What Is XML?

- eXtensible Markup Language for data
  - Standard for publishing and interchange
  - "Cleaner" SGML for the Internet
- Applications:
  - Data exchange over intranets, between companies
  - E-business
  - Native file formats (Word, SVG)
  - Publishing of data
  - Storage format for irregular data
  - …

How Does it Look?

– *Emerging format for data exchange on the web and between applications.*

```
<db>
 <book>
    <title>Complete Guide to DB2</title>
    <author>Chamberlin</author>
 </book>
 <book>
    <title>Transaction Processing</title>
    <author>Bernstein</author>
    <author>Newcomer</author>
 </book>
 <publisher>
   <name>Morgan Kaufman</name>
   <state>CA</state>
 </publisher>
</db>
```


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

Attributes and References

- XML distinguishes attributes from sub-elements.
- ID's and IDREFs are used to reference objects.

```
<db>
 <book ID="b1" pub="mkp" year=1992>
    <title>Complete Guide to DB2</title>
    <author>Chamberlin</author>
 </book>
 <book ID="b2" pub="mkp" year=1997>
    <title>Transaction Processing</title>
    <author>Bernstein</author>
    <author>Newcomer</author>
 </book>
 <publisher ID="mkp">
   <name>Morgan Kaufman</name>
   <state>CA</state>
 </publisher>
</db>
```

oids and references in XML are just syntax

What's Special about XML?

- Supported by almost everyone
- Easy to parse (even with no info about the doc)
- Can encode data with little or much structure
- Supports data references inside & outside document
- Presentation layer for publishing (XSL)
- Human readable. No need for proprietary formats anymore.
- Many, many tools

Origin of XML

- Comes from SGML (very nasty language).
- Principle: separate the data from the graphical presentation.

```
<UL>
 <li> <b> Complete Guide to DB2 </b>
    By <i> Chamberlin </i>.

 <li> <b> Transaction Processing </b> By
      <i> Bernstein and Newcomer </i>

 <li> <b> The guide to the good life
through database research. </b>
      By <i> Alon Levy </i>

<UL>
```

XML, After the roots

- A format for sharing *data*.
- Applications:
  - EDI: electronic data exchange:
    - Transactions between banks
    - Producers and suppliers sharing product data (auctions)
    - Extranets: building relationships between companies
    - Scientists sharing data about experiments.
  - Sharing data between different components of an application.
  - Format for storing all data in Office 2000.
- Basis for data sharing and integration.

7

Why are we DB'ers interested?

- It's data, stupid. That's us.
- Proof by Altavista:
  - database+XML -- 40,000 pages.
- Database issues:
  - How are we going to model XML? (graphs).
  - How are we going to query XML? (XML-QL)
  - How are we going to store XML (in a relational database? object-oriented?)
  - How are we going to process XML efficiently? (uh… well..., um..., ah..., get some good grad students!)

8

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

9

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)

10

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

11

Sample XML Schema

```
<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>
```

12

Subtyping in XML Schema

```
<schema version="1.0"
   xmlns="http://www.w3.org/1999/XMLSchema">
<type name="person">
  <attribute name="ssn">
  <element name="title" minOccurs="0" maxOccurs="1" />
  <element name="surname" />
  <element name="forename" minOccurs="0" maxOccurs="*" />
</type>
<type name="extended" source="person"
   derivedBy="extension">
  <element name="generation" minOccurs="0" />
</type>
<type name="notitle" source="person"
   derivedBy="restriction">
  <element name="title" maxOccurs="0" />
</type>
<key name="personKey">
  <selector>.//person[@ssn]</selector>
  <field>@ssn</field>
</key>
</schema>
```

Important XML Standards

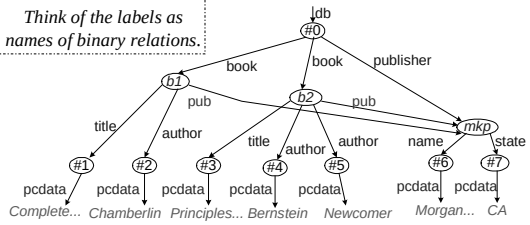
- **XSL/XSLT***: presentation and transformation standards
- **RDF**: resource description framework (meta-info such as ratings, categorizations, etc.)
- **Xpath/Xpointer/Xlink***: standard for linking to documents and elements within
- **Namespaces**: for resolving name clashes
- **DOM**: Document Object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

•***This weekend***, *somewhere in Germany*, a W3C committee is meeting to discuss standard query language.

14

XML Data Model (Graph)

Think of the labels as names of binary relations.



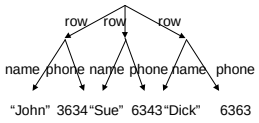
Issues:

- distinguish between attributes and sub-elements?
- Should we conserve order?

15

Comparison with Relational Data

name	phone
John	3634
Sue	6343
Dick	6363



- No strict typing
- Arbitrary nesting
- Data can be irregular
- Schema is part of the data

16

Querying XML

- Requirements:
  - Query a graph, not a relation.
  - The result should be a graph (representing an XML document), not a relation.
  - No schema.
  - We may not know much about the data, so we need to navigate the XML.

17

Query Languages

- First, there was XQL (from Microsoft).
- Very quickly realized that it was very limited.
- Then, a bunch of database researchers looked at XML and invented XML-QL.
  - XML-QL comes from the nicer StruQL language.
  - Many people got excited. Formed a committee.
- Last week: Quilt, a new language combining the best of XML-QL and XQL. Stay tuned.

18

Extracting Data by Query

- **Matching data using elements patterns.**

```
WHERE <book>
   <publisher><name>Addison-Wesley</></>
   <title> $t </>
   <author> $a </>
   </book> IN "www.a.b.c/bib.xml"
CONSTRUCT $a
```

19

Constructing XML Data

```
WHERE <book>
   <publisher><name>Addison-Wesley</></>
   <title> $t </>
   <author> $a </>
   </> IN "www.a.b.c/bib.xml
CONSTRUCT <result>
   <author> $a </>
   <title> $t</>
   </>
```

20

Grouping with Nested Queries

```
WHERE <book>
   <title> $t </>,
   <publisher><name>Addison-Wesley</></>
</> CONTENT_AS $p IN "www.a.b.c/bib.xml"
CONSTRUCT <result>
   <titre> $t </>
   WHERE <author> $a </> IN $p
   CONSTRUCT <auteur> $a</>
</>
```

21

Joining Elements by Value

```
WHERE
<article> <author> <firstname> $f </> <lastname> $l </>
          </> </> ELEMENT_AS $e IN "www.a.b.c/bib.xml"
<book year=$y> <author>
        <firstname> $f </> <lastname> $l </>
   </> </> IN "www.a.b.c/bib.xml" , y > 1995
CONSTRUCT $e
```

Find all articles whose writers also published a book after 1995.

22

Tag Variables

```
WHERE <article> <author>
                      <firstname> $f </> <lastname> $l </>
   </> </> ELEMENT_AS $e IN "www.a.b.c/bib.xml"
   <$t year=$y> <author>
                <firstname> $f </> <lastname> $l </>
   </> </> IN "www.a.b.c/bib.xml" , y > 1995
CONSTRUCT $e
```

Find all articles whose writers have done something after 1995.

23

Regular Path Expressions

```
WHERE
   <part*>
      <name>$r</>
      <brand>Ford</> </>
               IN  "www.a.b.c/bib.xml"
 CONSTRUCT
      <result>$r</>
```

Find all parts whose brand is Ford, no matter what level they are in the hierarchy.

24

Regular Path Expressions

```
WHERE
 <part+.(subpart|component.piece)>$r</>
        IN "www.a.b.c/parts.xml"
 CONSTRUCT
     <result> $r </>
```

25

XML Data Integration

Query can access more than one XML document.

```
WHERE <person>
          <name></> ELEMENT_AS $n
          <ssn> $ssn </>
      </> IN "www.a.b.c/data.xml"
      <taxpayer>
          <ssn> $ssn </>
          <income></> ELEMENT_AS $I
      </> IN "www.irs.gov/taxpayers.xml"
CONSTRUCT <result> $n $I </>
```

26

Skolem Functions in XML-QL

```
where <book language = $l>
          <author> $a </>
      </> in "www.a.b.c/bib.xml"
construct <result> <author id=F($a)> $a</>
                   <lang> $l </>
          </>
```

```
<result> <author>Smith</author>
         <lang>English</lang> <lang>Mandarin</lang>
</result>
<result> <author>Doe</author> <lang>English</lang> </result>
```

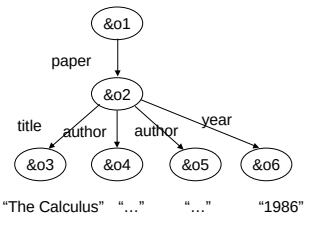
27

Query Processing For XML

- Approach 1: store XML in a relational database. Translate an XML-QL query into a set of SQL queries.
  - Leverage 20 years of research & development.
- Approach 2: store XML in an object-oriented database system.
  - OO model is closest to XML, but systems do not perform well and are not well accepted.
- Approach 3: build an entire DBMS tailored to XML.
  - Still in the research phase.

28

Store XML in Ternary Relation



Ref

Source	Label	Dest
&o1	paper	&o2
&o2	title	&o3
&o2	author	&o4
&o2	author	&o5
&o2	year	&o6

Val

Node	Value
&o3	The Calculus
&o4	…
&o5	…
&o6	1986

[Florescu, Kossman 1999]

29

Use DTD to derive Schema

- DTD:

```
<!ELEMENT employee (name, address, project*)>
<!ELEMENT address   (street, city, state, zip)>
```

- ODMG classes:

```
class Employee public type tuple
   (name:string, address:Address, project:List(Project))
class Address public type tuple (street:string, …)
```

- [Christophides et al. 1994 , Shanmugasundaram et al. 1999]

30

The Future

- Many research problems remain:
  - Efficient storage of XML
  - How to leverage relational DBMS
  - Update formalisms
  - Processing streaming data
  - Transactions
  - Everything else we think about in databases.

31