

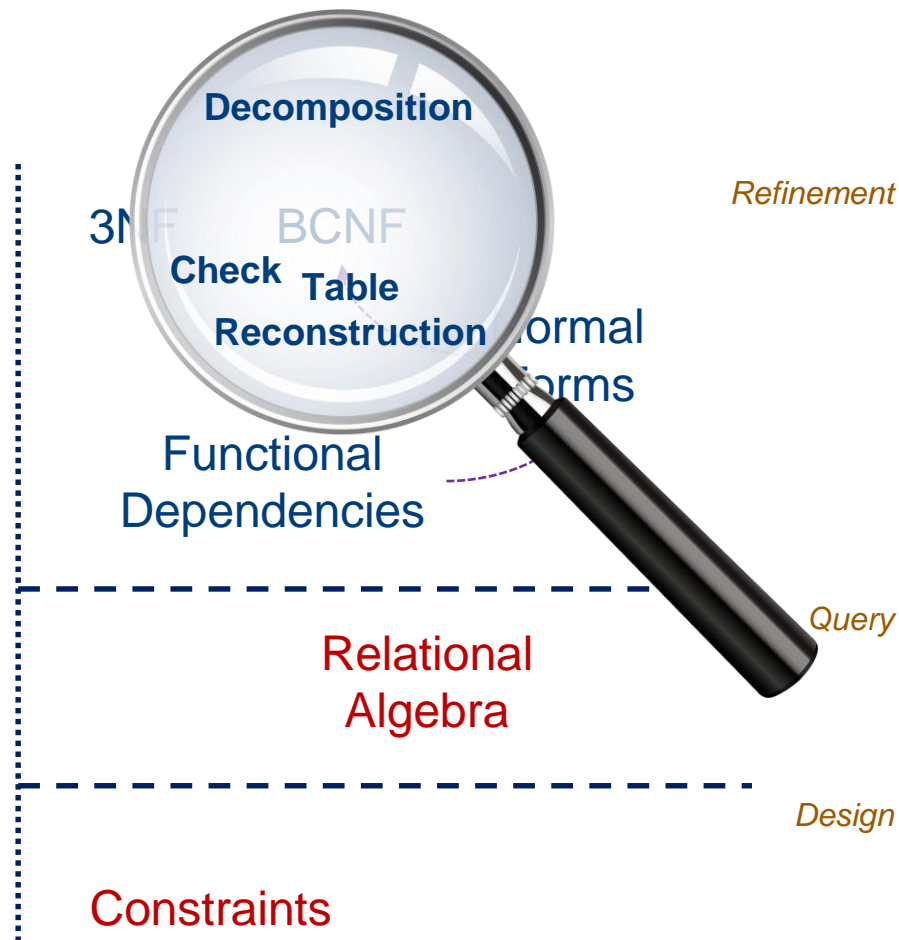
CS2102 Database Systems

Lecture 12 – Third Normal Form

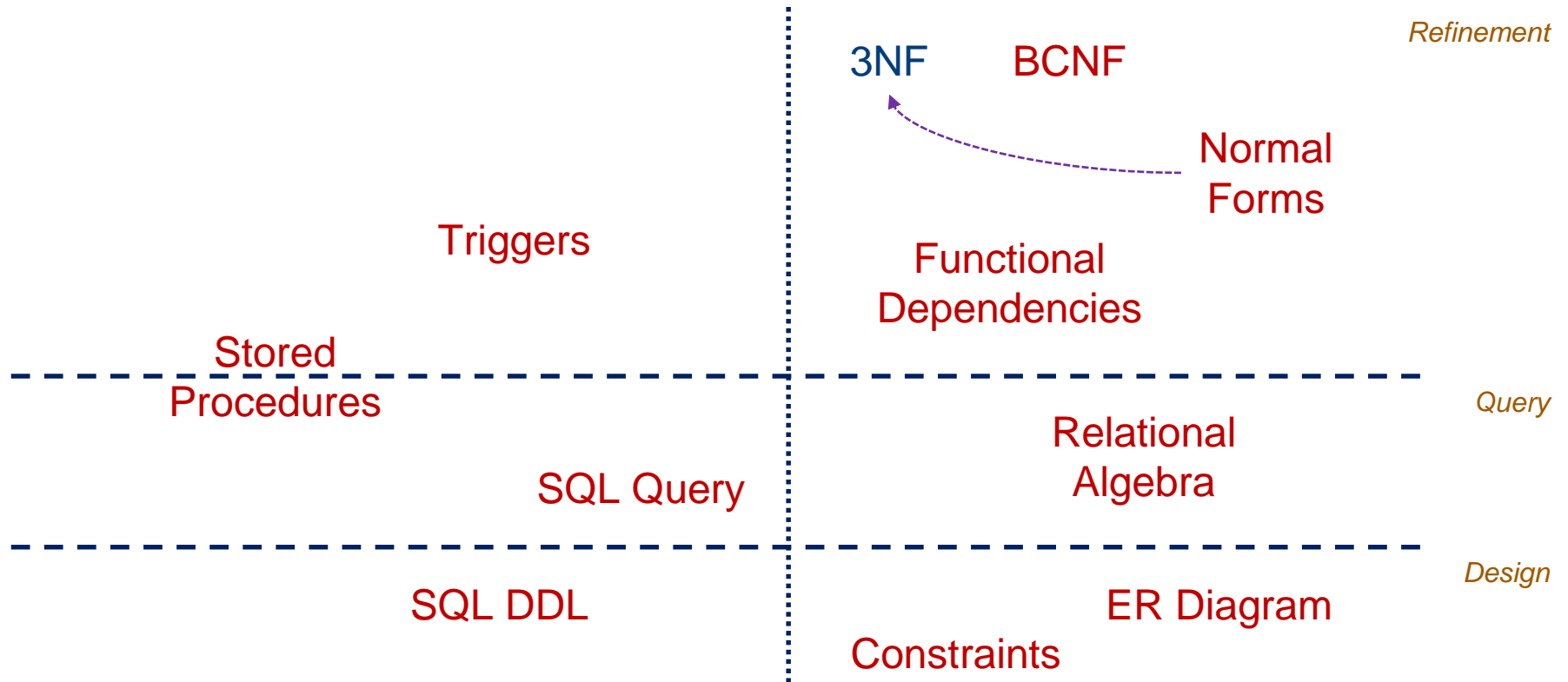
Roadmap

• Previously

- **BCNF Definition:**
 - Every *non-trivial & decomposed* FD has *superkey* as LHS
- **BCNF Check:**
 - Find violation: “*more but not all*”
- **BCNF Decomposition:**
 - For violation $A \rightarrow A^+$
 - $R_1(A^+)$
 - $R_2(A \cup \{R - A^+\})$
 - *Repeat* until all tables are in BCNF
- **BCNF Properties:**
 - ✓ No update or deletion anomalies
 - ✓ Small redundancies
 - ✓ Original table can be reconstructed
 - ✗ *Dependencies may not be preserved in the decomposed table*



Roadmap



Roadmap

- We will do this step by step

- Dependency Preservation

(why we need 3NF)



- Define 3NF



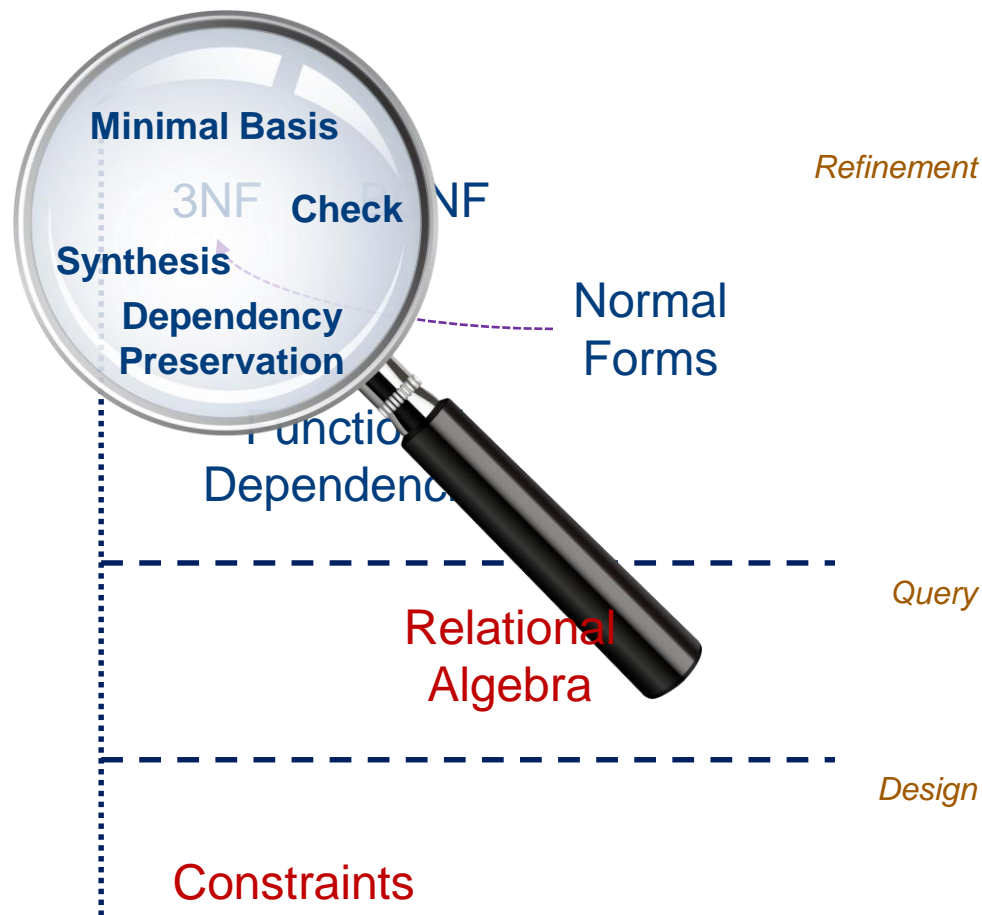
- Check 3NF



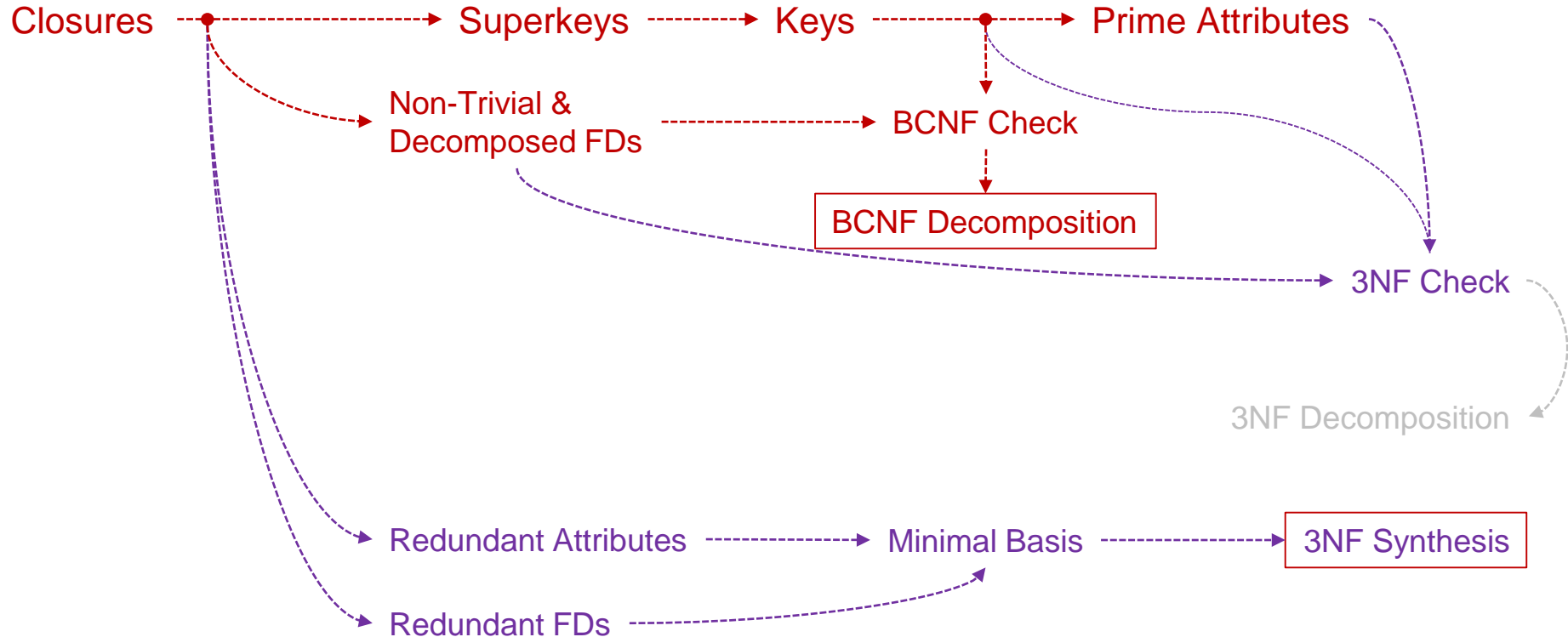
- Minimal Basis



- 3NF Synthesis



Algorithm Roadmap



Dependency Preservation

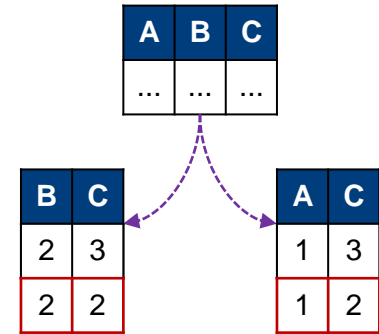
or why we need 3NF



Dependency Preservation

• Motivating Example

- Table $R(A, B, C)$ with $AB \rightarrow C$ and $C \rightarrow B$
- $\{C\} \rightarrow \{B, C\}$ violates BCNF of $R(A, B, C)$
 - $R1(B, C)$ *(non-trivial & decomposed FD on $R1$: $C \rightarrow B$)*
 - $R2(A, C)$ *(non-trivial & decomposed FD on $R2$: **nothing**)*
- Where is **$AB \rightarrow C$** ??
 - Cannot even be derived from FDs on $R1$ and $R2$
 - Totally “lost”

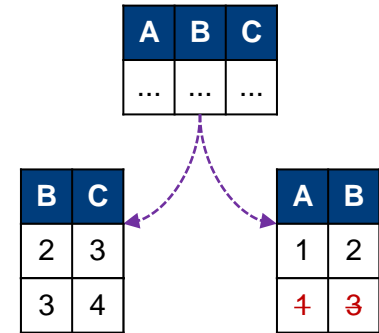


- ❖ This is why we say that a BCNF decomposition may not always preserve all FDs *(non dependency preserving decomposition)*
 - ❖ **Dilemma!**
 - ❖ **either the table has anomalies OR does not preserve constraints!**

Dependency Preservation

- “What We Want” Example

- Table $R(A, B, C)$ with $A \rightarrow B$, $B \rightarrow C$ and $A \rightarrow C$
 - $\{B\} \rightarrow \{B, C\}$ violates BCNF of $R(A, B, C)$
 - $R1(B, C)$ (non-trivial & decomposed FD on $R1: B \rightarrow C$)
 - $R2(A, B)$ (non-trivial & decomposed FD on $R2: A \rightarrow B$)
 - Where is $A \rightarrow C$??
 - Can be derived from FDs on $R1$ and $R2$
 - Compute $\{A\}^+$ w.r.t. $\{B \rightarrow C, A \rightarrow B\} = \{A, B, C\}$
 - So this FD is *preserved* even when it spans across multiple tables
- ❖ No valid insertion into $R1$ or $R2$ can violate these functional dependencies
- ❖ Gives rise to the notion of *FD equivalence*



Functional Dependency Equivalence

- **Definition**

- Let $F1$ and $F2$ be sets of FDs
- We say that $F1$ is *equivalent* to $F2$ (i.e., $F1 \equiv F2$) if and only if
 - Every FD in $F1$ can be derived from $F2$ (i.e., $F2 \vdash F1$)
 - Every FD in $F2$ can be derived from $F1$ (i.e., $F1 \vdash F2$)
 - ❖ *They can look different, but they carry the same information*
- ❖ In the context of decomposition, we can let $F2$ be the FD from the decomposed table
 - ❖ How do we get this FD from decomposed table?
 - ❖ *Union of projection*

Functional Dependency Equivalence

- **Example**

- **Example:** $R(A, B, C, D, E)$

- $F1 = \{A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E\}$

- $F2 = \{A \rightarrow BC, D \rightarrow AE\}$

- ❖ Show $F1 \equiv F2$

1. Prove that $F1$ can be derived from $F2$

- $A \rightarrow B$ and $D \rightarrow E$ can be derived easily using *decomposition rule*

- $\{AB\}^+ = \{ABC\}$ so $AB \rightarrow C$ is implied by $F2$

- $\{D\}^+ = \{ABCDE\}$ so $D \rightarrow AC$ is implied by $F2$

$\therefore F1$ can be derived from $F2$

Functional Dependency Equivalence

- **Example**

- **Example:** $R(A, B, C, D, E)$

- $F1 = \{A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E\}$

- $F2 = \{A \rightarrow BC, D \rightarrow AE\}$

- ❖ Show $F1 \equiv F2$

- 2. Prove that $F2$ can be derived from $F1$

- $\{A\}^+ = \{ABC\}$ so $A \rightarrow BC$ is implied by $F1$

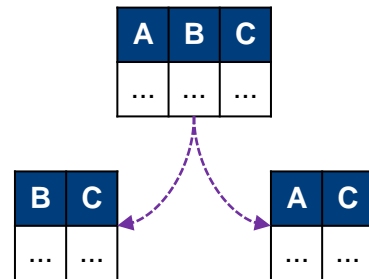
- $\{D\}^+ = \{ABCDE\}$ so $D \rightarrow AE$ is implied by $F1$

$\therefore F2$ can be derived from $F1$

Dependency Preservation

• Motivating Example

- Table $R(A, B, C)$ with $AB \rightarrow C$ and $C \rightarrow B$
- $\{C\} \rightarrow \{B, C\}$ violates BCNF of $R(A, B, C)$
 - $R1(B, C)$ *(non-trivial & decomposed FD on $R1$: $C \rightarrow B$)*
 - $R2(A, C)$ *(non-trivial & decomposed FD on $R2$: **nothing**)*



- $F1 = \{AB \rightarrow C, C \rightarrow B\}$
- $FR1 = \{C \rightarrow B\}, FR2 = \{\}$
 - $F2 = \{C \rightarrow B\}$

❖ We can show that $F1 \not\equiv F2$

- Can we still enforce $AB \rightarrow C$ on the decomposed table?
- YES! But not easy, need to use trigger!

3NF

Third Normal Form



3NF

- **Definition**

- A table R is in 3NF if *every non-trivial and decomposed FD either:*
 - *its left hand side is a superkey*
 - *the right hand side is a prime attributes* (*appears in a key*)

- **Example:** R(A, B, C) with $C \rightarrow B$, $AC \rightarrow B$ and $AB \rightarrow C$

- Key: {AB} and {AC}

- Prime Attributes: {ABC}

- Non-trivial and decomposed FDs on R:

- | | | |
|---------------------|----------------------|----------------------|
| - $C \rightarrow B$ | - $AB \rightarrow C$ | - $AC \rightarrow B$ |
| ○ B is PA | ○ AB is SK | ○ AC is SK |

∴ R satisfies 3NF

3NF

- **Definition**

- A table R is in 3NF if *every non-trivial and decomposed FD either:*
 - *its left hand side is a superkey*
 - *the right hand side is a prime attributes* (*appears in a key*)
- **Example:** R(A, B, C) with $A \rightarrow B$, $B \rightarrow C$, $AC \rightarrow B$ and $AB \rightarrow C$
 - Key: {A}
 - Prime Attributes: {A}
 - Non-trivial and decomposed FDs on R:
 - $A \rightarrow B$
 - A is SK
 - $B \rightarrow C$
 - B is **NOT** SK
 - C is **NOT** PA

∴ R violates 3NF

3NF vs BCNF

- **BCNF:** for any non-trivial and decomposed FD
 - The left hand side is a superkey

“Every attribute must depend ONLY on superkeys!”

NO EXCEPTION



- **3NF:** for any non-trivial and decomposed FD
 - The left hand side is a superkey
 - OR, the right hand side is a prime attribute

“Exceptions can be made for prime attributes”



3NF vs BCNF

- **BCNF**: for any non-trivial and decomposed FD
 - The left hand side is a superkey
- **3NF**: for any non-trivial and decomposed FD
 - The left hand side is a superkey
 - OR, the right hand side is a prime attribute
- **3NF** is more “*lenient*” than **BCNF**
 - Therefore
 - Satisfying BCNF \Rightarrow satisfying 3NF but not necessarily vice versa
 - Violating 3NF \Rightarrow violating BCNF but not necessarily vice versa

3NF

- **Checking 3NF**

- A table R is in **NOT** 3NF if *every non-trivial and decomposed FD both:*
 - *its left hand side is NOT a superkey*
 - *the right hand side is NOT a prime attributes (does not appear in any key)*
- **By Counterexample:**
 1. Consider all non-trivial and decomposed FDs of R
 2. For each non-trivial and decomposed FDs of R, check that
 - a. the left hand side is superkey
 - b. the right hand side is in prime attributes
 - If not one of them, then we have a counterexample
 3. If no counterexample found, then R satisfies 3NF

More but NOT All
 $\{S\} \subset \{S\}^+ \subset R$

3NF

- **Checking 3NF**

- A table R is in **NOT** 3NF if *every non-trivial and decomposed FD both:*
 - *its left hand side is NOT a superkey*
 - *the right hand side is NOT a prime attributes (does not appear in any key)*
- **By Counterexample:**
 1. Consider all subset of attributes of R
 2. Compute the closure of each subset
 3. Consider only the subset that are not superkey *(closure \neq R)*
 4. Remove the “trivial” attributes
 5. We have a counterexample, if
 - a. The resulting set is non-empty, **and**
 - b. There is an attribute in the right hand side that is not in the left hand side and not a prime attribute

3NF

• Checking 3NF

- **Example:** $R(A, B, C, D)$ with $AB \rightarrow C$, $C \rightarrow D$ and $D \rightarrow A$

1. Consider all subset of attributes of R
2. Compute the closure of each subset
3. Consider only the subset that are not superkey *(closure $\neq R$)*
4. Remove the “trivial” attributes
5. We have a counterexample, if
 - a. The resulting set is non-empty, **and**
 - b. There is an attribute in the right hand side that is not in the left hand side and not a prime attribute

- Keys = $\{AB\}$, $\{BC\}$, $\{BD\}$ Prime attributes = $\{ABCD\}$

$\{A\}^+ = \{A\}$	$\{B\}^+ = \{B\}$	$\{C\}^+ = \{ACD\}$	$\{D\}^+ = \{AD\}$
$\{AB\}^+ = \{ABCD\}$	$\{AC\}^+ = \{ACD\}$	$\{AD\}^+ = \{AD\}$	
$\{BC\}^+ = \{ABCD\}$	$\{BD\}^+ = \{ABCD\}$	$\{CD\}^+ = \{ACD\}$	
$\{ABC\}^+ = \{ABCD\}$	$\{ABD\}^+ = \{ABCD\}$	$\{ACD\}^+ = \{ACD\}$	$\{BCD\}^+ = \{ABCD\}$

3NF

Yes

No

• Checking 3NF

- A table R is in **NOT** 3NF if *every non-trivial and decomposed FD both:*
 - *its left hand side is NOT a superkey*
 - *the right hand side is NOT a prime attributes (does not appear in any key)*

- **Exercise:** R(A, B, C, D) with $B \rightarrow C$ and $B \rightarrow D$

- Keys = {AB} Prime attributes = {AB}

$\{A\}^+ =$	$\{B\}^+ =$	$\{C\}^+ =$	$\{D\}^+ =$
$\{AB\}^+ =$	$\{AC\}^+ =$	$\{AD\}^+ =$	
$\{BC\}^+ =$	$\{BD\}^+ =$	$\{CD\}^+ =$	
$\{ABC\}^+ =$	$\{ABD\}^+ =$	$\{ACD\}^+ =$	$\{BCD\}^+ =$

3NF

Yes

No

• Checking 3NF

- A table R is in **NOT** 3NF if *every non-trivial and decomposed FD both:*
 - *its left hand side is NOT a superkey*
 - *the right hand side is NOT a prime attributes (does not appear in any key)*
- **Exercise:** R(A, B, C, D) with $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$ and $D \rightarrow A$
 - Keys = {A}, {B}, {C}, {D} Prime attributes = {ABCD}

$\{A\}^+ =$	$\{B\}^+ =$	$\{C\}^+ =$	$\{D\}^+ =$
$\{AB\}^+ =$	$\{AC\}^+ =$	$\{AD\}^+ =$	
$\{BC\}^+ =$	$\{BD\}^+ =$	$\{CD\}^+ =$	
$\{ABC\}^+ =$	$\{ABD\}^+ =$	$\{ACD\}^+ =$	$\{BCD\}^+ =$

3NF

Yes

No

• Checking 3NF

- A table R is in **NOT** 3NF if *every non-trivial and decomposed FD both:*
 - *its left hand side is NOT a superkey*
 - *the right hand side is NOT a prime attributes (does not appear in any key)*
- **Exercise:** R(A, B, C, D) with $AB \rightarrow D$, $BD \rightarrow C$, $CD \rightarrow A$ and $AC \rightarrow B$
 - Keys = {AB}, {AC}, {BD}, {CD} Prime attributes = {ABCD}

$\{A\}^+$	=	$\{B\}^+$	=	$\{C\}^+$	=	$\{D\}^+$	=
$\{AB\}^+$	=	$\{AC\}^+$	=	$\{AD\}^+$	=		
$\{BC\}^+$	=	$\{BD\}^+$	=	$\{CD\}^+$	=		
$\{ABC\}^+$	=	$\{ABD\}^+$	=	$\{ACD\}^+$	=	$\{BCD\}^+$	=

3NF

Yes

No

- **Checking 3NF**

- A table R is in **NOT** 3NF if *every non-trivial and decomposed FD both:*
 - *its left hand side is NOT a superkey*
 - *the right hand side is NOT a prime attributes (does not appear in any key)*
- **Exercise:** $R(A, B, C, D, E)$ with $AB \rightarrow C$, $C \rightarrow E$, $E \rightarrow A$ and $E \rightarrow D$
 - Keys = $\{AB\}$, $\{BC\}$, $\{BE\}$ Prime attributes = $\{ABCE\}$

3NF

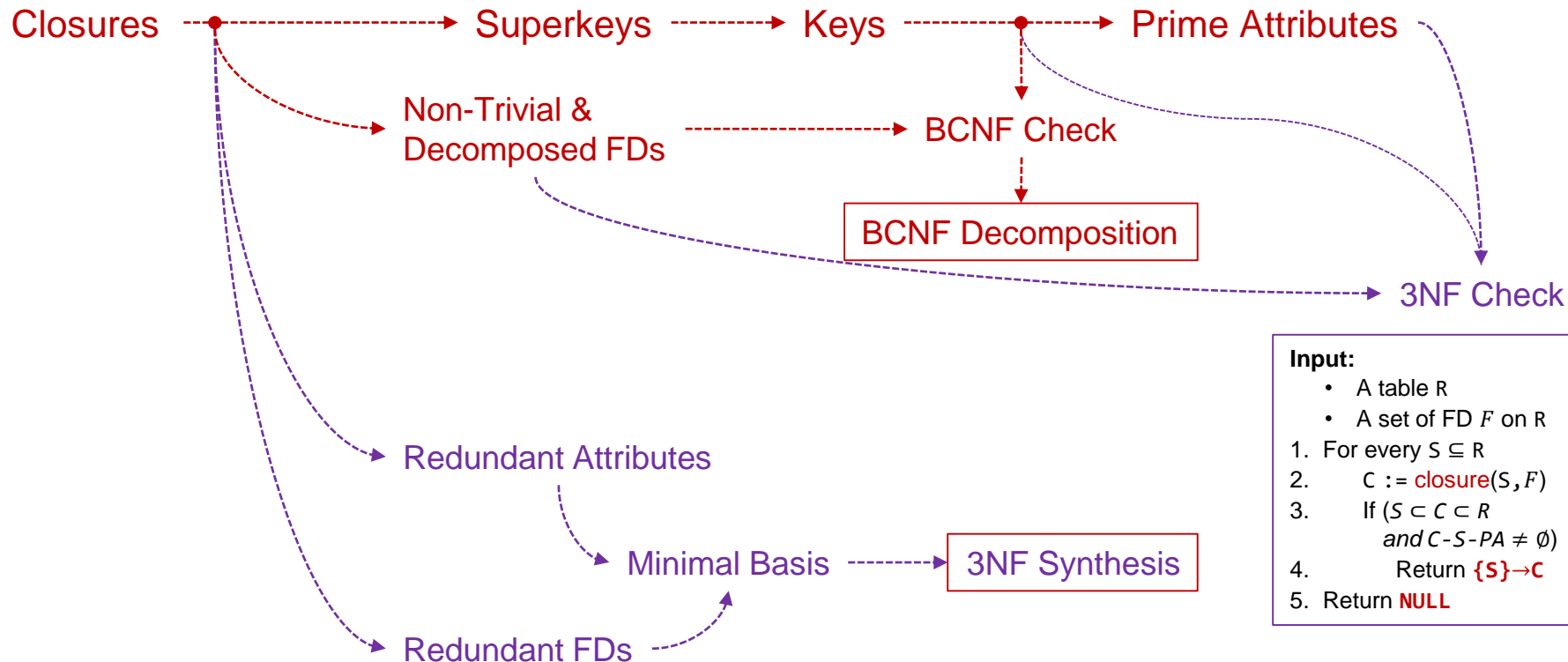
Yes

No

- **Checking 3NF**

- A table R is in **NOT** 3NF if *every non-trivial and decomposed FD both:*
 - *its left hand side is NOT a superkey*
 - *the right hand side is NOT a prime attributes (does not appear in any key)*
- **Exercise:** $R(A, B, C, D, E)$ with $AB \rightarrow C$, $DE \rightarrow C$ and $B \rightarrow E$
 - Keys = {ABD} Prime attributes = {ABD}

Algorithm Roadmap



Input:

- A table R
- A set of FD F on R

1. For every $S \subseteq R$
2. $C := \text{closure}(S, F)$
3. If $(S \subset C \subset R$
and $C - S - PA \neq \emptyset)$
4. Return $\{S\} \rightarrow C$
5. Return **NULL**

3NF Decomposition



3NF Decomposition

- **Normalization**

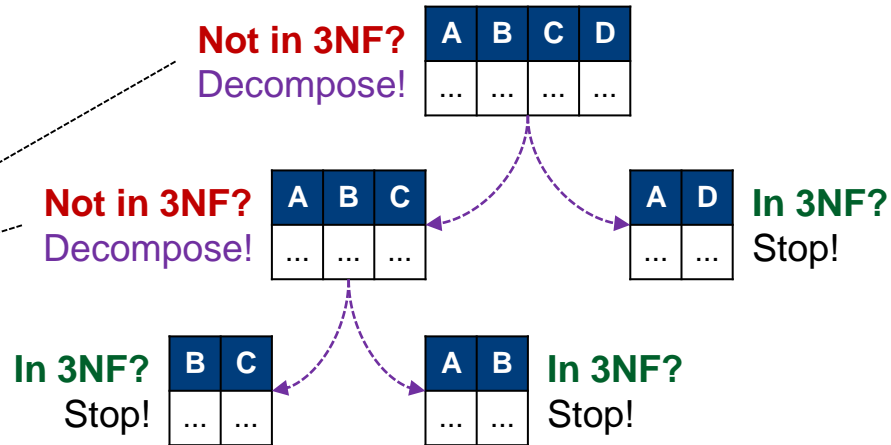
- **Same Idea as BCNF Decomposition:**

- Same potential problem of non dependency preserving decomposition
 - Is there a better idea?

Input:

- A table R
- A set of FD F on R

1. For every $S \subseteq R$
2. $C := \text{closure}(S, F)$
3. If $(S \subset C \subset R$
 and $C-S-PA \neq \emptyset)$
4. Return $\{S\} \rightarrow C$
5. Return **NULL**



Minimal Basis

on our path to 3NF synthesis



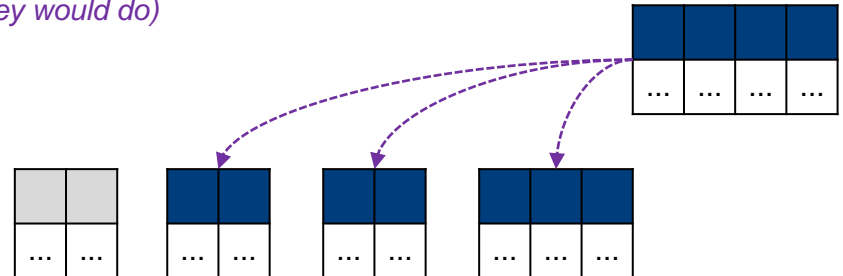
Minimal Basis

- A Sneak Peek

- 3NF Synthesis

- **Input:** Table R and a set of FDs F

1. Derive minimal basis of F
2. From the minimal basis, combine the FDs for which the left hand sides are the same
(union rule, producing what's called as canonical cover)
3. Create a table for each FDs remained in the minimal basis after union
4. If none of the tables contains a key of the original table R, create a table that contains a key of R
(any key would do)



Minimal Basis

- Definition

- Given a set F of FDs, the minimal basis of F is a *simplified* version of F
 - Also called minimal cover *(let's call this F_b)*
- How simplified?
 - Four conditions
 1. Every FD in F_b can be derived from F and vice versa
 2. Every FD in F_b are non-trivial and decomposed FD
 3. For each FD in F_b , none of the attributes on the left hand side is redundant
(no redundant attributes)
 4. No FD in F_b are redundant *(no redundant FD)*
- ❖ *Redundant means that we can remove them (attributes or FDs) without affecting the original FD (i.e., still equivalent)*

Minimal Basis

- **Definition**

- Given a set F of FDs, the minimal basis of F is a *simplified* version of F
 - Also called minimal cover *(let's call this F_b)*
- How simplified?
 - Four conditions
 1. Every FD in F_b can be derived from F and vice versa
 - $F = \{A \rightarrow BD, \quad AB \rightarrow C, \quad C \rightarrow D, \quad BC \rightarrow D\}$
 - ✓ $F1 = \{A \rightarrow B, \quad A \rightarrow C, \quad C \rightarrow D\}$
 - ✗ $F2 = \{A \rightarrow D, \quad A \rightarrow C, \quad C \rightarrow D\}$
 - ✗ $F3 = \{A \rightarrow B, \quad A \rightarrow C, \quad C \rightarrow D, \quad D \rightarrow C\}$

Minimal Basis

- Definition

- Given a set F of FDs, the minimal basis of F is a *simplified* version of F
 - Also called minimal cover *(let's call this F_b)*
 - How simplified?
 - Four conditions
2. Every FD in F_b are non-trivial and decomposed FD
- $F = \{A \rightarrow BD, \quad AB \rightarrow C, \quad C \rightarrow D, \quad BC \rightarrow D\}$
 - ✓ $F1 = \{A \rightarrow B, \quad A \rightarrow C, \quad C \rightarrow D\}$
 - ✗ $F2 = \{A \rightarrow BD, \quad A \rightarrow C, \quad C \rightarrow D\}$
 - ✓ $F3 = \{A \rightarrow B, \quad A \rightarrow C, \quad C \rightarrow D, \quad BC \rightarrow D\}$

Minimal Basis

- Definition

- Given a set F of FDs, the minimal basis of F is a *simplified* version of F
 - Also called minimal cover *(let's call this F_b)*
- How simplified?
 - Four conditions
- 3. For each FD in F_b , none of the attributes on the left hand side is redundant *(no redundant attributes)*
 - $F = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
 - Consider $AB \rightarrow C$, if we remove B from left hand side, we have $A \rightarrow C$
 - We can derive $A \rightarrow C$ from F since $\{A\}^+ = \{ABCD\}$ *(i.e., $A \rightarrow C$ is "hidden" in F)*
 - So we can add $A \rightarrow C$ without adding extraneous constraints
 - But once we add $A \rightarrow C$ then $AB \rightarrow C$ is redundant!
 - Effectively, we found that B is redundant in $AB \rightarrow C$
 - ∴ $AB \rightarrow C$ should not be in minimal basis

Minimal Basis

- **Definition**

- Given a set F of FDs, the minimal basis of F is a *simplified* version of F
 - Also called minimal cover *(let's call this F_b)*
- How simplified?
 - Four conditions
 - 4. No FD in F_b are redundant *(no redundant FD)*
 - $F = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
 - Consider $BC \rightarrow D$
 - We can derive it from $C \rightarrow D$
 - So we can remove it without removing any important information
 - $\therefore BC \rightarrow D$ should not be in minimal basis

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

- $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ $F_b = \{A \rightarrow B, B \rightarrow C\}$
- Is F_b a minimal basis for F ?
 1. $A \rightarrow C$ in F can be derived from F_b
 - F_b is F by removal of $A \rightarrow C$
 2. All FDs in F_b are non-trivial and decomposed
 3. For any FD in F_b , if we remove an attribute from left hand side, then the FD cannot be derived from F *(in fact, they have no left hand side!)*
 4. If any FD in F_b is removed, then some FD in F cannot be derived

$\therefore F_b$ is a minimal basis for F

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

- $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ $F_b = \{A \rightarrow B, AB \rightarrow C\}$
- Is F_b a minimal basis for F ?
 1. $A \rightarrow C$ in F can **NOT** be derived from F_b

$\therefore F_b$ is **NOT** a minimal basis for F

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

- $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ $F_b = \{A \rightarrow BC, A \rightarrow C\}$
- Is F_b a minimal basis for F ?
 1. $B \rightarrow C$ in F can **NOT** be derived from F_b
 2. Also...
 - $A \rightarrow BC$ is **NOT** a decomposed FD

$\therefore F_b$ is **NOT** a minimal basis for F

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow B\}$ $F_b = \{A \rightarrow B, AB \rightarrow C, C \rightarrow B\}$

■ Is F_b a minimal basis for F ?

1. ✓
2. ✓

3. B in $AB \rightarrow C$ can be removed in F_b
- $\{A\}^+ = \{ABC\}$

(redundant attribute)

$\therefore F_b$ is **NOT** a minimal basis for F

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

■ Is F_b a minimal basis for F ?

1. ✓
2. ✓
3. ✓
4. $A \rightarrow C$ can be removed!

$F_b = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

(redundant FD)

$\therefore F_b$ is **NOT** a minimal basis for F

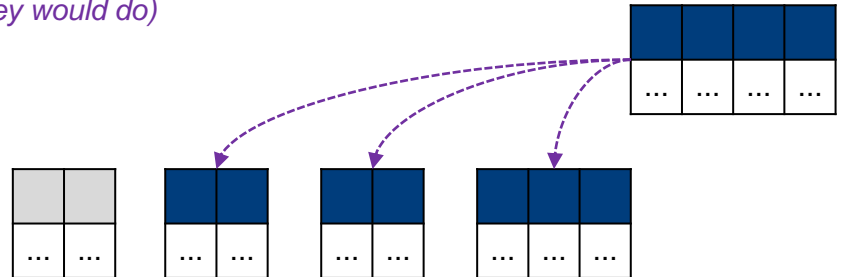
Minimal Basis

- A Reminder on Why We Need Minimal Basis

- 3NF Synthesis

- **Input:** Table R and a set of FDs F

1. Derive *minimal basis* of F
2. From the *minimal basis*, combine the FDs for which the left hand sides are the same *(union rule, producing what's called as canonical cover)*
3. Create a table for each FDs remained in the *minimal basis* after union
4. If none of the tables contains a key of the original table R, create a table that contains a key of R *(any key would do)*



Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Algorithm

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
 3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
 4. Remove redundant FDs *(maintain $F_b \equiv F$)*
- ❖ *Clearly, at the end, we have satisfied all conditions* *(as long as we maintain $F_b \equiv F$)*
- But how do we perform step 3?

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

● Algorithm: Remove Redundant Attribute

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

- Consider an FD $\{A\} \rightarrow B$ in F for any A and B *(in here, $\{A\}$ is a set of attributes)*
 1. Consider an attribute C in $\{A\}$
 - Compute $\{A-C\}^+$ using F *(in here, $\{A-C\}$ means we remove C from $\{A\}$)*
 - If $B \in \{A-C\}^+$, then we can remove C *(because $\{A-C\} \rightarrow B$)*
 - Repeat step 1 but with $\{A\} \rightarrow B$ changed with $\{A-C\} \rightarrow B$
 2. Do this for all FDs

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

● Algorithm: Remove Redundant Attribute

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

■ Example:

- $F = \{A \rightarrow B, A \rightarrow D, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
 - Consider $AB \rightarrow C$
 - $\{A\}^+ = \{A, B, C, D\}$
 - B can be removed from $AB \rightarrow C$
 - Repeat with
 - $F = \{A \rightarrow B, A \rightarrow D, A \rightarrow C, C \rightarrow D, BC \rightarrow D\}$

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Consider an attribute C in $\{A\}$
 - Compute $\{A-C\}^+$ using F
 - If B in $\{A-C\}^+$, replace $\{A\} \rightarrow B$ with $\{A-C\} \rightarrow B$ then repeat
 2. Do this for all FDs

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

Algorithm: Remove Redundant Attribute

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

Example:

- $F = \{A \rightarrow B, A \rightarrow D, A \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
 - Consider $BC \rightarrow D$
 - $\{B\}^+ = \{B\}$
 - C can **NOT** be removed from $BC \rightarrow D$
 - $\{C\}^+ = \{CD\}$
 - B can be removed from $BC \rightarrow D$
 - Repeat with
 - $F = \{A \rightarrow B, A \rightarrow D, A \rightarrow C, C \rightarrow D, C \rightarrow D\}$

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Consider an attribute C in $\{A\}$
 - Compute $\{A-C\}^+$ using F
 - If B in $\{A-C\}^+$, replace $\{A\} \rightarrow B$ with $\{A-C\} \rightarrow B$ then repeat
 2. Do this for all FDs

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

● Algorithm: Remove Redundant Attribute

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

■ Example:

- $F = \{A \rightarrow B, A \rightarrow D, A \rightarrow C, C \rightarrow D, C \rightarrow D\}$
 - Nothing else to remove

■ Question:

- Is this always gives us a unique solution?

Yes

No

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Consider an attribute C in $\{A\}$
 - Compute $\{A-C\}^+$ using F
 - If B in $\{A-C\}^+$, replace $\{A\} \rightarrow B$ with $\{A-C\} \rightarrow B$ then repeat
 2. Do this for all FDs

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

● Algorithm: Remove Redundant Attribute

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

■ Example:

- $F = \{A \rightarrow B, B \rightarrow A, AB \rightarrow C\}$
 - Consider $AB \rightarrow C$
 - $\{A\}^+ = \{A, B, C\}$
 - B can be removed from $AB \rightarrow C$
 - We could have started with
 - $\{B\}^+ = \{A, B, C\}$
 - A can be removed from $AB \rightarrow C$
 - BUT, we cannot remove both!
 - So two possible solutions

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Consider an attribute C in $\{A\}$
 - Compute $\{A-C\}^+$ using F
 - If B in $\{A-C\}^+$, replace $\{A\} \rightarrow B$ with $\{A-C\} \rightarrow B$ then repeat
 2. Do this for all FDs

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Algorithm

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
 3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
 4. Remove redundant FDs *(maintain $F_b \equiv F$)*
- ❖ *Clearly, at the end, we have satisfied all conditions* *(as long as we maintain $F_b \equiv F$)*
- But how do we perform **step 4**?

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

● Algorithm: Remove Redundant FD

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

- Consider an FD $\{A\} \rightarrow B$ in F for any A and B *(in here, $\{A\}$ is a set of attributes)*

■ Observation:

- If we can remove $\{A\} \rightarrow B$, that means $F - \{\{A\} \rightarrow B\} \equiv F$
- The only difference is the removal of $\{A\} \rightarrow B$
 - Clearly, all FD in $F - \{\{A\} \rightarrow B\}$ can be derived from F
 - So what we need to show is only that $\{A\} \rightarrow B$ can be derived from $F - \{\{A\} \rightarrow B\}$
- Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - Then check if B is in $\{A\}^+$

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

● Algorithm: Remove Redundant FD

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

- Consider an FD $\{A\} \rightarrow B$ in F for any A and B *(in here, $\{A\}$ is a set of attributes)*
 1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - If $B \in \{A\}^+$, then we can remove $\{A\} \rightarrow B$ *(because $\{A\} \rightarrow B$ even in its absence)*
 - Repeat with next FD but with $\{A\} \rightarrow B$ removed
 2. Do this for all FDs

Minimal Basis

• Algorithm: Remove Redundant FD

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

■ Example:

- $F = \{A \rightarrow B, A \rightarrow D, A \rightarrow C, C \rightarrow D\}$
 - Consider $A \rightarrow B$
 - $\{A\}^+$ w.r.t $\{A \rightarrow D, A \rightarrow C, C \rightarrow D\}$
 $= \{A, C, D\}$
 - $A \rightarrow B$ can **NOT** be remove

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - If B in $\{A\}^+$, remove $\{A\} \rightarrow B$ then repeat with next FD
 2. Do this for all FDs

Minimal Basis

● Algorithm: Remove Redundant FD

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

■ Example:

- $F = \{A \rightarrow B, A \rightarrow D, A \rightarrow C, C \rightarrow D\}$
 - Consider $A \rightarrow D$
 - $\{A\}^+$ w.r.t $\{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
 $= \{A, B, C, D\}$
 - $A \rightarrow D$ can be removed
 - Repeat with
 - $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - If B in $\{A\}^+$, remove $\{A\} \rightarrow B$
then repeat with next FD
 2. Do this for all FDs

Minimal Basis

- **Algorithm: Remove Redundant FD**

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

- **Example:**

- $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
 - Consider $A \rightarrow C$
 - $\{A\}^+$ w.r.t $\{A \rightarrow B, C \rightarrow D\}$
 $= \{A, B\}$
 - $A \rightarrow C$ can **NOT** be removed

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - If B in $\{A\}^+$, remove $\{A\} \rightarrow B$
then repeat with next FD
 2. Do this for all FDs

Minimal Basis

- **Algorithm: Remove Redundant FD**

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

- **Example:**

- $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
 - Consider $C \rightarrow D$
 - $\{C\}^+$ w.r.t $\{A \rightarrow B, A \rightarrow C\}$
 $= \{C\}$
 - $C \rightarrow D$ can **NOT** be removed

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - If B in $\{A\}^+$, remove $\{A\} \rightarrow B$ then repeat with next FD
 2. Do this for all FDs

Minimal Basis

- **Algorithm: Remove Redundant FD**

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

- **Example:**

- $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
 - No other FDs to consider

- **Question:**

- Is this always gives us a unique solution?

Yes

No

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - If B in $\{A\}^+$, remove $\{A\} \rightarrow B$ then repeat with next FD
 2. Do this for all FDs

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

● Algorithm: Remove Redundant FD

2. Transform the FDs so that it is non-trivial and decomposed *(maintain $F_b \equiv F$)*
3. Remove redundant attributes on the left hand sides of each FDs *(maintain $F_b \equiv F$)*
4. Remove redundant FDs *(maintain $F_b \equiv F$)*

■ Example:

- $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$
 - Can remove either one but not both
 - $A \rightarrow B$
 - $\{A\}^+$ w.r.t. $\{A \rightarrow C, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$
 $= \{A, B, C\}$
 - $A \rightarrow C$
 - $\{A\}^+$ w.r.t. $\{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$
 $= \{A, B, C\}$

Algorithm

- For $\{A\} \rightarrow B$ in F
 1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$
 - If B in $\{A\}^+$, remove $\{A\} \rightarrow B$ then repeat with next FD
 2. Do this for all FDs

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Algorithm

2. Transform the FDs so that it is non-trivial and decomposed

(maintain $F_b \equiv F$)

3. Remove redundant attributes on the left hand sides of each FDs

(maintain $F_b \equiv F$)

▪ For $\{A\} \rightarrow B$ in F

1. Consider an attribute C in $\{A\}$

- Compute $\{A-C\}^+$ using F

- If B in $\{A-C\}^+$, replace $\{A\} \rightarrow B$ with $\{A-C\} \rightarrow B$ then repeat

2. Do this for all FDs

4. Remove redundant FDs

(maintain $F_b \equiv F$)

▪ For $\{A\} \rightarrow B$ in F

1. Compute $\{A\}^+$ using $F - \{\{A\} \rightarrow B\}$

- If B in $\{A\}^+$, remove $\{A\} \rightarrow B$ then repeat with next FD

2. Do this for all FDs

❖ *Clearly, at the end, we have satisfied all conditions*

(as long as we maintain $F_b \equiv F$)

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow DE, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

2. Transform the FDs so that it is non-trivial and decomposed

(maintain $F_b \equiv F$)

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

3. Remove redundant attributes on the left hand sides of each FDs

(maintain $F_b \equiv F$)

■ Two candidates

1. $BC \rightarrow D$

2. $BC \rightarrow E$

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

3. Remove redundant attributes on the left hand sides of each FDs

(maintain $F_b \equiv F$)

■ Consider $BC \rightarrow D$

- $\{B\}^+ = \{B\}$
 - So C can **NOT** be removed
- $\{C\}^+ = \{C\}$
 - So B can **NOT** be removed

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

3. Remove redundant attributes on the left hand sides of each FDs

(maintain $F_b \equiv F$)

■ Consider $BC \rightarrow E$

- $\{B\}^+ = \{B\}$
 - So C can **NOT** be removed
- $\{C\}^+ = \{C\}$
 - So B can **NOT** be removed

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

2. Transform the FDs so that it is non-trivial and decomposed

(maintain $F_b \equiv F$)

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

3. Remove redundant attributes on the left hand sides of each FDs

(maintain $F_b \equiv F$)

■ No redundant attributes

4. Remove redundant FDs

(maintain $F_b \equiv F$)

■ Need to check everything

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

4. Remove redundant FDs

(maintain $F_b \equiv F$)

■ Consider $BC \rightarrow D$

- $\{BC\}^+$ w.r.t. $\{BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$
= $\{B, C\}$
 - So $BC \rightarrow D$ is **NOT** redundant

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

4. Remove redundant FDs

(maintain $F_b \equiv F$)

■ Consider $BC \rightarrow E$

- $\{BC\}^+$ w.r.t. $\{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

= $\{A, B, C, D, E\}$

○ So $BC \rightarrow E$ is **redundant**

❖ Continue with $F = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

4. Remove redundant FDs

■ Consider $A \rightarrow E$

- $\{A\}^+$ w.r.t. $\{BC \rightarrow D, D \rightarrow A, E \rightarrow B\}$
= $\{A\}$
 - So $A \rightarrow E$ is **NOT** redundant

(maintain $F_b \equiv F$)

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

4. Remove redundant FDs

■ Consider $D \rightarrow A$

- $\{D\}^+$ w.r.t. $\{BC \rightarrow D, A \rightarrow E, E \rightarrow B\}$
= $\{D\}$
 - So $D \rightarrow A$ is **NOT** redundant

(maintain $F_b \equiv F$)

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

4. Remove redundant FDs

■ Consider $E \rightarrow B$

- $\{E\}^+$ w.r.t. $\{BC \rightarrow D, A \rightarrow E, D \rightarrow A\}$
= $\{E\}$
 - So $E \rightarrow B$ is **NOT** redundant

(maintain $F_b \equiv F$)

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Example

■ $F = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

2. Transform the FDs so that it is non-trivial and decomposed

(maintain $F_b \equiv F$)

■ $F = \{BC \rightarrow D, BC \rightarrow E, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

3. Remove redundant attributes on the left hand sides of each FDs

(maintain $F_b \equiv F$)

■ No redundant attributes

4. Remove redundant FDs

(maintain $F_b \equiv F$)

■ $BC \rightarrow E$ is redundant

❖ $F_b = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

Minimal Basis

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes
4. No redundant FD

• Exercise

- $F = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$
- 2. Transform the FDs so that it is non-trivial and decomposed (*maintain $F_b \equiv F$*)
 - $F = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$
- 3. Remove redundant attributes on the left hand sides of each FDs (*maintain $F_b \equiv F$*)
 - Two candidates
 1. $AC \rightarrow D$ (*C is redundant*)
 2. $AD \rightarrow B$ (*D is redundant*)
 - $F = \{A \rightarrow C, A \rightarrow D, A \rightarrow B\}$
- 4. Remove redundant FDs (*maintain $F_b \equiv F$*)
 - No redundant FD
 - $F_b = \{A \rightarrow C, A \rightarrow D, A \rightarrow B\}$

3NF Synthesis

our final product



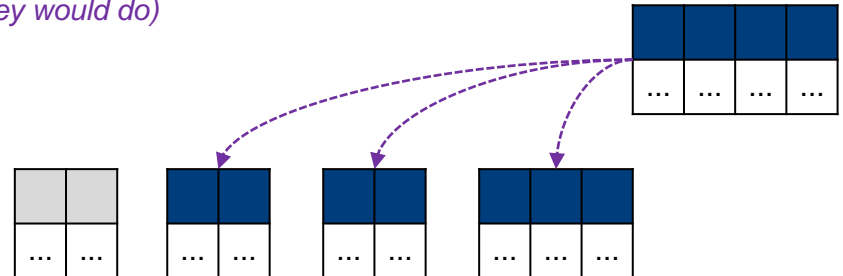
3NF Synthesis

- Algorithm

- 3NF Synthesis

- **Input:** Table R and a set of FDs F

1. Derive minimal basis of F
2. From the minimal basis, combine the FDs for which the left hand sides are the same *(union rule, producing what's called as canonical cover)*
3. **Create a table for each FDs remained in the minimal basis after union**
4. If none of the tables contains a key of the original table R, create a table that contains a key of R *(any key would do)*



3NF Synthesis

3NF Synthesis

1. Derive *minimal basis* of F
2. Produce *canonical cover*
3. Create a table for each FD
4. Add the *key* if missing

• Algorithm

■ **Example:** $R(A, B, C, D, E)$ with $BC \rightarrow DE, A \rightarrow E, D \rightarrow A$ and $E \rightarrow B$

■ $F = \{BC \rightarrow DE, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

1. Derive minimal basis of F

- $F_b = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

2. Produce canonical cover

- $F_c = \{BC \rightarrow D, A \rightarrow E, D \rightarrow A, E \rightarrow B\}$

3. Create a table for each FD

- $R_1(B, C, D), R_2(A, E), R_3(A, D)$ and $R_4(B, E)$

4. Add the key if missing

- Keys = $\{AC\}, \{BC\}, \{CD\}, \{CE\}$

- R1 contains a key

Why add Key?

Ensure *lossless join* decomposition

3NF Synthesis

3NF Synthesis

1. Derive *minimal basis* of F
2. Produce *canonical cover*
3. Create a table for each FD
4. Add the *key* if missing

• Lossless Join Decomposition

■ **Example:** $R(A, B, C, D)$ with $A \rightarrow B$ and $C \rightarrow D$

■ $F = \{A \rightarrow B, C \rightarrow D\}$

1. Derive minimal basis of F

- $F_b = \{A \rightarrow B, C \rightarrow D\}$

2. Produce canonical cover

- $F_c = \{A \rightarrow B, C \rightarrow D\}$

3. Create a table for each FD

- $R_1(A, B)$ and $R_2(C, D)$

4. Add the key if missing

- Keys = $\{AC\}$

- Add $R_3(A, C)$

Why add Key?

$R_1(A, B)$ and $R_2(C, D)$ cannot be used to reconstruct $R(A, B, C, D)$

3NF Synthesis

3NF Synthesis

1. Derive *minimal basis* of F
2. Produce *canonical cover*
3. Create a table for each FD
4. Add the *key* if missing

• Algorithm

- **Exercise:** $R(A, B, C, D, E)$ with $A \rightarrow B, A \rightarrow C, B \rightarrow C, E \rightarrow C$ and $E \rightarrow D$
 - $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, E \rightarrow C, E \rightarrow D\}$
 1. Derive minimal basis of F
 - $F_b = \{A \rightarrow B, B \rightarrow C, E \rightarrow C, E \rightarrow D\}$
 2. Produce canonical cover
 - $F_c = \{A \rightarrow B, B \rightarrow C, E \rightarrow CD\}$
 3. Create a table for each FD
 - $R1(A, B), R2(B, C)$ and $R3(C, D, E)$
 4. Add the key if missing
 - Keys = $\{AE\}$
 - Add $R4(A, E)$

3NF Synthesis

3NF Synthesis

1. Derive *minimal basis* of F
2. Produce *canonical cover*
3. Create a table for each FD
4. Add the *key* if missing

• Algorithm

- **Exercise:** $R(A, B, C, D, E)$ with $A \rightarrow B, AB \rightarrow C, C \rightarrow DE, E \rightarrow C$ and $E \rightarrow D$
 - $F = \{A \rightarrow B, AB \rightarrow C, C \rightarrow DE, E \rightarrow C, E \rightarrow D\}$
 1. Derive minimal basis of F
 - $F_b = \{A \rightarrow B, A \rightarrow C, C \rightarrow D, C \rightarrow E, E \rightarrow C\}$
 2. Produce canonical cover
 - $F_c = \{A \rightarrow BC, C \rightarrow DE, E \rightarrow C\}$
 3. Create a table for each FD
 - $R1(A, B, C), R2(C, D, E)$ and $R3(C, E)$
 4. Add the key if missing
 - Keys = $\{A\}$
 - R1 contains a key

Final Thought

closing statement

Summary

- Poorly designed tables give rise to redundancy, update anomalies and deletion anomalies
- **BCNF** eliminates these problems
 - BCNF: for any non-trivial and decomposed FD on a table R, its left hand side is a superkey for R
 - But BCNF does not always preserve all FDs *(non dependency preserving)*
 - We may need to perform a join of multiple tables to check whether an FD holds
- **3NF** is slightly weaker than BCNF *(has more redundancies, has update and deletion anomalies in some rare cases)* but preserves all FDs
 - 3NF: for any non-trivial and decomposed FD on a table R, either its left hand side is a superkey for R, *or its right hand side is a prime attribute*

BCNF or 3NF or *Lower*?

- BCNF is only *inferior* to 3NF in the sense that sometimes it does not preserve all FDs
- **Idea:**
 - Go for BCNF if we can find a BCNF decomposition that preserves all FDs
 - If such decomposition cannot be found, then
 - Still go for BCNF if preserving all FDs is not important
 - Or go for 3NF otherwise
 - If we are lucky, 3NF synthesis may actually find a BCNF decomposition!
- Should we go **lower** than 3NF?
 - Notice how there can be many tables produced by 3NF
 - What will happen to queries?
 - We may have to perform lots of joins
 - This can slow down queries by a lot since joins are expensive
 - Even 3NF may not be suitable in that case

QUESTION?