Normalization

Anomalies
Boyce-Codd Normal Form
3rd Normal Form

Design Goals

Goal of relational schema design is to avoid:
- redundancy
- anomalies.
Redundancy

◆ The same information can be extracted in multiple ways. Consequently:
  ▶ Deleting the information must be sure to delete all representations of the information.
  ▶ Updating the information must be kept consistent in all of the various ways.

Anomalies

◆ Update anomaly: one occurrence of a fact is changed, but not all occurrences.

◆ Deletion anomaly: valid fact is lost when a tuple is deleted.
Example of Bad Design

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

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beersLiked</th>
<th>manf</th>
<th>favBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>Bud</td>
<td>A.B.</td>
<td>WickedAle</td>
</tr>
<tr>
<td>Janeway</td>
<td>???</td>
<td>WickedAle</td>
<td>Pete’s</td>
<td>???</td>
</tr>
<tr>
<td>Spock</td>
<td>Enterprise</td>
<td>Bud</td>
<td>A.B.</td>
<td>Bud</td>
</tr>
</tbody>
</table>

Data is **redundant**, because each of the ???’s can be figured out by using given FD’s:
- name -> addr  favBeer
- beersLiked -> manf

This Bad Design Also Exhibits Anomalies

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beersLiked</th>
<th>manf</th>
<th>favBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>Bud</td>
<td>A.B.</td>
<td>WickedAle</td>
</tr>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>WickedAle</td>
<td>Pete’s</td>
<td>???</td>
</tr>
<tr>
<td>Spock</td>
<td>Enterprise</td>
<td>Bud</td>
<td>A.B.</td>
<td>Bud</td>
</tr>
</tbody>
</table>

- **Update** anomaly: if Janeway is transferred to Intrepid, each of her tuples need to be changed?

- **Deletion** anomaly: If nobody likes Bud, we lose track of the fact that Anheuser-Busch manufactures Bud.
Boyce-Codd Normal Form

We say a relation \( R \) is in \( BCNF \) if whenever \( X \rightarrow A \) is a nontrivial FD that holds in \( R \), \( X \) is a superkey.

- Remember: nontrivial means \( A \) is not a member of set \( X \).
- Remember, a superkey is any superset of a key (not necessarily a proper superset).

Explaining how bad things happen with non-BCNF (1)

Assume a relation \( ABCD \)

\[ \begin{array}{c}
A \\
B \\
C \\
D \\
\end{array} \]

\( C \) doesn’t depend on \( AB \): update anomaly. \( A \) is not a superkey, but \( A \rightarrow C \) is non-trivial.
Explaining how bad things happen with non-BCNF (2)

Assume a relation ABC

\[ \begin{array}{ccc}
A & \rightarrow & B \\
& \rightarrow & C \\
\end{array} \]

B is not a superkey, but B \( \rightarrow \) C is non-trivial. A tuple ABC is required to represent just BC information.

Example

- Drinkers(name, addr, beersLiked, manf, favBeer)
- FD’s: name->addr favBeer, beersLiked->manf
- Only key is \{name, beersLiked\}.
- In each FD, the left side is *not* a superkey.
- Any one of these FD’s shows Drinkers is *not* in BCNF
Another Example

- Beers(name, manf, manfAddr)
- FD’s: name->manf, manf->manfAddr
- Only key is {name}.
- name->manf does not violate BCNF, but manf->manfAddr does.

Decomposition into BCNF

- Given: relation R with FD’s F.
- Look among the given FD’s for a BCNF violation X -> B.
  - If any FD following from F violates BCNF, then there will surely be an FD in F itself that violates BCNF.
- Compute X⁺.
  - Not all attributes, or else X is a superkey.
Decompose $R$ Using $X \rightarrow B$

- Replace $R$ by relations with schemas:
  1. $R_1 = X^+$.
  2. $R_2 = (R - X^+) \cup X$.

- Project given FD’s $F$ onto **two new relations**: 
  1. Compute the closure of $F = \text{all nontrivial FD’s that follow from } F$.
  2. Use only those FD’s whose attributes are all in $R_1$ or all in $R_2$.

Decomposition Picture

![Decomposition Diagram]
Example

- Drinkers(name, addr, beersLiked, manf, favBeer)
- $F = \text{name} \rightarrow \text{addr}, \text{name} \rightarrow \text{favBeer}, \text{beersLiked} \rightarrow \text{manf}$
- Pick BCNF violation $\text{name} \rightarrow \text{addr}$.
- Close the left side: $\{\text{name}\}^+ = \{\text{name, addr, favBeer}\}$.
- Decomposed relations:
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers2(name, beersLiked, manf)

Example, Continued

- We are not done; we need to check Drinkers1 and Drinkers2 for BCNF.
- Projecting FD’s is complex in general, easy here.
- For Drinkers1(name, addr, favBeer), relevant FD’s are $\text{name} \rightarrow \text{addr}$ and $\text{name} \rightarrow \text{favBeer}$.
  - Thus, $\text{name}$ is the only key and Drinkers1 is in BCNF.
Example, Continued

- For Drinkers2(name, beersLiked, manf), the only FD is beersLiked->manf, and the only key is {name, beersLiked}.
  - Violation of BCNF.
- beersLiked+ = {beersLiked, manf}, so we decompose Drinkers2 into:
  1. Drinkers3(beersLiked, manf)
  2. Drinkers4(name, beersLiked)

Example, Concluded

- The resulting decomposition of Drinkers:
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers3(beersLiked, manf)
  3. Drinkers4(name, beersLiked)
  - Notice: Drinkers1 tells us about drinkers, Drinkers3 tells us about beers, and Drinkers4 tells us the relationship between drinkers and the beers they like.
Third Normal Form - Motivation

- There is one structure of FD’s that causes trouble when we decompose.
- $AB \rightarrow C$ and $C \rightarrow B$.
  - Example: $A =$ street address, $B =$ city, $C =$ zip code.
- There are two keys, $\{A,B\}$ and $\{A,C\}$.
- $C \rightarrow B$ is a BCNF violation, so we must decompose into $AC$, $BC$.

“Unenforceable” FD’s

- The problem is that if we use $AC$ and $BC$ as our database schema, we cannot enforce the FD $AB \rightarrow C$ by checking FD’s in these decomposed relations.
- Example with $A =$ street, $B =$ city, and $C =$ zip on the next slide.
An Unenforceable FD

<table>
<thead>
<tr>
<th>street</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>545 Tech Sq.</td>
<td>02138</td>
</tr>
<tr>
<td>545 Tech Sq.</td>
<td>02139</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>city</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cambridge</td>
<td>02138</td>
</tr>
<tr>
<td>Cambridge</td>
<td>02139</td>
</tr>
</tbody>
</table>

Join tuples with equal zip codes.

<table>
<thead>
<tr>
<th>street</th>
<th>city</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>545 Tech Sq.</td>
<td>Cambridge</td>
<td>02138</td>
</tr>
<tr>
<td>545 Tech Sq.</td>
<td>Cambridge</td>
<td>02139</td>
</tr>
</tbody>
</table>

Although no FD’s were violated in the decomposed relations, FD street city -> zip is violated by the database as a whole.

3NF Lets Us Avoid This Problem

- 3rd Normal Form (3NF) modifies the BCNF condition so we do not have to decompose in this problem situation.
- An attribute is prime if it is a member of some key.
- X -> A violates 3NF if and only if X is not a superkey, and also A is not prime.
Example

- In our problem situation with FD’s $AB \rightarrow C$ and $C \rightarrow B$, we have keys $AB$ and $AC$.

- Thus $A$, $B$, and $C$ are each prime.

- Although $C \rightarrow B$ violates BCNF, it does not violate 3NF.

What 3NF and BCNF Give You

- There are two important properties of a decomposition:
  1. **Recovery**: it should be possible to project the original relations onto the decomposed schema, and then reconstruct the original.
  2. **Dependency preservation**: it should be possible to check in the projected relations whether all the given FD’s are satisfied.
3NF and BCNF, Continued

- We can get (1) with a BCNF decomposition.
  - Explanation needs to wait for relational algebra.
- We can get both (1) and (2) with a 3NF decomposition.
- But we can’t always get (1) and (2) with a BCNF decomposition.
  - street-city-zip is an example.