CZ2007 Introduction to Database Systems (Week 4)

Topic 3: Boyce-Codd Normal Form (2)

This Lecture

- Tricky case of BCNF
- Properties of BCNF

Tricky Case of BCNF Decomposition

- R(A, B, C, D, E)
- $A \rightarrow B$, $BC \rightarrow D$
- Key of R: ACE (one single composite key)
- $A \rightarrow B$ is a violation.
- Decompose R
  - $\{A\}^+ = \{A, B\}$
  - $R_1(A, B)$, $R_2(A, C, D, E)$
  - R_1 is in BCNF
  - How about R_2?
- Key of R_2: ACE
- Violations any?
- A bit tricky …

Tricky Case of BCNF Decomposition

- In general, we may have a tricky case in BCNF decomposition, if
  - We are checking whether a table T satisfies BCNF, and there is an FD $X \rightarrow Y$ such that
    - X contains some attribute in T, but
    - Y contains some attribute NOT in T
- Example in the previous slide:
  - We are checking $R_2(A, C, D, E)$
  - FDs: $A \rightarrow B$, $BC \rightarrow D$
  - A is in R_2, but B is not
  - This leads to a tricky case
  - In this case, we have to use closures to check whether R_2 is in BCNF

Checking BCNF in a Tricky Case

- We are checking $R_2(A, C, D, E)$
- FDs that we have: $A \rightarrow B$, $BC \rightarrow D$
- Check the closures:
- $\{A\}^+ = \{A, B\}$
- B is not in $R_2(A, C, D, E)$, so the closure becomes
- $\{A\}^+ = \{A\}$
- Similarly, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$, $\{E\}^+ = \{E\}$
- So far, none of these indicate a violation of BCNF (trivial FDs)
- We check further: $\{AC\}+ = \{ACD\}$
- This indicates that $AC \rightarrow D$ and AC does not contain key ACE of R_2
- This means that R_2 is not in BCNF
- So we need to decompose it

Checking BCNF in a Tricky Case

- We are checking $R_2(A, C, D, E)$
- FDs that we have: $A \rightarrow B$, $BC \rightarrow D$
- We know that $AC \rightarrow D$ violates BCNF on R_2
- Decompose $R_2(A, C, D, E)$
  - $\{AC\}^+ = \{A, C, D\}$
  - $R_3(A, C, D)$, $R_4(A, C, E)$
- R_4 is in BCNF (ACE is key)
- What about R_3?
- Tricky case ($A \rightarrow B$); need to use closure again

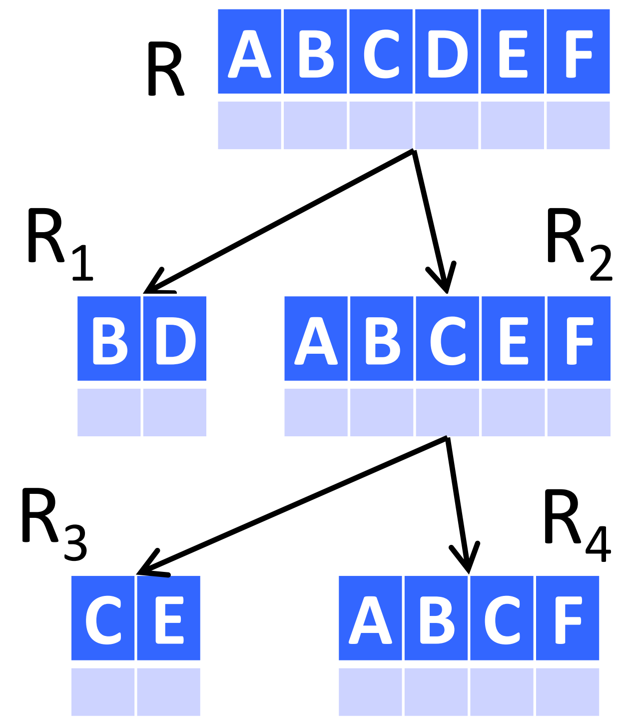
Checking BCNF in a Tricky Case

- We are checking $R_3(A, C, D)$
- FDs that we have: $A \rightarrow B$, $BC \rightarrow D$
- $\{A\}^+ = \{A, B\}$ ➔ $\{A\}^+ = \{A\}$ on R_3
- $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$
- $\{AC\}^+ = \{A, B, C, D\}$ ➔ $\{AC\}^+ = \{A, C, D\}$ on R_3
- $\{AD\}^+ = \{A, B, D\}$ ➔ $\{AD\}^+ = \{A, D\}$ on R_3
- $\{CD\}^+ = \{C, D\}$
- None of the closures indicate a violation of BCNF
- Therefore, R_3 is in BCNF
- Final decomposition: $R_1(A, B)$, $R_3(A, C, D)$, $R_4(A, C, E)$

Summary

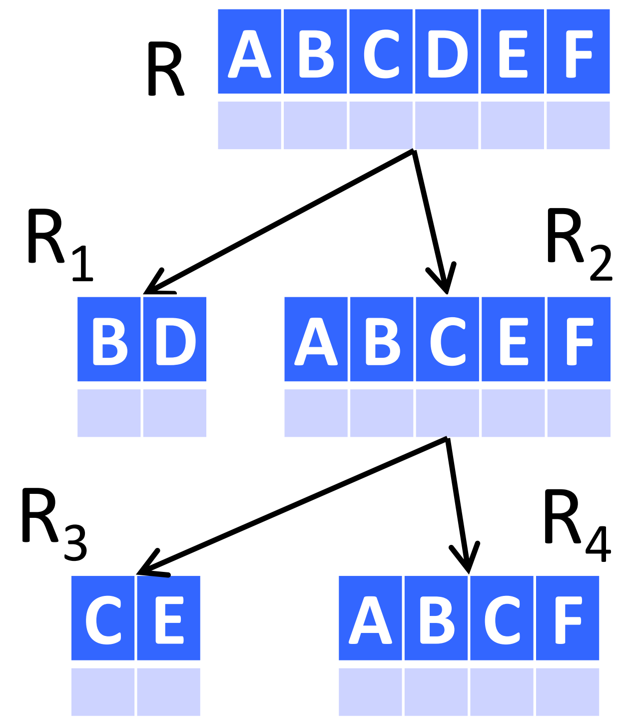
- Whenever we have a tricky case in checking BCNF, we need to resort to closures
- How to use closures to check that a table T is NOT in BCNF:
  - If there is a closure $\{X\}^+ = \{Y\}$, such that
    - Y does not contain all attributes in T, and (not all)
    - Y contains more attributes than X (more but)
- Previous example:
  - R_2(A, C, D, E), with A→B, BC→D
  - $\{AC\}^+ = \{A, C, D\}$
  - {A, C, D} does not contain all attributes in R_2, but
  - {A, C, D} contains more attributes than {AC}

Exercise



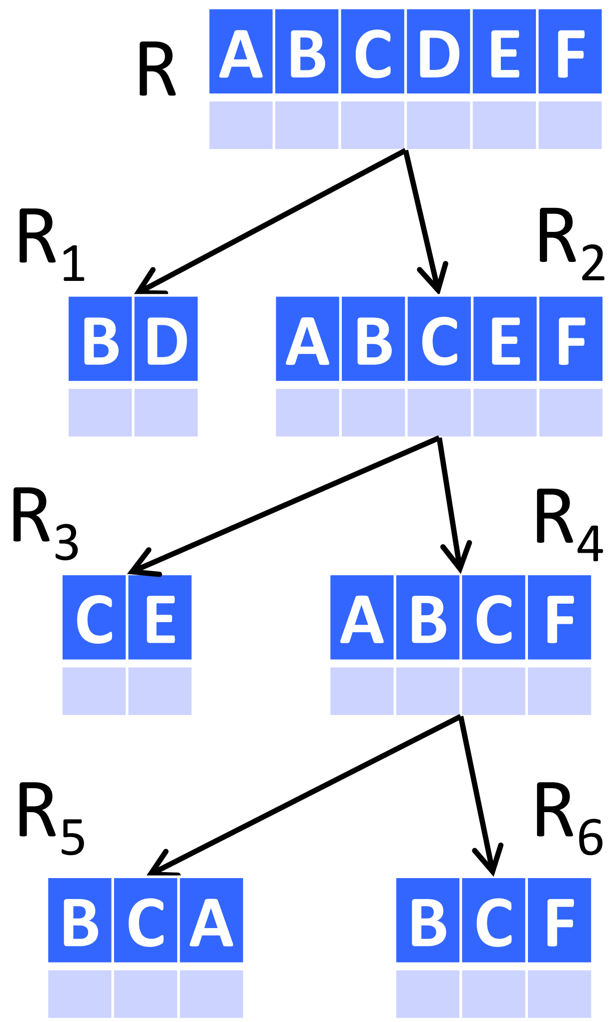
- R(A, B, C, D, E, F)
- Given FDs: B→D, C→E, DE→A
- Keys: BCF
- B→D violates BCNF
- Decompose R:
  - R1(B, D), R2(A, B, C, E, F)
- R1 is in BCNF
- What about R2? Tricky case. We could address the tricky case, but . . .
- we also notice C→E is violating. We can just decompose rightaway:
- Decompose R2, {C}+ = {C, E}
  - R3(C, E), R4(A, B, C, F)

Exercise



- R(A, B, C, D, E, F)
- Given FDs: B→D, C→E, DE→A
- R3(C, E) is in BCNF
- What about R4(A, B, C, F)?
- Tricky case
- Check closures
  - {A}+ = {A}, {B}+ = {BD}, {C}+ = {CE}, {F}+ = {F}
  - {AB}+ = {ABD}, {AC}+ = {ACE}, {AF}+ = {AF}, {BC}+ = {BCDEA}
- So there is a non-trivial FD: BC→A

Exercise



- R(A, B, C, D, E, F)
- Given FDs: B→D, C→E, DE→A
- R3(C, E) is in BCNF
- What about R4(A, B, C, F)?
- Keys of R4: BCF
- There is a non-trivial FD: BC→A
- It violates BCNF
- Decompose R4
  - R5(B, C, A), R6(B, C, F)
- Final decomposition:
  - R1(B, D), R3(C, E), R5(B, C, A), R6(B, C, F)

This Lecture

- Tricky case of BCNF
- Properties of BCNF

Properties of BCNF Decomposition

- Good properties
  - No update or deletion anomalies
  - Very small redundancy
  - The original table can always be reconstructed from the decomposed tables if functional dependencies are preserved
    (this is called the lossless join property)
  - Reconstruction is at the schema level only if some FDs not preserved

Lossless Join Property

Name	<u>NRIC</u>	<u>Phone</u>	Address
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- The table above can be perfectly reconstructed using the decomposed tables below as all FDs preserved:
- {NRIC, Phone} → Name, Address, NRIC → Name, Address

Name	<u>NRIC</u>	Address
Alice	1234	Jurong East
Bob	5678	Pasir Ris

<u>NRIC</u>	<u>Phone</u>
1234	67899876
1234	83848384
5678	98765432

Why BCNF guarantees lossless join?

- Say we decompose a table R into two tables R_1 and R_2
- The decomposition guarantees lossless join, whenever the common attributes in R_1 and R_2 constitute a superkey of R_1 or R_2
- Example
  - R(A, B, C) decomposed into R_1(A, B) and R_2 (B, C), with B being the key of R_2
  - R(A, B, C, D) decomposed into R_1(A, B, C) and R_2 (B, C, D), with BC being the key of R_1

Why BCNF guarantees lossless join?

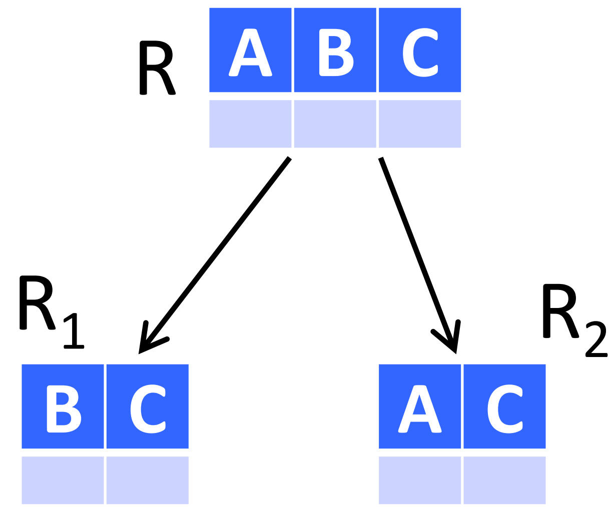
- The decomposition of R guarantees lossless join, whenever the common attributes in R_1 and R_2 constitute a superkey of R_1 or R_2
- BCNF Decomposition of R
  - Find a BCNF violation $X \rightarrow Y$
  - Compute $\{X\}^+$
  - R_1 contains all attributes in $\{X\}^+$
  - R_2 contains X and all attributes NOT in $\{X\}^+$
  - X is both in R_1 and R_2
  - And X is a superkey of R_1
  - Therefore, R_1 and R_2 is a lossless decomposition of R

Properties of BCNF Decomposition

- Good properties
  - No update or deletion anomalies
  - Very small redundancy
  - The original table can always be reconstructed from the decomposed tables if functional dependencies are preserved
    (this is called the lossless join property)
- Bad property
  - It may not preserve all functional dependencies

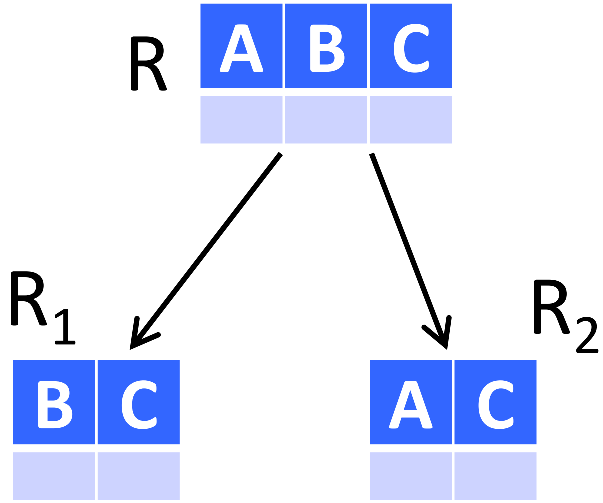
Dependency Preservation

- Given: Table R(A, B, C)
  - with AB→C, C→B
- Keys: AB, AC
- BCNF Decomposition
  - R_1(B, C)
  - R_2(A, C)
- Non-trivial FDs on R_1: C→B
- Non-trivial FDs on R_2: none
- The other FD, AB→C, should hold on any individual table, but it is "lost"
- We say that a BCNF decomposition does not always preserve all FDs



Dependency Preservation

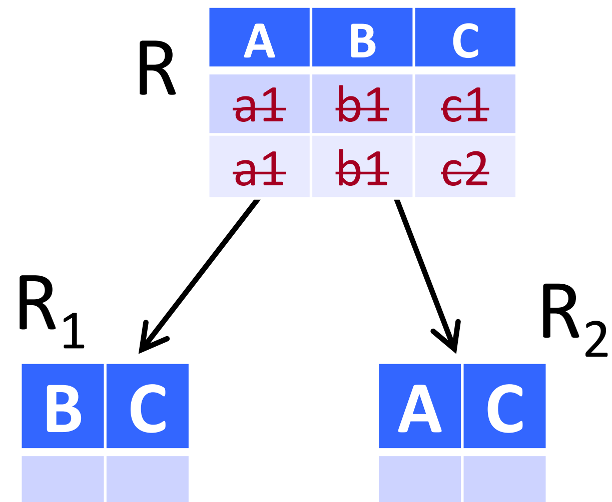
- Why do we want to preserve FDs?
- Because we want to make it easier to avoid "inappropriate" updates



- Previous example
  - We have two tables $R_1(B, C)$, $R_2(A, C)$
  - We have $C \rightarrow B$ and $AB \rightarrow C$
  - Due to $AB \rightarrow C$, we are not suppose to have two tuples (a1, b1, c1) and (a1, b1, c2)
  -
  -
  -

Dependency Preservation

- Why do we want to preserve FDs?
- Because we want to make it easier to avoid “inappropriate” updates



- Previous example
  - We have two tables $R_1(B, C)$, $R_2(A, C)$
  - We have $C \rightarrow B$ and $AB \rightarrow C$
  - Due to $AB \rightarrow C$, we are not suppose to have two tuples (a1, b1, c1) and (a1, b1, c2)
  - But as we store A and C separately in R_1 and R_2, it is not easy to check whether such two tuples exist at the same time

Dependency Preservation

- Why do we want to preserve FDs?
- Because we want to make it easier to avoid "inappropriate" updates
- Previous example
  - We have two tables $R_1(B, C)$, $R_2(A, C)$
  - We have $C \rightarrow B$ and $AB \rightarrow C$
  - Due to $AB \rightarrow C$, we are not suppose to have two tuples (a1, b1, c1) and (a1, b1, c2)
  - But as we store A and C separately in R_1 and R_2, it is not easy to check whether such two tuples exist at the same time
  - That is, if someone wants to insert (a1, b1, c2), it is not easy for us to check whether (a1, b1, c1) already exists
  - This is less than ideal

R

A	B	C
~~a1~~	~~b1~~	~~c1~~
~~a1~~	~~b1~~	~~c2~~

R_1

B	C
b1	c1
b1	c2

R_2

A	C
a1	c1
a1	c2

Third Normal Form (3NF)

- A relaxation of BCNF that
  - Is less strict
  - Allows decompositions that always preserve functional dependencies
- Will be discussed in the next lecture

Next lecture:

Topic 4: Third Normal Form (1)