

COMP67321

A Very Brief Overview of XML Databases

Goran Nenadic

School of Computer Science
University of Manchester

1

Aims

- Understand the need for managing semi-structured data
- Review the basic principles of design, storage, querying and retrieval in XML DBs
- Understand problems and challenges of XML databases
- Understand how XML can be used for data integration and sharing

2

Introduction – data integration

- Relational data does not have a “syntax”
 - I can’t “give” you my relational database (if we don’t use the same DBMS and share the schema)
 - need to import it from other syntax, like CSV (comma-separated-values; or SQL commands)
- We need a **flexible model** that supports rich syntax for complex data, and
 - can map any (existing) data into it
 - can store data in files, so exchange on the Web, etc.
 - query it directly

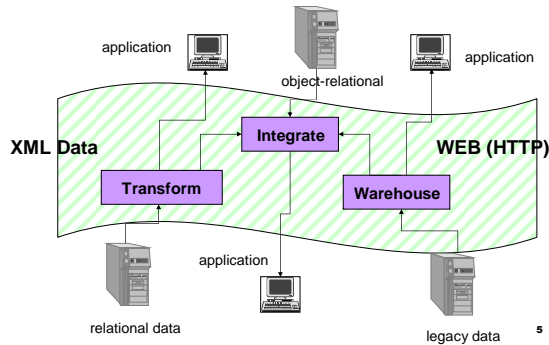
3

Solution: XML

- XML = eXtensible Mark-up Language
- Entity = Data + Mark-up
 - data entities can be strings, images, video, pubs, trains, visits, etc.
 - mark-up = assign annotation to data
 - explicit “structure”, “semantics”, “meaning”
- XML 1.0
 - recommendation from W3C, 1998

4

XML data sharing and exchange



5

HTML vs. XML

- HTML uses tags for formatting (e.g., “italic”)
 - describes the layout
- XML uses tags for structure and semantics (e.g., “this is an address, and it contains house number, street, and postcode”)

6

XML example (1)

```
<addrBook>
  <person NI="111-22-3333">
    <name> Caesar </name>
    <greet> Caesar Imperator</greet>
    <addr> The Capitol </addr>
    <addr> Rome, OH 98765 </addr>
    <tel> (321) 786 2543 </tel>
    <fax> (321) 786 2543 </fax>
    <email> jc@forum.rome.org </email>
  </person>
</addrBook>
```

7

XML: main concepts

- XML document
- Schema descriptions
 - defined by
 - DTD = Document Type Definition
 - XSD = XML Schema Definition
- Elements, including their
 - structure, and
 - attributes (describe elements)

8

XML terminology

- **tags**: book, title, author, ...
- **start tag**: <book>, **end tag**: </book>
- **elements**: <book>...</book>, <author>...</author>
- elements can be **nested**
- **empty element**: <red></red> or <red/>
- **attributes**: add additional features to data
 <book year="1992">...
 alternative: <book><year>1992</year>...

9

XML example (2)

```
<book year="1992">
  <title>Advanced Programming in the Unix environment</title>
  <author><last>Stevens</last><first>W.</first></author>
  <publisher>Addison-Wesley</publisher>
  <price>65.95</price>
</book>

<book year="2000">
  <title>Data on the Web</title>
  <author><last>Abiteboul</last><first>Serge</first></author>
  <author><last>Buneman</last><first>Peter</first></author>
  <author><last>Suciu</last><first>Dan</first></author>
  <publisher>Morgan Kaufmann Publishers</publisher>
  <price>39.95</price>
</book>
```

10

Document type definition (DTD)

- Essentially, a context-free grammar for describing XML tags and their nesting
- Each domain of interest (e.g., electronic components, bars-beers-drinkers) creates a DTD that describes all the documents (data) this group will share
- an XML document **may** have a DTD
 - not obligatory
 - but, validation is useful in data exchange

11

example

A simple DTD

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

12

XML schema definition (XSD)

- DTDs define structure, but do not provide type constraints
- XSD = XML schema definition
- XML schema is a W3C standard for data modelling using XML
- XSD can specify
 - which elements/attributes are mandatory/optional
 - element/attribute **types**
 - **cardinalities**
 - **relative** ordering

13

XSD example

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Employee"
    minOccurs="0"
    maxOccurs="unbounded">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="NI" type="xsd:string"/>
        <xsd:element name="Name" type="xsd:string"/>
        <xsd:element name="DateOfBirth" type="xsd:date"/>
        <xsd:element name="EmployeeType" type="xsd:string"/>
        <xsd:element name="Salary" type="xsd:long"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

cardinality

data type

14

XSD: summary

- XSD provides a wide range of modelling facilities for defining XML documents
- Can be used to define both **structure** and **types**
- However, intended mainly for machine processing
 - easy to parse/validate
 - human readability not considered!
 - typically much longer than related DTDs

15

Data modelling with XML

16

Semi-structured data

- Semi-structured data
 - no strict format, but can have some structure with optional elements and attributes
 - some attributes may be missing, some repeated, and new ones can be added later: change unpredictably
 - therefore, need explicit information about elements
 - ad-hoc, diverse data sources
 - examples
 - HTML document with titles (e.g. <H1>) and paragraphs (>p>), but no information about author(s), date, etc.
 - multimedia objects

17

example

Semi-structured data

- Missing attributes


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

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

no phone !
- Could be represented in a relational table with nulls (-)

name	phone
John	1234
Joe	-

18

example

Semi-structured data

- Repeated attributes


```
<person> <name> Mary</name>
  <phone>2345</phone>
  <phone>3456</phone>
</person>
```

two phones !
- Difficult in a table
(but what about two tables?)

name	phone		
Mary	2345	3456	???

19

example

Semi-structured data

- Relational schema and data are separated
 - relational schema: persons(name,phone)
- Typically, semi-structured data is **self-describing**
 - mixing data and meta-data (schema)
- Schema elements become part of the data
 - in XML, <person>, <name>, <phone> are part of the data, and are repeated for each entity (as required)
- Semi-structured data can be modeled by trees and/or (directed) graphs

20

Example: tree/graph model

21

Data modeling with XML

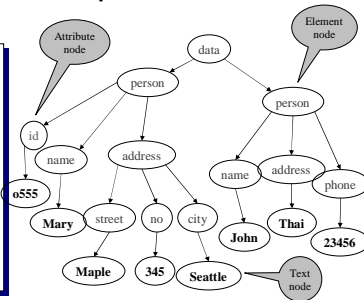
- XML is self-describing
 - consequence: XML is much more flexible
- Well-Formed XML with nested tags is exactly the same idea as trees of semi-structured data
- XML also enables non-tree structures, as does the semi-structured data model

XML is a natural choice to model semi-structured data

22

XML data: tree representation

```
<data>
  <person id="0555" >
    <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>
```



23

example

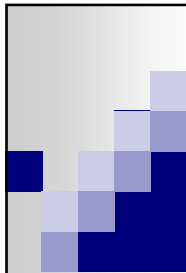
From relational to XML data

employees

name	phone
John	3634
Sue	6343
Dick	6363

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

24



Storing and querying XML data

25

Storing XML data

- XML-data can be stored in the following 3 types of DBMS
 - **traditional relational** database
 - DBMS basically does not 'deal' with XML
 - storage
 - without mapping (XML documents as CLOBs)
 - map an XML schema to relational schema
 - **XML-enabled** database
 - **XML-native** database

26

XML-enabled databases

- Use object or object-relational model to map XML-schema
 - model XML-elements as classes
 - e.g. element *Person* is modelled as a class *Person*
- Also, new data types provided to support XML
 - **SQL:2003** (SQL/XML) introduced XML extensions including a new native data type (*XML*) and a set of operators
 - DBMS now understands how to deal with XML
 - XMLCLOB, XMLFile (DB2); XMLTYPE (Oracle 9i)

27

example

XML-enabled databases

```
CREATE TABLE mail_user (
  user_name VARCHAR2(20),
  mailbox SYS.XMLTYPE
);
```

```
<mailbox>
  <inbox>
    <email id=1><subject text="Urgent info"/><from .../><attach .../> </email>
    <email id=2><subject text="Meeting"/><from .../><attach .../> </email>
    ....
  </inbox>
  <outbox>
    <email id=7><subject text="RE:Urgent"/><from .../><attach .../> </email>
    ....
  </outbox>
</mailbox>
```

28

example

XML-enabled databases

Querying using SQL-like expressions

return names of users that have any email in their inbox that contains string "Manchester" in its subject

```
SELECT   user_name
FROM     mail_user
WHERE    mailbox.extract('/mailbox/inbox/email/subject/text()').getStringVal()
LIKE    '%Manchester%'
```

↑

Elements
(full path)

↑

Attribute

29

XML-native databases

- Designed to store XML documents
 - do not rely on a relational or object model
 - basic unit of logical design is an XML-document
 - easy/natural storage and access to data
- Support for XML query languages
 - XPath = simple path through the tree
 - XQuery = the SQL of XML
- Example XML-native DBMS
 - Tamino, X-Hive (commercial)
 - dbXML, eXist (open-source)

30

XML query languages: XPath and XQuery

31

file: bib.xml

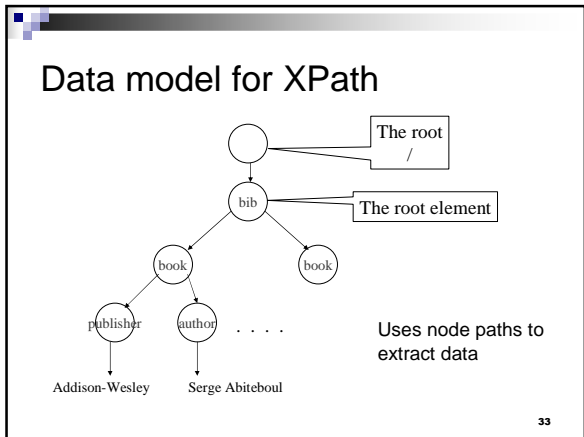
Sample data for queries

```

<bib>
  <book> <publisher> Addison-Wesley </publisher>
    <author> Serge Abiteboul </author>
    <author> <first-name> Rick </first-name>
      <last-name> Hull </last-name>
    </author>
    <author> Victor Vianu </author>
    <title> Foundations of Databases </title>
    <year> 1995 </year>
  </book>

  <book price="55">
    <publisher> Freeman </publisher>
    <author> Jeffrey D. Ullman </author>
    <title> Principles of Database and Knowledge Base Systems
    </title>
    <year> 1998 </year>
  </book>
</bib>

```



XPath: simple expressions

1. retrieve *titles* of all books from file "bib.xml"

```
doc("bib.xml")/bib/book/title
```

result

```
<title>Foundations of Databases</title>  
<title>Principles of Database and Knowledge Base Systems</title>
```

2.

```
/bib/book/year
```

 result

```
<year> 1995 </year>  
<year> 1998 </year>
```

34

examples

XPath: predicates

- Specify restrictions/conditions/constraints

1. retrieve all books published before 2000

```
doc("bib.xml")/bib/book[year<2000]
```

2. retrieve *titles* of all books published before 2000

```
doc("bib.xml")/bib/book[year<2000]/title
```

35

Examples

bib	matches a bib element
@price	matches a price attribute
/	matches the root element
/bib	matches a bib element under root
bib/paper	matches a paper in bib
bib/book/@price	matches price attribute in book, in bib

Also,
paper|book matches a paper or a book
* matches any element

36

XQuery: motivation

- XPath's expressivity insufficient
 - no join queries (as in SQL)
 - no quantifiers (as in SQL)
 - no aggregation and functions
 - no changes to the XML structure possible
- XQuery uses XPath to express more complex queries

37

XQuery: basic structure

```
FOR ...  
LET...  
WHERE...  
ORDER BY...  
RETURN...
```

38

XQuery: example

```
FOR $x IN document("bib.xml")/bib/book  
WHERE $x/year/text() > 1995  
RETURN $x/title
```

Retrieve all book titles published after 1995

39

XQuery: example

```
FOR $x IN document("bib.xml")/bib/book[year/text()>1995]/title
RETURN $x
```

Retrieve all book titles published after 1995

40

XQuery: joins and nesting

For each author of a book by *Addison-Wesley*,
list all books they published:

```
FOR $b IN document("bib.xml")/bib,
  $a IN $b/book[publisher/text()='Addison-Wesley']/author
RETURN { $a,
        FOR $t IN $b/book[author/text()=$a/text()]title
        RETURN $t
      }
```

In the RETURN clause, comma concatenates XML fragments

41

XQuery: aggregates

count = a function that counts
avg = computes the average
sum = computes the sum
distinct-values = eliminates duplicates

Find all books with more than 3 authors

```
FOR $x IN document("bib.xml")/bib/book
WHERE count($x/author)>3
RETURN $x
```

```
FOR $x IN document("bib.xml")/bib/book[count(author)>3]
RETURN $x
```

42

Summary - use XML for

- Representation, storage and content-based retrieval of semi-structured data
 - e.g. storing and querying multimedia, text, Semantic Web
- Data exchange
 - transform relational/object models into XML models
 - however, unlike relational/object databases, an XML database need not have a schema!
 - an XML schema may not be very prescriptive in terms of what can or cannot be stored

43

Summary – XML DB

- XML hosted in
 - relational DBs
 - XML-enabled DBs
 - XML-native DBs
- XML query languages
 - XPath and XQuery

44

Reading for this lecture

- Main text: Chapter 26 in [Elmasri & Navathe]
- See also, Chapter 30 in [Connolly & Begg] (30.1, 30.2, 30.4, 30.5, 30.6)
- See many on-line materials
 - e.g. www.w3.org
 - <http://www.alpha-works.ibm.com/xml/newto>
 - <http://www.oracle.com/technology/tech/xml/xmldb/index.html>

45
