

# XML & XPath

DSCI 551

Wensheng Wu

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
~
"core-site.xml" 24 lines, 889 characters

```

# Agenda

- XML:
  - What is it and why do we care?
  - Data model (ordered tree)
  - Query language: XPath

# XML

- eXtensible Markup Language
- XML 1.0 – a recommendation from W3C, 2008
- Root: SGML (standard generalized markup language)
- After the root: a format for sharing *data*
- *Ajax (x – XML)*
- *jquery (\$.ajax(..., format= 'XML'/'JSON'))*

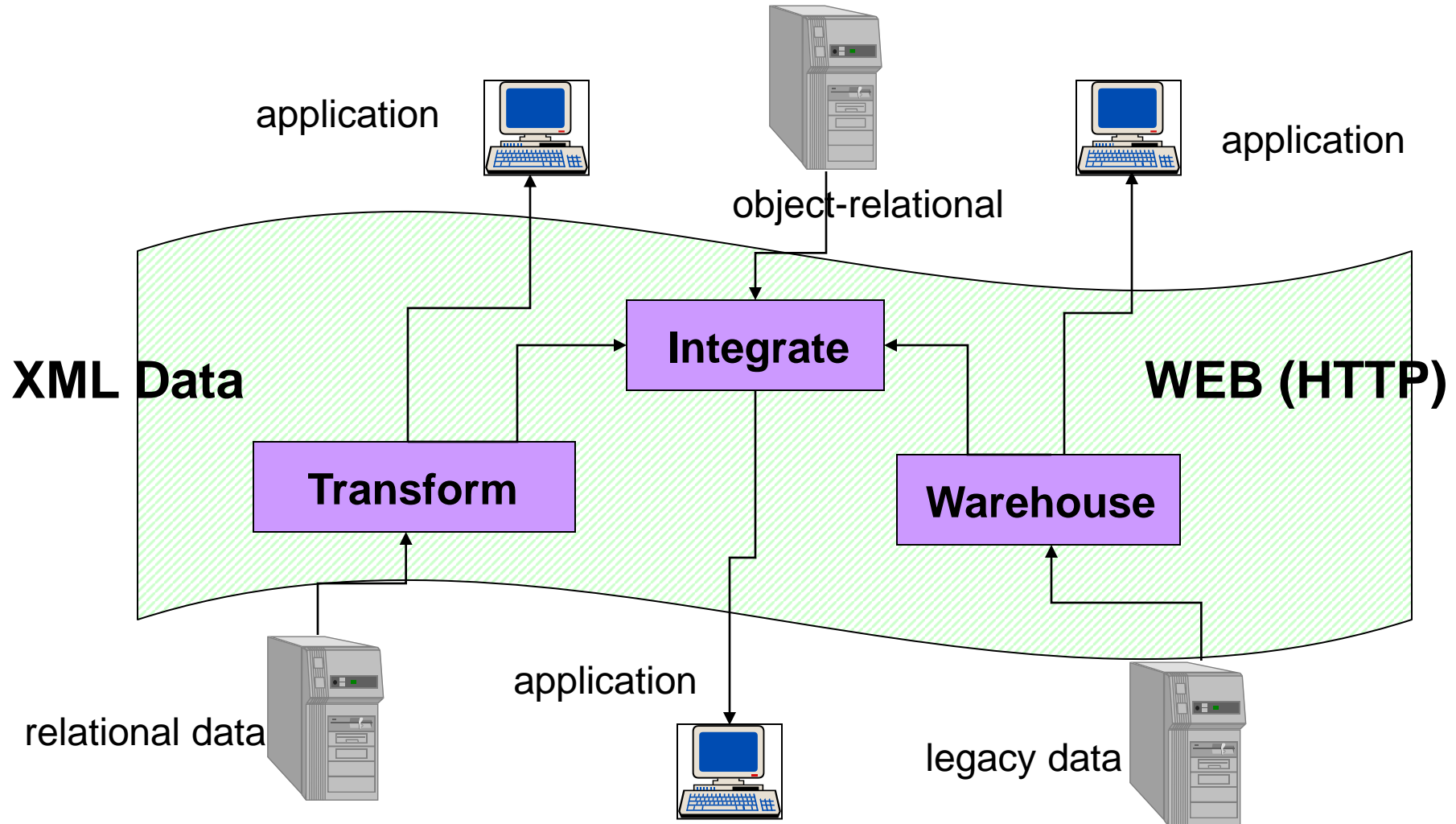
# SGML

- Derived from IBM's GML (generalized ML) developed in 1960's
  - Charles Goldfarb, Edward Mosher, and Raymond Lorie
  - For sharing of large-project documents
- Basis for HTML & XML
  - XML is roughly an augmented subset (adds more restrictions)
  - HTML is an application of SGML

# Why XML is of Interest to Us

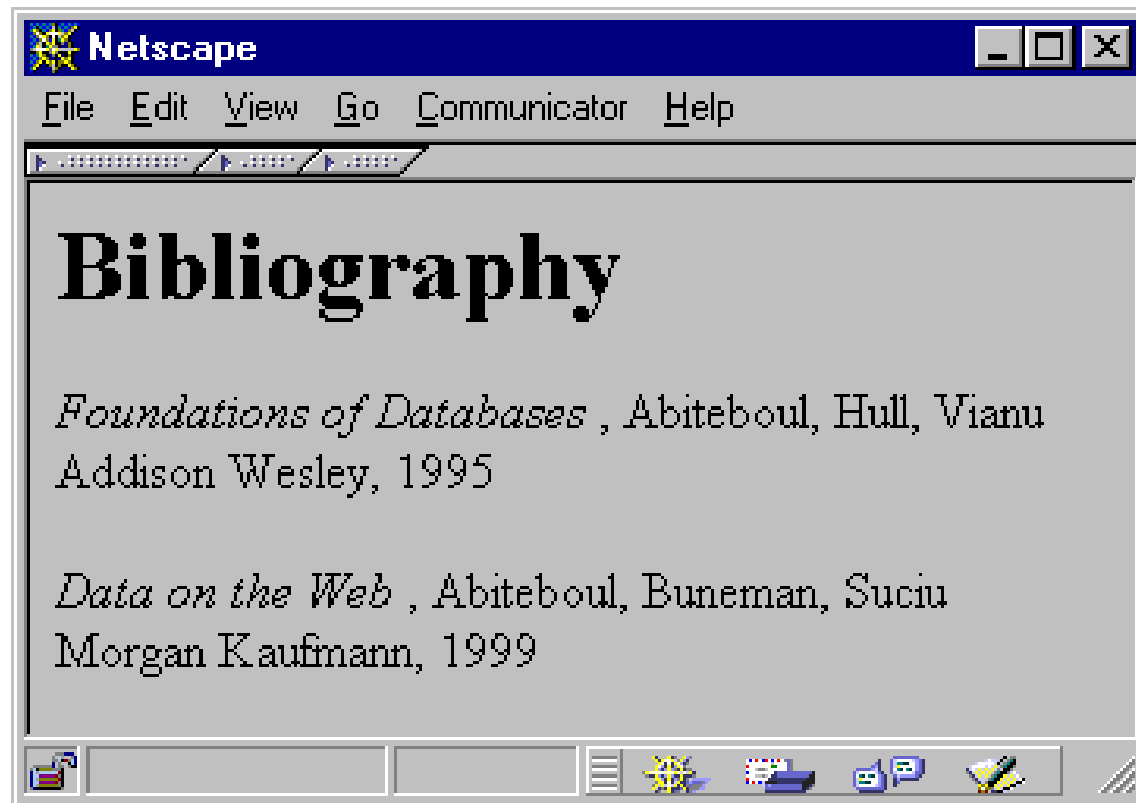
- XML is a syntax (serialization format) for data
- This is exciting because:
  - Can translate *any* data to XML
  - Can ship XML over the Web (HTTP)
  - Can input XML into any application
  - Thus: data sharing and exchange on the Web

# XML Data Sharing and Exchange



## Specific data management tasks

# From HTML to XML



HTML describes the presentation



# HTML

<h1> Bibliography </h1>

<p> <i> Foundations of Databases </i>

Abiteboul, Hull, Vianu

<br> Addison Wesley, 1995

<p> <i> Data on the Web </i>

Abiteoul, Buneman, Suciu

<br> Morgan Kaufmann, 1999

# 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 describes the content

# Web Services

- A software system designed to support interoperable machine-to-machine interaction over a network (from Wikipedia)
- Use http for machine-machine communications of files
  - E.g., in XML & JSON formats

# Ajax

- Asynchronous Javascript and XML
- Web clients send and receive data from server asynchronously
  - Benefit: more responsive web pages
- Common to use XML, JSON as data format

# Ajax in action ([link](#))

Amazon Associates SiteStripe

Get Link: [Text](#) [Image](#) [Text+Image](#) Share: [f](#) [t](#) [Earnings](#) [Help](#)

NEW & INTERESTING FINDS ON AMAZON [EXPLORE](#)

**amazon** Prime

Shop \$5 lobster

Departments ▾ Browsing History ▾ Wensheng's Amazon.com Hello, Wensheng Your Account ▾ Prime ▾ Lists ▾

1-16 of 15,749 results for "data mining" Sort by [Relevance](#)

☐ | FREE One-Day  
Get FREE One-Day Delivery on qualifying orders over \$35

Show results for

**Books** ▸  
Data Mining  
Computers & Technology  
Artificial Intelligence & Semantics  
Reference  
Probability & Statistics

**Apress Featured Resources in Professional Computing**  
Sponsored by Apress. Explore these professional computing resources.

**Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (Morgan Kaufmann Series in Data Management Systems)** by Ian H. Witten and Eibe Frank

Paperback  
**\$36.12** ~~\$69.95~~ | FREE One-Day  
Get it by **Tomorrow, Sep 29**  
More Buying Choices

★★★★☆ ▾ 73  
Trade in yours for an Amazon Gift Card up to \$12.61

# XML Terminology

- tags: **book**, **title**, **author**, ...
- start tag: **<book>**, end tag: **</book>**
- elements: **<book>...</book>**, **<author>...</author>**
- elements may be nested:  
**<book><author></author></book>**
- empty element (no content): **<red></red>** abbrev. **<red/>**
  - Note that an empty element can have attributes
  - **<author age="25"/>**
- an XML document: has a single *root element*
- ***Element names are case-sensitive!***

***Well-formed XML document: if it has matching tags***

# More XML: Attributes

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

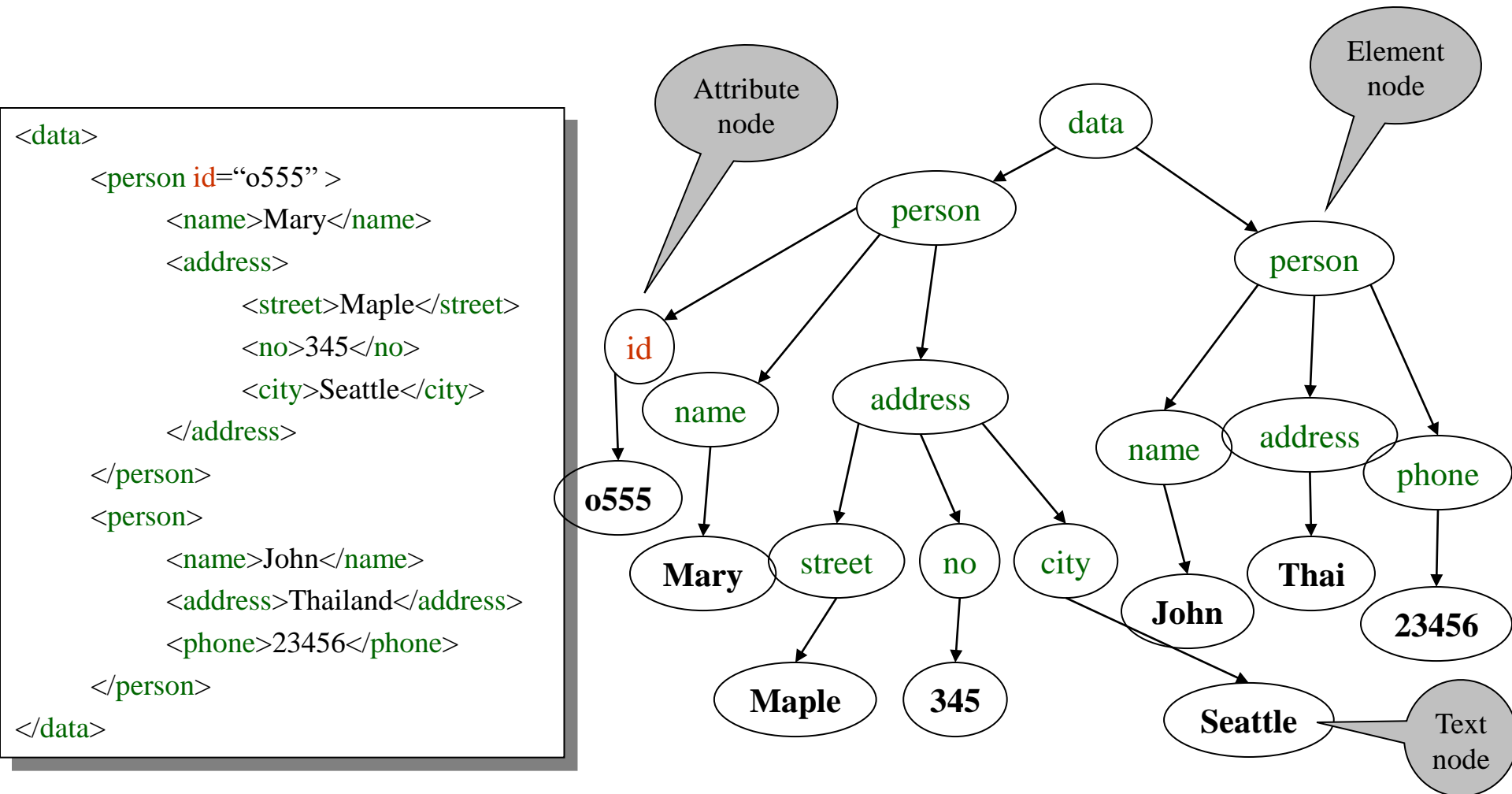
attributes are alternative ways to represent data

# Attributes

- `<book price = '55' currency = "USD">`
- Attribute values must be quoted, either double or single



# XML structure: an ordered tree



Order matters & index starts from 1 !!!

# XML Data

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

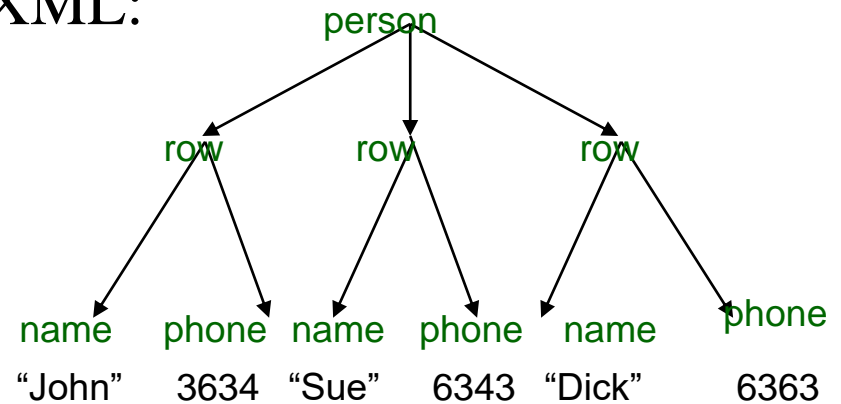
# Relational Data as XML

person

n a m e	p h o n e
J o h n	3 6 3 4
S u e	6 3 4 3
D i c k	6 3 6 3

Product

XML:



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

# 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	NULL

# 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 structures
- Heterogeneous contents:
  - <bib> contains both <book>'s and <cd>'s

# Document Type Definitions

## DTD

- A set of markup for describing schema of XML data
- an XML document may have a DTD
- XML document:
  - well-formed** = if tags are correctly closed
  - valid** = if it has a DTD and conforms to it
- validation is useful in data exchange

# Very Simple DTD

→ Root element

```
<!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)>  
>
```



# DTD as Part of XML Document

```
<?xml version="1.0"?>
```

```
<!DOCTYPE note
```

```
  [<!ELEMENT note (to,from,heading,body)>
```

```
    <!ELEMENT to (#PCDATA)>
```

```
    <!ELEMENT from (#PCDATA)>
```

```
    <!ELEMENT heading (#PCDATA)>
```

```
    <!ELEMENT body (#PCDATA)>]>
```

```
<note>
```

```
  <to>Tove</to>
```

```
  <from>Jani</from>
```

```
  <heading>Reminder</heading>
```

```
  <body>Don't forget me this weekend</body>
```

```
</note>
```

# XML schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">

  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

# Example XML for Company DTD

Example of 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

`<!ELEMENT tag (CONTENT)>`

content  
model

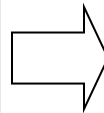
- Content model:
  - Complex = a regular expression over other elements
  - Text-only = #PCDATA/#CDATA
  - Empty = EMPTY
  - Any = ANY
  - Mixed content = (#PCDATA | A | B | C)\*
- #CDATA (#PCDATA)
  - Character data not are (are) parsed by parser
  - Tags inside #PCDATA will be treated as markup

# DTD: Regular Expressions

DTD

sequence

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



XML

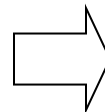
```
<name>  
  <firstName> ..... </firstName>  
  <lastName> ..... </lastName>  
</name>
```

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)))
```

# Processing instructions

- `<?xml version="1.0" encoding="UTF-8"?>`
- This is the first line of an XML document
  - Declaring that the following is an XML doc...
  - that follows standard version 1.0
  - and whose encoding is UTF-8

# Agenda

- XML:
  - What is it and why do we care?
  - Data model
  - Query language: XPath

# Querying XML Data

- XPath = simple navigation through the tree
- XQuery = the SQL of XML



<bib>

...

<book price="35">

<publisher>Addison-Wesley</publisher>

<author>Serge Abiteboul</author>

<author><first-name>Rick</first-name><last-name>Hull</last-name></author>

<author age="20">Victor Vianu</author>

<title>Foundations of Databases</title>

<year>1995</year>

<price>38.8</price>

</book>

<book price="55">

<publisher>Freeman</publisher>

<author>Jeffrey D. Ullman</author>

<title>Principles of Database and Knowledge Base Systems</title>

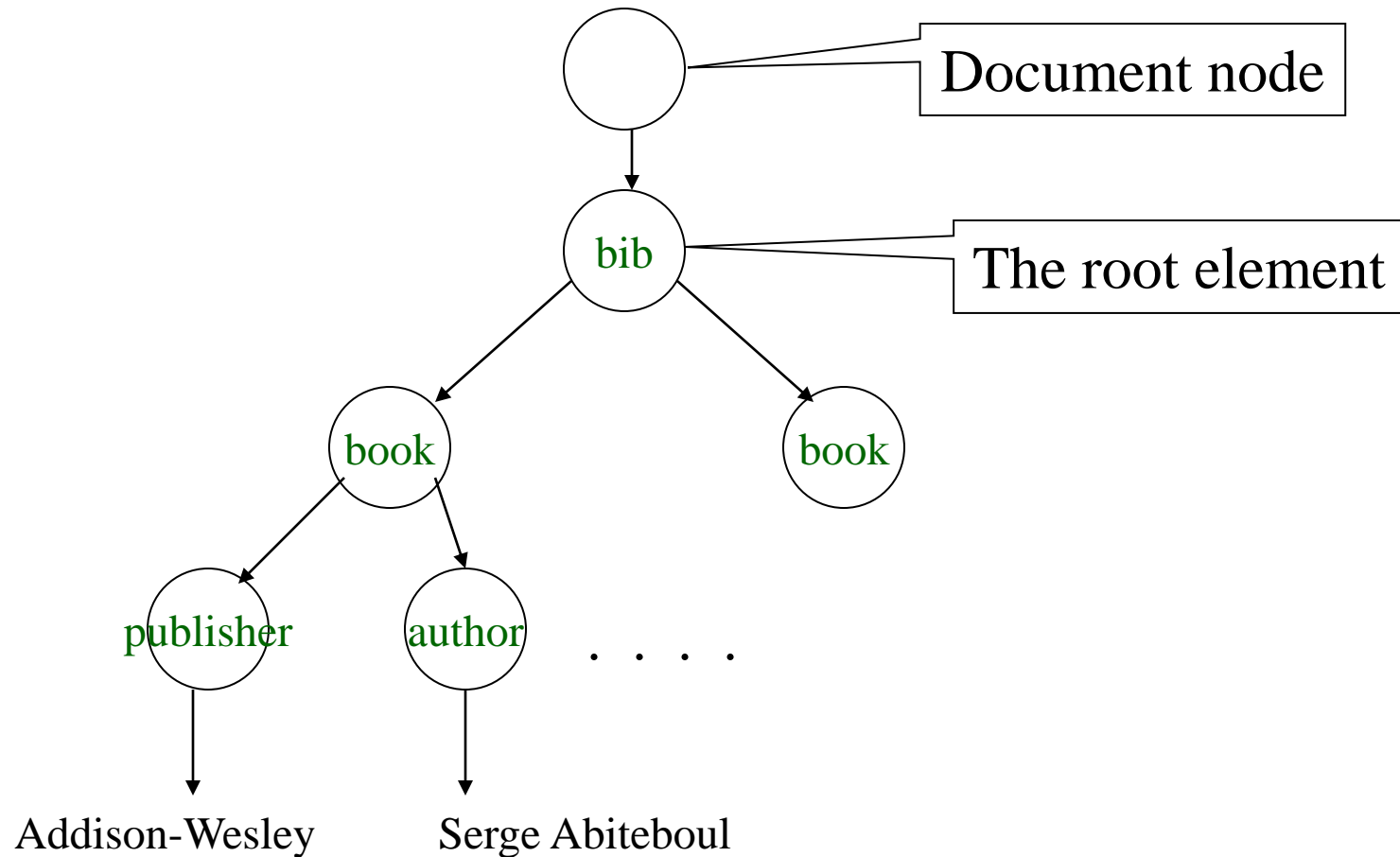
<year>1998</year>

</book>

...

</bib>

# Data Model for XPath



# XPath: Simple Expressions

`/bib/book/year`

Result: `<year> 1995 </year>`  
`<year> 1998 </year>`

`/bib/paper/year`

Result: empty (there were no papers)

# //: finding descendants

//author

Result: <author> Serge Abiteboul </author>  
          <author> <first-name> Rick </first-name>  
                    <last-name> Hull </last-name>  
          </author>  
          <author> Victor Vianu </author>  
          <author> Jeffrey D. Ullman </author>

/bib//first-name

Result: <first-name> Rick </first-name>

# Select Child by Index

- Index of children starts from 1
- `//author[1]`
- `/bib/book[2]/author`

# Xpath: Text Nodes

```
/bib/book/author/text()
```

Result: Serge Abiteboul  
Victor Vianu  
Jeffrey D. Ullman

Rick Hull doesn't appear because he has `firstname`, `lastname` elements

## Functions in XPath:

- `text()` = matches text nodes
- `*` = matches only element nodes
- `node()` = matches any node (element or text)

# Xpath: Wildcard

`//author/*`

Result: `<first-name>` Rick `</first-name>`  
`<last-name>` Hull `</last-name>`

\* Matches any element

# Xpath: Attribute Nodes

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

Result: ['35', '55']

@price means that price has to be an attribute

Is it the same as ?

```
/bib/book[@price]
```



# Xpath: Attribute nodes

- `/bib/book/@*`
  - Return all attribute nodes of book elements
- Result:
  - `['35', '55']`

# Xpath: Predicates

```
/bib/book/author[first-name]
```

Return author elements (under /bib/book) which have a child element called "first-name"

Result: `<author> <first-name> Rick </first-name>  
          <last-name> Hull </last-name>  
          </author>`

# Xpath: More Predicates

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

Return lastname of author elements which have child element firstname and child element "address" which itself has ...

Result: <lastname> ... </lastname>  
          <lastname> ... </lastname>

# // inside predicate

`/bib/book[author//first-name]`

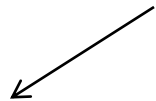
`/bib/book[//first-name]`

// starts from root here!

# Xpath: More Predicates

```
/bib/book[@price < 60]
```

```
/bib/book[author/@age < 25]
```



```
/bib/book[author/text()]
```

Return books under bib that have an author element with a text node

# Xpath: More Predicates

```
/bib/book[contains(author, 'Ullman')]
```




Return books under bib whose (*first*) author subelement contains the word 'Ullman' in its text node  
(note contains is case-sensitive)

What about `//book/author[contains(., "Ullman")]` ?

# Xpath: More Predicates

- `/bib/book[author = "Victor Vianu"]`
- `/bib/book[author/text() = "Victor Vianu"]`
- `/bib/book/author[. = 'Victor Vianu']`

# Xpath: More Predicates

- `/bib/book[price > 30 or year > 1995]`
- `/bib/book[price > 30 and year >= 1995]`
- `/bib/book[not(price > 30)]`  **Parenthesis required for not**
- Note: **and, or, not** should be all lowercases



# Xpath: More Predicates

- `/bib/book[not(publisher)]`
- What about `/bib/book[author[not(node())]]?`


# Xpath: alternatives

```
/bib/book|/bib/cd
```

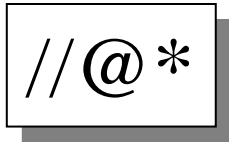
Return book and cd elements under /bib

# Questions

What do these return?



```
//*
```



```
//@*
```

# Resources

- Comparison of SGML and XML
  - <https://www.w3.org/TR/NOTE-sgml-xml-971215/>
- XML
  - <http://www.w3schools.com/xml/default.asp>
- XPath
  - [http://www.w3schools.com/xml/xml\\_xpath.asp](http://www.w3schools.com/xml/xml_xpath.asp)

# Resources

- Testers
  - <https://codebeautify.org/Xpath-Tester> (no support for alternation such as `"/bib/(book|cd)"`, but `/bib/book|/bib/cd` is ok)
  - <https://www.freeformatter.com/xpath-tester.html> ( no support for "contains", but support both forms of alternations above)
  - <http://www.xpathtester.com/xpath>

```
<data>
  <person id="100" >
    <name> Mary </name>
    <address>
      <street>Maple</street>
      <no>345</no>
      <city>Seattle</city>
    </address>
    <age>25</age>
  </person>
  <person id="200">
    <name>John</name>
    <address>Thailand </address>
    <phone>23456 </phone>
    <age>30</age>
  </person>
</data>
```