Semistructured data and XML

CS 445

Fall 2008

Some slide content courtesy of Ramakrishnan & Gehrke, Dan Suciu, Zack Ives.

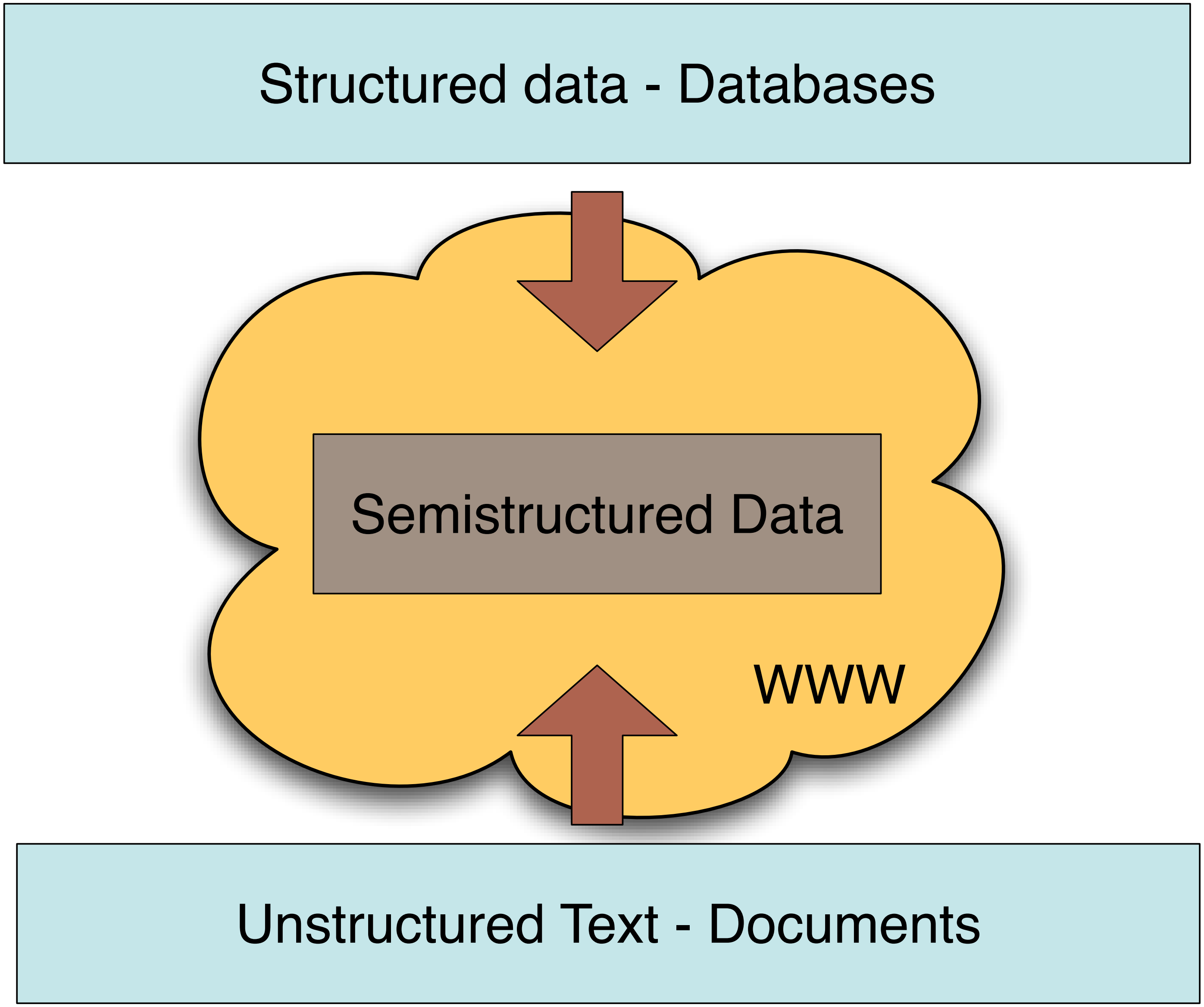
Today’s lecture

- Semistructured data
  - History and motivation
- Querying XML data: XPath
- XML: syntax and typing
- Querying XML data: XQuery

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
  - authoring is constrained: schema-first
  - changes to structure not easy
  - querying constrained: must know schema
  - data exchange hard: integration of diff schema

Some reasons why more data is not in databases



Structured data - Databases

Semistructured Data

WWW

Unstructured Text - Documents

XML data

```
<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>
```

Need for loose structure

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

XML is the preeminent format for semi-structured data

XML is the confluence of many factors:

- The Web needed a more declarative format for data
- Documents needed a mechanism for extended tags
- Database people needed a more flexible interchange format
- It's parsable even if we don't know what it means!

Original expectation:

- The whole web would go to XML instead of HTML

Today's reality:

- Not so… But XML is used all over "under the covers"

Why DB People Like XML

Can get data from all sorts of sources

- Allows us to touch data we don’t own!
- This was actually a huge change in the DB community

Blends schema and data into one format

- Unlike relational model, where we need schema first
- … But too little schema can be a drawback, too!

XML: Syntax

XML Syntax

- 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*

An XML document is **well formed** if it has matching tags

XML Syntax

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

attributes are alternative ways to represent data

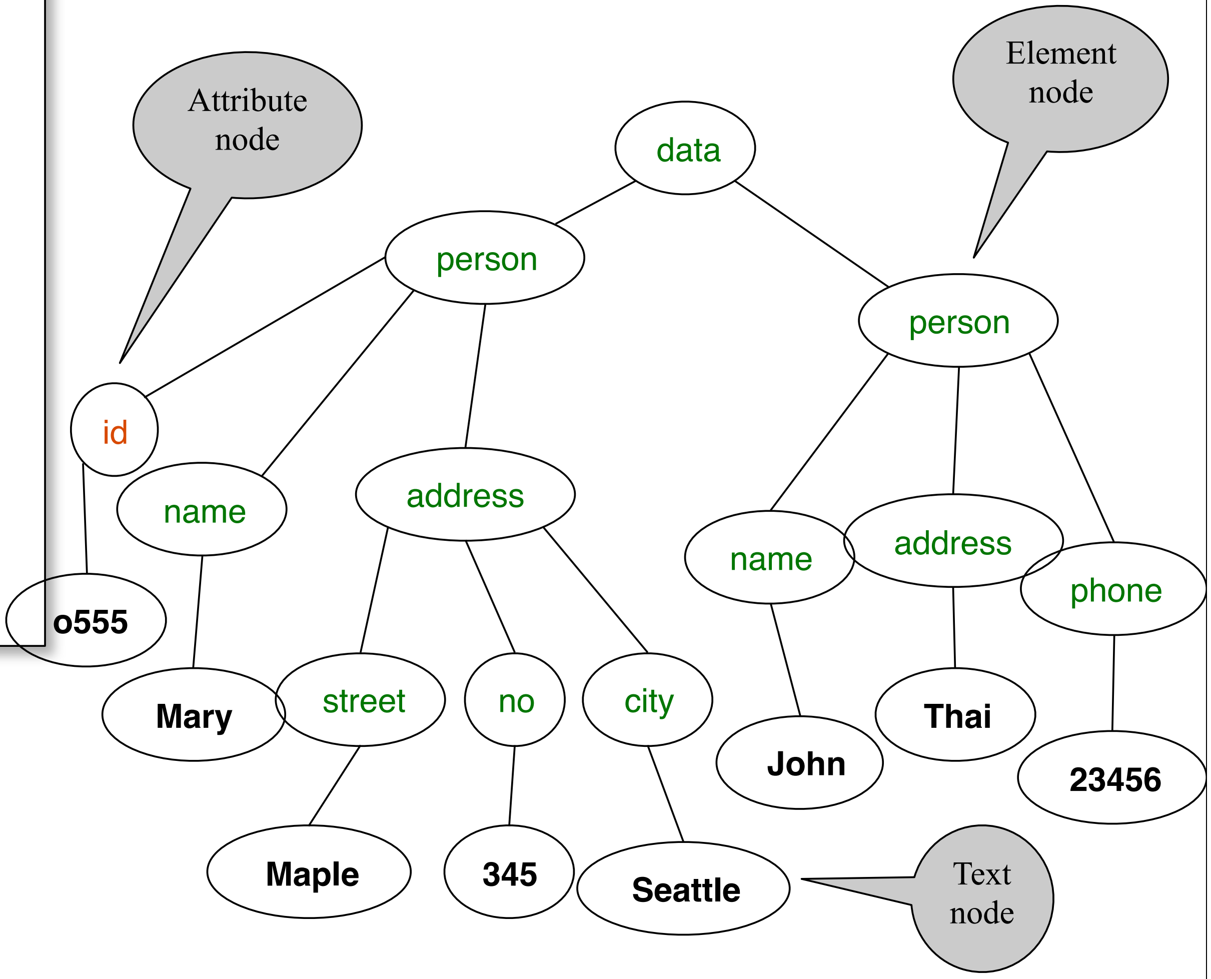
XML Syntax

```
<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
  - 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:

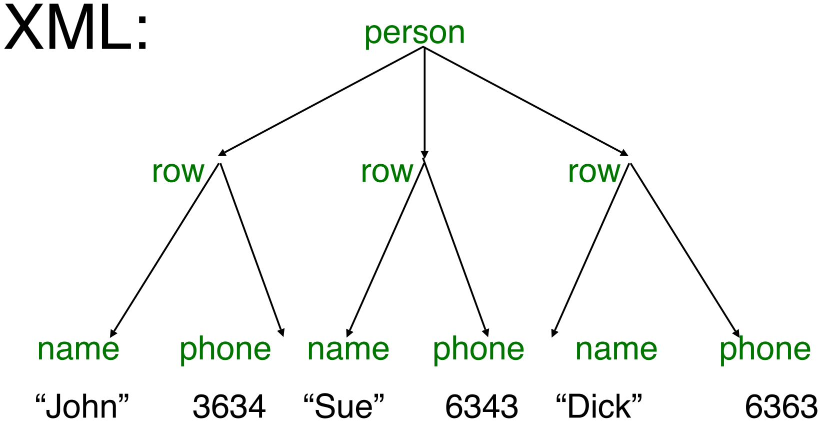
http://www.cs.washington.edu/research/xmldatasets/

Relational Data as XML

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>
```

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

- Repeated attributes

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

← two phones !

- Impossible in tables:

name	phone	
Mary	2345	3456

??
?

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 (non 1NF)
- Heterogeneous collections:
  - <db> contains both <book>s and <publisher>s

Querying XML Data

- Querying XML has two components
  - Selecting data
    - pattern matching on structural & path properties
    - typical selection conditions
  - Construct output, or transform data
    - construct new elements
    - restructure
    - order

Querying XML Data

- XPath = simple navigation through the tree
- XQuery = the SQL of XML
  - next time
- XSLT = recursive traversal
  - will not discuss in class

Querying XML

How do you query a directed graph? a tree?

The standard approach used by many XML, semistructured-data, and object query languages:

- Define some sort of a template describing traversals from the root of the directed graph
- In XML, the basis of this template is called an XPath

XPath is widely used

- XML Schema uses simple XPaths in defining keys and uniqueness constraints
- XQuery
- XSLT
- XLink and XPointer, hyperlinks for XML

XPaths

In its simplest form, an XPath is like a path in a file system:

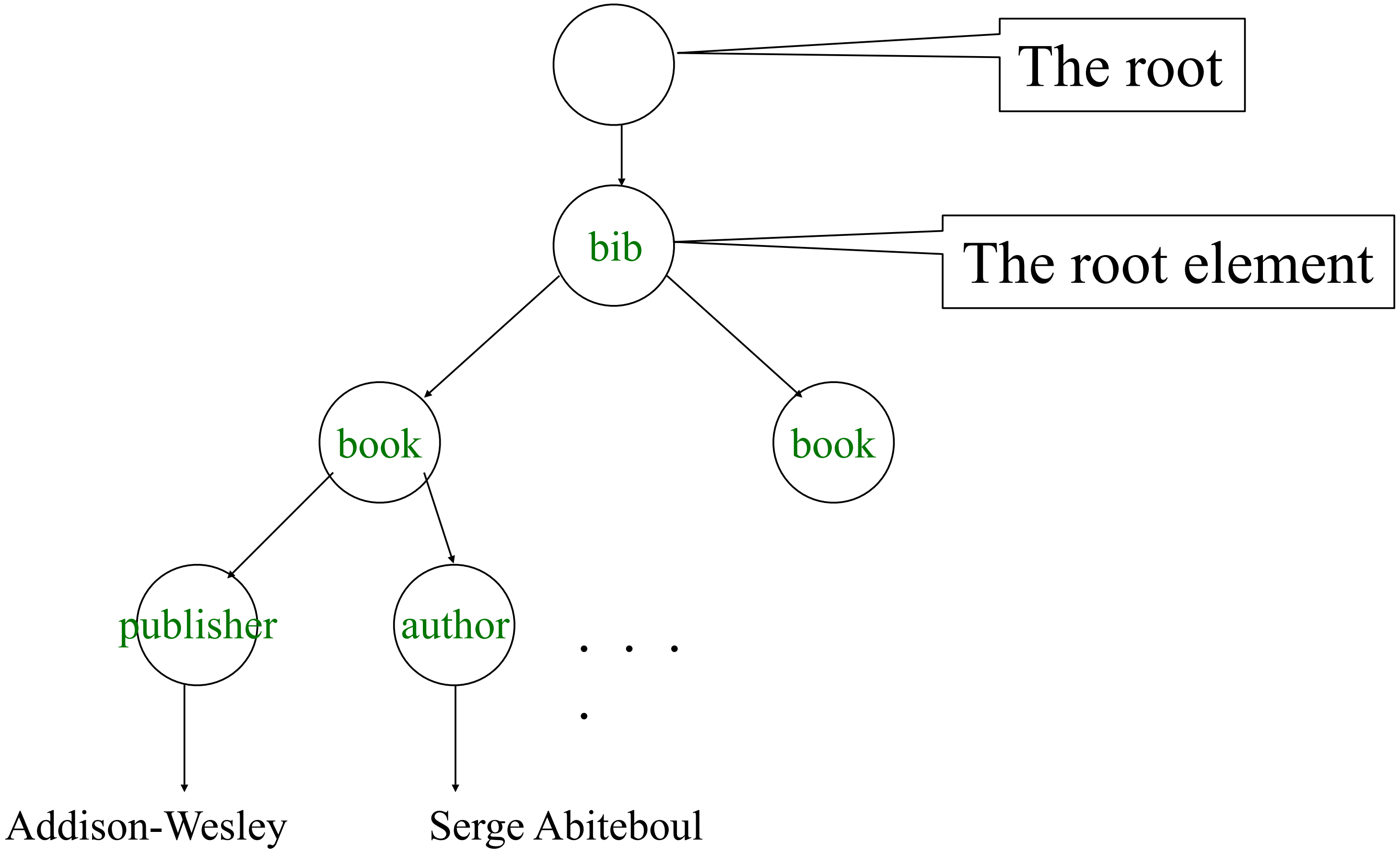
/mypath/subpath/*/morepath

- The XPath returns a **node set** representing the XML nodes (and their subtrees) at the end of the path
- XPaths can have node tests at the end, returning only particular node types, e.g., text(), element(), attribute()
- XPath is fundamentally an ordered language: it can query in order-aware fashion, and it returns nodes in order

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>
```

Data Model for XPath



The root

The root element

XPath

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

```
/bib/paper/year
```

```
//author
```

```
/bib//first-name
```

```
//author/*
```

```
/bib/book/@price
```

```
/bib/book/author[firstname]
```

```
/bib/book/author[firstname][address[.//zip][city]]/lastname
```