Middleware and Distributed Systems

Syed Gillani

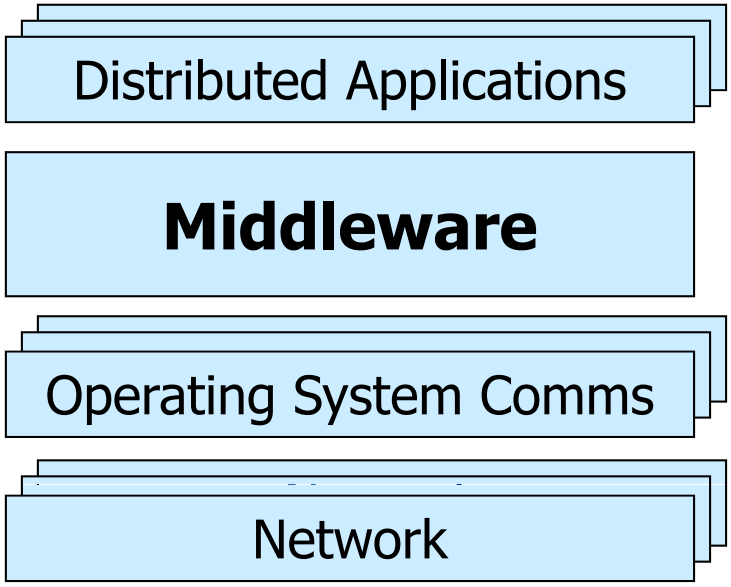
[What is this course about?]

- Intro to the workings of Distributed Systems (DS)
- How to communicate between clients and DS (the middleware)
- Working with different design models of middleware
- How to Implement some of the middleware models (in Java)!!

You should have the basic knowledge of Java

Introduction to Middleware I

- What is Middleware?
  - Layer between OS and distributed applications
  - Hides complexity and heterogeneity of distributed system
  - Bridges gap between low-level OS communications and programming language abstractions
  - Provides common programming abstraction and infrastructure for distributed applications
  - Overview at: *http://www.middleware.org*



(remote calls, object invocation, messages, …)

(sockets, IP, TCP, UDP, …)

(packets, bits…)

Introduction to Middleware II

- Middleware provides support for (some of):
  - Naming, Location, Service discovery, Replication
  - Protocol handling, Communication faults, QoS
  - Synchronisation, Concurrency, Transactions, Storage
  - Access control, Authentication

- Middleware dimensions:
  - Request/Reply vs. Asynchronous Messaging
  - Language-specific vs. Language-independent
  - Proprietary vs. Standards-based
  - Small-scale vs. Large-scale
  - Tightly-coupled vs. Loosely-coupled components

[The Web/WWW]

- …is a distributed document delivery system that uses Internet protocols
- …links documents stored in computers communicating by the Internet
- Main authority is the W3 Consortium
  - www.w3.org

[The Web/WWW]

What/Why is THE WEB?

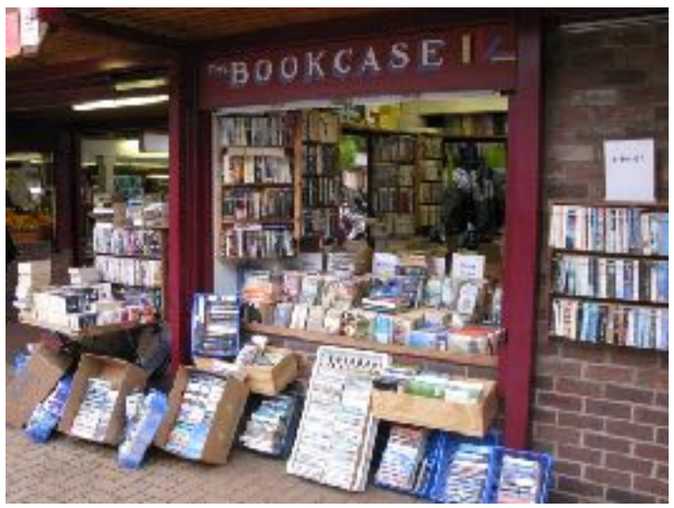
[The Web / WWW]

Publish Your Book

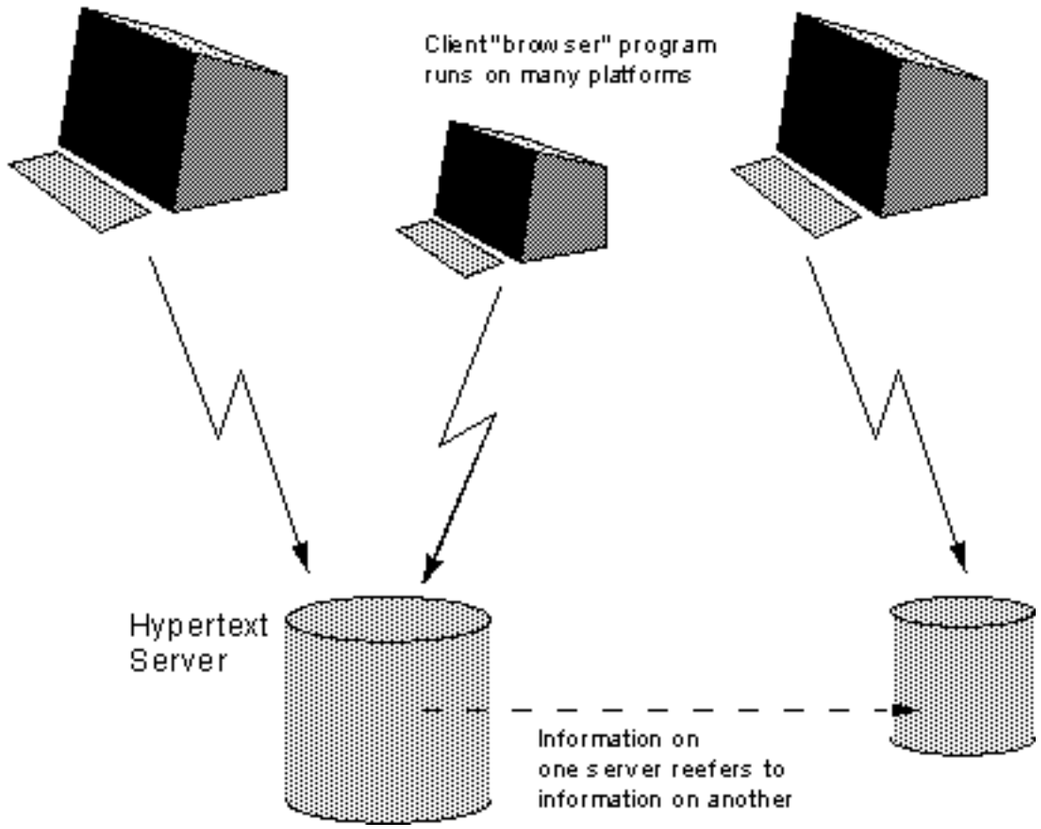
Send it to the Publisher

Print it and
send it to the Book Shop

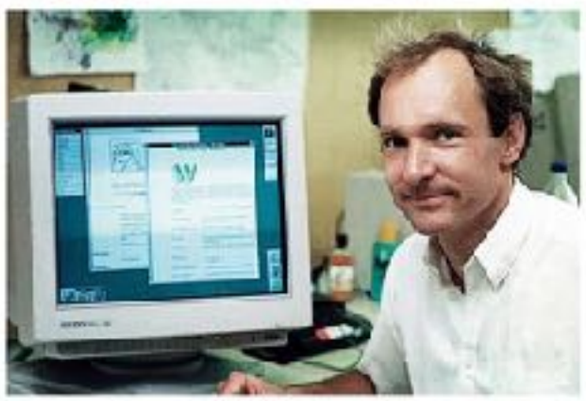
A small audience
will read it



[The Web / WWW]



Tim Berners Lee



Needs something over TCP to send information (**HTTP**)

How to structure information (**HTML**)

[The Web]

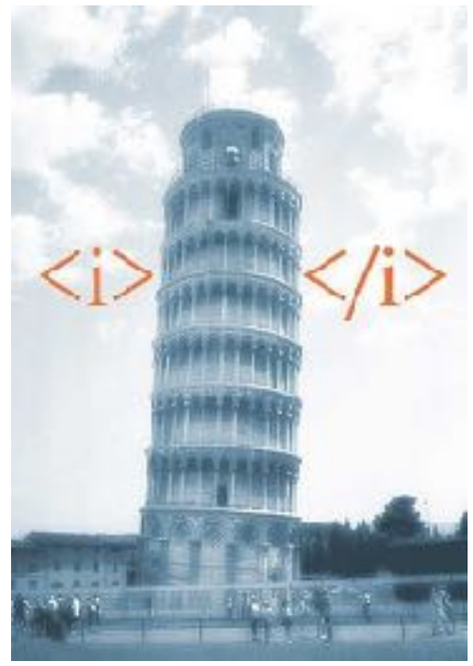
Distribution of information with global access at low cost!!

- Two Primarily Requirements
  - ✓ Machine Readable Structure
  - ✓ Global System of Interlinking such structural documents

- How?
  - ✓ Hyper Text Markup Language (HTML)
  - ✓ Global Unique Addresses: Unique Resource Locators (URLs)

[The Web]

HTML was designed to **display** data and how data should **look**



How to describe data, its semantics, most importantly How to Query such structural data?

Middleware mostly handle such questions: **Formats** (structures) to describe data, **send/receive** such data (protocols), **generate** such data (programs and databases)

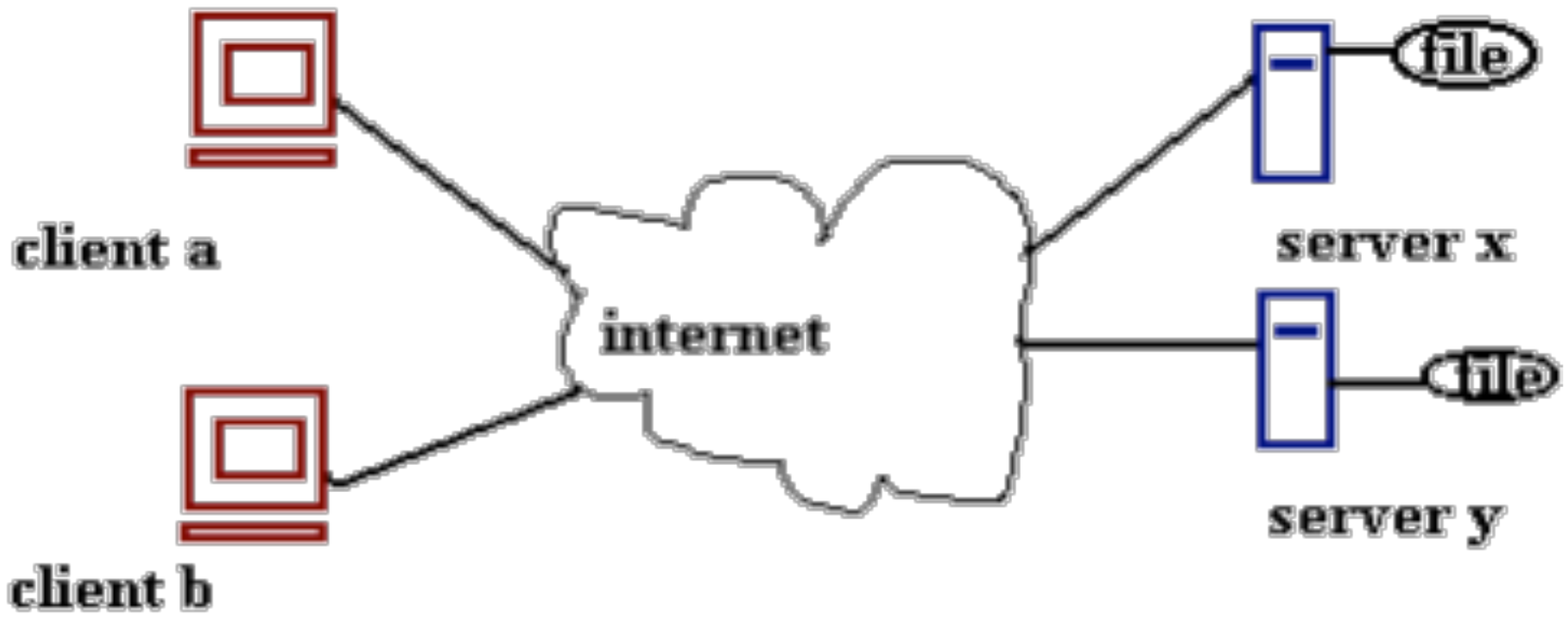
[Basic Definitions]

- *Web server* – machine that services Internet request
- *Web client* – machine that initiates Internet request
- *Browser* – software to interact with Internet data at the web client
- *TCP/IP* – internet data protocol
- *FTP* – internet file transfer protocol
- *HTTP* – hypertext transfer protocol
- *HTML* – hypertext markup language

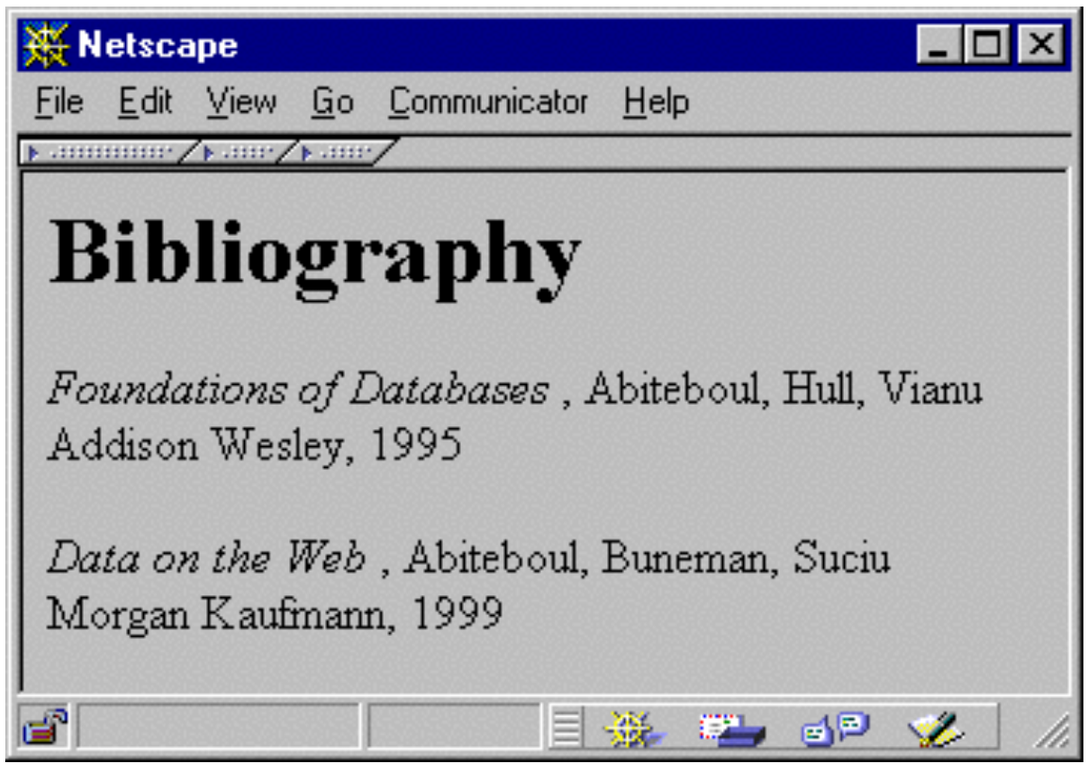
[Servers and Clients]

- *Servers* – computer systems at the end of a network that store files and provide other services
- *Clients* – computer systems that are end points for users of the data

[Servers and Clients]

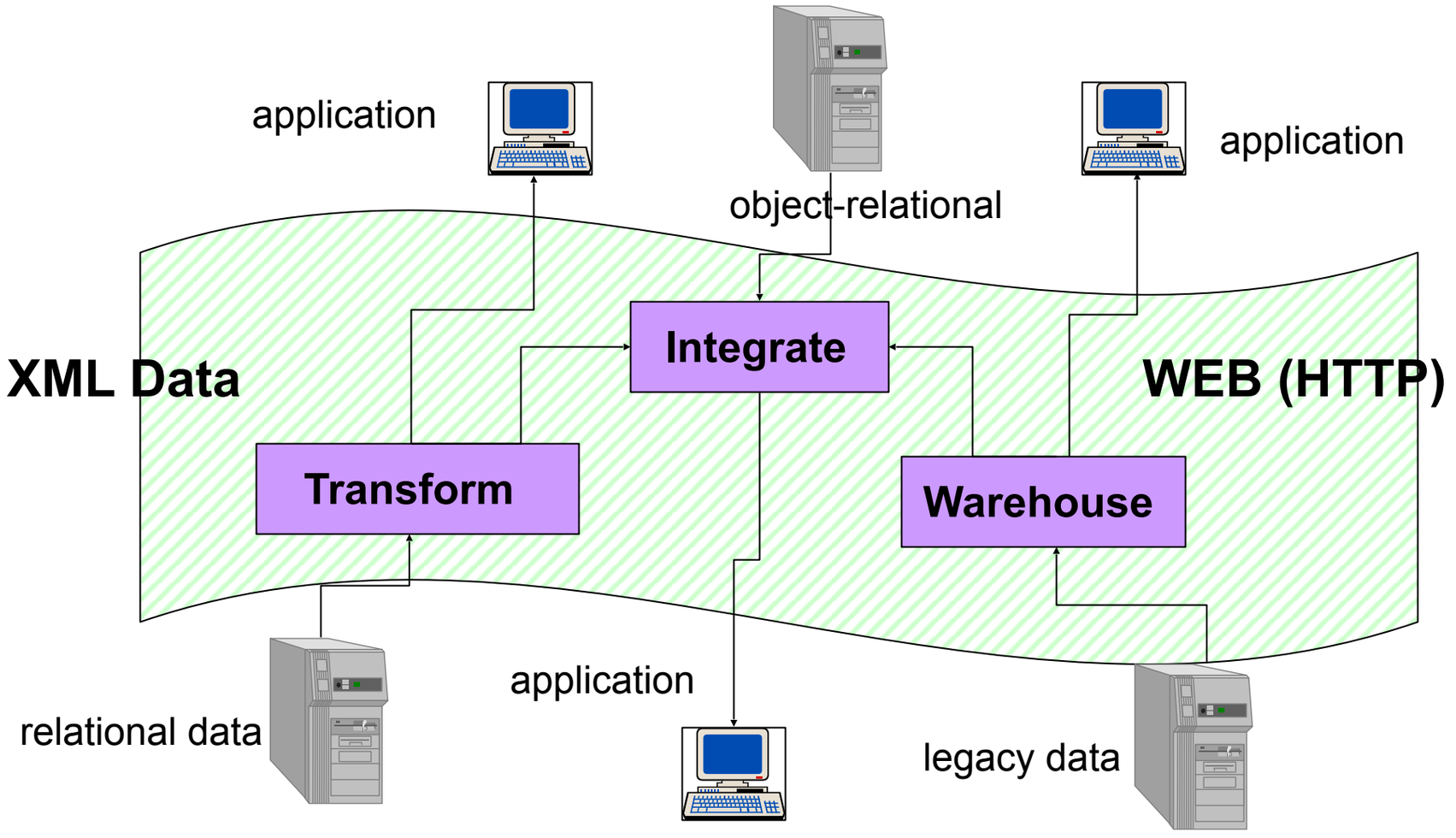


[From HTML to XML]



HTML describes the presentation

[XML Data Sharing and Exchange]



[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

[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

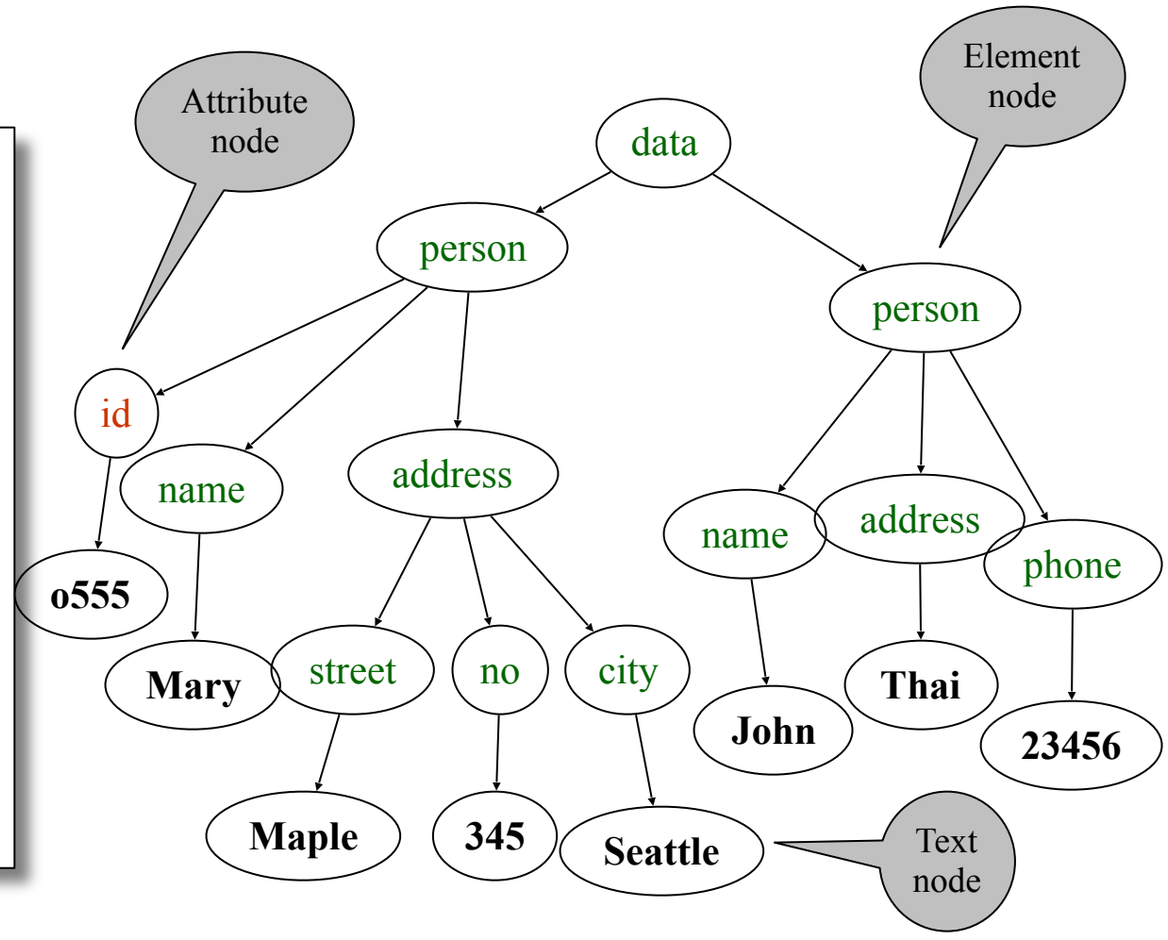
[More XML: Oids and References]

```
<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]

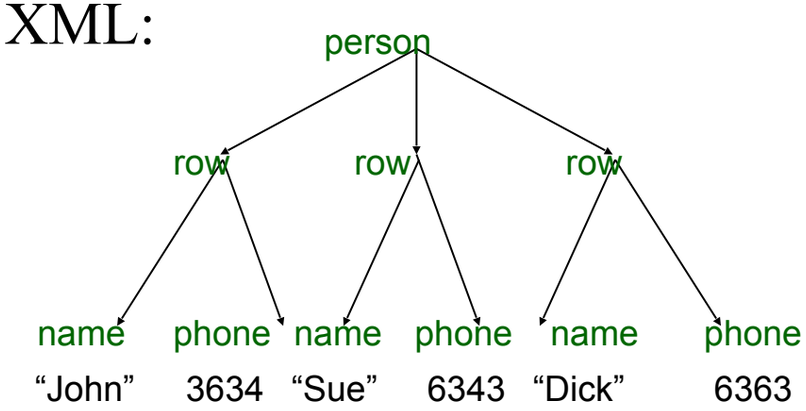
- ✓ XML is self-describing
- ✓ Schema elements become part of the data
  - ✓ Reational schema: persons(name,phone)
  - ✓ In XML <persons>, <name>, <phone> are part of the data, and are repeated many times
- ✓ Consequence: XML is much more flexible
- ✓ XML = semistructured data

[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]

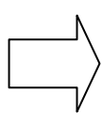
- Content model:
  - ✓ Complex = a regular expression over other elements
  - ✓ Text-only = #PCDATA
  - ✓ Empty = EMPTY
  - ✓ Any = ANY
  - ✓ Mixed content = (#PCDATA | A | B | C)*

[DTD: The Regular Expression]

sequence

DTD

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



XML

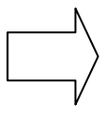
```
<name>
   <firstName>  . . . . . </firstName>
   <lastName>  . . . . . </lastName>
</name>
```

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)))
```