Motivation:

Relational Data Model is quite rigid. … powerful, but rigid.

With the explosive growth of the Internet, electronic information is all around us, and tends not to be warehoused, centrally located, or conforming to a rigid set of relations along with their interrelationships.

Information is more dynamic, and *information interchange* (data exchange between applications) becomes the focus. The goal is to allow an apparent integration of data and/or databases from multiple sources.

When we wish to exchange data between entities, the data forms and formats may not be identical. The medium for information exchange becomes what is termed "semistructured data" -- a new data model designed to cope with problems of information integration.

XML is a standard language for describing semistructured data schemas and representing data.

Some of the problems

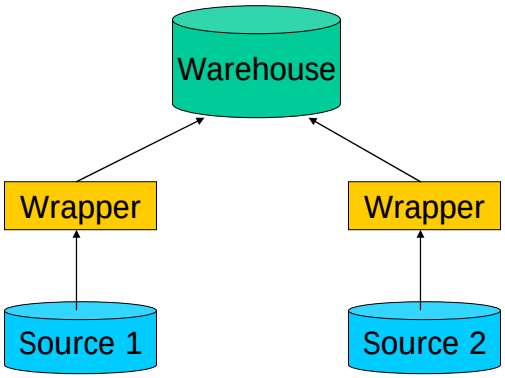
- Related data exists in many places and could, in principle, work together.
- But different databases differ in:
  1. Model (relational, object-oriented?).
  2. Schema (normalized/unnormalized?).
  3. Terminology: are consultants employees? Retirees? Subcontractors?
  4. Conventions (meters versus feet?).

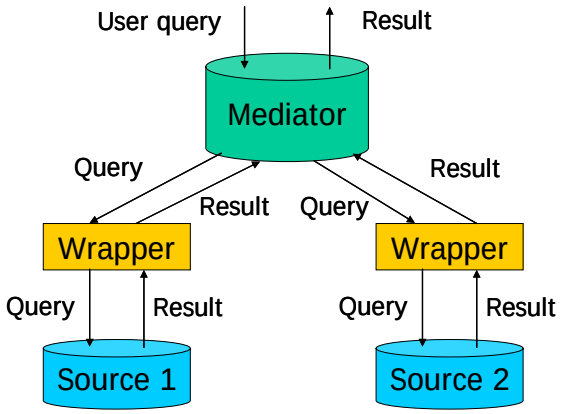
Example:

- Every bar has a database.
  - One may use a relational DBMS; another keeps the menu in an MS-Word document.
  - One stores the phones of distributors, another does not.
  - One distinguishes ales from other beers, another doesn't.
  - One counts beer inventory by bottles, another by cases.

Two Approaches to Integration

1. *Warehousing* : Make copies of the data sources at a central site and transform it to a common schema.
   - Reconstruct data daily/weekly, but do not try to keep it more up-to-date than that.
2. *Mediation* : Create a view of all sources, as if they were integrated.
   - Answer a view query by translating it to terminology of the sources and querying them.



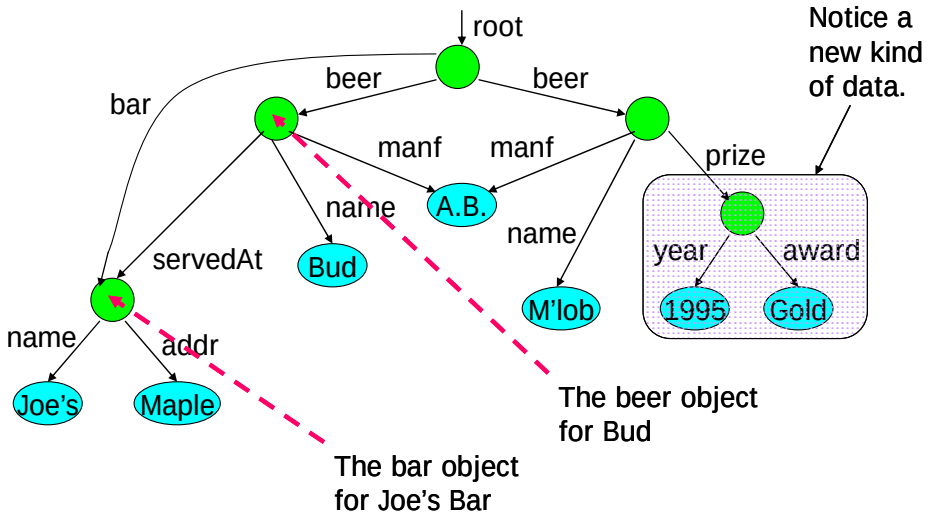


Semistructured Data Model

- Purpose: represent data from independent sources more flexibly than either relational or object-oriented models.
- Think of objects, but with the type of each object its own business, not that of its “class.”
- Labels to indicate meaning of substructures.

Think of Semistructured Data as a Graph

- 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.



XML as a language for specifying semistructured data

- 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 XML* allows you to invent your own tags.
  - Similar to labels in semistructured data.
- *Valid XML* involves a DTD (Document Type Definition), which limits the labels and gives a grammar for their use.

- Start the document with a *declaration*, surrounded by <? ... ?> .
- Normal declaration is:

```
<? XML VERSION = "1.0"
  STANDALONE = "yes" ?>
```

  - "Standalone" = "no DTD provided."
- Balance of document is a *root tag* surrounding nested tags.

- Tags, as in HTML, are normally matched pairs, as <FOO> ... </FOO> .
- Tags may be nested arbitrarily.
- Tags requiring no matching ender, like <P> in HTML, are also permitted.

```
<? XML VERSION = "1.0" STANDALONE = "yes" ?>
<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>
```

- The <BARS> XML document is: