BCNF Decomposition Algorithm

BCNFDecomp(R):

Find X s.t.: X + ≠ X and X + ≠ [all attributes]
if (not found) then Return R
let Y = X + - X, Z = (X + )
Decompose R into R1(X È Y) and R2(X È Z)
Return BCNFDecomp(R1), BCNFDecomp(R2)
BCNF Decomposition Algorithm

BCNFDecomp(R):

Find a set of attributes X s.t.: \( X^+ \neq X \) and \( X^+ \neq \) [all attributes]

Find a set of attributes X which has non-trivial “bad” FDs, i.e. is not a superkey, using closures
BCNF Decomposition Algorithm

BCNFDdecomp(R):
Find a set of attributes X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) then Return R

If no “bad” FDs found, in BCNF!
BCNFDecomp(R):
Find a set of attributes X s.t.: \( X^+ \neq X \) and \( X^+ \neq \) [all attributes]

if (not found) then Return R

let \( Y = X^+ - X \), \( Z = (X^+)^C \)

Let \( Y \) be the attributes that \( X \) functionally determines (+ that are not in \( X \))
And let \( Z \) be the other attributes that it doesn’t
BCNF Decomposition Algorithm

BCNFDecomp(R):
   Find a set of attributes X s.t.: X⁺ ≠ X and X⁺ ≠ [all attributes]

   if (not found) then Return R

   let Y = X⁺ - X, Z = (X⁺)ᶜ
   decompose R into R₁(X ∪ Y) and R₂(X ∪ Z)

Split into one relation (table) with X plus the attributes that X determines (Y)…
BCNF Decomposition Algorithm

BCNFDecomp(R):
Find a set of attributes $X$ s.t.: $X^+ \neq X$ and $X^+ \neq \text{[all attributes]}

if (not found) then Return $R$

let $Y = X^+ - X$, $Z = (X^+)^C$

decompose $R$ into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

And one relation with $X$ plus the attributes it does not determine ($Z$)
BCNF Decomposition Algorithm

BCNFDcomp(R):
  Find a set of attributes X s.t.: X⁺ ≠ X and X⁺ ≠ [all attributes]

  if (not found) then Return R

  let Y = X⁺ - X, Z = (X⁺)ᶜ
  decompose R into R₁(X ∪ Y) and R₂(X ∪ Z)

  Return BCNFDcomp(R₁), BCNFDcomp(R₂)

Proceed recursively until no more “bad” FDs!
Another way of representing the same concept

BCNFDcomp(R):
If $X \rightarrow A$ causes BCNF violation:

Decompose $R$ into

$R1 = XA$

$R2 = R - A$

(Note: $X$ is present in both $R1$ and $R2$)

Continue decomposing until no BCNF violation
Example

BCNFDekomp(R):
Find a set of attributes \( X \) s.t.: \( X^+ \neq X \) and \( X^+ \neq \) [all attributes]

\[
\text{if} \ (\text{not found}) \text{ then Return } R
\]

\[
\text{let } Y = X^+ - X, \ Z = (X^+)^C
\]

\[
\text{decompose } R \text{ into } R_1(X \cup Y) \text{ and } R_2(X \cup Z)
\]

\[
\text{Return BCNFDekomp}(R_1), \ \text{BCNFDekomp}(R_2)
\]

BCNFDekomp(R):
If \( X \rightarrow A \) causes BCNF violation:
Decompose R into
\( R_1 = XA \)
\( R_2 = R - A \)
(Note: \( X \) is present in both \( R_1 \) and \( R_2 \))

Continue decomposing until no BCNF violation

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

\[
\{A\} \rightarrow \{B,C\} \\
\{C\} \rightarrow \{D\}
\]
Example

\[ R(A, B, C, D, E) \]
\[ \{A\}^+ = \{A, B, C, D\} \neq \{A, B, C, D, E\} \]

\[ R_1(A, B, C, D) \]
\[ \{C\}^+ = \{C, D\} \neq \{A, B, C, D\} \]

\[ R_{11}(C, D) \]
\[ R_{12}(A, B, C) \]
\[ R_2(A, E) \]

\[ \{A\} \rightarrow \{B, C\} \]
\[ \{C\} \rightarrow \{D\} \]
DECOMPOSITIONS
Recap: Decompose to remove redundancies

1. We saw that **redundancies** in the data ("bad FDs") can lead to data anomalies

2. We developed mechanisms to **detect and remove redundancies by decomposing tables into BCNF**
   1. BCNF decomposition is *standard practice* - very powerful & widely used!

3. However, sometimes decompositions can lead to **more subtle unwanted effects**...

When does this happen?
Decompositions in General

$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$

$R_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$

$R_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$

$R_1 = \text{the projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m$

$R_2 = \text{the projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p$
# Theory of Decomposition

Sometimes a decomposition is “correct”

I.e., it is a **Lossless decomposition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Lossy Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

However sometimes it isn’t

What’s wrong here?

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Lossless Decompositions

A decomposition $R$ to $(R_1, R_2)$ is **lossless** if $R = R_1 \Join R_2$.
BCNF decomposition is always lossless. Why?

If \( \{A_1, ..., A_n\} \rightarrow \{B_1, ..., B_m\} \)

Then the decomposition is lossless

Note: don’t need
\( \{A_1, ..., A_n\} \rightarrow \{C_1, ..., C_p\} \)
A relation TEACH that is in 3NF but not in BCNF

- Two FDs exist in the relation TEACH:
  - \{student, course\} \rightarrow instructor
  - instructor \rightarrow course

- \{student, course\} is a candidate key for this relation
- So this relation is in 3NF \textit{but not in BCNF}
- A relation NOT in BCNF should be decomposed
Three possible decompositions for relation TEACH

- D1: \{\text{student, instructor}\} and \{\text{student, course}\}

- D2: \{\text{course, instructor}\} and \{\text{course, student}\}

✓ D3: \{\text{instructor, course}\} and \{\text{instructor, student}\}
A problem with BCNF

Problem: To enforce a FD, must reconstruct original relation—on each insert!
A Problem with BCNF

We do a BCNF decomposition on a “bad” FD:
\{Unit\}+ = \{Unit, Company\}

We lose the FD \{Company, Product\} \rightarrow \{Unit\}!!
So Why is that a Problem?

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galaga99</td>
<td>UW</td>
<td>Databases</td>
</tr>
<tr>
<td>Bingo</td>
<td>UW</td>
<td>Databases</td>
</tr>
</tbody>
</table>

No problem so far. All *local* FD’s are satisfied.

Let’s put all the data back into a single table again:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galaga99</td>
<td>UW</td>
<td>Databases</td>
</tr>
<tr>
<td>Bingo</td>
<td>UW</td>
<td>Databases</td>
</tr>
</tbody>
</table>

Violates the FD \( \{\text{Company}, \text{Product}\} \rightarrow \{\text{Unit}\} \!! \)
The Problem

• We started with a table $R$ and FDs $F$

• We decomposed $R$ into BCNF tables $R_1$, $R_2$, ... with their own FDs $F_1$, $F_2$, ...

• We insert some tuples into each of the relations—which satisfy their local FDs but when reconstruct it violates some FD across tables!

Practical Problem: To enforce FD, must reconstruct $R$—on each insert!
Possible Solutions

• Various ways to handle so that decompositions are all lossless / no FDs lost
  – For example 3NF- stop short of full BCNF decompositions.

• Usually a tradeoff between redundancy / data anomalies and FD preservation...

BCNF still most common- with additional steps to keep track of lost FDs...
• Constraints allow one to reason about **redundancy** in the data

• Normal forms describe how to **remove** this redundancy by **decomposing** relations
  – Elegant—by representing data appropriately certain errors are essentially impossible
  – For FDs, BCNF is the normal form.

• A tradeoff for insert performance: 3NF
Another Example From Problem Set 5

• Let the relation schema \( R(A,B,C,D) \) is given. For each of the following set of FDs do the following:

• i) indicate all the BCNF violations. Do not forget to consider FD’s that are not in the given set. However, it is not required to give violations that have more than one attribute on the right side.

• ii) Decompose the relations, as necessary, into collections of relations that are in BCNF.

• 1. FDs \( AB \rightarrow C, C \rightarrow D, \) and \( D \rightarrow A \)
There are 14 nontrivial dependencies. They are:
C→A, C→D, D→A, AB→D, AB→C,
AC→D, BC→A, BC→D, BD→A, BD→C, CD→A,
ABC→D, ABD→C, and BCD→A.
• One choice is to decompose using the violation C→D.
• Using the above FDs, we get ACD (Because of C→D and C→A) and BC as decomposed relations.
• BC is surely in BCNF, since any two-attribute relation is.
• we discover that ACD is not in BCNF since C is its only key.
• We must further decompose ACD into AD and CD.
• Thus, the three relations of the decomposition are BC, AD, and CD.
Two other topics to Study

• Cover and Minimal Cover

• Let $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$.
• Let $G = \{A \rightarrow CD, E \rightarrow AH\}$.

• Show that:
• 1. $G$ covers $F$
• 2. $F$ covers $G$
• 3. $F$ and $G$ are equivalent
We say that a set of functional dependencies $F$ covers another set of functional dependencies $G$, if every functional dependency in $G$ can be inferred from $F$. More formally, $F$ covers $G$ if $G^+ \subseteq F^+$.

$F$ is a **minimal cover** of $G$ if $F$ is the smallest set of functional dependencies that cover $G$. 
1. Show: G covers F or \((F \subseteq G^+)\)

    let’s check each FD in F:

    - \(A \rightarrow C\)
      So, let’s find \(A^+\) in G.
      \(A^+\) in G = \(\{ACD\}\) which includes C. So, continue;

    - \(AC \rightarrow D\)
      Let’s get \(AC^+\) in G,
      \(AC^+\) in G = \(\{ACD\}\) which contains D. So, continue;

    - \(E \rightarrow AD\)
      Let’s get \(E^+\) in G,
      \(E^+\) in G = \(\{EACDH\}\) which contains \(AD\). So, continue;

We found that every dependency in F can be inferred from G.
So, we can say that G covers F.
Problem Set 5


• Solution:

• Really important for Midterm!
Acknowledgement

• Some of the slides in this presentation are taken from the slides provided by the authors.

• Many of these slides are taken from cs145 course offered by Stanford University.

• Thanks to YouTube, especially to Dr. Daniel Soper for his useful videos.