Relational Database Design Theory
Part II

CPS 116
Introduction to Database Systems

Announcements (October 13)

- Midterm graded; sample solution available
  - Please verify your grades on Blackboard
- Project milestone #1 due today

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 → zip
  - zip → city, state
- Keys
  - {street_address, city, state}
  - {street_address, zip}
- BCNF?

To decompose or not to decompose

- Address₁ (zip, city, state)
- Address₂ (street_address, zip)
- FD’s in Address₁
  - zip → city, state
- FD’s in Address₂
  - None!
- Hey, where is street_address, city, state → zip?
  - Cannot check without joining Address₁ and Address₂ 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 \( X \rightarrow A \) (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 \( X \rightarrow A \) would "break" the key containing \( A \)

- So \textit{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?

- \textit{Student} (\textit{SID}, \textit{CID}, \textit{club})
  - Suppose your classes have nothing to do with the clubs you join
  - FD's?
  - BNCF?
  - Redundancies?

Multivalued dependencies

- A multivalued dependency (MVD) has the form \( X \rightarrow Y \), where \( X \) and \( Y \) are sets of attributes in a relation \( R \)
- \( X \rightarrow Y \) 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 \( R \)
MVD examples

Student (SID, CID, club)

- SID → CID

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 hypothesis 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

In \(R(A, B, C, D)\), does \(A \rightarrow B\) and \(B \rightarrow C\) imply that \(A \rightarrow C\)?

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

Another proof by chase

In \(R(A, B, C, D)\), does \(A \rightarrow B\) and \(B \rightarrow C\) imply that \(A \rightarrow C\)?

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

In general, both new tuples and new equalities may be generated.

Counterexample by chase

In \(R(A, B, C, D)\), does \(A \rightarrow BC\) and \(CD \rightarrow B\) imply that \(A \rightarrow B\)?

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

Counterexample!
4NF

A relation \( R \) is in Fourth Normal Form (4NF) if
- For every non-trivial MVD \( X \rightarrow Y \) in \( R \), \( X \) is a superkey
- That is, all FD's and MVD's follow from "key \( \rightarrow \) other attributes" (i.e., no MVD's, and no FD's besides key functional dependencies)

4NF is stronger than BCNF
- Because every FD is also a MVD

4NF decomposition algorithm

- 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
- Almost identical to BCNF decomposition algorithm
- Any decomposition on a 4NF violation is lossless

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>142 CPS116 ballet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>142 CPS116 sumo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>142 CPS114 ballet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123 CPS116 golf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4NF violation: \( SID \rightarrow CID \)
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: Slightly more relaxed than 3NF

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!