Lecture 07: Decomposition to Normal Forms

Outline

- Attribute closure algorithm
- Schema decomposition methods
- Problems with obtaining both dependency preservation and lossless-join property
Closure

By soundness and completeness

\[
\text{closure}(F, X) = \{ A \mid F \vdash X \rightarrow A \} = \{ A \mid X \rightarrow A \in F^+ \}
\]

Claim 2 (from previous lecture)

\(Y \rightarrow W \in F^+\) if and only if \(W \subseteq \text{closure}(F, Y)\).

If we had an algorithm for \(\text{closure}(F, X)\), then we would have a (brute force!) algorithm for enumerating \(F^+\):

\(F^+\)

- for every subset \(Y \subseteq \text{atts}(F)\)
  - for every subset \(Z \subseteq \text{closure}(F, Y)\),
    - output \(Y \rightarrow Z\)
Attribute Closure Algorithm

Input: a set of FDs $F$ and a set of attributes $X$.

Output: $Y = \text{closure}(F, X)$

1. $Y := X$

2. while there is some $S \rightarrow T \in F$ with $S \subseteq Y$ and $T \not\subseteq Y$, then
   $Y := Y \cup T$. 

An Example (UW1997, Exercise 3.6.1)

$R(A, B, C, D)$ with $F$ made up of the FDs

$$A, B \rightarrow C$$
$$C \rightarrow D$$
$$D \rightarrow A$$

What is $F^+$?

Brute force!

Let’s just consider all possible nonempty sets $X$ — there are only 15...
Example (cont.)

\[ F = \{ A, B \rightarrow C, \ C \rightarrow D, \ D \rightarrow A \} \]

For the single attributes we have

- \( \{A\}^+ = \{A\} \),
- \( \{B\}^+ = \{B\} \),
- \( \{C\}^+ = \{A, C, D\} \),

- \( \{C\} \xrightarrow{C \rightarrow D} \{C, D\} \xrightarrow{D \rightarrow A} \{A, C, D\} \)

- \( \{D\}^+ = \{A, D\} \)

- \( \{D\} \xrightarrow{D \rightarrow A} \{A, D\} \)

The only new dependency we get with a single attribute on the left is \( C \rightarrow A \).
Example (cont.)

\[ F = \{A, B \rightarrow C, \ C \rightarrow D, \ D \rightarrow A\} \]

Now consider pairs of attributes.

- \{A, B\}^+ = \{A, B, C, D\},
  - so \(A, B \rightarrow D\) is a new dependency

- \{A, C\}^+ = \{A, C, D\},
  - so \(A, C \rightarrow D\) is a new dependency

- \{A, D\}^+ = \{A, D\},
  - so nothing new.

- \{B, C\}^+ = \{A, B, C, D\},
  - so \(B, C \rightarrow A, D\) is a new dependency

- \{B, D\}^+ = \{A, B, C, D\},
  - so \(B, D \rightarrow A, C\) is a new dependency

- \{C, D\}^+ = \{A, C, D\},
  - so \(C, D \rightarrow A\) is a new dependency
Example (cont.)

\[ F = \{A, B \rightarrow C, \ C \rightarrow D, \ D \rightarrow A\} \]

For the triples of attributes:

- \( \{A, C, D\}^+ = \{A, C, D\} \)  
  - so \( A, B, D \rightarrow C \) is a new dependency

- \( \{A, B, D\}^+ = \{A, B, C, D\} \)  
  - so \( A, B, C \rightarrow D \) is a new dependency

- \( \{B, C, D\}^+ = \{A, B, C, D\} \)  
  - so \( B, C, D \rightarrow A \) is a new dependency

And since \( \{A, B, C, D\}^+ = \{A, B, C, D\} \), we get no new dependencies with four attributes.
We generated 11 new FDs:

<table>
<thead>
<tr>
<th>Left Side</th>
<th>Right Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>A, C</td>
<td>D</td>
</tr>
<tr>
<td>B, C</td>
<td>D</td>
</tr>
<tr>
<td>B, D</td>
<td>C</td>
</tr>
<tr>
<td>A, B, C</td>
<td>D</td>
</tr>
<tr>
<td>B, C, D</td>
<td>A</td>
</tr>
<tr>
<td>A, B</td>
<td>D</td>
</tr>
<tr>
<td>B, C</td>
<td>A</td>
</tr>
<tr>
<td>B, D</td>
<td>A</td>
</tr>
<tr>
<td>C, D</td>
<td>A</td>
</tr>
</tbody>
</table>

Can you see the Key?

{A, B}, {B, C}, and {B, D} are keys.

Note: this schema is already in 3NF! Why?
General Decomposition Method (GDM)

GDM

1. Understand your FDs $F$ (compute $F^+$),
2. find $R(X) = R(Z, W, Y)$ (sets $Z$, $W$ and $Y$ are disjoint) with FD $Z \rightarrow W \in F^+$ violating a condition of desired NF,
3. split $R$ into two tables $R_1(Z, W)$ and $R_2(Z, Y)$
4. wash, rinse, repeat

Reminder

For $Z \rightarrow W$, if we assume $Z \cap W = \{\}$, then the conditions are

1. $Z$ is a superkey for $R$ (2NF, 3NF, BCNF)
2. $W$ is a subset of some key (2NF, 3NF)
3. $Z$ is not a proper subset of any key (2NF)
The lossless-join condition is guaranteed by GDM

- This method will produce a lossless-join decomposition because of (repeated applications of) Heath’s Rule!
- That is, each time we replace an $S$ by $S_1$ and $S_2$, we will always be able to recover $S$ as $S_1 \bowtie S_2$.
- Note that in GDM step 3, the FD $Z \rightarrow W$ may represent a key constraint for $R_1$.

But does the method always terminate? Please think about this ....
Return to Example — Decompose to BCNF

\[ R(A, B, C, D) \]

\[ F = \{ A, B \rightarrow C, \; C \rightarrow D, \; D \rightarrow A \} \]

Which FDs in \( F^+ \) violate BCNF?

\[
\begin{align*}
C & \rightarrow A \\
C & \rightarrow D \\
D & \rightarrow A \\
A, C & \rightarrow D \\
C, D & \rightarrow A
\end{align*}
\]
Return to Example — Decompose to BCNF

Decompose \( R(A, B, C, D) \) to BCNF

Use \( C \rightarrow D \) to obtain

- \( R_1(C, D) \). This is in BCNF. Done.
- \( R_2(A, B, C) \) This is not in BCNF. Why? \( A, B \) and \( B, C \) are the only keys, and \( C \rightarrow A \) is a FD for \( R_1 \). So use \( C \rightarrow A \) to obtain
  - \( R_{2.1}(A, C) \). This is in BCNF. Done.
  - \( R_{2.2}(B, C) \). This is in BCNF. Done.

Exercise: Try starting with any of the other BCNF violations and see where you end up.
The GDM **does not** always preserve dependencies!

\[ R(A, B, C, D, E) \]

- \( \{A, B\}^+ = \{A, B, C, D\} \),
- so \( A, B \rightarrow C, D \),
- and \( \{A, B, E\} \) is a key.

- \( \{B, E\}^+ = \{B, C, D, E\} \),
- so \( B, E \rightarrow C, D \),
- and \( \{A, B, E\} \) is a key (again)

Let’s try for a BCNF decomposition …
Decompose \( R(A, B, C, D, E) \) using \( A, B \rightarrow C, D \) :

- \( R_1(A, B, C, D) \). Decompose this using \( B \rightarrow D \):
  - \( R_{1.1}(B, D) \). Done.
  - \( R_{1.2}(A, B, C) \). Done.

- \( R_2(A, B, E) \). Done.

But in this decomposition, how will we enforce this dependency?

\[
D, E \rightarrow C
\]
Decompose $R(A, B, C, D, E)$ using $B, E \rightarrow C, D$:

- $R_3(B, C, D, E)$. Decompose this using $D, E \rightarrow C$
  - $R_{3.1}(C, D, E)$. Done.
  - $R_{3.2}(B, D, E)$. Decompose this using $B \rightarrow D$:
    - $R_{3.2.1}(B, D)$. Done.
    - $R_{3.2.2}(B, E)$. Done.

- $R_4(A, B, E)$. Done.

But in this decomposition, how will we enforce this dependency?

$$A, B \rightarrow C$$
Summary

- It always is possible to obtain BCNF that has the lossless-join property (using GDM)
  - But the result may not preserve all dependencies.
- It is always possible to obtain 3NF that preserves dependencies and has the lossless-join property.
  - Using methods based on “minimal covers” (for example, see EN2000).