Normalization 2

Todd Bacastow
IST 210
Normalization

- Methods
  - Inspection
  - Closure
- Functional dependencies are key
First Normal Form (1NF)

- All appropriate data into the empty columns of rows containing the repeating data (‘flattening’ the table), i.e., no repeating items in columns
Second Normal Form (2NF)

- In 1NF and every non-key column is fully dependent on the (entire) primary key
Third Normal Form (3NF)

- In 2NF, every non-key column is mutually independent, i.e., no transitive dependencies.
Functional Dependencies

- Describes the relationship between attributes in a relation

Example
- If A and B are attributes of relation R, A functionally determines B (A \( \rightarrow \) B), if each value of A is associated with exactly one value of B.
Functional Dependencies

- Functional dependencies arise from the nature of the real world that the database models.
- Often $A$ and $B$ are facts about an entity where $A$ might be some identifier for the entity and $B$ some characteristic.
- Functional dependencies cannot be automatically determined by studying one or more instances of a database.
- They can be determined only by a careful study of the real world and a clear understanding of what each attribute means.
Functional Dependencies

- Important as a constraint on the data that may appear within a relation
  - Scheme-level control of data
- Mathematical tool for explaining the process of “normalization” ---- vital for DB redesigning
**Example**

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beerLiked</th>
<th>manf</th>
<th>favoriteBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jane</td>
<td>16801</td>
<td>Bud</td>
<td>A.B.</td>
<td>WickedAle</td>
</tr>
<tr>
<td>Jane</td>
<td>16801</td>
<td>WickedAle</td>
<td>Pete’s</td>
<td>WickedAle</td>
</tr>
<tr>
<td>Mike</td>
<td>16803</td>
<td>Bud</td>
<td>A.B.</td>
<td>Bud</td>
</tr>
</tbody>
</table>

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

- Reasonable FD’s to assert:
  - name $\rightarrow$ addr
  - name $\rightarrow$ favoriteBeer
  - beersLiked $\rightarrow$ manf

- Note: these happen to imply the underlined key, but the FD’s give more detail than the mere assertion of a key
Example (cont.)

- Key (in general) functionally determines all attributes.
  - name beersLiked → addr favoriteBeer beerManf

- Shorthand: combine FD’s with common left side by concatenating their right sides

- When FD’s are not of the form (Key) → other attribute(s), then there is typically an attempt to “cram” too much into one relation.

- Sometimes, several attributes jointly determines another attribute, although neither does by itself.
  - Example: beer bar → price
Formal Notion of Key

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

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

- {name,beerLiked} FD’s all attributes, as seen
  - Shows {name,beerLiked} is a superkey (functionally determines all attributes)
- name $\rightarrow$ beerLiked is false, so name not a superkey
- beersLiked $\rightarrow$ name also false, so beersLiked not a superkey
- Thus, {name,beersLiked} is a key
- No other keys in this example
  - Neither name nor 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-to-one relationship declarations
Rule of thumb

- FD’s either come from keyness (i.e., a set of attributes that functionally determines all attributes), many-to-one relationship, or from physics
  - E.g., “no two courses can meet in the same room at the same time” (at least we don’t know a way) yields
  room time $\rightarrow$ course
Boyce-Codd Normal Form

- Goal = BCNF = Boyce-Codd Normal Form = All FD’s follow from the fact key → everything
- Formally, R is in BCNF if every nontrivial FD for R, say X → A, has X a superkey (i.e., X functionally determines all attributes).
- Why important?
  - Guarantees no redundancy due to FD’s
  - Guarantees no update anomalies
  - Guarantees no deletion anomalies
Using Closure to Reach BCNF

- Setting: relation R, given FD’s F. Suppose relation R has BCNF violation $X \rightarrow B$.
- We need only look among FD’s of F for a BCNF violation.
  - Compute $X^+$ (cannot be all attributes)
  - Decompose R into $X^+$ and $(R-X^+) \cup X$.
  - Find the FD’s for the decomposed relations
Example

Drinkers \((\text{name, addr, beersLiked, manf, favoriteBeer})\)

- FD’s:
  - name \(\rightarrow\) addr
  - name \(\rightarrow\) favoriteBeer
  - beersLiked \(\rightarrow\) manf

- Pick BCNF violation name \(\rightarrow\) addr

  - Close the left side: \(\text{name}^+ = \text{name addr favoriteBeer}\)
  - Decomposed relations:
    - Drinkers1(\text{name, addr, favoriteBeer})
    - Drinkers2(\text{name, beersLiked, manf})

- New FD’s:
  - For Drinker1: name \(\rightarrow\) addr and name \(\rightarrow\) favoriteBeer (BCNF)
  - For Drinker2: beersLiked \(\rightarrow\) manf (violates BCNF, decompose again)
Example (cont.)

- Final relations all in BCNF
  Drinker1(name, addr, favoriteBeer)
  Drinker3(beersLiked, manf)
  Drinker4(name, beersLiked)