

CSC 261/461 – Database Systems

Lecture 13

Spring 2018

BCNF Decomposition Algorithm

BCNFDecomp(R):

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

BCNFDecomp(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!

BCNF Decomposition Algorithm

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):

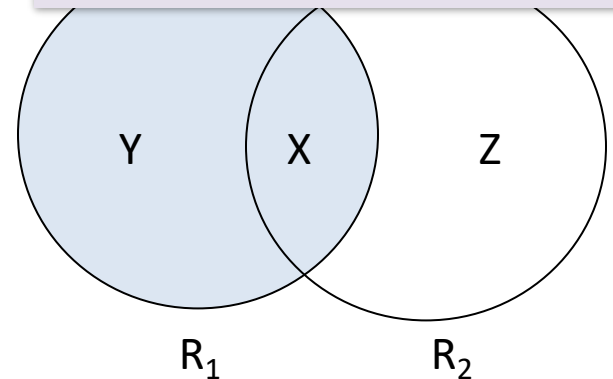
Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) then Return R

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

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

Split into one relation (table) with X plus the attributes that X determines (Y)...



BCNF Decomposition Algorithm

BCNFDecomp(R):

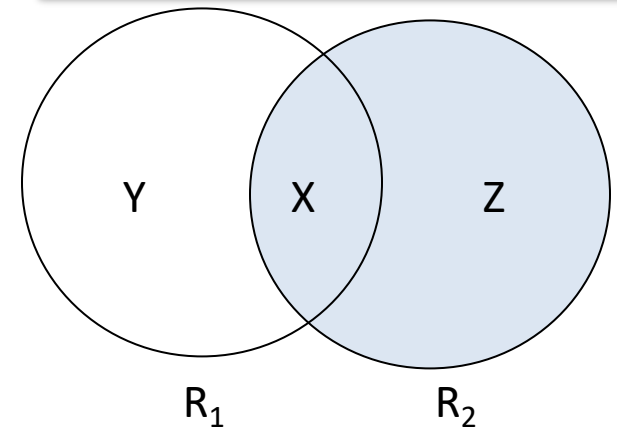
Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

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

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 $Y = X^+ - X$, $Z = (X^+)^c$

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

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

Proceed recursively until no
more “bad” FDs!

Another way of representing the same concept

BCNFDecomp(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

Example

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 $Y = X^+ - X$, $Z = (X^+)^c$

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

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

$R(A, B, C, D, E)$

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

BCNFDecomp(R):

If $X \rightarrow A$ causes BCNF violation:

Decompose R into

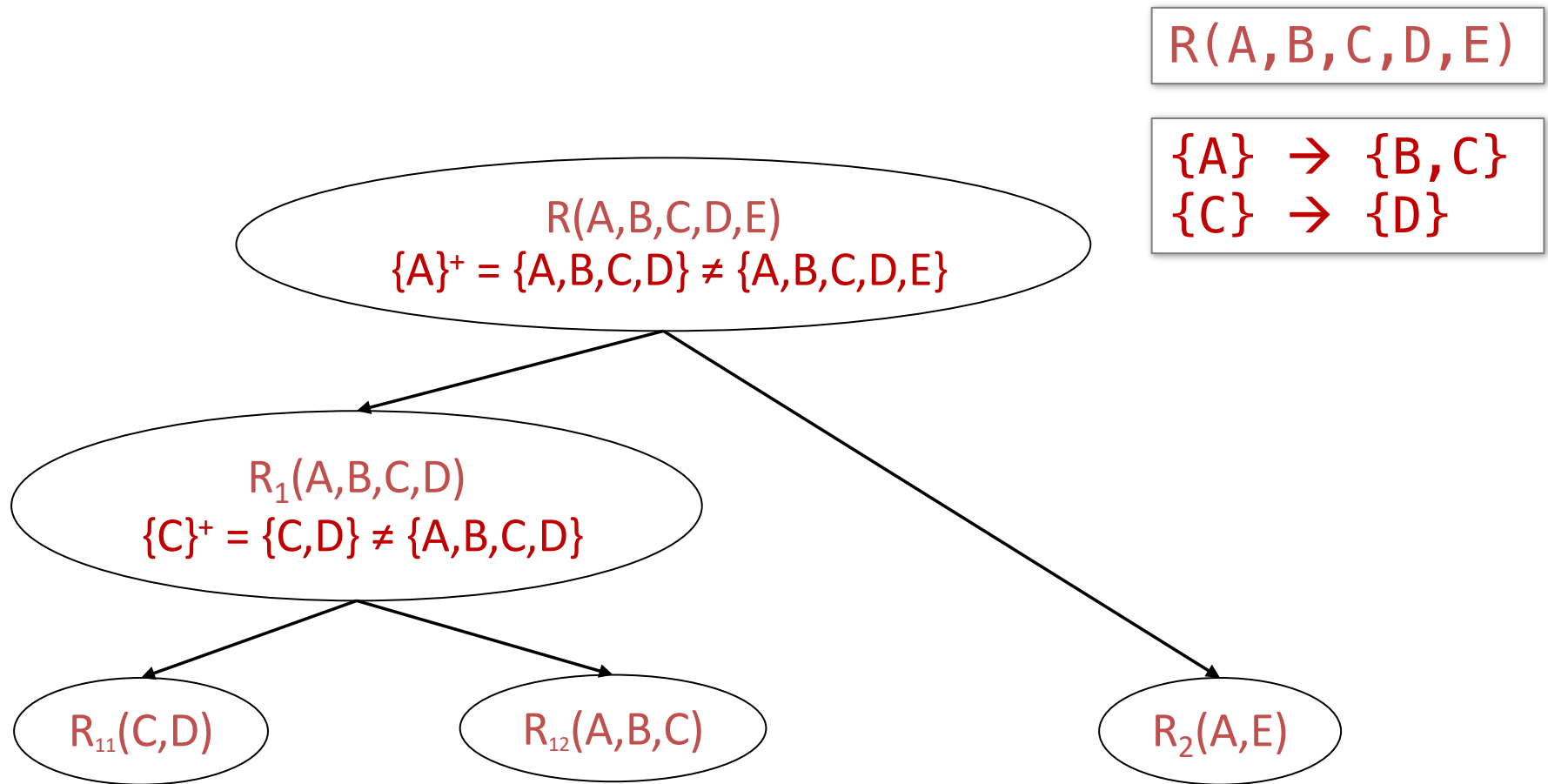
$R_1 = XA$

$R_2 = R - A$

(Note: X is present in both R_1 and R_2)

Continue decomposing until no BCNF violation

Example



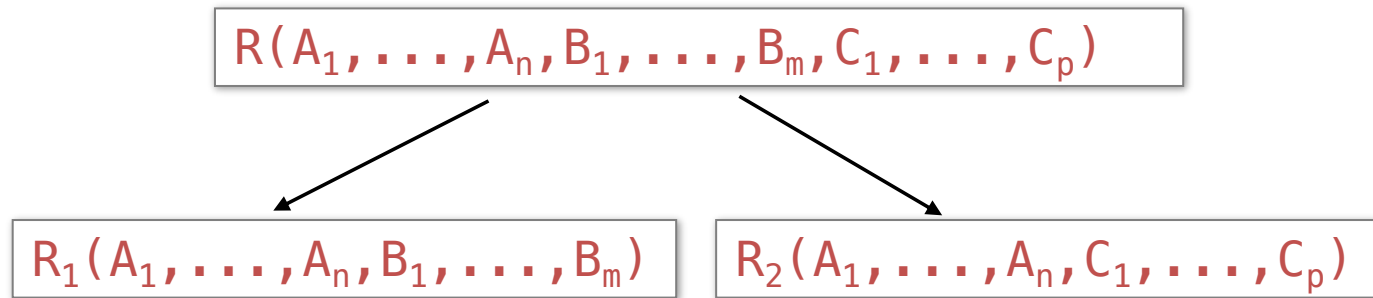
DECOMPOSITIONS

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_1 = the *projection* of R on $A_1, \dots, A_n, B_1, \dots, B_m$


R_2 = the *projection* of R on $A_1, \dots, A_n, C_1, \dots, C_p$

Theory of Decomposition


Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Sometimes a decomposition is “correct”

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



Name	Price
Gizmo	19.99
OneClick	24.99
Gizmo	19.99




Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Lossy Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

*However
sometimes it isn't*

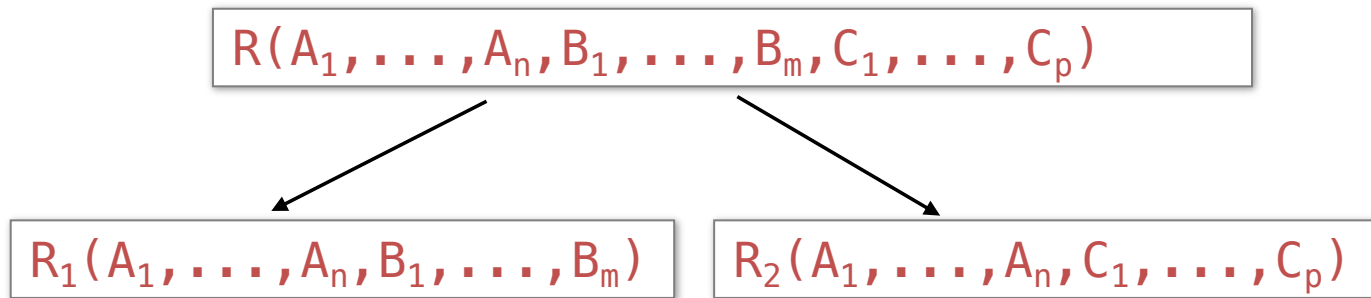
What's wrong
here?



Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

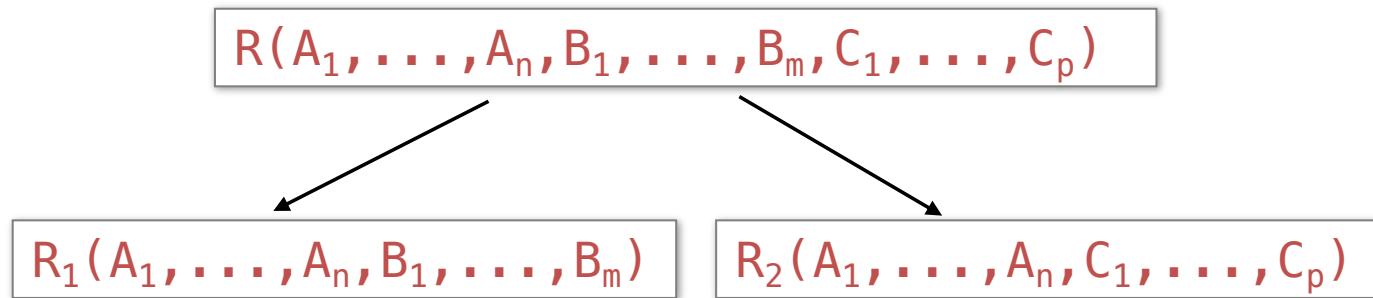
Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossless Decompositions



A decomposition R to (R_1, R_2) is **lossless** if $R = R_1 \text{ Join } R_2$

Lossless Decompositions



If $\{A_1, \dots, A_n\} \rightarrow \{B_1, \dots, B_m\}$
Then the decomposition is lossless

Note: don't need $\{A_1, \dots, A_n\} \rightarrow \{C_1, \dots, C_p\}$

BCNF decomposition is always lossless. Why?

A relation TEACH that is in 3NF but not in BCNF

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

- Two FDs exist in the relation TEACH:

X

→ A

- {student, course} → instructor
- instructor → course

- {student, course} is a candidate key for this relation
- So this relation is in 3NF *but not in* BCNF
- A relation **NOT** in BCNF should be decomposed

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
 - D1: {student, instructor} and {student, course}
 - D2: {course, instructor } and {course, student}
 - ✓ – D3: {instructor, course } and {instructor, student}

A problem with BCNF

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

A Problem with BCNF

Unit	Company	Product
...

<u>Unit</u>	Company
...	...

Unit	Product
...	...

$\{\text{Unit}\} \rightarrow \{\text{Company}\}$

$\{\text{Unit}\} \rightarrow \{\text{Company}\}$
 $\{\text{Company}, \text{Product}\} \rightarrow \{\text{Unit}\}$

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

We lose the FD $\{\text{Company}, \text{Product}\} \rightarrow \{\text{Unit}\}!!$

So Why is that a Problem?

<u>Unit</u>	Company
Galaga99	UW
Bingo	UW

Unit	Product
Galaga99	Databases
Bingo	Databases

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

$\{\text{Unit}\} \rightarrow \{\text{Company}\}$

Unit	Company	Product
Galaga99	UW	Databases
Bingo	UW	Databases

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

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, \dots with their own FDs F_1, F_2, \dots
- 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...

Summary

- 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 $D \rightarrow A$

Find Non-trivial dependencies

There are 14 nontrivial dependencies, They are:

$C \rightarrow A$, $C \rightarrow D$, $D \rightarrow A$, $AB \rightarrow D$, $AB \rightarrow C$,
 $AC \rightarrow D$, $BC \rightarrow A$, $BC \rightarrow D$, $BD \rightarrow A$, $BD \rightarrow C$, $CD \rightarrow A$,
 $ABC \rightarrow D$, $ABD \rightarrow C$, and $BCD \rightarrow A$.

Proceed From There

- One choice is to decompose using the violation $C \rightarrow D$.
- Using the above FDs, we get ACD (Because of $C \rightarrow D$ and $C \rightarrow A$) 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

Cover

- 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

Example of Cover

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

- <http://www.cs.rochester.edu/courses/261/spring2018/ps/ps5.pdf>
- Solution:
- <http://www.cs.rochester.edu/courses/261/spring2018/ps/ps5sol.pdf>
- 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.