Schema Refinement & Normalization Theory 5

Week 13 - 1
Third Normal Form: Motivation

• There are some situations where
  – BCNF is not dependency preserving, and
  – efficient checking for FD violation on updates is important

• Solution: define a weaker normal form, called Third Normal Form (3NF)
  – Allows some redundancy (with resultant problems; we will see examples later)
  – But functional dependencies can be checked on individual relations without computing a join.
  – There is always a lossless-join, dependency-preserving decomposition into 3NF.
Third Normal Form (3NF)

- If R is in BCNF, obviously in 3NF.
- If R is in 3NF, some redundancy is possible. It is a compromise, used when BCNF not achievable (e.g., no “good” decomposition, or performance considerations).
  - Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible.
3NF

• Relation R with FDs $F$ is in 3NF if, for each FD $X \rightarrow A$ ($X \in R$ and $A \in R$) in $F$, one of the following statements is true:
  
  – $A \in X$ (trivial FD), or
  – $X$ is a superkey, or
  – $A$ is part of some key for R

\[\text{If one of these two is satisfied for ALL FDs, then}\]
\[\text{R is in BCNF}\]

\[\text{Not just superkey! (why not?)}\]
What Does 3NF Achieve?

• If 3NF is violated by $X \rightarrow A$, one of the following holds:
  – $X$ is a subset of some key $K$ (partial redundancy)
    • We store $(X, A)$ pairs redundantly.
  – $X$ is not a proper subset of any key.
    • There is a chain of FDs $K \rightarrow X \rightarrow A$, which means that we cannot associate an $X$ value with a $K$ value unless we also associate an $A$ value with an $X$ value.

• But: even if reln is in 3NF, these problems could arise.
  – e.g., Reserves SBDC (sid, bid, date, credit_card). Keys are SBD, CBD. FD = \{S → C, C → S\}. R is in 3NF, but for each reservation of sailor $S$, same (S, C) pair is stored.

• Thus, 3NF is indeed a compromise relative to BCNF.
Decomposition into 3NF

• Obviously, the algorithm for lossless join decomp into BCNF can be used to obtain a lossless join decomp into 3NF (typically, can stop earlier).

• To ensure dependency preservation, one idea:
  – If $X \rightarrow Y$ is not preserved, add relation $XY$.
  – Problem is that $XY$ may violate 3NF!

• Refinement: Instead of the given set of FDs $F$, use a *minimal cover for $F$*. 
Minimal Cover for a Set of FDs

• **Minimal cover** $G$ for a set of FDs $F$:
  - Closure of $F = \text{closure of } G$.
  - Right hand side of each FD in $G$ is a single attribute.
  - If we modify $G$ by deleting an FD or by deleting attributes from an FD in $G$, the closure changes.

• Intuitively, every FD in $G$ is needed, and “as small as possible” in order to get the same closure as $F$. 
Obtaining Minimal Cover

• Step 1: Put the FDs in a standard form (i.e. right-hand side should contain only single attribute)
• Step 2: Minimize the left side of each FD
• Step 3: Delete redundant FDs
• Find minimal cover for $F = \{ABH \rightarrow CK, A \rightarrow D, C \rightarrow E, BGH \rightarrow L, L \rightarrow AD, E \rightarrow L, BH \rightarrow E\}$
• Step 1: Make RHS of each FD into a single attribute:

\[ F = \{ABH \rightarrow C, ABH \rightarrow K, A \rightarrow D, C \rightarrow E, BGH \rightarrow L, L \rightarrow A, L \rightarrow D, E \rightarrow L, BH \rightarrow E\} \]
\[ F = \{ABH \rightarrow C, ABH \rightarrow K, A \rightarrow D, C \rightarrow E, BGH \rightarrow L, L \rightarrow A, L \rightarrow D, E \rightarrow L, BH \rightarrow E\} \]

- **Step 2: Eliminate redundant attributes from LHS, e.g. Can an attribute be deleted from \( ABH \rightarrow C \)?**
  - Compute \((AB)^+, (BH)^+, (AH)^+\) and see if any of them contains \( C \). (Why?)
  - \((AB)^+ = ABD, (BH)^+ = ABCDEHKL, (AH)^+ = ADH\). Since \( C \in (BH)^+\), \( BH \rightarrow C \) is entailed by \( F \). So \( A \) is redundant in \( ABH \rightarrow C \). Similarly, \( A \) is also redundant in \( ABH \rightarrow K \). Check further to see if \( B \) or \( H \) is redundant as well.

  - Similarly, for \( BGH \rightarrow L \), \( G \) is redundant since \( L \in (BH)^+\).

  - \( F = \{BH \rightarrow C, BH \rightarrow K, A \rightarrow D, C \rightarrow E, BH \rightarrow L, L \rightarrow A, L \rightarrow D, E \rightarrow L, BH \rightarrow E\} \)
• \( F = \{ \text{BH} \rightarrow \text{C}, \text{BH} \rightarrow \text{K}, \text{A} \rightarrow \text{D}, \text{C} \rightarrow \text{E}, \text{BH} \rightarrow \text{L}, \text{L} \rightarrow \text{A}, \text{L} \rightarrow \text{D}, \text{E} \rightarrow \text{L}, \text{BH} \rightarrow \text{E} \} \)

• Step 3: Delete redundant FDs from \( F \).
  - If \( F - \{ f \} \) infers \( f \), then \( f \) is redundant, i.e. if \( f \) is \( X \rightarrow \text{A} \), then check if \( X^+ \) using \( F - f \) still contains \( \text{A} \). If it does, then it means \( X \rightarrow \text{A} \) can be inferred by other FDs.
  - E.g. For \( \text{BH} \rightarrow \text{L} \), \( (\text{BH})^+ \) (not using \( \text{BH} \rightarrow \text{L} \)) = \text{ACDEKL}, which contains \( \text{L} \). This means \( \text{BH} \rightarrow \text{L} \) can be inferred by other FDs, so it’s a redundant FD.
  - In fact, \( \text{BH} \rightarrow \text{L} \) can be inferred by \( \text{BH} \rightarrow \text{E}, \text{E} \rightarrow \text{L} \).
  - Check other FDs using the same algorithm.

• Note: the order of Step 2 and Step 3 should not be exchanged.
What to do with Minimal Cover?

• After obtaining the minimal cover, for each FD $X \rightarrow A$ in the minimal cover that is not preserved, create a table consisting of $XA$ (so we can check dependency in this new table, i.e. dependency is preserved).

• Why does this new table is guaranteed to be in 3NF (whereas if we created the new table from F, it might not?)
  – Since $X \rightarrow A$ is in the minimal cover, $Y \rightarrow A$ does not hold for any $Y$ that is a strict subset of $X$.
    • So $X$ is a key for $XA$ (satisfies condition #2)
    • If any other dependencies hold over $XA$, the right side can involve only attributes in $X$ because $A$ is a single attribute (satisfies condition #3).
Comparison of BCNF and 3NF

• It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  – the decomposition is lossless
  – the dependencies are preserved

• It is always possible to decompose a relation into a set of relations that are in BCNF such that:
  – the decomposition is lossless
  – it may not be possible to preserve dependencies.
Normalization Review

- Identify all FD’s in $F^+$
- Identify candidate keys
- Identify (strongest, or specific) normal forms
  - BCNF, 3NF
- Schema decomposition
  - When to decompose
  - How to check if a decomposition is lossless-join and/or dependency preserving
    - Use projection of $F^+$ to check for dependency preservation
  - Decompose into:
    - Lossless-join
    - Dependency preserving
      - Use minimal cover
Normalization Theory - Practice Questions
### Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FDs with A as the left side:</th>
<th>Satisfied by the relation?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A→A</td>
<td>Yes (trivial FD)</td>
</tr>
<tr>
<td>A→B</td>
<td>Yes</td>
</tr>
<tr>
<td>A→C</td>
<td>No: tuples 1&amp;2</td>
</tr>
<tr>
<td>AB →A</td>
<td>Yes (trivial FD)</td>
</tr>
<tr>
<td>AC →B</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example

Let $F=\{ A \rightarrow BC, B \rightarrow C \}$. Is $C \rightarrow AB$ in $F^+$?

Answer: No. Either of the following 2 reasons is ok:

Reason 1) $C^+=C$, and does not include $AB$.

Reason 2) We can find a relation instance such that it satisfies $F$ but does not satisfy $C \rightarrow AB$.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
List all the non-trivial FDs in $F^+$

- Given $F = \{ A \rightarrow B, B \rightarrow C \}$. Compute $F^+$ (with attributes A, B, C).

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>AC</th>
<th>BC</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>AC</td>
<td>√</td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attribute closure:

- $A^+ = ABC$
- $B^+ = BC$
- $C^+ = C$
- $AB^+ = ABC$
- $AC^+ = ABC$
- $BC^+ = BC$
- $ABC^+ = ABC$
Example

- Given $F=\{ A \rightarrow B, B \rightarrow C \}$. Find a relation that satisfies $F$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

- Given $F=\{ A \rightarrow B, B \rightarrow C \}$. Find a relation that satisfies $F$ but does not satisfy $B \rightarrow A$. Well, the above example suffices.

- Can you find an instance that satisfies $F$ but not $A \rightarrow C$? No. Because $A \rightarrow C$ is in $F^+$
Examples

R(A, B, C, D, E),
F = {A → B, C → D}

Candidate key: ACE. How do we know?

Intuitively,
- B cannot be in a candidate key.
- A is not determined by any other attributes (like E), and A has to be in a candidate key (because a candidate key has to determine all the attributes).
- Now if A is in a candidate key, B cannot be in the same candidate key, since we can drop B from the candidate without losing the property of being a “key”.
- Same reasoning apply to others attributes.
Example

$R(A, B, C, D, E)$,
$F = \{A \rightarrow B, C \rightarrow D\}$ [Same as previous]

Which normal form?

Not in BCNF. This is the case where all attributes in the FDs appear in $R$. We consider $A$, and $C$ to see if either is a superkey of not. Obviously, neither $A$ nor $C$ is a superkey, and hence $R$ is not in BCNF. More precisely, we have $A \rightarrow B$ is in $F^+$ and non-trivial, but $A$ is not a superkey of $R$. 
Example

R(A, B, C, D, E)
F = \{A \rightarrow B, C \rightarrow D\} [Same as previous]

Which normal form?

We already know that it’s not in BCNF. Not in 3NF either. We have A \rightarrow B is in F^+ and non-trivial, but A is not a superkey of R. Furthermore, B is not in any candidate key (since the only candidate key is ACE).
Example

• $R(A,B,F)$, $F = \{AC \to E, B \to F\}$.
• Candidate key? $AB$
• BCNF? No, because of $B \to F$ ($B$ is not a superkey).
• 3NF? No, because of $B \to F$ ($F$ is not part of a candidate key).
Example

- $R(D, C, H, G)$, $F = \{ A \rightarrow I, I \rightarrow A \}$
- Candidate key? DCHG
- BCNF? Yes
- 3NF? Yes
Example

- $R(A, B, C, D, E, G, H)$
  $F = \{AB \rightarrow C, AC \rightarrow B, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$

- Candidate keys?
  - $H$ has to be in all candidate keys
  - $E$ has to be in all candidate keys
  - $G$ cannot be in any candidate key (since $E$ is in all candidate keys already).
  - Since $AB \rightarrow C, AC \rightarrow B$ and $BC \rightarrow A$, we know no candidate key can have $ABC$ together.
  - $AEH, BEH, CEH$ are not superkeys.
  - Try $ABEH, ACEH, BCEH$. They are all superkeys. And we know they are all candidate keys (since above properties)
  - These are the only candidate keys: (1) each candidate key either contains $A$, or $B$, or $C$ since no attributes other than $A, B, C$ determine $A, B, C$, and (2) if a candidate key contains $A$, then it must contain either $B$, or $C$, and so on.
Example

- Same as previous
- Not in BCNF, not in 3NF
- Decomposition:

R(A, B, C, D, E, G, H)
F={AB → C, AC → B, B → D, BC → A, E → G}
Example

• $R(A, B, C, D, E, G, H)$
  
  $F = \{ AB \rightarrow C, AC \rightarrow B, B \rightarrow D, BC \rightarrow A, E \rightarrow G \}$

• Decomposition: BD, ABC, EG, ABEH

• Why good decomposition?
  – They are all in BCNF
  – Lossless-join decomposition
  – All dependencies are preserved.
Example

• R(A, B, D, E) decomposed into R1(A, B, D), R2 (A, B, E)
• F={AB → DE}
• It is a dependency preserving decomposition!
  – AB → D can be checked in R1
  – AB → E can be checked in R2
  – {AB → DE} is equivalent to {AB → D, AB → E}