The Relational Data Model

Tables
Schemas
Conversion from E/R to Relations
A Relation is a Table

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winterbrew</td>
<td>Pete’s</td>
</tr>
<tr>
<td>Bud Lite</td>
<td>Anheuser-Busch</td>
</tr>
</tbody>
</table>

Attributes (column headers)

Tuples (rows)
Schemas

◆ Relation schema = relation name + attributes, in order (+ types of attributes).
  ‣ Example: Beers(name, manf)
    or Beers(name: string, manf: string)

◆ Relational Database = collection of relations.

◆ Relational Database schema = set of all relation schemas in the database.
Why Relations?

- Very simple model.
- Often matches how we think about data.
- Abstract model that underlies SQL, the most important database language today.
  - But SQL uses bags, while the relational model is a set-based model.
From E/R Diagrams to Relations

◆ Entity sets become relations with the same set of attributes.

◆ Relationship sets become relations whose attributes are only:
  ▶ The **keys** of connected entity sets.
  ▶ Attributes of the relationship itself.
Entity Set -> Relation

Relation: Beers(name, manf)
Relationship -> Relation

- Drinkers
- Beers
- Buddies
- Married
- Likes
- Favorite

Relations:
- Likes(drinker, beer)
- Favorite(drinker, beer)
- Buddies(name1, name2)
- Married(husband, wife)
Combining Relations

- It is OK to combine the relation for an entity-set $E$ with the relation $R$ for a many-one relationship from $E$ to another entity set $E'$.

Example:
- $E = \text{Drinkers}(\text{name, addr})$
- $R = \text{Favorite}(\text{drinker, beer})$ many-one

- $E' = \text{Drinker1}(\text{name, addr, favBeer})$. 
Risk with Many-Many Relationships

Combining Drinkers with Likes would be a mistake. It leads to redundancy, as:

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sally</td>
<td>123 Maple</td>
<td>Bud</td>
</tr>
<tr>
<td>Sally</td>
<td>123 Maple</td>
<td>Miller</td>
</tr>
</tbody>
</table>

Redundancy
Handling Weak Entity Sets

◆ The *relation* for a weak *entity* set must *include* attributes for its complete key (including those belonging to other entity sets), *as well as its own*, nonkey attributes.

◆ A relation corresponding to supporting (double-diamond) relationship is thus *redundant*, and should not be included in the relational schema.
Example

Teams(teamName)
Players(name, number, teamName) including key attributes
Playson(name, number, teamName) redundant relation
Ullman’s Example

- `Hosts(hostName)`
- `Logins(loginName, hostName, time)`
- `At(loginName, hostName, hostName2)`

`At` becomes part of `Logins` and must be the same.
Entity Sets With Subclasses

Three approaches:

1. **Object-oriented**: each entity belongs to exactly one class; create a relation for each class, with all its attributes.

2. **E/R style**: create one relation for each subclass, with only the key attribute(s) and attributes attached to that E.S.; entity represented in all relations to whose subclass/E.S. it belongs.

3. **Use nulls**: create one relation; entities have null in attributes that don’t belong to them.
Example

Beers

name

manf

isa

Ales

color
### Object-Oriented Style [??]
[knowing that every Ale is a Beer]

Beers

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bud</td>
<td>Anheuser-Busch</td>
</tr>
</tbody>
</table>

Ales

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summerbrew</td>
<td>Pete’s</td>
<td>dark</td>
</tr>
</tbody>
</table>
# E/R Style

## Beers

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bud Summerbrew</td>
<td>Anheuser-Busch</td>
</tr>
<tr>
<td></td>
<td>Pete’s</td>
</tr>
</tbody>
</table>

## Ales

<table>
<thead>
<tr>
<th>name</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summerbrew</td>
<td>dark</td>
</tr>
</tbody>
</table>
Using Nulls

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bud Summerbrew</td>
<td>Anheuser-Busch</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td>Pete’s</td>
<td>dark</td>
</tr>
</tbody>
</table>
Comparisons

◆ O-O approach good for queries like “find the color of ales made by Pete’s.”
  ▶ Just look in Ales relation.

◆ E/R approach good for queries like “find all beers (including ales) made by Pete’s.”
  ▶ Just look in Beers relation.

◆ Using nulls saves space unless there are *lots* of attributes that are usually null.