Announcement

- Project proposal/progress report due today
- Midterm next Thursday in class
  - Everything up to today’s lecture, with a focus on the materials covered by the first two homework assignments
  - Open book, open notes
- Will assign an optional problem set tonight as a study guide for midterm
  - Entirely optional
- If you turn it in on Tuesday in class, you can use its grade to replace your lowest homework grade so far
- Solution will be posted on Tuesday midnight
- Graded Homework #2 will be available on Tuesday

Review

- Functional dependencies
  - $X \rightarrow Y$: If two rows agree on $X$, they must agree on $Y$
  - A generalization of the key concept
- Non-key functional dependencies: a source of redundancy
  - Non-trivial $X \rightarrow Y$ where $X$ is not a superkey
    - Called a BCNF violation
- BCNF decomposition: a method for removing redundancies
  - Given $R(X, Y, Z)$ and a BCNF violation $X \rightarrow Y$, decompose $R$ into $R_1(X, Y)$ and $R_2(X, Z)$
    - A lossless join decomposition
  - Schema in BCNF has no redundancy due to FD’s

Next

- 3NF (BCNF is too much)
  - Multivalued dependencies: another source of redundancy
- 4NF (BCNF is not enough)

Motivation for 3NF

- $Address\ (street\_address, city, state, zip)$
  - $street\_address, city, state \rightarrow zip$
  - $zip \rightarrow city, state$
- Keys
  - $\{street\_address, city, state\}$
  - $\{street\_address, zip\}$
- BCNF?
  - Violation: $zip \rightarrow city, state$

To decompose or not to decompose

$Address_1\ (zip, city, state)$
$Address_2\ (street\_address, zip)$

- FD’s in $Address_1$
  - $zip \rightarrow city, state$
- FD’s in $Address_2$
  - None!
- Hey, where is $street\_address, city, state \rightarrow zip$?
  - Cannot check without joining $Address_1$ and $Address_2$ back together
- Problem: Some lossless join decomposition is not dependency-preserving
- Dilemma: Should we get rid of redundancy at the expense of making constraints harder to enforce?
3NF
- R is in Third Normal Form (3NF) if for every non-trivial FD X → A (where A is single attribute), either
  - X is a superkey of R, or
  - A is a member of at least one key of R
- Intuitively, BCNF decomposition on X → A would "break" the key containing A
- So Address is already in 3NF
- Tradeoff:
  - Can enforce all original FD’s on individual decomposed relations
  - Might have some redundancy due to FD’s

BNCF = no redundancy?
- Student (SID, CID, club)
  - Suppose your classes have nothing to do with the clubs you join
  - FD’s?
    - None
  - BNCF?
    - Yes
  - Redundancies?
    - Tons!

Multivalued dependencies
- A multivalued dependency (MVD) has the form X ! Y, where X and Y are sets of attributes in a relation R
- X → Y means that whenever two rows in R agree on all the attributes of X, then we can swap their Y components and get two new rows that are also in R
  - Must be in R too

MVD examples
- Student (SID, CID, club)
  - SID → CID
  - SID → club
    - Intuition: given SID, CID and club are “independent”
  - SID, CID → club
    - Trivial: LHS ∪ RHS = all attributes of R
  - SID, CID → SID
    - Trivial: LHS ⊇ RHS

Complete MVD + FD rules
- FD reflexivity, augmentation, and transitivity
- MVD complementation:
  - If X → Y, then X → attr(R) – X – Y
- MVD augmentation:
  - If X → Y and V ⊆ W, then XW → YV
- MVD transitivity:
  - If X → Y and Y → Z, then X → Z – Y
- Replication (FD is MVD):
  - If X → Y, then X → Y
- Coalescence:
  - If X → Y and Z ⊆ Y and there is some W disjoint from Y such that W → Z, then X → Z

An elegant solution: chase
- Given a set of FD’s and MVD’s D, does another dependency d (FD or MVD) follow from D?
- Procedure
  - Start with the hypotheses of d, and treat them as “seed” tuples in a relation
  - Apply the given dependencies in D repeatedly
    - If we apply an FD, we infer equality of two symbols
    - If we apply an MVD, we infer more tuples
  - If we infer the conclusion of d, we have a proof
  - Otherwise, if nothing more can be inferred, we have a counterexample
Proof by chase

\[ \text{In } R(A, B, C, D), \text{ does } A \rightarrow B \text{ and } B \rightarrow C \text{ imply that } A \rightarrow C? \]

<table>
<thead>
<tr>
<th>Have</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \rightarrow B )</td>
<td>( b )</td>
</tr>
<tr>
<td>( B \rightarrow C )</td>
<td>( c )</td>
</tr>
</tbody>
</table>

Counterexample by chase

\[ \text{In } R(A, B, C, D), \text{ does } A \rightarrow BC \text{ and } CD \rightarrow B \text{ imply that } A \rightarrow B? \]

<table>
<thead>
<tr>
<th>Have</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \rightarrow BC )</td>
<td>( b_1 = b_2 )</td>
</tr>
</tbody>
</table>

4NF decomposition example

<table>
<thead>
<tr>
<th>Student (SID, CID, club)</th>
<th>Enroll (SID, CID)</th>
<th>Join (SID, club)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID CID club</td>
<td>SID CID</td>
<td>SID club</td>
</tr>
<tr>
<td>142 CPS196 ballet</td>
<td>142</td>
<td>142</td>
</tr>
<tr>
<td>142 CPS196 sumo</td>
<td>142</td>
<td>142</td>
</tr>
<tr>
<td>123 CPS196 chess</td>
<td>123</td>
<td>123</td>
</tr>
</tbody>
</table>

4NF decomposition algorithm

\[ \text{Find a 4NF violation} \]
- A non-trivial MVD \( X \rightarrow Y \) in \( R \) where \( X \) is not a superkey
- Decompose \( R \) into \( R_1 \) and \( R_2 \), where
  - \( R_1 \) has attributes \( X \cup Y \)
  - \( R_2 \) has attributes \( X \cup Z \) (\( Z \) contains attributes not in \( X \) or \( Y \))
- Repeat until all relations are in 4NF

\[ \text{Almost identical to BCNF decomposition algorithm} \]
\[ \text{Any decomposition on a 4NF violation is lossless} \]
### 3NF, BCNF, 4NF, and beyond

<table>
<thead>
<tr>
<th>Anomaly/normal form</th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lose FD's?</td>
<td>No</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Redundancy due to FD's</td>
<td>Possible</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Redundancy due to MVD's</td>
<td>Possible</td>
<td>Possible</td>
<td>No</td>
</tr>
</tbody>
</table>

- **Of historical interests**
  - 1NF: All column values must be atomic
  - 2NF: There is no partial functional dependency (a non-trivial FD X → A where X is a proper subset of some key)

### Summary

- Philosophy behind BCNF, 4NF:
  Data should depend on the key, the whole key, and nothing but the key!

- Philosophy behind 3NF:
  … But not at the expense of more expensive constraint enforcement.