

XML

Semistructured Data
Extensible Markup Language
Document Type Definitions

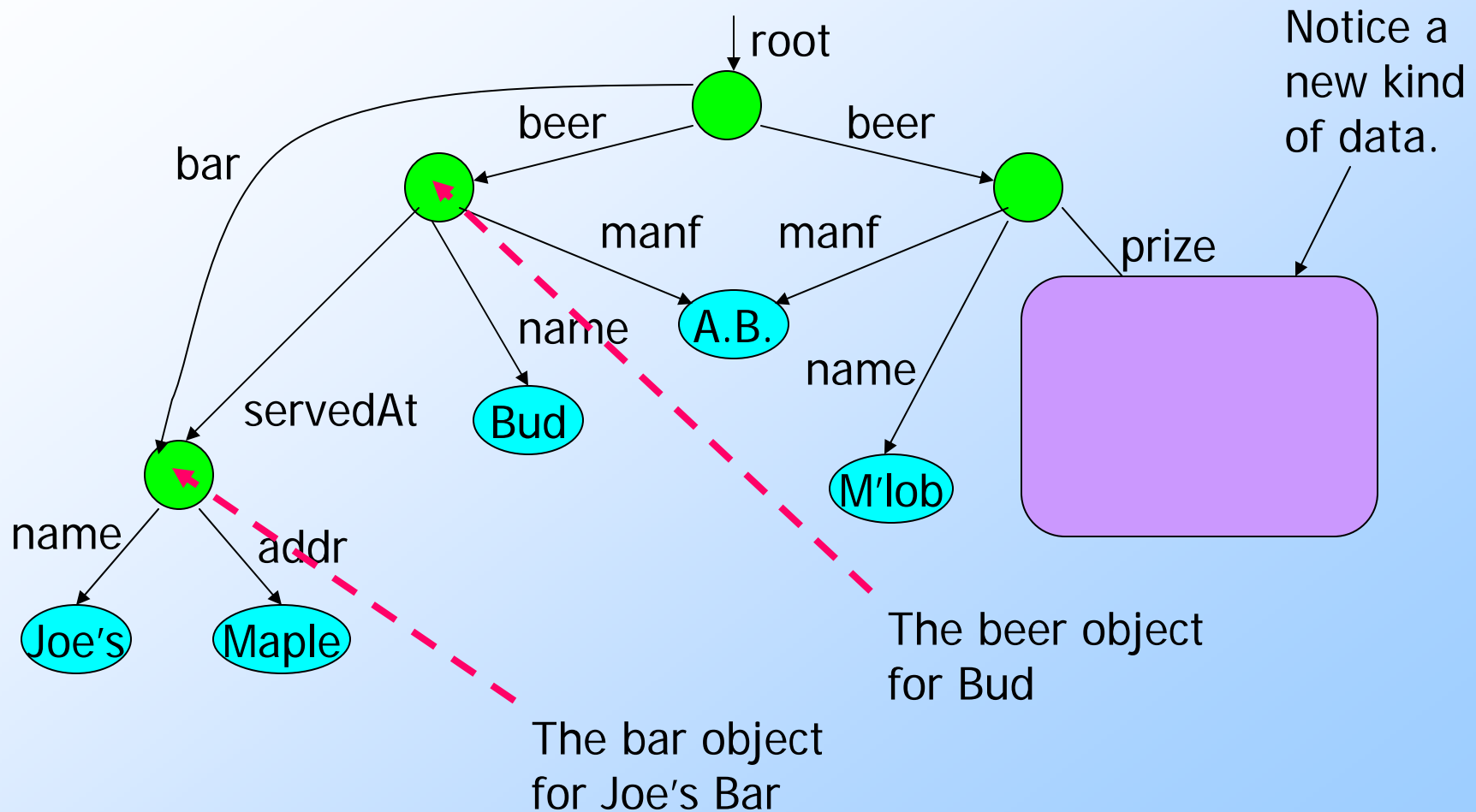
Semistructured Data

- ◆ Another data model, based on trees.
- ◆ **Motivation**: flexible representation of data.
 - ◆ Often, data comes from multiple sources with differences in notation, meaning, etc.
- ◆ **Motivation**: sharing of *documents* among systems and databases.

Graphs of Semistructured Data

- ◆ Nodes = objects.
- ◆ Labels on arcs (attributes, relationships).
- ◆ Atomic values at leaf nodes (nodes with no arcs out).
- ◆ Flexibility: no restriction on:
 - ◆ Labels out of a node.
 - ◆ Number of successors with a given label.

Example: Data Graph



XML

- ◆ XML = *Extensible Markup Language*.
- ◆ While HTML uses tags for formatting (e.g., "italic"), XML uses tags for semantics (e.g., "this is an address").
- ◆ **Key idea**: create tag sets for a domain (e.g., genomics), and translate all data into properly tagged XML documents.

Well-Formed and Valid XML

- ◆ *Well-Formed XML* allows you to invent your own tags.
 - ◆ Similar to labels in semistructured data.
- ◆ *Valid XML* involves a DTD (*Document Type Definition*), a grammar for tags.

Well-Formed XML

- ◆ Start the document with a *declaration*, surrounded by `<?xml ... ?>` .
- ◆ Normal declaration is:
`<?xml version = "1.0"
standalone = "yes" ?>`
 - ◆ "Standalone" = "no DTD provided."
- ◆ Balance of document is a *root tag* surrounding nested tags.

Tags

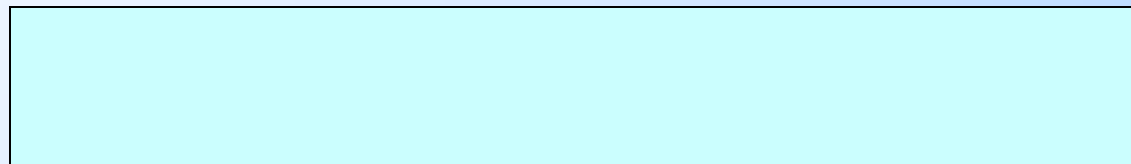
- ◆ Tags, as in HTML, are normally matched pairs, as `<FOO> ... </FOO>` .
- ◆ Tags may be nested arbitrarily.
- ◆ XML tags are case sensitive.

Example: Well-Formed XML

<?xml version = "1.0" standalone = "yes" ?>



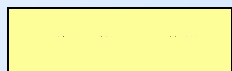
A NAME
subobject



<BEER><NAME>Miller</NAME>

<PRICE>3.00</PRICE></BEER>

A BEER
subobject



<BAR> ...

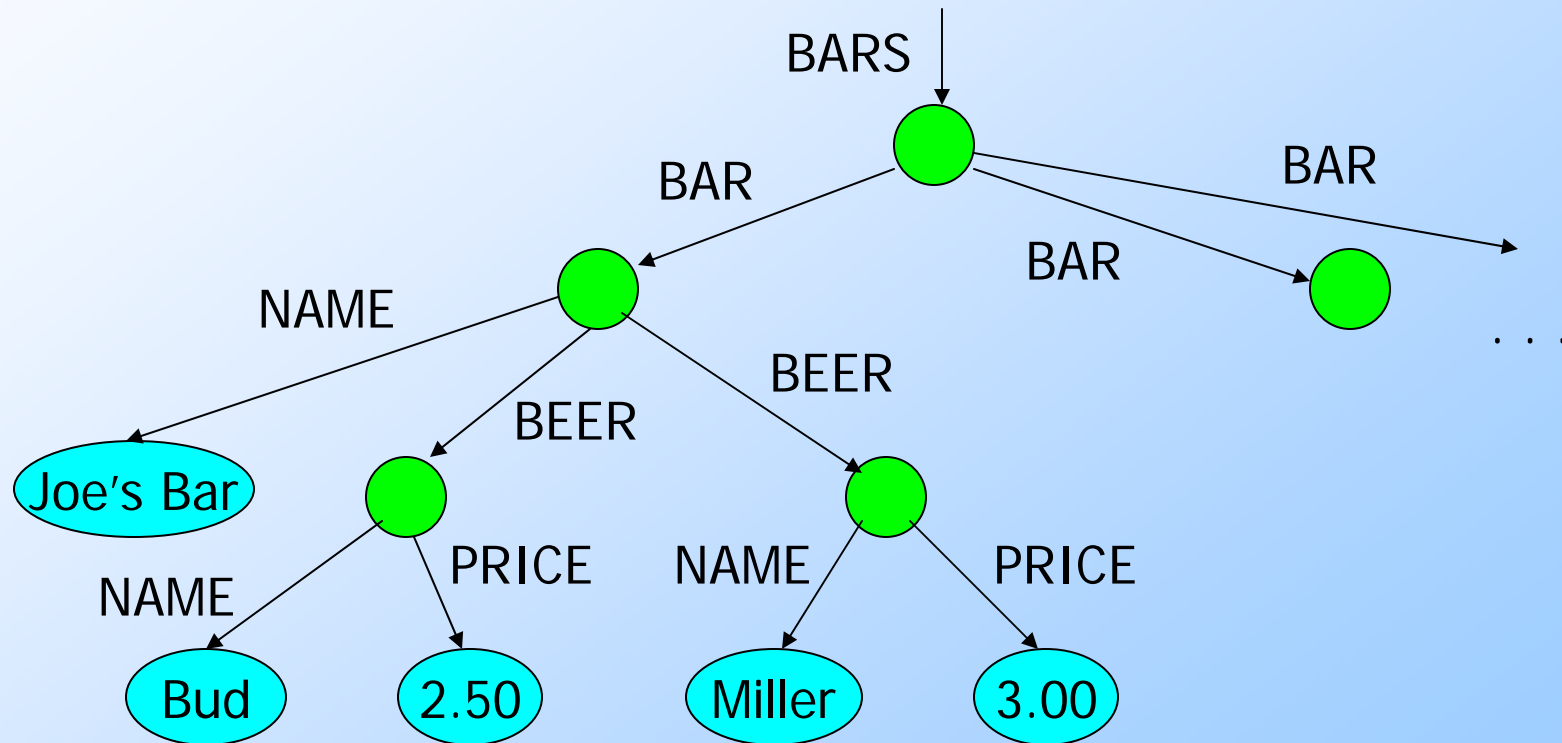


XML and Semistructured Data

- ◆ Well-Formed XML with nested tags is exactly the same idea as trees of semistructured data.
- ◆ We shall see that XML also enables nontree structures, as does the semistructured data model.

Example

◆ The <BARS> XML document is:



DTD Structure

```
<!DOCTYPE <root tag> [  
  <!ELEMENT <name> ( <components> ) >  
  . . . more elements . . .  
>
```

DTD Elements

- ◆ The description of an element consists of its name (tag), and a parenthesized description of any nested tags.
 - ◆ Includes order of subtags and their multiplicity.
- ◆ Leaves (text elements) have #PCDATA (*Parsed Character DATA*) in place of nested tags.

Example: DTD

```
<!DOCTYPE BARS [
```

```
<!ELEMENT
```

```
<!ELEMENT
```

```
<!ELEMENT
```

```
<!ELEMENT
```

```
<!ELEMENT
```

```
]>
```

NAME and PRICE
are text.

A BARS object has
zero or more BAR's
nested within.

A BAR has one
NAME and one
or more BEER
subobjects.

A BEER has a
NAME and a
PRICE.

Element Descriptions

- ◆ Subtags must appear in order shown.
- ◆ A tag may be followed by a symbol to indicate its multiplicity.
 - ◆ * = zero or more.
 - ◆ + = one or more.
 - ◆ ? = zero or one.
- ◆ Symbol | can connect alternative sequences of tags.

Example: Element Description

- ◆ A name is an optional title (e.g., "Prof."), a first name, and a last name, in that order, or it is an IP address:

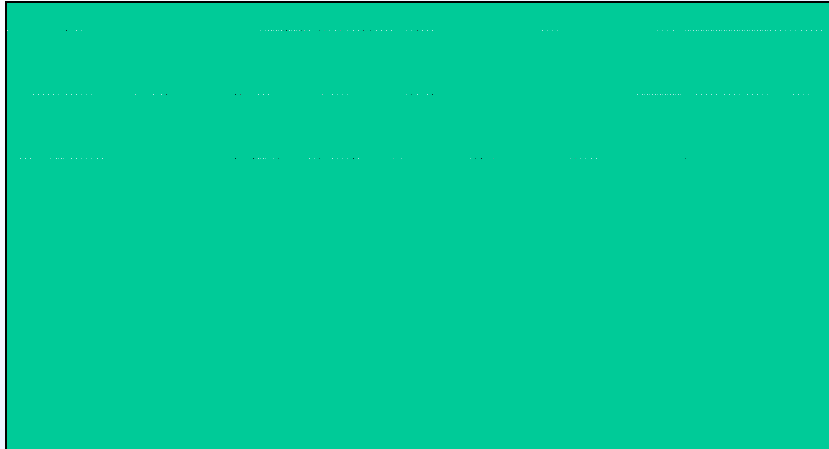
```
<!ELEMENT NAME (  
    (TITLE?, FIRST, LAST) | IPADDR  
)>
```


Use of DTD's

1. Set standalone = "no".
2. Either:
 - a) Include the DTD as a preamble of the XML document, or
 - b) Follow DOCTYPE and the <root tag> by SYSTEM and a path to the file where the DTD can be found.

Example (a)

<?xml version = "1.0" standalone = "no" ?>



The DTD

The document



Example (b)

◆ Assume the BARS DTD is in file bar.dtd.

```
<?xml version = "1.0" standalone = "no" ?>
```

```
<[REDACTED]>
```

```
<BARS>
```

```
  <BAR><NAME>Joe's Bar</NAME>
```

```
    <BEER><NAME>Bud</NAME>
```

```
      <PRICE>2.50</PRICE></BEER>
```

```
  <BEER><NAME>Miller</NAME>
```

```
    <PRICE>3.00</PRICE></BEER>
```

```
  </BAR>
```

```
  <BAR> ...
```

```
</BARS>
```

Get the DTD
from the file
bar.dtd

Attributes

- ◆ Opening tags in XML can have *attributes*.

- ◆ In a DTD,


`<!ATTLIST E . . . >`

declares an attribute for element E ,
along with its datatype.


Example: Attributes

- ◆ Bars can have an attribute `kind`, a character string describing the bar.

```
<!ELEMENT BAR (NAME BEER* )>
```

```
<!ATTLIST BAR kind  >
```


Attribute is optional
opposite: #REQUIRED



Character string
type; no tags

Example: Attribute Use

- ◆ In a document that allows BAR tags, we might see:

```
<BAR kind =  >
  <NAME>Akasaka</NAME>
  <BEER><NAME>Sapporo</NAME>
    <PRICE>5.00</PRICE></BEER>
  ...
</BAR>
```

Note attribute values are quoted

ID's and IDREF's

- ◆ Attributes can be pointers from one object to another.
 - ◆ Compare to HTML's NAME = "foo" and HREF = "#foo".
- ◆ Allows the structure of an XML document to be a general graph, rather than just a tree.

Creating ID's

- ◆ Give an element E an attribute A of type ID.
- ◆ When using tag $\langle E \rangle$ in an XML document, give its attribute A a unique value.
- ◆ Example:

$\langle E \quad A = \text{"xyz"} \rangle$

Creating IDREF's

- ◆ To allow objects of type F to refer to another object with an ID attribute, give F an attribute of type IDREF.
- ◆ Or, let the attribute have type IDREFS, so the F -object can refer to any number of other objects.

Example: ID's and IDREF's

- ◆ Let's redesign our BARS DTD to include both BAR and BEER subelements.
- ◆ Both bars and beers will have ID attributes called `name`.
- ◆ Bars have SELLS subobjects, consisting of a number (the price of one beer) and an IDREF `theBeer` leading to that beer.
- ◆ Beers have attribute `soldBy`, which is an IDREFS leading to all the bars that sell it.

The DTD

Bar elements have name as an ID attribute and have one or more SELLS subelements.

```
<!DOCTYPE BARS [
```

```
  <!ELEMENT BARS (BAR*, BEER*)>
```

```
  <!ELEMENT BAR
```

```
    <!ATTLIST BAR
```

```
  <!ELEMENT SELLS
```

```
    <!ATTLIST SELLS
```

```
  <!ELEMENT BEER
```

```
    <!ATTLIST BEER
```

```
    <!ATTLIST BEER
```

```
]>
```

Explained
next

Beer elements have an ID attribute called name,
and a soldBy attribute that is a set of Bar names.

SELLS elements
have a number
(the price) and
one reference
to a beer.

Example Document

<BARS>

<BAR name = "JoesBar">

<SELLS theBeer = "Bud">2.50</SELLS>

<SELLS theBeer = "Miller">3.00</SELLS>

</BAR> ...

<BEER name = "Bud" soldBy = "JoesBar
SuesBar ..."/> ...

</BARS>

Empty Elements

- ◆ We can do all the work of an element in its attributes.
 - ◆ Like BEER in previous example.
- ◆ **Another example:** SELLS elements could have attribute `price` rather than a value that is a price.

Example: Empty Element

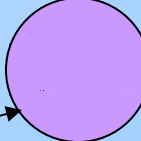
◆ In the DTD, declare:

```
<!ELEMENT SELLS EMPTY>
```

```
<!ATTLIST SELLS theBeer IDREF #REQUIRED>
```

```
<!ATTLIST SELLS price CDATA #REQUIRED>
```

◆ Example use:

```
<SELLS theBeer = "Bud" price = "2.50" 
```

Note exception to
"matching tags" rule

