Relations in SQL: Declaring keys

- An **attribute** or **list of attributes** may be declared PRIMARY KEY or UNIQUE
  - There is a very subtle difference
- Either form indicates that no two tuples of the relation may agree in all attributes on the list
- This is an example of a constraint in the data model.
- Example: **single-attribute key**

```
CREATE TABLE Beers(
    name    CHAR(20) UNIQUE,
    brewery VARCHAR(20),
    abv     REAL
);
```

Relations in SQL: Declaring keys

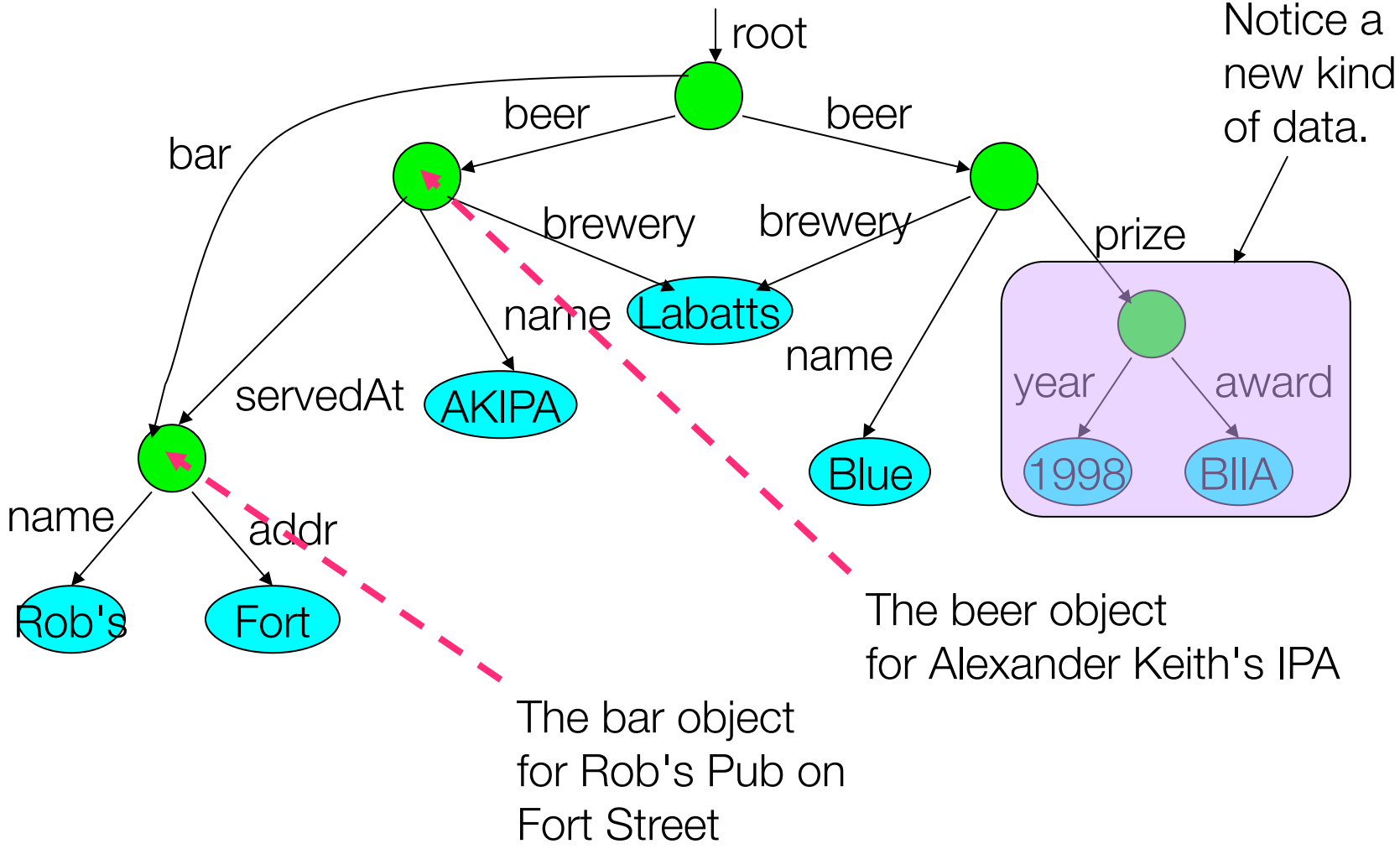
- Multi-attribute keys
  - Key declaration can also be another element in a CREATE TABLE statement
  - This form must be used for keys made up of more than one attribute.
  - However, it can also be used for one-attribute keys

```
CREATE TABLE Sells (
    pub    CHAR(20),
    beer   VARCHAR(20),
    price  REAL,
    PRIMARY KEY (pub, beer)
);
```

Semistructured Data

- We will spend a bit of time now with a **tree-based data model**
- Most widely-used flavour of this today: XML
  - (Long before XML, however, was SGML)
  - Tree-structured data models have been around for quite some time
- Two motivations for this:
  - Permits a more flexible representation of data (think "something other than flat structure")
  - Sharing of documents (with the meaning inherent in their structure) among systems and databases

Example: Data Graph (i.e., not quite a tree)



XML

- **Extensible Markup Language**
- Uses tags for adding semantics to document
  - Think of how HTML once used tags for formatting
  - (Now with HTML5 tags in HTML are meant to be for semantic structure.)
- Key idea:
  - Create tag sets for a domain (example: geomatics; math notation; western music notation; server configuration)
  - Translate all data for that domain into properly-tagged XML documents

XML documents

- Starting declaration is at top
- Followed by the root tag
- Root tag itself surrounds nested tags
- Tags are matched pairs: <foo> ... </foo>
  - An optional single-tag form also exists: <foo/>
- Tag nesting could be arbitrary
  - But tags may not overlap
- XML tags are case sensitive...

Example: an XML Document

```
<?xml version = “1.0” encoding = “utf-8” ?>
  <pubs>
    <pub><name>Rob's Pub</name>
        <beer><name>Amnesiac</name>
                <brewery>Phillips</brewery>
                <price>7.50</price>
        </beer>
        <beer><name>Big Tug IPA</name>
                <brewery>Driftwood</brewery>
                <price>6.75</price>
        </beer>
    </pub>
    <pub> …
  </pubs>
```

A name subobject

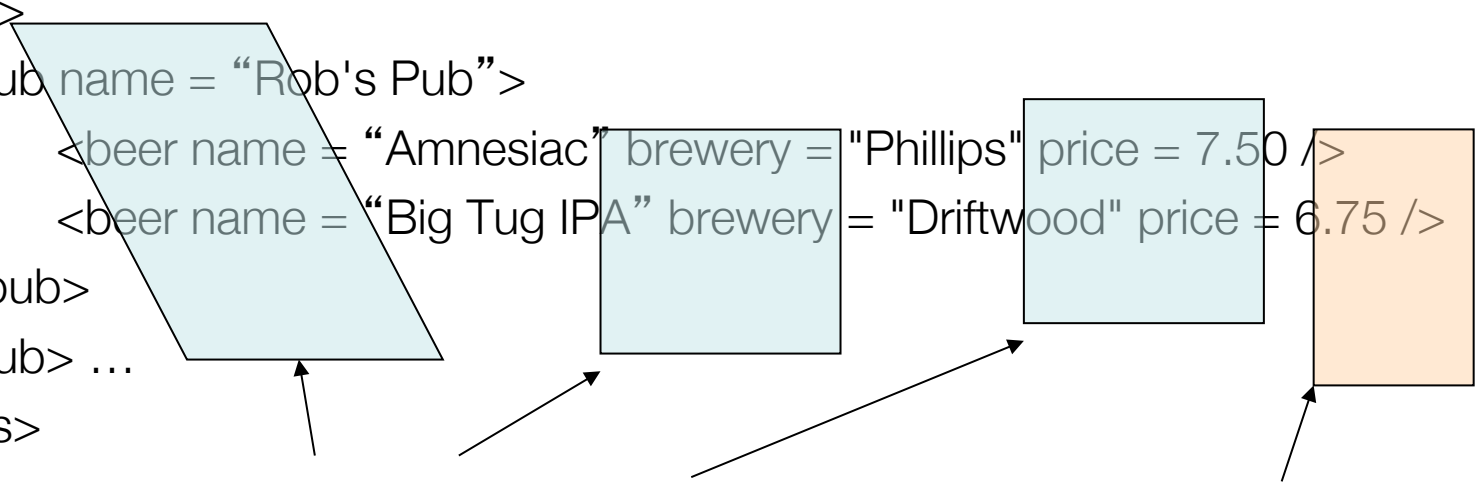
A beer subobject

XML documents: attributes

- An opening XML tag can have "attribute = value" pairs
- Attributes also allow for linking among elements
  - This is one way of obtaining a structure that is a more general graph rather than a strict tree
  - Won't worry about this right now...
- This suggests an alternative way of writing XML for the pubs...
  - Use tags for structure
  - Use attribute/value pairs for associating value

Pubs, but now using Attributes

```
<?xml version = “1.0” encoding = “utf-8” ?>
<pubs>
    <pub name = “Rob's Pub”>
        <beer name = “Amnesiac” brewery = "Phillips" price = 7.50 />
        <beer name = “Big Tug IPA” brewery = "Driftwood" price = 6.75 />
    </pub>
    <pub> …
</pubs>
```



name, **brewery** and **price** are attributes

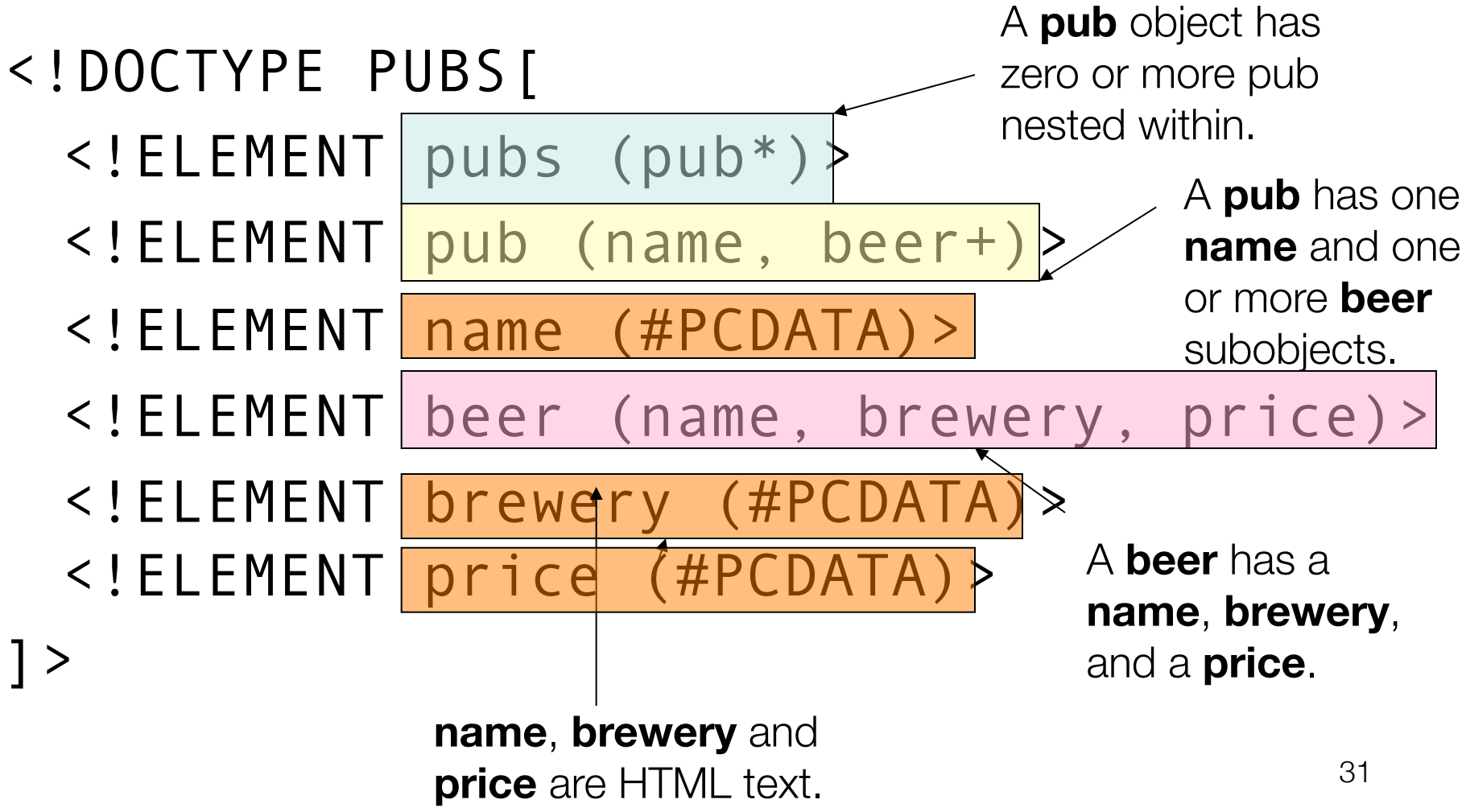
Notice Beer elements have only opening tags with attributes.

XML documents: DTDs

- We sometimes need a notation to capture the allowed structure for a set of XML files
  - Note that the "allowed structure" doc and actual XML file are often separate
- Structure within DTD seems to mirror that of the XML itself
- Definition form:

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

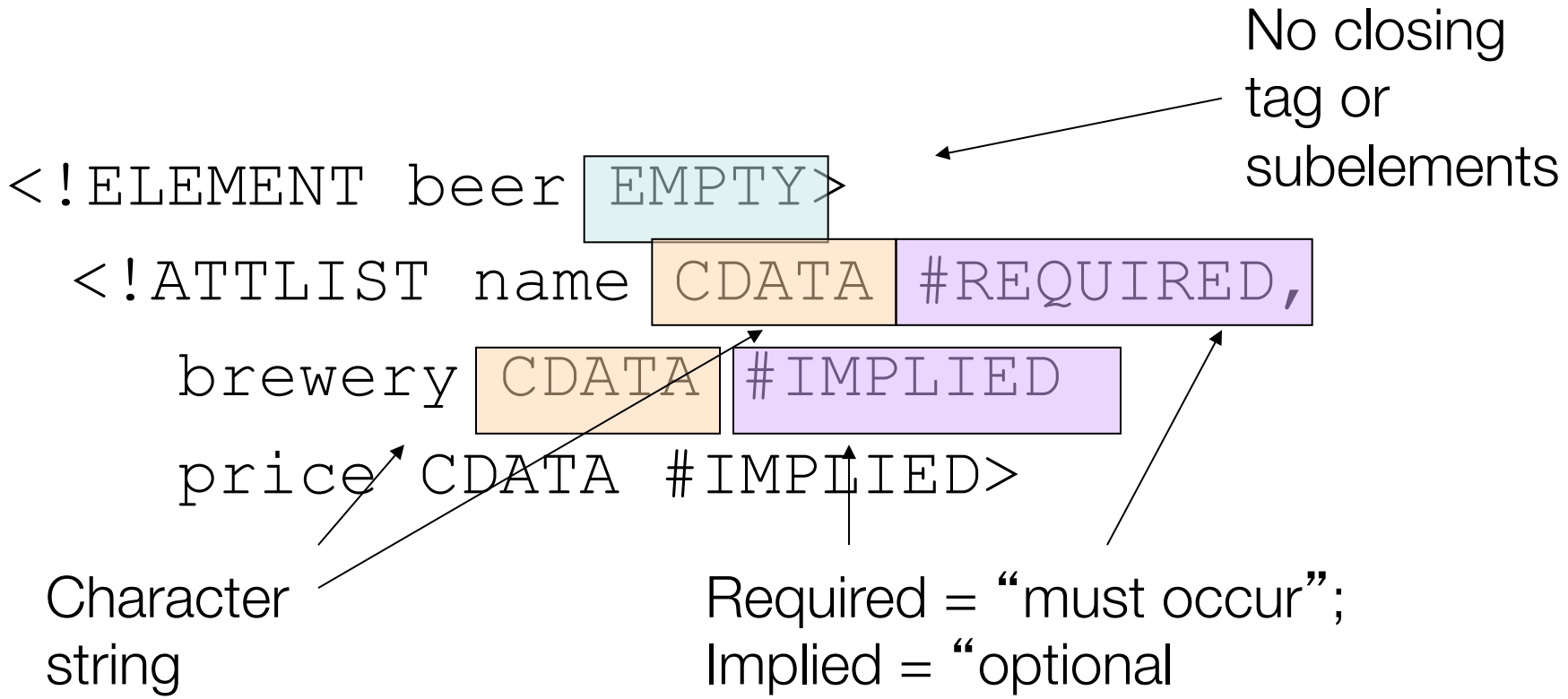
Example: DTD w/o using attributes



XML documents: DTDs & attributes

- As we've seen, opening tags in XML can have attributes.
- This is something that can be captured in a DTD

Example: Attributes



Example use:
<beer name="Amnesiac" />

Summary so far

- This course does not exclusively focusing on SQL
  - But we will see how relational data models is a strong fit for SQL
- Data model
  - "Relational" is one
  - There are others
  - Remember: we use it as an abstraction
- Some SQL for describing data models
  - We've kept it simple, it will get more complex quickly once we get to queries
- Semi-structured data (XML)

Colophon

- Some slide material is from Stanford CS145 (Jeffrey D. Ullman, Fall 2007)