CS 235:
Introduction to Databases
Svetlozar Nestorov
Lecture Notes #3

Announcements
- Assignment 1:
  - Due next Tuesday, Oct 15.
  - Submit it in class!

The Big Picture
- Stages of building a database application:
  - Real-world domain.
    - understand client needs.
  - Design data model:
    - using entity-relationship (E/R) model
  - Database data model:
    - using relational model
  - Create schema in DBMS, load data.
  - Open for business!

Outline
- Relational model.
- From E/R diagrams to relations.
- Functional dependencies.
- Keys.

Relational Model
- Table = relation.
- Column headers = attributes.
- Row = tuple.
- Beers example:

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honkers Ale</td>
<td>Goose Island</td>
</tr>
<tr>
<td>BudLite</td>
<td>A.B.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Relational Model
- Relation schema:
  - name (attributes)
  - other structure info., e.g., keys, other constraints.
- Example: Beers(name, manf).
- Order of attributes is arbitrary.
  - In practice we need to assume the order given in the relation schema.
- Relation instance is current set of rows for a relation schema.
- Database schema is collection of relation schemas.
Why Relations?

- Very simple model.
- Often a good match for the way we think about our data.
- Abstract model that underlies SQL, the most important language in DBMS's today.
  - But SQL uses bags while the abstract relational model is set-oriented.

Relational Design

- Simplest approach (not always best):
  - convert each E.S. to a relation
  - convert each relationship to a relation.

Entity Set → Relation

- E.S. attributes become relational attributes.
  
  name \[\rightarrow\] Beers \[\rightarrow\] manf

  - Becomes: Beers(name, manf)

Keys in Relations

- An attribute or set of attributes K is a key for a relation R if we expect that in no instance of R will two different tuples agree on all the attributes of K.
- Indicate a key by underlining the key attributes.
- Example: If name is a key for Beers: Beers(name, manf)

E/R Relationships → Relations

- Relation has attribute(s) for key attributes of each E.S. that participates in the relationship.
- Add any attributes that belong to the relationship itself.
- Renaming attributes OK.
  - Essential if multiple roles for an E.S.

Example
Combining Relations

- Common case: Relation for an E.S. $E$ plus the relation for some many-one relationship from $E$ to another E.S.
- Example:
  - Combine Drinkers($\text{name, addr}$) with Favorite($\text{drinker, beer}$).
  - Resulting in: Drinkers2($\text{name,addr, favBeer}$).
- Danger in pushing this idea too far: redundancy.
- Example:
  - Combining Drinkers with Likes causes the drinker's address to be repeated, viz.:

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>111 E Ohio</td>
<td>Guinness</td>
</tr>
<tr>
<td>Mike</td>
<td>111 E Ohio</td>
<td>Newcastle</td>
</tr>
</tbody>
</table>

  The difference: Favorite is many-one; Likes is many-many.

Weak Entity Sets, Relationships → Relations

- Relation for a weak E.S. must include its full key (i.e., attributes of related entity sets) as well as its own attributes.
- A supporting (double-diamond) relationship yields a relation that is actually redundant and should be deleted from the database schema.

Example

```
Hosts(name)
Users(name, hostName)
At(userName, hostName, hostName2)
```

- In $At$, hostName and hostName2 must be the same host, so delete one of them.
- Then, $Users$ and $At$ become the same relation; delete one of them.
- In this case, $Hosts$ schema is a subset of $Users$ schema. Delete $Hosts$?

Subclasses → Relations

- Three approaches:
  - Object-oriented
  - E/R style
  - Using nulls

Object-oriented Style

- Each entity is in one class.
- Create a relation for each class, with all the attributes for that class.
- Don’t forget inherited attributes.
E/R Style

- An entity is in a network of classes related by \( isa \).
- Create one relation for each E.S.
- Relation has only the attributes attached to that E.S. + key.

<table>
<thead>
<tr>
<th>Name</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>BudLite</td>
<td>A.B.</td>
</tr>
<tr>
<td>Honkers Ale</td>
<td>Goose Island</td>
</tr>
</tbody>
</table>

Using NULLs

- Create one relation for the root class or root E.S., with all attributes found anywhere in its network of subclasses.
- Put \( NULL \) in attributes not relevant to a given entity.

<table>
<thead>
<tr>
<th>Name</th>
<th>Manufacturer</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>BudLite</td>
<td>A.B.</td>
<td>NULL</td>
</tr>
<tr>
<td>Honkers Ale</td>
<td>Goose Island</td>
<td>dark</td>
</tr>
</tbody>
</table>

OO-Style

- Create one relation for each E.S.
- Each relation has all attributes found anywhere in that E.S.’s network of subclasses.
- To make things pretty, put NULL in attributes not relevant to a given entity.

<table>
<thead>
<tr>
<th>Name</th>
<th>Manufacturer</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>BudLite</td>
<td>A.B.</td>
<td>NULL</td>
</tr>
<tr>
<td>Honkers Ale</td>
<td>Goose Island</td>
<td>dark</td>
</tr>
</tbody>
</table>

Functional Dependencies

- \( X \rightarrow A \)
  - assertion about a relation \( R \) that whenever two tuples agree on all the attributes of \( X \), then they must also agree on attribute \( A \).
  - Important as a constraint on the data that may appear within a relation.
  - Schema-level control of data.
  - Mathematical tool for explaining the process of “normalization” – vital for redesigning database schemas when original design has certain flaws.

Example

Drinkers(name, addr, beersLiked, manufacturer, favoriteBeer)

- Reasonable FD’s to assert:
  1. \( \text{name} \rightarrow \text{beersLiked} \text{, manufacturer, favoriteBeer} \)
  2. \( \text{addr} \rightarrow \text{favoriteBeer} \text{, manufacturer} \)
  3. \( \text{beersLiked} \rightarrow \text{manufacturer} \)
- Note: FD’s can give more detail than just assertion of a key.

Properties of FD’s

- Key (in general) functionally determines all attributes. In our example:
  \( \text{name} \rightarrow \text{beersLiked} \text{, manufacturer} \text{, favoriteBeer} \)
- Shorthand: combine FD’s with common left side by concatenating their right sides.
- When FD’s are \( not \) of the form Key \( \rightarrow \) other attribute(s), then there is typically an attempt to “cram” too much into one relation.
Properties of FD’s

- Sometimes, several attributes jointly determine another attribute, although neither does by itself.
- Example: 
  \[ \text{beer bar} \rightarrow \text{price} \]

Formal Notion of Key

- \( K \) is a key for relation \( R \) if:
  1. \( K \rightarrow \) all attributes of \( R \).
  2. For no proper subset of \( K \) is (1) true.
- If \( K \) at least satisfies (1), then \( K \) is a superkey.

FD Conventions

- \( X \), etc., represent sets of attributes; \( A \) etc., represent single attributes.
- No set formers in FD’s, e.g., \( ABC \) instead of \( \{A, B, C\} \).

Example

Drinkers(name, addr, beersLiked, manf, favoriteBeer)

- \( \{\text{name, beersLiked}\} \) FD’s all attributes, as seen.
  - Shows \( \{\text{name, beersLiked}\} \) is a superkey.
- \( \text{name} \rightarrow \text{beersLiked} \) is false, so \( \text{name} \) not a superkey.
- \( \text{beersLiked} \rightarrow \text{name} \) also false, so \( \text{beersLiked} \) not a superkey.
- Thus, \( \{\text{name, beersLiked}\} \) is a key.
- No other keys in this example.
- Neither \( \text{name} \) nor \( \text{beersLiked} \) is on the right of any observed FD, so they must be part of any superkey.

Who Determines Keys/FD’s?

- We could define a relation schema by simply giving a single key \( K \).
  - Then the only FD’s asserted are that \( K \rightarrow A \) for every attribute \( A \).
  - No surprise: \( K \) is then the only key for those FD’s, according to the formal definition of "key."
- Or, we could assert some FD’s and deduce one or more keys by the formal definition.
  - E/R diagram implies FD’s by key declarations and many-one relationship declarations.
- Rule of thumb: FD’s either come from keyness, many-1 relationship, or from physics.
  - E.g., "no two courses can meet in the same room at the same time" yields \( \text{room time} \rightarrow \text{course} \).