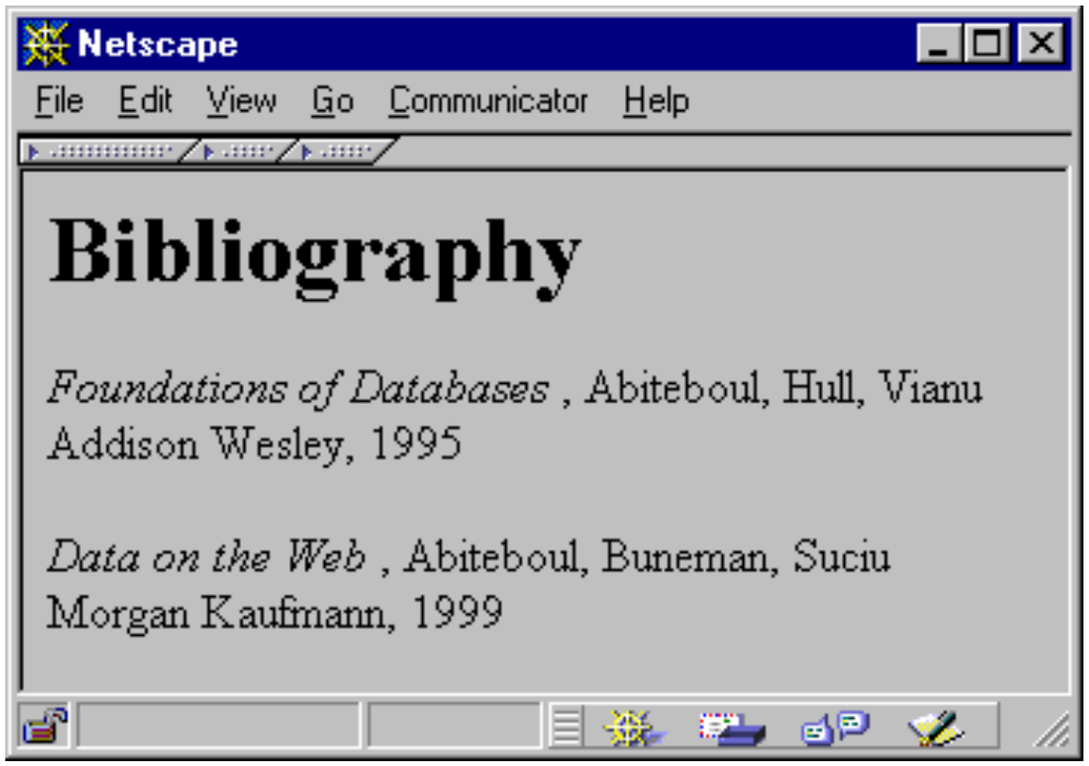
- This is exciting because:
  - ✓Can translate *any* data to XML
  - ✓Can ship XML over the Web (HTTP)
  - ✓Can input XML into any application
  - ✓Thus: data sharing and exchange on the Web

[XML Data Sharing and Exchange]



Specific data management tasks

[From HTML to XML]



HTML describes the presentation

[From HTML to XML]

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

[From HTML to XML]

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

XML describe the content…

[XML Terminology]

- ✓ tags: book, title, author, …
- ✓ start tag: <book>, end tag: </book>
- ✓ elements: <book>…<book>,<author>…</author>
- ✓ elements are nested
- ✓ empty element: <red></red> abbrv. <red/>
- ✓ an XML document: single *root element*

well-formed XML document: if it has matching tags

[More XML: Attributes]

Describe the property of each tag

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

Attributes are alternative ways to represent data

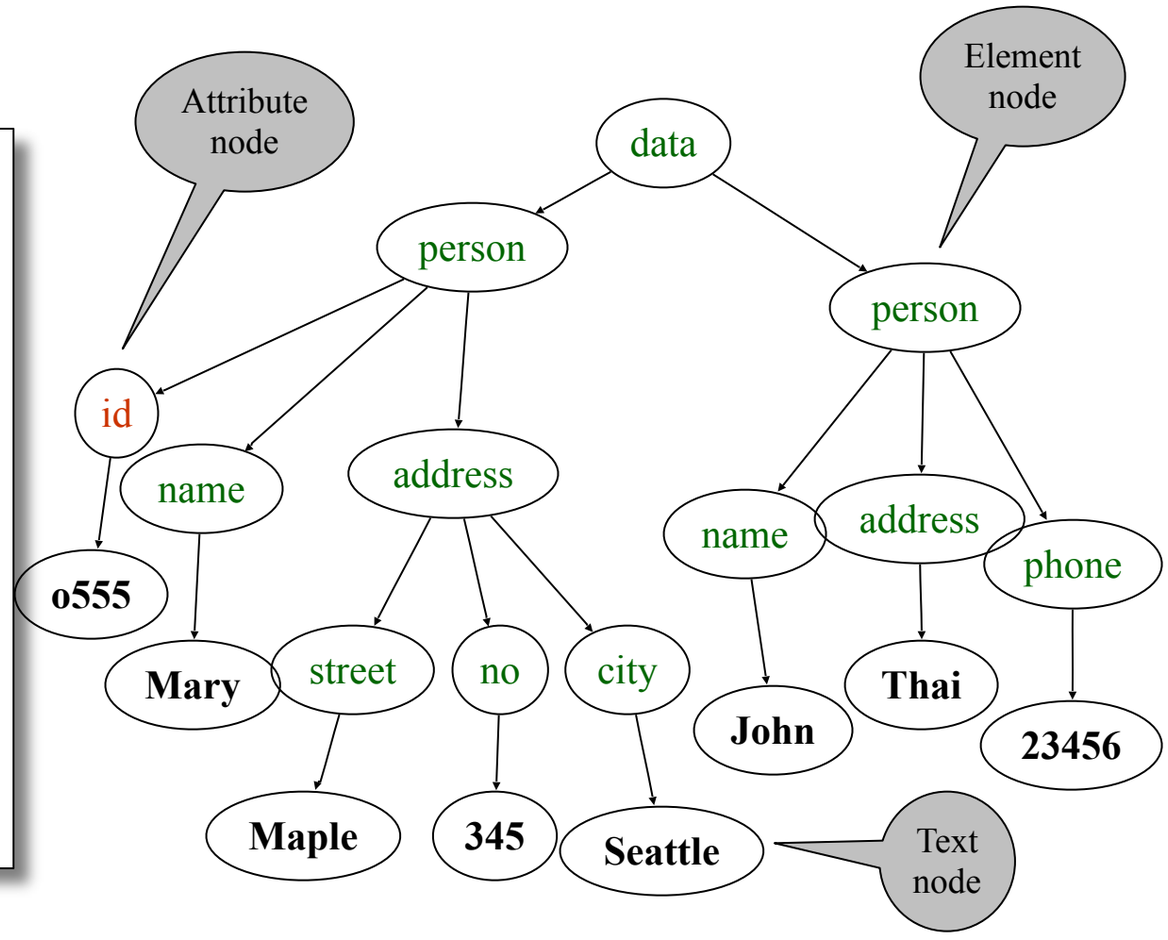
[More XML: Oids and References]

```
<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

[XML Semantics: a Tree !]

```
<data>
        <person id="o555" >
                <name> Mary </name>
                <address>
                        <street> Maple </street>
                        <no> 345 </no>
                        <city> Seattle </city>
                </address>
        </person>
        <person>
                <name> John </name>
                <address> Thailand  </address>
                <phone> 23456  </phone>
        </person>
</data>
```



Order matters !!!

[XML Data]

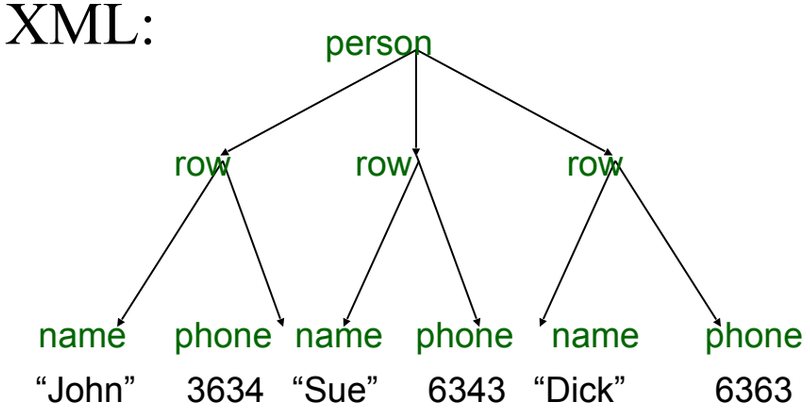
- ✓ XML is self-describing
- ✓ Schema elements become part of the data
  - ✓ Reational schema: persons(name,phone)
  - ✓ In XML <persons>, <name>, <phone> are part of the data, and are repeated many times
- ✓ Consequence: XML is much more flexible
- ✓ XML = semistructured data

[Relational Data as XML]

person

name	phone
John	3634
Sue	6343
Dick	6363

XML:




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

[Relational Data as XML]

Recall Attributes and Oids!!

```
<person id =“1”>
      <name>John</name>
      <phone> 3634</phone>

</person>

<person id =“2”>
       <name>Sue</name>
       <phone> 6343</phone>

</person>

<person id =“3”>
      <name>Dick</name>
       <phone> 6363</phone>

</person>
```

[XML is Semi-structured Data]

✓ Missing attributes:

```
<person>   <name> John</name>
           <phone>1234</phone>
 </person>

<person>  <name>Joe</name>
</person>
```

← no phone !

✓ Could represent in a table with nulls

name	phone
John	1234
Joe	-

[XML is Semi-structured Data]

✓ Attributes with different types in different objects

```
<person> <name>  <first> John </first>
                 <last> Smith </last>
         </name>
         <phone>1234</phone>
</person>
```

← structured name !

✓ Nested collections

✓ Heterogeneous collections:

– <db> contains both <book>s and <publisher>s

[XML is Semi-structured Data]

- How to Validate XML
  - ✓ How to force someone to follow a structure for XML
  - ✓ How to validate that an XML document is the one intended for
  - ✓ What is happening when an XML parser is confused
  - ✓ Well-formed = if tags are correctly closed
  - ✓ Validation is useful in data exchange

Use Document Type Definitions (DTD)

- An XML document may have a DTD
- **Valid** = if it has a DTD and conforms to it

[Document Type Definitions (DTD)]

OR

Kleene-* = zero or more

Optional = If exists take it

```
<!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)>
]>
```

Simple Data/no types

Check the DTD against XML documents to make sure it follows the syntax rules!

[Document Type Definitions (DTD)]

DTD

```
<!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 a valid XML document:

```
<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]

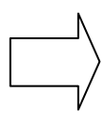
- Content model:
  - ✓ Complex = a regular expression over other elements
  - ✓ Text-only = #PCDATA
  - ✓ Empty = EMPTY
  - ✓ Any = ANY
  - ✓ Mixed content = (#PCDATA | A | B | C)*

[DTD: The Regular Expression]

sequence

DTD

```
<!ELEMENT name
                (firstName, lastName))
```



XML

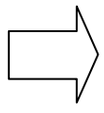
```
<name>
   <firstName>  . . . . . </firstName>
   <lastName>  . . . . . </lastName>
</name>
```

optional

```
<!ELEMENT name  (firstName?, lastName))
```

Kleene star

```
<!ELEMENT person   (name, phone*))
```



```
<person>
   <name>  . . . . . </name>
   <phone>  . . . . . </phone>
   <phone>  . . . . . </phone>
   <phone>  . . . . . </phone>
   . . . . . .
</person>
```

alternation

```
<!ELEMENT person   (name, (phone|email)))
```

[XML Schema: XSD]

- It Provides:
  - ✓ XML-based alternative to DTD
  - ✓ Describe the structure of an XML document
  - ✓ A W3C recommendation

- Why move to XSD from DTD?
  - ✓ Support data types (String, integer, float, date time, etc.)
  - ✓ Support namespaces
  - ✓ Written in XML
  - ✓ XSD is richer and more powerful than DTD

[XML Schema: XSD]

DTD: <!ELEMENT paper (title,author*,year, (journal|conference))>

```
<xsd:element name="paper" type="papertype"/>
<xsd:complexType name="papertype">
    <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="author" minOccurs="0"/>
        <xsd:element name="year" type="xsd:date"/>
        <xsd: choice>
< xsd:element name="journal" type="xsd:string" />
 <xsd:element name="conference" type="xsd:string"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:element>
```

[XML Schema: XSD]

- A large number of built-in data types (some examples):
  - ✓ xs:string
  - ✓ xs:decimal
  - ✓ xs:integer
  - ✓ xs:boolean
  - ✓ xs:date
  - ✓ xs:time

XML

```
<lastname>Obama</lastname>
<age>52</age>
<dateborn>1958-03-27</dateborn>
```

XSD

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

[Moving Forward]

- XML is nice, but…
  - ✓Sometimes too complicated for non-experts,… and even for expert
  - ✓Security
  - ✓Expensive to parse, and sometimes too verbose
  - ✓A semi-structure data model… Semantics?
  - ✓Usually contains cycles for a tree-structure data (querying?)

[La Fin]

★Acknowledgments: *Alon Halevy, University of Washington*